HIMAT Robust Performance Design Example

This section contains an idealized example of μ-synthesis as applied to the design of a pitch axis controller of an experimental highly maneuverable airplane, HIMAT. The airplane model is taken from aerodynamic data for the HIMAT vehicle. The problem is posed as a robust performance problem, with multiplicative plant uncertainty at the plant input and plant output weighted sensitivity function as the performance criterion. The design procedure presented in this section involves several steps:

1. Specification of closed-loop feedback structure.
2. Specification of model uncertainty and performance objectives in terms of frequency-dependent weighting matrices.
3. Construction of open-loop interconnection for control synthesis routines.
4. Loop shape controller design for the open-loop interconnection.
5. H_∞ optimal controller design for the open-loop interconnection.
6. Analysis of robust performance properties of the resulting closed-loop systems using the structured singular value, μ (μ-analysis).
7. Use of frequency dependent similarity scalings, obtained in the μ-analysis step, to scale the open-loop interconnection, and redesign H_∞ controller (iterating on steps 5, 6, and 7 constitutes the approach to μ-synthesis called "$D - K$ iteration," which is described in detail in Chapter 5).

The main objective of this section is to illustrate μ-synthesis design methods (steps 1, 2, 3, 5, 6, 7). The loop shape controller (step 4) is included to illustrate that robust stability and nominal performance do not necessarily imply robust performance.

Many of the command outputs are not displayed in the text, since it is assumed that the reader is simultaneously working through the example on a computer.

HIMAT Vehicle Model and Control Objectives

The HIMAT vehicle model and control objectives are taken from a paper by Safonov *et al*. (1981). The interested reader should consult this paper as well as Hartman *et al*. (1979) and Merkel and Whitmoyer (1976) for more details. The HIMAT vehicle was a scaled, remotely piloted vehicle (RPV) version of an advanced fighter, which was flight tested in the late 1970s. The actual HIMAT vehicle is currently on display in the Smithsonian National Aerospace Museum in Washington, D.C. The design example will consider only the longitudinal dynamics of the airplane. These dynamics are assumed to be uncoupled from the lateral-directional dynamics. Linearized models for a collection of flight conditions can be found in [HartBG]. The state vector consists of the vehicle's basic rigid body variables.

$$x^{\mathrm{T}} = (\delta v, \alpha, q, \theta)$$

representing the forward velocity, angle-of-attack, pitch rate, and pitch angle, respectively. The flight path angle (γ) is defined as $\gamma = \theta - \alpha$. The state variables used to describe motions in the vertical plane are given below.

δv — perturbations along the velocity vector

α — angle between velocity vector and aircraft's longitudinal axis

q — rate-of-change of aircraft attitude angle

θ — aircraft attitude angle

The control inputs are the elevon (δ_e) and the canard (δ_c). The variables to be measured are α and θ.

There are three longitudinal maneuvers to be considered.

Vertical Translation: Control the vertical velocity at a constant θ (α varies). This implies that the attitude is held constant as the velocity vector rotates.

Pitch Pointing: Control the attitude at a constant flight path angle (i.e., $\theta - \alpha$ = constant). In this case the velocity vector does not rotate.

Direct Lift: Control the flight path angle at constant angle-of-attack (i.e., $\gamma = \theta$). This maneuver produces a normal acceleration response without changing the angle-of-attack.

These control objectives are accounted for within the performance specification.

Closed-Loop Feedback Structure

A diagram for the closed-loop system, which includes the feedback structure of the plant and controller, and elements associated with the uncertainty models and performance objectives, is shown inFigure 7-47.

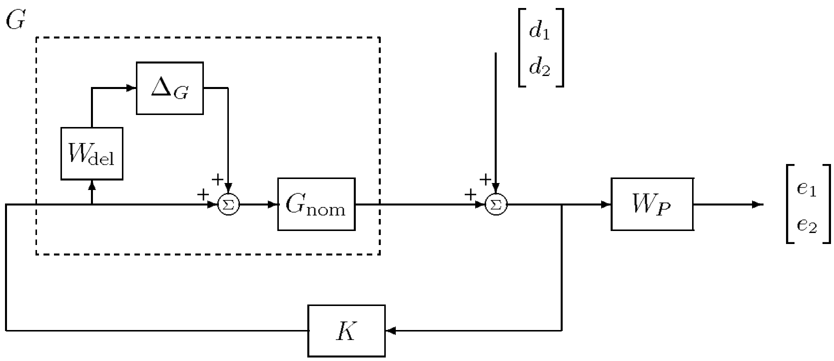


Figure 7-47: HIMAT Closed-Loop Interconnection Structure

The dashed box represents the true airplane, with associated transfer function G. Inside the box is the nominal model of the airplane dynamics, G_{nom}, and two elements, w_{del} and Δ_G, which parametrize the uncertainty in the model. This type of uncertainty is called multiplicative uncertainty at the plant input, for obvious reasons. The transfer function w_{del} is assumed known, and reflects the amount of uncertainty in the model. The transfer function Δ_G is assumed to be stable and unknown, except for the norm condition, $\|\Delta_G\|_\infty < 1$. The performance objective is that the transfer function from d to e be small, in the $\|\cdot\|_\infty$ sense, for all possible uncertainty transfer functions Δ_G. The weighting function W_P is used to reflect the relative importance of various frequency ranges for which performance is desired.

The control design objective is to design a stabilizing controller K such that for all stable perturbations $\Delta_G(s)$, with $\|\Delta_G\|_\infty < 1$, the perturbed closed-loop system remains stable, and the perturbed weighted sensitivity transfer function,

$$S(\Delta_G) := W_P(I + P(I + \Delta_G W_{\mathrm{del}})K)^{-1}$$

has $\|S(\Delta_G)\|_\infty < 1$ for all such perturbations. Recall that these mathematical objectives exactly fit in the structured singular value framework.

Uncertainty Models

The airplane model we consider has two inputs: elevon command (δ_e) and canard command (δ_c); and two measured outputs: angle-of-attack (α) and pitch angle (θ).

A first principles set of uncertainties about the aircraft model would include:

- Uncertainty in the canard and the elevon actuators. The electrical signals that command deflections in these surfaces must be converted to actual mechanical deflections by the electronics and hydraulics of the actuators. This is not done perfectly in the actual system, unlike the nominal model.
- Uncertainty in the forces and moments generated on the aircraft, due to specific deflections of the canard and elevon. As a first approximation, this arises from the uncertainties in the aerodynamic coefficients, which vary with flight conditions, as well as uncertainty in the exact geometry of the airplane. An even more detailed view is that surface deflections generate the forces and moments by changing the flow around the vehicle in very complex ways. Thus there are uncertainties in the force and moment generation that go beyond the quasi-steady uncertainties implied by uncertain aerodynamic coefficients.
- Uncertainty in the linear and angular accelerations produced by the aerodynamically generated forces and moments. This arises from the uncertainty in the various inertial parameters of the airplane, in addition to neglected dynamics such as fuel slosh and airframe flexibility.
- Other forms of uncertainty that are less well understood.

In this example, we choose not to model the uncertainty in this detailed manner, but rather to lump all of these effects together into one full-block uncertainty at the input of a four-state, nominal model of the aircraft rigid body. This nominal model has no (i.e., perfect) actuators and only quasi-steady dynamics. The nominal model for the airplane is loaded from the `mutools/subs` directory.

The simple model of the airplane has four states: forward speed (v), angle-of-attack (α), pitch rate (q) and pitch angle (θ); two inputs: elevon command (δ_e) and canard command (δ_c); and two measured outputs: angle-of-attack (α) and pitch angle (θ).

```
mkhimat;
minfo(himat)
seesys(himat,'%9.le')
```

```
-2.3e-02  -3.7e+01  -1.9e+01  -3.2e+01 |   0.0e+00   0.0e+00

 0.0e+00  -1.9e+00   9.8e-01   0.0e+00 |  -4.1e-01   0.0e+00

 1.2e-02  -1.2e+01  -2.6e+00   0.0e+00 |  -7.8e+01   2.2e+01

 0.0e+00   0.0e+00   1.0e+00   0.0e+00 |   0.0e+00   0.0e+00

 ------------------------------------- |  -----------------

 0.0e+00   5.7e+01   0.0e+00   0.0e+00 |   0.0e+00   0.0e+00

 0.0e+00   0.0e+00   0.0e+00   5.7e+01 |   0.0e+00   0.0e+00
```

The partitioned matrix represents the `[A B; C D]` state space data. Given this nominal model `himat` (i.e., $G_{\text{nom}}(s)$) we also specify a stable, 2×2 transfer matrix $W_{\text{del}}(s)$, called the uncertainty weight. These two transfer matrices parametrize an entire set of plants, $\mathcal{G}$, which must be suitably controlled by the robust controller K.

$$\mathcal{G} := \{G_{\text{nom}}(I + \Delta_G W_{\text{del}}) : \Delta_G \text{ stable}, \|\Delta_G\|_\infty \leq 1\}.$$

All of the uncertainty in modeling the airplane is captured in the normalized, unknown transfer function Δ_G. The unknown transfer function $\Delta_G(s)$ is used to parametrize the potential differences between the nominal model $G_{\text{nom}}(s)$, and the actual behavior of the real airplane, denoted by G. The dependence on frequency of the uncertainty weight indicates that the level of uncertainty in the airplane's behavior depends on frequency.

In this example, the uncertainty weight W_{del} is of the form $W_{\text{del}}(s) := w_{\text{del}}(s)I_2$, for a given scalar valued function $w_{\text{del}}(s)$. The fact that the uncertainty weight is diagonal, with equal diagonal entries, indicates that the modeled uncertainty is in some sense a round ball about the nominal model G_{nom}. The scalar weight associated with the multiplicative input uncertainty is constructed using the command nd2sys. The weight chosen for this problem is $w_{\text{del}} = \frac{50(s+100)}{s+10000}$.

```
wdel = nd2sys([1 100],[1 10000],50);
```

The set of plants that are represented by this uncertainty weight is

$$\mathcal{G} := \left\{ G_{nom}\left(I_2 + \frac{50(s+100)}{s+10000}\Delta_G(s)\right) : \Delta_G(s) \text{ stable, } \|\Delta_G\|_\infty \le 1 \right\}$$

The weighting function is used to normalize the size of the unknown perturbation Δ_G. At any frequency ω, $|\omega_{\text{del}}(j\omega)|$ can be interpreted as the percentage of uncertainty in the model at that frequency.

```
omega = logspace(0,5,50);
wdel_g = frsp(wdel,omega);
vplot('liv,lm',wdel_g)
title('Multiplicative Uncertainty Weighting Function')
xlabel('Frequency (rad/s)')
```

The particular uncertainty weight chosen for this problem indicates that at low frequency, there is potentially a 50% modeling error, and at a frequency of 173 rad/sec, the uncertainty in the model is up to 100%, and could get larger at higher and higher frequencies. A frequency response of w_{del} is shown in Figure 7-48.

Specifications of Closed-Loop Performance

The performance of the closed loop system will be evaluated using the output sensitivity transfer function, $(I + GK)^{-1}$. Good performance will be characterized in terms of a weighted H_∞ norm on this transfer function. Given a 2×2 stable, rational transfer matrix W_P, we say that nominal performance is achieved if $\|W_P(I + GK)^{-1}\|_\infty < 1$. As in the uncertainty modeling, the weighting function W_p is used to normalize specifications, in this case, to define performance as whether a particular norm is less than 1.

In this problem, we choose a simple weight of the form $W_P(s) = w_p(s)I_2$, where $w_p(s) = \frac{0.5(s+3)}{s+0.03}$.

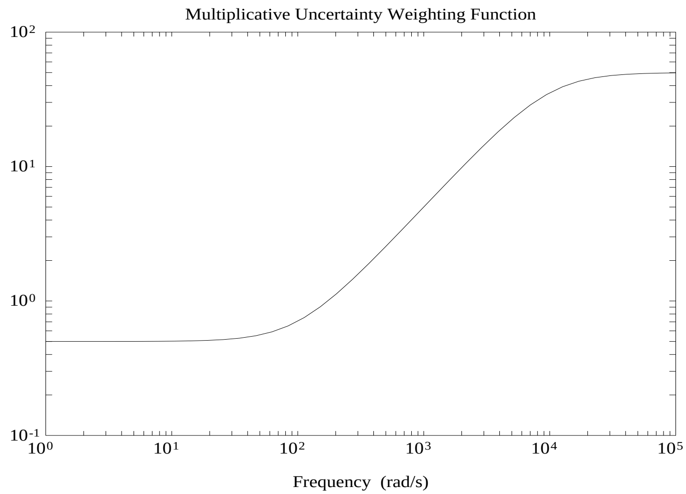


Figure 7-48: HIMAT Multiplicative Uncertainty Weighting Function

```
wp = nd2sys([1 3],[1 0.03],0.5);
```

For performance to be achieved, $\|W_P(I + GK)^{-1}\|_\infty < 1$, and since W_P is a scalar (times a 2×2 identity), the maximum singular value plot of the sensitivity transfer function $(I + GK)^{-1}$ must lie below the plot of $\frac{1}{|w_p|}$ at every frequency.

