H-Infinity Control Design Via Convex Optimization: Toward A Comprehensive Design Environment

by

Karan Puttannaiah

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved November 2013 by the
Graduate Supervisory Committee:

Armando A. Rodriguez, Chair
Konstantinos S. Tsakalis
Jennie Si

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

The problem of systematically designing a control system continues to remain a subject of intense research. In this thesis, a very powerful control system design environment for Linear Time-Invariant (LTI) Multiple-Input Multiple-Output (MIMO) plants is presented. The environment has been designed to address a broad set of closed loop metrics and constraints; e.g. weighted $\mathcal{H}^{\infty}$ closed loop performance subject to closed loop frequency and/or time domain constraints (e.g. peak frequency response, peak overshoot, peak controls, etc.). The general problem considered – a generalized weighted mixed-sensitivity problem subject to constraints – permits designers to directly address and tradeoff multivariable properties at distinct loop breaking points; e.g. at plant outputs and at plant inputs. As such, the environment is particularly powerful for (poorly conditioned) multivariable plants. The Youla parameterization is used to parameterize the set of all stabilizing LTI proper controllers. This is used to convexify the general problem being addressed. Several bases are used to turn the resulting infinite-dimensional problem into a finite-dimensional problem for which there exist many efficient convex optimization algorithms. A simple cutting plane algorithm is used within the environment. Academic and physical examples are presented to illustrate the utility of the environment.

To my parents

ACKNOWLEDGMENTS

I am very grateful for the cooperation and interest of my M.S advisor, Professor Armando A. Rodriguez who showed a great deal of confidence in my abilities. Dr. Rodriguez has given me a great amount of technical knowledge, and has guided and supported me throughout my research.

I would like to thank the members of my thesis committee: Dr. Konstantinos Tsakalis, and Dr. Jennie Si for their support in my research and their help in my courses.

I would like to thank all the faculty members, especially Dr. Mittelmann who have helped me acquire knowledge required for this research.

I would also like to acknowledge the help and guidance of Srikanth Sridharan.

Finally I would like to thank my family and friends for their consistent support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION AND OVERVIEW

1.1 Motivation

This dissertation presents a powerful control system design environment for linear time invariant (LTI) multiple-input multiple-output (MIMO) plants. A generalized weighted mixed-sensitivity problem subject to constraints is formulated and solved using the design environment. This user-friendly Graphical User Interface (GUI) tool permits designers to directly address and tradeoff closed loop properties at distinct loop breaking points.

In multivariable systems, feedback properties must be analysed at different loop breaking points [8; 10; 11]. Loop shaping at one loop breaking point might not result in good properties at a different loop breaking point [8]. This is true especially for ill-conditioned plants [9; 22]. Relating closed loop functions with controller transfer function matrix is not straightforward. This makes the control problem difficult [17]. Hence a design tool that addresses loop shaping at distinct loop breaking points could give the designer freedom to trade-off closed loop properties depending on problem objectives.

Our work is motivated by the following control design objectives:

1. A design tool that can handle broad class of SISO and MIMO plants helps in systematically designing controllers with desired control objectives.

2. Closed loop properties at distinct loop breaking points, e.g., plant output and input need to be shaped in order to achieve an acceptable trade-off.

3. Closed loop metrics, specifications and constraints make the design problem difficult. A design tool that can handle broad class of control objectives need to be developed.

1.2 Control Methodology

$\mathcal{H}^\infty$ control problems address have been used extensively as a frequency domain loopshaping technique [27; 28; 15; 14]. Minimizing the $\mathcal{H}^\infty$ norm of the weighted closed loop transfer functions minimizes the peak of largest singular value of the system. In this work, in order to address loopshaping at distinct loop-breaking points, we reformulate the problem as a generalized mixed sensitivity minimization. This is discussed in Chapter 2.

1.3 Approach Taken

The generalized weighted mixed-sensitivity problem subject to constraints that is formulated is a nonlinear problem in controller (K). Youla Q-parameterization is used to convexify the problem. This parameterizes the set of all possible stable LTI controllers [25; 7; 13; 24; 26] . Several bases are used to turn the resulting infinite-dimensional problem into a finite-dimensional problem. A broad class of control system design specifications may be posed as convex constraints on the closed loop transfer function matix [2]. Convex optimizations algorithm is employed to solve the problem efficiently [12; 6; 3]. The design environment uses these concepts to find the solution to our generalized weighted mixed-sensitivity problem.

1.4 Overview of Thesis

In Chapter 2 we formulate the generalized mixed-sensitivity problem. In Chapter 3, we formulate the problem into a convex optimization in parameter-Q. Chapter 6 shows the utility of our design environment. In Chapter **??**, we illustrate the design using examples and applications.

Chapter 2

GENERALIZED $\mathcal{H}^\infty$ MIXED SENSITIVITY OPTIMIZATION PROBLEM

2.1 Introduction

In this chapter, the formulation of generalized $\mathcal{H}^\infty$ mixed-sensitivity minimization problem subject to convex constraints is discussed.

Consider the feedback system in Figure 2.1.

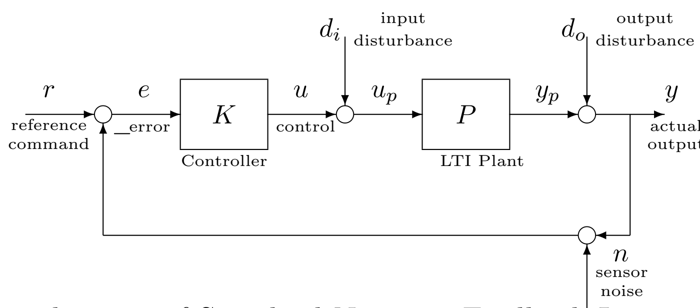


Figure 2.1: Visualization of Standard Negative Feedback Loop

We assume that P and K are MIMO LTI systems (i.e. transfer function matrices).

Closed Loop Transfer Function Matrices. The closed loop transfer function matrices with the loop broken at plant output and input are given by [10],[19]:

- *Sensitivity at plant output*

$$S_o \stackrel{\text{def}}{=} [I + PK]^{-1} \tag{2.1}$$

- *Reference to control transfer function*

$$T_{ru} \stackrel{\text{def}}{=} KS = K[I + PK]^{-1}. \tag{2.2}$$

- *Complementary sensitivity at plant output*

$$T_o \stackrel{\text{def}}{=} I - S_o = PK[I + PK]^{-1}. \tag{2.3}$$

- *Sensitivity at plant input*

$$S_i \stackrel{\text{def}}{=} [I + KP]^{-1}. \tag{2.4}$$

- *Input disturbance to output transfer function*

$$PS_i \stackrel{\text{def}}{=} P[I + KP]^{-1}. \tag{2.5}$$

- *Complementary sensitivity at plant output*

$$T_i \stackrel{\text{def}}{=} [I + KP]^{-1}KP. \tag{2.6}$$

Closed Loop Design Objectives. General closed loop objectives associated with feedback design may be stated as follows:

- the closed loop system should be stable

- $\sigma_{max}\left[S_o(j\omega)\right]$ and $\sigma_{max}\left[S_i(j\omega)\right]$ should be small at low fequencies for good low frequency command following and disturbance attenuation

- $\sigma_{max}\left[K(j\omega)S_o(j\omega)\right]$ should not be too large to prevent the controls from getting too large for anticipated exogneous signals

- $\sigma_{max}\left[P(j\omega)S_i(j\omega)\right]$ should be small at high frequencies for good high frequency input disturbance attenuation

- $\sigma_{max}\left[P(j\omega)S_i(j\omega)\right]$ should be small at low frequencies for good low frequency input disturbance attenuation

- $\sigma_{max}\left[T_o(j\omega)\right]$ and $\sigma_{max}\left[T_i(j\omega)\right]$ should be small at high frequencies for good high frequency noise attenuation

- $\sigma_{max}\left[T_o(j\omega)\right]$ and $\sigma_{max}\left[T_i(j\omega)\right]$ 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, $\sigma_{max}\left[M\right]$ denotes the maximum singular value of M.

2.2 Standard $\mathcal{H}^\infty$ Mixed-Sensitivity Minimization Problem: Pros and Cons

The Standard Weighted $\mathcal{H}^\infty$ mixed sensitivity optimization problem that addresses closed loop maps at plant output is as follows [27; 28; 15; 21; 5; 1]:

$$K = \arg\{ \min_{K\ stabilizing} \gamma \;\; | \;\; \left\| \begin{bmatrix} W_1 S_o \\ W_2 K S_o \\ W_3 T_o \end{bmatrix} \right\|_{\mathcal{H}^\infty} < \gamma \;\; \} \tag{2.7}$$

where W_1, W_2, W_3 are frequency-dependent weighting matrices that are used to trade-off the properties of S_o, KS_o, and T_o.

One of the main drawback of having only the transfer function matrices from reference r to output y is that, it might result in bad feedback properties at the plant input. In other words, good feedback properties at plant output does not guarantee good properties at plant input [11].

$$\frac{1}{\kappa[P]}\sigma_j\left[S_o\right] \leq \sigma_j\left[S_i\right] \leq \kappa[P]\sigma_j\left[S_o\right] \tag{2.8}$$

where κ represents the condition number of the plant. If the condition number of the plant is high, achieving good feedback properties at plant output might result in bad properties at plant input.

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

In this work, to address our design problem, we consider the following weighted $\mathcal{H}^\infty$ mixed sensitivity problem:

$$K = \arg\{ \min_{K\ stabilizing} \gamma \mid max \left(\left\| \begin{bmatrix} W_1 S_o \\ W_2 K S_o \\ W_3 T_o \end{bmatrix} \right\|_{\mathcal{H}^\infty}, \rho \left\| \begin{bmatrix} W_4 S_i \\ W_5 P S_i \\ W_6 T_i \end{bmatrix} \right\|_{\mathcal{H}^\infty} \right) < \gamma \ \} \quad (2.9)$$

where $W_1, W_2, W_3, W_4, W_5, W_6$ are frequency-dependent weighting matrices that are used to trade-off the properties of S_o, KS_o, T_o, S_i, PS_i and T_i, and ρ is a scalar used to trade-off properties at the two loop breaking points.

Solution Method: The approach taken in this work is as described below.

- Achieving Convexity Via Youla Parameterization. The approach relies on using the Youla Q-Parameterization [25] to transform the transfer matrices (S_o, KS_o, T_o, S_i, PS_i and T_i) that depend nonlinearly on K into transfer matrices that depend affinely on the stable Youla parameter on Q (stable transfer matrix). This results in a transfer function matrices that are convex in Q. Since the $\mathcal{H}^\infty$ norm is also a convex functional [2], the Youla parameterization results in a convex problem in Q.

- Obtaining a Finite-Dimensional Convex Problem. Because Q can be an arbitrary stable transfer function matrix, the resulting problem is infinite-dimensional. Fortunately, any real-rational Q may be approximated by a finite linear combination of a real-rational stable transfer function matrices. This permits us to transform the infinite-dimensional convex problem in Q to a finite-dimensional convex optimization problem in the coefficients defining the above linear combination.

It should be noted that many control system performance specifications may be posed as convex constraints [2], namely overshoot and peak magnitude frequency response.

2.4 Accomodating Convex Constraints

The plant P and the weighting functions should be viewed as forming an *generalized plant* G as shown in Figure 2.2. $w \in \mathcal{R}^w$ represents exogenous signals (e.g. reference commands), $u \in \mathcal{R}^u$ represents controls, $e \in \mathcal{R}^e$ represents measurements, and $z \in \mathcal{R}^z$ represents regulated variables.

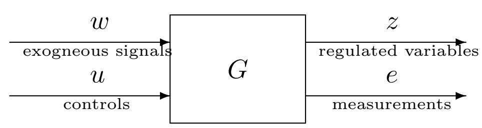


Figure 2.2: Visualization of Augmented Plant G

General Control System Design Problem. Given the above, the new optimization problem to be solved is that of finding a stablizing finite-dimensional LTI controller K that minimizes the $\mathcal{H}^\infty$ norm of the transfer function matrix from while satisfying all the constraints.

This optimization problem may be posed as follows:

$$K = \arg\{ \min_{K\ stabilizing} \gamma \quad | \quad \left(\left\| \begin{bmatrix} W_1 S_o \\ W_2 K S_o \\ W_3 T_o \end{bmatrix} \right\|_{\mathcal{H}^\infty}, \left\| \begin{bmatrix} \rho W_4 S_i \\ \rho W_5 P S_i \\ \rho W_6 T_i \end{bmatrix} \right\|_{\mathcal{H}^\infty} \right) < \gamma \quad (2.10)$$

$$C_i \begin{pmatrix} W^i_{1c} S_o \\ W^i_{2c} K S_o \\ W^i_{3c} T_o \\ W^i_{4c} S_i \\ W^i_{5c} P S_i \\ W^i_{6c} S_i \end{pmatrix} \leq c_i \quad i = 1, 2, \ldots \ \} \ (2.11)$$

where $C_k(\cdot)$ denotes the k^{th} constraint functional and $c_k \in \mathcal{R}$.

It should be noted that the augmented plant G contains all subsystems essential to carry out the optimization. After the optimization process is carried out, the resulting controller K can then be inserted into the unity feedback system shown in Figure 2.1.

2.5 Summary and Conclusions

Observations about General Control System Design Problem. Given the above formulation, it is important to note the following:

- the above optimization problem for K is nonlinear and infinite-dimensional
- no closed form solution or direct approach exists for the above problem.

The next chapter discusses about posing the control system design problem as a convex optimization problem. In this chapter, we formulated a problem to shape the feedback properties at different loop-breaking points.

Chapter 3

CONVEXIFICATION OF THE PROBLEM

3.1 Introduction

The original nonlinear infinite-dimenisonal optimization problem may be transformed to a finite-dimensional convex optimization problem for which efficient algorithms exist. This is done in several steps.

1. *Achieving Convexity.* First, the the Youla Q-Parameterization [25] is used to parameterize the set of all stabilizing controllers for an LTI plant. It is shown how this parameterization leads to an affine closed loop transfer function matrix $T_{wz}(Q)$ in the parameter Q. This transforms our problem to a convex optimization problem - albeit infinite-dimensional.

2. *Achieving Finite-Dimensionality.* Next, approximation ideas are used to approximate Q and transform the inifinite-dimensional problem to a finite- dimensional problem for which efficient algorithms exist.

3.2 Youla Q-Parameterization of All Stabilizing Controllers

This section describes the Youla Q-Parameterization - a parameterization for the set of all LTI compensators that stabilize an LTI plant.

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 follows:

$$S(P) = K(Q)|Q \in \mathcal{H}^\infty \tag{3.1}$$

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 $Q \in \mathcal{H}^\infty$.

Observer Based Youla Q-Parameterization The parameterization $K(Q)$ may be constructed in terms of a model based compensator $K_{mbc} = [A - BF - L(C - DF), -L, -F]$ that stabilizes P and a stable transfer function matrix Q ($Q \in \mathcal{H}^\infty$) as in Figure 3.1.

$$\dot{x_k} = (A - BF - L(C - DF))x_k - Le + (B - LD)\hat{v} \tag{3.2}$$

$$u = -Fx_k + \hat{v} \tag{3.3}$$

$$\hat{v} = Qv \tag{3.4}$$

$$v = -(C - DG)x_k - e - D\hat{v} \tag{3.5}$$

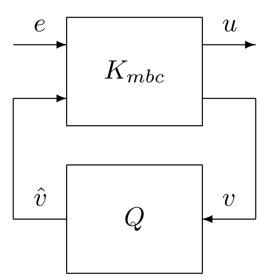


Figure 3.1: Visualization Q Connected to an Observer-Based Controller J

where $Q \in \mathcal{H}^\infty$ is any stable transfer function matrix. $K(Q)$ may be re-written as follows:

$$K_{mbc} = \left[\begin{array}{c|cc} A + BF + LC + LDF & -L & B + LD \\ \hline F & 0 & I \\ -(C + DF) & I & -D \end{array}\right] \tag{3.6}$$

$$\hat{v} = Qv \tag{3.7}$$

The observer based structure of $K(Q)$ is shown in Figure 3.2

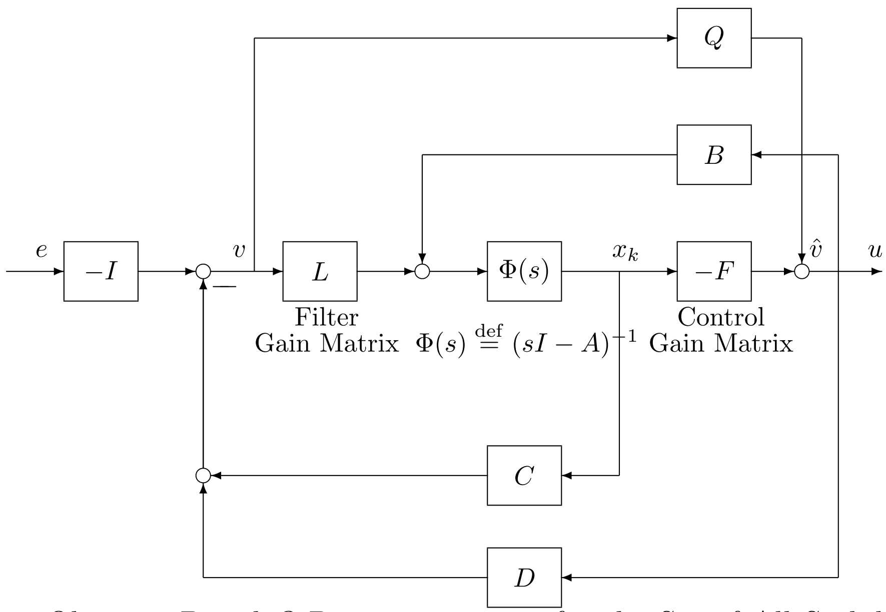


Figure 3.2: Observer Based Q-Parameterization for the Set of All Stabilizing LTI Controllers $K(Q)$

Controller State Space Representation. If x_Q denotes the state of

$$Q \stackrel{\text{def}}{=} \left[\begin{array}{c|c} A_Q & B_Q \\ \hline C_Q & D_Q \end{array}\right] \tag{3.8}$$

This yields the following state space representation for the controller $K(Q)$:

$$K(Q) = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right] = \left[\begin{array}{cc|c} A - BF - LC - BD_QC & BC_Q & BD_Q - L \\ -B_QC & A_Q & B_Q \\ \hline F - D_QC & C_Q & D_Q \end{array}\right]. \tag{3.9}$$

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.

Coprime Factorization Approach

The set of all proper LTI controllers $K(Q)$ that internally stabilize P may be parameterized as follows [7; 20]:

$$K(Q) = (N_k - D_p Q)(D_k - N_p Q)^{-1} \tag{3.10}$$

where

$$N_p = \left[\begin{array}{c|c} A - BF & B \\ \hline C - DF & D \end{array}\right] \qquad D_p = \left[\begin{array}{c|c} A - BF & B \\ \hline -F & I \end{array}\right] \tag{3.11}$$

$$N_k = -\left[\begin{array}{c|c} A - BF & L \\ \hline -F & 0 \end{array}\right] \qquad D_k = \left[\begin{array}{c|c} A - BF & L \\ \hline C - DF & I \end{array}\right] \tag{3.12}$$

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 follows:

$$K(Q) = (\tilde{D}_k - Q\tilde{N}_p)^{-1}(\tilde{N}_k - Q\tilde{D}_p) \tag{3.13}$$

where

$$\tilde{N}_p = \left[\begin{array}{c|c} A - LC & B - LD \\ \hline C & D \end{array}\right] \qquad \tilde{D}_p = \left[\begin{array}{c|c} A - LC & -L \\ \hline C & I \end{array}\right] \tag{3.14}$$

$$\tilde{N}_k = -\left[\begin{array}{c|c} A - LC & L \\ \hline -F & 0 \end{array}\right] \qquad \tilde{D}_k = \left[\begin{array}{c|c} A - LC & -(B - LD) \\ \hline F & I \end{array}\right] \tag{3.15}$$

$\tilde{K}_o = \tilde{D}_k^{-1}\tilde{N}_k$ 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:

$$S_o(Q) = [I + PK(Q)]^{-1} \tag{3.16}$$

$$= [D_k - N_pQ]\,\tilde{D}_p \tag{3.17}$$

$$S_i(Q) = [I + K(Q)P]^{-1} \tag{3.18}$$

$$= D_p\left[\tilde{D}_k - Q\tilde{N}_p\right] \tag{3.19}$$

$$K(Q)S_o(Q) = S_i(Q)K(Q) \tag{3.20}$$

$$= D_p[\tilde{N}_k + Q\tilde{D}_p] \tag{3.21}$$

$$PS_i(Q) = N_p[\tilde{D}_k - Q\tilde{N}_p] \tag{3.22}$$

$$= [D_k - N_pQ]\tilde{N}_p \tag{3.23}$$

$$T_o(Q) = I - S_o(Q) \tag{3.24}$$

$$= N_p[\tilde{N}_k + Q\tilde{D}_p] \tag{3.25}$$

$$T_i(Q) = S_i(Q)K(Q)P \tag{3.26}$$

$$= [D_p N_k \tilde{D}_p^{-1} + D_p Q]\tilde{N}_p \tag{3.27}$$

Achieving Convexity: Q-Parameterization for T_{rz} and T_{diz}.

The closed loop transfer matrices at plant output and plant input are augmented separately to form two distinct transfer function matrices.

$$T_{rz} = \begin{bmatrix} S_o \\ KS_o \\ T_o \end{bmatrix} \tag{3.28}$$

$$T_{diz} = \begin{bmatrix} S_i \\ PS_i \\ T_i \end{bmatrix} \tag{3.29}$$

T_{rz} and T_{diz} can be shown to be affine in the Youla Q-Parameter as follows: The closed loop system can be represented as shown in Figure 3.3 where Q is Youla's parameter and the system T is to be determined below. Note that the general transfer function matrices can be visualized as both T_{rz} and T_{diz}. Hence it is sufficient to show the affine relation between T and Q. The state space representation for T. With x

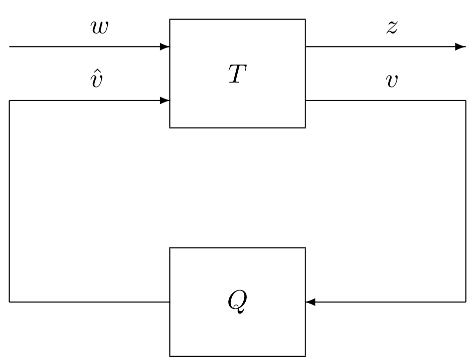


Figure 3.3: Visualization of the Closed Loop System T_{rz} and T_{diz} in terms of T and Q

denoting the states of F and x_k the states of K_{mbc}, we obtain the following

$$\begin{aligned}
\dot{x} &= Ax + BFx - BF(x - x_k) + Bw + B\hat{v} && (3.30)\\
\frac{d}{dt}(x - x_k) &= (A + LC)(x - x_k) + (B + LD)w && (3.31)\\
z &= (C + DF)x - DF(x - x_k) + Dw + D\hat{v} && (3.32)\\
v &= C(x - x_k) + Dw && (3.33)
\end{aligned}$$

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

$$T = \begin{bmatrix} T_1 & T_2 \\ T_3 & 0 \end{bmatrix} = \left[\begin{array}{cc|cc} A + BF & -BF & B & B \\ 0 & A + LC & B + LD & 0 \\ \hline C + DF & -DF & D & D \\ 0 & C & D & 0 \end{array}\right]. \qquad (3.34)$$

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

$$\begin{aligned}
T(Q) &= F_l(T, Q) && (3.35)\\
&= T_1 + T_2 Q T_3. && (3.36)
\end{aligned}$$

This shows that

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

Given the above, we no longer have to search for a stabilizing controller K. Instead, we search over the convex set consisting of all stable transfer function matrices Q. 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.

3.3 Achieving Finite Dimensionality: Introducing a Q-Basis

To obtain a finite-dimensional problem, we express the Q-parameter as a finite linear combination of *a priori* selected stable transfer functions q_k; i.e.

$$Q_N = \sum_{k=1}^{N} X_k q_k \tag{3.37}$$

where

$$X_k = \begin{bmatrix} x_k^{11} & \cdots & x_k^{1n_e} \\ \vdots & & \vdots \\ x_k^{n_u 1} & \cdots & x_k^{n_u n_e} \end{bmatrix} \in \mathcal{R}^{n_u \times n_e} \tag{3.38}$$

Basis Used to Approximate Q. In this work, the following basis chose [18] is chosen

$$q_k = \left(\frac{s - \alpha_a + \alpha_b}{s + \alpha_a + \alpha_b}\right)^{k-1} \quad k = 1, 2, ..., N \tag{3.39}$$

where both α_a and α_b are positive real numbers. Given this, Q can be rewritten as

$$Q_N = X_1 + X_2 \left(\frac{s - \alpha_a + \alpha_b}{s + \alpha_a + \alpha_b}\right) + X_3 \left(\frac{s - \alpha_a + \alpha_b}{s + \alpha_a + \alpha_b}\right)^2 + \cdots \tag{3.40}$$

$$+ X_N \left(\frac{s - \alpha_a + \alpha_b}{s + \alpha_a + \alpha_b}\right)^{N-1}. \tag{3.41}$$

Additional motivation for using a basis consisting of real-rational functions is provided by the following fundamental approximation result. **(Uniform Real-Rational Approximation in $\mathcal{H}^\infty$)**

A function $Q \in \mathcal{H}^\infty$ can be uniformly approximated by real-rational $\mathcal{H}^\infty$ functions if and only if Q is continuous on the extended imaginary axis.

Substituting Q_N into (3.36), then yields the following structure for T_{wz}:

$$T_{wz} = T_1 + T_2 \left(\sum_{k=1}^{N} X_k \, q_k\right) T_3 \tag{3.42}$$

$$= T_1 + \sum_{k=1}^{N} T_2 X_k T_3 \, q_k \tag{3.43}$$

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

$$X_k = \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} B^{ij} x_k^{ij} \tag{3.44}$$

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

$$\begin{aligned} T &= T_1 + \sum_{k=1}^{N} T_2 \left(\sum_{j=1}^{n_e} \sum_{i=1}^{n_u} B^{ij} x_k^{ij} \right) T_3 \, q_k & (3.45) \\ &= T_1 + \sum_{k=1}^{N} \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} T_2 \, B^{ij} \, x_k^{ij} \, T_3 \, q_k & (3.46) \\ &= T_1 + \sum_{k=1}^{N} \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} T_2 \, B^{ij} \, T_3 \, q_k \, x_k^{ij}. & (3.47) \end{aligned}$$

This expression may be written as

$$T = M_o + \sum_{k=1}^{N} \sum_{j=1}^{n_e} \sum_{i=1}^{n_u} M_k^{ij} \, x_k^{ij} \tag{3.48}$$

where

$$\begin{aligned} M_o &= T_1 & (3.49) \\ M_k^{ij} &= T_2 \, B^{ij} \, T_3 \, q_k. & (3.50) \end{aligned}$$

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:

$$\begin{aligned} x_l &= x_k^{ij} & (3.51) \\ M_l &= M_k^{ij}. & (3.52) \end{aligned}$$

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

$$T \;=\; M_o + \sum_{l=1}^{n_u \times n_e \times N} M_l\, x_l. \tag{3.53}$$

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 $x_l = x_k^{ij}$.

3.4 Control System Design Specifications as Convex Constraints

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 [2, page 172]. 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. This makes the approach taken very appealing.

3.5 Convex Optimization Algorithm Used: Pros and Cons

There exist efficient algorithms to solve convex optimization problems. Cutting Plane Methods (CPMs) were proposed independently by Kelley [12] and Cheney and Goldstein [4] for solving constrained convex optimization problems.

Pros and Cons of CPMs)

- Pros

  1. Information Required. CPMs only requires one subgradient at each iteration. No additional effort is needed for nondifferentiable functions.
  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.

- Cons

  1. Speed. Most CPMs are slow, but there are new advanced methods that overcome this problem.
  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.

In this work, Kelley's cutting-plane method is used to solve our Generalized Mixed-Sensitivity $\mathcal{H}^\infty$ optimization problem in Equation 2.9 by

1. generating piecewise affine (linear) functions that provide affine lower bounds to the objective and constraint functions and then
2. solving the linear program formed by these bounds.

The affine lower bounds are generated using only function values and subgradient information, thus the cutting-plane method can be applied to nondifferentiable optimization (NDO) problems. As the method progresses, 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.

3.6 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 Q-parameterization and approximation ideas may be used to transform the problem to a finite-dimensional convex optimization problem for which efficient numerical algorithms exist. 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 will be applied to several control system design problems in what follows.

Chapter 4

SISO $\mathcal{H}^\infty$ DESIGN EXAMPLES

4.1 Introduction

In this chapter, the utility of the design tool is illustrated by designing feedback compensators are for Single Input Single Output plants. The results obtained by solving Generalized $\mathcal{H}^\infty$ mixed sensitivity problem Equation 2.9 are compared with that obtained using Matlab Robust Control Toolbox which uses Standard $\mathcal{H}^\infty$ mixed sensitivity problem in Equation 2.11. Exploiting the Youla Q-Parameterization, an all-pass basis, $\mathcal{H}^\infty$ norm subgradient information, Kelly's cutting plane algorithm permits us to effectively use subgradient information to address non-smooth problems. Convex constraints such as input saturation and peak sensitivity frequency response are also incorporated in the design problem. basis parameters are picked for each plant by using a optimal basis study.

Weighting Functions: The weighting functions used for the SISO examples are as follows:

$$W_1 = \frac{0.01s + 3}{s + 0.03} \tag{4.1}$$

$$W_2 = \frac{100s + 10}{s + 10000} \tag{4.2}$$

$$W_3 = \frac{100s + 40}{s + 2000} \tag{4.3}$$

$$W_4 = 1 \tag{4.4}$$

$$W_5 = 1 \tag{4.5}$$

$$W_6 = 1 \tag{4.6}$$

$$\rho = 10^{-6} \tag{4.7}$$

Selection of the weighting functions:

- W_1 requests a low frequency sensitivity gain of 0.01 (-40 dB) and a sensitivity unity gain crossover frequency of 3 rad/sec.

- W_2 requests a control sensitivity peak no larger than 1000 (60dB) and control sensitivity unity gain crossover frequency of 100 rad/sec.

- W_3 requests a complementary sensitivity unity gain crossover of 20 rad/sec with a high frequency slope of -20 dB/dec.

- A very low value for ρ means that the feedback properties at plant input are penalized negligibly.

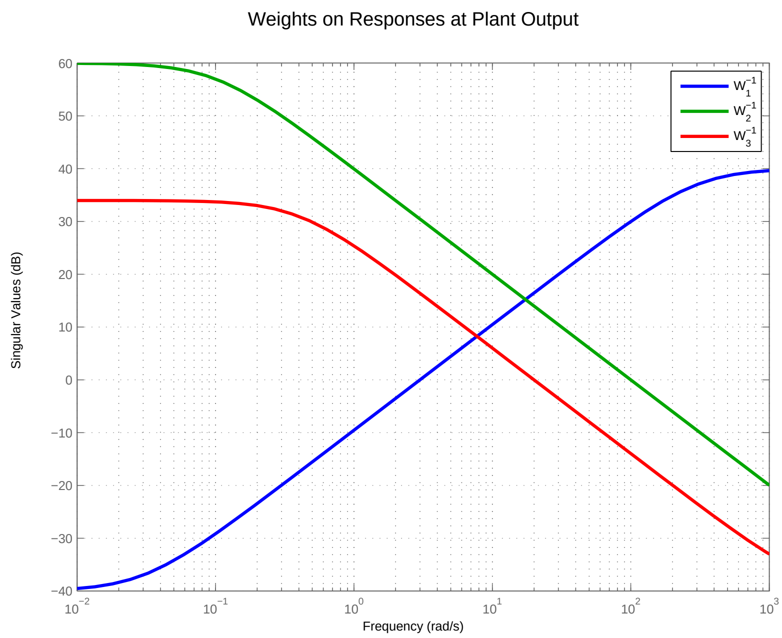


Figure 4.1: Weighting functions

4.2 SISO Stable Plant

Design 1a: SISO Stable plant

$$P = \frac{1}{s+1} \tag{4.8}$$

The optimal Q-Basis parameters used are:

$$Basis = \frac{12-s}{s+12} \qquad N = 7 \tag{4.9}$$

For N > 7, the peak performance, γ does not improve by more than 5%. But for very high values of N, the problem becomes ill-conditioned resulting in very high values of γ.

Design 1b: SISO Stable plant, Constrained

$$P = \frac{1}{s+1} \tag{4.10}$$

Constraint: Control input ≤ 3

The optimal Q-Basis parameters used are:

$$Basis = \frac{20-s}{s+20} \qquad N = 15 \tag{4.11}$$

4.2.1 Unconstrained Case

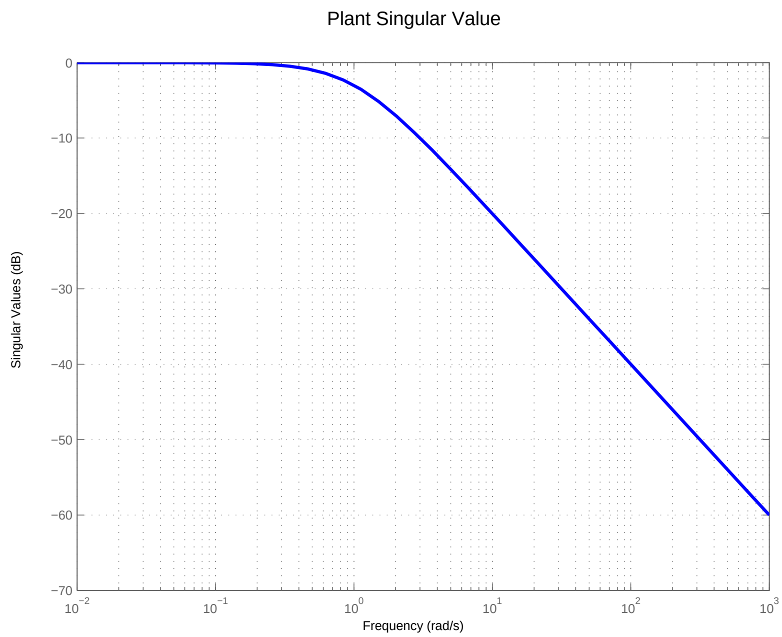


Figure 4.2: Design 1 a and b: Plant Frequency Response

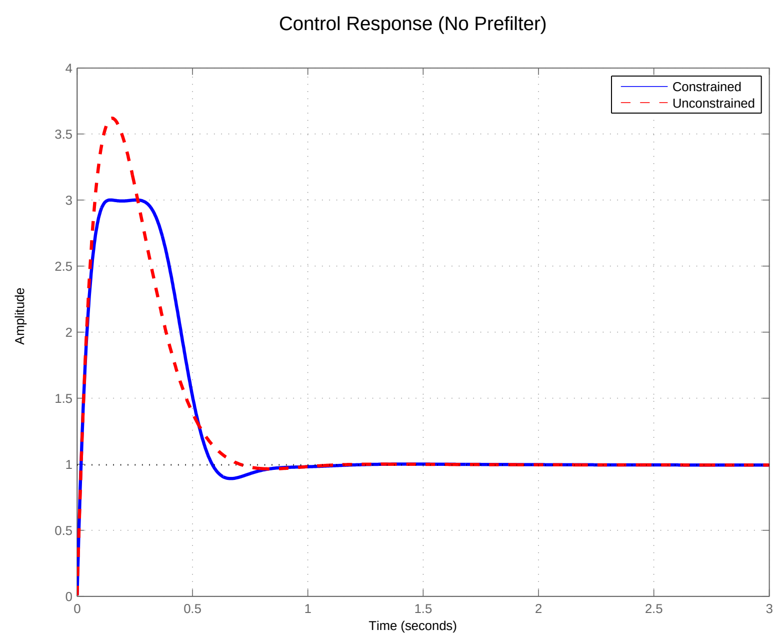


Figure 4.3: Design 1 a and b: Control Time Response

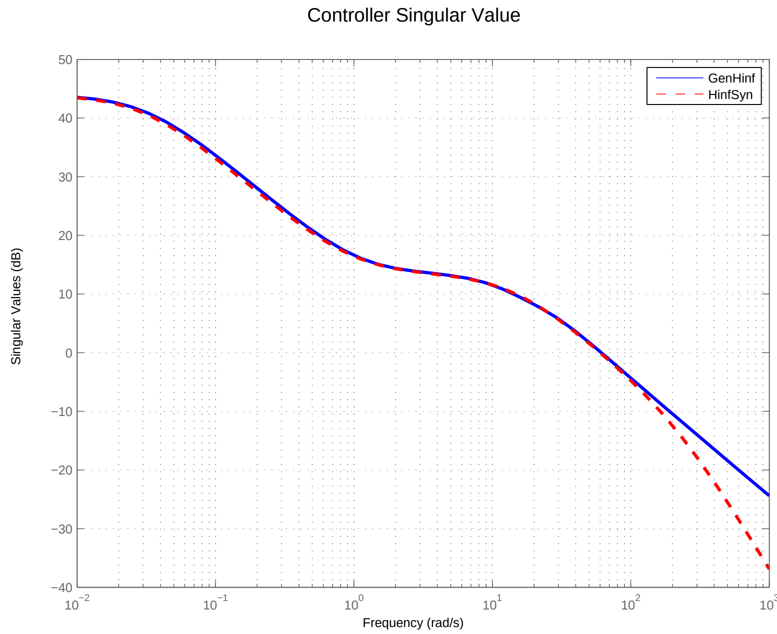


Figure 4.4: Design 1a: Controller Frequency Response

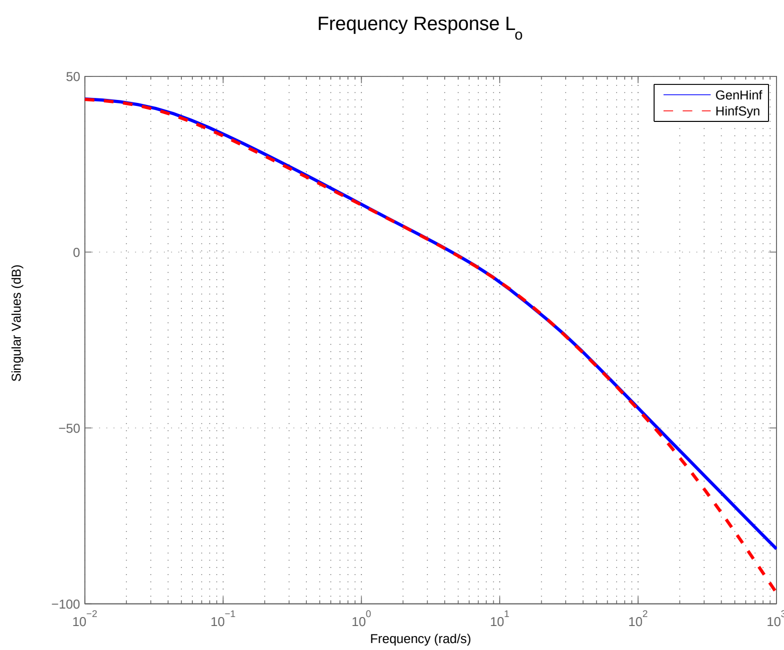


Figure 4.5: Design 1a: Open Loop transfer function

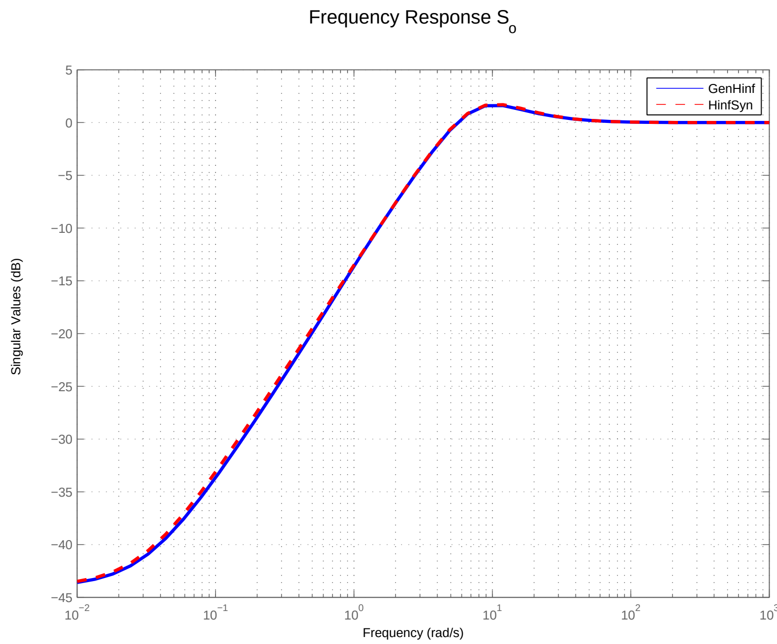


Figure 4.6: Design 1a: Sensitivity Frequency Response

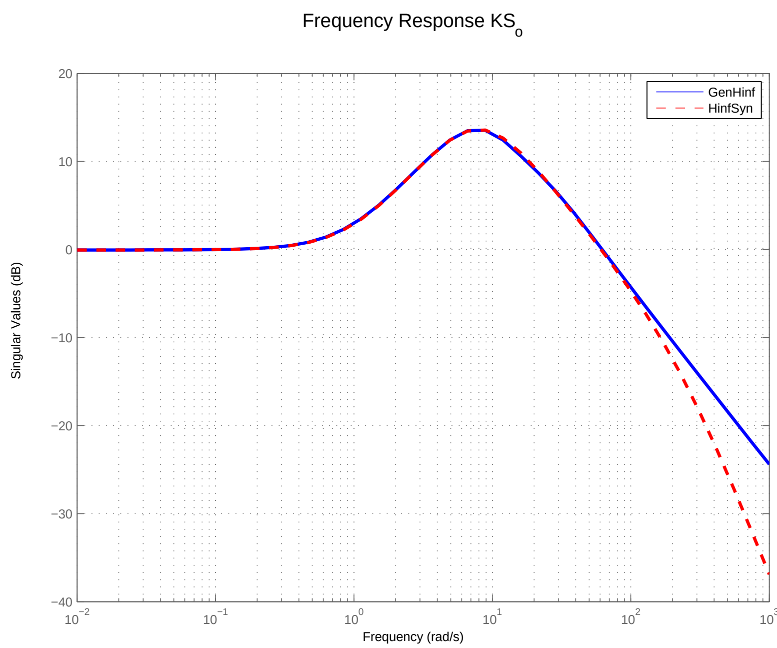


Figure 4.7: Design 1a: K*So

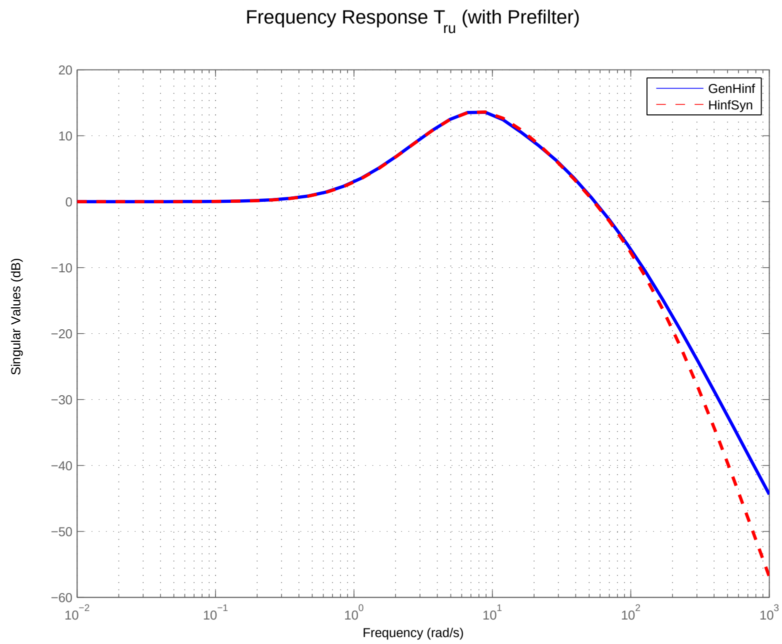


Figure 4.8: Design 1a: Reference to Control transfer function

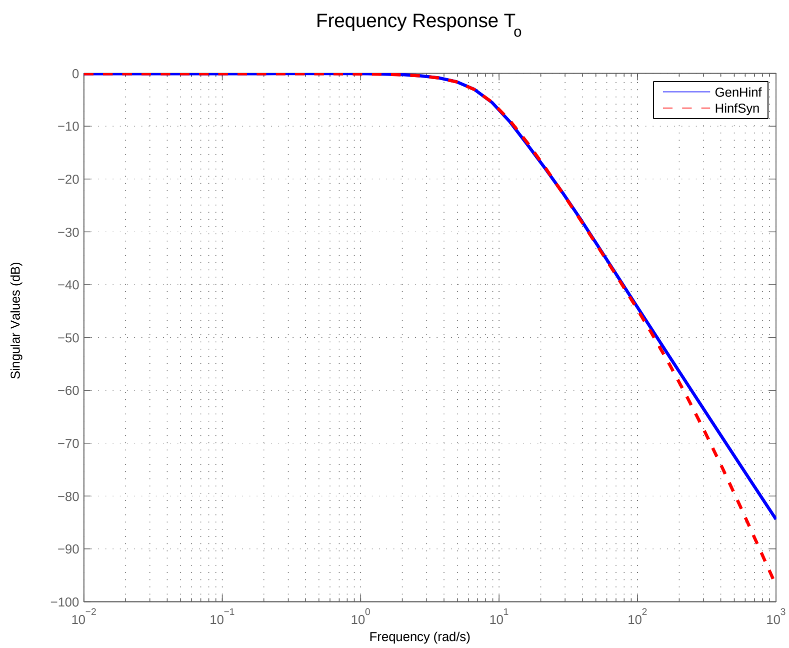


Figure 4.9: Design 1a: Complementary Sensitivity

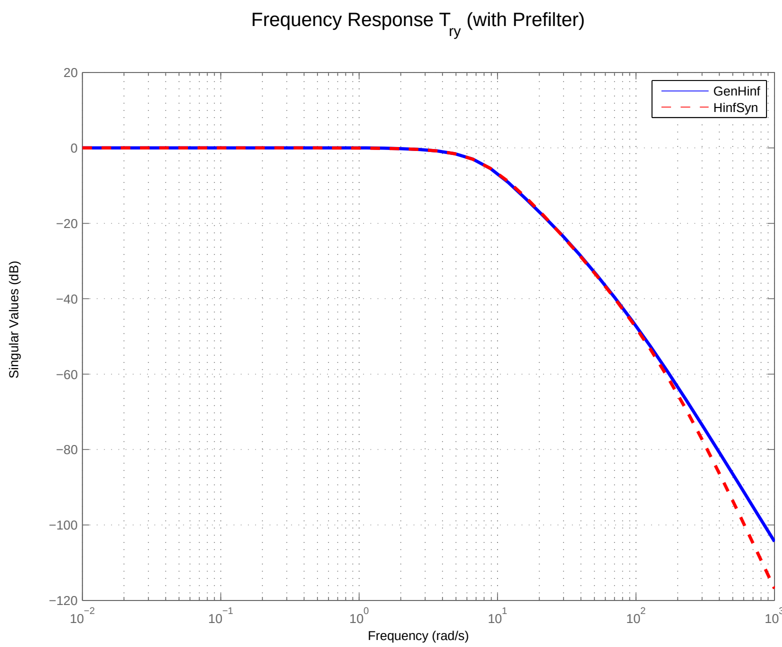


Figure 4.10: Design 1a: Reference to output transfer function

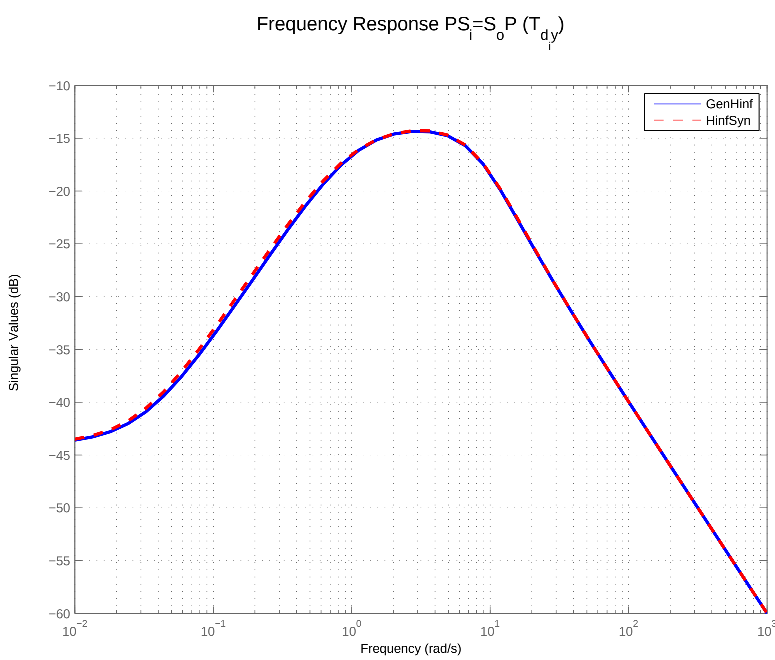


Figure 4.11: Design 1a: $PS_i = S_oP$

Table 4.1: Design 1a using Generalized $\mathcal{H}^{\infty}$: Closed Loop Poles

Poles	Damping	Frequency (rad/sec)
-1.20e+001	1.00e+000	1.20e+001
-1.20e+001 + 2.82e-002i	1.00e+000	1.20e+001
-1.20e+001 - 2.82e-002i	1.00e+000	1.20e+001
-1.20e+001 + 2.82e-002i	1.00e+000	1.20e+001
-1.20e+001 - 2.82e-002i	1.00e+000	1.20e+001
-1.20e+001	1.00e+000	1.20e+001
-1.41e+000	1.00e+000	1.41e+000
-1.41e+000	1.00e+000	1.41e+000

Table 4.2: Design 1a using Generalized $\mathcal{H}^{\infty}$: Closed Loop Zeros

Zeros	Damping	Frequency (rad/sec)
-1.00e+004	1.00e+000	1.00e+004
-2.83e+001	1.00e+000	2.83e+001
-1.25e+001 + 1.10e+001i	7.53e-001	1.66e+001
-1.25e+001 - 1.10e+001i	7.53e-001	1.66e+001
-5.86e+000	1.00e+000	5.86e+000
-1.70e+000	1.00e+000	1.70e+000
-1.21e+000	1.00e+000	1.21e+000

Table 4.3: Design 1a using Matlab HinfSyn: Closed Loop Poles

Poles	Damping	Frequency (rad/sec)
-2.30e+02	1.00e+00	2.30e+02
-1.00e+00	1.00e+00	1.00e+00
-6.64e+00 + 4.46e+00i	8.30e-01	7.99e+00
-6.64e+00 - 4.46e+00i	8.30e-01	7.99e+00
-2.00e+03	1.00e+00	2.00e+03

Table 4.4: Design 1a using Matlab HinfSyn: Closed Loop Zeros

Zeros	Damping	Frequency (rad/sec)
-2.00e+03	1.00e+00	2.00e+03
-1.00e+04	1.00e+00	1.00e+04
-1.00e+00	1.00e+00	1.00e+00

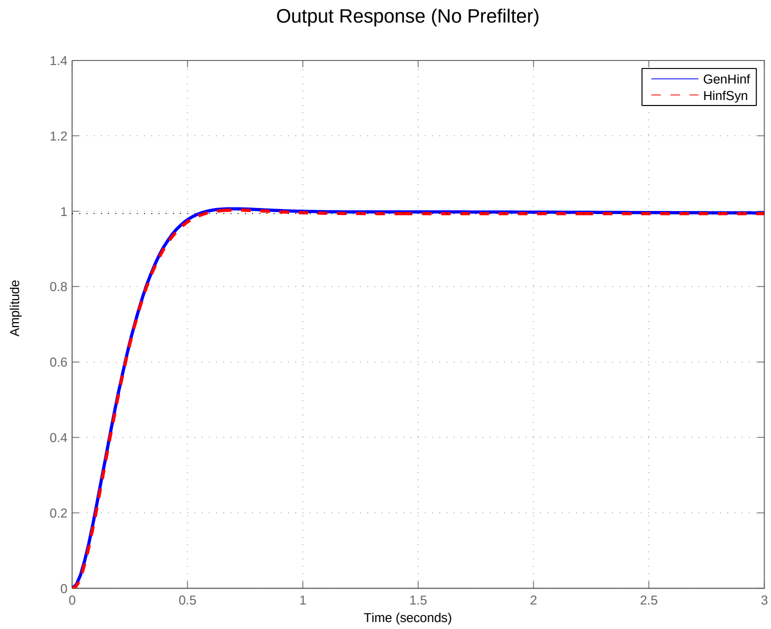


Figure 4.12: Design 1a: Output Time Response (no Pre-filter)

Table 4.5: Design 1a: $\mathcal{H}^\infty$ norms of individual transfer functions (dB)

$S_o = S_i$	$T_o = T_i$	KS_o	PS_i
1.6267	-0.0550	13.6740	-14.3075

4.2.2 Constrained Case

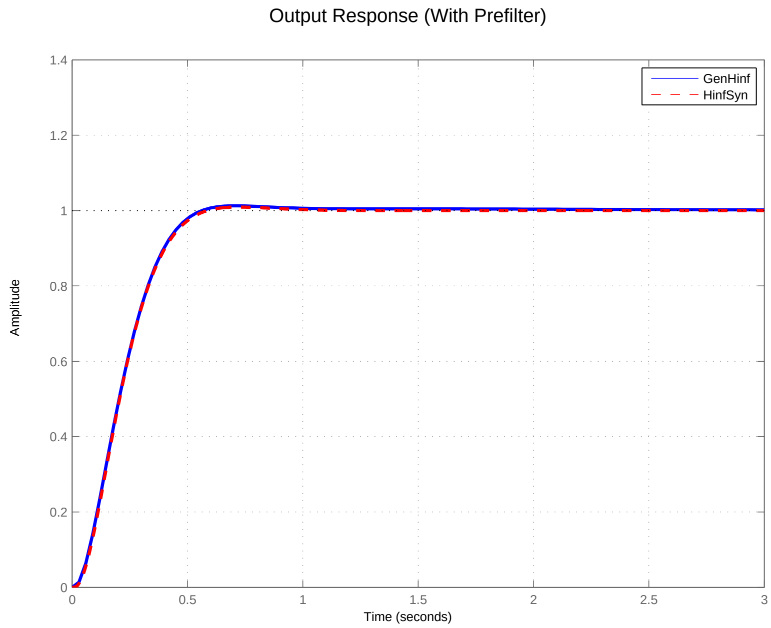


Figure 4.13: Design 1a: Output Time Response (with Pre-filter)

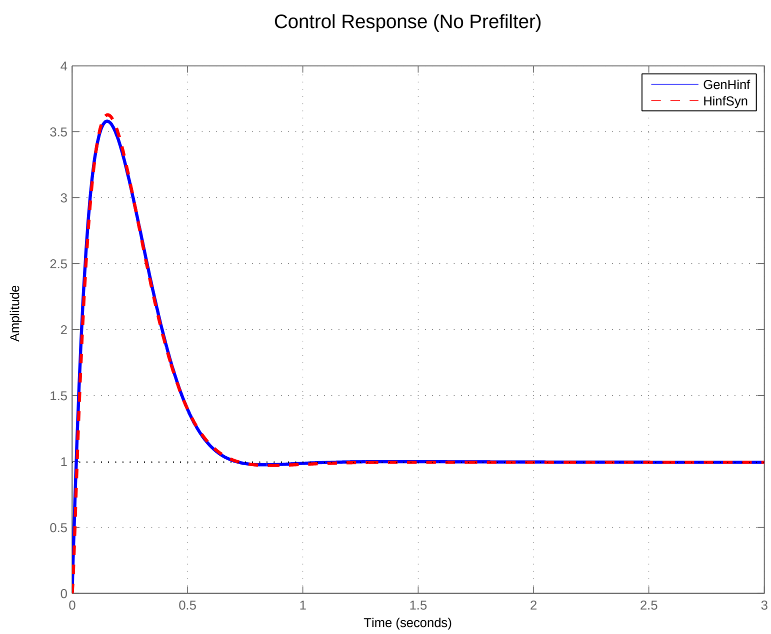


Figure 4.14: Design 1a: Control Time Response (no Pre-filter)

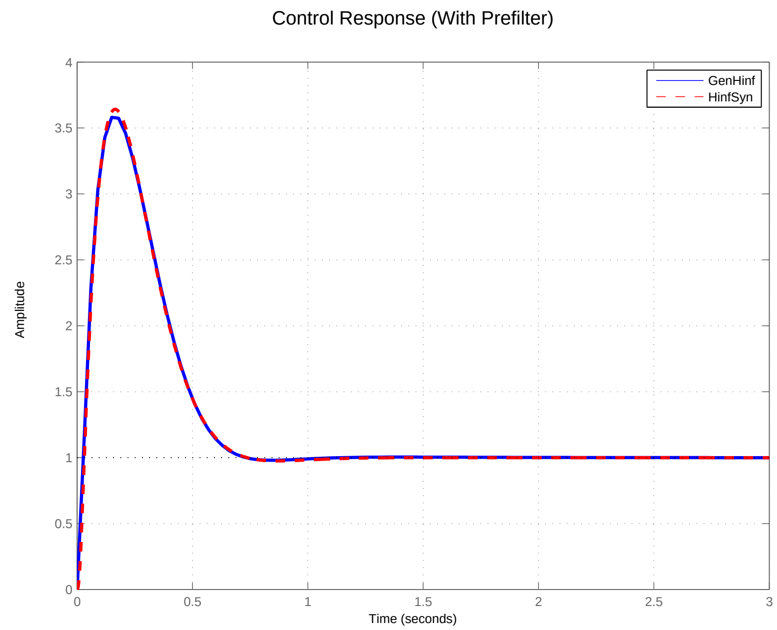


Figure 4.15: Design 1a: Control Time Response (with Pre-filter)

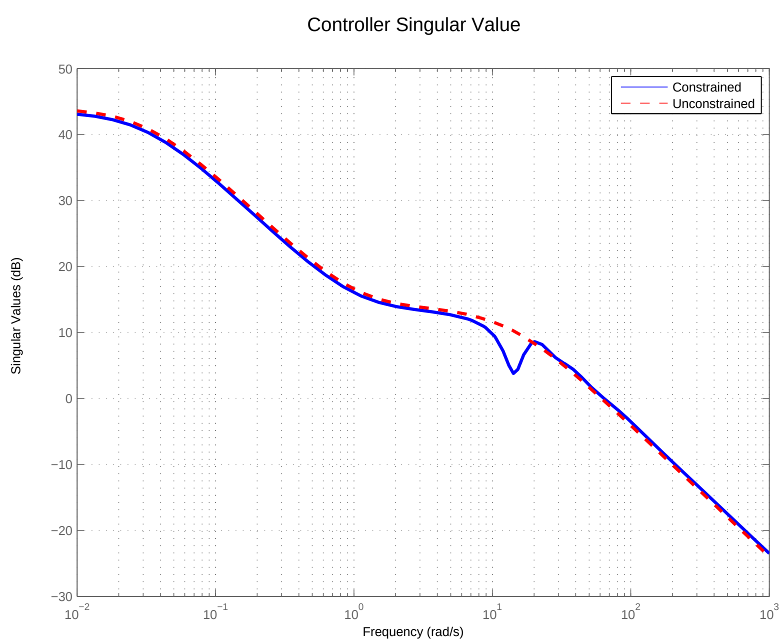


Figure 4.16: Design 1b: Controller Frequency Response

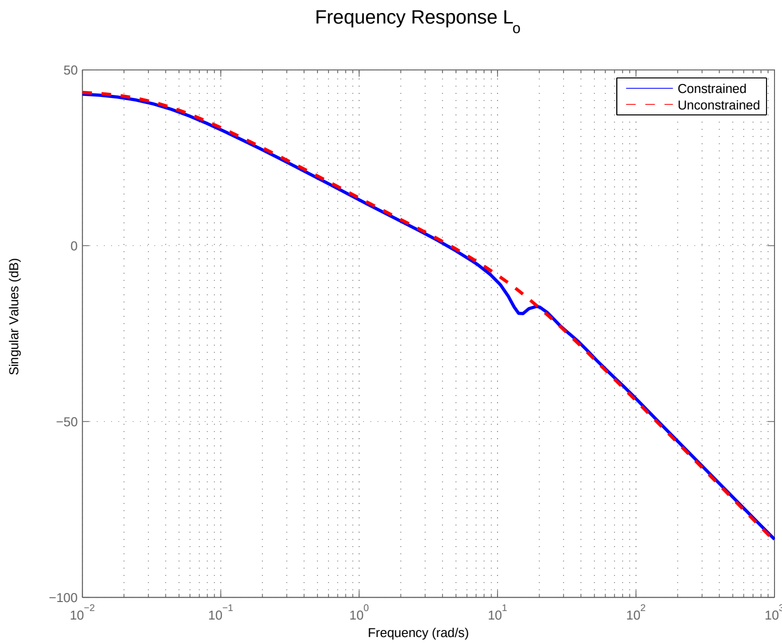


Figure 4.17: Design 1b: Open Loop transfer function

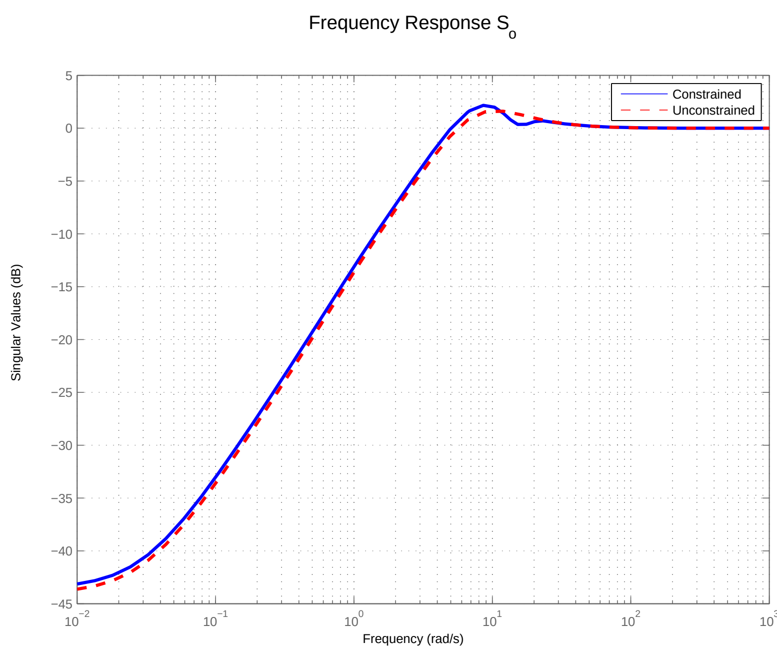


Figure 4.18: Design 1b: Sensitivity Frequency Response

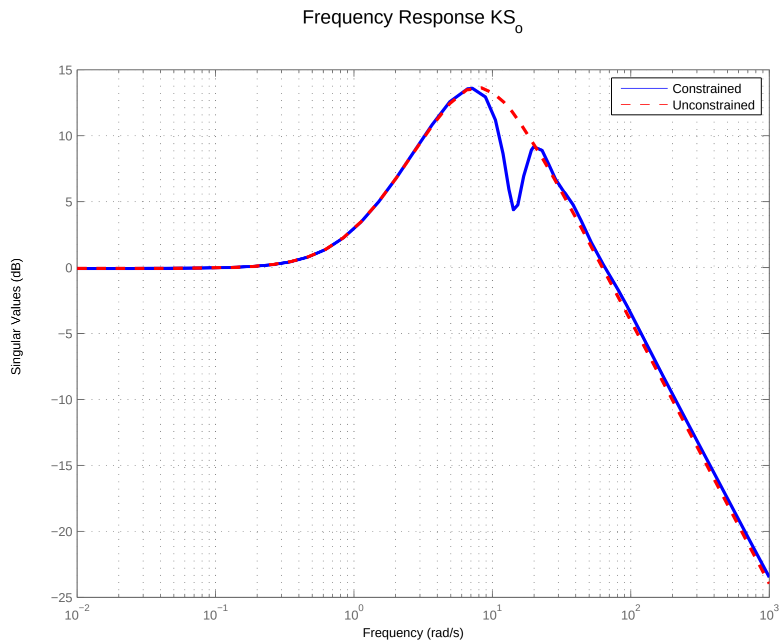


Figure 4.19: Design 1b: K*So

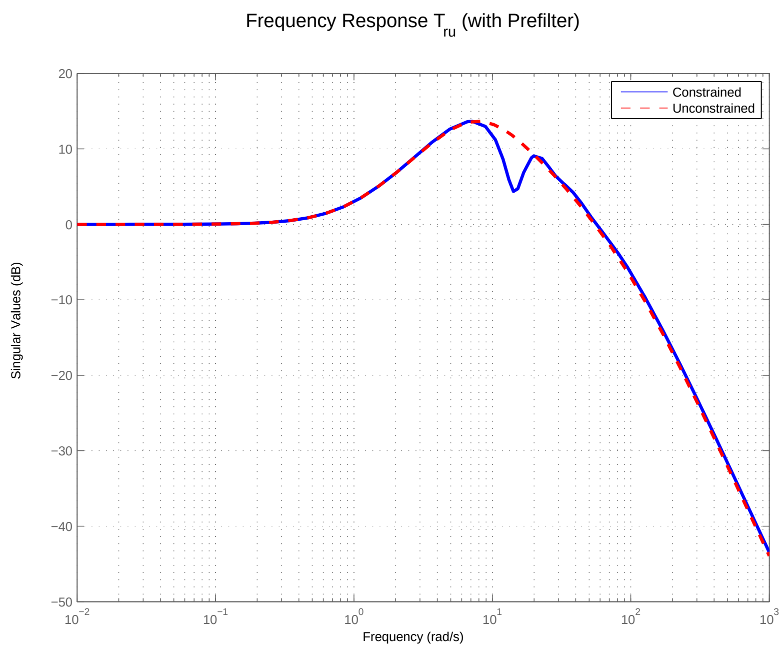


Figure 4.20: Design 1b: Reference to Control transfer function

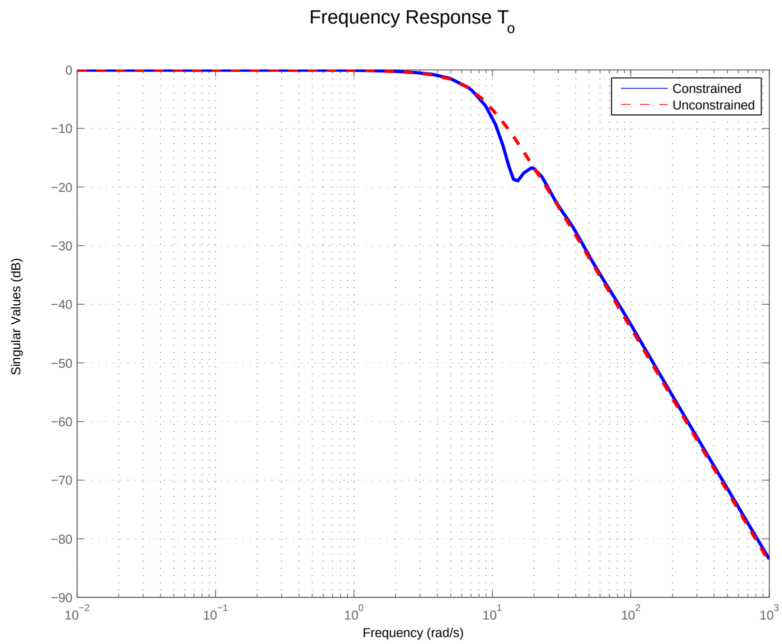


Figure 4.21: Design 1b: Complementary Sensitivity

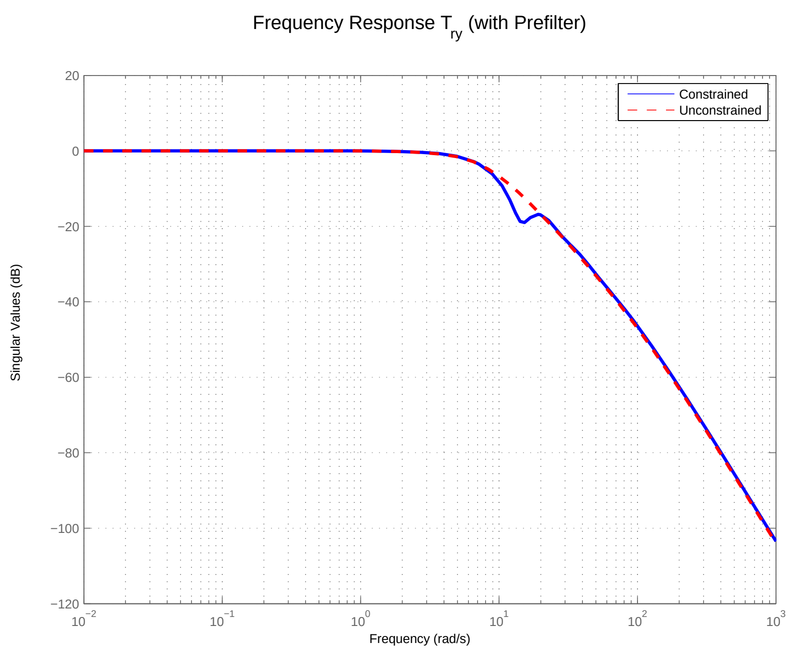


Figure 4.22: Design 1b: Reference to output transfer function

Table 4.6: Design 1b using Generalized $\mathcal{H}^{\infty}$: Closed Loop Poles

Poles	Damping	Frequency (rad/sec)
-2.22e+001	1.00e+000	2.22e+001
-2.20e+001 + 9.86e-001i	9.99e-001	2.20e+001
-2.20e+001 - 9.86e-001i	9.99e-001	2.20e+001
-2.13e+001 + 1.75e+000i	9.97e-001	2.14e+001
-2.13e+001 - 1.75e+000i	9.97e-001	2.14e+001
-2.04e+001 + 2.13e+000i	9.95e-001	2.05e+001
-2.04e+001 - 2.13e+000i	9.95e-001	2.05e+001
-1.95e+001 + 2.08e+000i	9.94e-001	1.96e+001
-1.95e+001 - 2.08e+000i	9.94e-001	1.96e+001
-1.86e+001 + 1.62e+000i	9.96e-001	1.87e+001
-1.86e+001 - 1.62e+000i	9.96e-001	1.87e+001
-1.81e+001 + 8.72e-001i	9.99e-001	1.81e+001
-1.81e+001 - 8.72e-001i	9.99e-001	1.81e+001
-1.79e+001	1.00e+000	1.79e+001
-1.41e+000	1.00e+000	1.41e+000
-1.41e+000	1.00e+000	1.41e+000

Table 4.7: Design 1b using Generalized $\mathcal{H}^{\infty}$: Closed Loop Zeros

Zeros	Damping	Frequency (rad/sec)
-1.00e+004	1.00e+000	1.00e+004
-9.11e+001	1.00e+000	9.11e+001
-5.10e+001 + 4.65e+001i	7.39e-001	6.90e+001
-5.10e+001 - 4.65e+001i	7.39e-001	6.90e+001
-1.93e+001 + 3.73e+001i	4.60e-001	4.21e+001
-1.93e+001 - 3.73e+001i	4.60e-001	4.21e+001
-7.48e+000 + 2.49e+001i	2.87e-001	2.60e+001
-7.48e+000 - 2.49e+001i	2.87e-001	2.60e+001
-1.23e+000 + 1.40e+001i	8.75e-002	1.41e+001
-1.23e+000 - 1.40e+001i	8.75e-002	1.41e+001
-4.58e+000 + 5.89e+000i	6.14e-001	7.46e+000
-4.58e+000 - 5.89e+000i	6.14e-001	7.46e+000
-4.08e+000	1.00e+000	4.08e+000
-1.75e+000	1.00e+000	1.75e+000
-1.20e+000	1.00e+000	1.20e+000

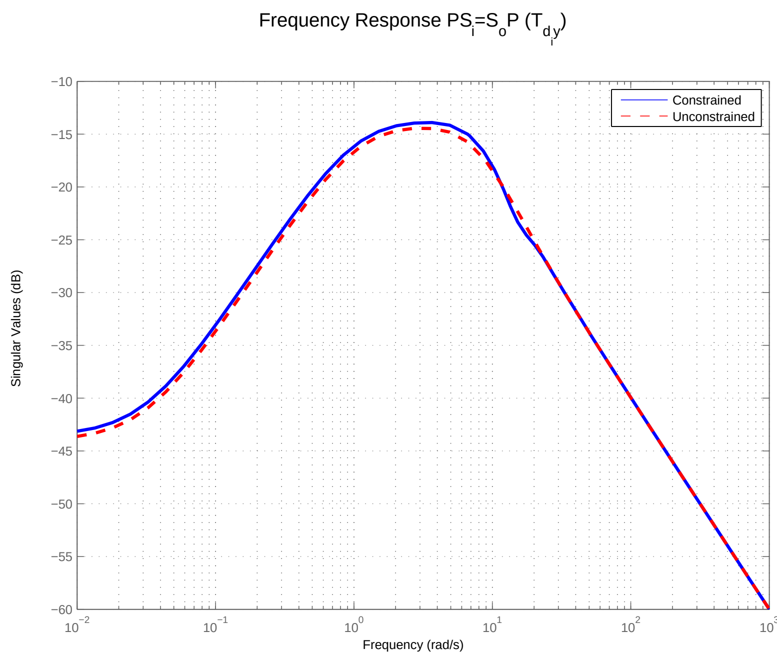


Figure 4.23: Design 1b: $PS_i = S_oP$

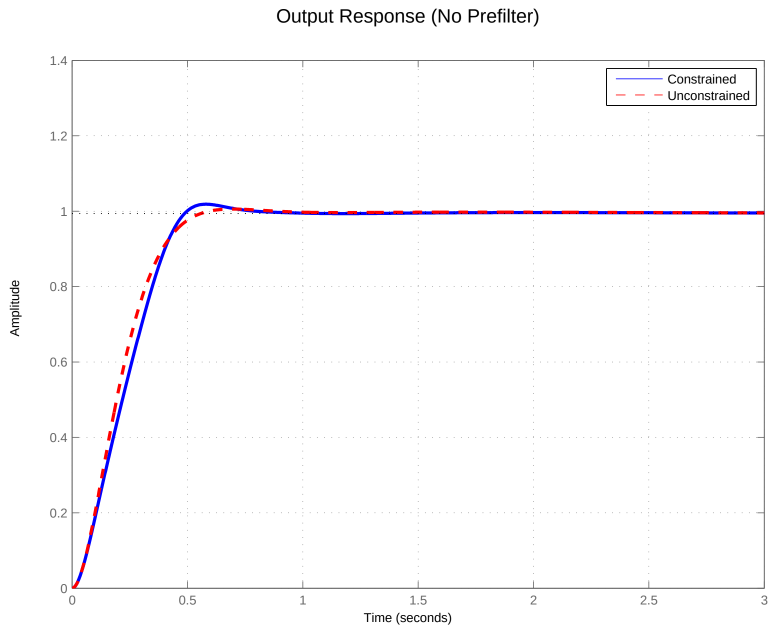


Figure 4.24: Design 1b: Output Time Response (no Pre-filter)

Table 4.8: Design 1b: $\mathcal{H}^\infty$ norms of individual transfer functions (dB)

$S_o = S_i$	$T_o = T_i$	KS_o	PS_i
2.2992	-0.0578	13.7793	-13.8667

4.3 SISO Unstable Plant

Design 2a: SISO Unstable plant

$$P = \frac{1}{s-1} \tag{4.12}$$

The optimal Q-Basis parameters used are:

$$Basis = \frac{5-s}{s+5} \qquad\qquad N = 4 \tag{4.13}$$

Design 2b: SISO Unstable plant, Constrained

$$P = \frac{1}{s-1} \tag{4.14}$$

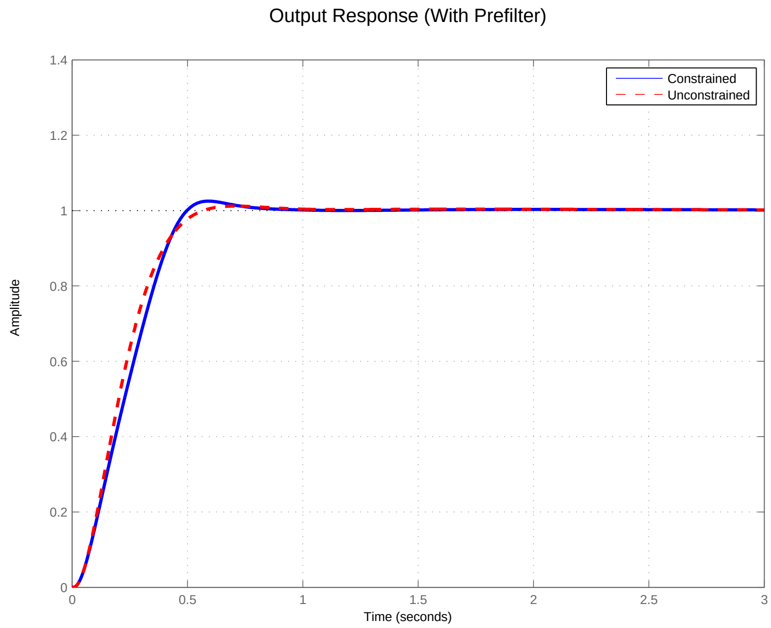


Figure 4.25: Design 1b: Output Time Response (with Pre-filter)

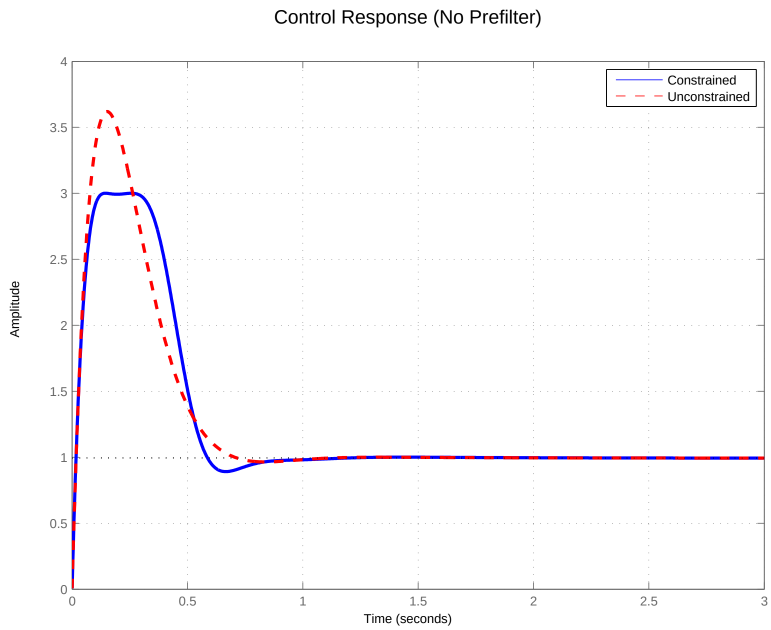


Figure 4.26: Design 1b: Control Time Response (no Pre-filter)

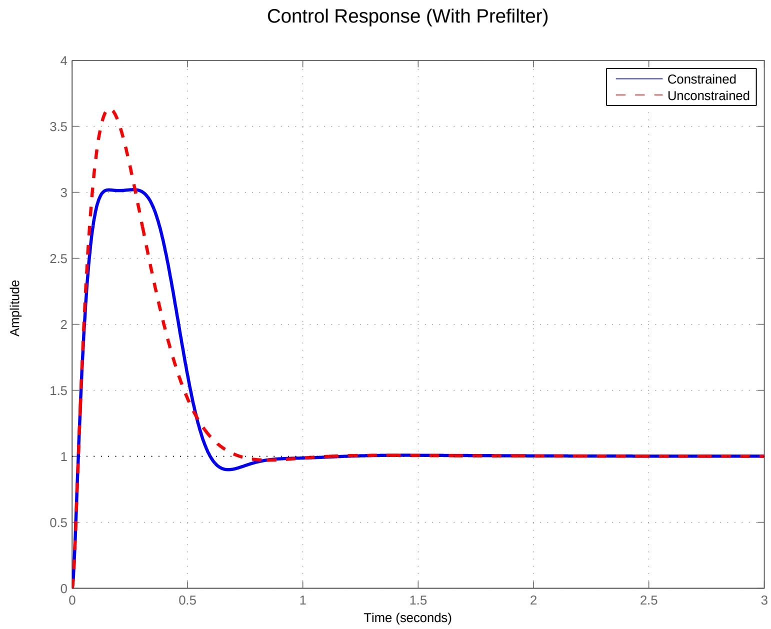


Figure 4.27: Design 1b: Control Time Response (with Pre-filter)

