Robust Control Toolbox

For Use with MATLAB®

- Modeling
- Simulation
- Implementation

User's Guide

Version 2

How to Contact The MathWorks:

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup

`support@mathworks.com`	Technical support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 Phone

508-647-7001 Fax

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 Mail

For contact information about worldwide offices, see the MathWorks Web site.

Robust Control Toolbox User's Guide

Printing History:			
	August 1992	First printing	New for Version 1
	January 1998	Online only	Revised for Version 2
	June 2001	Online only	Revised for Version 2.08 (Release 12.1)
	June 2004	Online only	Revised for Release 14

Contents

1 Tutorial

2 Function Reference

1

Tutorial

The MATLAB (R) collection of matrix manipulation routines has proved to be extremely useful to control engineers and system researchers in developing the software tools to do control system design in many different fields.

The Robust Control Toolbox is written in M-files using the matrix functions of the Control System Toolbox and MATLAB. It enables you to do "robust" multivariable feedback control system modeling, analysis and design based on the singular-value Bode plot. Many of the functions described in this guide incorporate theory originally developed at USC by the authors. The early version of the Robust Control Toolbox called LINF was distributed widely [2].

The Robust Control Toolbox includes tools that facilitate the following:

- **Robust Analysis**

  Singular Values [12, 29].

  Characteristic Gain Loci [25].

  Structured Singular Values [31, 32, 13].
- **Robust Synthesis**

  μ synthesis [33, 15].

  LQG/LTR, Frequency-Weighted LQG [12, 29].

  $\mathbf{H}^2$, $\mathbf{H}^\infty$ [16, 34, 18, 36, 37, 28, 24, 19].
- **Robust Model Reduction**

  Optimal Descriptor Hankel (with Additive Error Bound) [37].

  Schur Balanced Truncation (with Additive Error Bound) [39].

  Schur Balanced Stochastic Truncation (with Multiplicative Error Bound) [40].
- **Sampled-Data Robust Control** [35, 38]

Useful features of the Robust Control Toolbox include the structured singular value (`perron`, `psv`, `osborne`, `ssv`), μ synthesis tools (`fitd`, `augd`) and an optional system data structure (`mksys`, `branch`, `tree`) that simplifies user interaction and saves typing by enabling a system and related matrices to be represented by a single MATLAB variable. The function `hinf` has been improved in a number of ways, including more informative displays and more reliable algorithms. The function `hinfopt` automatically computes optimal $\mathbf{H}^\infty$ control laws via the so-called "gamma-iteration."

A demonstration M-file called `rctdemo` runs through the above features of the Robust Control Toolbox with a modern fighter aircraft and a large space

structure design example. To start the demo, execute `rctdemo` from inside MATLAB.

Optional System Data Structure

This section introduces a useful feature of the Robust Control Toolbox—a hierarchical data structure that can simplify the user interaction with the toolbox. If this is your first time reading, you may skip this section and come back to it later.

Among the features of the Robust Control Toolbox is a set of M-files which permit data describing a system or collection of systems to be incorporated in, and extracted from, a single MATLAB variable called a “tree”, which can be created by the MATLAB function `tree`. The tree data structure simplifies MATLAB operations tremendously by allowing you to represent systems of matrices (and even systems of systems or systems of matrices) by a single MATLAB variable. In particular, a single variable can be used to represent the matrices describing a plant, a controller or both, thereby vastly simplifying user interaction with MATLAB.

The following M-files have been developed to endow MATLAB with the hierarchical tree data structure. They are

```
mksys          branch          tree               graft
```

These functions enable many matrices, along with their names and relationships to each other to be represented by a single tree variable. For example, a state-space system representation (`A,B,C,D`) is a special kind of tree. The following elaborate the use of this data structure.

`mksys`: This function can pack matrices describing a system into a single MATLAB variable. For example,

```
ssg = mksys(ag,bg,cg,dg);
TSS = mksys(A,B1,B2,C1,C2,D11,D12,D21,D22,'tss');
```

allows the four state-space system matrices (`ag,bg,cg,dg`) to be represented by `ssg`, and the two-port state-space system (`A,B1,B2,...`) to be packed into `TSS`. A variety of system types can be similarly handled via an identification variable at the end of the input arguments of the function mksys. For example, the command `desg = mksys(ag,bg,cg,dg,eg,'des')`; packs a descriptor system into `desg`, etc.