That is, $\|W_P(I + GK)^{-1}\|_\infty < 1$, if and only if at all frequencies,

$\bar{\sigma}[(I+GK)^{-1}(j\omega)] < |1/w_p(j\omega)|$.

```
omega = logspace(-3,2,50);
wp_g = frsp(wp,omega);
vplot('liv,lm',minv(wp_g))
title('Inverse of Performance Weighting function')
xlabel('Frequency (rad/s)')
```

This sensitivity weight indicates that at low frequency, the closed-loop (both nominal and perturbed) should reject disturbances at the output by a factor of 50-to–1. Expressed differently, steady-state tracking errors in both channels, due to reference step-inputs in either channel should be on the order of 0.02 or smaller. This performance requirement gets less and less stringent at higher and higher frequencies. The closed-loop system should perform better than open-loop for frequencies up to 1.73 radians/second, and for higher frequencies, the closed-loop performance should degrade gracefully, always lying underneath the inverse of the weight, wp. The frequency response of $\frac{1}{w_p}$ is shown in Figure 7-49.

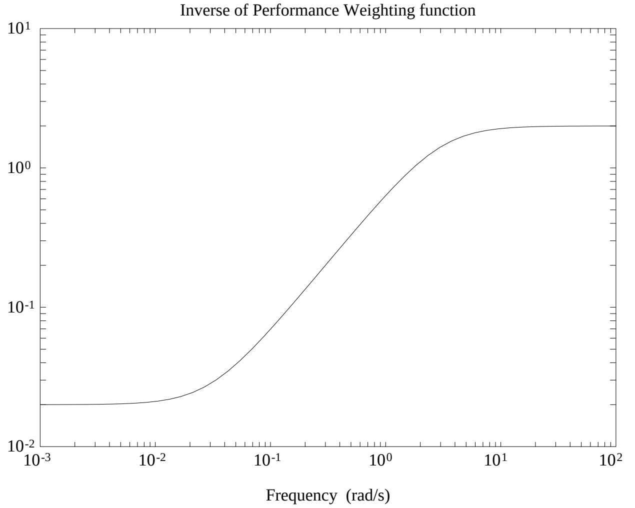


Figure 7-49: Inverse of the HIMAT Performance Weight

The 2×2 weighting matrices in the interconnection involve the scalar functions we have discussed, and identity matrices of dimension 2. We can build these matrices using the command daug, which stands for diagonal augmentation. Each new weight has two states, two inputs and two outputs as one can see using minfo.

```
wdel = daug(wdel,wdel);
wp = daug(wp,wp);
minfo(wdel)
minfo(wp)
```

The engineering motivation for a performance specification like this might come from the desire to have independent tracking of the angle of attack and pitch angle. This allows the vehicle to be pointed in pitch independently from vertical motions. We would expect this to be difficult to achieve, given that it is obviously easier for the vehicle to simultaneously pitch up and accelerate up than it is to simultaneously pitch down and accelerate up.

Robust Stability, Nominal Performance, Robust Performance

The phrases robust stability, nominal performance, and robust performance are used in this framework extensively.

Nominal Performance. The closed-loop system achieves nominal performance if the performance objective is satisfied for the nominal plant model, G_{nom}.

In this problem, that is equivalent to:

$$\text{Nominal Performance} \Leftrightarrow \|W_P(I + G_{\text{nom}}K)^{-1}\|_\infty < 1$$

Robust Stability. The closed-loop system achieves robust stability if the closed loop system is internally stable for all of the possible plant models $G \in \mathcal{G}$.

In this problem, that is equivalent to a simple norm test on a particular nominal closed-loop transfer function.

$$\text{Robust Stability} \Leftrightarrow \|W_{\text{del}}KG_{\text{nom}}(I + KG_{\text{nom}})^{-1}\|_\infty < 1$$

Robust Performance. The closed-loop system achieves robust performance if the closed-loop system is internally stable for all $G \in \mathcal{G}$, and in addition to that, the performance objective,

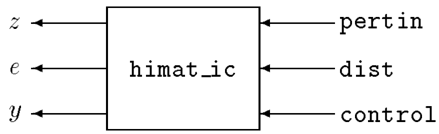
$$\|W_P(I + GK)^{-1}\|_\infty < 1,$$

is satisfied for every $G \in \mathcal{G}$. The property of robust performance is equivalent to a structured singular value test (a generalization of the two H_∞ norm tests in the previous conditions) on a particular, nominal closed- loop transfer function. This is discussed further in Chapter 4, "Modeling and Analysis of Uncertain Systems".

Building the Open-Loop Interconnection

The command `sysic` is used to construct the open-loop interconnection. We will often refer to this open-loop system as the generalized plant. In this particular example, we store the system in the MATLAB variable `himat_ic`. The command `sysic` will build any specified interconnection of smaller subsystems, provided the correct information about the interconnection is in the MATLAB workspace.

A six-input, six-output SYSTEM matrix, himat_ic, (also referred to as *P*)



has internal structure shown in Figure 7-50. The variables control, pertin, dist, and y are two element vectors.

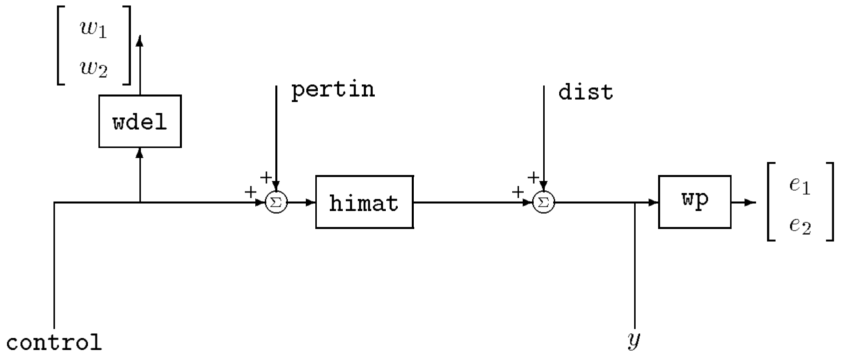


Figure 7-50: HIMAT Open-Loop Interconnection Structure

This can be produced with nine MATLAB commands, listed below. The first eight lines describe the various aspects of the interconnection, and may appear in any order. The last command, sysic, produces the final interconnection. The commands can be placed in an M-file, or executed at the command line.

```
systemnames = ' himat wp wdel ';
inputvar = '[ pertin{2} ; dist{2} ; control{2} ]';
outputvar = '[ wdel ; wp ; himat + dist ]';
input_to_himat = '[ control + pertin ]';
input_to_wdel = '[ control ]';
input_to_wp = '[ himat + dist ]';
sysoutname = 'himat_ic';
cleanupsysic = 'yes';
sysic;
```

Since the system himat_ic is still open-loop, its poles are simply the poles of the various components that make up the interconnection.

```
minfo(himat_ic)
spoles(himat_ic)
spoles(himat)
spoles(wdel)
spoles(wp)
```

μ-synthesis and D – K Iteration

For notational purposes, let $P(s)$ denote the transfer function of the six-input, six-output open-loop interconnection, himat_ic. Define a block structure Δ as

$$\Delta := \left\{ \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} : \Delta_1 \in \mathbf{C}^{2\times 2}, \Delta_2 \in \mathbf{C}^{2\times 2} \right\} \subset \mathbf{C}^{4\times 4} \text{ .}$$

The first block of this structured set corresponds to the full-block uncertainty Δ_G used in section to model the uncertainty in the airplane's behavior. The second block, Δ_2 is a fictitious uncertainty block, used to incorporate the H_∞ performance objectives on the weighted output sensitivity transfer function into the μ-framework.

Using theorem 4.5 from the "Robust Performance" section in Chapter 4, a stabilizing controller K achieves closed-loop, robust performance if and only if for each frequency $\omega \in [0, \infty]$, the structured singular value

$$\mu_\Delta[F_L(P,K)(j\omega)] < 1$$

Using the upper bound for μ, (recall that in this case, two full blocks, the upper bound is exactly equal to μ) we can attempt to minimize the peak closed-loop μ value by posing the optimization problem

$$\min_{\substack{K \\ \text{stabilizing}}} \quad \min_{\substack{d(s) \\ \text{stable,min-phase}}} \left\| \begin{bmatrix} d(s)I_2 & 0 \\ 0 & I_2 \end{bmatrix} F_L(P,K) \begin{bmatrix} d^{-1}(s)I_2 & 0 \\ 0 & I_2 \end{bmatrix} \right\|_\infty$$

Finding the value of this minimum and constructing controllers K that achieve levels of performance arbitrarily close to the optimal level is called μ-synthesis. A more detailed discussion of $D - K$ iteration is given in Chapter 5.

Before plunging into the $D - K$ iteration design procedure, we begin with a controller designed via basic MIMO loop shaping methods.

Loop Shaping Control Design

One approach to control design for the HIMAT model is to synthesize a loop shaping controller. We want the loop shape controller, K_{loop}, to make the open-loop gain act as an integrator at low frequency and at crossover. At high frequencies, we won't worry too much about the details of the roll-off, provided that it is at least first order. To achieve this, we'll roughly invert the plant $G(s)$ (G has only 1 MIMO finite zero, at $s \approx -0.026$; it also has zeros at $s = \infty$, so our inverse is only approximate) and augment the desired loop gain dynamics to the controller. The series of commands below constructs one such controller and plots the open-loop gain (broken at the input to the controller), as seen in Figure 7-51. The interested reader may want to explore various alternative schemes for constructing loop shape controllers discussed in Freudenberg and Looze (1988).

```
[a,b,c,d] = unpck(himat);
cn = c*a*a + 1000*c*a;
dn = c*a*b + 1000*c*b;
kloop = mscl(minv(pck(a,b,cn,dn)),-9000);
L = mmult(himat,kloop);
omega = logspace(-1,4,50);
Lg = frsp(L,omega);
vplot('liv,lm',vsvd(Lg))
title('Loop Gain Plot with Loop Shape Controller')
xlabel('Frequency (rad/s)')
```

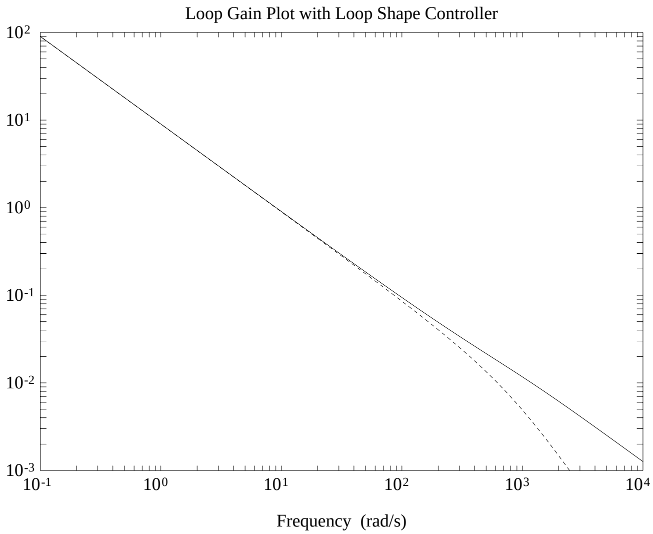


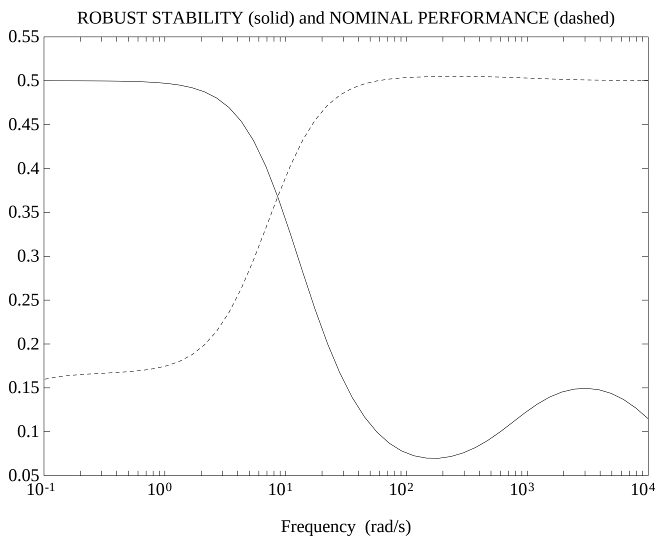
Figure 7-51: Loop Gain of the Loop Shaping Controller

The open-loop gain plot satisfies both the low frequency performance objective and the high frequency robustness goals. We have only plotted the singular values of GK_{loop}, but $K_{loop}G$ looks similar. Hence, you would expect the controller to satisfy the robust stability and nominal performance requirements.

