

This is an explanation of the program from 1996. Some of the Matlab commands may be old and need to be updated. Send comments and questions to Hitay Özbay: (hitay@bilkent.edu.tr)

A Matlab Based Program for \mathcal{H}^∞ Optimal/Suboptimal Controller Design

This brief paper is intended to be a user guide for the Matlab based program called **HINFCON**, which implements the algorithm given in [5]. This program computes the \mathcal{H}^∞ optimal performance and generates optimal and suboptimal controllers in the mixed sensitivity minimization/reduction problems for SISO infinite dimensional systems. For a copy of [5] send an e-mail to ozbay@ee.eng.ohio-state.edu or tokero@ee.eng.ohio-state.edu).

1 Controller Formulae

In this section the controller formulae, which are originally given in [5], are presented. In Sections 2–4, the reader will be assumed to be familiar with the below formulae.

1.1 \mathcal{H}^∞ optimal and suboptimal mixed sensitivity

The plant and the controller are represented by their transfer functions $P(s)$ and $C(s)$, respectively. A system, whose transfer function is $G(s)$, will be said to be stable if $G \in \mathcal{H}^\infty(\mathbb{C}_+)$.

1.1.1 Assumptions on the plant

It is assumed that the plant does not have any poles on the imaginary axis. If this assumption is not satisfied originally, one can replace $\mathcal{H}^\infty(\mathbb{C}_+)$ by $\mathcal{H}^\infty(\mathbb{C}_\sigma)$ for some small $\sigma > 0$, in the definition of stability. Then the modified plant, $P(s - \sigma)$, satisfies this assumption, provided $P(s)$ has finitely many poles in \mathbb{C}_σ , and no poles on $\text{Re}(s) = -\sigma$. With respect to the original stability definition, this situation corresponds to stability with a margin of at least σ . For simplicity it will also be assumed that the poles of $P(s)$ in \mathbb{C}_+ , denoted by $\alpha_1, \dots, \alpha_l$, are distinct. With this assumption, $P(s)$ can be written as $N_p(s)/m_d(s)$, where

$$m_d(s) = \prod_{k=1}^l \frac{s - \alpha_k}{s + \alpha_k},$$

and $N_p \in \mathcal{H}^\infty(\mathbb{C}_+)$. It is well known that $\mathcal{H}^\infty(\mathbb{C}_+)$ functions admit inner outer factorizations, [2]. Therefore, N_p can be factored as $N_p = m_n N_o$, where $m_n \in \mathcal{H}^\infty(\mathbb{C}_+)$ is

inner (i.e. all-pass function, possibly infinite dimensional) and $N_o \in \mathcal{H}^\infty(\mathbb{C}_+)$ is outer (i.e. minimum phase, possibly infinite dimensional). This factorization is unique up to multiplication by -1 . It will also be assumed that N_o can further be factored as $N_o = N_{o1}N_{o2}$ where $N_{o1}, N_{o1}^{-1}, N_{o2} \in \mathcal{H}^\infty(\mathbb{C}_+)$, and N_{o2} is rational. This factorization is not unique. But it will only be necessary to conclude that the optimal/suboptimal \mathcal{H}^∞ controller obtained here is proper. It is easy to see that if two plants P_1 and P_2 satisfy the above assumptions, then so does their product P_1P_2 .

It should be noted that inner outer factorization of an arbitrary function in $\mathcal{H}^\infty(\mathbb{C}_+)$ may require infinite dimensional spectral factorization, e.g. N_o is the spectral factor of $N_p(-s)N_p(s) = N_o(-s)N_o(s)$. However, in several interesting cases it is possible to obtain the inner outer factorization by inspection. A delay system example is given below.

Example: ([3]) Following plant satisfies above assumptions:

$$P(s) = \frac{e^{-hs}(s - 0.05)}{(s + 1)(s + 0.1 - e^{-h_1s})}, \quad h_1 = 2 \ln\left(\frac{5}{3}\right), \quad h > 0.$$

Note that the only point in $\overline{\mathbb{C}}_+$ where the term $(s + 0.1 - e^{-h_1s})$ becomes zero is $s = 0.5$. So, the plant has only one pole in \mathbb{C}_+ . It is easy to verify that the multiplicity of this pole is one. Therefore, in this example factors of P can be identified as follows:

$$\begin{aligned} m_n(s) &= e^{-hs} \frac{s - 0.05}{s + 0.05} & m_d(s) &= \frac{s - 0.5}{s + 0.5} \\ N_{o1}(s) &= \frac{(s - 0.5)(s + 0.05)}{(s + 0.1 - e^{-h_1s})(s + 0.5)} & N_{o2}(s) &= \frac{1}{s + 1}. \end{aligned}$$

1.1.2 Problem statement

The problem dealt here is the standard mixed sensitivity minimization. Let $W_1(s)$ and $W_2(s)$ be rational weighting functions. For properness of the optimal controller it will be assumed that $W_1(s)$ is non-constant and $W_1, W_1^{-1} \in \mathcal{H}^\infty(\mathbb{C}_+)$, and that $(W_2N_{o2}), (W_2N_{o2})^{-1} \in \mathcal{H}^\infty(\mathbb{C}_+)$, see [3] for a detailed discussion on these assumptions, and relations to other types of two block \mathcal{H}^∞ control problems.