Type	V1, V2, V3,..., Vn	Description
'ss'	(a,b,c,d,ty)	Standard State-Space (default)
'des'	(a,b,c,d,e,ty)	Descriptor System
'tss'	(a,b1,b2,c1,c2,d11,d12,d21,d22,e,ty)	Two-Port State-Space
'tdes'	(a,b1,b2,c1,c2,d11,d12,d21,d22,e,ty)	Two-Port Descriptor
'gssv	(sm,dimx,dimu,dimy,ty)	General State-Space
'gdes'	(e,sim,dimx,dimu,dimy,ty)	General Descriptor
'gpsm'	(psm,deg,dimx,dimu,dimy,ty)	General Polynomial System Matrix
'tf'	(num,den,ty)	Transfer Function
'tfm'	(num,den,m,n,ty)	Transfer Function Matrix
'imp'	(y,ts,nu,ny)	Impulse Response

`branch`: This function recovers the matrices packed in a system or tree variable selectively. For example,

```
[D11,C2] = branch(TSS,'d11,c2');
```

recovers the matrices `D11` and `C2` from the system `TSS` and

```
ag = branch(ssg,'a');
```

recovers the matrix `ag` from the state-space system `ssg`.

To recover all the matrices from `ssg` at once, you may type

```
[ag,bg,cg,dg] = branch(ssg);
```

`tree`: This function provides a general tool for creating hierarchical data structures containing matrices, strings and even other trees. It is used as a subroutine by `mksys`. For example, if you wish to keep track of the two-port plant (`A,B1,B2,...`), along with the controller (`af,bf,cf,df`), the frequency response [`w;sv`] along with the name `Aircraft Design Data`, you simply do the following

```
fr = tree('w,sv',w,sv);
DesignData = ...
tree('plant,controller,freq,name',TSS,ssf,fr,...
'Aircraft Design Data');
```

Figure 1-1, Branch Structure of the tree Variable shows the branch structure of the tree variable `DesignData`. This tree variable has two levels, since the branches named `plant`, `controller`, and `freq` are themselves trees. However, there is in general no limit to the number of levels a tree can have.

To recover the variable name from the first level of the tree `DesignData`, we type

```
name = branch(DesignData,'name')
ans =
Aircraft Design Data
```

The list of names of “root branches” of the tree is always stored in the tree as branch 0. For example, to find the names of the root branches in the tree variable `DesignData`, type the following

```
branch(DesignData,0)
ans =
plant,controller,freq,name
```

To recover the value of the matrix `c1` in the branch `plant` of the second level of the tree `DesignData`, we type

```
C1 = branch(DesignData,'plant/c1');
```