Constraint: KS_o frequency response $\leq 3.9811\,(12dB)$ The optimal Q-Basis parameters used are:

$$Basis = \frac{7-s}{s+7} \qquad\qquad N = 15 \tag{4.15}$$

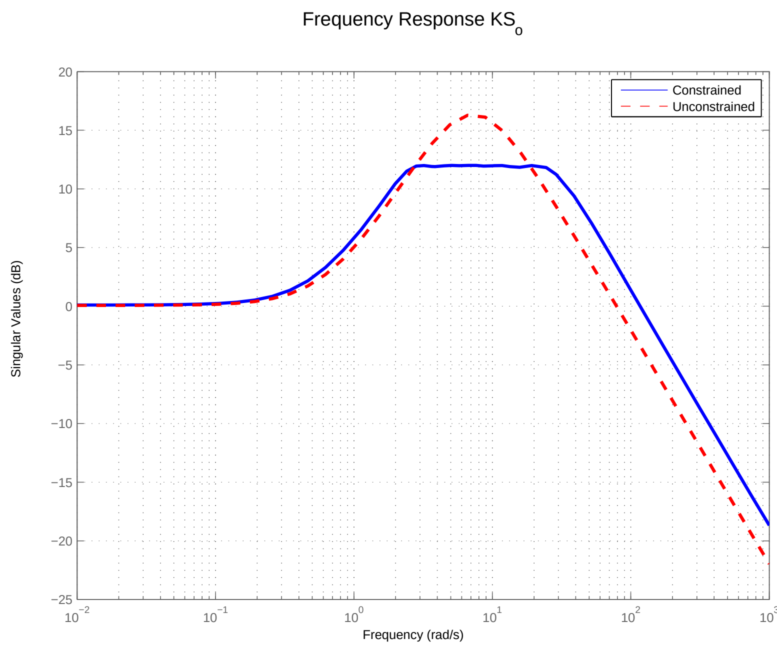


Figure 4.28: Design 2 a and b: K*So

4.3.1 Unconstrained Case

Table 4.9: Design 2a using Generalized $\mathcal{H}^{\infty}$: Closed Loop Poles

Poles	Damping	Frequency (rad/sec)
-2.30e+02	1.00e+00	2.30e+02
-1.00e+00	1.00e+00	1.00e+00
-6.64e+00 + 4.46e+00i	8.30e-01	7.99e+00
-6.64e+00 - 4.46e+00i	8.30e-01	7.99e+00
-2.00e+03	1.00e+00	2.00e+03

Table 4.10: Design 2a using Generalized $\mathcal{H}^{\infty}$: Closed Loop Zeros

Zeros	Damping	Frequency (rad/sec)
-5.00e+000	1.00e+000	5.00e+000
-5.00e+000 + 4.53e-005i	1.00e+000	5.00e+000
-5.00e+000 - 4.53e-005i	1.00e+000	5.00e+000
-1.41e+000	1.00e+000	1.41e+000
-1.41e+000	1.00e+000	1.41e+000

Table 4.11: Design 2a using Matlab HinfSyn: Closed Loop Poles

Poles	Damping	Frequency (rad/sec)
-1.09e+04	1.00e+00	1.09e+04
-1.00e+00	1.00e+00	1.00e+00
-6.05e+00 + 4.08e+00i	8.29e-01	7.30e+00
-6.05e+00 - 4.08e+00i	8.29e-01	7.30e+00
-2.00e+03	1.00e+00	2.00e+03

Table 4.12: Design 2a using Matlab HinfSyn: Closed Loop Zeros

Zeros	Damping	Frequency (rad/sec)
-1.00e+04	1.00e+00	1.00e+04
-2.00e+03	1.00e+00	2.00e+03
-6.80e-01	1.00e+00	6.80e-01

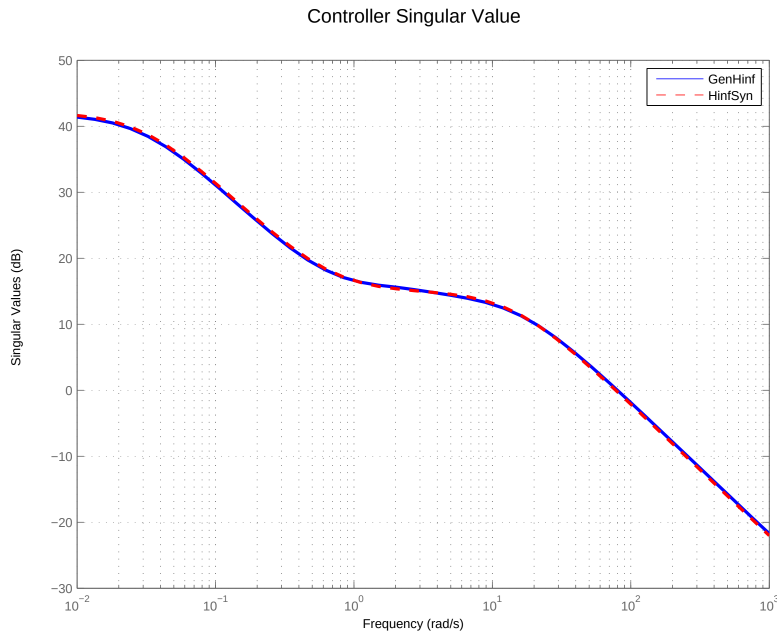


Figure 4.29: Design 2a: Controller Frequency Response

Table 4.13: Design 2a: $\mathcal{H}^\infty$ norms of individual transfer functions (dB)

$S_o = S_i$	$T_o = T_i$	KS_o	PS_i
2.3807	2.6960	16.1427	-12.3574

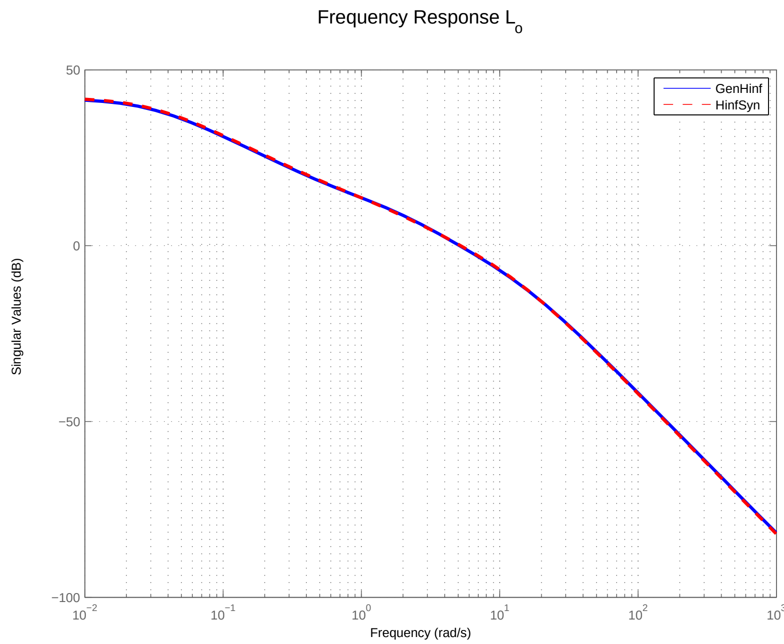


Figure 4.30: Design 2a: Open Loop transfer function

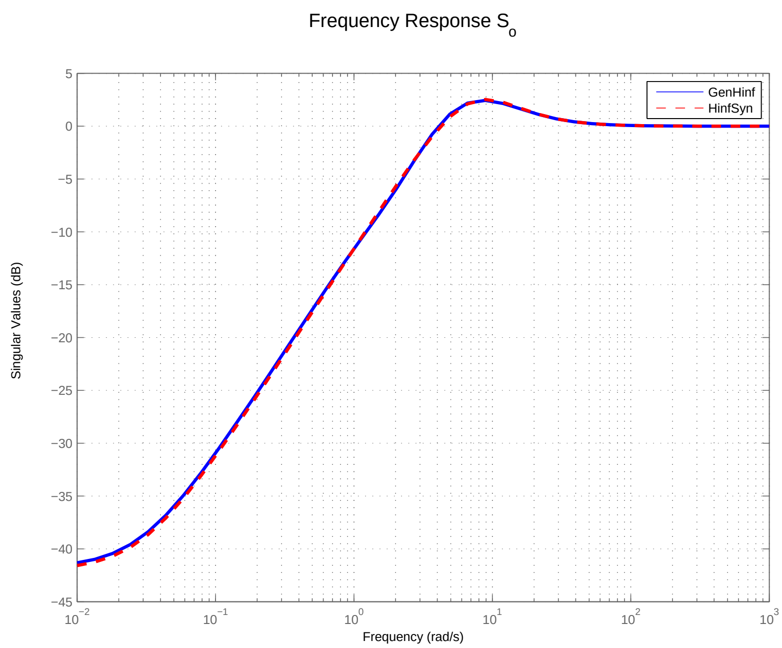


Figure 4.31: Design 2a: Sensitivity Frequency Response

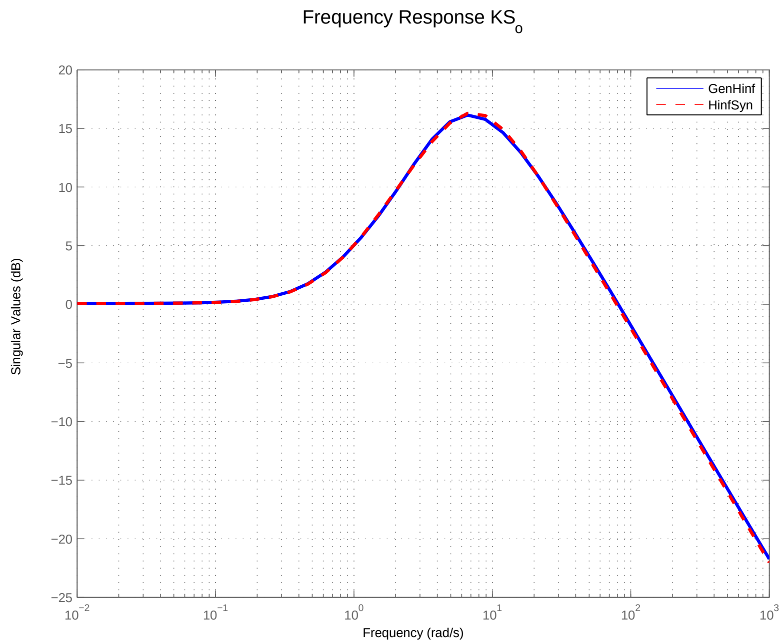


Figure 4.32: Design 2a: K*So

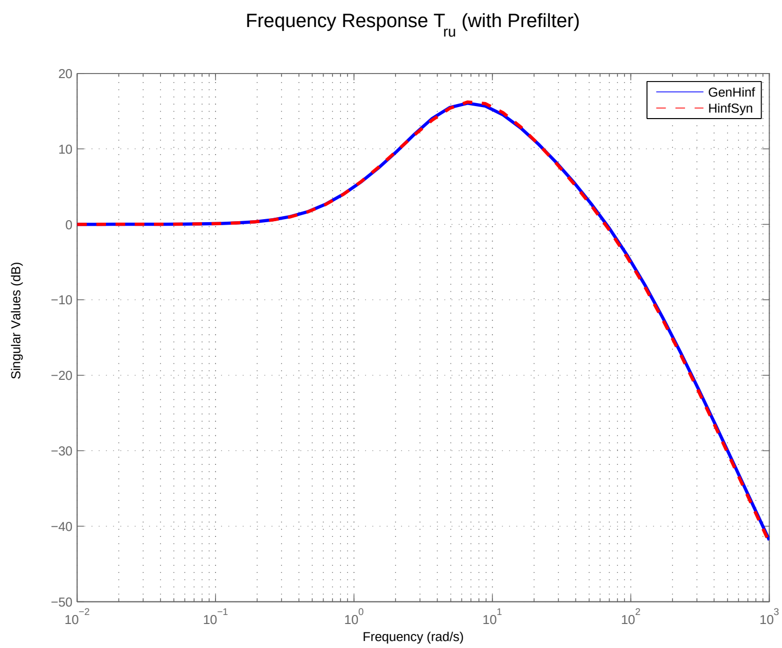


Figure 4.33: Design 2a: Reference to Control transfer function

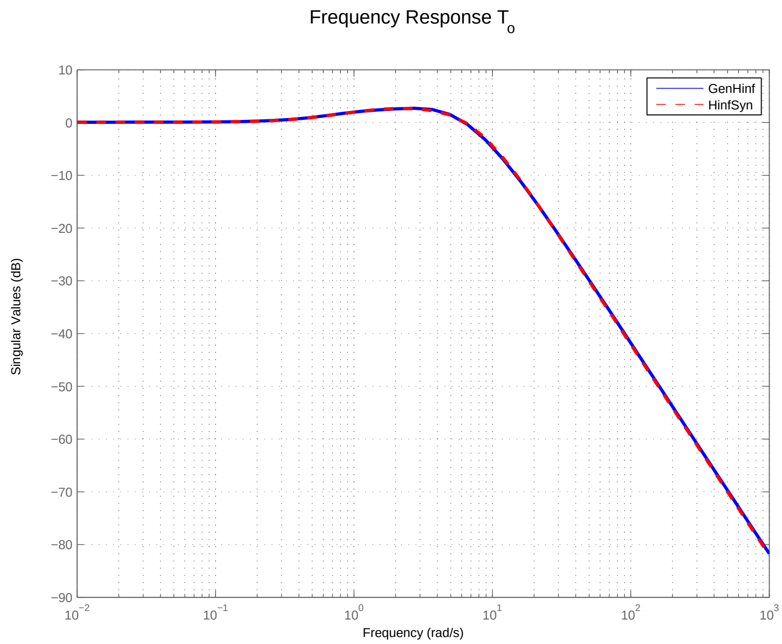


Figure 4.34: Design 2a: Complementary Sensitivity

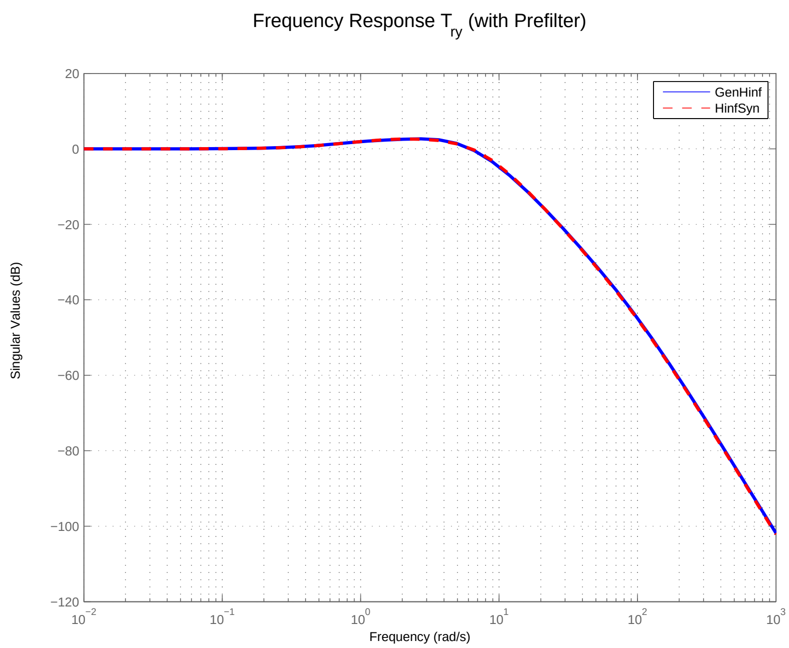


Figure 4.35: Design 2a: Reference to output transfer function

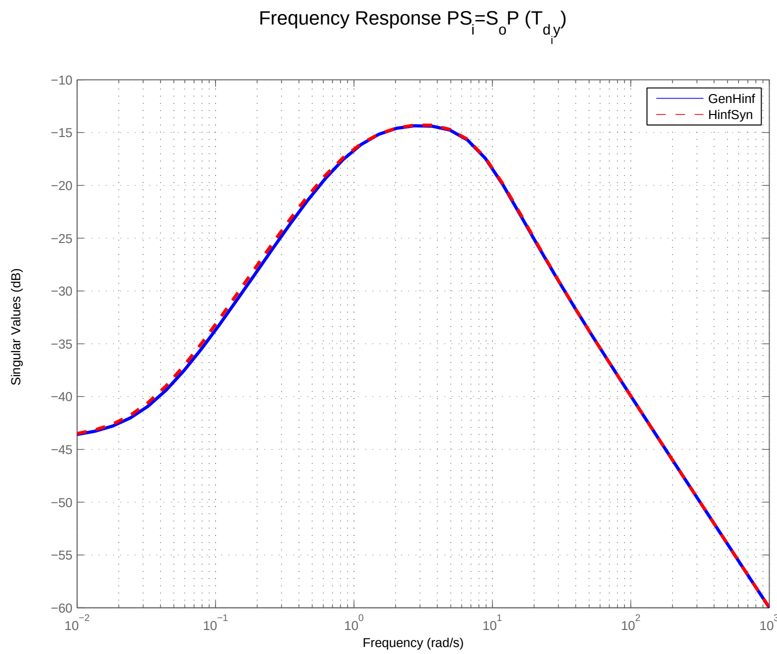


Figure 4.36: Design 2a: $PS_i = S_oP$

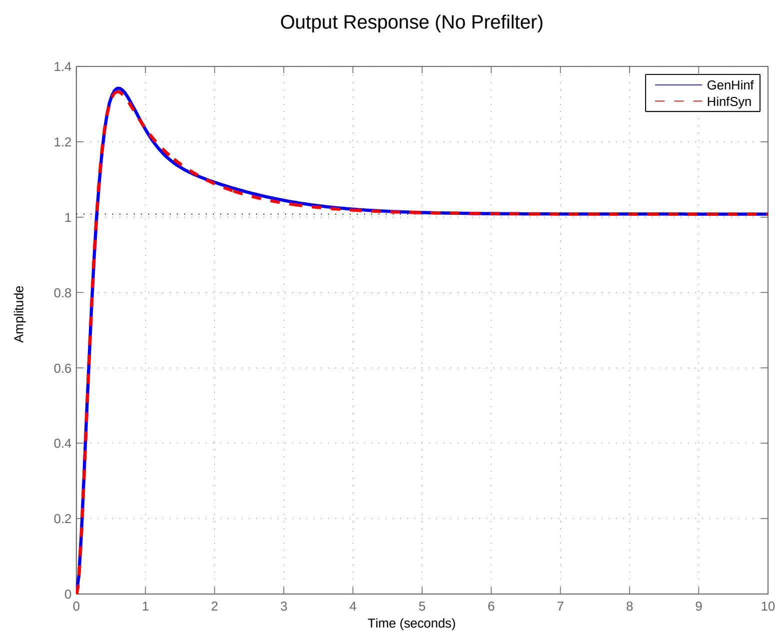


Figure 4.37: Design 2a: Output Time Response (no Pre-filter)

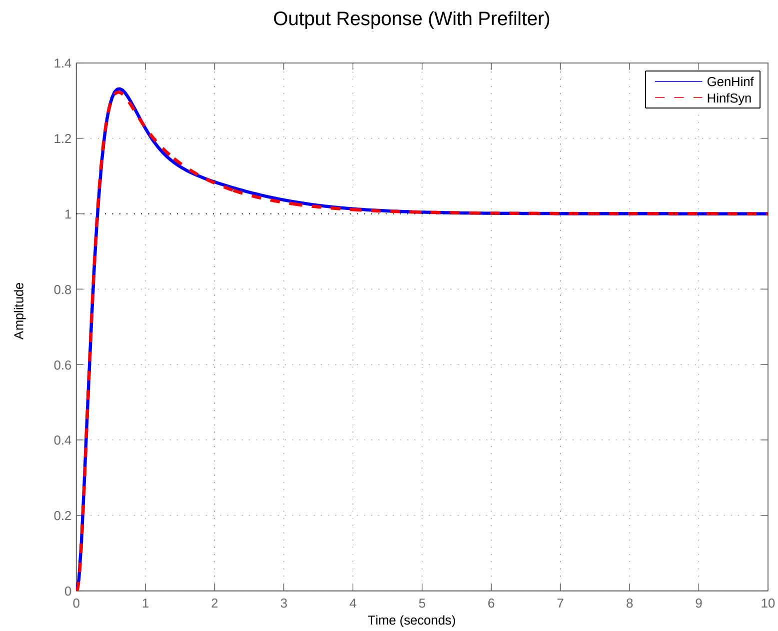


Figure 4.38: Design 2a: Output Time Response (with Pre-filter)

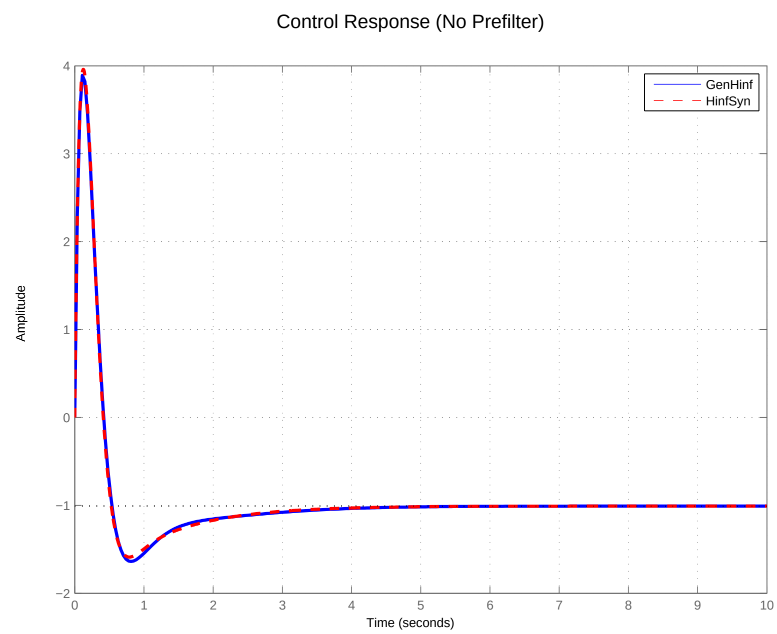


Figure 4.39: Design 2a: Control Time Response (no Pre-filter)

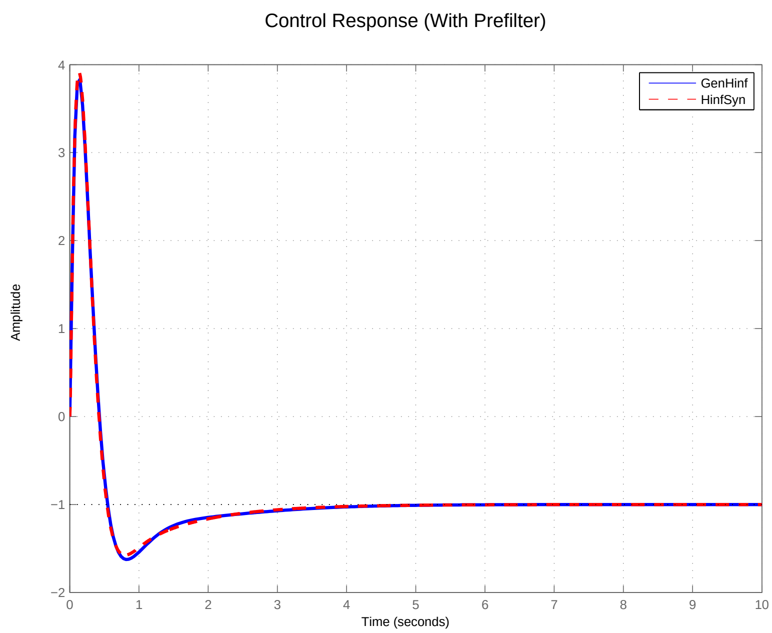


Figure 4.40: Design 2a: Control Time Response (with Pre-filter)

4.3.2 Constrained Case

Table 4.14: Design 2b using Generalized $\mathcal{H}^{\infty}$: Closed Loop Poles

Poles	Damping	Frequency (rad/sec)
-7.77e+000	1.00e+000	7.77e+000
-7.69e+000 + 3.34e-001i	9.99e-001	7.69e+000
-7.69e+000 - 3.34e-001i	9.99e-001	7.69e+000
-7.47e+000 + 6.00e-001i	9.97e-001	7.50e+000
-7.47e+000 - 6.00e-001i	9.97e-001	7.50e+000
-7.16e+000 + 7.42e-001i	9.95e-001	7.20e+000
-7.16e+000 - 7.42e-001i	9.95e-001	7.20e+000
-6.83e+000 + 7.36e-001i	9.94e-001	6.86e+000
-6.83e+000 - 7.36e-001i	9.94e-001	6.86e+000
-6.53e+000 + 5.86e-001i	9.96e-001	6.55e+000
-6.53e+000 - 5.86e-001i	9.96e-001	6.55e+000
-6.32e+000 + 3.24e-001i	9.99e-001	6.33e+000
-6.32e+000 - 3.24e-001i	9.99e-001	6.33e+000
-6.25e+000	1.00e+000	6.25e+000
-1.41e+000	1.00e+000	1.41e+000
-1.41e+000	1.00e+000	1.41e+000

Table 4.15: Design 2b using Generalized $\mathcal{H}^{\infty}$: Closed Loop Zeros

Zeros	Damping	Frequency (rad/sec)
-1.00e+004	1.00e+000	1.00e+004
-3.35e+001	1.00e+000	3.35e+001
-7.14e+000 + 1.29e+001i	4.85e-001	1.47e+001
-7.14e+000 - 1.29e+001i	4.85e-001	1.47e+001
-3.96e+000 + 8.33e+000i	4.30e-001	9.22e+000
-3.96e+000 - 8.33e+000i	4.30e-001	9.22e+000
-1.07e+001	1.00e+000	1.07e+001
-2.57e+000 + 5.68e+000i	4.12e-001	6.23e+000
-2.57e+000 - 5.68e+000i	4.12e-001	6.23e+000
-1.68e+000 + 3.64e+000i	4.19e-001	4.01e+000
-1.68e+000 - 3.64e+000i	4.19e-001	4.01e+000
-1.34e+000 + 1.26e+000i	7.30e-001	1.84e+000
-1.34e+000 - 1.26e+000i	7.30e-001	1.84e+000
-6.24e-001	1.00e+000	6.24e-001
-1.32e+000	1.00e+000	1.32e+000

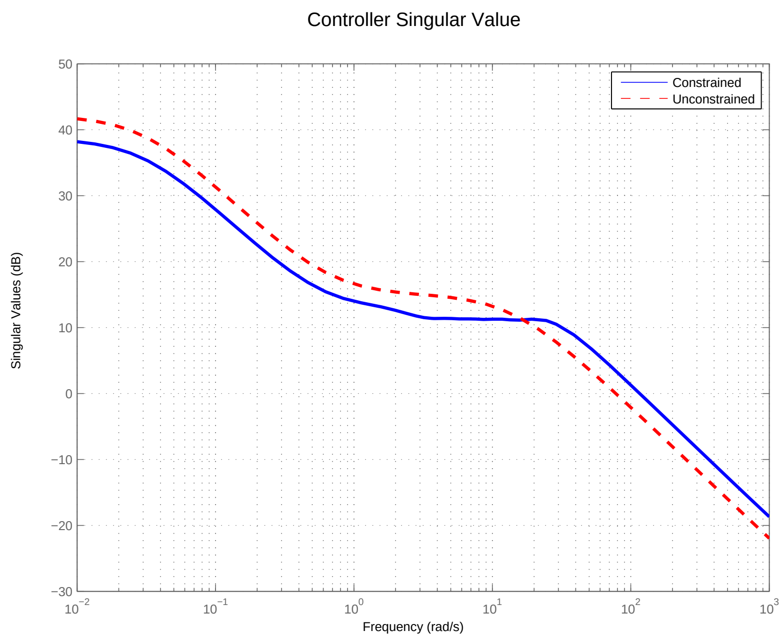


Figure 4.41: Design 2b: Controller Frequency Response

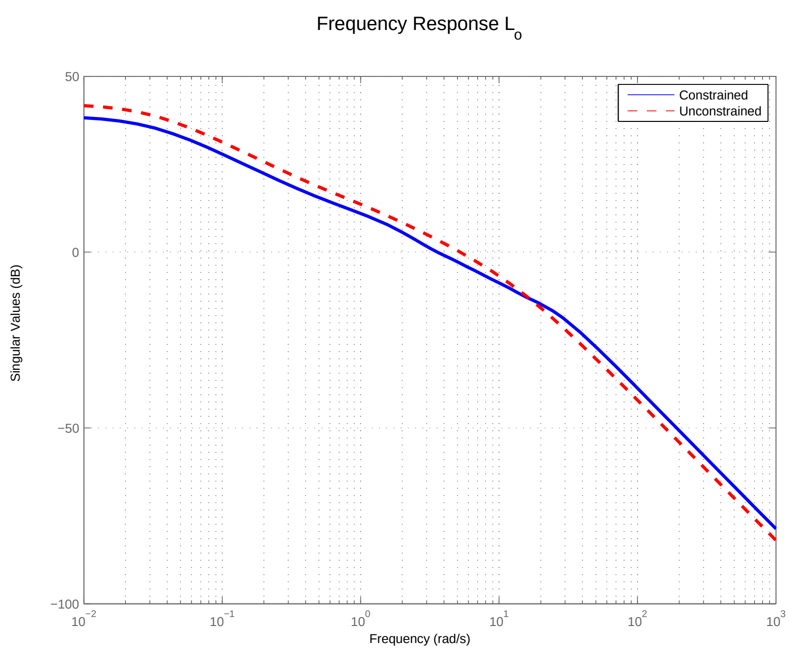


Figure 4.42: Design 2b: Open Loop transfer function

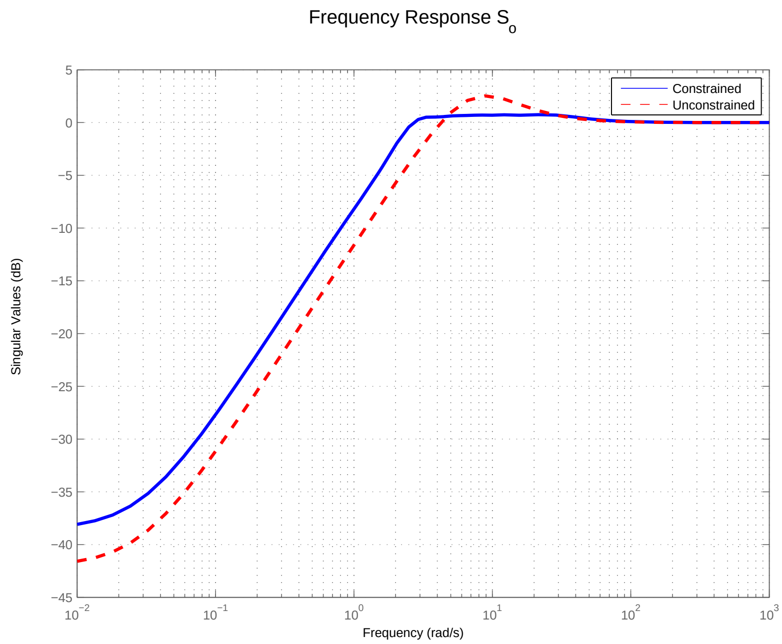


Figure 4.43: Design 2b: Sensitivity Frequency Response

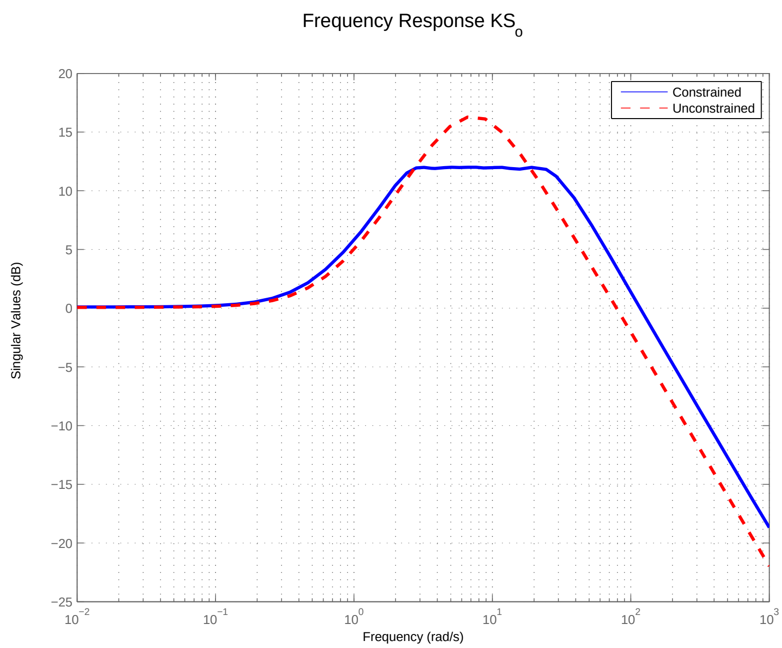


Figure 4.44: Design 2b: K*So

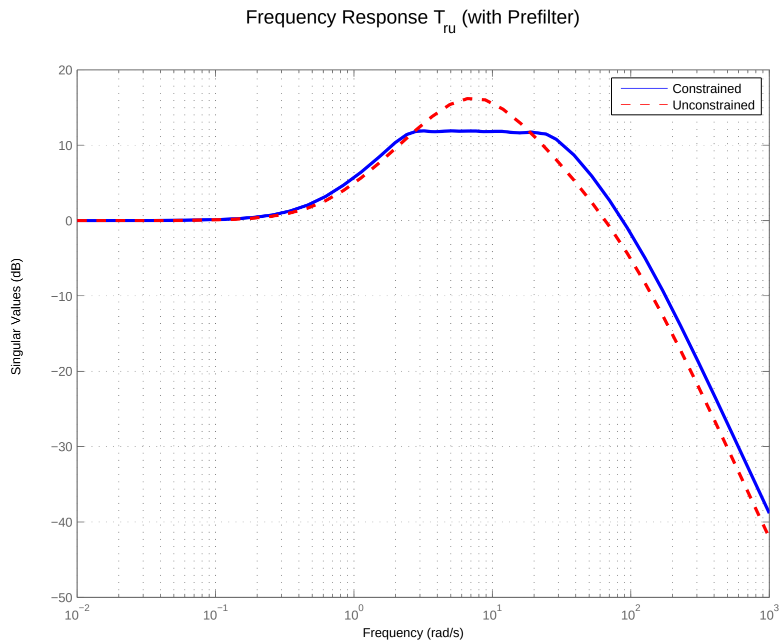


Figure 4.45: Design 2b: Reference to Control transfer function

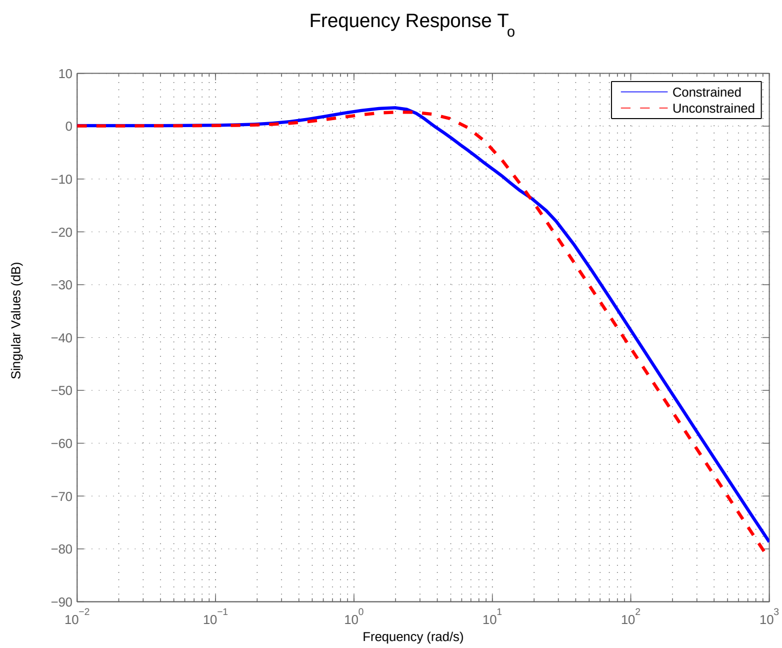


Figure 4.46: Design 2b: Complementary Sensitivity

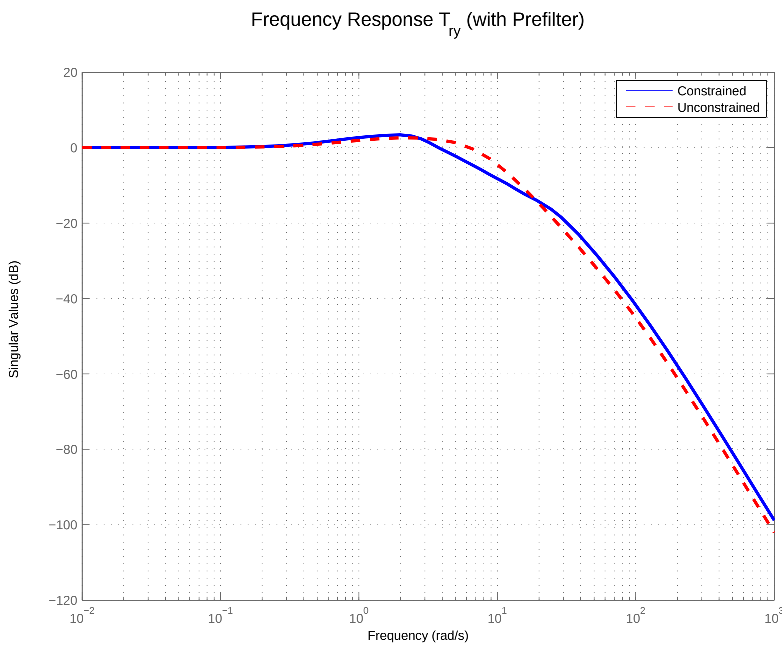


Figure 4.47: Design 2b: Reference to output transfer function

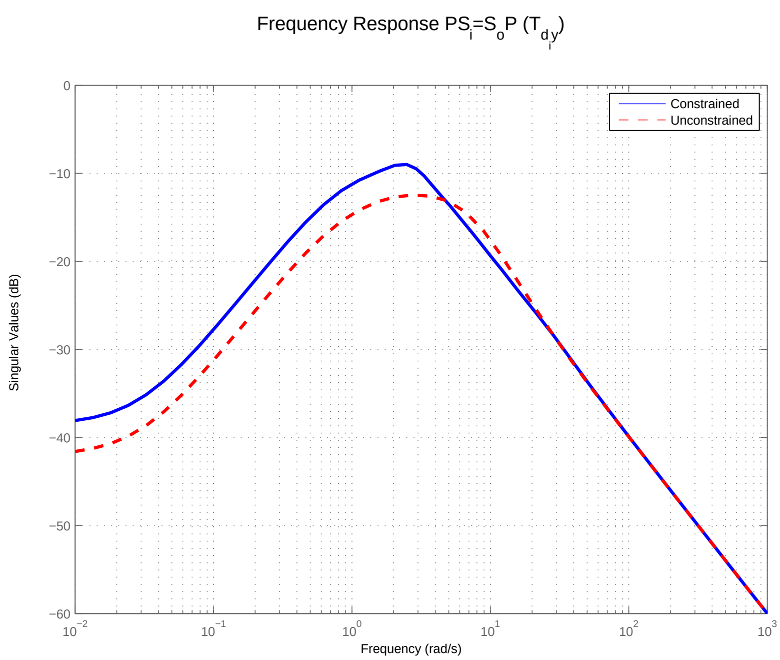


Figure 4.48: Design 1a: $PS_i = S_oP$

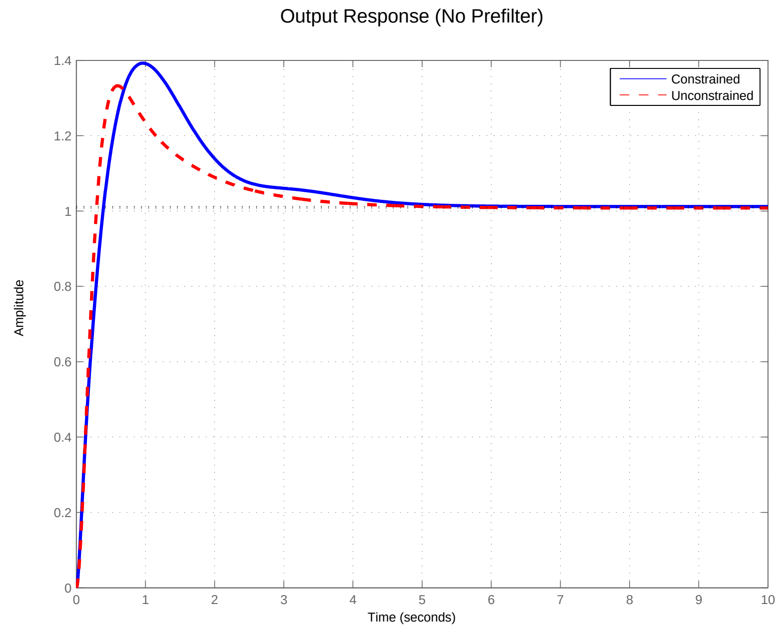


Figure 4.49: Design 2b: Output Time Response (no Pre-filter)

Table 4.16: Design 2b: $\mathcal{H}^\infty$ norms of individual transfer functions (dB)

$S_o = S_i$	$T_o = T_i$	KS_o	PS_i
0.6925	3.4667	12.0087	-9.0707

4.4 Summary and Conclusions

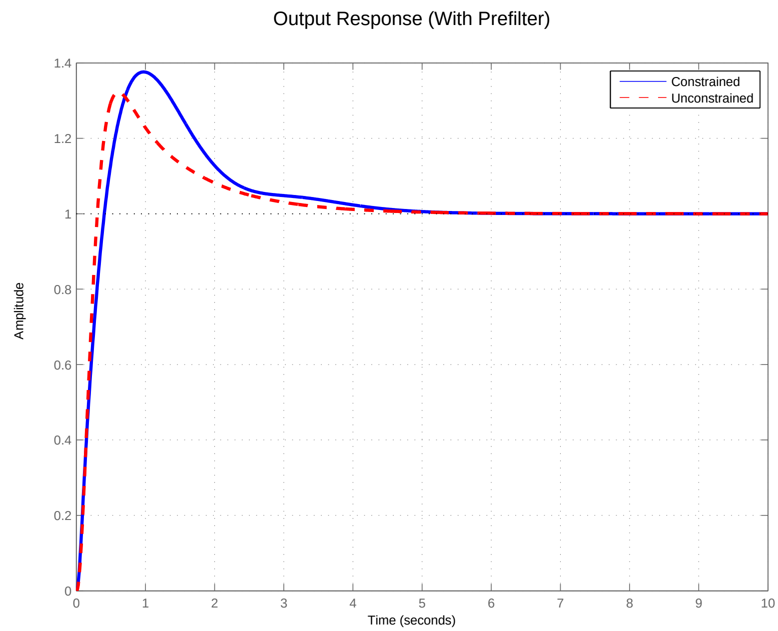


Figure 4.50: Design 2b: Output Time Response (with Pre-filter)

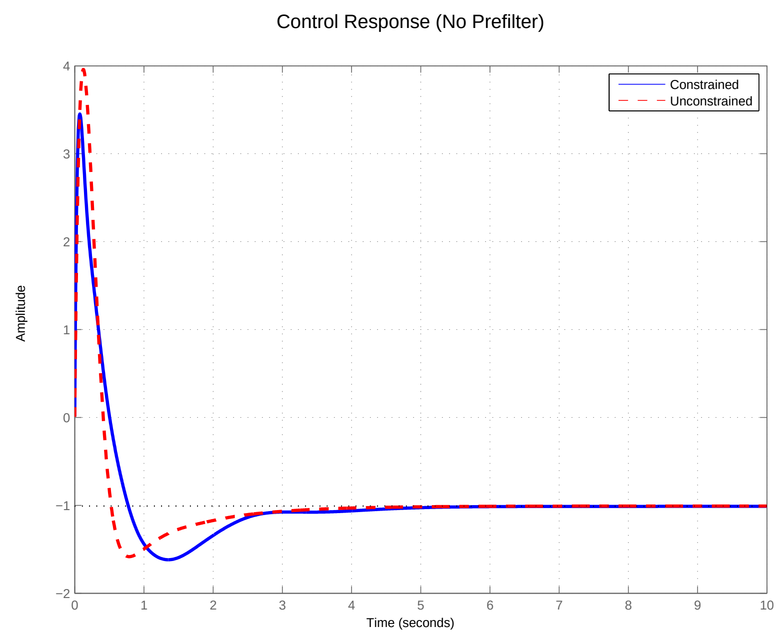


Figure 4.51: Design 2b: Control Time Response (no Pre-filter)

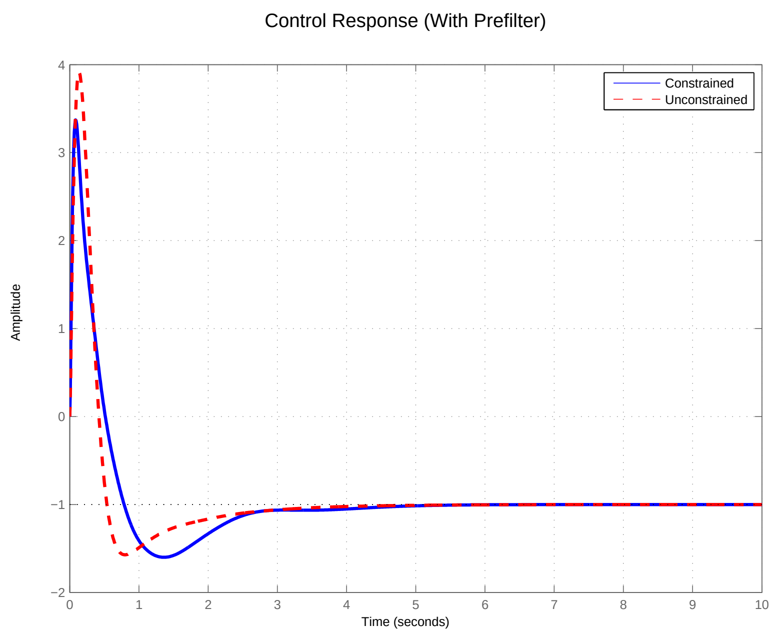


Figure 4.52: Design 2b: Control Time Response (with Pre-filter)

Chapter 5

MIMO $\mathcal{H}^\infty$ DESIGN EXAMPLES

5.1 Introduction

In this chapter feedback compensators are designed for two MIMO systems. In Section 5.2 an ill-conditioned 2X2 system is controlled. In Section 5.3 a compensator for Lateral (yaw-roll) dynamics of a forward swept wing X-29 aircraft [23] is designed. Parameter ρ in Equation 2.9 is varied to trade-off feedback properties at plant output and input.