The optimal \mathcal{H}^∞ performance is defined by

$$\gamma_o := \inf_{C \text{ stabilizes } P} \left\| \begin{bmatrix} W_1S \\ W_2T \end{bmatrix} \right\|_\infty$$

where $S = (1 + PC)^{-1}$ and $T = PC(1 + PC)^{-1}$ are the sensitivity and complementary sensitivity functions. The condition “ C stabilizes P ” means that closed loop transfer functions S , CS and PS must belong to $\mathcal{H}^\infty(\mathbb{C}_+)$. The optimal \mathcal{H}^∞ controller, denoted by C_{opt} , is the one which stabilizes the plant P , and yields

$$\left\| \begin{bmatrix} W_1(1 + PC_{opt})^{-1} \\ W_2PC_{opt}(1 + PC_{opt})^{-1} \end{bmatrix} \right\|_\infty = \gamma_o.$$

The suboptimal \mathcal{H}^∞ control problem is to parametrize the set

$$\mathcal{C}_\rho = \left\{ C \quad : \quad C \text{ stabilizes } P, \quad \left\| \begin{bmatrix} W_1S \\ W_2T \end{bmatrix} \right\|_\infty \leq \rho \right\} \quad (1)$$

for a given $\rho > \gamma_o$.

1.2 Optimal and suboptimal \mathcal{H}^∞ controllers

Let $\eta_1, \dots, \eta_{n_1} \in \overline{\mathbb{C}}_+$, $n_1 \geq 1$, be the poles of $W_1(-s)$; if η_i has multiplicity ℓ_i then it is assumed to be repeated ℓ_i times in this list. The zeros of

$$E_\rho(s) := \left(\frac{W_1(-s)W_1(s)}{\rho^2} - 1 \right) \quad (2)$$

are denoted by $\beta_1, \dots, \beta_{2n_1}$, and they are assumed to be distinct. Then, β_i 's can be enumerated in such a way that $\beta_1, \dots, \beta_{n_1}$ are in $\overline{\mathbb{C}}_+$, and $\beta_{n_1+i} = -\beta_i$. Now define

$$F_\rho(s) := G_\rho(s) \prod_{k=1}^{n_1} \frac{s - \eta_k}{s + \eta_k} \quad (3)$$

where $G_\rho \in \mathcal{H}^\infty(\mathbb{C}_+)$ is minimum phase and determined from the spectral factorization

$$G_\rho(s)G_\rho(-s) := \left(1 - \left(\frac{W_1(-s)W_1(s)}{\rho^2} - 1 \right) \left(\frac{W_2(-s)W_2(s)}{\rho^2} - 1 \right) \right)^{-1}. \quad (4)$$

Then (under certain genericity assumptions, see [5]) the optimal \mathcal{H}^∞ controller is given by

$$C_{opt}(s) = E_{\gamma_o}(s)m_d(s) \frac{N_o(s)^{-1}F_{\gamma_o}(s)L(s)}{1 + m_n(s)F_{\gamma_o}(s)L(s)} \quad (5)$$

where $L(s) = L_2(s)/L_1(s)$ with $L_1(s)$ and $L_2(s)$ being polynomials of degrees less than or equal to $(n_1 + l - 1)$, satisfying interpolation conditions

$$0 = L_1(\beta_k) + m_n(\beta_k)F_{\gamma_o}(\beta_k)L_2(\beta_k) \quad k = 1, \dots, n_1 \quad (6)$$

$$0 = L_1(\alpha_k) + m_n(\alpha_k)F_{\gamma_o}(\alpha_k)L_2(\alpha_k) \quad k = 1, \dots, l \quad (7)$$

$$0 = L_2(-\beta_k) + m_n(\beta_k)F_{\gamma_o}(\beta_k)L_1(-\beta_k) \quad k = 1, \dots, n_1 \quad (8)$$

$$0 = L_2(-\alpha_k) + m_n(\alpha_k)F_{\gamma_o}(\alpha_k)L_1(-\alpha_k) \quad k = 1, \dots, l. \quad (9)$$

Note that (6–7) corresponds to interpolation conditions that the denominator term $(1 + m_n(s)F_{\gamma_o}(s)L(s))$ must cancel the closed right half plane zeros of $E_{\gamma_o}(s)m_d(s)$. This means, in particular, that $m_d(s)$ term in the numerator of $C_{opt}(s)$ does not cancel the unstable poles of the plant. Moreover, since $(W_2N_o)^{-1} \in \mathcal{H}^\infty(\mathbb{C}_+)$ and $W_1, W_1^{-1} \in \mathcal{H}^\infty(\mathbb{C}_+)$, the term $F_{\gamma_o}N_o^{-1}$ is proper. It should also be noted that (6–9) constitute $2(n_1 + l)$ linear homogenous equations in $2(n_1 + l)$ unknown coefficients of $L_1(s)$ and $L_2(s)$.

If γ_o is replaced by a variable, say γ , in equations (6–9), then a new set of linear homogenous equations is obtained, in terms of $2(n_1 + l)$ unknown coefficients, for each fixed γ . Under certain genericity assumptions, γ_o is the largest value of γ for which there is a non-trivial solution to these $2(n_1 + l)$ linear homogenous equations. In other words, γ_o can be found by plotting the smallest singular value of the matrix representation of these equations, as γ varies in an interval, see example given in the Appendix; the largest value of γ for which the plot shows a zero is γ_o . If there is no such γ , then γ_o is equal to the essential norm of the associated infinite rank Hankel operator which can be computed very easily (see [5]). This situation can be avoided by proper choice of the weighting functions.