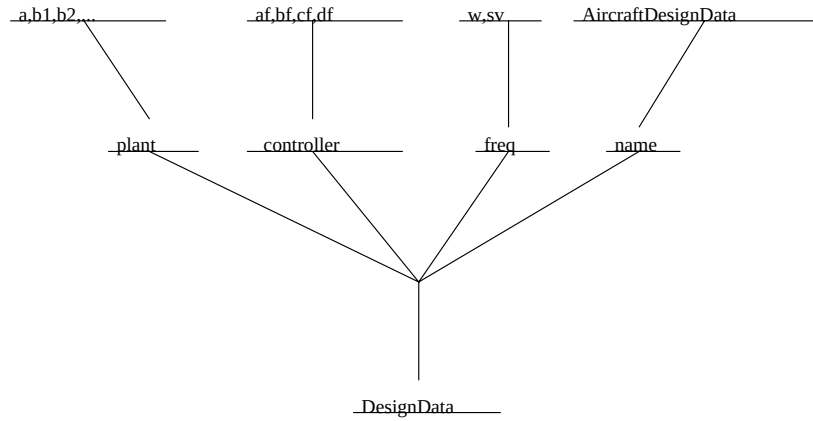


Figure 1-1: Branch Structure of the tree Variable

The M-files in the Robust Control Toolbox have been reinforced to optionally accept the system data structure to simplify user interaction and to reduce the amount of typing required. Whenever a Robust Control Toolbox function encounters a tree variable representing a system among it input arguments, it automatically checks to see if the variable is in fact a system. If it is a system, then the function automatically expands the input argument list, replacing the system variable by the matrices stored in it. For example, the following two commands perform the same computation.

```
hinf(TSS);
hinf(A,B1,B2,C1,C2,D11,D12,D21,D22);
```

The latter, longer form illustrates the fact that the use of system variables is entirely optional in the Robust Control Toolbox. The traditional but cumbersome method of passing system matrices one at a time as multiple arguments to a function is still acceptable, thus ensuring compatibility with other MATLAB toolboxes and earlier versions of the Robust Control Toolbox. See the Reference chapter for details on `mksys`, `branch`, and `tree`.

Singular Values, H^2 and H^∞ Norms

The *singular values* of a rank r matrix $A \in C^{m \times n}$, denoted σ_i are the non-negative square-roots of A^*A the eigenvalues of ordered such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p, \; p \; \min\{m, \; n\}$$

If $r < p$ then there are $p - r$ zero singular values, i.e.,

$$\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_p = 0$$

There exist two unitary matrices $U \in C^{m \times m}$, $V \in C^{m \times n}$ and a diagonal matrix $\Sigma \in R^{m \times n}$ such that

$$A = U\Sigma V^* = U\begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} V^*$$

where $\Sigma_r = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$; this is called the *singular-value decomposition* (SVD) of matrix A. The greatest singular value σ_1 is sometimes denoted

$$\bar{\sigma}(A) = \sigma_1$$

If A is a square $n \times n$ matrix, then the n-th singular value (i.e., the least singular value) is denoted

$$\bar{\sigma}(A) \triangleq \sigma_n$$

Some useful properties of singular values are

1 $\bar{\sigma}(A) = \max_{x \in C^n} \dfrac{\|Ax\|}{\|x\|}$

2 $\underline{\sigma}(A) = \min_{x \in C^n} \dfrac{\|Ax\|}{\|x\|}$

3 $\underline{\sigma}(A) \leq |\lambda_i(A)| \leq \bar{\sigma}(A)$, where λ_i denotes the i-th eigenvalue of A.

4 If A^{-1} exists, $\underline{\sigma}(A) = \dfrac{1}{\bar{\sigma}(A^{-1})}$

5 If A^{-1} exists, $\bar{\sigma}(A) = \dfrac{1}{\underline{\sigma}(A^{-1})}$

6 $\bar{\sigma}(\alpha A) = |\alpha|\bar{\sigma}(A)$

7 $\bar{\sigma}(A + B) \le \bar{\sigma}(A) + \bar{\sigma}(B)$

8 $\bar{\sigma}(AB) \le \bar{\sigma}(A)\bar{\sigma}(B)$

9 $\underline{\sigma}(A) - \bar{\sigma}(E) \le \underline{\sigma}(A + E) \le \underline{\sigma}(A) + \bar{\sigma}(E)$

10 $\max\{\bar{\sigma}(A),\bar{\sigma}(B)\} \le \bar{\sigma}([AB]) \le \sqrt{2}\max\{\bar{\sigma}(A),\bar{\sigma}(B)\}$

11 $\max_{i,j}|a_{i,j}| \le \bar{\sigma}(A) \le n \ \max_{i,j}|a_{i,j}|$

12 $\sum_{i=1}^{n} \sigma_i^2 = \mathrm{Trace}(A^*A)$

Property 1 is especially important because it establishes the greatest singular value of a matrix A as the maximal "gain" of the matrix as the input vector "x" varies over all possible directions.