The two 2×2 transfer functions associated with robust stability and nominal performance can be evaluated for the loop shaping controller. Simply close the open-loop interconnection P (`himat_ic`) with the loop shaping controller, K_{loop} (`kloop`) and evaluate the pertinent transfer functions using the command `sel`.

In using sel, the desired outputs (or rows) are specified first, followed by the desired inputs (or columns). The results are seen in Figure 7-52.

```
clp = starp(himat_ic,kloop,2,2);
spoles(clp)
rs_loop = sel(clp,1:2,1:2);
np_loop = sel(clp,3:4,3:4);
rs_loopg = frsp(rs_loop,omega);
np_loopg = frsp(np_loop,omega);
vplot('liv,m',vnorm(rs_loopg),vnorm(np_loopg))
tmp1 = 'ROBUST STABILITY (solid) and';
tmp2 = ' NOMINAL PERFORMANCE (dashed)';
title([tmp1 tmp2])
xlabel('Frequency(rad/s)')
```



Robust stability (solid line)   Nominal Performance (dashed line)

Figure 7-52: Robust Stability and Nominal Performance Plots for the Loop Shaping Controller

The interpretation of the plots in Figure 7-52 is as follows:

- The controlled system (with loop shaping controller) achieves nominal performance. This conclusion follows from the singular value plot of the nominal weighted output sensitivity function, which has a peak value of 0.50.
- The controlled system (with loop shaping controller) achieves robust stability. This conclusion stems from the singular value plot of the nominal weighted input complementary sensitivity function, which has a peak value of 0.50.

H_∞ Design on the Open-Loop Interconnection

In this section, we carry out the first step of the $D - K$ iteration, which is an H_∞ (sub)optimal control, design for the open-loop interconnection, `himat_ic`. In terms of the iteration, this amounts to holding the d variable fixed (at 1), and minimizing the $\|\cdot\|_\infty$ norm of $F_L(P,K)$, over the controller variable K. Recall that $F_L(P,K)$ is the nominal closed loop transfer function from the perturbation inputs and disturbances (sysic variables `pertin` and `dist`) to the perturbation outputs and errors (z and e), shown in Figure 7-53.

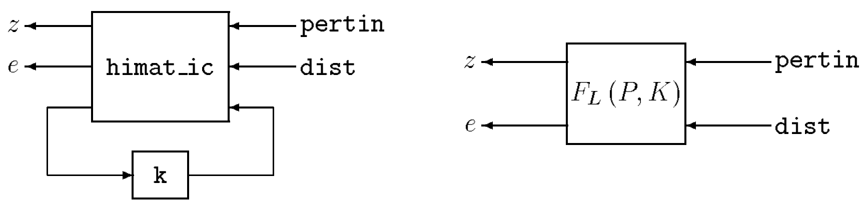


Figure 7-53: Closed-Loop Linear Fractional Transformation

The function `hinfsyn` designs a (sub)optimal H_∞ control law based on the open-loop interconnection structure provided. Syntax, input and output arguments of `hinfsyn` are

```
[k,clp] = hinfsyn(p,nmeas,ncon,glow,ghigh,tol)
```

The arguments are as follows.

Inputs

open-loop interconnection (SYSTEM matrix)	p
number of measurements	nmeas
number of controls	ncons
lower bound for bisection	glow
upper bound for bisection	ghigh
absolute tolerance for bisection method	tol

Outputs

controller (SYSTEM matrix)	k
closed-loop (SYSTEM matrix)	clp

In this example, the open-loop interconnection is himat_ic, with two measurements, two control inputs, and the bisection algorithm will search for the optimal achievable closed-loop norm, to an absolute tolerance of 0.06, between lower and upper limits of 0.8 and 6.0, respectively. Since we are planning on performing several iterations of the $D - K$ iteration procedure, we label the resulting controller k1. The resulting closed loop system (4-input, 4-output), from [pertin;dist] to [z;e] is labeled clp1.

```
[k1,clp1] = hinfsyn(himat_ic,2,2,0.8,6.0,.06);
```

The controller is stable, and its Bode plot is shown in Figure 7-54.

Properties of Controller

```
minfo(k1)
omega = logspace(-1,4,50);
spoles(k1)
k1_g = frsp(k1,omega);
vplot('bode',k1_g)
subplot(211), title('Frequency Response of k1')
```

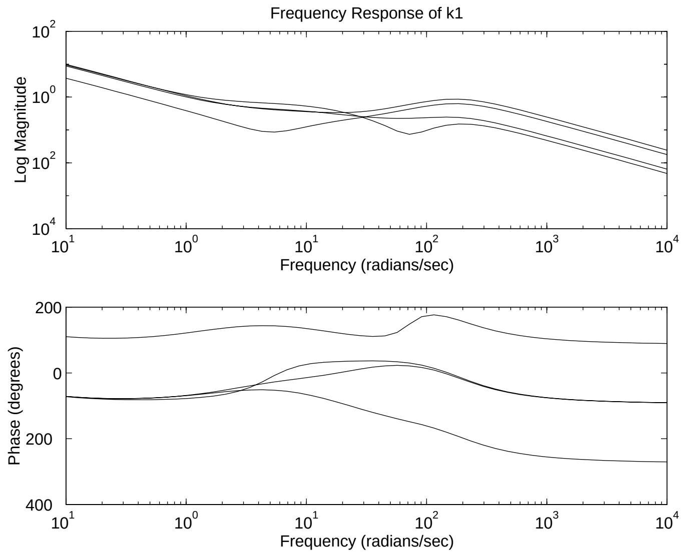


Figure 7-54: Bode Plot of k1

Figure 7-55 shows the singular values of the closed-loop system `clp1`. Although `clp1` is 4×4, at each frequency it only has rank equal to 2, hence only two singular values are nonzero.

Closed-Loop Properties

```
rifd(spoles(clp1))
clp1g = frsp(clp1,omega);
clp1gs = vsvd(clp1g);
vplot('liv,m',clp1gs)
title('Singular Value Plot of clp1')
xlabel('Frequency (rad/s)')
```

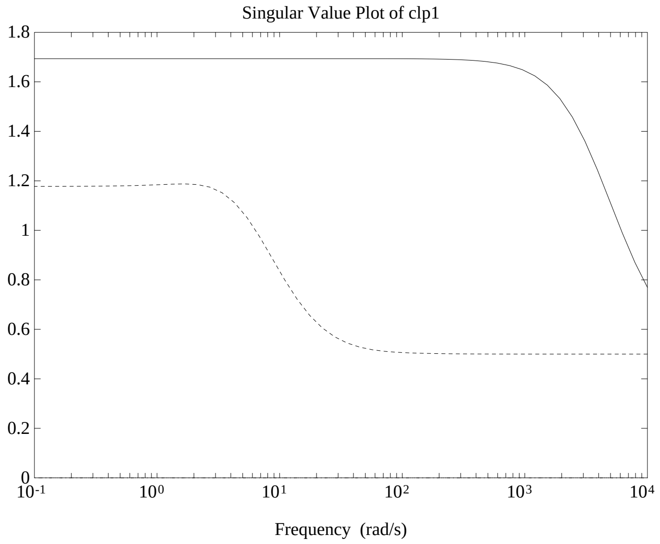


Figure 7-55: Singular Value Plot of the Closed-Loop System with k1

The two 2×2 transfer functions associated with robust stability and nominal performance may be evaluated separately, using the command `sel`. Recall that the robust stability test is performed on the upper 2×2 transfer function in `clp1`, and the nominal performance test is on the lower 2×2 transfer function in `clp1`. Since a frequency response of `clp1` is already available, (in `clp1g`) we simply perform the `sel` on the frequency response, and plot the norms.

```
rob_stab = sel(clp1g,[1 2],[1 2]);
nom_perf = sel(clp1g,[3 4],[3 4]);
minfo(rob_stab)
minfo(nom_perf)
vplot('liv,m',vnorm(rob_stab),vnorm(nom_perf))
tmp1 = 'ROBUST STABILITY (solid) and';
tmp2 = ' NOMINAL PERFORMANCE (dashed)';
title([tmp1 tmp2])
xlabel('Frequency (rad/s)')
```

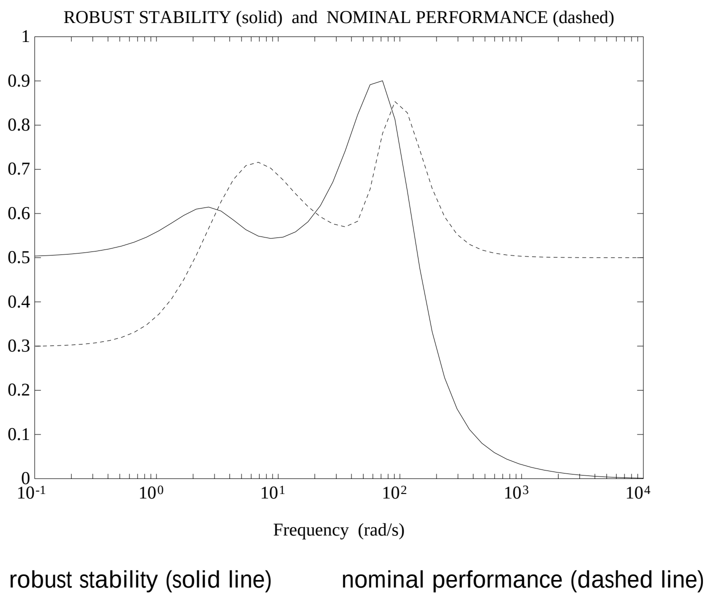


Figure 7-56: Robust Stability and Nominal Performance Plots Using Controller k1

The interpretation of the plots in Figure 7-56 is as follows:

- The controlled system achieves nominal performance. This conclusion follows from the singular value plot of the nominal weighted output sensitivity function, which has a peak value of 0.92.
- The controlled system achieves robust stability. This conclusion stems from the singular value plot of the nominal weighted input complementary sensitivity function, which has a peak value of 0.86.

Assessing Robust Performance with μ

The robust performance, HIMAT example properties of the two different closed-loop systems can be analyzed using μ-analysis. The closed-loop systems, `clp1` associated with the H_∞ controller, and clp, associated with the loop shaping controller, each have four inputs and four outputs. The first two inputs/outputs correspond to the two channels across which the perturbation Δ_G connects, while the third and fourth inputs/outputs correspond to the weighted output sensitivity transfer function. Therefore, for a frequency domain μ-analysis of robust performance properties, the block structure should consist of a 2×2 uncertainty block, and a 2×2 performance block.

$$\Delta := \left\{ \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} : \Delta_1 \in \mathbf{C}^{2 \times 2}, \Delta_2 \in \mathbf{C}^{2 \times 2} \right\}$$

Referring back to the “Robust Performance” section in Chapter 4, robust performance (with respect to the uncertainty and performance weighting functions specified above) is achieved if and only if for every frequency, $\mu_\Delta(\cdot)$ of the closed-loop frequency response is less than 1.