All suboptimal \mathcal{H}^∞ controllers are in the form

$$C_{subopt}(s) = E_\rho(s)m_d(s) \frac{N_o(s)^{-1}F_\rho(s)L_U(s)}{1 + m_n(s)F_\rho(s)L_U(s)} \quad (10)$$

where

$$L_U(s) = \frac{L_2(s) + L_1^r(s)U(s)}{L_1(s) + L_2^r(s)U(s)} \quad , \quad U \in \mathcal{H}^\infty(\mathbb{C}_+) \quad , \quad \|U\|_\infty \leq 1,$$

$L_1^r(s) = (-1)^{n_1+l}L_1(-s)$, $L_2^r(s) = (-1)^{n_1+l}L_2(-s)$, and $L_1(s), L_2(s)$ are polynomials of degree $\leq n_1 + l$ satisfying (6–9) with γ_o replaced by ρ , and the following two conditions:

$$0 = L_2(-a) + (E_\rho(a) + 1)F_\rho(a)m_n(a)L_1(-a) \quad (11)$$

$$1 = L_1(-a), \quad (12)$$

for some arbitrary $a \in \mathbf{R}$, and $a > 0$, (a is related to conformal map used in [5]). For different values of a one can obtain different parametrizations of all suboptimal controllers.

1.3 Example: Computation of the Optimal Performance

An explicit formula for the optimal \mathcal{H}^∞ controller is given above. In order to find C_{opt} , one needs to compute the optimal performance γ_o and corresponding $L(s)$. Now a delay system example will be considered to illustrate the computation procedure.

For $P(s) = e^{-hs}/(s-1)$, choose $W_1(s) = 2(s+1)/(10s+1)$ and $W_2(s) = 0.2(s+1.1)$. The same \mathcal{H}^∞ optimal control problem has been studied in [1] with slightly different weights. In this example $m_n(s) = e^{-hs}$, $m_d(s) = (s-\alpha_1)/(s+\alpha_1)$ and $N_o(s) = 1/(s+\alpha_1)$, where $\alpha_1 = 1$. Then, since $n_1 = 1$ and $l = 1$, polynomials $L_1(s)$ and $L_2(s)$ are both first order. Therefore, there are four unknown coefficients, $L_{10}, L_{20}, L_{11}, L_{21}$, where

$$L_1(s) =: L_{11}s + L_{10} \quad \text{and} \quad L_2(s) =: L_{21}s + L_{20}.$$

For $h > 0$, a lower bound for γ_o can be found as $1/5$. An upper bound can also be found (see e.g. [1, 4]). When $h = 0.2$, for slightly different weights, γ_o has been computed in [1] as $\gamma_o \approx 0.6667$. So, for this example one expects that γ_o is close to 0.67, and in particular $\gamma_o \leq 1.5$. In the range $0.2 < \gamma < 1.5$,

$$E_\gamma(s) = \frac{(4 - \gamma^2) + (100\gamma^2 - 4)s^2}{\gamma^2(1 - 100s^2)}$$

has two zeros on the imaginary axis. Only one of them, say $\beta_1 = j\sqrt{\frac{4-\gamma^2}{100\gamma^2-4}}$, is used in the formulae (6-9). Note also that, in the above range of γ , F_γ is given by

$$F_\gamma(s) = \frac{\gamma^2(1 - 10s)}{(s + p_1)(s + p_2)\sqrt{4\gamma^2 - 0.16}},$$

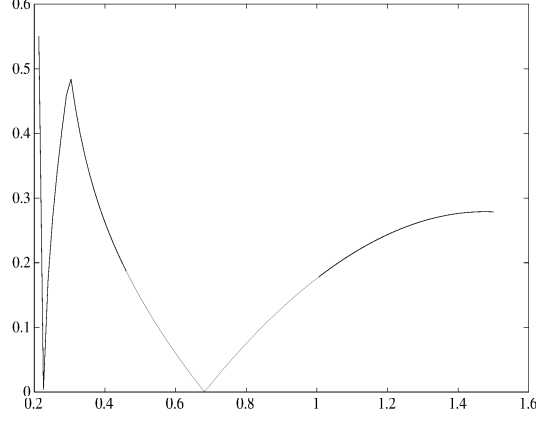


Figure 1: $\sigma_{\min}(M_\gamma)$ versus γ

where $p_1 = \sqrt{b + \sqrt{b^2 - c}}$ and $p_2 = \sqrt{b - \sqrt{b^2 - c}}$, with

$$b = \frac{8.88\gamma^2 - 0.3536}{8\gamma^2 - 0.32} \quad \text{and} \quad c = \frac{4.0484\gamma^2 - 0.1936}{4\gamma^2 - 0.16}.$$

The set of equations (6–9) can be written as $M_\gamma \Psi = 0$, where

$$M_\gamma := \begin{bmatrix} 1 & \beta_1 & m_n(\beta_1)F_\gamma(\beta_1) & \beta_1 m_n(\beta_1)F_\gamma(\beta_1) \\ 1 & \alpha_1 & m_n(\alpha_1)F_\gamma(\alpha_1) & \alpha_1 m_n(\alpha_1)F_\gamma(\alpha_1) \\ m_n(\beta_1)F_\gamma(\beta_1) & -\beta_1 m_n(\beta)F_\gamma(\beta_1) & 1 & -\beta_1 \\ m_n(\alpha_1)F_\gamma(\alpha_1) & -\alpha_1 m_n(\alpha_1)F_\gamma(\alpha_1) & 1 & -\alpha_1 \end{bmatrix}$$

and $\Psi := [L_{10}, L_{11}, L_{20}, L_{21}]^T$.