5.2 Ill-Conditioned Two-Input Two-Output system

The Plant transfer function matrix is shown below:

$$P = \begin{bmatrix} \frac{1}{s+1} & 0 \\ 0 & \frac{1}{s+2} \end{bmatrix} \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} \tag{5.1}$$

$$\tag{5.2}$$

The scaling matrix has near zero determinant value (0.1900) makes the plant P to have high condition number as seen in Figure 5.2. We saw that from Equation 2.8, for plants with high condition number, feedback properties could be drastically different at loop breaking points.

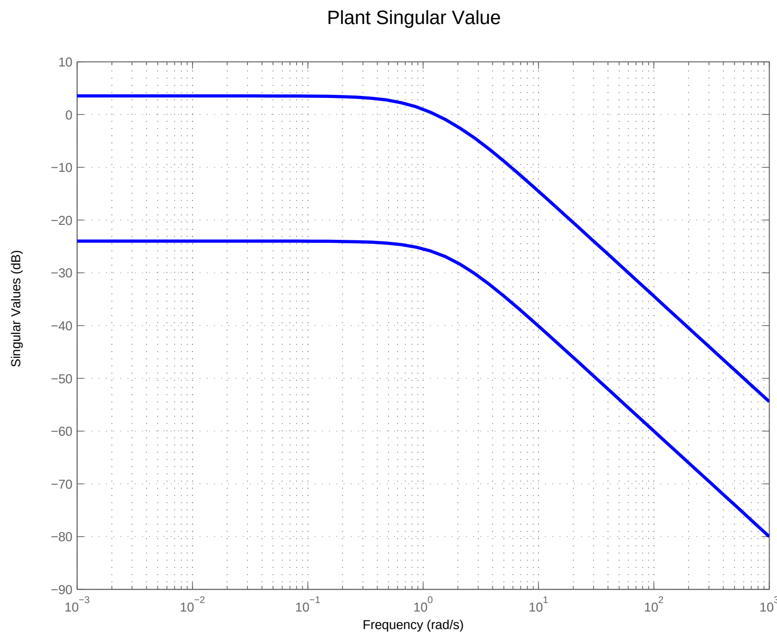


Figure 5.1: Plant Singular values

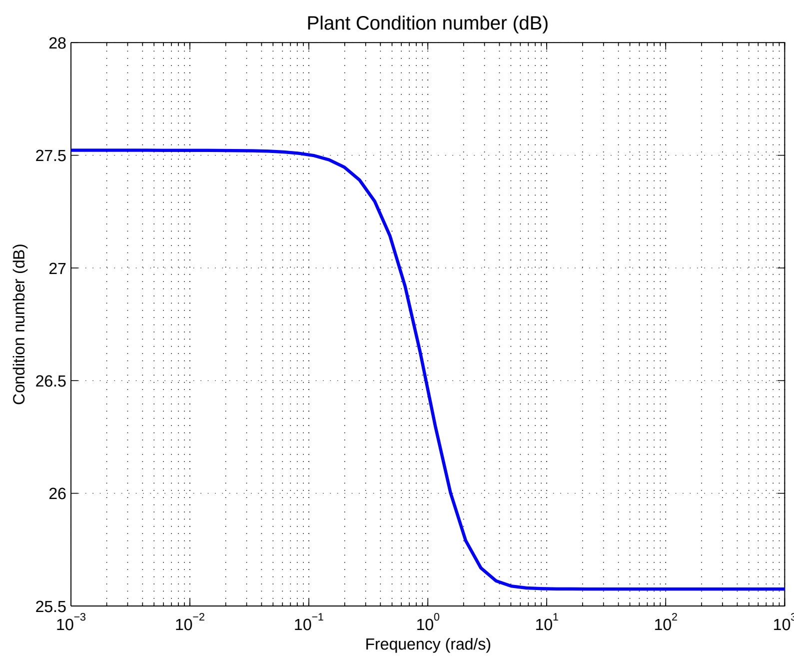


Figure 5.2: Plant Condition number

5.2.1 $\rho = 10^{-6}$ *(Approximation to standard mixed sensitivity problem)*

Basis parameters used:

$$Basis = \frac{0.7 - s}{s + 0.7} \qquad\qquad N = 4 \tag{5.3}$$

By choosing a near zero value for design parameter $rho = 10^{-6}$, the generalized $\mathcal{H}^\infty$ mixed sensitivity problem is approximated to a standard $\mathcal{H}^\infty$ mixed sensitivity problem. It is able to achieve good properties at plant output, while giving up on properties at plant input. The Output Sensitivity S_o represents transfer function from reference r to error e, while Input Sensitivity S_i represents the transfer function from input disturbance d_i to plant input u_p. Both S_o and S_i are desired to be low, as they represent good command following and good disturbance rejection respectively. By comparing Figure 5.9 and Figure 5.10, it is seen that for the current design, a good low frequency command following is achieved in all input directions, whereas low frequency input disturbance attenuation depends on the direction of input disturbance. If good low frequency input disturbance attenuation is desired, the feedback properties at plant input are penalized in the Generalized $\mathcal{H}^\infty$ problem.

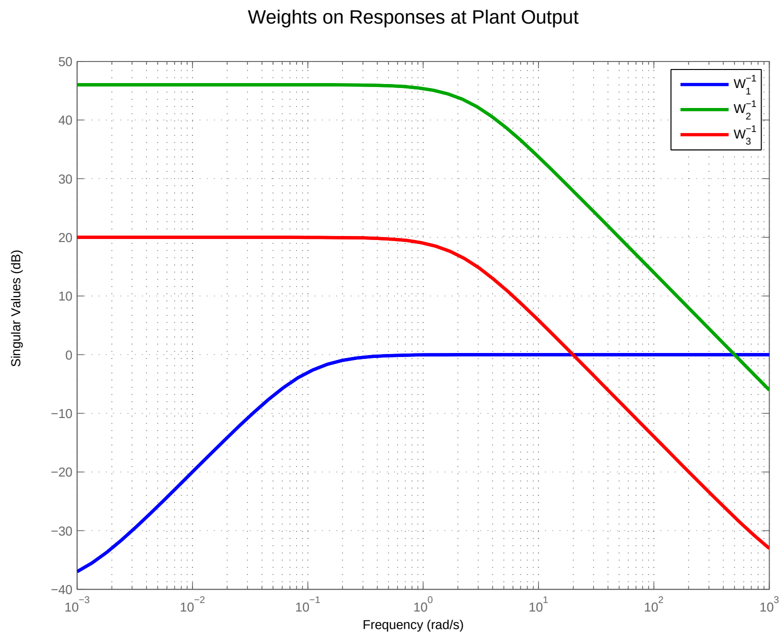


Figure 5.3: Weighting functions output due to reference command

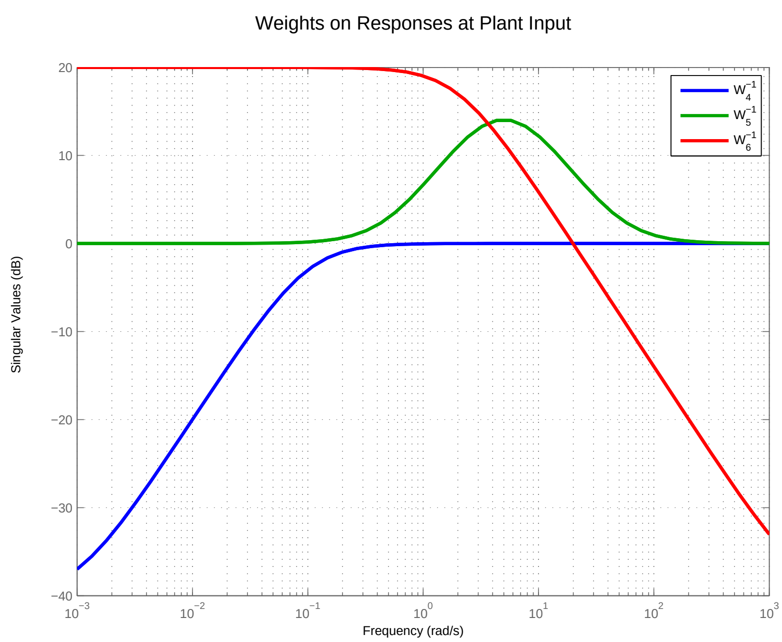


Figure 5.4: Weighting functions on output due to disturbance

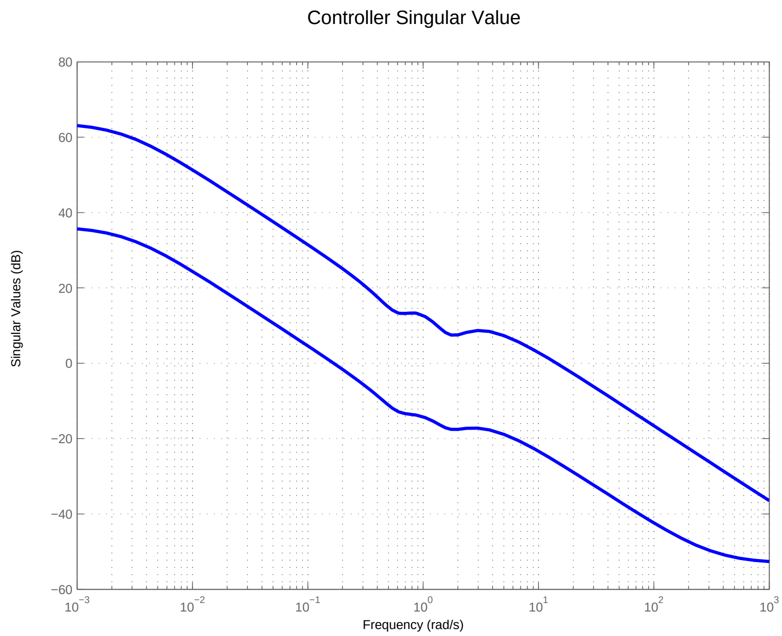


Figure 5.5: Controller Singular value

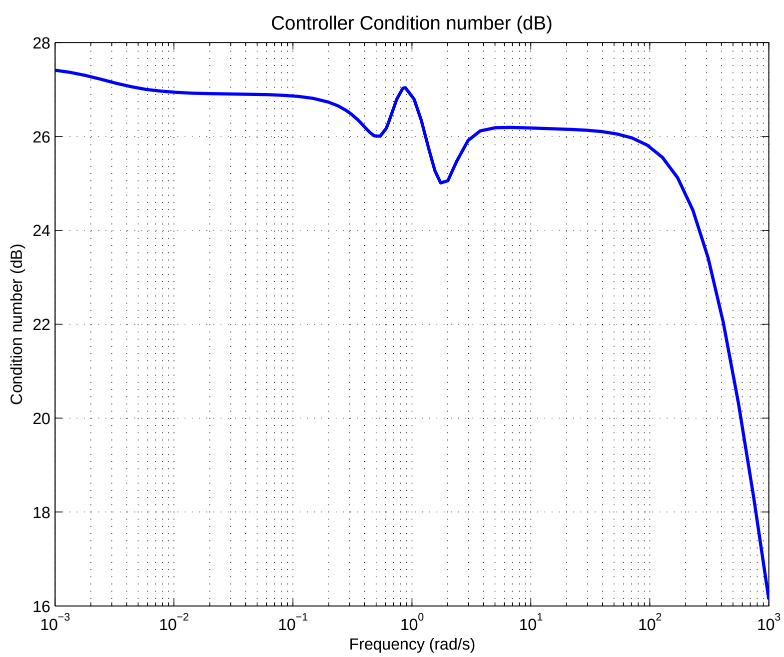


Figure 5.6: Controller Condition number

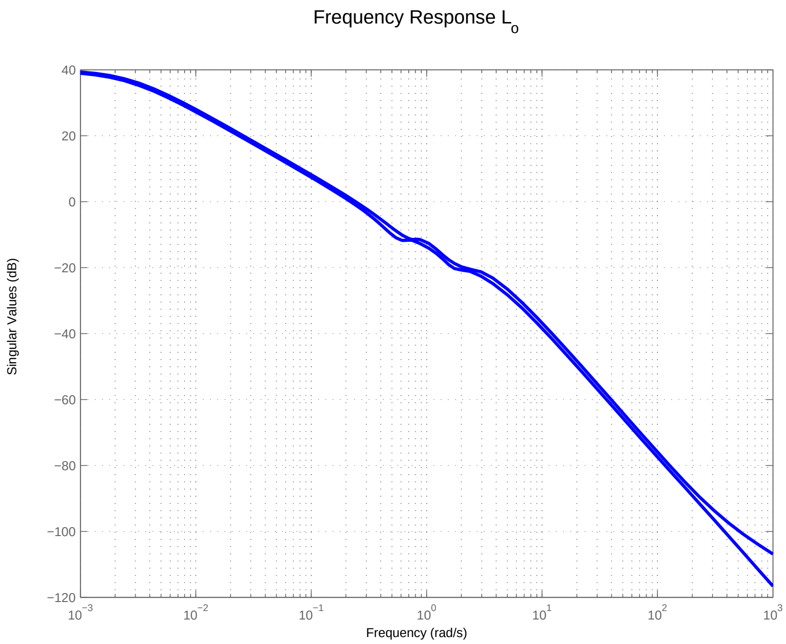


Figure 5.7: Open Loop transfer function at Plant output

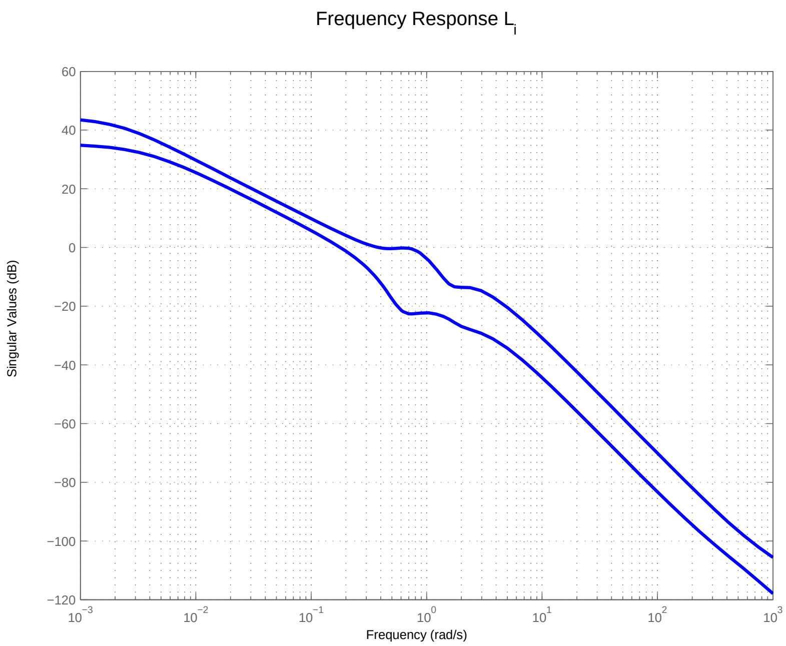


Figure 5.8: Open Loop transfer function at Plant input

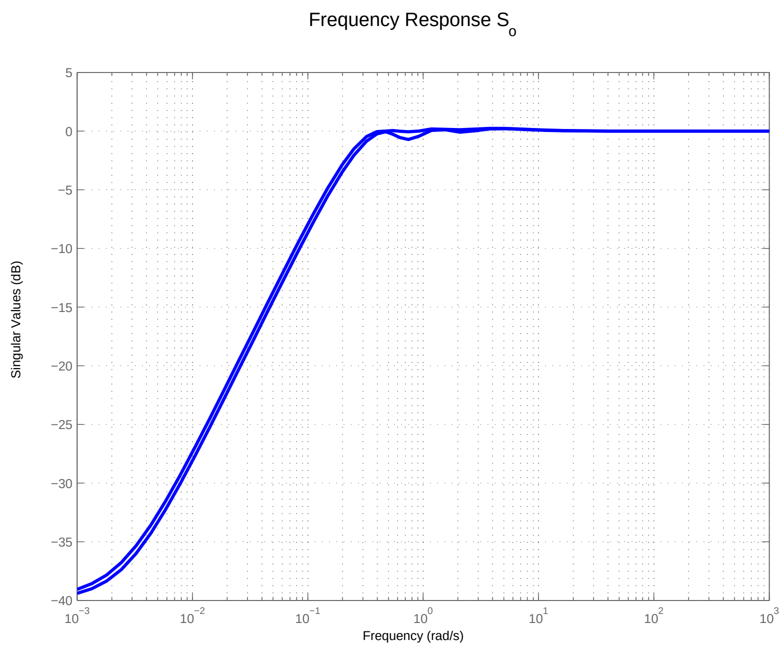


Figure 5.9: Output Sensitivity

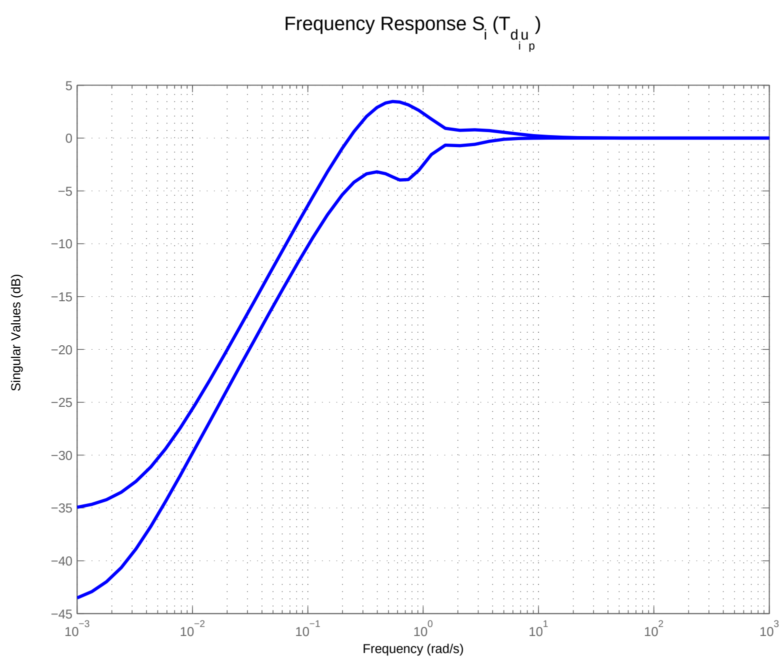


Figure 5.10: Input Sensitivity

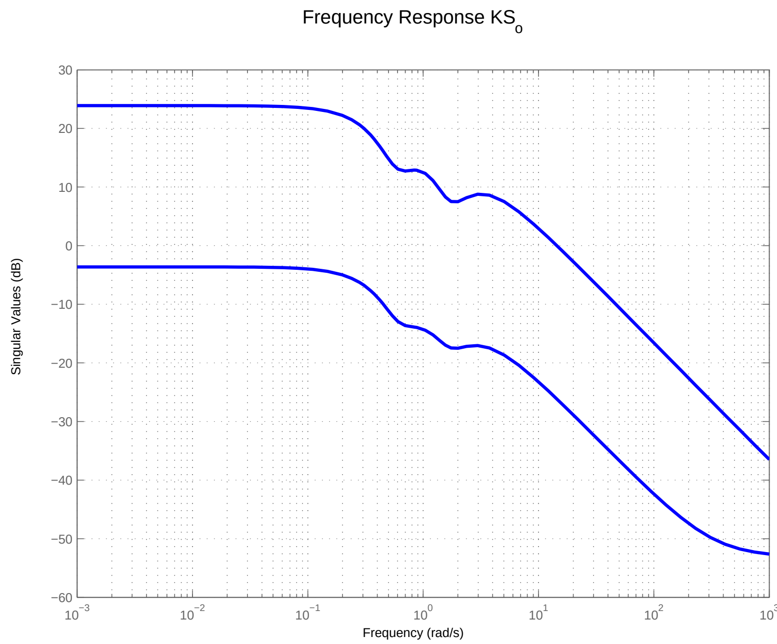


Figure 5.11: K*So

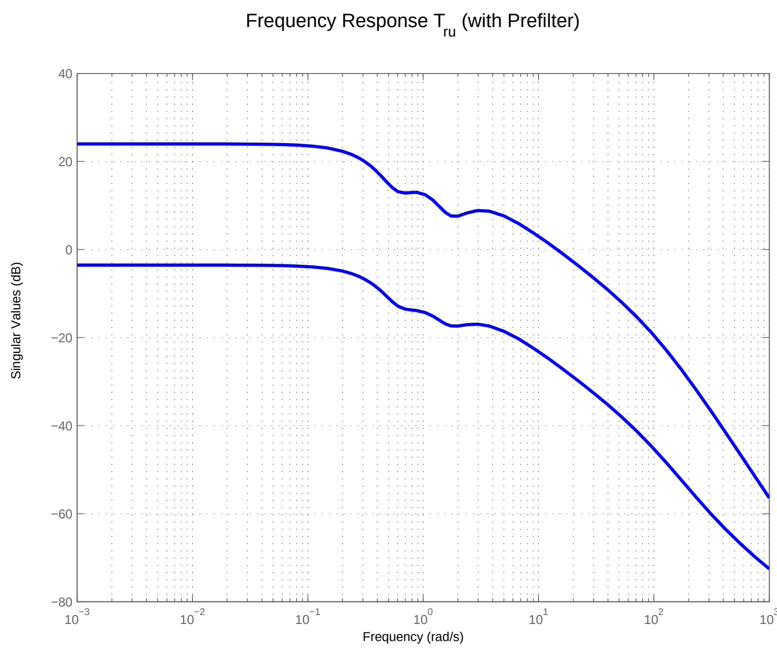


Figure 5.12: Reference to Control transfer function

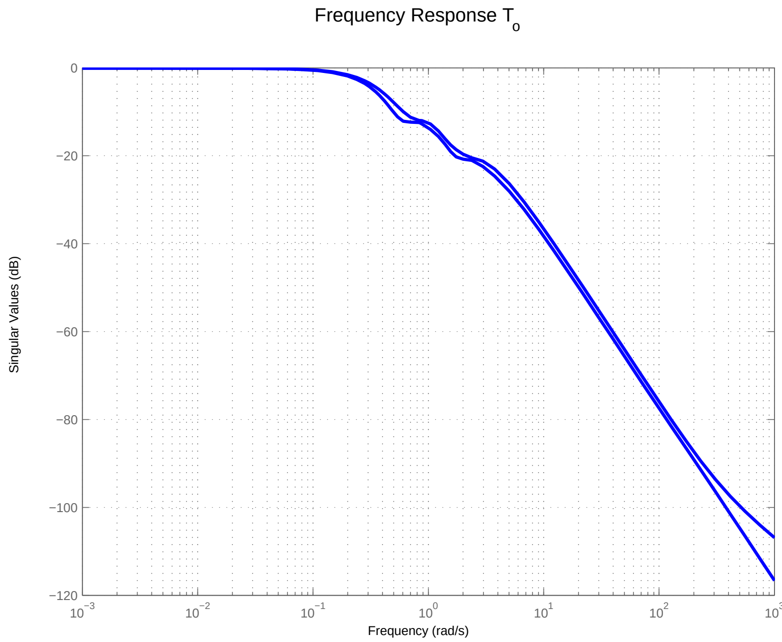


Figure 5.13: Output Complementary Sensitivity

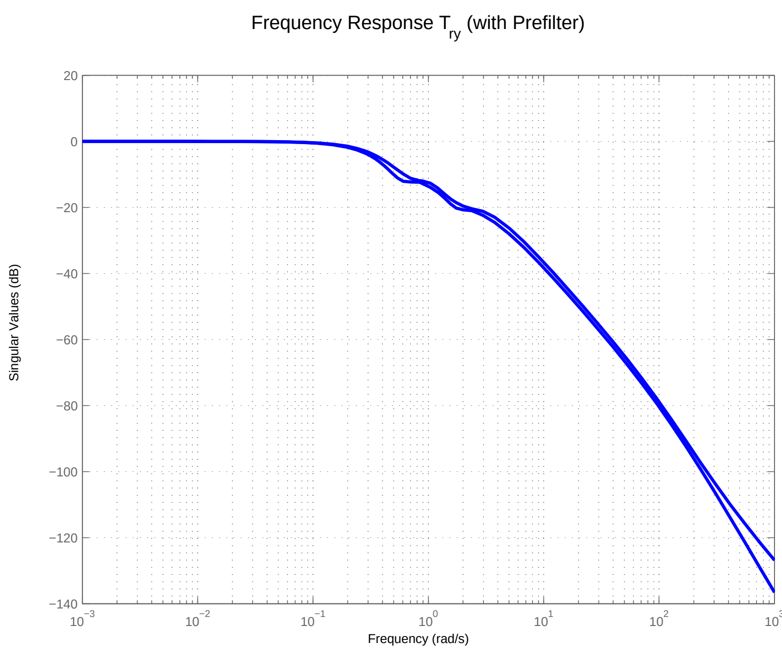


Figure 5.14: Reference to output transfer function

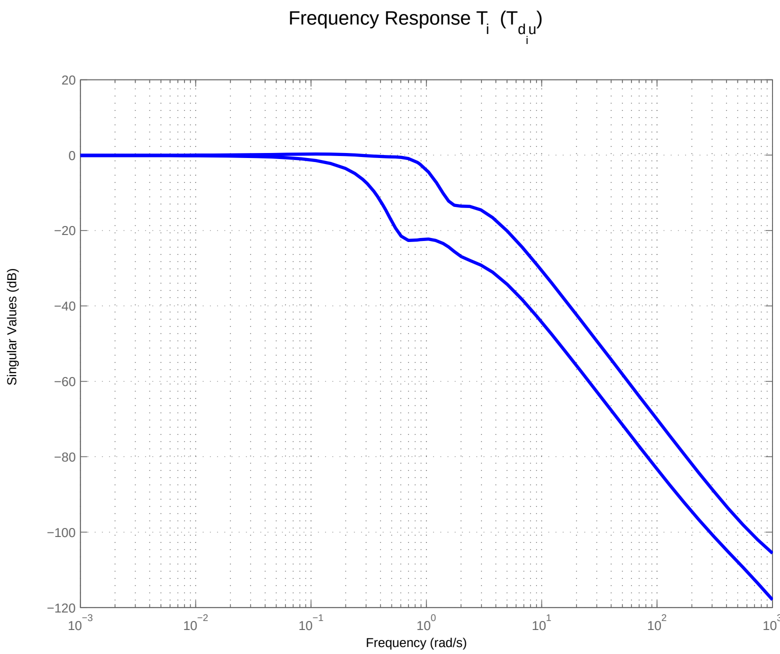


Figure 5.15: Input Complementary Sensitivity

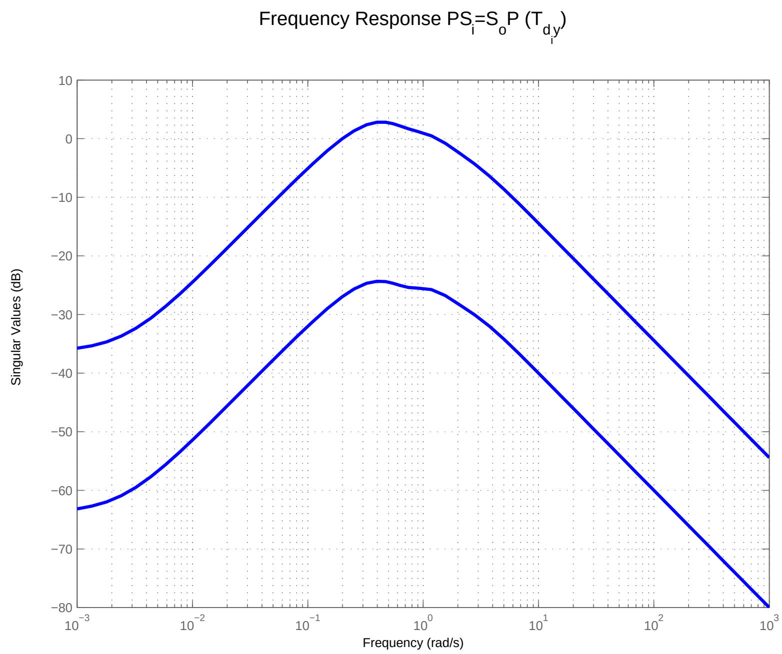


Figure 5.16: $PS_i = S_oP$

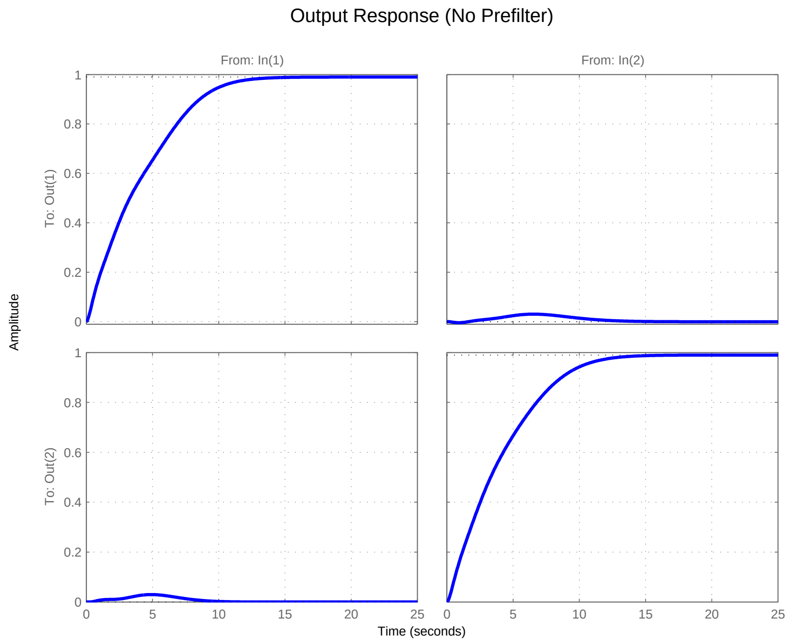


Figure 5.17: Output Time Response (no Pre-filter)

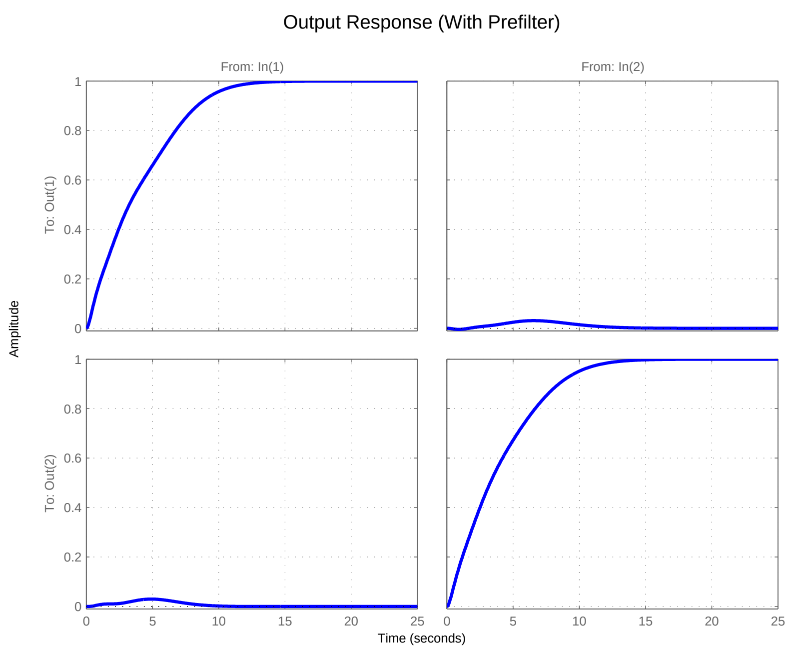


Figure 5.18: Output Time Response (with Pre-filter)

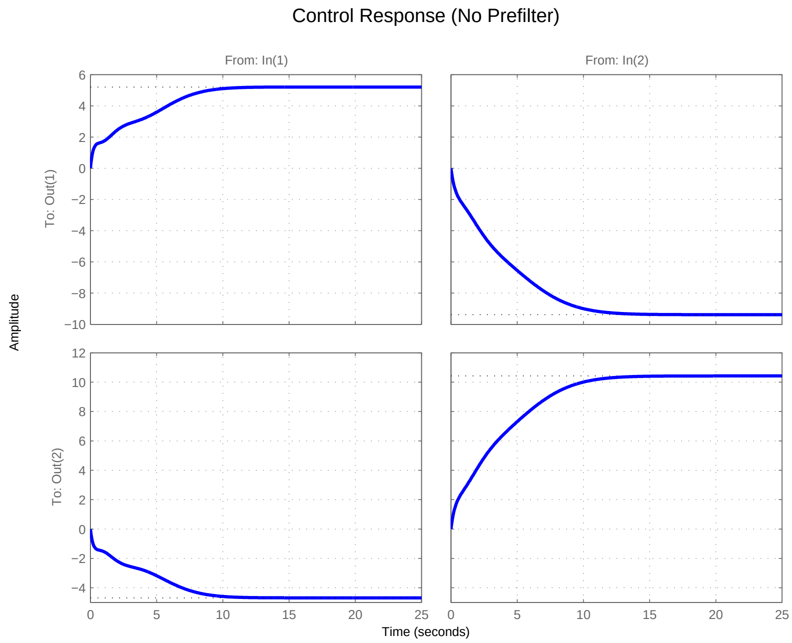


Figure 5.19: Control Time Response (no Pre-filter)

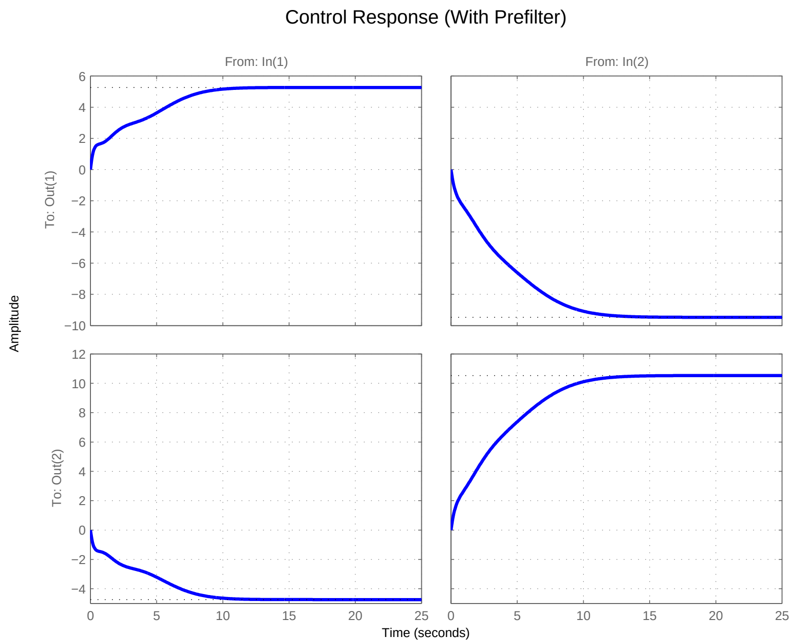


Figure 5.20: Control Time Response (with Pre-filter)

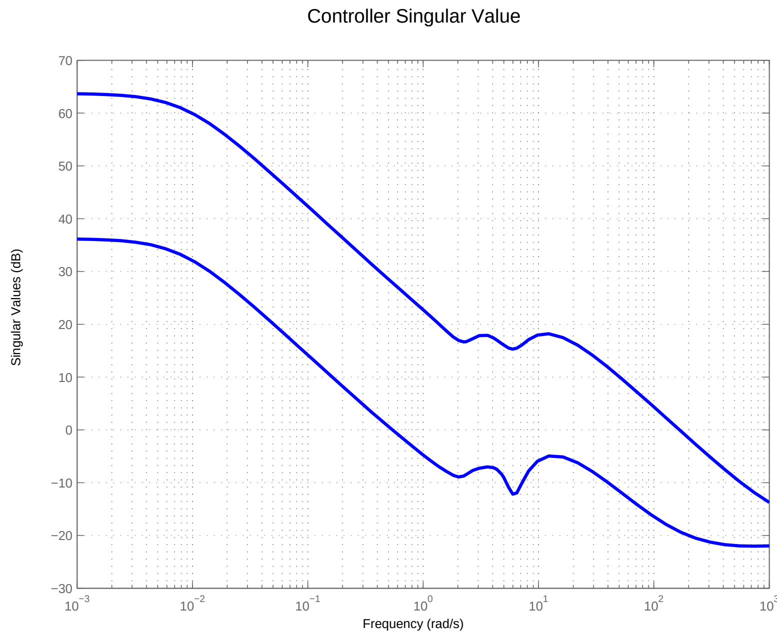


Figure 5.21: Controller Singular value

5.2.2 $\rho = 10$ *(Penalizing Properties at Plant Input)*

To achive good low frequency input disturbance attenuation, ρ is increased to 10. Basis parameters used:

$$Basis = \frac{4-s}{s+4} \qquad\qquad N = 7 \tag{5.4}$$

Figure 5.25 shows that good low frequency input disturbance attenuation is achieved for all directions. But this trades-off good command following property.