The syntax of a general μ calculation is:

```
[bnds,dvec,sens,pvec] = mu(matin,deltaset)
```

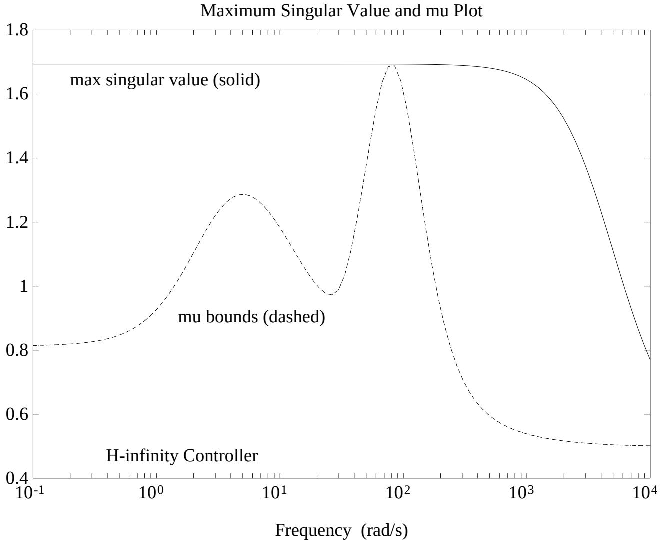
The μ-analysis program, `mu`, calculates upper and lower bounds for the structured singular value of the matrix `matin`, with respect to the block structure `deltaset`. The matrix `matin` can be a CONSTANT MATLAB matrix, or a VARYING matrix, such as a frequency response matrix of a closed-loop transfer function. In this example, the frequency response is `clp1g` and the block structure is two, 2×2 full blocks. mu returns the upper and lower bounds in 1×2 VARYING matrix `bnds1`, the frequency-varying D-scaling matrices in `dvec1`, the frequency dependent perturbation associated with the lower bound in `rp1`, and the sensitivity of the upper bound to the D-scales in sens1.

The bounds can be calculated by specifying the block structure, and running `mu`.

μ Analysis of H_∞ Design

The H_∞ design is analyzed with respect to structured uncertainty using μ. First, the density of points in the frequency response is increased from 50 to 100 to yield smoother plots. Then the upper and lower bounds for μ are calculated on the 4×4 closed-loop response of the matrix `clp_g1`. The upperand lower bounds for μ are plotted (in this example they lie on top of one another) along with the maximum singular value in Figure 7-57.

```
deltaset=[2 2; 2 2];
omega1 = logspace(-1,4,100);
clp_g1 = frsp(clp1,omega1);
[bnds1,dvec1,sens1,pvec1] = mu(clp_g1,deltaset);
vplot('liv,m',vnorm(clp_g1),bnds1)
title('Maximum Singular Value and mu Plot')
xlabel('Frequency (rad/s)')
text(.15,.84,'max singular value (solid)','sc')
text(.3,.4,'mu bounds (dashed)','sc')
text(.2,.15,'H-infinity Controller','sc')
```



$F_L(P,K1)(j\omega)$ (solid line)   robust performance μ (dashed line)

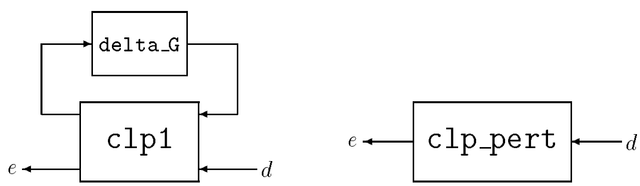
Figure 7-57: Maximum Singular Value of the 4 × 4 Closed-Loop Transfer Function $F_L(P,K_1)(j\omega)$ and Robust Performance μ

Hence, the controlled system (from H_∞) does not achieve robust performance. This conclusion follows from the μ plot in Figure 7-57, which peaks to a value of 1.69, at a frequency of 73.6 rad/sec. This means that there is a perturbation matrix Δ_G, with $\|\Delta_G\|_\infty = \frac{1}{1.69}$, for which the perturbed weighted sensitivity gets large

$$\|W_P(I + G_{\mathrm{nom}}(I + W_{\mathrm{del}}\Delta_G)K^{-1}\|_\infty = 1.69$$

This perturbation, Δ_G, can be constructed using dypert. The input variables to the command dypert consist of two outputs from μ, the perturbation matrix and the bounds, along with the block structure, and the numbers of the blocks for which the rational matrix construction should be carried out. Often times, some of the blocks correspond to performance blocks and therefore need not be constructed. Here, only the first block is an actual perturbation, so the construction is only done for this 2 × 2 perturbation (fourth argument of dypert).

```
delta_G = dypert(pvec1,deltaset,bnds1,1);
minfo(delta_G) % 2 by 2
rifd(spoles(delta_G)) % stable
hinfnorm(delta_G) % 1/1.69
clp_pert = starp(delta_G,clp1,2,2); % close top loop with delta
minfo(clp_pert)
rifd(spoles(clp_pert)) % stable, since RS passed
hinfnorm(clp_pert) % degradation of performance
```



μ-Analysis of Loop Shape Design

Robust performance for the system with the loop shape controller, K_{loop}, can also be analyzed using μ. You might think that the loop shaping controller would exhibit good robust performance properties, based on its excellent nominal performance and robust stability properties.

```
clpg = frsp(clp,omega1);
bnds_loop = mu(clpg,deltaset);
vplot('liv,m',bnds_loop)
title('mu Plot of Closed-loop System')
xlabel('Frequency (rad/s)')
text(.6,.85,'Loop Shape Controller','sc')
```

However, the closed-loop system with the loop shaping controller does not achieve robust performance. In fact, μ reaches a peak value of 11.7 at a frequency of 0.202 rad/sec, as seen in Figure 7-58. This means that there is a perturbation matrix Δ_G, with $\|\Delta_G\|_\infty = \frac{1}{11.7}$, for which the perturbed weighted sensitivity gets large

$$\|W_P(I + G_{\text{nom}}(I + W_{\text{del}}\Delta_G)K^{-1}\|_\infty = 11.7$$

Notice that this perturbation is 8.2 times smaller than the perturbation associated with the H_∞ control design, but that the subsequent degradation in closed-loop performance is 8.2 times worse. Therefore, the loop shaping controller will most likely perform poorly on the real system.

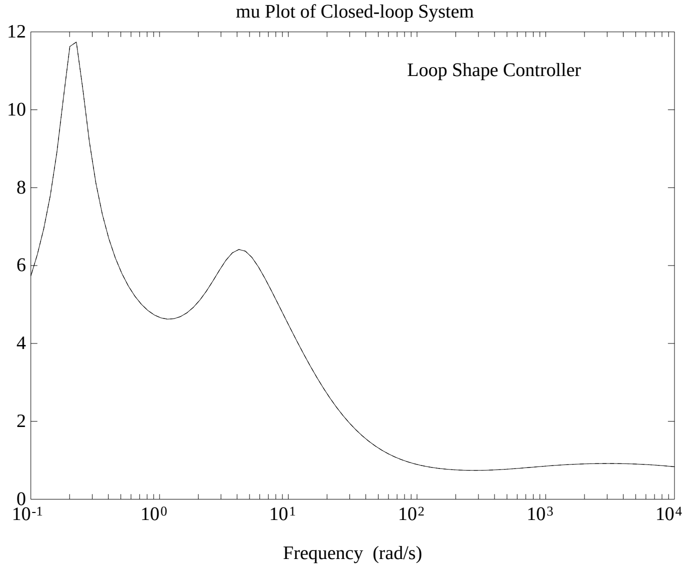


Figure 7-58: Robust Performance μ Plot of the Closed-Loop HIMAT System with the Loop Shaping Controller

The structured singular value μ is large in the low frequency range due to the off-diagonal elements of `clpg` being large. One can see this using the command `blknorm`, which outputs the individual norms of the respective blocks. The coupling between the off-diagonal terms associated with 0.202 rads/sec point to the problem — the upper right entry is 0.14, somewhat small, but not small enough to counteract the large (nearly 1000) lower left entry. As expected, $\mu \approx \sqrt{0.14 * 959} = 11.6$.

```
blkn_cl = blknorm(clpg,deltaset);
see(xtract(blkn_cl,.15,.3))

2 rows 2 columns

iv = 0.159986
   4.9995e-01   1.4127e-01
   5.6525e+02   1.6402e-01

iv = 0.202359
   4.9991e-01   1.4193e-01
   9.5950e+02   1.6520e-01

iv = 0.255955
   4.9985e-01   1.4294e-01
   7.5635e+02   1.6607e-01
```

Recapping Results

Let's summarize what has been done so far:

- The generalized plant, `himat_ic`, which includes the aircraft model, uncertainty and performance weighting functions, and the interconnection of all of these components was built using `sysic`.
- A controller was designed using `hinfsyn`.
- The robust performance characteristics of the closed-loop system were analyzed with a structured singular value frequency domain test using `mu`.

The structured singular value analysis involved computing μ at each frequency of this 4×4 closed loop response, with respect to a block structure Δ which is made up of two 2×2 full blocks. The blocks represent, respectively, uncertainty in the aircraft model, and the performance objectives.

At this stage, the controller which has been designed using H_∞ techniques (nearly) minimizes the H_∞ norm of the closed loop transfer function from the 4×1 vector of perturbation inputs and disturbance inputs to the 4×1 vector of perturbation outputs and error signals. The structured singular value analysis shows that the μ analysis improves on the $\bar{\sigma}(\cdot)$ bound at most frequencies, but there is no improvement at the frequency of 73.6 rads/sec.

Hence, the peak value of the μ-plot is as high as the peak value on the singular value plot, the μ analysis seems to have been of little use. However, at most of the frequencies, μ is smaller than $\bar{\sigma}$, and in the next iteration of synthesis, the controller can essentially focus its efforts at the problem frequency, and lower the peak of the μ-plot.

D – K Iteration for HIMAT Using dkit

The μ-Tools M-file `dkit` automates the μ-synthesis procedure via $D - K$ iteration. This example is a modified version of the HIMAT problem considered earlier (see Figure 7-47) and is extended to include a frequency dependent sensor noise signal, as shown in the closed-loop interconnection diagram in Figure 7-59. This sensor noise signal is included to represent a more realistic performance objective.

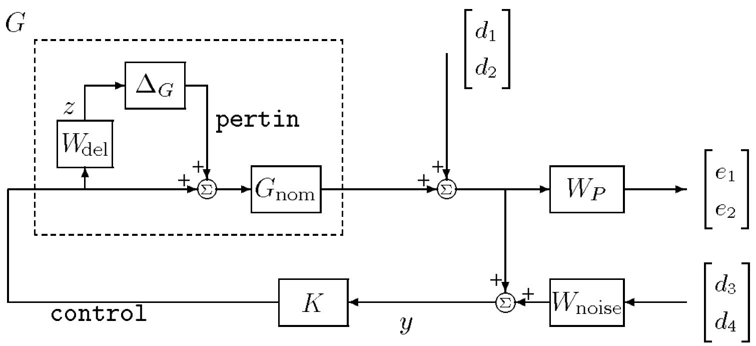
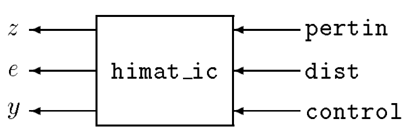


Figure 7-59: HIMAT Closed-Loop Interconnection Structure

Now, the open-loop interconnection structure is the eight input, six output linear system, shown below



with internal structure, as in Figure 7-60.

The M-file `mkhicn` creates the plant model, weighting functions and the interconnection structure shown in Figure 7-60. This can be produced with nine MATLAB commands, listed below, and also in the M-file `mkhicn` (which creates the plant and weighting functions).