The largest value of γ which makes M_γ singular is $\gamma_o = 0.6819$; it is obtained from Figure 1, where the smallest singular value of M_γ is plotted. Once γ_o is obtained from this plot, a non-zero Ψ_o , which satisfies $M_{\gamma_o} \Psi_o = 0$, can be easily obtained. The entries of Ψ_o give $L(s) = L_2(s)/L_1(s)$. For the above example $L(s)$ can be computed as

$$L(s) = \frac{s + 0.2129}{s - 0.2129}.$$

2 About the Matlab Program

It is assumed that the following matlab m-files are in one directory:

No.m
No1.m
No2.m
P.m
U.m
arrow.m
data.m
eps2.m
eps4.m
md.m
mn.m
opt.m
opt2.m
polyadd.m
pspec.m
spec.m
star.m
subopt.m

The user is not allowed to modify the m-files **No.m**, **arrow.m**, **eps2.m**, **eps4.m**, **opt.m**, **opt2.m**, **spec.m**, **star.m**, **subopt.m**, **polyadd.m**, **pspec.m**, except the value of **EPS** in **polyadd.m**, **pspec.m** (suggested value for **EPS** is 10^{-6}). The program **polyadd.m** adds two polynomials and **pspec.m** performs spectral factorization on polynomials.

3 How to enter $P(s)$, $W_1(s)$, $W_2(s)$ etc. ?

The user should enter functions by modifying the corresponding m-files in the obvious way, i.e. by typing a legal Matlab expression for the function definition. The user should enter $P(s)$ to **P.m**, $N_{o1}(s)$ to **No1.m**, $N_{o2}(s)$ to **No2.m**, $m_d(s)$ to **md.m**, and $m_n(s)$ to **mn.m**. For the suboptimal controller design problem, the user should also write $U(s)$ to **U.m** for the program to generate the Bode plot of the controller and the Nyquist

plot of the system.

For example if the plant is $P(s) = e^{-0.2s}/(s - 1)$, then **P.m** file must be:

```
function y=P(s)
y=exp(-0.2*s)/(s-1);
```

To enter $W_1(s)$ and $W_2(s)$, the file **data.m** should be edited appropriately. For example, if $W_1(s) = 2(s + 1)/(10s + 1)$ and $W_2(s) = 0.2(s + 1.1)$ then **data.m** must be as below:

```
function [nw1,dw1,nw2,dw2,alpha,Npoints,amp,lw1,lw2,Lpoints,epsn]=data(),
%
% W1(s)=nw1(s)/dw1(s)
%
nw1 = 2*[1 1];
dw1 = [10 1];
%
% W2(s)=nw2(s)/dw2(s)
%
nw2 = 0.2*[1 1.1];
dw2 = 1;
.....
.....
```

There are also other variables in **data.m**, that must be set before running the program: **alpha** is the vector which contains the unstable poles of the plant $P(s)$; **Npoints** is the number of points in the min singular value versus γ plot (suggested value for **Npoints** is 100); **amp** is the conformal map parameter used in the bilinear map (required only for the suboptimal case). The user should set **amp** to a random number, because some **amp** values *may* violate the genericity conditions. The variables **lw1**, **lw2**, **Lpoints** are used for Bode, Nyquist and performance plots, **lw1** is $\log_{10}(\omega_{min})$, **lw2** is $\log_{10}(\omega_{max})$, and **Lpoints** is the number of points in these plots. Finally, **epsn** is the epsilon value used for null-space computations for the optimal control problem (suggested values for **epsn** are 10^{-2} or 10^{-3}).

4 How to run the program ?

First, the user should run m-files **eps2.m** and **eps4.m** which give a lower bound **eps2** and an upper bound **eps4** for γ values to be used in **opt.m**. Then, the user should run **opt.m**, where the program will ask **gmin** and **gmax**. At this point the user should enter the values **eps2** and **eps4** for **gmin** and **gmax** respectively. Then, by looking at the plot of min singular value versus γ , the user may rerun **opt.m** and enter a smaller interval for γ_o search.

The program **opt.m** will give **gamma_min**, which is the max γ value at which min singular value is zero. Then, the user should run **opt2.m**, where the program will ask **GAMMA_OPT**. Here, the user should enter **gamma_min** value for **GAMMA_OPT**. Then, **opt2.m** gives the polynomials **nL**, **dL**, **nF**, **dF** where $L(s) = \mathbf{nL}/\mathbf{dL}$ and $F(s) = \mathbf{nF}/\mathbf{dF}$. Following this, program will give the performance plot, which must be flat with constant value equal to **GAMMA_OPT**. If this is not the case, then at that gamma value, i.e. at $\gamma = \mathbf{gamma_min}$, genericity conditions are violated. The user can check whether the genericity conditions are violated at $\gamma = \mathbf{gamma_min}$ as follows:

- (1) If program gives a warning message.
- (2) If the performance plot is not flat.
- (3) After running **opt2.m**, if **beta** vector contains elements which are close to zero, or contains two elements which are close to each other.
- (4) If changing the plant parameters by a small amount, and keeping weighting functions the same, does not change the value of **gamma_min**.

If any of the above situations occur, the user should suspect that genericity conditions are violated and should rerun **opt.m** for **gmin=eps2** and **gmax=gamma_min - 0.01**, and search γ_o in this interval.

To design a suboptimal controller **subopt.m** should be used. The program will ask a value for **rho**, and then it will compute the polynomials **L1**, **L2**, **nF**, **dF** where $F(s) = \mathbf{nF}/\mathbf{dF}$. The program also generates Bode, Nyquist and performance plots using $U(s)$ defined in **U.m**. If these plots are not required, the user may wish to stop the program after **L1**, **L2**, **nF**, **dF** are computed.