For stable Laplace transform matrices $G(s) \in C^{m \times n}$, $p = \min\{m,n\}$, define the $\mathbf{H}^2$-norm and the $\mathbf{H}^\infty$-norm terms of the frequency-dependent singular values of $G(j\omega)$:

$\mathbf{H}^2$-norm:

$$\|G\|_2 \triangleq \left[\int_{-\infty}^{\infty} \sum_{i=1}^{p} (\sigma_i(G(j\omega)))^2 d\omega\right]^{\frac{1}{2}}$$

$\mathbf{H}^\infty$-norm:

$$\|G\|_\infty \triangleq \sup_{\omega} \bar{\sigma}(G(j\omega)) \quad \text{(sup: the least upper bound)}$$

The Robust Control Problem

In the past two decades there have been great advances in the theory for the design of robustly uncertainty-tolerant multivariable feedback control systems [8, 9]. Many of the questions that created the much lamented "gap" of the 1970's between the theory and practice of control design have been resolved, at least partially, in the wake of the renewed concern of control theorists with such feedback issues such as stability margin, sensitivity, disturbance attenuation and so forth. Out of this renewed concern has emerged the singular value Bode plot as a key indicator of multivariable feedback system performance (e.g., [12, 29]). The singular value thus joins such previously used measures of multivariable feedback system performance as dominant pole locations (related to disturbance rejection bandwidth and transient response), transmission zeros (related to steady-state response and "internal models") and rms error of control signals (from the $\mathbf{L}^2$ Wiener-Hopf/LQG optimal control theory, [1, 45, 46]).

The real problem in robust multivariable feedback control system design is to synthesize a control law which maintains system response and error signals to within prespecified tolerances despite the effects of uncertainty on the system. Uncertainty may take many forms but among the most significant are noise/disturbance signals and transfer function modeling errors. Another source of uncertainty is unmodeled nonlinear distortion. Uncertainty in any form is no doubt the major issue in most control system designs. Consequently people have adopted a standard quantitative measure for the size of the uncertainty, viz., the $\mathbf{H}^\infty$ norm.

The general *robust control problem* is described mathematically as follows (See Figure 1-2, Canonical Robust Control Problem):

Given a multivariable plant P(s), find a stabilizing controller F(s) such that the closed-loop transfer function $T_{y_1u_1}$ *satisfies*

$$\frac{1}{K_M(T_{y_1u_1}(j\omega))} < 1$$

where

$$K_M(T_{y_1u_1}) \stackrel{def}{=} \inf_{\Delta} \{\bar{\sigma}(\Delta) \mid \det((I - T_{y_1u_1})\Delta) = 0\}$$