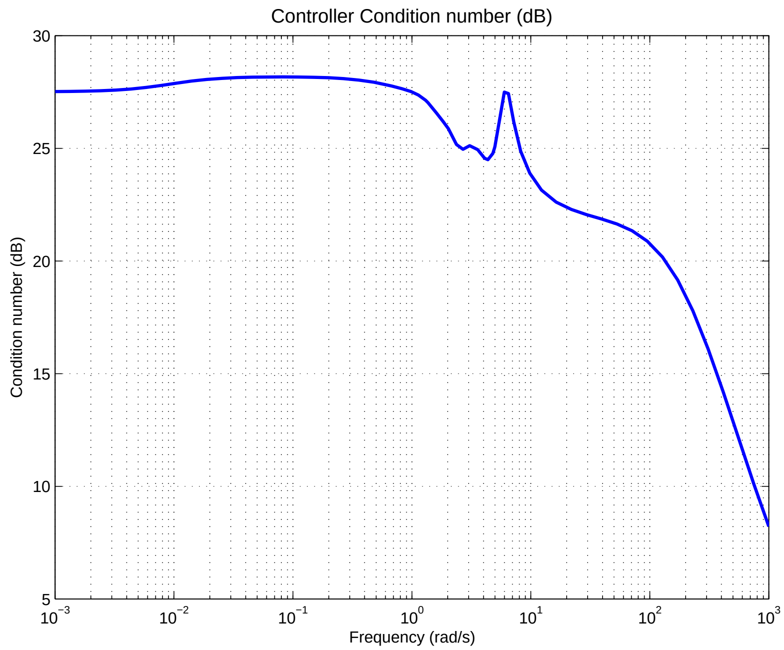


Figure 5.22: Controller Condition number

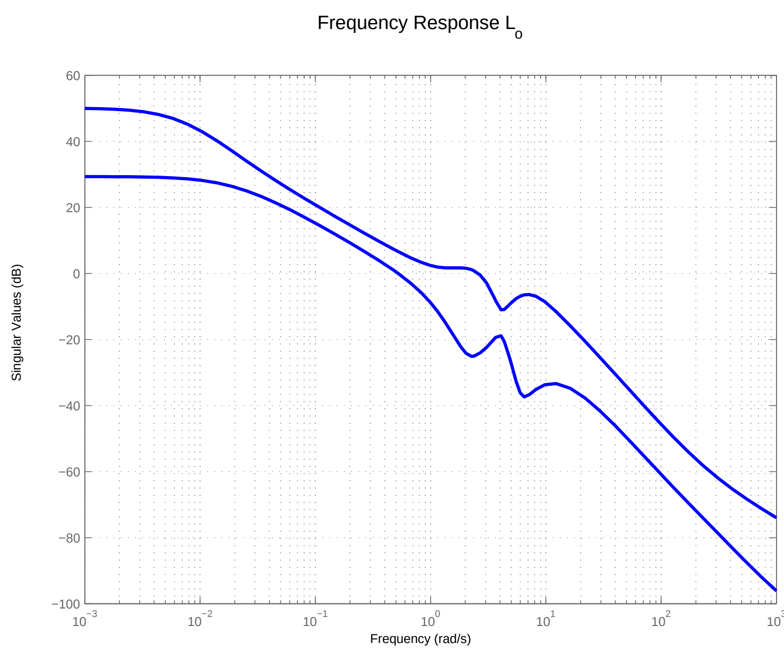


Figure 5.23: Open Loop transfer function at Plant output

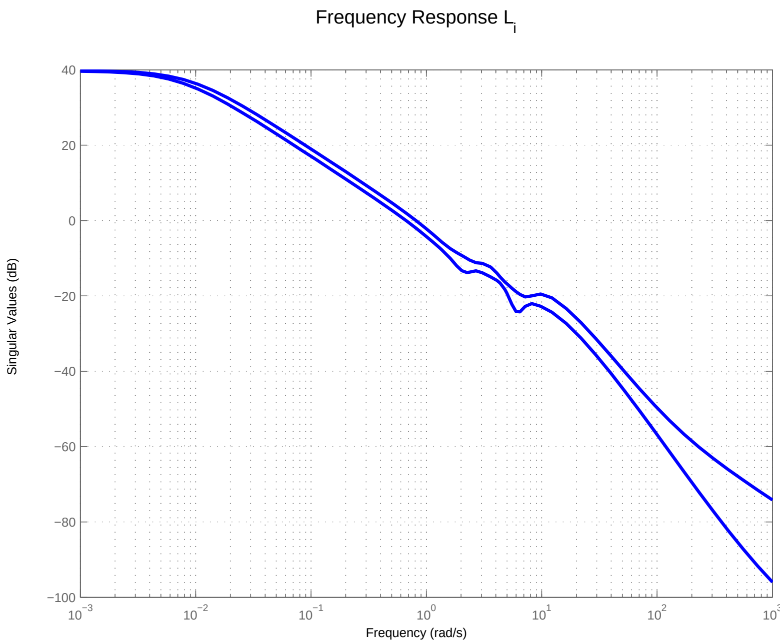


Figure 5.24: Open Loop transfer function at Plant input

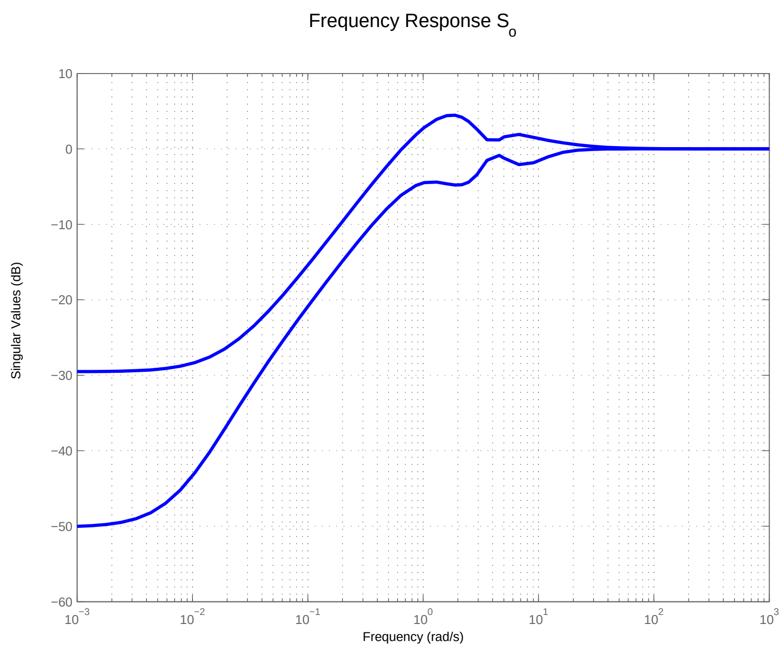


Figure 5.25: Output Sensitivity

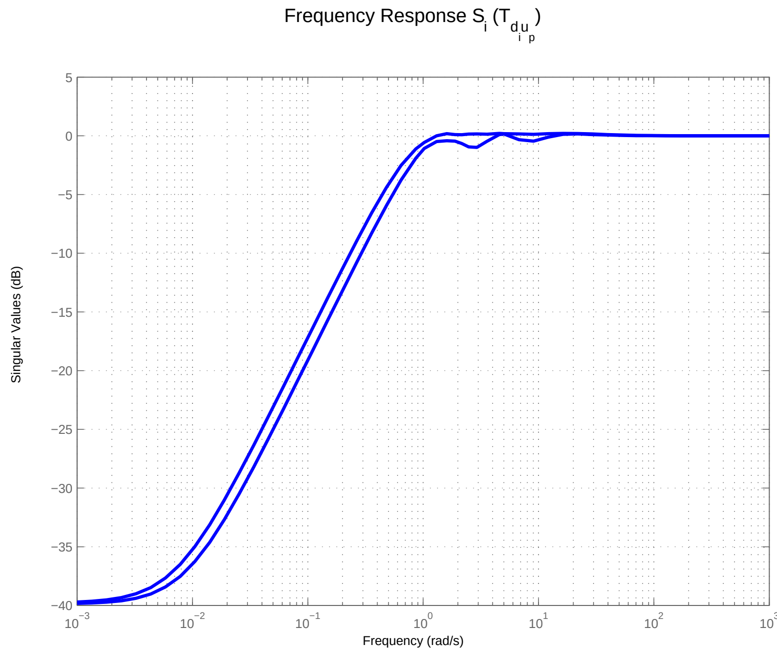


Figure 5.26: Input Sensitivity

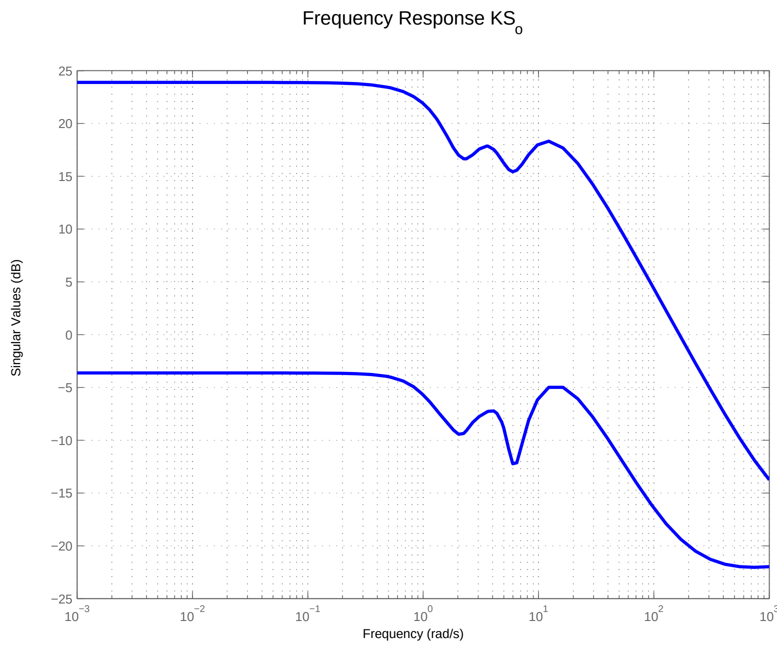


Figure 5.27: K*So

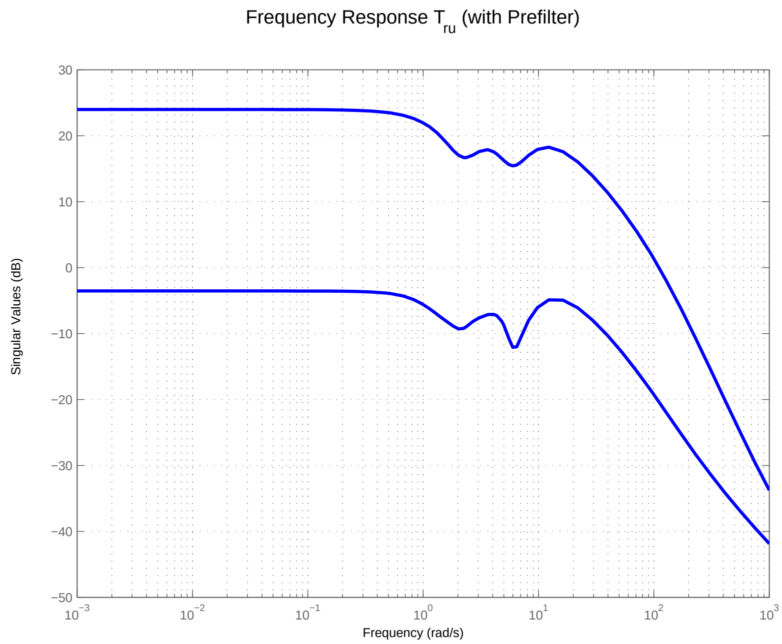


Figure 5.28: Reference to Control transfer function

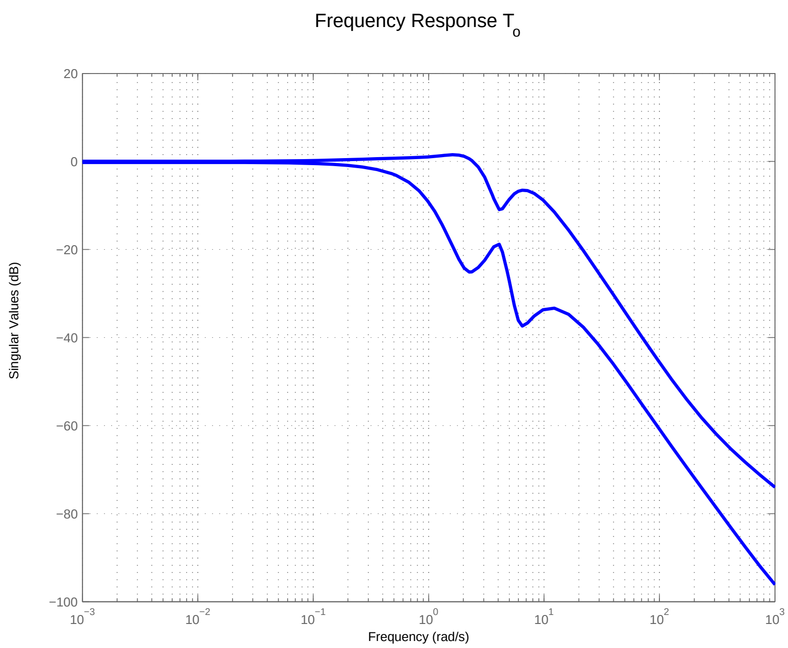


Figure 5.29: Output Complementary Sensitivity

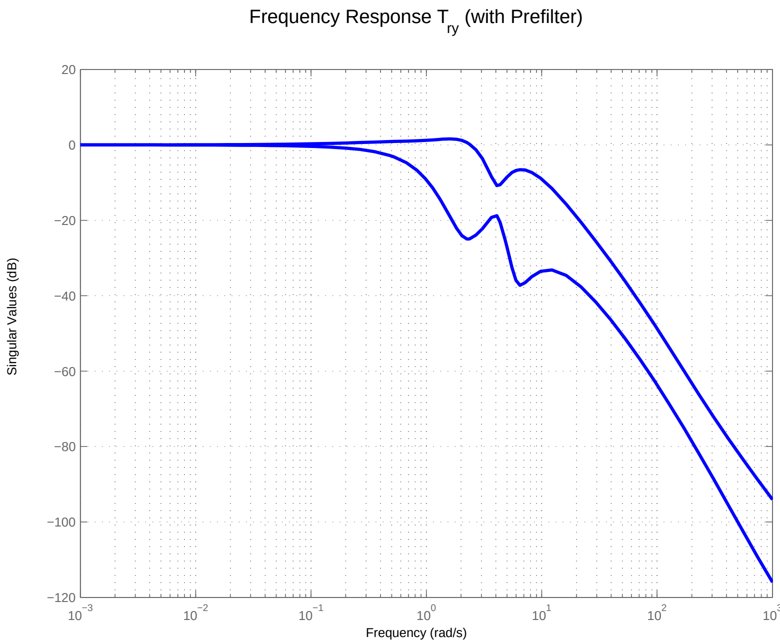


Figure 5.30: Reference to output transfer function

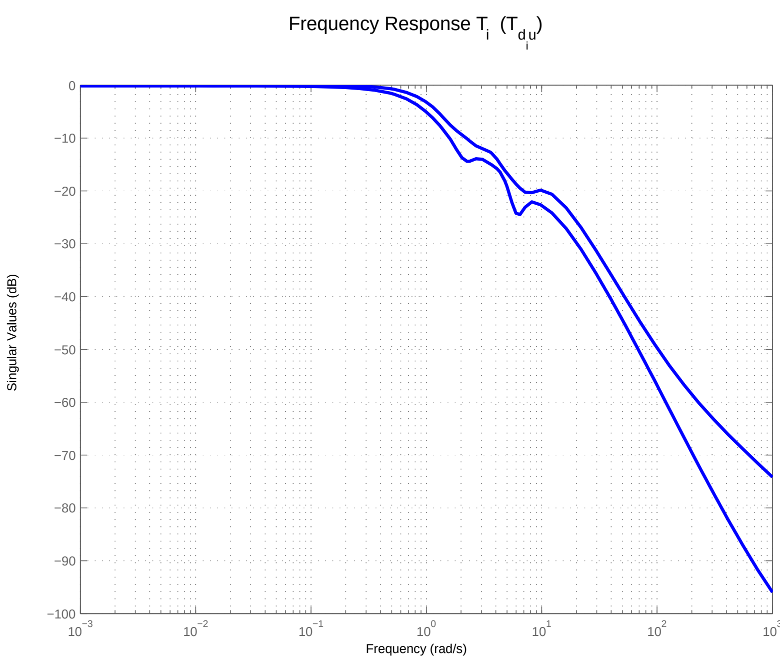


Figure 5.31: Input Complementary Sensitivity

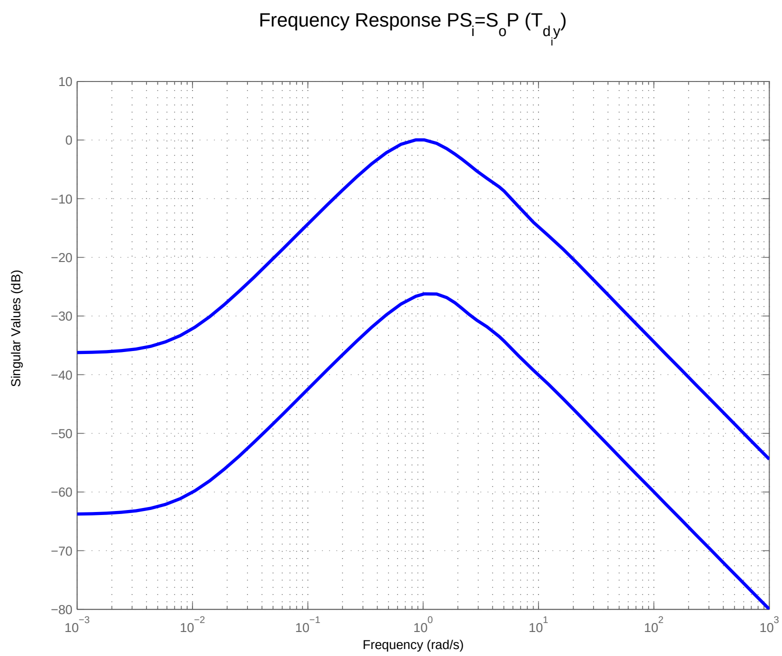


Figure 5.32: $PS_i = S_oP$

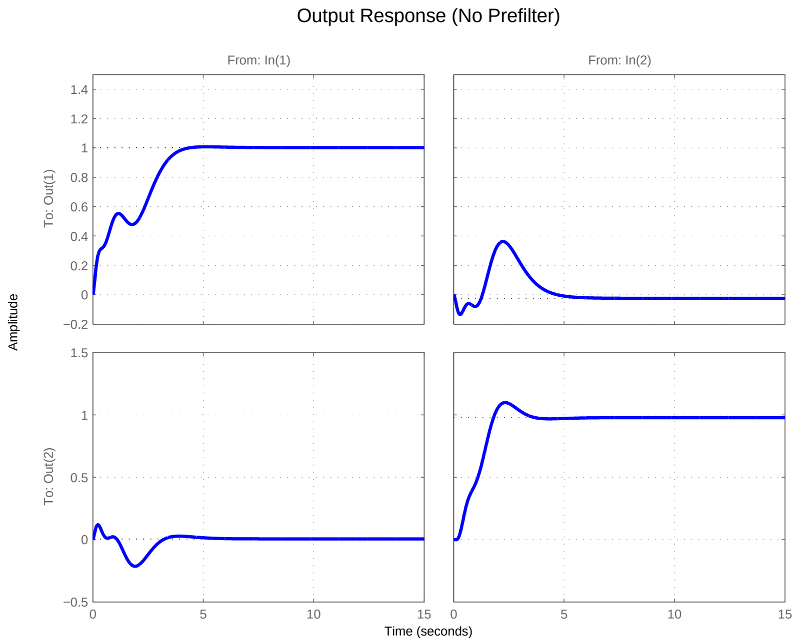


Figure 5.33: Output Time Response (no Pre-filter)

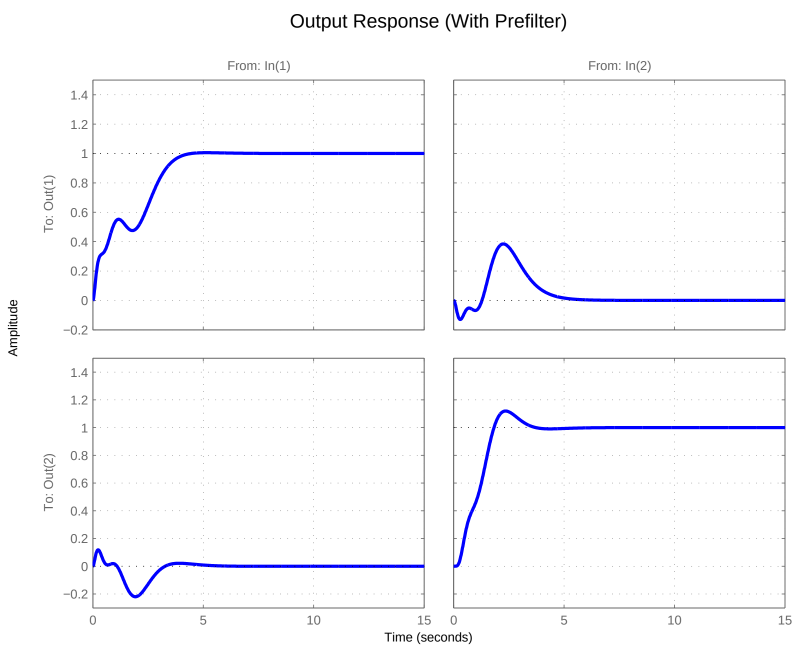


Figure 5.34: Output Time Response (with Pre-filter)

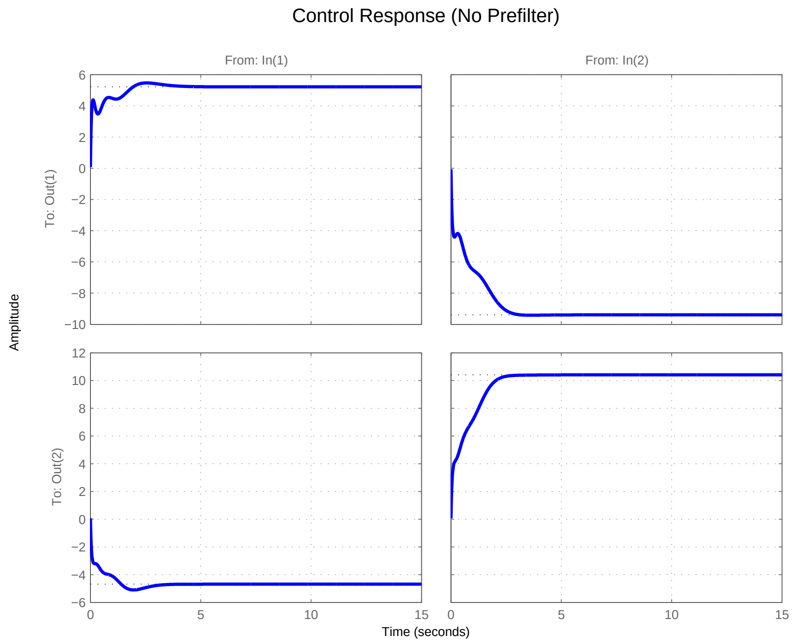


Figure 5.35: Control Time Response (no Pre-filter)

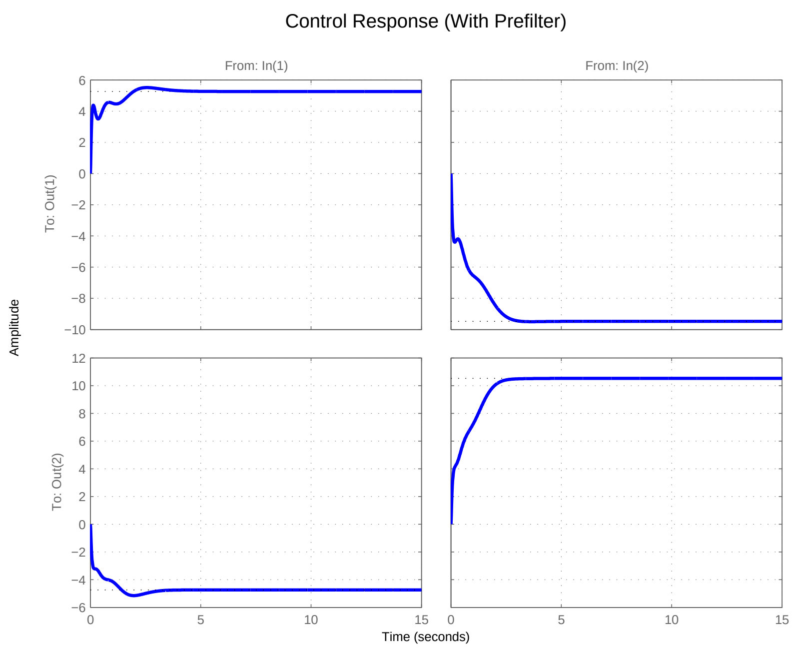


Figure 5.36: Control Time Response (with Pre-filter)

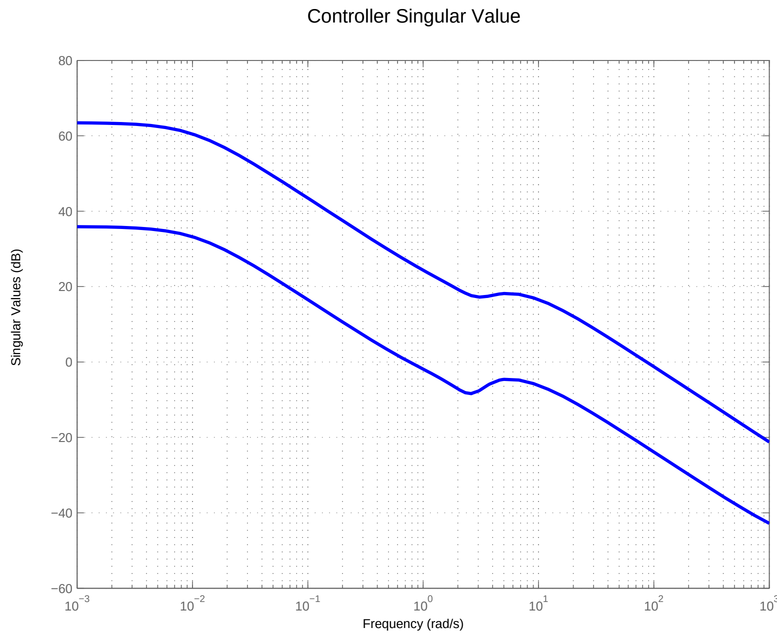


Figure 5.37: Controller Singular value

5.2.3 $\rho = 1$ (Trade-off between Properties and Plant Input and Output)

To achieve comparable low frequency command following and comparable low frequency input disturbance attenuation, ρ set to 1. From Figure **??** and Figure 5.42, a good trade-off which achieves reasonable properties at both plant input and output is achieved. Basis parameters used:

$$Basis = \frac{3-s}{s+3} \qquad\qquad N = 5 \tag{5.5}$$

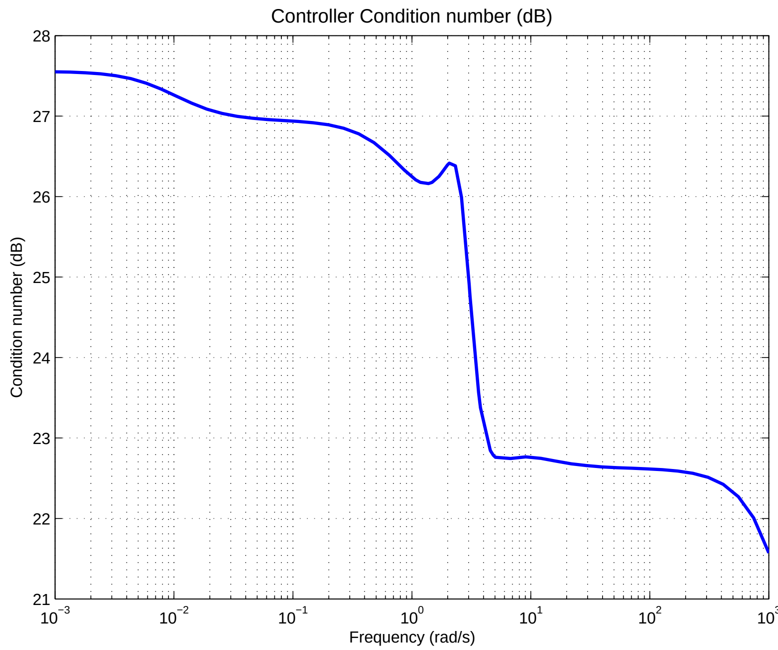


Figure 5.38: Controller Condition number

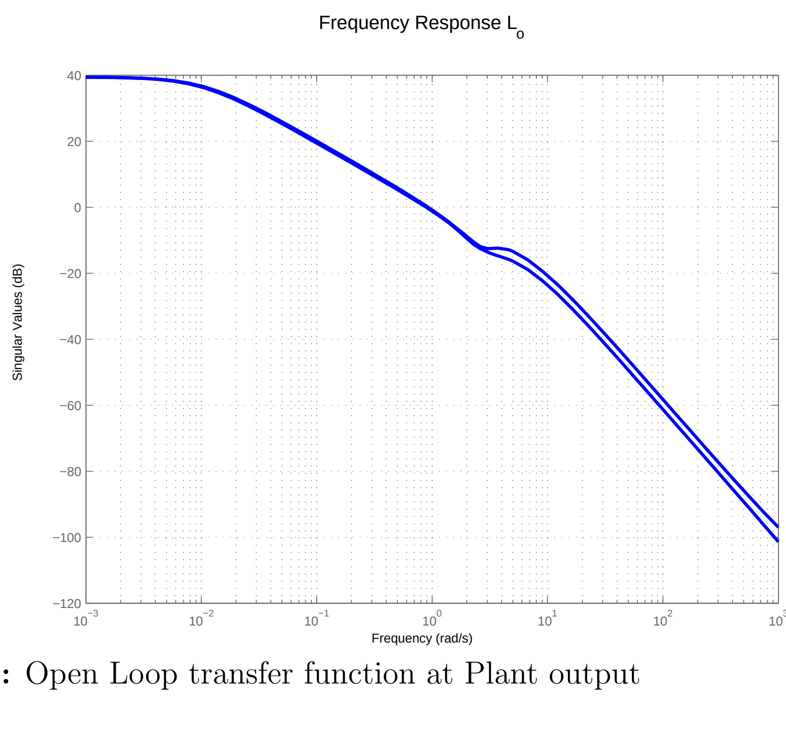


Figure 5.39: Open Loop transfer function at Plant output

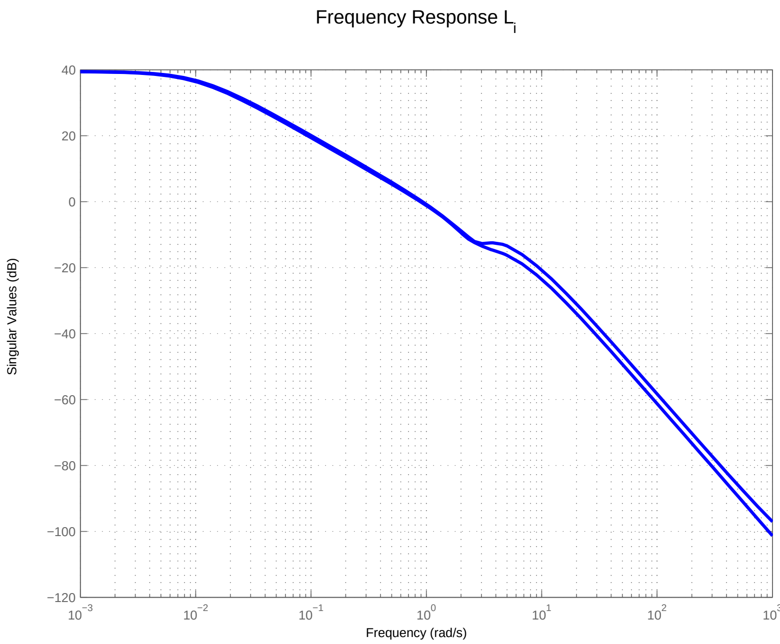


Figure 5.40: Open Loop transfer function at Plant input

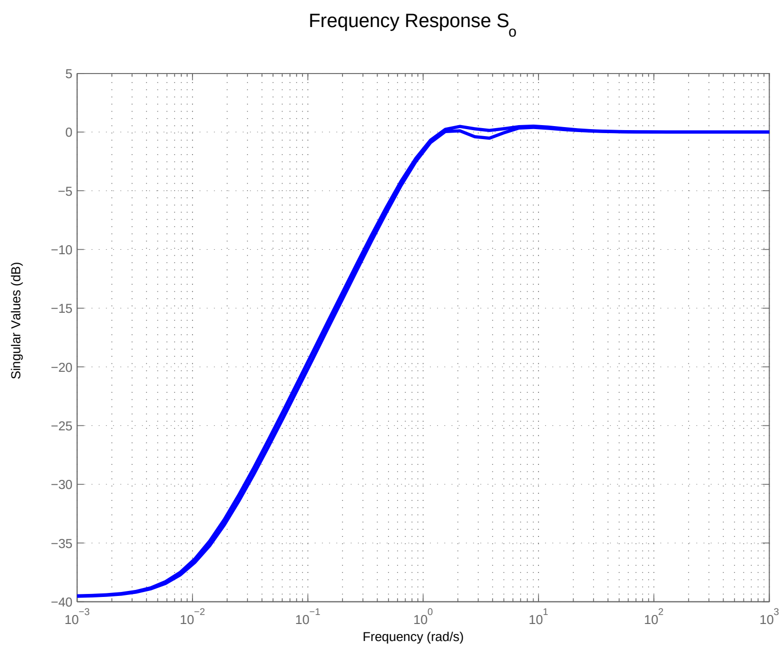


Figure 5.41: Output Sensitivity

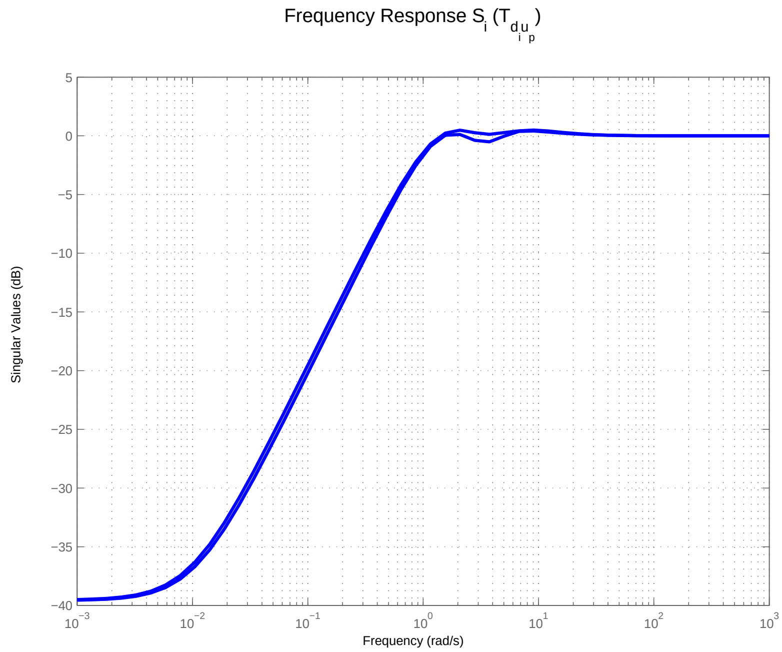


Figure 5.42: Input Sensitivity

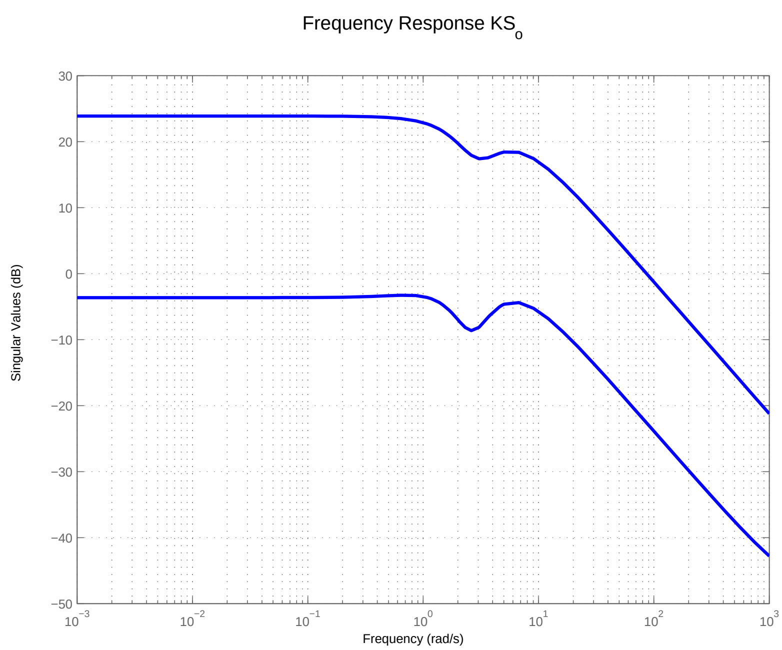


Figure 5.43: K*So

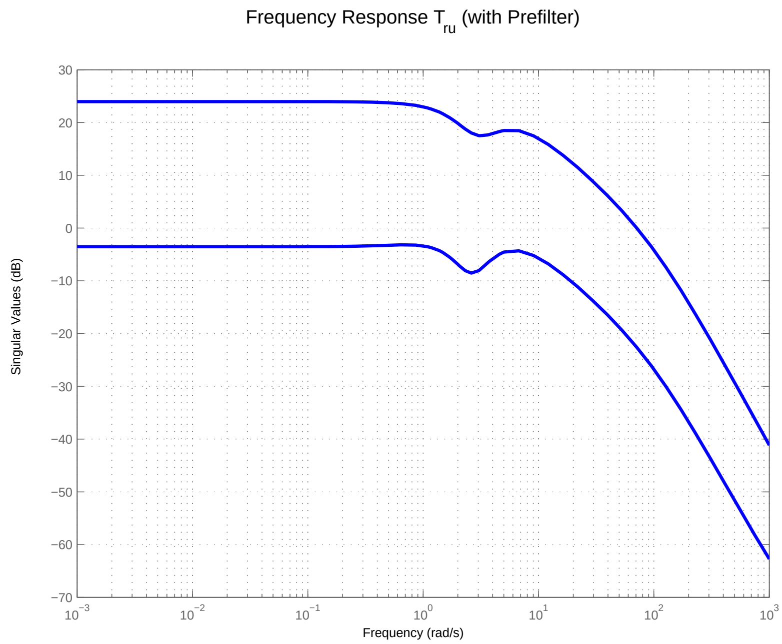


Figure 5.44: Reference to Control transfer function

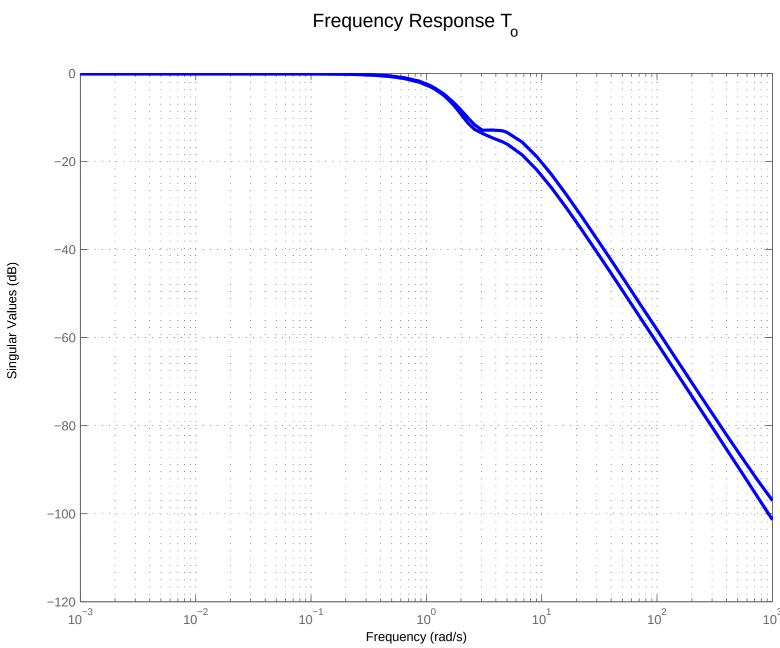


Figure 5.45: Output Complementary Sensitivity

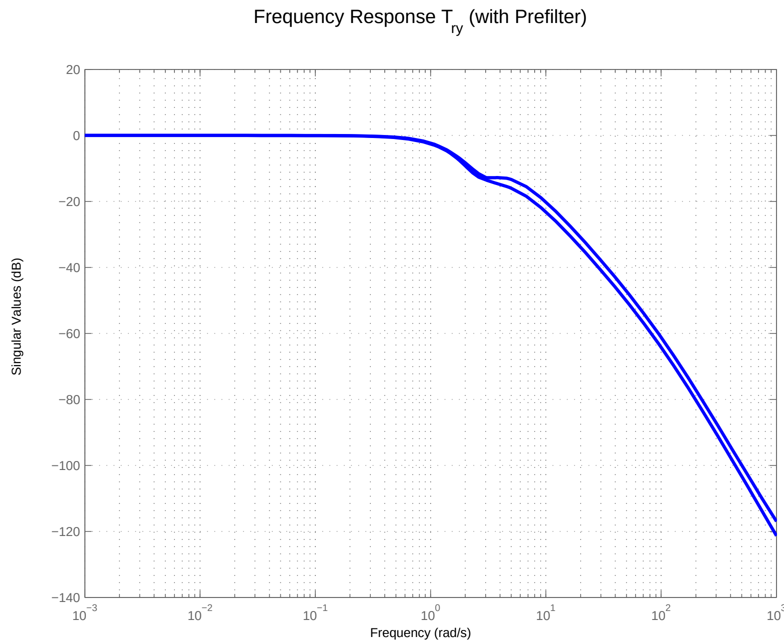


Figure 5.46: Reference to output transfer function

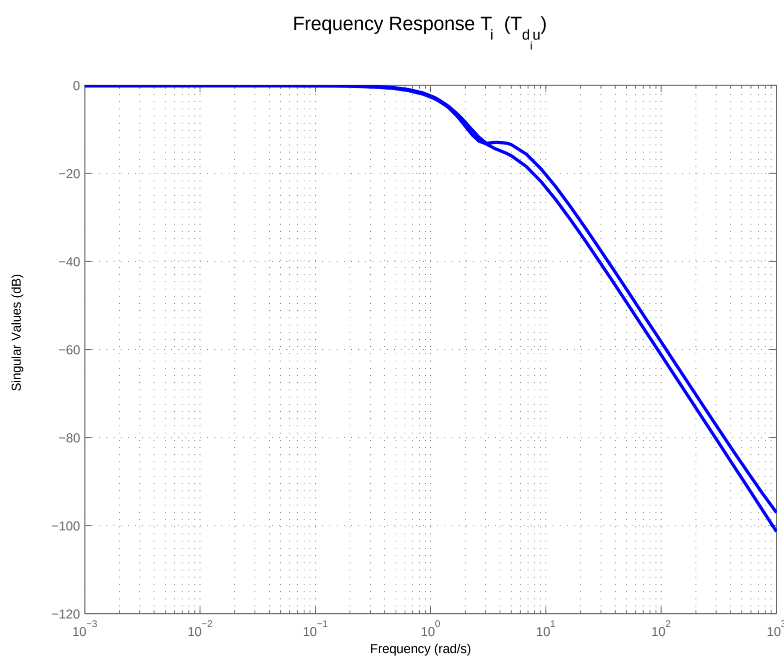


Figure 5.47: Input Complementary Sensitivity

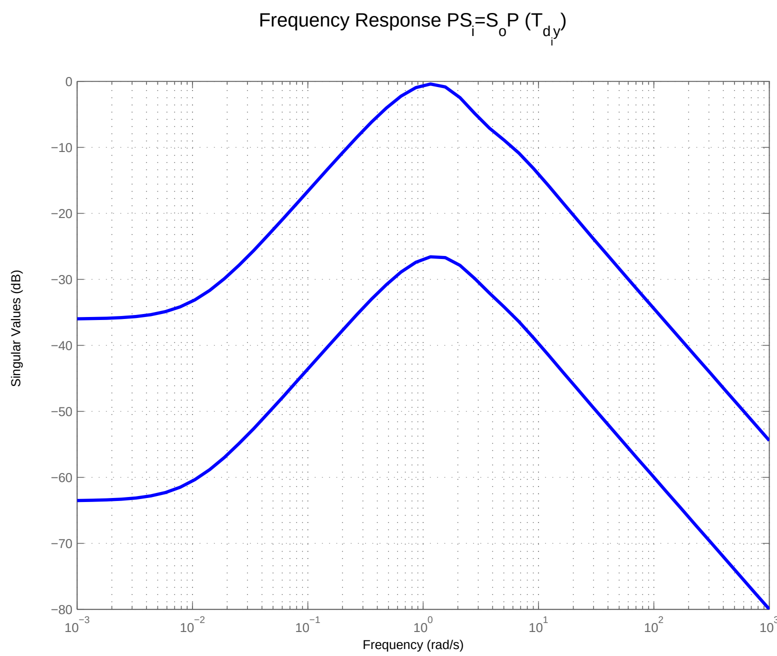


Figure 5.48: $PS_i = S_oP$

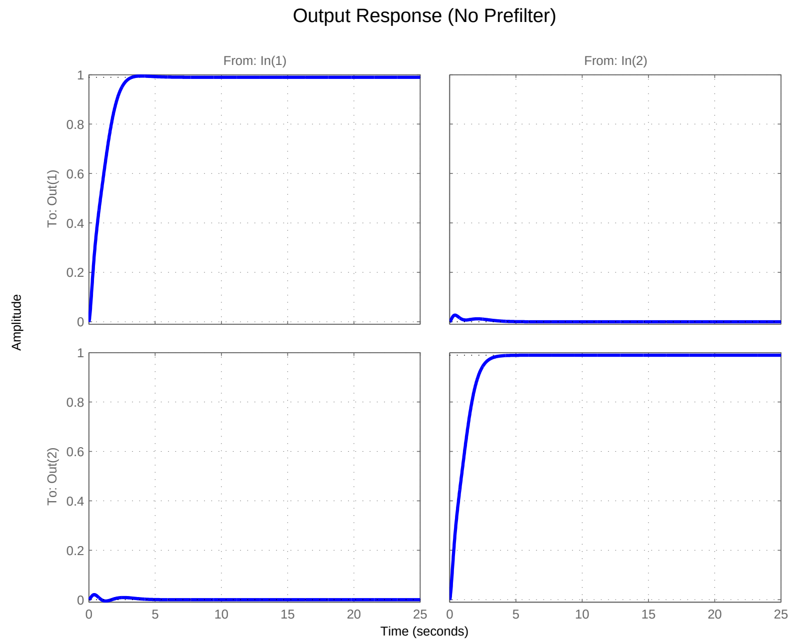


Figure 5.49: Output Time Response (no Pre-filter)

The Table 5.1 shows the $\mathcal{H}^{\infty}$ norms of individual transfer functions matrices.

Table 5.1: 2X2 stable coupled plant: Comparison of Design Results (dB)

ρ	S_o	S_i	KS_o	PS_i	T_o	T_i
10^{-6}	0.0285	0.0699	23.8968	2.7872	-0.0869	-0.0860
10	4.4799	0.1687	23.8954	0.0677	1.5139	-0.0882
1	0.4627	0.4577	23.8921	-0.3822	-0.0916	-0.0916

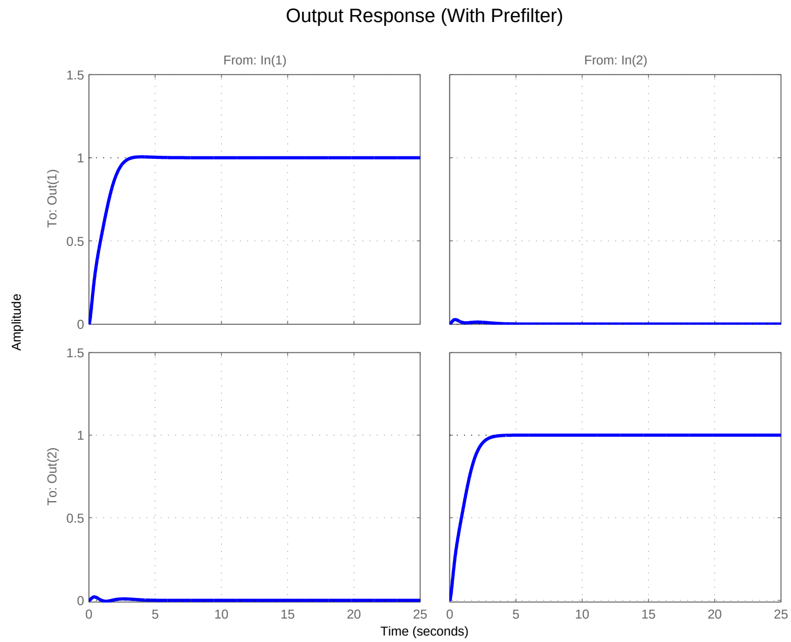


Figure 5.50: Output Time Response (with Pre-filter)

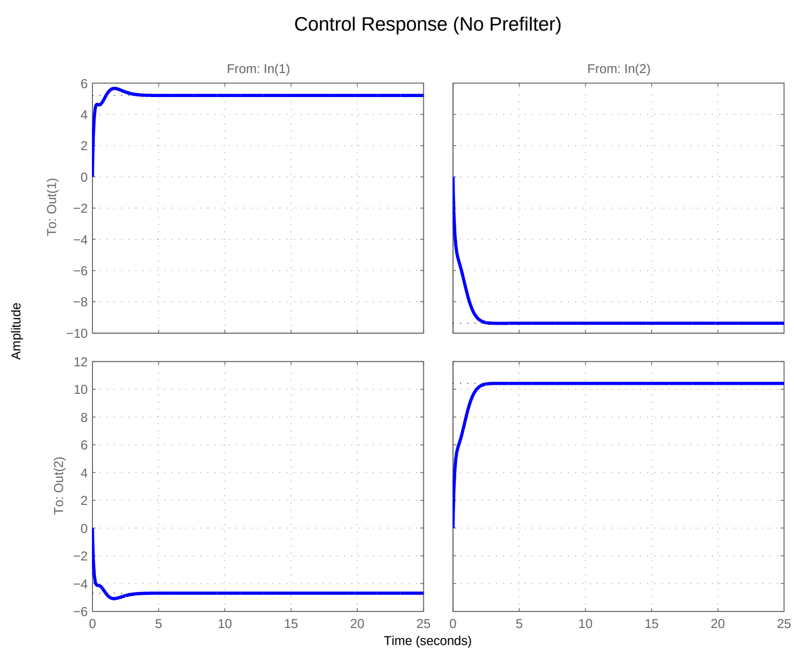


Figure 5.51: Control Time Response (no Pre-filter)

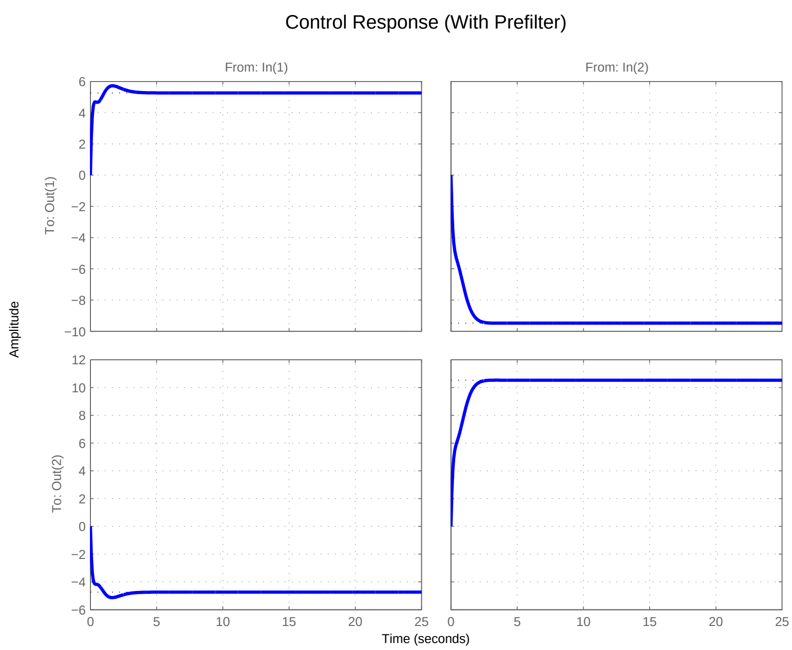


Figure 5.52: Control Time Response (with Pre-filter)

5.3 X-29 Lateral Dynamics Model

The TITO LTI model for the X-29 lateral dynamics (powered approach, Mach (0.259, 4.000 ft altitude, 14.777 lbs) is as follows [23]:

$$\dot{x} = Ax + Bu \tag{5.6}$$

$$y = Cx + Du \tag{5.7}$$

$$u = \begin{bmatrix} \delta_{df} & - & differential flap & (deg) \\ \delta_r & - & rudder\ flap & (deg) \end{bmatrix} \tag{5.8}$$

$$x = \begin{bmatrix} \beta & - & side\ slip\ angle & (ft/sec) \\ \text{p} & - & roll\ rate & (deg/sec) \\ \text{r} & - & yaw\ rate & (deg/sec) \\ \phi & - & roll\ angle (deg) & \end{bmatrix} \tag{5.9}$$

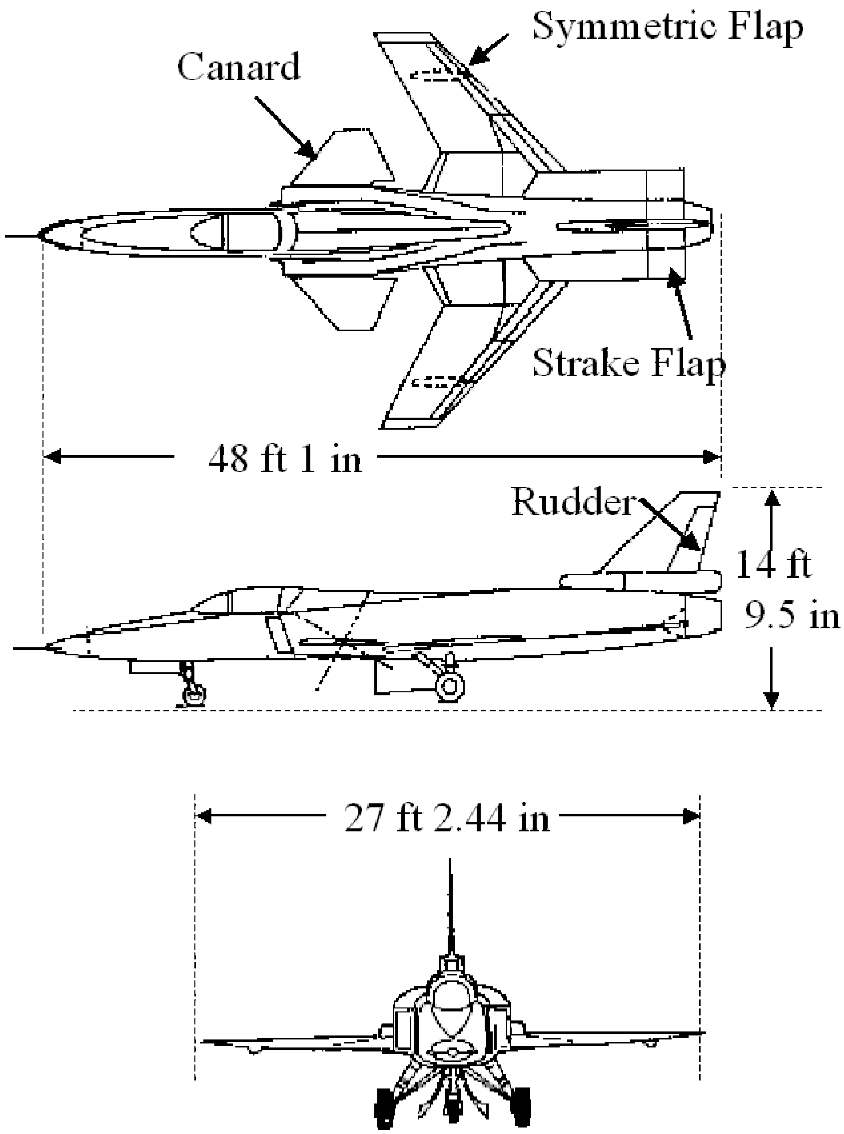
$$y = \begin{bmatrix} \phi & - & roll\ angle & (deg) \\ \beta & - & side\ slip\ angle & (deg) \end{bmatrix} \tag{5.10}$$

$$A = \begin{bmatrix} -0.1850 & 0.1475 & -0.9825 & 0.1120 \\ -3.4670 & -1.7100 & 0.9029 & 0.0000 \\ 1.1740 & -0.0825 & -0.1826 & -0.0000 \\ 0 & 1.0000 & 0.1492 & 0 \end{bmatrix} \tag{5.11}$$

$$B = \begin{bmatrix} -0.0256 & 0.0230 \\ 21.2869 & 3.1446 \\ 1.5202 & -0.7741 \\ 0 & 0 \end{bmatrix} \tag{5.12}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.13}$$

The aircraft is characterized by a lightly damped stable Dutch roll mode ($s =$



$-0.2455 \pm j1.2703$), a stable roll subsidence mode ($s = -1.6183$), and an unstable spiral divergence mode($s = 0.0318$). Fundamentally, the differential flap is used to control roll while the rodder is to control or your slip.

A bilinear transformation shifting is done in order to prevent its lightly damped roll poles from being canceled by the controller.

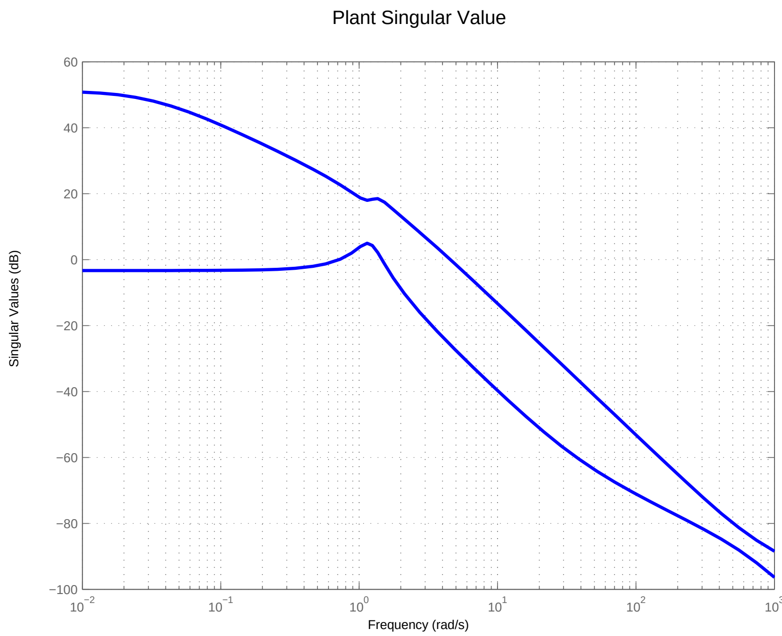


Figure 5.53: Plant Singular values

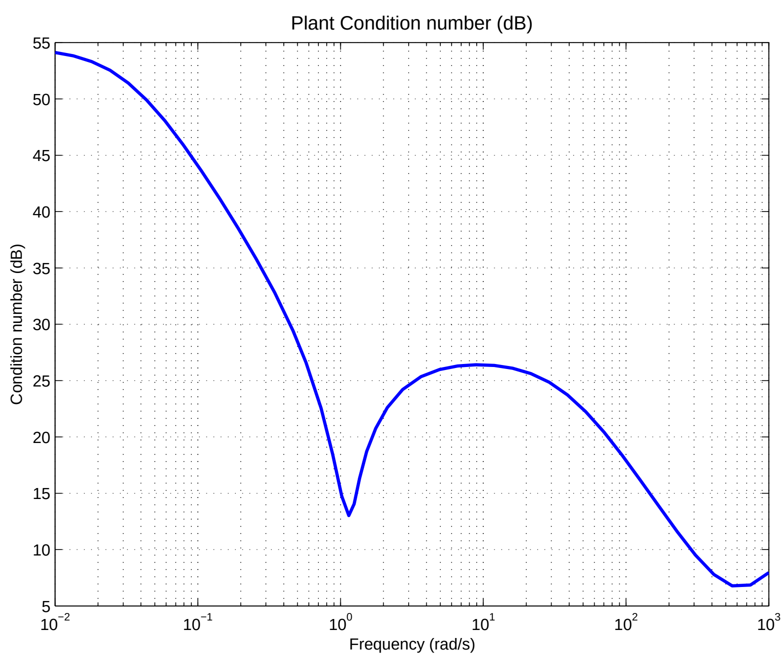


Figure 5.54: Plant Condition number

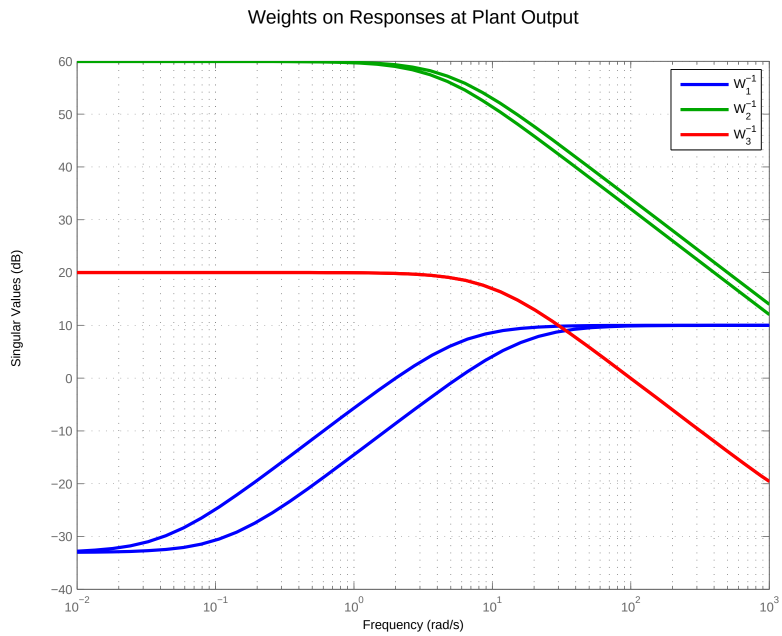


Figure 5.55: Weighting functions on output due to reference command

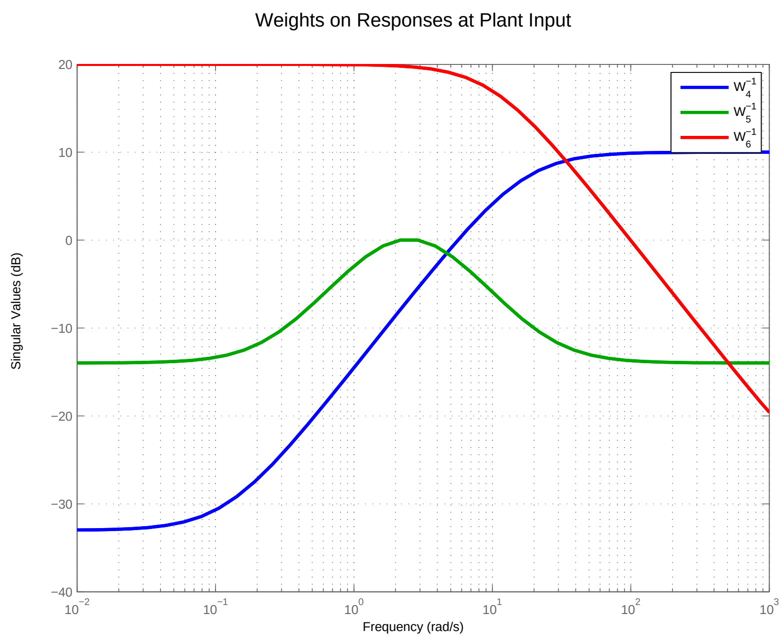


Figure 5.56: Weighting functions on output due to disturbance

Table 5.2: $\rho = 10^{-6}$: Plant Poles

Poles	Damping	Frequency (rad/sec)
-1.62e+00	1.00e+00	1.62e+00
-2.46e-01 + 1.27e+00i	1.90e-01	1.29e+00
-2.46e-01 - 1.27e+00i	1.90e-01	1.29e+00
3.18e-02	-1.00e+00	3.18e-02

Table 5.3: $\rho = 10^{-6}$: Plant Zeros

Zeros	Damping	Frequency (rad/sec)
-3.75e+01	1.00e+00	3.75e+01

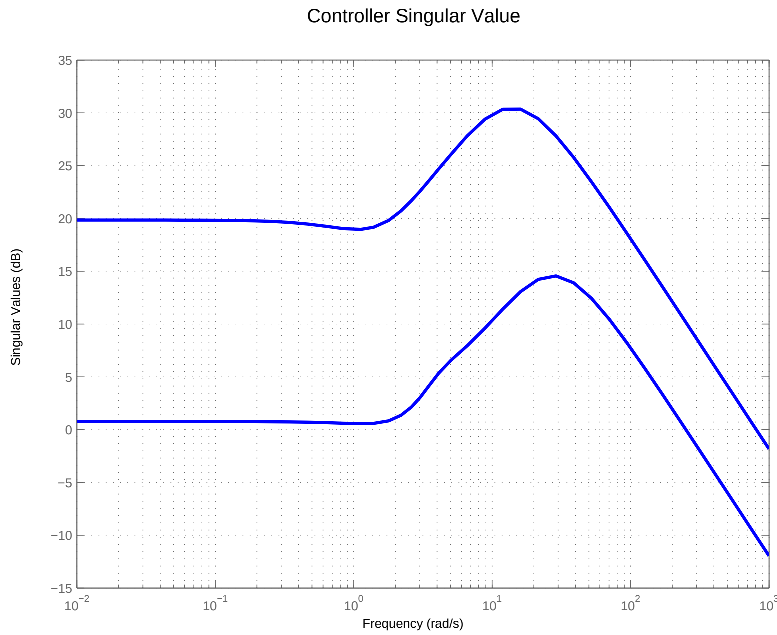


Figure 5.57: Controller Singular value

5.3.1 $\rho = 10^{-6}$ Approximation to standard mixed sensitivity problem

By choosing a near zero value for design parameter $rho = 10^{-6}$, we approximate the generalized mixed sensitivity problem to the standard mixed sensitivity problem. It is able to achieve good properties at plant output, while giving up on properties at plant input.