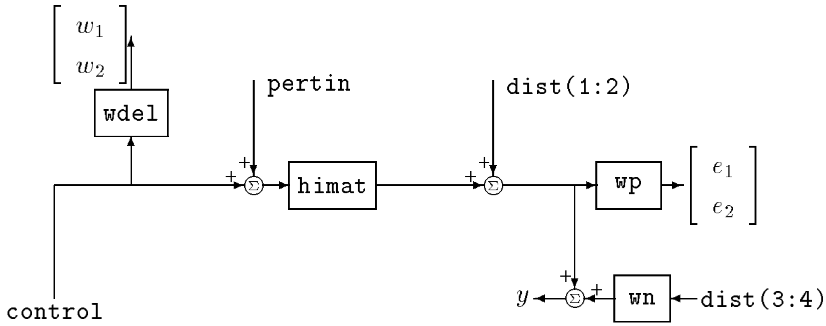


Figure 7-60: HIMAT Open-Loop Interconnection Structure

```
mkhicn
```

file: mkhicn.m

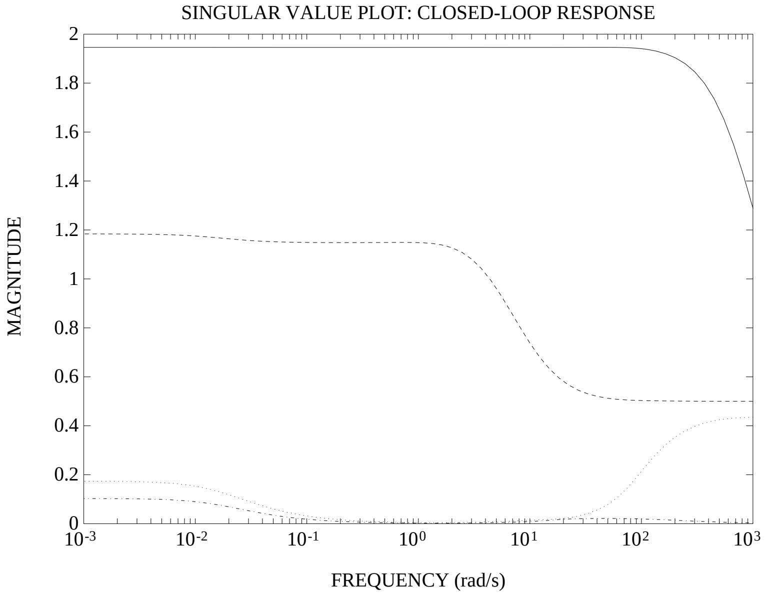
```
mkhimat;
wdel = nd2sys([50 5000],[1 10000]);
wp = nd2sys([0.5 0.9],[1 0.018]);
poleloc = 320;
Wn = nd2sys([2 0.008*poleloc],[1 poleloc]);
wdel = daug(wdel,wdel);
wp = daug(wp,wp);
Wn = daug(wn,wn);
   systemnames = ' himat wp wdel wn ';
   inputvar = '[ pertin{2} ; dist{4} ; control{2} ]';
   outputvar = '[ wdel ; wp ; himat + dist(1:2) + wn ]';
   input_to_himat = '[ control + pertin ]';
   input_to_wdel = '[ control ]';
   input_to_wp = '[ himat + dist(1:2) ]';
   input_to_wn = '[ dist(3:4) ]';
   sysoutname = 'himat_ic';
   cleanupsysic = 'yes';
sysic;
```

The dkit file himat_dk has been set up with the necessary variables to design robust controllers for HIMAT using $D - K$ iteration. A listing of the himat_dk file follows. You can copy this file into your directory from the μ-Tools subroutines directory, mutools/subs, and modify it for other problems, as appropriate.

```
% himat_dk
%
% This script file contains the USER DEFINED VARIABLES for the
%mutools DKIT script file. The user MUST define the 5
%variables below.
%------------------------------------------%
   REQUIRED USER DEFINED VARIABLES
%------------------------------------------%
% Nominal plant interconnection structure
NOMINAL_DK = himat_ic;

% Number of measurements
NMEAS_DK = 2;

% Number of control inputs
NCONT_DK = 2;

% Block structure for mu calculation
BLK_DK = [2 2;4 2];

% Frequency response range
OMEGA_DK = logspace(-3,3,60);

%----------------------end of himat_dk------------------------%
```

After the himat_dk.m file has been set up, you need to let the dkit program know which setup file to use. This is done by setting the string variable DK_DEF_NAME in the MATLAB workspace equal to the setup filename. Typing dkit at the MATLAB prompt will then begin the $D - K$ iteration procedure.

```
DK_DEF_NAME = 'himat_dk';
dkit
starting mu iteration #: 1

Iteration Number: 1
-------------------

Information about the Interconnection Structure IC_DK:
system10 states 6 outputs8 inputs
Test bounds: 0.0000 < gamma <= 100.0000

  gamma   hamx_eig   xinf_eig   hamy_eig   yinf_eig    nrho_xy    p/f

100.000   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.0003     p

 50.000   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.0011     p

 25.000   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.0046     p

 12.500   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.0183     p

  6.250   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.0742     p

  3.125   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.3117     p

  1.562   2.3e-02    0.0e+00    1.7e-02    0.0e+00     1.5583#    f

  2.152   2.3e-02    0.0e+00    1.8e-02    0.0e+00     0.7100     p

Gamma value achieved: 2.1516
```

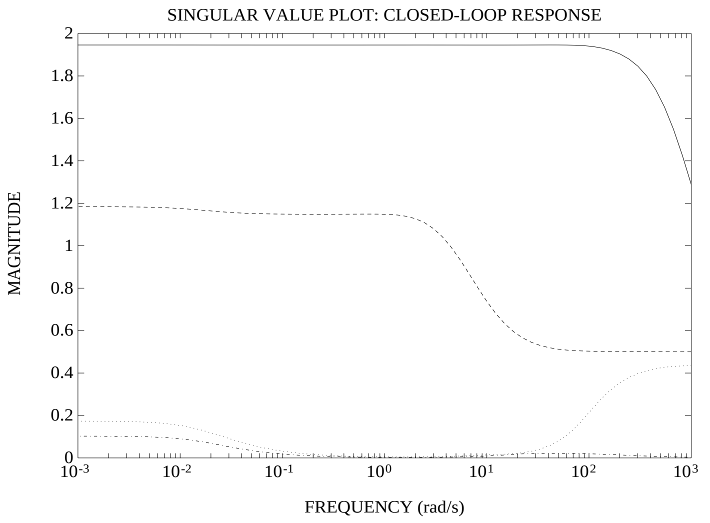


Singular value plot of closed-loop system in graphics window. Make sure that the chosen frequency range is appropriate.

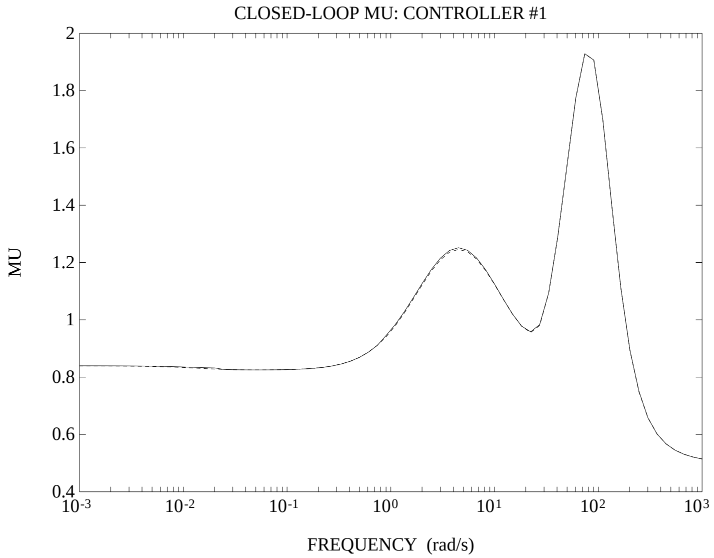
Next, we get to change the frequency range, if desired. For illustrative purposes, we will change the number of logarithmically spaced points from 60 to 70.

```
Do you want to modify OMEGA_DK? (y/n): y

Current Frequency Variable
---------------------------------------------------------------
(s)Frequency Spacinglog
(n) # Frequency Points60
(b)Frequency - bottom1.00e-03
(h)Frequency - high1.00e+03

Enter (s n b and/or h) to change OMEGA, (e) to exit unchanged: n

------------CHANGING # of Points--------------

Enter desired # of points: 70

Current Frequency Variable
---------------------------------------------------------------
(s)Frequency Spacinglog
(n) # Frequency Points70
(b)Frequency - bottom1.00e-03
(h)Frequency - high1000

Enter (s n b and/or h) to change, (e) to exit: e
```

By typing `e`, we exit the frequency range modification, and the closed-loop singular value frequency response is recalculated and plotted. In this case, the plot looks exactly the same.



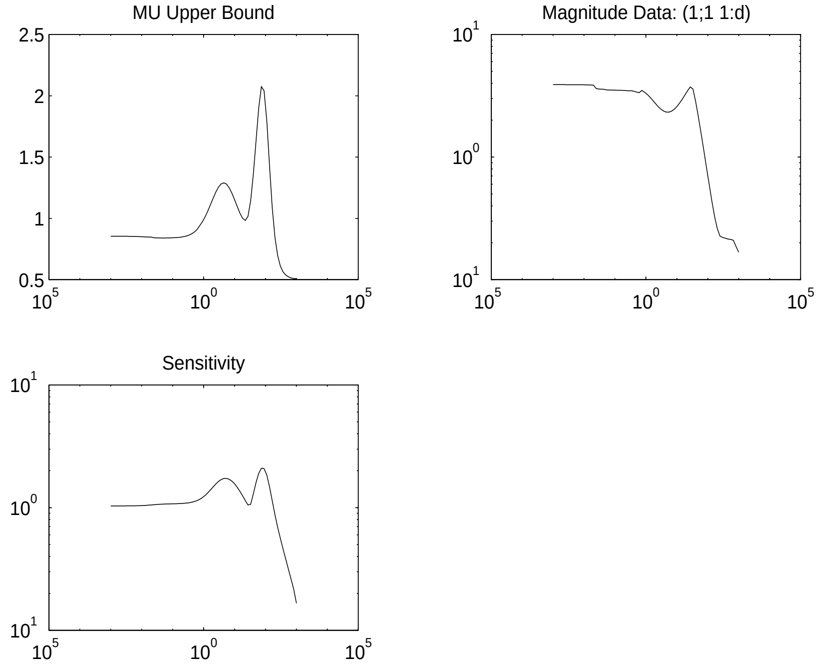

```
Singular value plot of closed-loop system in graphics window.
Make sure that chosen Frequency range is appropriate.

   Do you want to modify OMEGA_DK? (y/n): n

   RERUN H_inf with DIFFERENT bounds/tolerances? (y/n): n

Calculating MU of closed-loop system: g_dk
points completed....
1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.
18.19.20.21.22.23.24.25.26.27.28.29.30.31.32.33.34.35.
36.37.38.39.40.41.42.43.44.45.46.47.48.49.50.51.52.53 .
54.55.56.57.58.59.60.61.62.63.64.65.66.67.68.69.70.
```




```
   MU plot for control design:Press any key to continue

Iteration Summary
------------------------------------
Iteration #1
Controller Order10
Total D-Scale Order0
Gamma Achieved2.152
Peak mu-Value2.075

   Another D-K itera;tion? (y/n): y
```

Proceeding with the $D-K$ iteration, we must fit the D-scaling variable that was calculated in the μ upper-bound computation. This rational D-scaling will then be absorbed into the open-loop interconnection.

A plot of the μ upper bound, the first frequency-dependent *D*-scaling data (this is the curve we want to fit), and the sensitivity of the μ upper bound. The sensitivity measure roughly shows (across frequency) the relative importance of the accuracy of the curve fit. It is used in the curve fit optimization to weight some frequency ranges differently than others.