with $\Delta = diag(\Delta_1,\ldots,\Delta_n)$

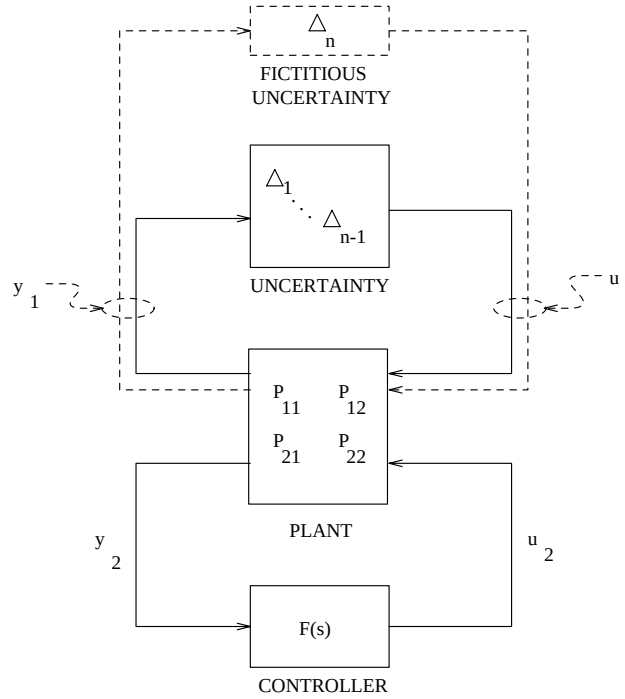


Figure 1-2: Canonical Robust Control Problem

The condition $1/K_M(T_{y_1u_1}(j\omega)) < 1$ is "robustness criterion." The quantity K_M is "size" of the smallest uncertainty Δ, as measured by the singular value at each frequency, that can destabilize the closed-loop system. The function K_M is the so-called *diagonally perturbed multivariable stability margin* (MSM) introduced by Safonov and Athans [30, 32], the reciprocal of which is known as μ, the *structured singular value* (SSV) [13] —. i.e.,$K_M = \frac{1}{\mu}$. More precisely, when Δ_n is not present, this problem is called the *robust stability problem*. Doyle, Wall and Stein [14] introduced the extra uncertainty Δ_n to represent the performance specification $\bar{\sigma}(T_{ed}(j\omega)) \le 1$ which, according to their *robust performance theorem*, is satisfied if and only if $1/K_M(T_{y_1u_1}(j\omega)) \le 1$. Thus, the problem set-up in Figure 1-2, Canonical Robust Control Problem completely addresses the issues in robust control system design, i.e., robustness and performance.

Unfortunately the computation of $K_M(T_{y_1u_1})$ involves a nonconvex optimization over Δ and so cannot, in general, be solved by the standard gradient-descent nonlinear programming techniques for which convexity of constraints and cost is required to assure convergence. Fortunately, computable upper bounds on $1/K_M$ do exist and have provided simple alternatives for computing K_M:

$$\frac{1}{K_m(T_{y_1u_1})} = \mu(T_{y_1u_1}) = \inf_{D \in \mathbf{D}} \left\| DT_{y_1u_1}D^{-1} \right\|_\infty = \left\| D_p T_{y_1u_1} D_p^{-1} \right\|_\infty$$

where $D_p \in D$ denotes the Perron optimal scaling matrix [32], and $\mathbf{D} := \{\mathrm{diag}|(d_1 I,\ldots,d_n I) \mid d_i > 0\}$. Clearly, $\left\| T_{y_1u_1} \right\|_\infty$ is also an upper bound on $1/K_M$, albeit possibly a very conservative one. If any of the upper bounds satisfies the robust performance constraints, it is sufficient to guarantee that μ, or K_M, satisfies them as well.

Therefore, from a robust control *synthesis* point of view, the problem is to find a stabilizing *F(s)* to "shape" the $\mu(T_{y_1u_1})$ function (or its upper bounds) in the frequency domain. On the other hand, from a robust control *analysis* point of view, the problem is to compute the MSM $K_m(T_{y_1u_1})$, (or its bounds).

Structured and Unstructured Uncertainty

Practically, each of the Δ_i's (i = 1, …, n) may itself be a matrix and represent a different kind of physical uncertainty. Two types of uncertainty are defined in robust control — *unstructured* and *structured*.

Unstructured uncertainty usually represents frequency-dependent elements such as actuator saturations and unmodeled structural modes in the high frequency range or plant disturbances in the low frequency range. Their relations to the *nominal plant* can be either *additive*

$$G = \overline{G} + \Delta_A$$

or *multiplicative*

$$G = (I + \Delta_M)\overline{G}$$

Both can be considered as norm bounded quantities, i.e., using $\mathbf{H}^\infty$ norm $\|\Delta\|_\infty < r$. where r is a given positive number.

Figure 1-3, Additive and Multiplicative Unstructured Uncertainty shows the block diagrams of these two unstructured uncertainties.