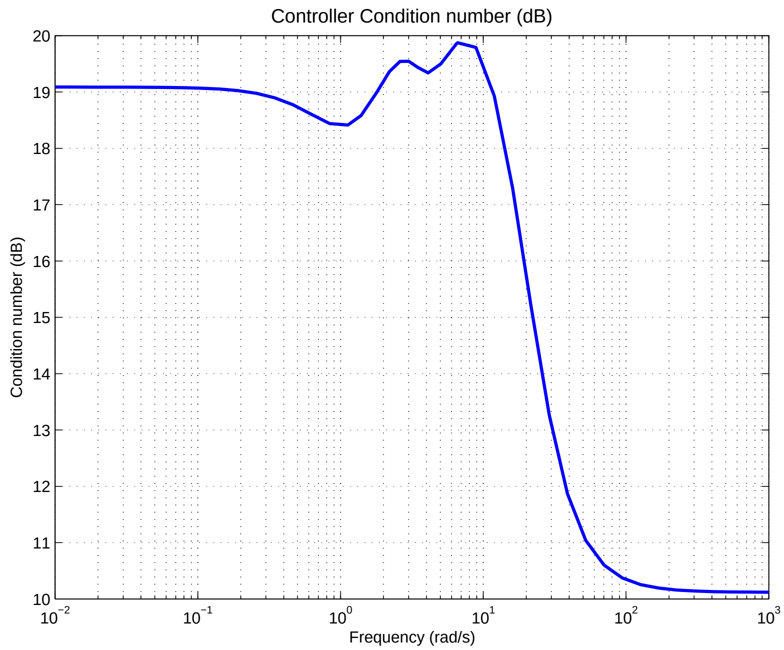


Figure 5.58: Controller Condition number

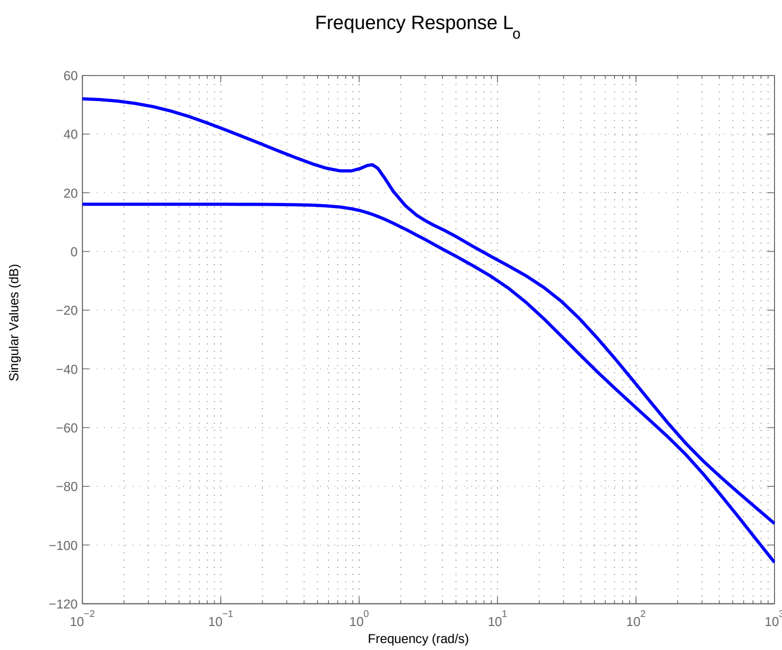


Figure 5.59: Open Loop transfer function at Plant output

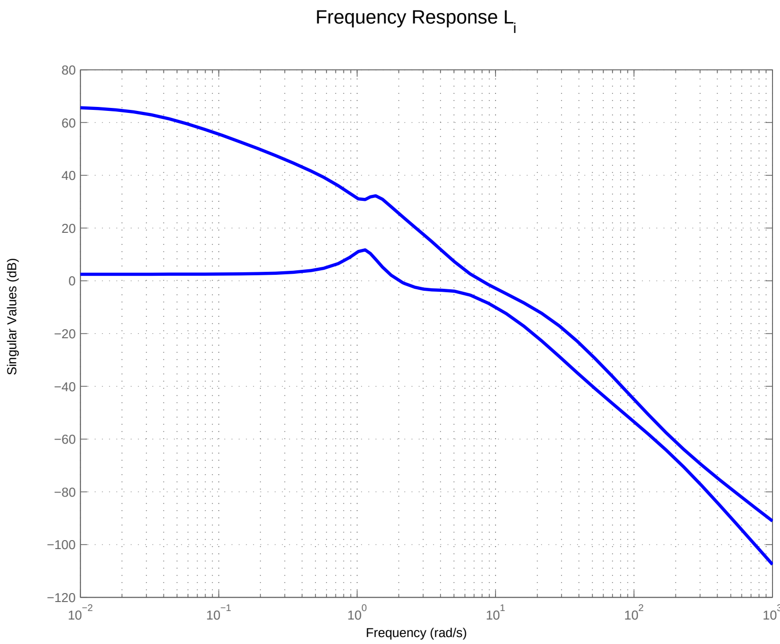


Figure 5.60: Open Loop transfer function at Plant input

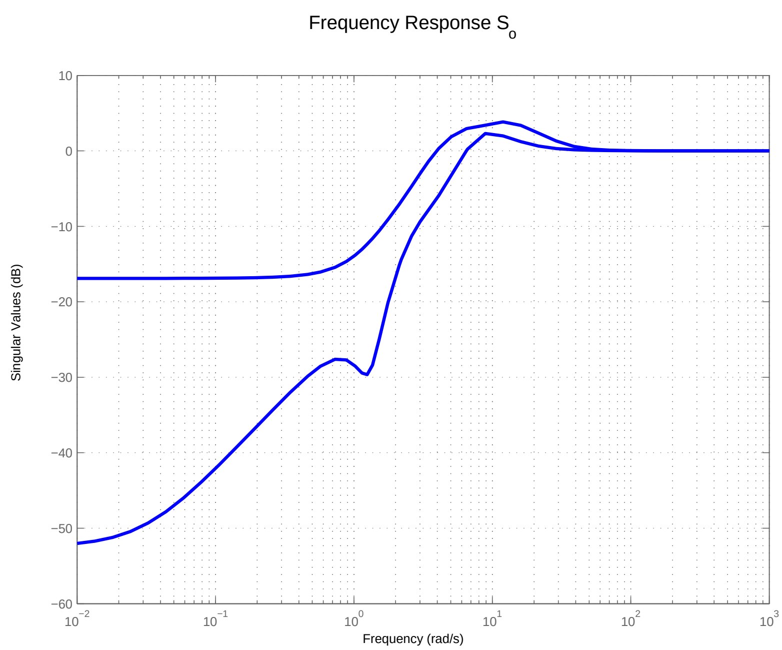


Figure 5.61: Output Sensitivity

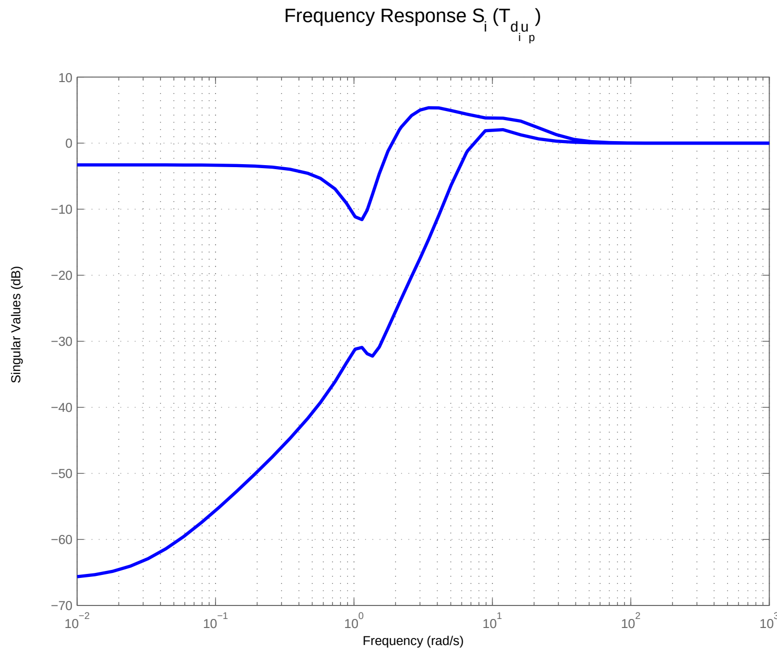


Figure 5.62: Input Sensitivity

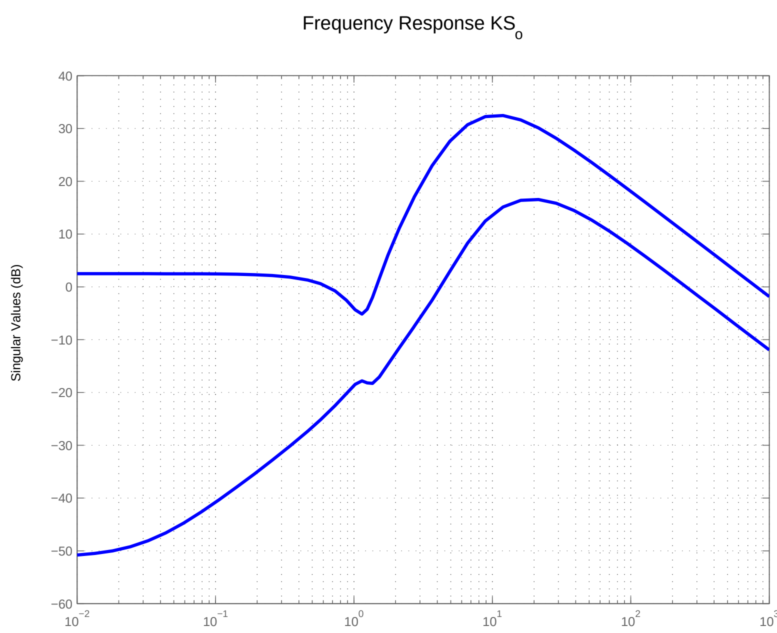


Figure 5.63: K*So

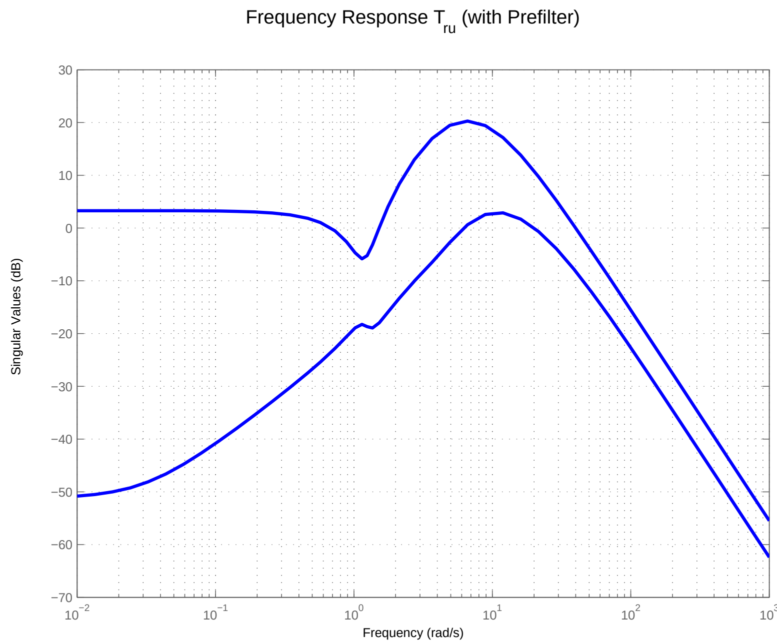


Figure 5.64: Reference to Control transfer function

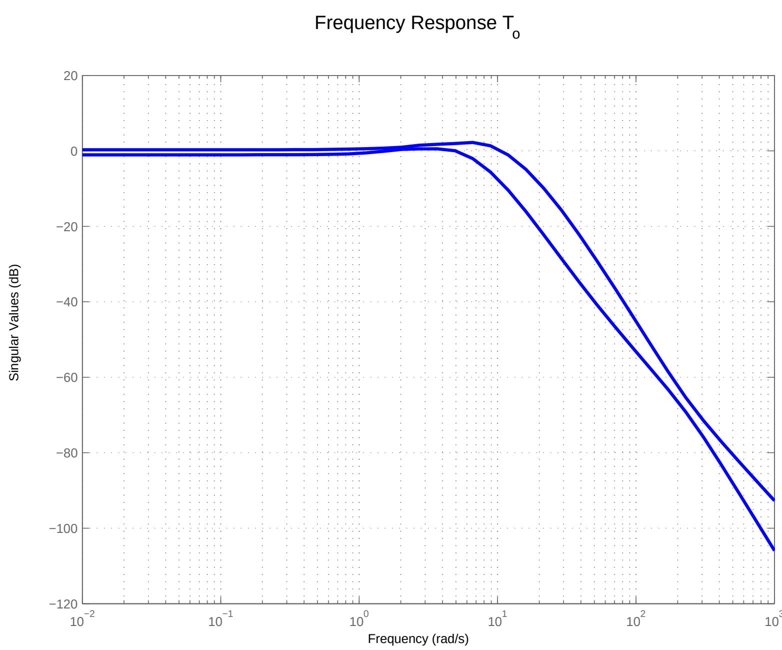


Figure 5.65: Output Complementary Sensitivity

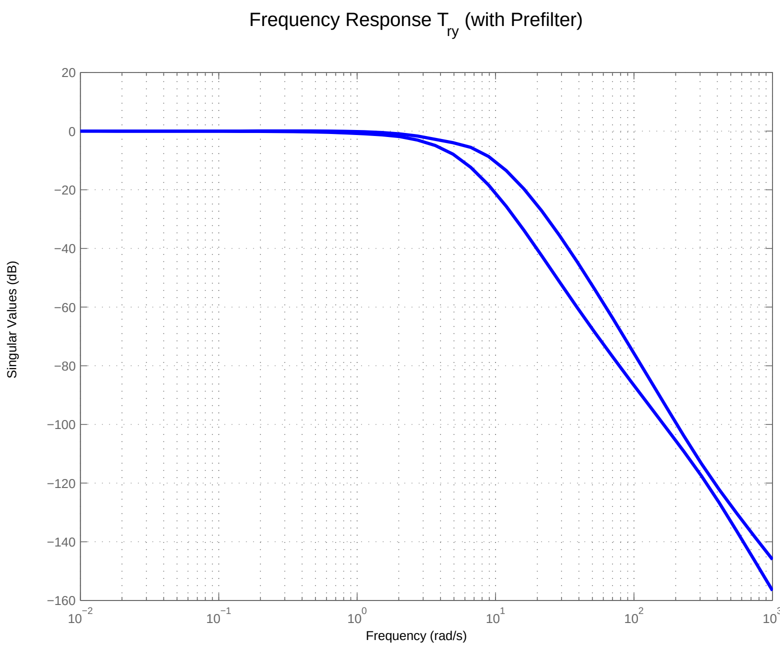


Figure 5.66: Reference to output transfer function

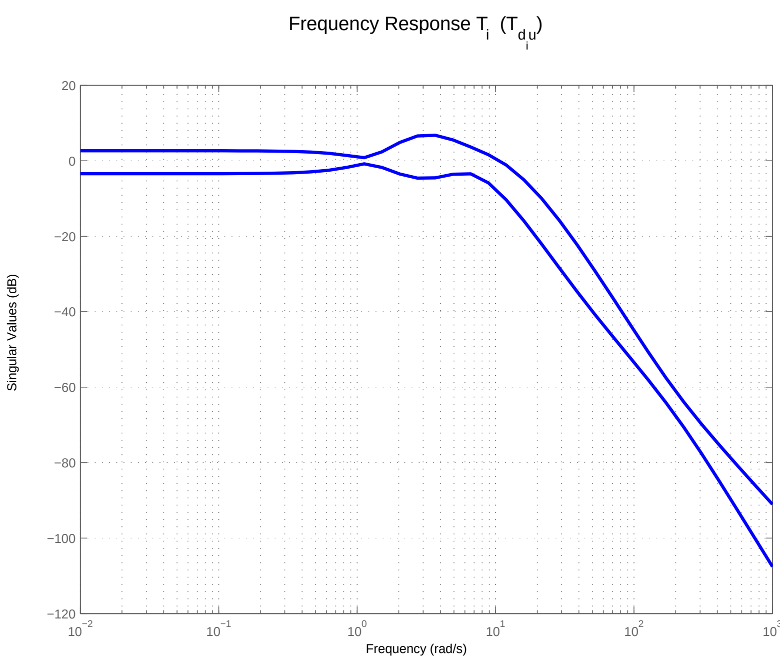


Figure 5.67: Input Complementary Sensitivity

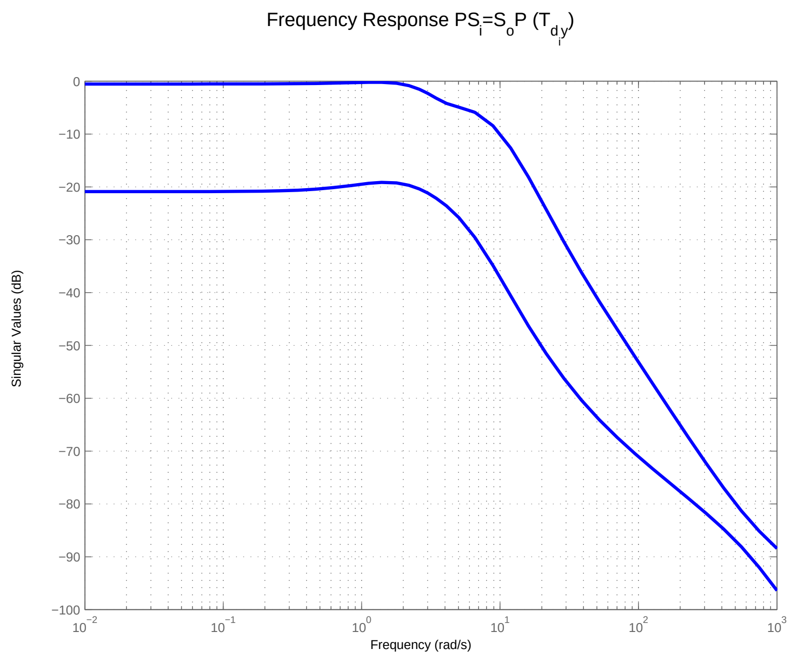


Figure 5.68: $PS_i = S_oP$

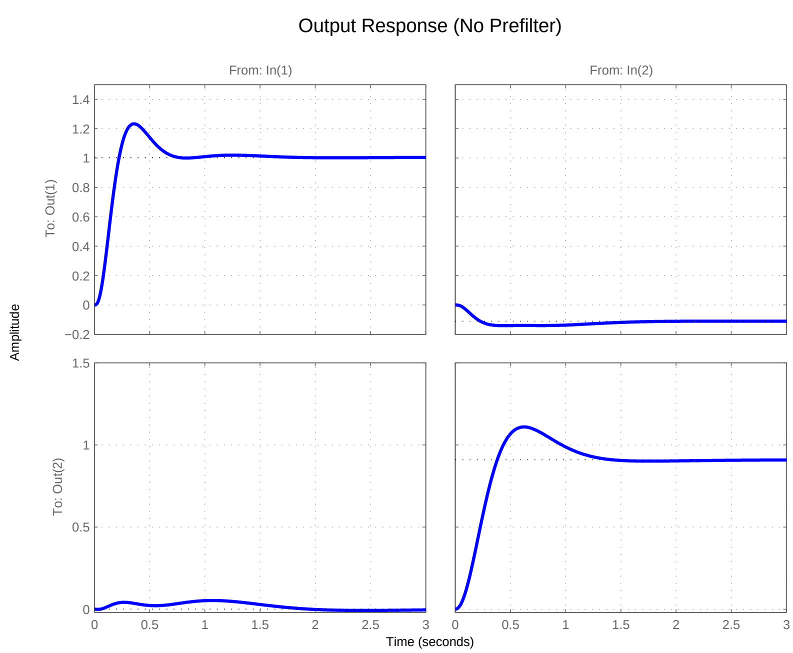


Figure 5.69: Output Time Response (no Pre-filter)

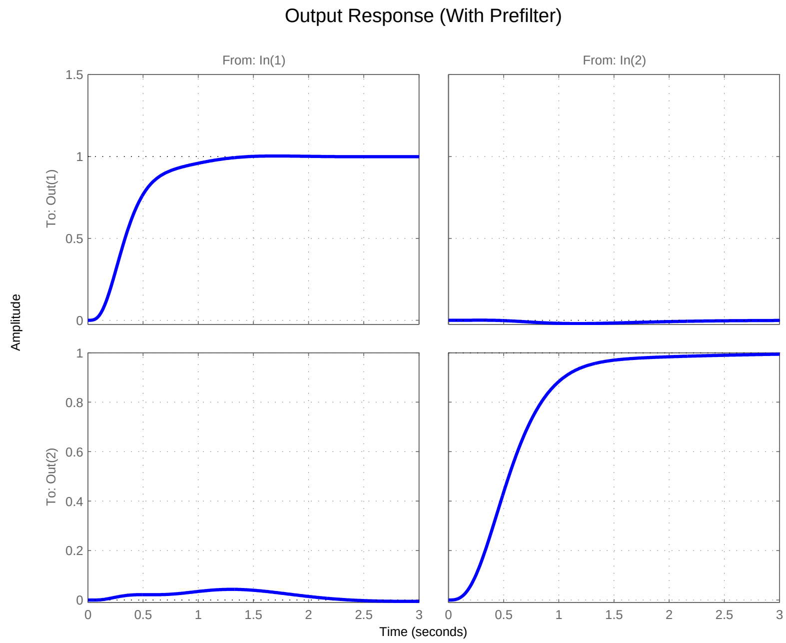


Figure 5.70: Output Time Response (with Pre-filter)

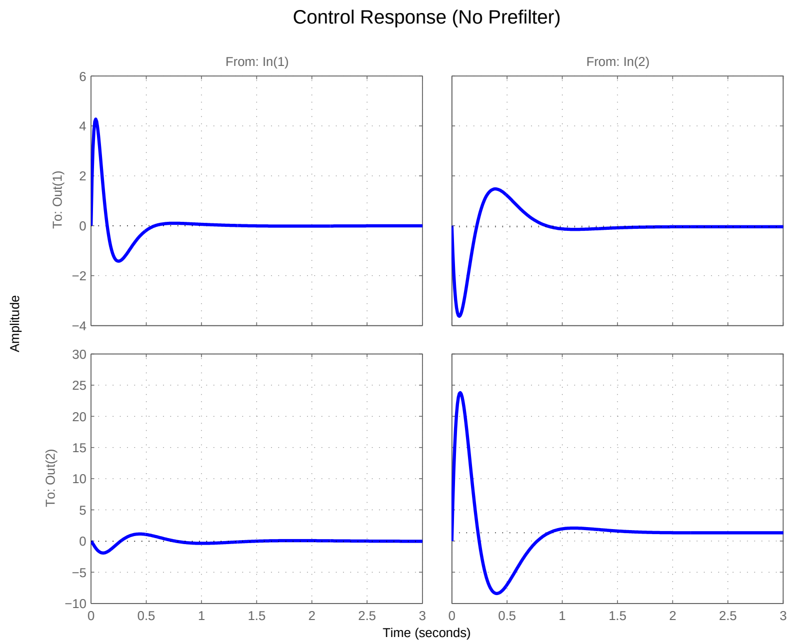


Figure 5.71: Control Time Response (no Pre-filter)

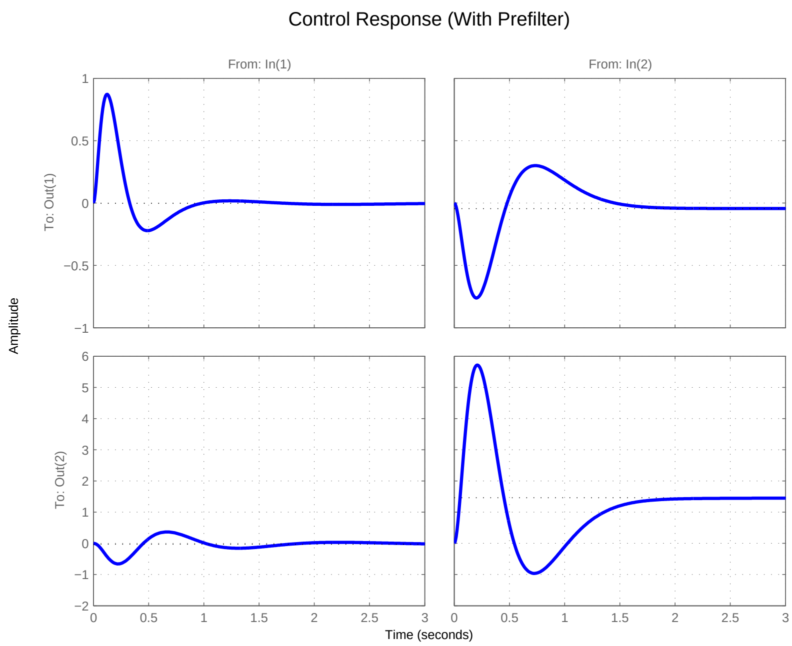


Figure 5.72: Control Time Response (with Pre-filter)

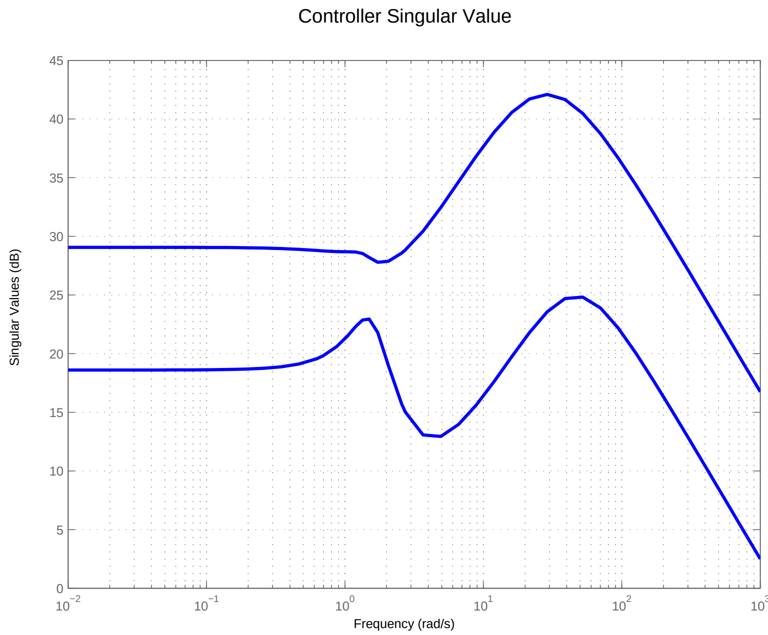


Figure 5.73: Controller Singular value

5.3.2 $\rho = 1$ *(A design with tradeoff)*

For $\rho = 1$, the feedback properties at plant output as well as input are penalized equally. This achieves a satisfactory trade-off between performances at the two loop-breaking points.

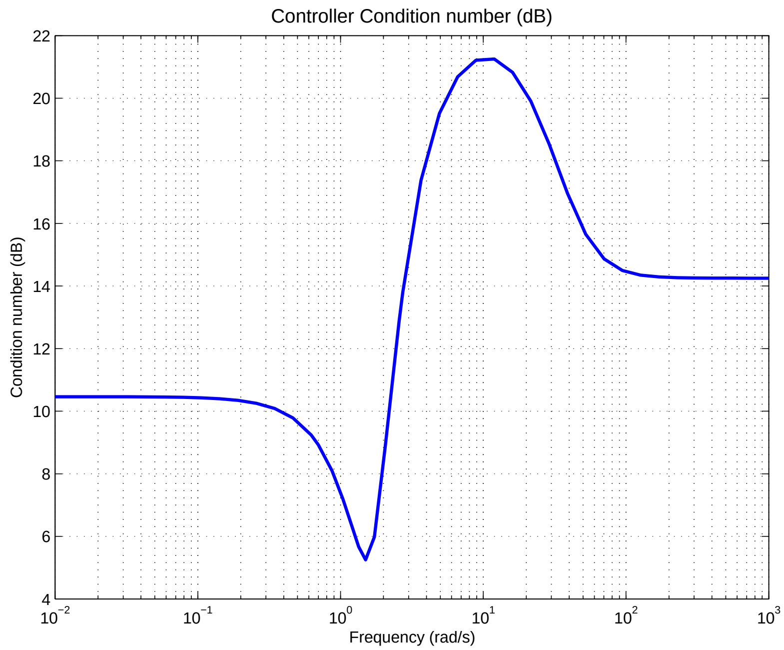


Figure 5.74: Controller Condition number

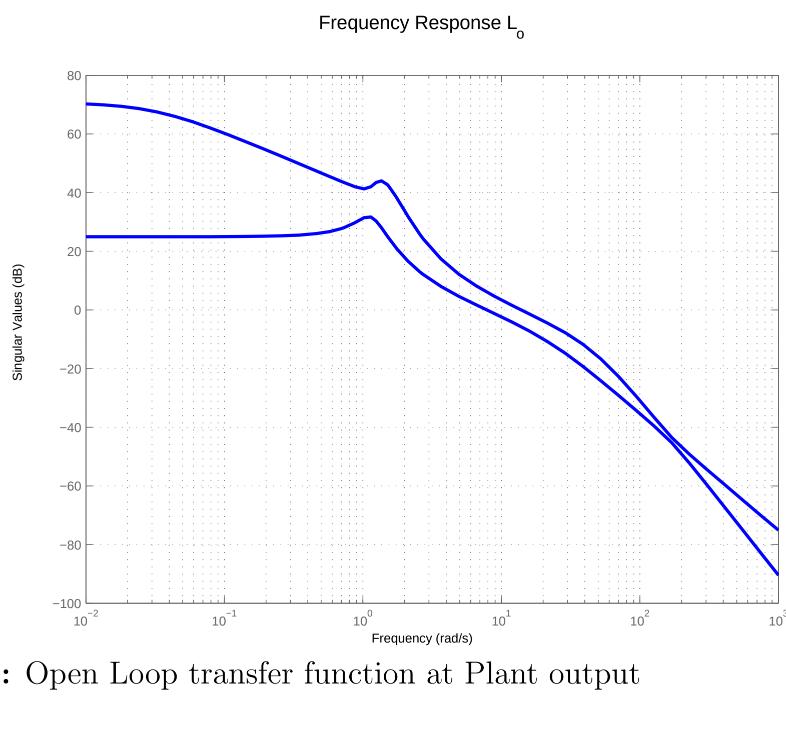


Figure 5.75: Open Loop transfer function at Plant output

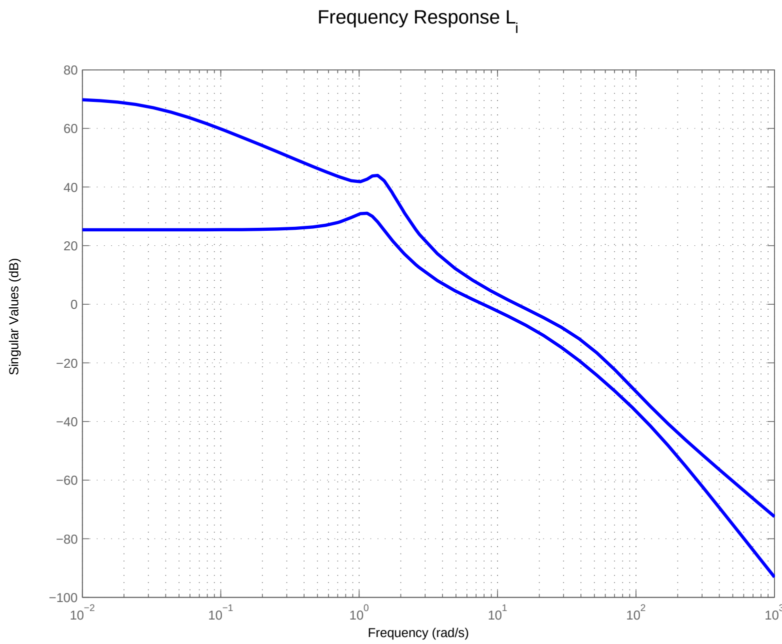


Figure 5.76: Open Loop transfer function at Plant input

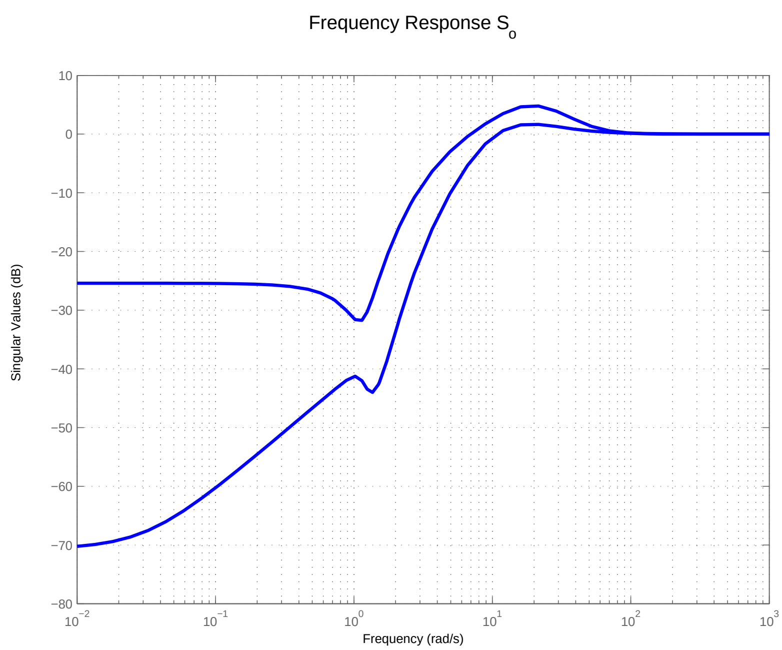


Figure 5.77: Output Sensitivity

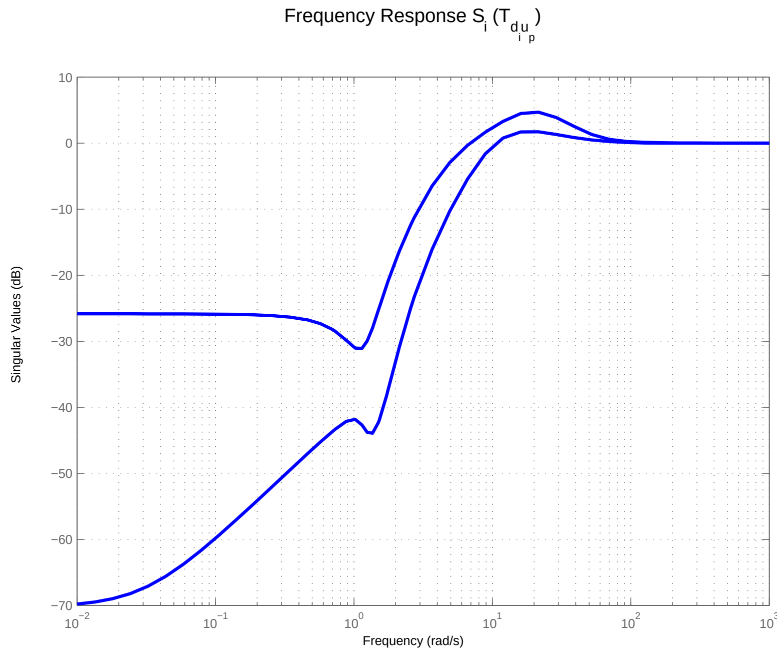


Figure 5.78: Input Sensitivity

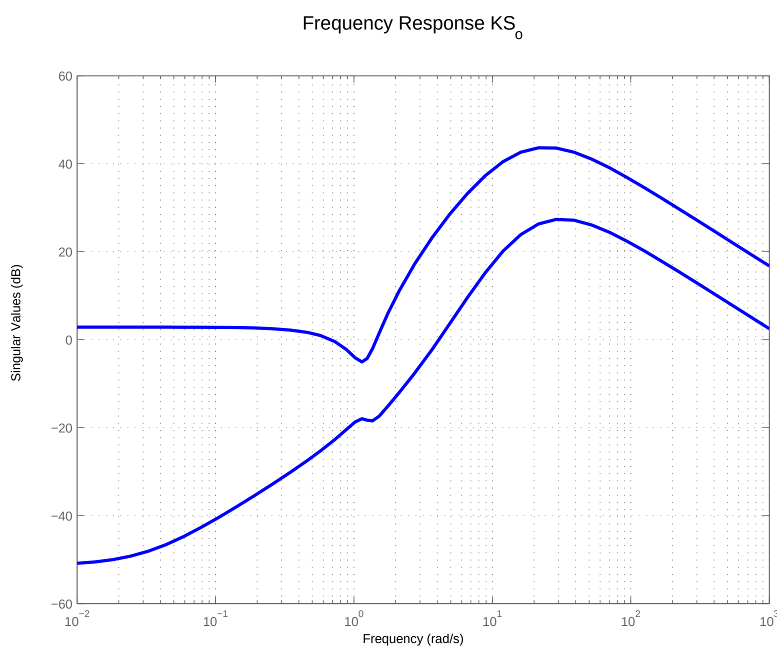


Figure 5.79: K*So

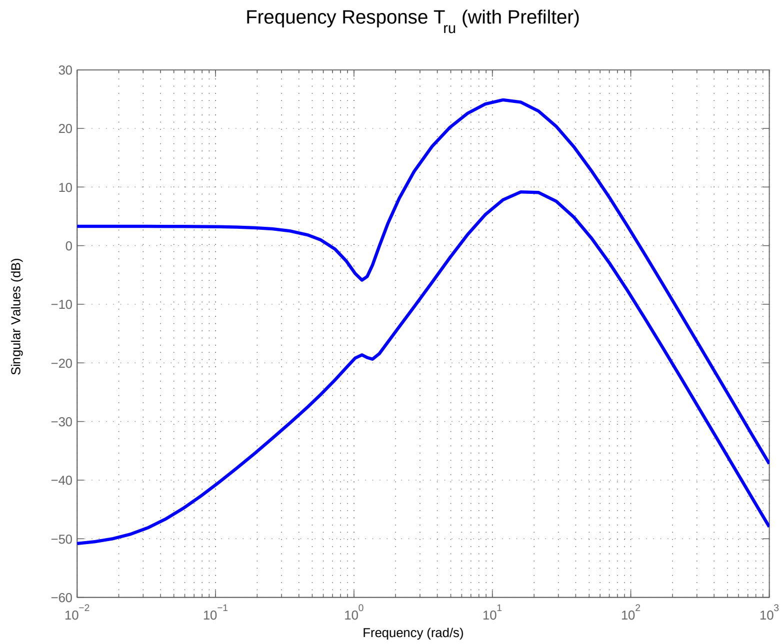


Figure 5.80: Reference to Control transfer function

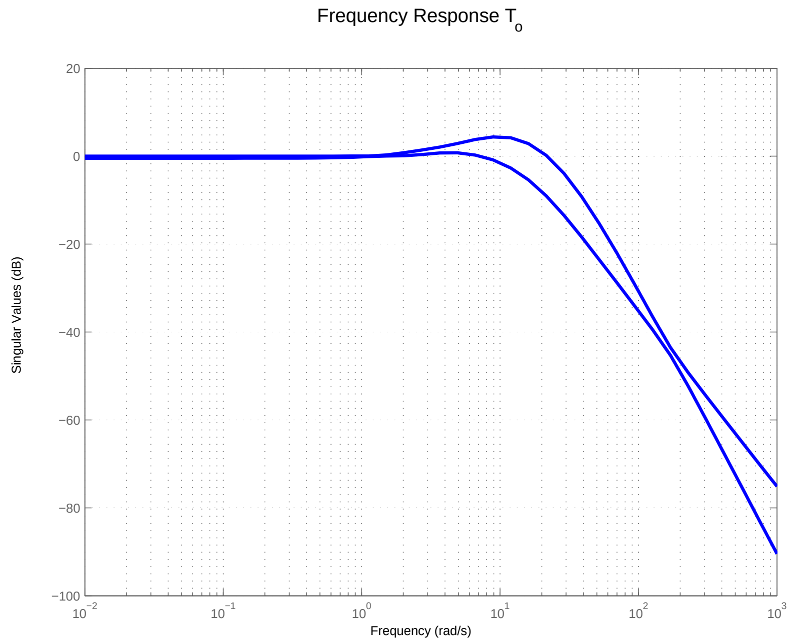


Figure 5.81: Output Complementary Sensitivity

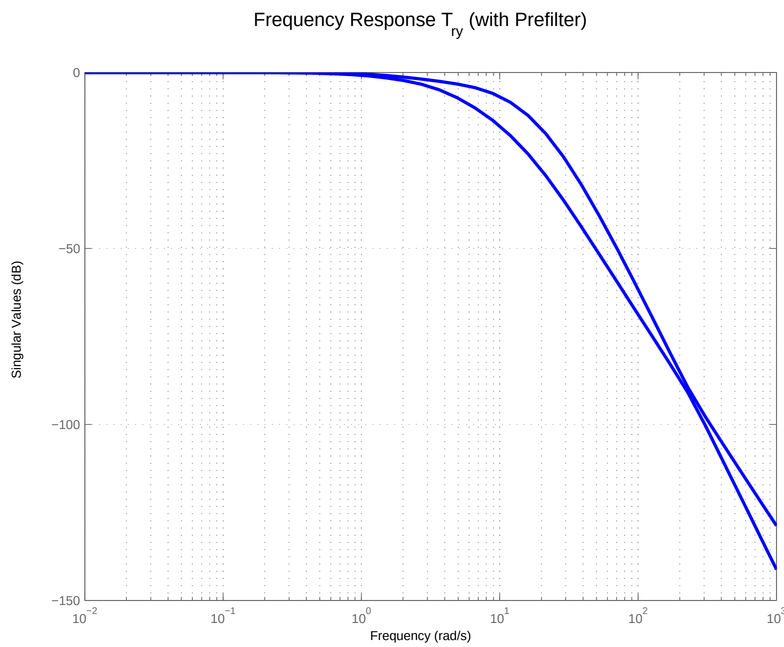


Figure 5.82: Reference to output transfer function

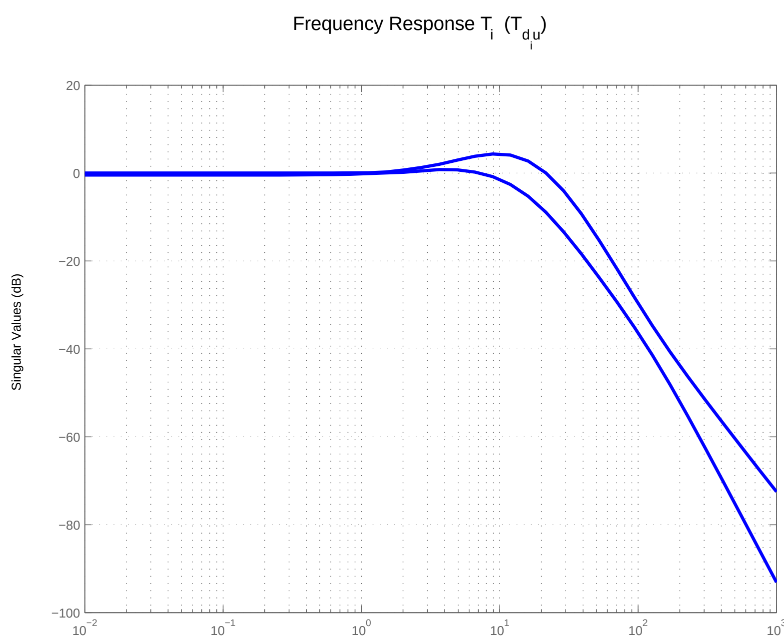


Figure 5.83: Input Complementary Sensitivity

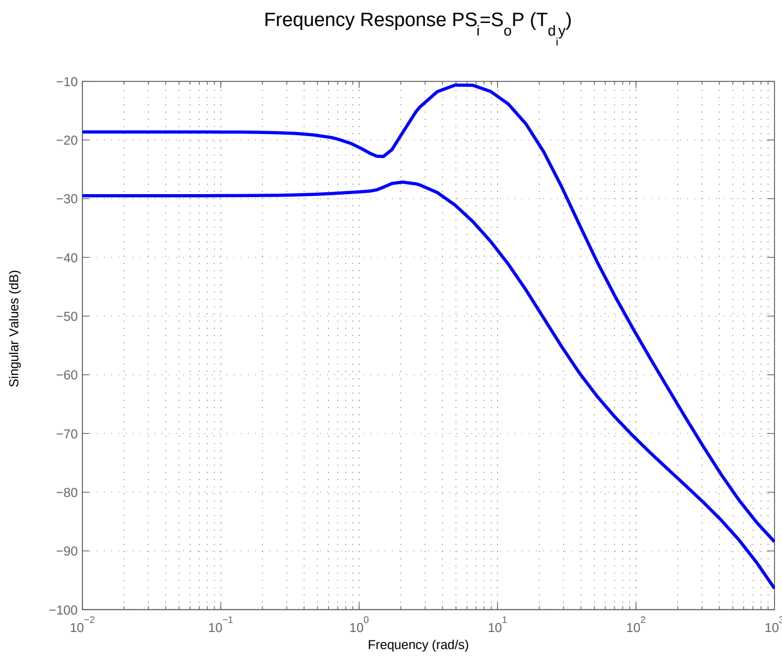


Figure 5.84: $PS_i = S_oP$

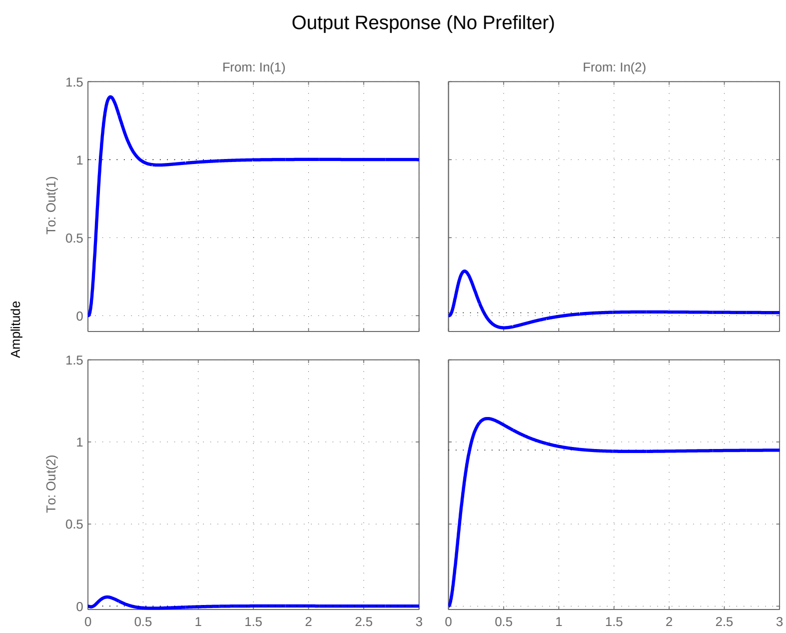


Figure 5.85: Output Time Response (no Pre-filter)

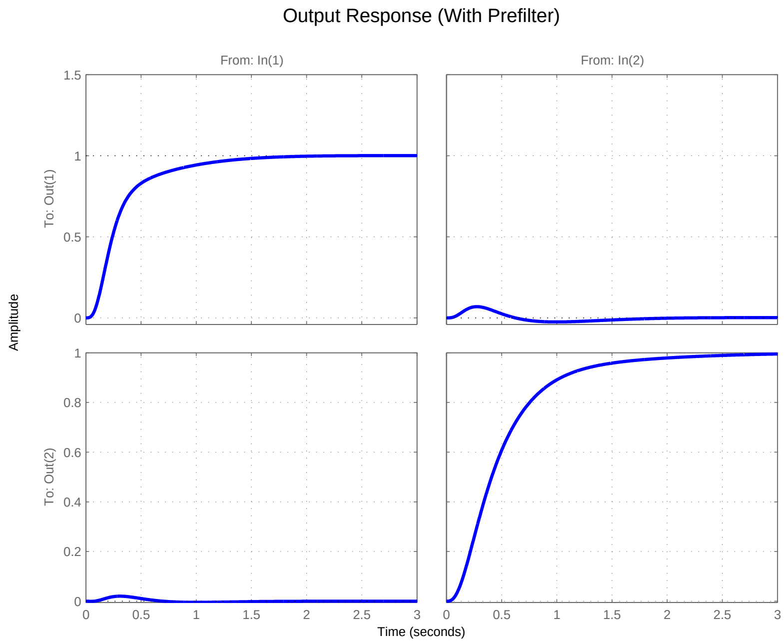


Figure 5.86: Output Time Response (with Pre-filter)

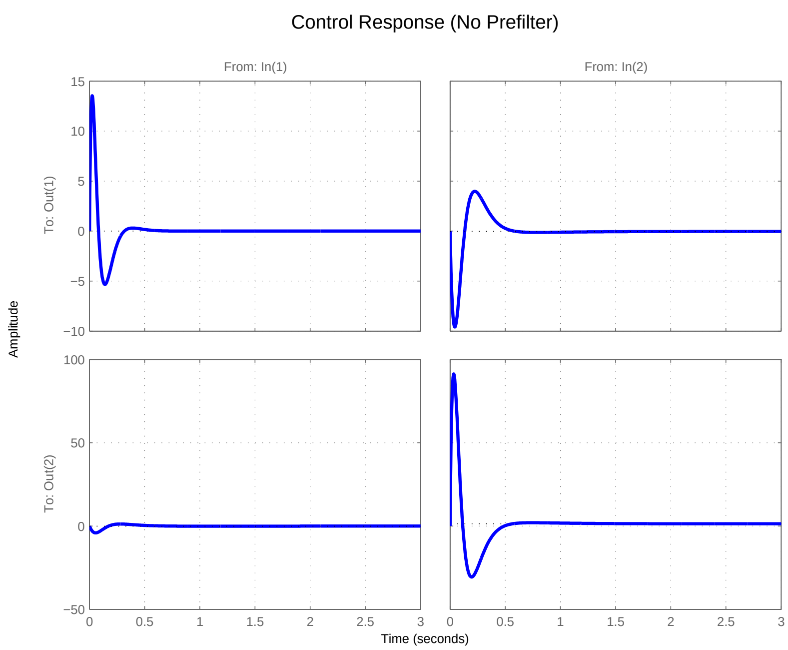


Figure 5.87: Control Time Response (no Pre-filter)

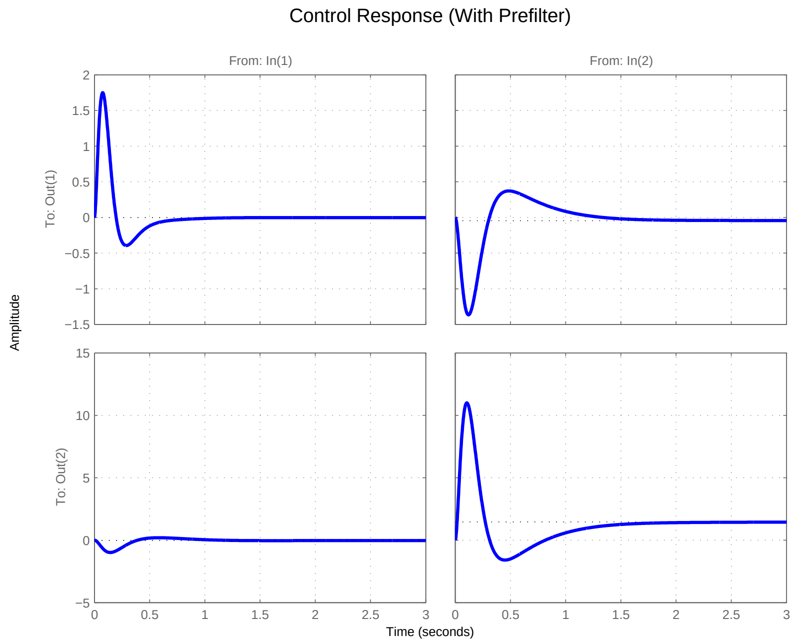


Figure 5.88: Control Time Response (with Pre-filter)

The Table 5.1 shows the $\mathcal{H}^{\infty}$ norms of individual transfer functions matrices.

Table 5.4: X-29 Aircraft: Comparison of Design Results (Values in dB)

ρ	S_o	S_i	KS_o	PS_i	T_o	T_i
10^{-6}	3.7828	5.3675	32.4998	-0.2467	2.2403	6.8925
1	4.7749	4.7560	43.6869	-10.5374	4.4067	4.3594

5.4 Summary and Conclusions

Two examples, namely Ill-Conditioned 2X2 coupled system and Lateral dynamics LTI model of X-29 Aircraft were used to illustrate the utility of the design environment. Results for different values of design parameter ρ were used to compare the design using standard mixed sensitivity problem and that using our generalized mixed sensitivity problem. Standard mixed sensitivity problem produced good results at plant input, whereas by choosing appropriate design parameters, generalized mixed-sensitivity problem achieved good trade-off of properties at different loop-breaking points.

Chapter 6

DESIGN ENVIRONMENT PLANNING

6.1 Introduction

In the previous chapters, the design tool which control design problems was illustrated. The utilities of the tool, namely achieving specifications, constraint handling and trade-offs of feedback properties help in addressing broad class of control design problems. A user-friendly Computer Aided Design (CAD) design environment is proposed in this chapter.

6.2 Functionalities

The design environment is a Matlab-based CAD tool [16]. It provides a platform for the user to design controllers without the need to interact with Matlab command line. The functionality of the tool are as follows:

- User-friendly Graphical User Interface (GUI) window to address broad class of control design problems
- Trade-off between feedback properties at distinct loop-breaking points
- Tuning Weighting functions
- Time domain and frequency domain constraints
- Optimal Q-parameter basis selection

6.3 GUI environment components

In this section the tools for controller design are explained.