You are prompted to enter your choice of options for fitting the *D*-scaling data. Press return to see your options.

```
Enter Choice (return for list):
Choices:

nd       Move to Next D-Scaling
nb       Move to Next D-Block
i        Increment Fit Order
d        Decrement Fit Order
apf      Auto-PreFit
mx 3     Change Max-Order to 3
at 1.01  Change Auto-PreFit Tol to 1.01
0        Fit with zeroth order
2        Fit with second order
n        Fit with n'th order
e        Exit with Current Fittings
s        See Status
```

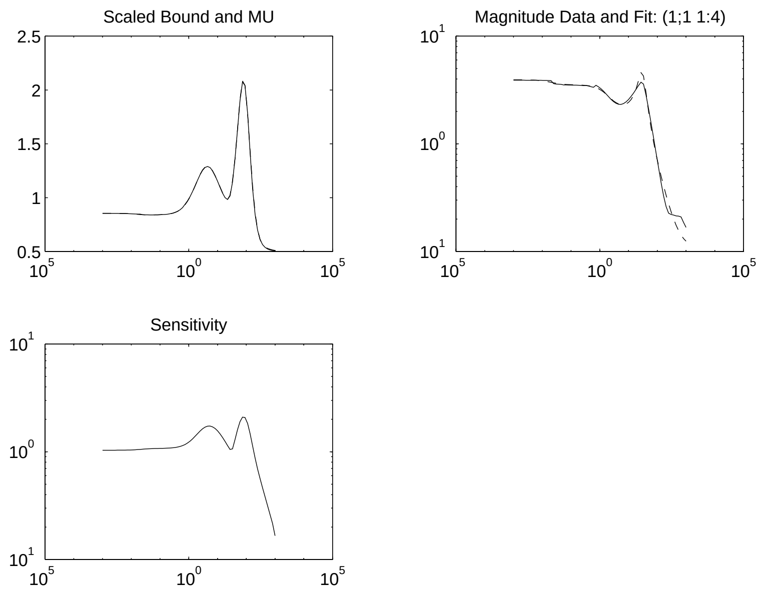
- `nd` and `nb` allow you to move from one *D*-scale data to another. `nd` moves to the next scaling, whereas `nb` moves to the next scaling block. For scalar *D*-scalings, these are identical operations, but for problems with full *D*-scalings, (perturbations of the form δI) they are different. In the (1,2) subplot window, the title displays the *D*-scaling Block number, the row/column of the scaling that is currently being fit, and the order of the current fit (with `d` for data, when no fit exists).
- The order of the current fit can be incremented or decremented (by 1) using `i` and `d`.
- `apf` automatically fits each *D*-scaling data. The default maximum state order of individual *D*-scaling is 5. The `mx` variable allows you to change the maximum *D*-scaling state order used in the automatic pre-fitting routine. `mx` must be a positive, nonzero integer. `at` allows you to define how close the rational, scaled μ upper bound is to approximate the actual μ upper bound in a norm sense. Setting `at 1` would require an exact fit of the *D*-scale data, and is not allowed. Allowable values are greater than 1, and the meaning is

explained in Chapter 5, “Control Design via m Synthesis”. This setting plays a role (mildly unpredictable, unfortunately) in determining where in the (*D*,*K*) space the *D* – *K* iteration converges.

- Entering a positive integer at the prompt will fit the current *D*-scale data with that state order rational transfer function.
- `e` exits the *D*-scale fitting to continue the *D* – *K* iteration.
- The variable `s` will display a status of the current and fits.

Select `apf` to automatically fit the *D*-scale data. After a few seconds of calculation, the first *D*-scale is fit with a fourth order rational curve as shown in the top-right plot, along with the frequency-dependent magnitude data that is being fit. Also shown in the top-left portion of the graphics window is a plot comparing the upper bound of μ (using the frequency dependent *D*-scalings) along with the maximum singular value of the closed-loop transfer function after being scaled by the just-computed rational fit. Note that the second *D*-scale data, which corresponds to the performance block, is fit with a constant. This is expected since one of the *D*-scalings can always be normalized to be 1. Enter `s` after the *D*-scale fitting is completed to see the status.

```
Enter Choice (return for list): apf
Starting Auto-PreFit...
Block 1 , Order = 0 1 2 3 4
Block 2 , Order = 0

Done

Enter Choice (return for list): s

Block 1: 4
Block 2: 0
Auto PreFit Fit Tolerance: 1.03
Auto PreFit Maximum Order: 5

Enter Choice (return for list):
```

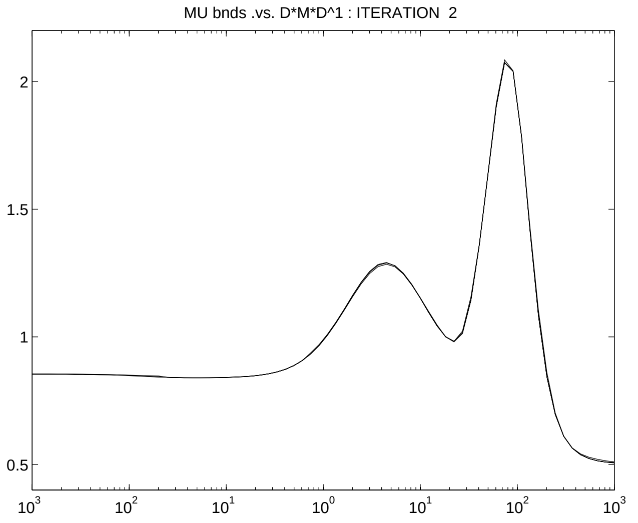


In this case, the μ upper bound with the *D*-scale data is very close to the μ upper bound with the rational *D*-scale fit. The fourth order fit is quite adequate in scaling the closed-loop transfer function. The curve fitting procedure for this scaling variable is concluded by entering `e` at the `Enter Choice` prompt.

```
Enter Choice (return for list): e
```

In this problem, the block structure consists of two complex full blocks: the 2×2 block associated with the multiplicative uncertainty model for the aircraft, and the 4×2 performance block. Since there are two blocks, there is only one *D*-scaling variable, and we are completely done with the curve fitting in this iteration.

At the conclusion of the curve fitting procedure, a frequency response plot is shown, which compares the norm of the rationally scaled, closed-loop system to the lower and upper bound for μ.

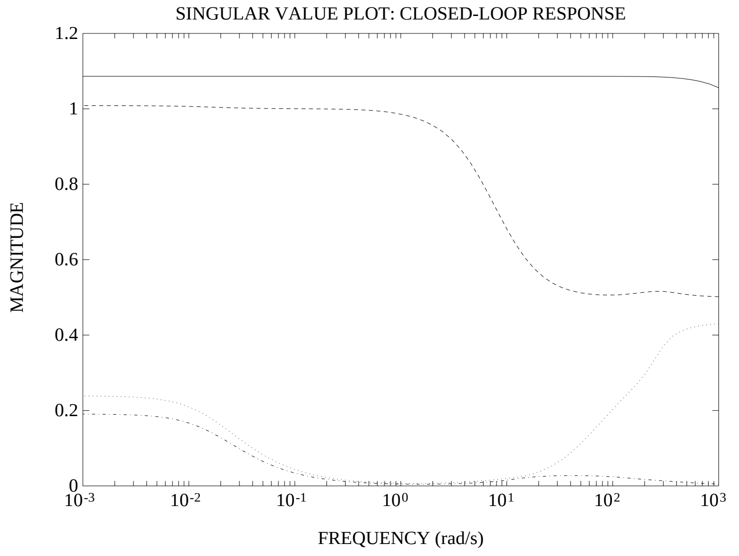


Finally, before the next H_∞ synthesis procedure, we get the option of changing the parameters used in the `hinfsyn` routine. This is useful to change the lower bound in the γ-iteration. In this example, we make no changes, and simply continue.

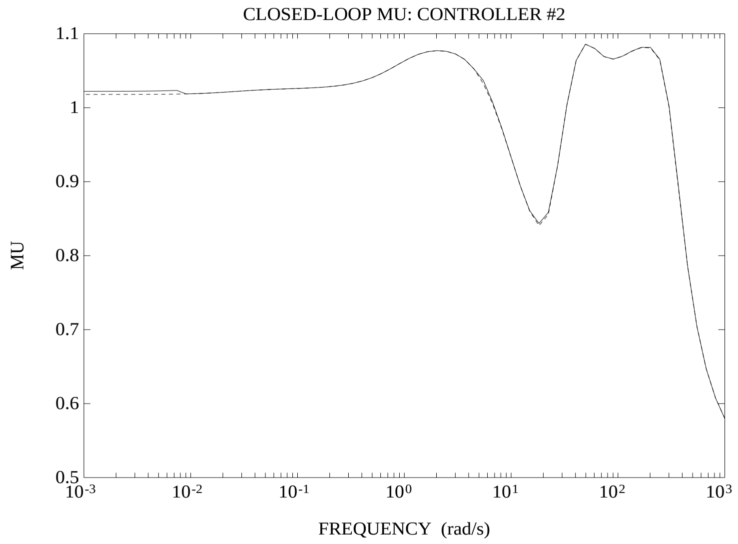
```
Altering the HINFSYN settings for next synthesis...

HINFSYN Settings Previously  Next
------------------------------------------------------------
(u)GAMMA Upper Bound100 2.146
(l)GAMMA Lower Bound  0.00e+000.00e+00
(t)Bisection Tolerance1.0004.29e-02
(p)Riccati PSD epsilon1.00e-061.00e-06
(j)Riccati j-w epsilon1.00e-081.00e-08

Enter (u l t p j) to change, (e) to exit: e
```

The iteration proceeds by computing the H_∞ optimal controller for the scaled (using the rational scalings from the curve fitting) open-loop system.

```
Iteration Number:  2
--------------------

Information about the Interconnection Structure IC_DK:
system:26 states6 outputs8 inputs
Test bounds: 0.0000 <  gamma  <= 2.1461


gamma   hamx_eig   xinf_eig   hamy_eig   yinf_eig  nrho_xy   p/f
2.146    2.0e–02  –5.2e–14    1.8e–02  –1.5e–16   0.1557 p
1.073    1.9e–02  –4.6e–14    1.7e–02  –3.4e–17   0.9958    p
0.537    1.4e–14#   *******    1.2e–02  –1.7e–18   ******    f
0.805    1.8e–02  –7.7e–13    1.6e–02  –5.8e–18  4.9336#    f
1.019    1.9e–02  –6.0e–14    1.7e–02  –2.4e–17  1.2077#    f
1.055    1.9e–02  –2.3e–13    1.7e–02  –3.8e–17  1.0589#    f

Gamma value achieved:1.0730
```

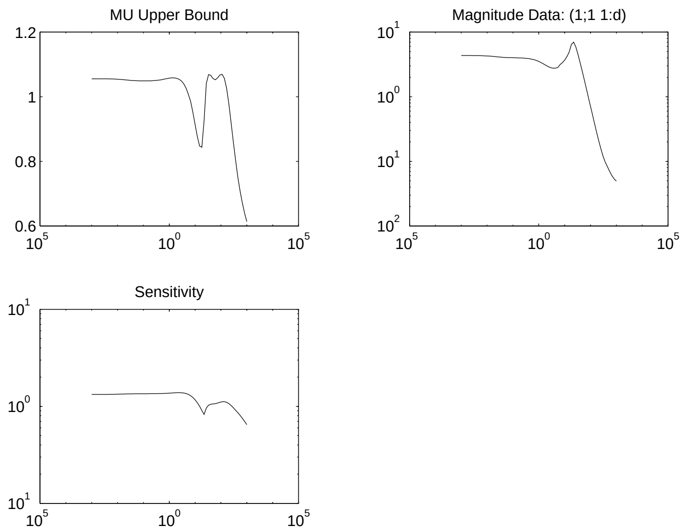



```
Singular value plot of closed-loop system in graphics window.
Make sure that chosen Frequency range is appropriate.

Do you want to modify OMEGA_DK? (y/n): n

RERUN H_inf with DIFFERENT bounds/tolerances? (y/n): n

Calculating MU of closed-loop system: g_dk
points completed....
1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.
18.19.20.21.22.23.24.25.26.27.28.29.30.31.32.33.34.35.
36.37.38.39.40.41.42.43.44.45.46.47.48.49.50.51.52.53.
54.55.56.57.58.59.60.61.62.63.64.65.66.67.68.69.70.
```



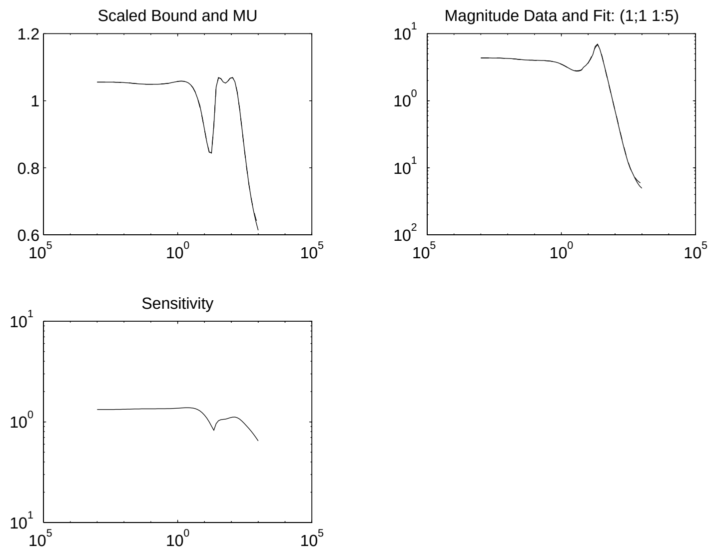

```
MU plot for control design:Press any key to continue

Iteration Summary
--------------------------------------------
Iteration #12
Controller Order1026
Total D-Scale Order016
Gamma Achieved2.1521.073
Peak mu-Value2.0751.073

Another D-K iteration? (y/n): y
```

The third iteration begins by fitting the new frequency-dependent *D*-scaling.