Appendix

A Delay System Example

In this appendix, a mixed sensitivity minimization problem is solved for a delay system using the program **HINFCON**. Let the problem data be given as below,

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

then

$$m_n(s) = e^{-0.2s}, \quad m_d(s) = \frac{s-1}{s+1}, \quad N_{o1}(s) = 1, \quad , N_{o2}(s) = \frac{1}{s+1}$$

One should enter $P(s), m_n(s), m_d(s), N_{o1}(s), N_{o2}(s), U(s)$ to the m-files **P.m, mn.m, md.m, No1.m, No2.m, U.m** respectively, as shown below.

```
-----
% P.m
function y=P(s)
y=exp(-0.2*s)/(s-1);
```

```
-----
% mn.m
function y=mn(s),
y=exp(-0.2*s);
```

```
-----
% md.m
function y=md(s)
y=(s-1)/(s+1);
```

```
-----
% No1.m
```

```
function y=No1(s)
y=1;
```

```
-----

% No2.m
function y=No2(s)
y=1/(s+1);
```

```
-----

% U.m (Used only for the suboptimal design)
% U(s) can be any stable function whose infinity norm is  $\leq 1$ 
function y=U(s);
y=0.7;
```

The **data.m** file must contain $W_1(s) = \mathbf{nw1}/\mathbf{dw1}$, $W_2(s) = \mathbf{nw2}/\mathbf{dw2}$, **alpha** (the vector which contains the unstable poles of the plant $P(s)$), **Npoints**, **amp**, **lw1**, **lw2**, **Lpoints**, **epsn**. The m-file **data.m** must be as below:

```
% data.m
function [nw1,dw1,nw2,dw2,alpha,Npoints,amp,lw1,lw2,Lpoints,epsn]=data(),
%
% W1(s)=nw1(s)/dw1(s)
%
nw1 = 2*[1 1];
dw1 = [10 1];
%
% W2(s)=nw2(s)/dw2(s)
%
nw2 = 0.2*[1 1.1];
dw2 = 1;
```

```

%
% alpha = Unstable poles
%
alpha = [1];
%
% Npoints = Number of points for "gamma search"
%
Npoints = 100;
%
% amp = conformal map parameter
%
amp = 2.3456;
%
% for Bode plot, Nyquist plot and performance plots
%
% lw1 = log(wmin), lw2 = log(wmax), Lpoints = Number of points
%
lw1 = -2;
lw2 = +5;
Lpoints = 100;
%
% epsn = epsilon value for null(M)
%
epsn=1e-2;

```

Now, **eps2.m** and **eps4.m** can be run. The program gives the following result after running **eps2.m**:

```

eps2 =
    0.2187
>>

```

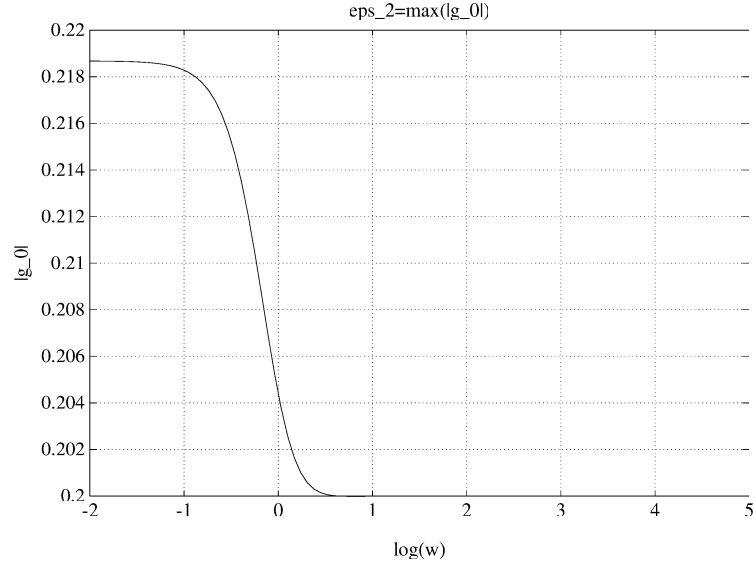


Figure 2: eps2 plot

The m-file **eps2.m** also gives the plot of $|g_0(j\omega)|$ versus $\log_{10}(\omega)$, see Figure 2. This program computes the maximum of $|g_0(j\omega)|$.

The user obtains the following output by running **eps4.m**

```
eps4 =
    42.0820

>>
```

The program **eps4.m** also gives the plot of $|g_0(j\omega)| + |w_0(j\omega) + \widehat{w}_0(j\omega)|$ versus $\log_{10}(\omega)$, see Figure 3. This program computes the maximum of $|g_0(j\omega)| + |w_0(j\omega) + \widehat{w}_0(j\omega)|$.

If the user runs **opt.m** with

```
Min gamma value = 0.2187
Max gamma value = 42.0820
```

then the plot will show no zero above $\gamma = 3.4$.

Now the user should rerun **opt.m** to search for γ_o in the following interval:

```
Min gamma value = 0.2187
Max gamma value = 3.4064
```

Note that 3.4064 is chosen to be slightly larger than 3.4.

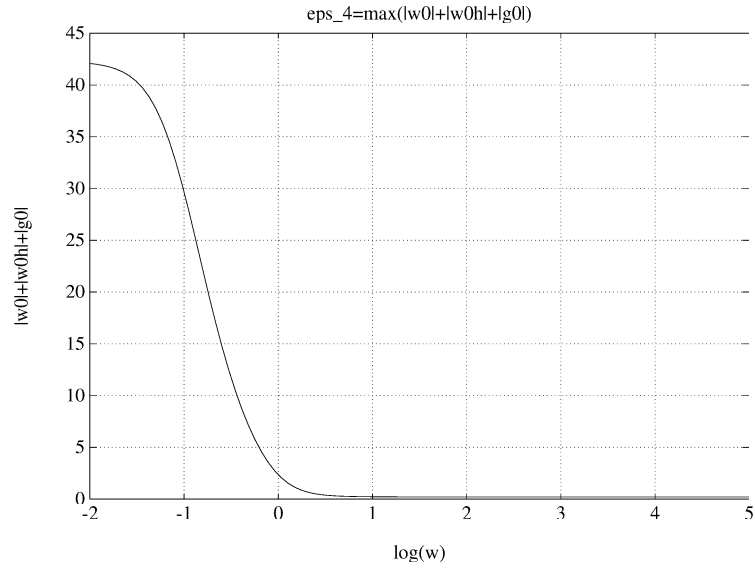


Figure 3: eps4 plot

The output of the program will be:

```
.....
.....
gamma_min =
    0.6969
```

along with the plot shown in Figure 4.