6.3.1 MIMO LTI plant selection

The user can input any MIMO Plant Selection in state space or transfer function representation. Additionally, integrator action in the controller can be selected. Bilinear transformation is also done in order to avoid the undesirable effects of stable poles that are near imaginary axis.

6.3.2 Weight tuning

Weighting functions are manipulated by the user to influence the $\mathcal{H}^\infty$ problem to achieve desired closed loop specifications. They may be used to penalize tracking errors, actuator and other signal levels, state estimation errors, etc. By making the weight on a signal large in a specific frequency range, we are indirectly telling the optimization problem to find a controller that makes the signal small in that range.

$$\begin{aligned}
W_1(s) &= \tfrac{1}{M_s}\left(\tfrac{s+M_s w_b}{s+\in w_b}\right) \\
W_2(s) &= \tfrac{1}{\in}\left(\tfrac{s+w_{b_u} M_u}{s+\frac{w_{b_u}}{\in}}\right) \\
W_3(s) &= \tfrac{s+\frac{w_{bc}}{M_y}}{w_{b_c}} \\
W_4(s) &= \tfrac{1}{Mi_s}\left(\tfrac{s+Mi_s wi_b}{s+\in wi_b}\right) \\
W_5(s) &= \left(\tfrac{wi_{51}}{s+wi_{51}}\right)\left(\tfrac{s+wi_{52}}{wi_{52}}\right)\left(\tfrac{wi_{53}}{s+wi_{53}}\right) \\
W_6(s) &= \tfrac{s+\frac{wi_{bc}}{Mi_y}}{wi_{b_c}}
\end{aligned} \tag{6.1}$$

6.3.3 *Constraint specification*

The following is a partial list of control system design specifications that are convex in the closed loop transfer function matrix.

Convex Design Specifications. Each of the following is convex in the closed loop transfer function matrix.

- overshoot and undershoot
- time and frequency domain envelop constraints
- decoupling specifications (peak values)
- command following, disturbance attenuation, and noise attenuation requirements
- frequency response upper bounds
- actuator constraints
- rate limit constraints
- norm based performance and robustness specifications (e.g.$\mathcal{H}^\infty$, $\mathcal{H}^2$, $\mathcal{L}^\infty$, $\mathcal{L}^2$, $\mathcal{L}^1$, $\mathcal{L}^p$).

There are very important specifcations that are not convex. Rise time and settling time, for example, are not convex. They are only quasi-convex [2, pp. 132-133, pp. 175-177]; i.e.

$$f(\theta x_1 + (1-\theta)x_2) \leq \max\{\, f(x_1), f(x_2)\,\} \tag{6.2}$$

for all $\theta \in [0, 1]$ and x_1, x_2 in the domain of f.

6.3.4 Q-Basis parameters selection

1. Fixed pole basis

$$q_k = \left(\frac{p}{s+p}\right)^{k-1} \tag{6.3}$$

2. Fixed pole inner basis

$$q_k = \left(\frac{p-s}{s+p}\right)^{k-1} \tag{6.4}$$

3. Variable pole first order term basis

$$q_k = 1, q_{k+1} = \frac{kp}{s+kp} \tag{6.5}$$

4. Variable pole first order term inner basis

$$q_k = \frac{(k-1)p - s}{s + (k-1)p} \tag{6.6}$$

5. Fixed pole fixed zero basis

$$q_k = \left(\frac{z-s}{s+p}\right)^{k-1} \tag{6.7}$$

6.4 Summary and Conclusions

The utilites of the proposed Matlab-GUI tool is a powerful user-friendly tool. It addresses broad class of control design problems. The tool makes it easy for the user in designing controllers for required objectives and specifications which allows to trade-off feedback properties in the loop. It also handles constraints in both time domain and frequency domain.

Chapter 7

SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

7.1 Summary

In this dissertation, a comprehensive control design environment was developed. The utility of the design tool provides a user-friendly approach to achieving performance objectives at different loop-breaking points. A generalized mixed sensitivity problem was formulated to achieve trade-off between loop-breaking points. Implementation of LTI model of several applications illustrated the trade-offs and achieved our design objective.

7.2 Directions for Future Research

Future research will develop generalized mixed sensitivity problems for Linear Parameter Varying (LPV) plants, non-linear plants and infinite dimensional plants. In some applications, high controller order might be undesirable. Strategies to reduce controller order will be analysed. In addition, basis selection strategies in order to achieve optimal $\mathcal{H}^\infty$ performance will be developed. The design environment will be extended to include quasiconvex and eventually a broad class of non-linear performance specifications and constraints.

REFERENCES

[1] Balas, G., R. Chiang, A. Packard and M. Safonov, *Robust Control Toolbox Users Guide*, Natick, MA (2013).

[2] Boyd, S. P., C. H. Barratt, S. P. Boyd and S. P. Boyd, *Linear controller design: limits of performance* (Prentice Hall Englewood Cliffs, NJ, 1991).

[3] Boyd, S. P. and L. Vandenberghe, *Convex optimization* (Cambridge university press, 2004).

[4] Cheney, E. W. and A. A. Goldstein, "Newton's method for convex programming and tchebycheff approximation", Numerische Mathematik **1**, 253–268 (1959).

[5] Cifdaloz, O., *H-infinity mixed-sensitivity optimization for infinite dimensional plants subject to convex constraints*, vol. 68 (2007).

[6] Elhedhli, S., J.-L. Goffin and J.-P. Vial, *Cutting plane methods for nondifferentiable optimization* (Groupe d'études et de recherche en analyse des décisions, 2000).

[7] Francis, B. A., "Lecture notes in control and information sciences", (1987).

[8] Freudenberg, J., "Directionality, coupling, and multivariable loop-shaping", in "Decision and Control, 1988., Proceedings of the 27th IEEE Conference on", pp. 399–340 (IEEE, 1988).

[9] Freudenberg, J., "Analysis and design for ill-conditioned plants: Part 1. lower bounds on the structured singular value", International Journal of Control **49**, 3, 851–871 (1989).

[10] Freudenberg, J. and D. Loose, "Relations between properties of multivariable feedback systems at different loop-breaking points: Part i", in "Decision and Control, 1985 24th IEEE Conference on", vol. 24, pp. 250–256 (IEEE, 1985).

[11] Freudenberg, J. and D. Looze, "Relations between properties of multivariable feedback systems at different loop-breaking points: Part ii", in "American Control Conference, 1986", pp. 771–777 (IEEE, 1986).

[12] Kelley, J. E., Jr, "The cutting-plane method for solving convex programs", Journal of the Society for Industrial & Applied Mathematics **8**, 4, 703–712 (1960).

[13] Kučera, V., "Algebraic theory of discrete optimal control for multivariable systems [i.]", Kybernetika **10**, 7, 1–3 (1974).

[14] MacFarlane, A. G. J. and H. Nyquist, *Frequency-response methods in control systems*, vol. 1979 (ieee Press New York, NY, 1979).

[15] Maciejowski, J. M. and J. M. Maciejowski, *Multivariable feedback design*, vol. 8 (Addison-Wesley Wokingham, 1989).

[16] MathWorks, *MATLAB Creating Graphical User Interfaces*, Natick, MA (2013).

[17] Owens, D. H., "A historical view of multivariable frequency domain control", in "World Congress", vol. 15, pp. 1426–1426 (2002).

[18] Polak, E. and S. Salcudean, "On the design of linear multivariable feedback systems via constrained nondifferentiable optimization in h¡ sup¿ infinity¡/sup¿ spaces", Automatic Control, IEEE Transactions on **34** (1989).

[19] Rodriguez, A., *Analysis and Design of Multivariable Feedback Control Systems* (CONTROL3D,L.L.C., Tempe, AZ, 2002).

[20] Safonov, M., E. Jonckheere, M. Vermaj and D. Limebeer, "Synthesis of positive real multivariable feedback systems", International Journal of Control **45**, 3, 817–842 (1987).

[21] Shayeb, M., *Multivariable control system design via convex optimization*, Master's thesis, Arizona State University (2002).

[22] Skogestad, S., M. Morari and J. C. Doyle, "Robust control of ill-conditioned plants: High-purity distillation", Automatic Control, IEEE Transactions on **33**, 1092–1105 (1988).

[23] Stein, G., "Respect the unstable", Control Systems, IEEE **23**, 4, 12–25 (2003).

[24] Vidyasagar, M., *Control system synthesis: a factorization approach* (Morgan & Claypool Publishers, 2011).

[25] Youla, D., H. Jabr and J. Bongiorno Jr, "Modern wiener-hopf design of optimal controllers–part ii: The multivariable case", Automatic Control, IEEE Transactions on **21**, 3, 319–338 (1976).

[26] Zames, G., "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses", Automatic Control, IEEE Transactions on **26**, 2, 301–320 (1981).

[27] Zhou, K. and J. C. Doyle, *Essentials of robust control*, vol. 104 (Prentice Hall Upper Saddle River, NJ, 1998).

[28] Zhou, K., J. C. Doyle, K. Glover *et al.*, *Robust and optimal control*, vol. 40 (Prentice Hall New Jersey, 1996).

APPENDIX A

MATLAB CODE

```
% ******************* GenHinf_MixSens_OptiProblem **********************

tol_obj=0.01; tol_feas=0.01; % Define Tolerances
xmax1=10; xmin1=-10; % Bounds on optimization solution
N=4; p=0.7; % Q-Basis Parameters
x01=0; % Initial point for optimization
warning off;

FilePath0='DataFiles'; FileType='fig';
WtPlt=0; % Plot tfm's along with weighting functions
ShowPlot=0; % Show plots
ShowDamp=1; % Show poles and tzeros

UseHinfSyn=0; % For Finding K from hinfsyn
SecondPlot=0; % Plot figures for different controllers on same window

% Legend for figures
% LegendName1='Constrained'; LegendName2='Unconstrained';
LegendName1='GenHinf'; LegendName2='HinfSyn';

% % Select Plant
% PlntLabel='SISO_Unstable'; FilePath1='SISO_Unstable'; FreqMin=10^-2; ...
%     FreqMax=10^3; TFinal=10; MaxIter=80; Bilinear=0;

% PlntLabel='SISO_Stable'; FilePath1='SISO_Stable'; FreqMin=10^-2; ...
%     FreqMax=10^3; TFinal=3; MaxIter=80; Bilinear=0;

% PlntLabel='IllCondPlnt'; FilePath1='IllCondPlnt'; FreqMin=10^-3; ...
%     FreqMax=10^3; TFinal=15; MaxIter=250; Bilinear=0;

% PlntLabel='X29_Lateral'; FilePath1='X29_Lateral'; FreqMin=10^-2; ,,,
%     FreqMax=10^3; TFinal=3; MaxIter=300; Bilinear=1;

mu1=1; mu2=1; mu3=1;

% Select the value of rho
rho=1e-6; FilePath2='1em6';
% rho=1e-4; FilePath2='1em4';
% rho=1e-2; FilePath2='1em2';
% rho=1e-1; FilePath2='1em1';
% rho=9e-1; FilePath2='9em1';
% rho=1e0; FilePath2='1e0';
% rho=1e1; FilePath2='1e1';
rho1=rho; rho2=rho; rho3=rho;

%% Plant


switch PlntLabel

    case 'SISO_Unstable'
        P_tf = tf([1],[1 -1]);%*[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);

    case 'IllCondPlnt'
```

```
        s=tf('s');
        P_tf=[1/(s+1) 0; 0 1/(s+2)]*[1 0.9; 0.9 1];
        P_ss=ss(P_tf);
        [Ap, Bp, Cp, Dp] = ssdata(P_ss);

    case 'SISO_Stable'
        P_tf = tf([1],[1 1]);%*[1 0.1; 0.1 1];
        P_ss = ss(P_tf);
        [Ap, Bp, Cp, Dp] = ssdata(P_ss);

    case 'X29_Lateral'
        % X-29 Lateral dynamics
        Ap=[-0.1850 0.1475 -0.9825 0.1120; -3.4670 -1.7100 0.9029 ...
            0.0000; 1.1740 -0.0825 -0.1826 -0.0000; 0 1.0000 0.1492 0];
        Bp=[-0.0256 0.0230; 21.2869 3.1446; 1.5202 -0.7741; 0 0];
        Cp=[0 0 0 1; 1 0 0 0]; Dp=[0 0; 0 0];
        P_ss=ss(Ap,Bp,Cp,Dp);
        p2 = -1e20; p1 = -1.2;

end

%%
[n_e, n_u] = size(P_ss);

%% Bilinear Transformation
if  Bilinear==1
    P0=P_ss;
    [At,Bt,Ct,Dt]=bilin(P_ss.a,P_ss.b,P_ss.c,P_ss.d,1,'Sft_jw',[p2 p1]);
    P_ss=ss(At,Bt,Ct,Dt);
    Ap=At; Bp=Bt; Cp=Ct; Dp=Dt;
end

%% Objective Weighting Functions

switch PlntLabel
    case {'SISO_Unstable','SISO_Stable'}
        % Marco weighting Wd2 new
        Eps=0.01;
        Ms=100; wb=3;
        W1 = tf([1/Ms wb], [1 wb*Eps])*eye(n_e);
        Mu=0.001; wbu=100; Mu2=0.002; wbu2=120;
        W2 = [tf([1 wbu*Mu],[Eps wbu])]*eye(n_u);
        My=50; wbc=20;
        W3 = tf([1 wbc/My], [Eps wbc])*eye(n_e);
        Wd1=W1(1,1)*eye(n_u);
        wd21=0.1; wd22=1; wd23=10; s=tf('s');
        Wd2=((wd21/(s+wd21))*((s+wd22)/wd22)^2*(wd23/(s+wd23)))*eye(n_e);
        Wd3=W3(1,1)*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);

    case {'IllCondPlnt'}
        % Academic example
        Eps=0.01;
        Ms1=1; wb1=0.1; Ms2=1; wb2=0.1;
```

```
        W1 = [tf([1/Ms1 wb1], [1 wb1*Eps]) 0; 0 tf([1/Ms2 wb2], ...
            [1 wb2*Eps])];
        Mu1=0.005; wbu1=500; Mu2=0.005; wbu2=500;
        W2 = [tf([1 wbu1*Mu1],[Eps wbu1]) 0;0 tf([1 wbu2*Mu2],[Eps wbu2])];
        My=10; wbc=20; %
        W3 = tf([1 wbc/My], [Eps wbc])*eye(n_e);
        Wd1=W1(1,1)*eye(n_u);
        wd21=0.5; wd22=5; wd23=50; s=tf('s');
        Wd2=((wd21/(s+wd21))*((s+wd22)/wd22)^2*(wd23/(s+wd23)))*eye(n_e);
        Wd3=W3(1,1)*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);

    case 'X29_Lateral'
        Eps = 0.001;
        Ms1 = 10; Ms2=10; wb1=5.35; wb2 = 1.90;
        W11 = tf([1/sqrt(Ms1) wb1], [1 wb1*sqrt(Eps*0.5)]);
        W12 = tf([1/sqrt(Ms2) wb2], [1 wb2*sqrt(Eps*0.5)]);
        W1=[W11 0; 0 W12];
        wbu1 = 4000; wbu2=5000; Mu1=0.001; Mu2 = 0.001;
        W21 = tf([1 wbu1*Mu1], [Eps wbu1]);
        W22 = tf([1 wbu2*Mu2], [Eps wbu2]);
        W2=[W21 0; 0 W22];
        My1 = 100; My2=100; wbc1=100; wbc2=100;
        W31 = tf([1 wbc1/sqrt(My1)], [sqrt(Eps) wbc1]);
        W32 = tf([1 wbc2/sqrt(My2)], [sqrt(Eps) wbc2]);
        W3=[W31 0; 0 W32];
        Wd1=W1(1,1)*eye(n_u);
        wd21=0.25; wd22=2.5; wd23=25; s=tf('s');
        Wd2=5*((wd21/(s+wd21))*((s+wd22)/wd22)^2*(wd23/(s+wd23)))*eye(n_e);
        % Wd2=W2(1,1)*eye(n_e);
        Wd3=W3(1,1)*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);
end

%% Select Constraints
W1c=[]; W2c=[]; W3c=[]; Wd1c=[]; Wd2c=[]; Wd3c=[];


%     FilePath1=[FilePath1 '_Con'];
%
%     W2c{1}.tfm = tf(1,1)*eye(n_e);   % Constraint Weigthing
%     W2c{1}.Fun = 'conHINF'; % Constraint Type
%     W2c{1}.Val = db2mag(12);
%
% W1c{1}.tfm = tf(1,1)*eye(n_e);   % Constraint Weigthing
% W1c{1}.Fun = 'conHINF'; % Constraint Type
% W1c{1}.Val = 1.2;
%
%     W2c{1}.tfm = tf(1,1)*eye(n_u);   % Constraint Weigthing
%     W2c{1}.Fun = 'conPEAK_MIMO_AllStep'; % Constraint Type
%     W2c{1}.Val = 3;       % Constraint Value
```

```
%%

% Nominal Controller
n_x=size(Ap,1); n_e=size(Cp,1); n_u=size(Bp,2);
F = lqr(Ap, Bp, eye(n_x), eye(n_u));
L = lqr(Ap',Cp',eye(n_x), eye(n_e));
L=L';
Ko = ss(Ap-Bp*F-L*Cp+L*Dp*F, -L, -F, 0);

% 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)); ...
    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));...
    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; TIn12=DenP; TIn21=NumPt;
TIn11=DenP*NumKt*inv(DenPt)*NumPt; TIn12=DenP; TIn21=NumPt;

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
if isempty(W1c)
    T11rz1c=[]; T12rz1c=[]; T21rz1c=[];
else
    T11rz1c=W1c{1}.tfm*SOut11; T12rz1c=W1c{1}.tfm*SOut12; T21rz1c=SOut21;
end
if isempty(W2c)
    T11rz2c=[]; T12rz2c=[]; T21rz2c=[];
else
```

```
    T11rz2c=W2c{1}.tfm*KSOut11; T12rz2c=W2c{1}.tfm*KSOut12;T21rz2c=KSOut21;
end
if isempty(W3c)
    T11rz3c=[]; T12rz3c=[]; T21rz3c=[];
else
    T11rz3c=W3c{1}.tfm*TOut11; T12rz3c=W3c{1}.tfm*TOut12; T21rz3c=TOut21;
end
if isempty(Wd1c)
    T11dz1c=[]; T12dz1c=[]; T21dz1c=[];
else
    T11dz1c=Wd1c{1}.tfm*SensIn11; T12dz1c=Wd1c{1}.tfm*SensIn12;
    T21dz1c=SensIn21;
end
if isempty(Wd2c)
    T11dz2c=[]; T12dz2c=[]; T21dz2c=[];
else
    T11dz2c=Wd2c{1}.tfm*SInP11; T12dz2c=Wd2c{1}.tfm*SInP12; T21dz2c=SInP21;
end
if isempty(Wd3c)
    T11dz3c=[]; T12dz3c=[]; T21dz3c=[];
else
    T11dz3c=Wd3c{1}.tfm*TIn11; T12dz3c=Wd3c{1}.tfm*TIn12; T21dz3c=TIn21;
end

%%
q = conBASIS(N,p,0,2);

%% 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;
%%
x0 = x01*ones(N*n_u*n_e,1);
xmin = xmin1*ones(N*n_u*n_e,1);
xmax =  xmax1*ones(N*n_u*n_e,1);
Q = conFORMQN(x0, q, n_u, n_e, N);
%% Problem Data
[n_e, n_u, ProblemDatarz, ProblemDatadz] = conORGANIZE_Gen(P_ss, W1, ...
    W2, W3, Wd1, Wd2, Wd3, W1c, W2c, W3c, Wd1c, Wd2c, Wd3c);

%%
[Mrz, Mobjrz, Mconrz]=conVECTORIZE(T11rz,T12rz,T21rz,q,N,n_u,n_e,...
    ProblemDatarz);
[Mdz, Mobjdz, Mcondz]=conVECTORIZE(T11dz,T12dz,T21dz,q,N,n_u,n_e,...
    ProblemDatadz);

%% Kelley's Cutting Plane Method

Datarz=ProblemDatarz; Datadz=ProblemDatadz;

NQ=N;

% 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);

N = length(xk);             % Dimension of problem
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;
iter = 0; % Iteration count
w=zeros(N+1,1);

% LP solver options
options = optimset('Display','off','simplex', 'on');
% START
Q = conFORMQN(xk, q, n_u, n_e, NQ); %% Q - Parameter
while UkminLkrz > tol_obj || UkminLkdz > tol_obj || ...
        (constraint_flagrz>0) || (constraint_flagdz>0)
    Ac=[]; bc=[];

    [forz, Gfo] =feval('conHINF', 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('conHINF', 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';
        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*0,options);

    UkminLkrz=forz-c'*w; fprintf('\n%d %1.6f  %1.6f  ', iter,forz, ...
        UkminLkrz);
    if ~isempty(frz{1})
        fprintf('%1.6f  ', frz{1});
    end
    UkminLkdz=fodz-c'*w; fprintf('%1.6f  %1.6f  ', fodz, UkminLkdz);
    if ~isempty(fdz{1})
        fprintf('%1.6f  ', fdz{1});
    end
    foStore(:,iter+1)=[forz; fodz];

    % Update xk
    xk = w(1:N); xkStore(:,iter+1)=xk; ExitFlagStore(1,iter+1)=exitflag;

    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('I CANNOT SOLVE THIS \n')
        break;
    end
    Q = conFORMQN(xk, q, n_u, n_e, NQ); %% Q - Parameter
end
% ****************************************************
%%
disp('12. Form Q')
```

```
Q = conFORMQN(xk, q, n_u, n_e, NQ);
disp(' ')
%%
[forz, Gfo] =feval('conHINF', Mobjrz, xk, T11rz, T12rz, T21rz, Q, ...
    Datarz.ObjVec);
[fodz, Gfo] =feval('conHINF', Mobjdz, xk, T11dz, T12dz, T21dz, Q, ...
    Datadz.ObjVec);
fx=max([forz,fodz]); %Check about this
%%

% Form K

% Q - Parameter
Aq = Q.a; Bq = Q.b; Cq = Q.c; Dq = Q.d;

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);

%% Matlab Hinf
if UseHinfSyn==1
    GenP=augw(P_ss,W1,W2,W3);
    [K,CL,GAM]=hinfsyn(GenP);
    LegendUseHinfSyn=1;
else
    LegendUseHinfSyn=0;
end

%% *************** Inverse Bilinear Transformations ***************
if  Bilinear==1
    [Acp1,Bcp1,Ccp1,Dcp1] = ssdata(K);
    [Atk1,Btk1,Ctk1,Dtk1]=bilin(Acp1,Bcp1,Ccp1,Dcp1,-1,'Sft_jw',[p2 p1]);
    K=ss(Atk1,Btk1,Ctk1,Dtk1);
    P_ss=P0;
end

%% Closed Loop Maps

[Lo,Li,So,Si,To,Ti,KS,PS] = f_CLTFM(P_ss,K);
% PS=feedback(P_ss,K);

% Try and Tru with prefilter
switch PlntLabel
    case {'SISO_Stable','SISO_Unstable'}
        Prefilter=(1/dcgain(To))*tf(1e2,[1 1e2]);
    case 'IllCondPlnt'
        Prefilter=inv(dcgain(To))*[tf(1e2,[1 1e2]) 0; 0 tf(1e2,[1 1e2])];
    case {'X29_Lateral','X29_Lateral_Con'}
```

```
        Prefilter=inv(dcgain(To))*[tf(3,[1 3]) 0; 0 tf(1.9,[1 1.9])];
    otherwise
        Prefilter=tf(1e3,[1 1e3]);
end
Try_w = To*Prefilter;
Tru_w = KS*Prefilter;

%% Plots
FilePath=fullfile(FilePath0,FilePath1,FilePath2);
mkdir(FilePath);

if ShowPlot==1
    %     close all;
    PosX=550; PosY=100; SizeX=500; SizeY=390;
    f_Plots(P_ss,n_e,n_u,K,Lo,Li,So,Si,KS,PS,To,Ti,Try_w,Tru_w,W1,W2,W3,...
        Wd1,Wd2,Wd3,PosX,PosY,SizeX,SizeY,FreqMin,FreqMax,TFinal,...
        FilePath,FileType,WtPlt,SecondPlot,LegendName1,LegendName2);
end

%% Poles and zeros
if ShowDamp==1
    [PlntPole_DampFreq,PlntPole_Damp,PlntPole_DampPole,PlntZero_DampFreq...
        ,PlntZero_Damp,PlntZero_DampZero,KPole_DampFreq,KPole_Damp,...
        KPole_DampPole,KZero_DampFreq,KZero_Damp,KZero_DampZero,...
        ToPole_DampFreq,ToPole_Damp,ToPole_DampPole,ToZero_DampFreq,...
        ToZero_Damp,ToZero_DampZero]=f_Damp(P_ss,K,To);
end

%% Performance Measure

NormInf=mag2db([norm(So,inf), norm(Si,inf), norm(KS,inf), norm(PS,inf), ...
    norm(To,inf), norm(Ti,inf)]);

PerformMeasOutOrigWts=norm([W1*So; W2*KS; W3*To],inf);
PerformMeasInOrigWts=norm([Wd1*Si; Wd2*PS; Wd3*Ti],inf);
PerformMeasCombOrigWts=max(PerformMeasOutOrigWts,PerformMeasInOrigWts)

%% Save Workspace

if UseHinfSyn==1
    save(fullfile(FilePath,'SavWrkSpcHinfSyn.mat'));
else
    save(fullfile(FilePath,'SavWrkSpcGenHinf.mat'));
end
```

```
% ************************** f_Plots **********************************

function f_Plots(P_ss,n_e,n_u,K,Lo,Li,So,Si,KS,PS,To,Ti,Try_w,Tru_w,W1,...
    W2,W3,Wd1,Wd2,Wd3,PosX,PosY,SizeX,SizeY,FreqMin,FreqMax,TFinal,...
    FilePath,FileType,WtPlt,SecondPlot,LegendName1,LegendName2)

% Open loop tf's
if SecondPlot==0
    figure(1); sigma(P_ss,{FreqMin,FreqMax}); grid on; ...
        title('Plant Singular Value','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); ...
```

```
        set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    saveas(gcf,fullfile(FilePath,'Freq_Plnt_Sing'),FileType);
end

if SecondPlot==0
    [SVP,Ww]=sigma(P_ss,{FreqMin,FreqMax});
    figure(2); semilogx(Ww,mag2db(SVP(1,:)./SVP(end,:))); grid on; ...
        title('Plant Condition number (dB)','FontSize',12); ...
        xlabel('Frequency (rad/s)'); ylabel('Condition number (dB)');
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); ...
        set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    saveas(gcf,fullfile(FilePath,'Freq_Plnt_Cond'),FileType);
end

figure(3);
if SecondPlot==0
    sigma(K,{FreqMin,FreqMax}); grid on; title...
        ('Controller Singular Value','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); set...
        (gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(K,{FreqMin,FreqMax},'--r'); grid on; title...
        ('Controller Singular Value','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_K_Sing'),FileType);

[SVP,Ww]=sigma(K,{FreqMin,FreqMax});
figure(4);
if SecondPlot==0
    semilogx(Ww,mag2db(SVP(1,:)./SVP(end,:))); grid on; ...
        title('Controller Condition number (dB)','FontSize',12);
    xlabel('Frequency (rad/s)'); ylabel('Condition number (dB)');
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); set(gcf,...
        'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    semilogx(Ww,mag2db(SVP(1,:)./SVP(end,:)),'--r'); grid on;
    title('Controller Condition number (dB)','FontSize',12);
    xlabel('Frequency (rad/s)'); ylabel('Condition number (dB)');
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_K_Cond'),FileType);

figure(5);
if SecondPlot==0
    sigma(Lo,{FreqMin,FreqMax}); grid on; title...
        ('Frequency Response L_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
```

```
    sigma(Lo,{FreqMin,FreqMax},'--r'); grid on; title...
        ('Frequency Response L_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_Lo'),FileType);

figure(6);
if SecondPlot==0
    sigma(Li,{FreqMin,FreqMax}); grid on; title...
        ('Frequency Response L_i','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(Li,{FreqMin,FreqMax},'--r'); grid on; title...
        ('Frequency Response L_i','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_Li'),FileType);


% Closed loop tf's
for ii=1:n_e
    inv_W1(ii,ii)=inv(W1(ii,ii));
end
figure(7);
if SecondPlot==0
    sigma(So,{FreqMin,FreqMax}); grid on;
    if WtPlt==1
        hold on; sigma(inv_W1); legend('S_o','W_{1}^{-1}');
    end
    title('Frequency Response S_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(So,{FreqMin,FreqMax},'--r'); grid on;
    title('Frequency Response S_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_So'),FileType);

for ii=1:n_u
    inv_Wd1(ii,ii)=inv(Wd1(ii,ii));
end
figure(8);
if SecondPlot==0
    sigma(Si,{FreqMin,FreqMax}); grid on;
    if WtPlt==1
        hold on; sigma(inv_Wd1); legend('S_i','W_4^{-1}');
    end
```

```
    title('Frequency Response S_i (T_{d_iu_p})','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(Si,{FreqMin,FreqMax},'--r'); grid on;
    title('Frequency Response S_i (T_{d_iu_p})','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_Si'),FileType);


for ii=1:n_u
    inv_W2(ii,ii)=inv(W2(ii,ii));
end
figure(9);
if SecondPlot==0
    sigma(KS,{FreqMin,FreqMax}); grid on;
    if WtPlt==1
        hold on; sigma(inv_W2); legend('KS_o','W_{2}^{-1}');
    end
    title('Frequency Response KS_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(KS,{FreqMin,FreqMax},'--r'); grid on;
    title('Frequency Response KS_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_KS'),FileType);

figure(10);
if SecondPlot==0
    sigma(Tru_w,{FreqMin,FreqMax}); grid on; title...
        ('Frequency Response T_{ru} (with Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(Tru_w,{FreqMin,FreqMax},'--r'); grid on; title...
        ('Frequency Response T_{ru} (with Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_Tru'),FileType);

for ii=1:n_e
    inv_W3(ii,ii)=inv(W3(ii,ii));
end
figure(11);
if SecondPlot==0
```

```
    sigma(To,{FreqMin,FreqMax}); grid on;
    if WtPlt==1
        hold on; sigma(inv_W3);  legend('T_o','W_{3}^{-1}');
    end
    title('Frequency Response T_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(To,{FreqMin,FreqMax},'--r'); grid on;
    title('Frequency Response T_o','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_To'),FileType);

for ii=1:n_u
    inv_Wd3(ii,ii)=inv(Wd3(ii,ii));
end
figure(12);
if SecondPlot==0
    sigma(Ti,{FreqMin,FreqMax}); hold on; grid on;
    if WtPlt==1
        hold on; sigma(inv_Wd3);  legend('Ti','W_6^{-1}');
    end
    title('Frequency Response T_i  (T_{d_iu})','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(Ti,{FreqMin,FreqMax},'--r'); hold on; grid on;
    title('Frequency Response T_i  (T_{d_iu})','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_Ti'),FileType);

figure(13);
if SecondPlot==0
    sigma(Try_w,{FreqMin,FreqMax}); grid on; title...
        ('Frequency Response T_{ry} (with Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(Try_w,{FreqMin,FreqMax},'--r'); grid on; title...
        ('Frequency Response T_{ry} (with Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); set...
        (gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_Try'),FileType);

for ii=1:n_e
    inv_Wd2(ii,ii)=inv(Wd2(ii,ii));
```

```
end
figure(14);
if SecondPlot==0
    sigma(PS,{FreqMin,FreqMax}); grid on;
    if WtPlt==1
        hold on; sigma(inv_Wd2); legend('PS_i','W_5^{-1}');
    end
    title('Frequency Response PS_i=S_oP (T_{d_iy})','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); set...
        (gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    sigma(PS,{FreqMin,FreqMax},'--r'); grid on;
    title('Frequency Response PS_i=S_oP (T_{d_iy})','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2); set...
        (gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Freq_PS'),FileType);

figure(15);
if SecondPlot==0
    step(To,TFinal); grid on; title...
        ('Output Response (No Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    step(To,TFinal,'--r'); grid on; title...
        ('Output Response (No Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Step_To'),FileType);

figure(16);
if SecondPlot==0
    step(Try_w,TFinal); grid on; title...
        ('Output Response (With Prefilter)','FontSize',12); h = ...
        findobj(gcf,'type','line'); set(h,'linewidth',2); set...
        (gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    step(Try_w,TFinal,'--r'); grid on; title...
        ('Output Response (With Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Step_Try'),FileType);

figure(17);
if SecondPlot==0
    step(KS,TFinal); grid on; title...
        ('Control Response (No Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
```

```
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    step(KS,TFinal,'--r'); grid on; title...
        ('Control Response (No Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Step_KS'),FileType);

figure(18);
if SecondPlot==0
    step(Tru_w,TFinal); grid on; title...
        ('Control Response (With Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
else
    hold on;
    step(Tru_w,TFinal,'--r'); grid on; title...
        ('Control Response (With Prefilter)','FontSize',12);
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    legend(LegendName1,LegendName2);
end
saveas(gcf,fullfile(FilePath,'Step_Tru'),FileType);

if SecondPlot==0
    figure(19); sigma(inv_W1,{FreqMin,FreqMax}); hold on; grid on;
    sigma(inv_W2); sigma(inv_W3);
    title('Weights on Responses at Plant Output','FontSize',12);
    legend('W_{1}^{-1}','W_{2}^{-1}','W_{3}^{-1}');
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    saveas(gcf,fullfile(FilePath,'Freq_Ws'),FileType);
end

if SecondPlot==0
    figure(20); sigma(inv_Wd1,{FreqMin,FreqMax}); hold on;
    grid on; sigma(inv_Wd2); sigma(inv_Wd3);
    title('Weights on Responses at Plant Input','FontSize',12);
    legend('W_4^{-1}','W_5^{-1}','W_6^{-1}');
    h = findobj(gcf,'type','line'); set(h,'linewidth',2);
    set(gcf,'Position',[PosX,PosY,SizeX,SizeY]);
    saveas(gcf,fullfile(FilePath,'Freq_Wds'),FileType);
end
```

```
% ********************** conPEAK_MIMO_AllStep ****************************

function [value, sg, ConValVec, varargout] = conPEAK_MIMO_AllStep(M, x,...
    T11, T12, T21, Q, vec, varargin)
% Compute peak value and subgradients
% [value sg] = conPEAK(M, x, T11, T12, T21, Q, vec)
% M          :
% x          :
% T11        :
```

```
% T12       :
% T21       :
% Q         :
% vec       : location of objective function matrices in Twz
tvec = 0:0.001:10;% tvec = 0:0.001:5;
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

                [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;
```

```
% ************************** conHINF *******************************

function [value sg varargout] = conHINF(M, x, T11, T12, T21,Q,vec,varargin)
% Compute H-infinity norm and subgradients
% [value sg] = conHINF(M, x, T11, T12, T21, Q, vec)
```

```
% M          :
% x          :
% T11        :
% T12        :
% T21        :
% Q          :
% vec        : location of objective function matrices in Twz
if nargin == 8
 conval = varargin{1};
 varargout{1} = conval;
end
n = length(x);

[n_u, n_e, n_s] = size(Q);
Twz = parallel(T11,series(series(T21,Q),T12));
%Twz = minreal(Twz);
Twz = Twz(vec,:);

[ninf, fpeak] = norm(Twz, inf, 1e-8);
value = ninf;

Hjwo = freqresp(Twz,fpeak);
[U,S,V] = svd(Hjwo);     % SVD at W0
uo      = U(:,1);        % Maximum Left Singular Vector
vo      = V(:,1);        % Maximum Right Singular Vector
subgradient = [];
for i = 1:n
    Hjwo = freqresp(M{i},fpeak);
    magHjwo = abs(Hjwo);
    subgradient = [subgradient; real(uo'*Hjwo*vo)];
end
sg = subgradient;
%
% figure(1000)
% sigma(Twz)
% title(num2str(20*log10(value)), 'FontSize', 16)
% pause
```

```
% *************************** conBASIS ********************************

function q = conBASIS(N, p, z, basis_type)
% Form the basis
q{1} = tf(1,1);
if basis_type == 1  % fixed pole Laguerre
    for k=2:N
        q{k} = zpk([],-p,p)^(k-1);
    end
elseif basis_type == 2 % fixed pole inner
    for k=2:N
        q{k} = zpk(p,-p,-1)^(k-1);
    end
elseif basis_type == 3 % variable pole first order term
    for k=2:N
        q{k} = zpk([],-p*(k-1),p*(k-1));
    end
elseif basis_type == 4 % variable pole first order term inner
```

```
    for k=2:N
        q{k} = zpk(p*(k-1),-p*(k-1),-1);
    end
elseif basis_type == 5 % variable pole first order term inner
    for k=2:N
        q{k} = zpk(z,-p,-1)^(k-1);
    end
end
```

```
% ************************** conFORMQN ********************************

function QN = conFORMQN(x, qk, n_u, n_e, N)
% From Q_N
% QN = conFORMQN(x, qk, n_u, n_e, N)
% INPUTS:
% x          : optimization variable (vector)
% qk         : basis (cell array of transfer functions, zpk)
% n_u        : number of control inpts
% n_e        : number of measurements
% N          : basis order
% OUTPUT:
% QN         : QN (state space)
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);
    temp = QN + X{i} * qk{i};
    QN = minreal(temp);
end
QN = ss(QN);
```

```
% ************************* conORGANIZE_Gen ******************************

function [n_e, n_u, DATArz,DATAdz] = conORGANIZE_Gen(P, W1, W2, W3, Wd1,...
    Wd2, Wd3, W1c, W2c, W3c, Wd1c, Wd2c, Wd3c)
% Extract data from problem setup
% DATArz.ObjVec
% DATArz.ConVec{ConstraintCounter} = TotalRows+1:TotalRows+n_e;
% DATArz.ConNam{ConstraintCounter} = W3c{i}.Fun;
% DATArz.ConVal{ConstraintCounter} = W3c{i}.Val;
% DATArz.ConNum = ConstraintCounter;

[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;
```

```matlab
    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;
```

```matlab
% *************************** conVECTORIZE ********************************

function [M Mobj Mcon] = conVECTORIZE(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 = 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);
            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
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
```

```
% ************************** f_CLTFM **********************************

function [Lo,Li,So,Si,To,Ti,KS,PS] = f_CLTFM(P,K)

[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_ps = [Ak-Bk*Dp*Mi*Ck Bk*Dp*Mi*Dk*Cp-Bk*Cp; Bp*Mi*Ck Ap-Bp*Mi*Dk*Cp];
B_ps = [-Bk*Dp*Mi; Bp*Mi];
C_ps = [Mo*Dp*Ck Mo*Cp];
D_ps = Mo*Dp;
PS = ss(A_ps,B_ps,C_ps,D_ps);
```

```
% *************************** f_Damp ***********************************

function [PlntPole_DampFreq,PlntPole_Damp,PlntPole_DampPole,...
    PlntZero_DampFreq,PlntZero_Damp,PlntZero_DampZero,KPole_DampFreq,...
    KPole_Damp,KPole_DampPole,KZero_DampFreq,KZero_Damp,KZero_DampZero,...
    ToPole_DampFreq,ToPole_Damp,ToPole_DampPole,ToZero_DampFreq,...
    ToZero_Damp,ToZero_DampZero]=f_Damp(P_ss,K,To)

```

```
disp('Plant Poles'); damp(pole(P_ss))
[PlntPole_DampFreq,PlntPole_Damp,PlntPole_DampPole]=damp(pole(P_ss));
disp('Plant zeros');
x=sym('x'); PZeros=solve(det([x*eye(size(P_ss.a))-P_ss.a -P_ss.b; ...
    P_ss.c P_ss.d])==0); damp(double(PZeros))
[PlntZero_DampFreq,PlntZero_Damp,PlntZero_DampZero]=damp(double(PZeros));

disp('Controller order'); order(K)
disp('Controller poles'); damp(pole(K))
[KPole_DampFreq,KPole_Damp,KPole_DampPole]=damp(pole(K));
disp('Controller zeros');
KZeros=solve(det([x*eye(size(K.a))-K.a -K.b; K.c K.d])==0);
damp(double(KZeros))
[KZero_DampFreq,KZero_Damp,KZero_DampZero]=damp(double(KZeros));
% disp('Controller order No Minreal'); order(K1)
% disp('Controller pole No Minreal'); damp(pole(K1))
% disp('Controller zero No Minreal'); damp(tzero(K1))
% CLOSED LOOP
disp('CLOSED LOOP POLES'); damp(pole(To))
[ToPole_DampFreq,ToPole_Damp,ToPole_DampPole]=damp(pole(To));
disp('CLOSED LOOP ZEROS');
ToZeros=solve(det([x*eye(size(To.a))-To.a -To.b; To.c To.d])==0);
damp(double(ToZeros))
% damp(tzero(To))
[ToZero_DampFreq,ToZero_Damp,ToZero_DampZero]=damp(double(ToZeros));
```