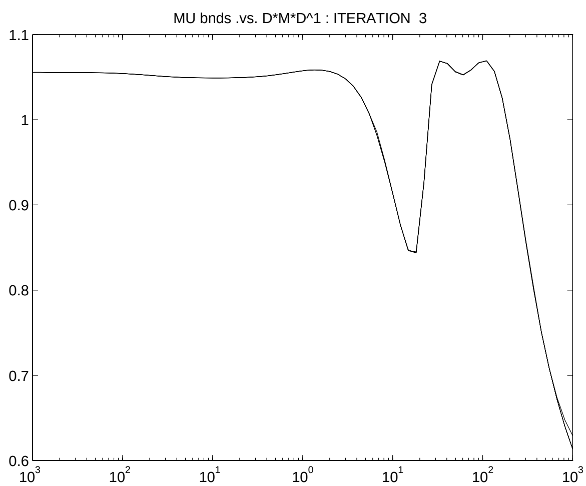
Again, enter the automatic pre-fitting option `apf`.

```
Enter Choice (return for list): apf
Starting Auto-PreFit...
Block 1 , Order = 0 1 2 3 4 5
Block 2 , Order = 0
Done
```

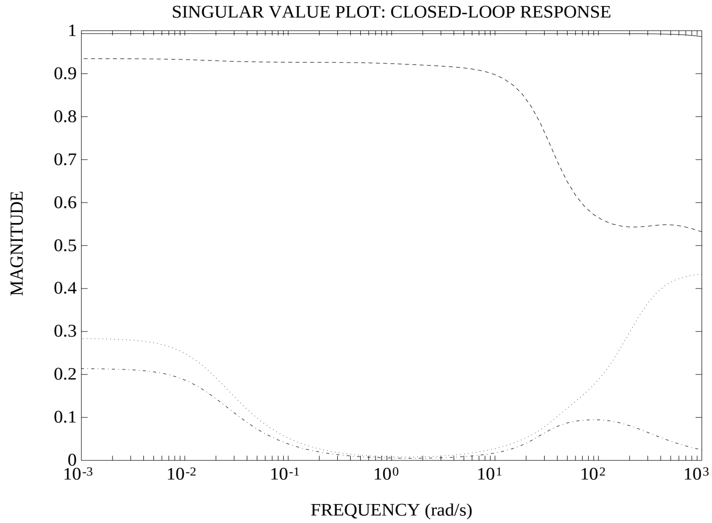


This fifth order fit works well in scaling the transfer function, so we exit the curve fitting routine.

```
Enter Choice (return for list):e
```

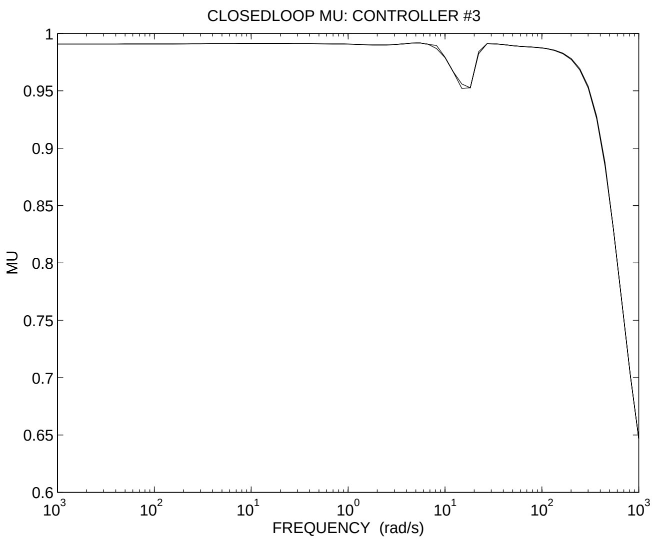


```
Altering the HINFSYN settings for next synthesis...

HINFSYN Settings       Previously          Next
----------------------------------------------------------------
(u)GAMMA Upper Bound   2.146               1.095
(l)GAMMA Lower Bound   0.00e+00            0.00e+00
(t)Bisection Tolerance 4.29e-02            2.19e-02
(p)Riccati PSD epsilon 1.00e-06            1.00e-06
(j)Riccati j-w epsilon 1.00e-08            1.00e-08

Enter (u l t p j) to change,  (e) to exit: e


Iteration Number:  3
--------------------

Information about the Interconnection Structure IC_DK:
system:30 states6 outputs8 inputs
Test bounds: 0.0000 <  gamma  <= 1.0947


gamma hamx_eig xinf_eig hamy_eig yinf_eig  nrho_xy   p/f

1.095  2.1e-02 -1.2e-13  1.7e-02 -3.1e-18   0.5970    p

0.547  9.1e-13#  *******  1.2e-02 -4.2e-17  ******     f

0.821  2.0e-02 -1.2e-11  1.6e-02 -5.5e-16  45.1126#    f

1.040  2.1e-02 -7.0e-13  1.7e-02 -2.4e-16   0.7263    p

0.996  2.1e-02 -8.4e-13  1.6e-02 -2.9e-17   0.8741    p

0.961  2.1e-02 -1.8e-13  1.6e-02 -2.5e-17   1.0433#    f

0.970  2.1e-02 -7.9e-13  1.6e-02 -2.2e-16   0.9922    p

Gamma value achieved:0.9704
```



Singular Value plot of closed-loop system in GRAPHICS window. Make sure that chosen Frequency range is appropriate

```
Do you want to modify OMEGA_DK? (y/n): n

RERUN H_inf with DIFFERENT bounds/tolerances? (y/n): n

Calculating MU of closed-loop system: g_dk
points completed....
1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.
18.19.20.21.22.23.24.25.26.27.28.29.30.31.32.33.34.35.
36.37.38.39.40.41.42.43.44.45.46.47.48.49.50.51.52.53.
54.55.56.57.58.59.60.61.62.63.64.65.66.67.68.69.70.
```




```
MU plot for control design:Press any key to continue

Iteration Summary
-----------------------------------------------------------
Iteration #123
Controller Order102630
Total D-Scale Order01620
Gamma Achieved2.1521.0730.970
Peak mu-Value2.0751.0730.973

Another D-K itera;tion? (y/n): n
echo off
Next MU iteration number: 4
```

At this point, we have achieved the robust performance objective, and we end the $D - K$ iteration. We have designed a 30 state controller using $D - K$ iteration which achieves a μ value less than 1.

In this example, it is also possible to reduce the controller order to 12, using truncated balanced realizations, and still maintain closed-loop stability and robust performance.

```
max(real(spoles(kd_k3)))

ans =
   -1.6401e-02

[k_dk3bal,hsv] = sysbal(k_dk3);
[k_dk3red] = strunc(k_dk3bal,12);
clpred_12 = starp(himat_ic,k_dk3red);
max(real(spoles(clpred_12)))

ans =
   -6.9102e-03

clpred_12g = frsp(clpred_12,OMEGA_DK);
[bnds] = mu(clpred_12g,[2 2;4 2],'c');
pkvnorm(sel(bnds,1,1))

ans =
   9.9910e-01
```

H_∞ Loop Shaping Design for HIMAT

Now consider H_∞ loop shaping control design for the HIMAT example discussed in previous sections. Recall that the objective is to reject disturbances up to about 1 rad/sec in the presence of substantial plant uncertainty above 100 rad/sec. A loop-shaping design that gives a bandwidth of approximately 10 rad/sec and robustness which should be satisfactory.

Design Precompensator

First form the HIMAT system and plot its maximum singular values across frequency (see Figure 7-61).

```
mkhimat
[type,p,m,n] = minfo(himat);
om = logspace(-2,4,100);
himatg = frsp(himat,om);
vplot('liv,lm',vsvd(himatg),1);
title('SINGULAR VALUES OF HIMAT')
ylabel('SINGULAR VALUES'); xlabel('FREQUENCY (RAD/SEC)');
```

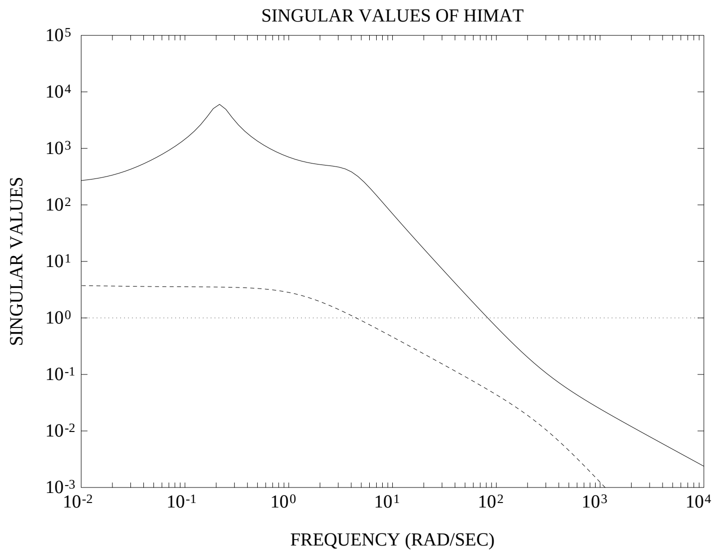


Figure 7-61: Singular Values of HIMAT

The singular values of `himat` are plotted in Figure 7-61, and although the unity gain cross over frequency is approximately correct, the low frequency gain is too low. We therefore introduce a proportional plus integral (P+I) precompensator with transfer function $(1 + s^{-1})I_{2\times 2}$ to boost the low frequency gain and give zero steady state errors. The singular values of `himat` and `himat` augmented with the P+I compensator are shown in Figure 7-62.

```
sysW1 = daug(nd2sys([1 1],[1 0]),nd2sys([1 1],[1 0]));
sysGW = mmult(himat,sysW1);
sysGWg = frsp(sysGW,om);
vplot('liv,lm',vsvd(himatg),'-.',vsvd(sysGWg),'-',1,'--')
title('SINGULAR VALUES OF HIMAT AND AUGMENTED HIMAT')
ylabel('SINGULAR VALUES');
xlabel('FREQUENCY (RAD/SEC)');
```

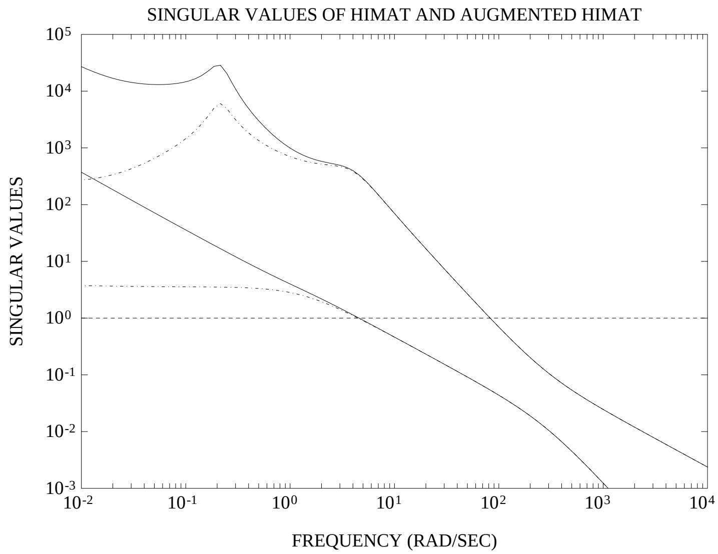


Figure 7-62: Singular Values of HIMAT (dashed-dotted) and Augmented Plant with the H_∞ Loop Shaping Controller (solid)

H_∞ Loop Shaping Feedback Compensator Design

The optimally robust controller can now be designed for the frequency shaped plant.

```
[sysK1,emax] = ncfsyn(sysGW,1);
disp(['emax = ' num2str(emax)]);
emax = 0.436
```

The value of `emax = 0.436` is a very satisfactory stability margin. The closed-loop norm can be checked by forming the open-loop interconnection of Figure 7-63, denoted by `p_ic`, and checking the reciprocal of the H_∞ gain. See the "Loop Shaping Using H• Synthesis" section in Chapter 3 for more details about this control design technique.