The user should expect that $\gamma_o = 2$ and rerun **opt.m** for

```
Min gamma value = 1.9
Max gamma value = 2.1
```

Program output will be:

```
.....
.....
gamma =
    2
warning =
```

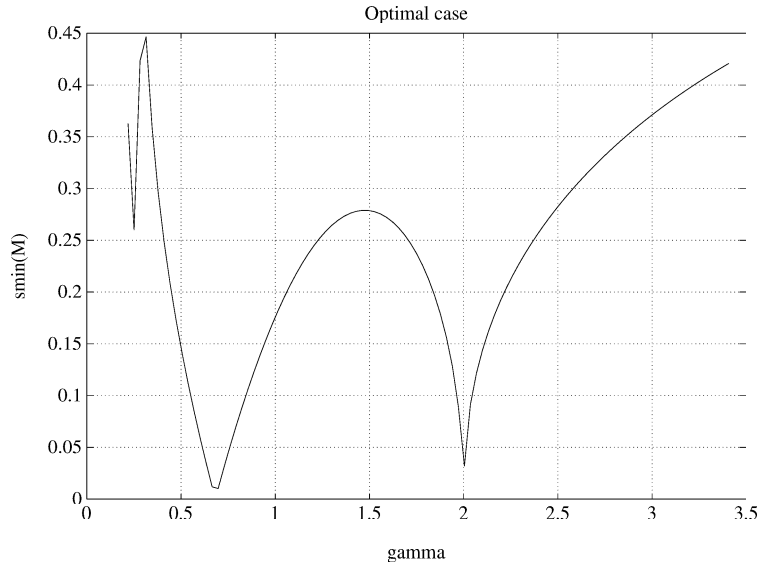


Figure 4: Search for γ_o

Non-generic case and/or illegal GAMMA value

message =

Press ENTER/RETURN to continue

If the user continues the program, output will be:

.....

.....

gamma_min =

2.0020

Then, the user should run **opt2.m** at which point the program asks

GAMMA_OPT =

If user enters 2.0020, after the end of the program **beta** will be

>> beta

beta =

0.0045

which is close to zero and user should suspect that genericity conditions are violated.

Elements of **beta** must be distinct and nonzero.

This shows that at that point, genericity conditions are violated. In this case, user should check the neighborhood of 0.7 for γ_o .

On the other hand if the user reruns **opt.m** for

```
Min gamma value = 1.9
```

```
Max gamma value = 2.2
```

Program output will be:

```
.....
```

```
.....
```

```
gamma_min =
```

```
1.9990
```

Then, the user should run **opt2.m** at which point the program asks

```
GAMMA_OPT =
```

If user enters 1.9990, after the end of the program **beta** will be

```
>> beta
```

```
beta =
```

```
0 + 0.0032i
```

which is close to zero and user should suspect that genericity conditions are violated. Elements of **beta** must be distinct and nonzero.

From the above results, it can be deduced that at $\gamma = 2$, at least one of the genericity conditions are violated. In order to determine whether $2 = \gamma_o$ or not, one can increase the value of h slightly (for general plants replace $m_n(s)$ by $e^{-\epsilon s}m_n(s)$, where ϵ is a “small” positive number), and obtain the new minimum singular value versus γ plot: if the new plots still shows **gamma_min**= 2 then one concludes that $\gamma_o \neq 2$, otherwise $\gamma_o = 2$. This is true because a small increase in time delay should increase the optimal performance.

For our example the reader should verify that $2 \neq \gamma_o$. Now, since the second largest γ value for which the plot 4 shows a zero is near 0.7, the user should rerun **opt.m** with

```
Min gamma value = 0.6
```

```
Max gamma value = 0.8
```

The program will give:

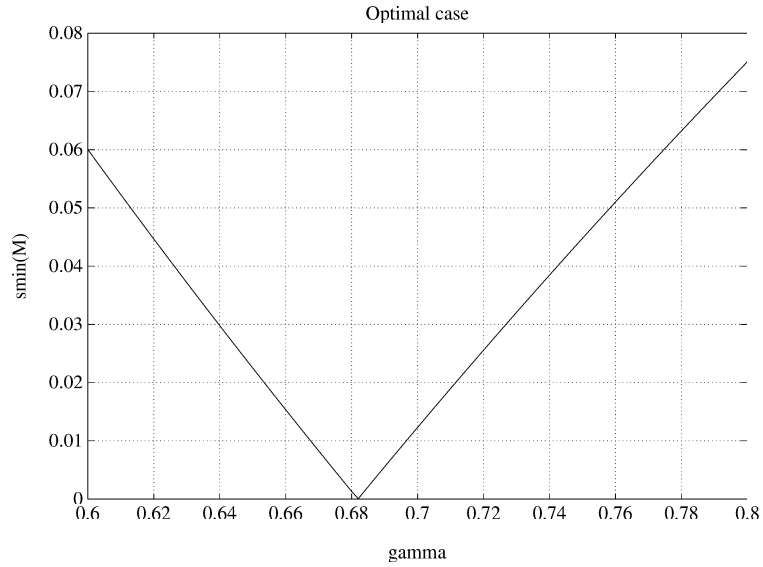


Figure 5: Search for γ_o

```
.....
.....
gamma_min =
    0.6820
```

and the plot shown in Figure 5.

Therefore, it can be concluded that $\gamma_o = 0.6820$. In order to generate the optimal controller the user should now run **opt2.m** by entering

```
GAMMA_OPT = 0.6820
```

The program output will be

```
nL =
    0.6916 + 0.0003i
    0.1472 + 0.0000i
dL =
   -0.6916 - 0.0004i
    0.1472 + 0.0001i
```

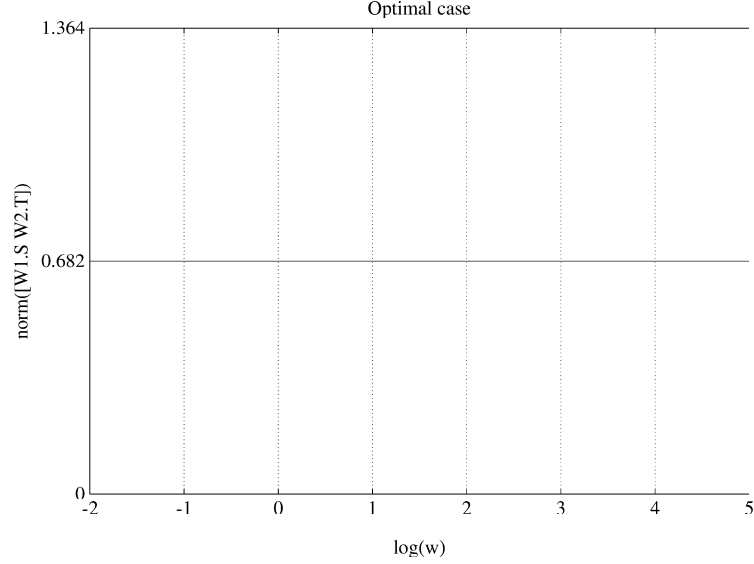


Figure 6: Performance plot

```

nF =
    1.0000    -0.0000   -0.0100

dF =
    0.2804    0.6036    0.3370    0.0279

```

These vectors determine the rational functions appearing in the optimal controller formula. The performance plot for the closed loop system is shown in Figure 6. The program will also generate the Bode plot for the controller, and the Nyquist plot of the system, these are shown in Figure 7, and Figure 8, respectively.

To design a suboptimal controller, the user should run **subopt.m** at which point the program asks

```
RHO =
```

the user should enter a value $> \gamma_o$. Of course, if the value 2.0 is entered, the program will give a warning and terminate. On the other hand, if the user enters a value close to 2.0 but not equal, than after the end of program, **beta** will contain an element which is close to zero and it will be concluded that this point violates the genericity conditions.

For **RHO** = 0.7 program will work properly and the output will be:

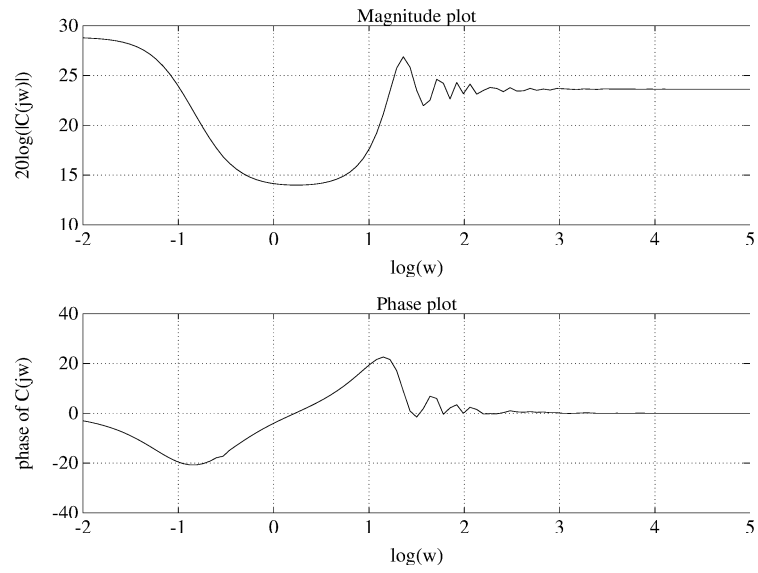


Figure 7: Bode plot of the controller

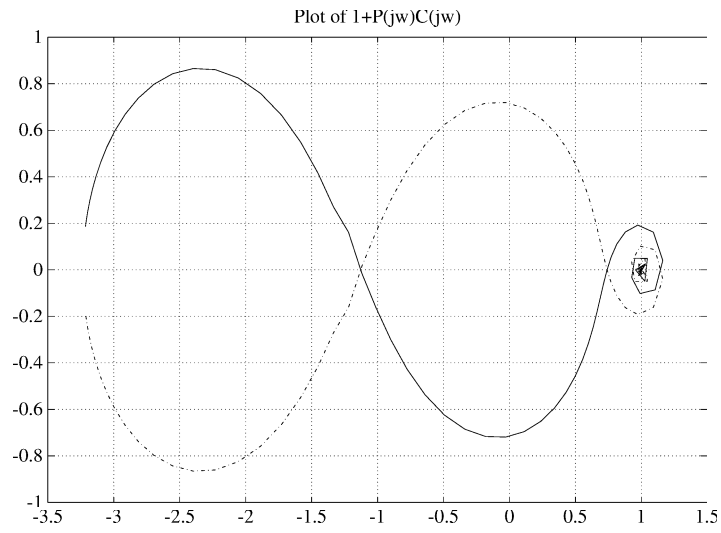


Figure 8: Nyquist plot

```

L1 =
    4.2463 + 0.0000i
    8.6808 + 0.0000i
   -2.0005 - 0.0000i
L2 =
   -4.0532 - 0.0000i
  -10.3603 - 0.0000i
   -2.0315 - 0.0000i
nF =
    1.0000    -0.0000   -0.0100
dF =
    0.2738    0.5895    0.3293    0.0273

```

These polynomials determine $L_U(s)$ and $F_\rho(s)$ which appear in the parametrization of all suboptimal controllers.