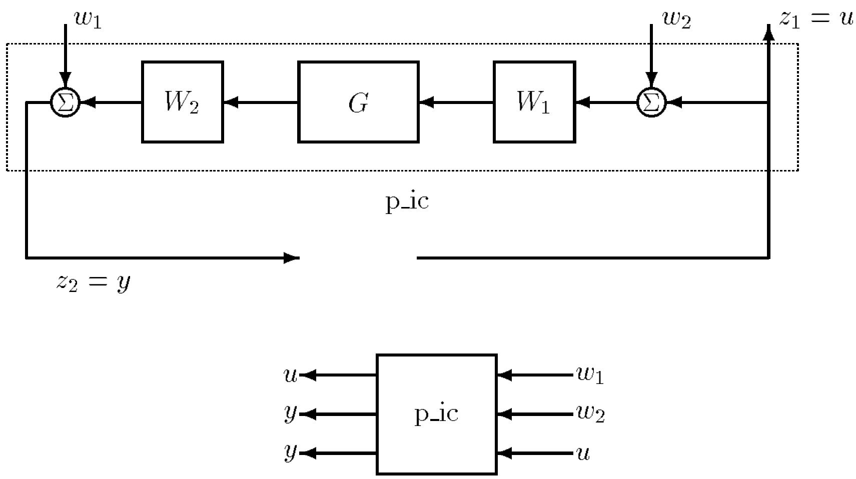


Figure 7-63: H_∞ Loop Shaping Standard Block Diagram

```
systemnames = 'sysGw';
inputvar = '[ w12; w22; u2 ]';
outputvar = '[ u; w1+sysGw; w1+sysGw ]';
input_to_sysGw = '[ w2+u ]';
sysoutname = 'p_ic';
cleanupsysic = 'yes';
sysic;
ncf_cl = starp(p_ic,sysK1);
ncf_cl_nm = hinfnorm(ncf_cl);
1/ncf_cl_nm(1)
ans =
   4.3598e-01
```

The implemented controller involves the pre- and postweighting functions W_1 and W_2, as shown in Figure 7-64.

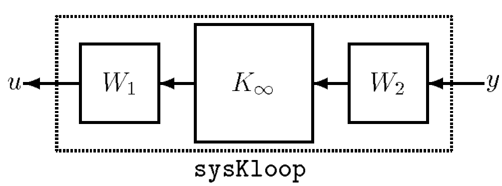


Figure 7-64: Actual Implemented H_∞ Loop Shaping Controller

In this example $W_2 = I_{2\times2}$, therefore, the implemented loop shaping controller is:

```
sysKloop = mmult(sysW1,sysK1);
```

Assessing Robust Performance with μ

We can now assess this controller by testing the original specification by using a μ-test as in previous designs. First the interconnection structure needs to be formed.

```
wdel = nd2sys([50,5000],[1,10000]);
wp = nd2sys([0.5,1.5],[1,0.03]);
wdel = daug(wdel,wdel);
wp = daug(wp,wp);
himatic
clear wp wdel
```

Now form the closed-loop and evaluate the robust performance μ with the H_∞ loop shaping compensator implemented (see Figure 7-65).

```
clp1 = starp(himat_ic,sysKloop,2,2);
clp_g1 = frsp(clp1,om);
deltaset = [2 2; 2 2];
[bnds1,dvec1,sens1,pvec1] = mu(clp_g1,deltaset);
vplot('liv,m',bnds1);
title('ROBUST PERFORMANCE MU WITH LOOPSHAPE CONTROLLER')
ylabel('MU');
xlabel('FREQUENCY (RAD/SEC)');
disp(['mu value is ' num2str(pkvnorm(sel(bnds1,1,1)))])
mu value is 1.323
```

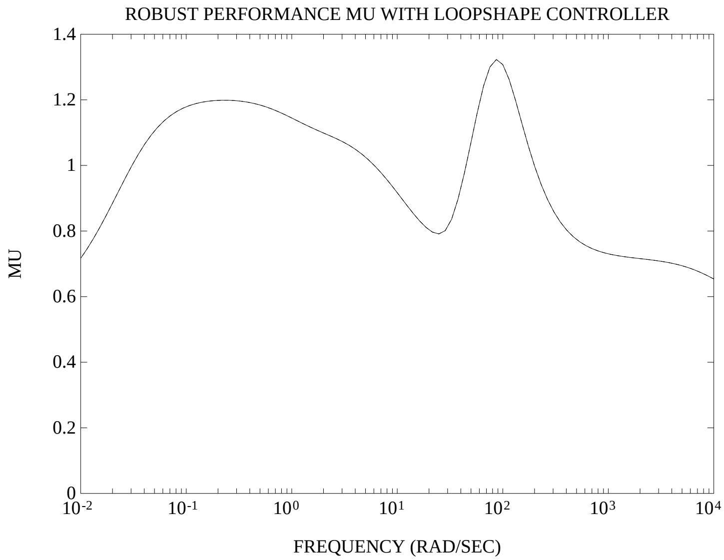


Figure 7-65: Robust Performance μ with sysK1

The plot of μ is shown in Figure 7-65 (solid line), the μ-value is close to that required, giving a satisfactory design without exploiting the details of the performance and uncertainty weights. This substantiates the claim that this design method can give a very robust initial design which does not require detailed trade-offs between weights to be studied.

Reduced Order Designs

The previously designed controller will typically have one less state than the precompensator plus the plant. It is therefore often desirable to reduce the number of states in the controller. There are systematic techniques for doing this based on model reduction in the ν gap metric, δ_ν, which is roughly equivalent to model reduction of normalized coprime factors of the plant and controller.

First reduce the weighted plant model order and measure the resulting gap.

```
[sysGW_cf,sigGW]=sncfbal(sysGW);
sigGW
sigGW =

    8.9996e-01
    7.1355e-01
    3.3542e-01
    7.9274e-02
    8.5314e-04
    2.1532e-04

sysGW_4 = cf2sys(hankmr(sysGW_cf,sigGW,4,'d'));
gapGW_4 = nugap(sysGW,sysGW_4)
gapGW_4 =

    8.6871e-04
```

It is seen that a fourth order model is essentially indistinguishable from the full order model due to the small value of the ν gap. Now design the controller for this reduced order system.

```
[sysK1_3,emax_3] = ncfsyn(sysGW_4,1);
emax_3
emax_3 =

    4.3597e-01
```

This three state controller can be reduced to two states using Hankel model reduction techniques (`hankmr`).

```
[sysK1_3_cf,sigK1_3] = sncfbal(sysK1_3);
sigK1_3
sigK1_3 =

    3.1674e-01
    2.7851e-01
    6.9959e-02

sysK1_2 = cf2sys(hankmr(sysK1_3_cf,sigK1_3,2,'d'));
gapK_2=nugap(sysK1_3,sysK1_2)
gapK_2 =

   6.9959e-02
```

The robustness bound in the “Loop Shaping Using H• Synthesis” section in Chapter 3, equation 3-23, can now be applied to give a lower bound on robustness,

```
e_bound=sin(asin(emax_3)-asin(gapGW_4)-asin(gapK_2))
e_bound =

    3.7114e-01
```

and this can be compared with the actual stability margin with the reduced order controller as follows.

```
cl_red = starp(p_ic,sysK1_2);
tmp = hinfnorm(cl_red);
e_act=1/tmp(1)
e_act =

    4.0786e-01
```

It is seen that the actual robustness is about half way between the optimal and this lower bound. The important use of the bounds is that they indicate what level of reduction is guaranteed not to degrade robustness significantly.

This gives a third-order controller together with the second-order P+I term. The μ-value for this controller, not shown here, turns out to have essentially the same μ -value as the closed-loop system with the full order controller.

Introducing a Reference Signal

A reference signal can be introduced into the loop shaping control design as follows.

```
[sysK3,emax] = ncfsyn(sysGW,1.1,'ref');
cl_ref = starp(p_ic,sysK3,2,2);
minfo(cl_ref)
system: 12 states  4 outputs  6 inputs
```

When the `ncfsyn` option ref is specified, the controller includes an extra set of reference inputs. The second input argument to `ncfsyn` is 1.1. This implies we are designing a suboptimal controller with 10% less performance than at the optimal. In practice, a 10% suboptimal design often performs better in terms of robust performance than the optimal controller on the actual system.

The last two inputs to cl_ref correspond to the reference signals, the first two outputs are the outputs of the controller and the last two outputs are the inputs to the controller (plant output plus observation noise). This design makes the closed-loop transfer function from reference to plant output the numerator of a normalized coprime factorization of sysGW. An external reference compensator could also be added to improve the command response and there are many possibilities. Here we first diagonalize the closed-loop reference to output transfer function and then insert some phase advance to increase the speed of response.

```
cl_ref_yr=sel(cl_ref,3:4,5:6);
P0 = transp(mmult([0 1; -1 0],cl_ref_yr,[0 1; -1 0]));
P1 = nd2sys([10 50],[1 50]);
P2 = daug(P1,P1);
sysQ = mmult(P0,P2);
```

Now reduce the order of sysQ to four states using the balanced realization technique (sysbal), and incorporate into the controller.

```
[sysQ_b,sig_Q] = sysbal(sysQ);
sig_Q
sig_Q =
   3.9665e+00
   2.9126e+00
   7.2360e-01
   4.5915e-01
   2.3600e-02
   1.0016e-02
   1.2526e-06
   5.2197e-07
sysQ4 = strunc(sysQ_b,4);
sysK_ref = mmult(sysK3,daug(eye(2),sysQ4));
```

Finally form the closed-loop and calculate the step response.

```
sys_cl_ref = starp(p_ic,sysK_ref,2,2);
y = trsp(sys_cl_ref,[0;0;0;0;1;0],0.5,.001);
vplot(sel(y,1,1),'-.',sel(y,2,1),'.',sel(y,3,1),'-',...
   sel(y,4,1),'--')
title('CLOSED LOOP TIME RESPONSE WITH SYSK1')
xlabel('TIME (SECONDS)')
```

The step responses are plotted in Figure 7-66. The first output (solid) tracks the command well with a rise time of less than 0.1 second and no overshoot. The output of the second channel (dashed) is zero, indicating that there is *no* cross coupling between the output channels in the nominal closed-loop system. The controller output commands (dotted and dashed-dotted lines) are also plotted. This is just the nominal step response and further tests are needed to check the sensitivity of the closed-loop to the plant uncertainty.

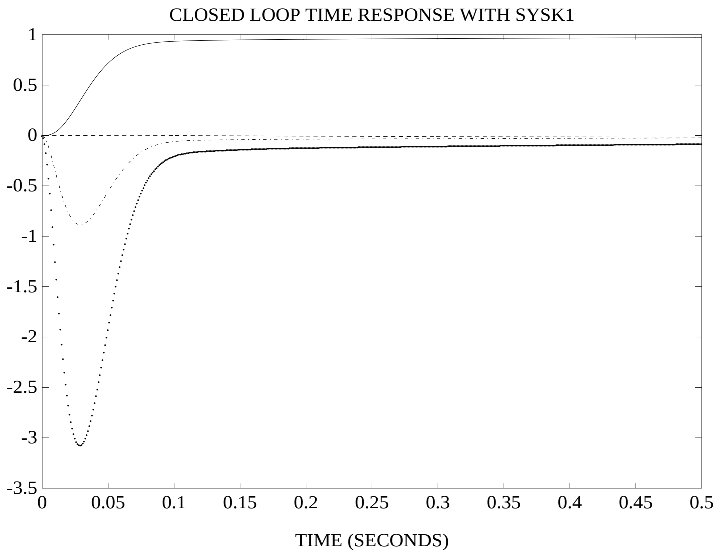


Figure 7-66: Time Response of Closed Loop System: sysK1

HIMAT References

Freudenberg, J., and D. Looze, “Frequency domain properties of scalar and multivariable feedback systems,” *Lecture Notes in Control and Information Sciences*, Springer-Verlag, 1988.

Hartman, G.L., M.F. Barrett and C.S. Greene, “Control designs for an unstable vehicle,” NASA Dryden Flight Research Center, *Contract Report* NAS 4–2578, December, 1979.

Merkel, P.A., and R.A. Whitmoyer, “Development and evaluation of precision control modes for fighter aircraft,” *Proceedings of the AIAA Guidance and Control Conference*, San Diego, CA, Paper 76–1950, 1976.

Safonov, M.G., A.J. Laub, and G.L. Hartman, “Feedback properties of multivariable systems: The role and use of the return difference matrix,” *IEEE Transactions on Automatic Control*, Vol. AC–26, No. 1, pp. 47–65, February, 1981.