For the function $U(s)$, which is written in **U.m**, the program generates the following plots: performance plot of the closed loop system shown in Figure 9, Bode plot of the controller shown in Figure 10, and Nyquist plot of the system shown in Figure 11.

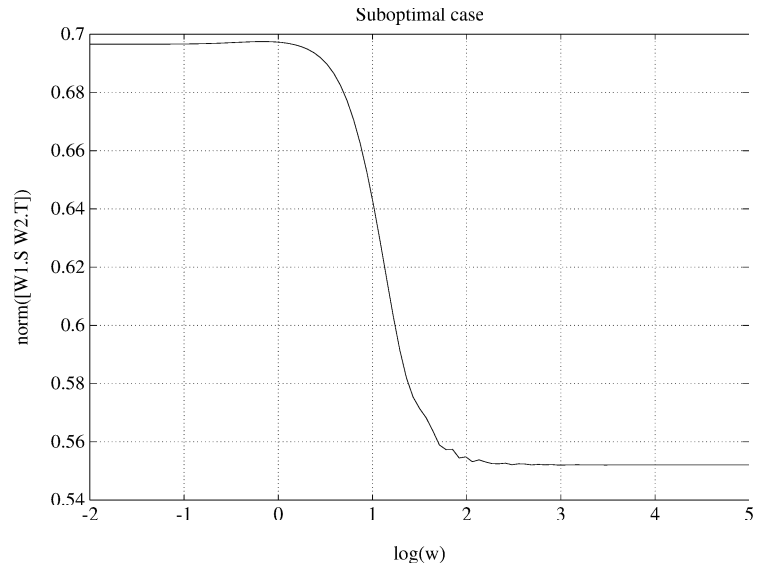


Figure 9: Performance plot

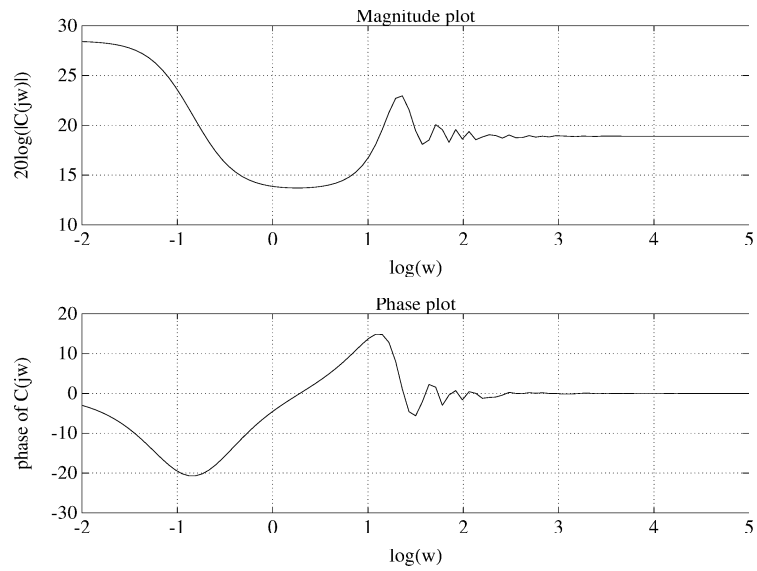


Figure 10: Bode plot of the controller

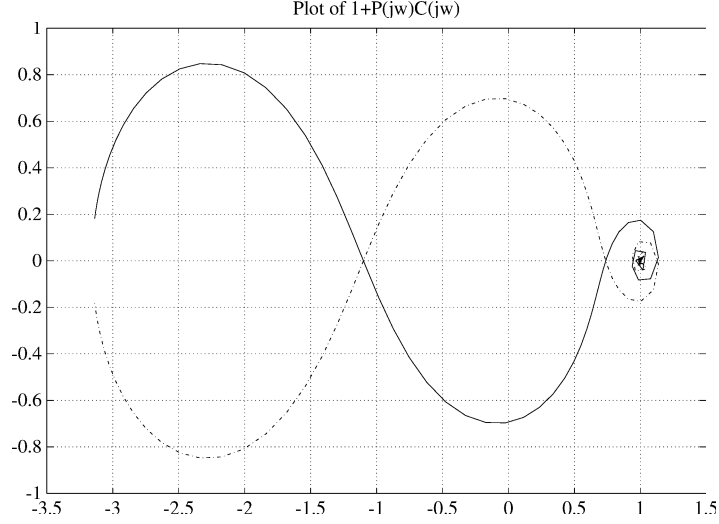


Figure 11: Nyquist plot

References

- [1] Enns, D., H. Özbay and A. Tannenbaum, “Abstract model and controller design for an unstable aircraft,” *Journal of Guidance Control and Dynamics*, **15** (1992), pp. 498–508.
- [2] Hoffman, K., *Banach Spaces of Analytic Functions*, Dover Publications, New York, reprint, originally published by Prentice Hall, 1962.
- [3] Özbay, H., “ \mathcal{H}^∞ Optimal controller design for a class of distributed parameter systems”, to appear in *Int. Journal of Control*.
- [4] Özbay, H., M. C. Smith and A. Tannenbaum, “Mixed sensitivity optimization for a class of unstable infinite dimensional systems,” *Linear Algebra and its Applications*, vol. 178 (1993), pp. 43-83.
- [5] Toker, O., and H. Özbay, “ \mathcal{H}^∞ Optimal and suboptimal controllers for infinite dimensional SISO plants,” submitted for publication.