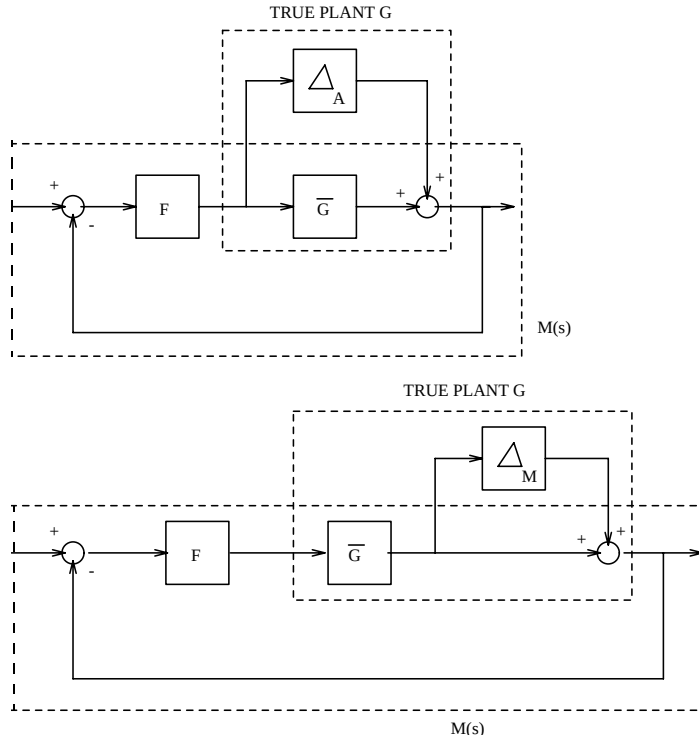


Figure 1-3: Additive and Multiplicative Unstructured Uncertainty

Structured Uncertainty represents parametric variations in the plant dynamics, for example:

1. Uncertainties in certain entries of state-space matrices (*A, B, C, D*), e.g., the uncertain variations in an aircraft's stability and control derivatives.
2. Uncertainties in specific poles and/or zeros of the plant transfer function.
3. Uncertainties in specific loop gains/phases.

The very general setup in Figure 1-2, Canonical Robust Control Problem allows a control system designer to capture all these uncertainties, both structured and unstructured, and formulate them into the design. The provides software tools for robustness analysis and robust control law synthesis within this very general framework.

Positive Real and Sector Uncertainty

The setup of the *robust control problem* in Figure 1-2, Canonical Robust Control Problem handles much more than just the case of $\Delta_i(j\omega)$ satisfying $\|\Delta_i\|_\infty \le 1$. Using the *sector transform* [50, 28], this setup readily extends to admit transfer function matrix $\Delta_i(s)$'s and even nonlinear Δ_i's satisfying a general, possibly frequency-dependent sector condition.

Definition: *Given matrices A(s) and B(s) and let* $\Delta(s)$ *be a stable transfer function matrix. If*

$$Re[(y - Ax)^*(y - Bx)] \le 0$$

for all $s = j\omega$ *and* $y = \Delta(j\omega)x$*, then we say*

$$\Delta(s) \subset \text{sector}[A,B]$$

More generally, if $A_{11}(s)$*,* $S_{12}(s)$*,* $S_{21}(s)$*,* $S_{22}(s)$ *are stable transfer function matrices and if*

$$Re[(S_{11}(s)x + S_{12}(s)y)^*(S_{21}(s)x + S_{22}(s)y)] \le 0$$

for all $s = j\omega$ *and all* $y = \Delta(j\omega)x$*, then we say*

$$\Delta(s) \subset \text{sector}[S(s)]$$

For example, physically-dissipative force-velocity transfer function matrices such as those associated with mechanical structures having collocated actuators and sensors are positive real, i.e., inside *sector[0,* ∞], and the transformation

$$\bar{y} = y + u$$
$$\bar{u} = y + u$$

transforms a positive-real relation $u = \Delta y$ into an equivalent relation $\tilde{u} = \Delta\tilde{y}$ satisfying

$$\left\|\tilde{\Delta}\right\|_\infty < 1$$

The case of general $A(s)$, $B(s)$ matrices may be handled similarly.

The function `sectf.m` in the Robust Control Toolbox allows you to perform the sector transform in the state-space framework. See the Reference section for details.

Robust Control Analysis

The goal of robust analysis is to measure the *Multivariable Stability Margin* (MSM) "seen" by the uncertainties using a proper, nonconservative analytical tool. In other words, we are interested in finding out *how big* Δ *can be before instability occurs*.

Two tasks are involved in computing the MSM:

Task 1: Define the uncertainty model

Task 2: Pull out the uncertainty channels (structured or unstructured) into a M-Δ form as shown in Figure 1-4, Robust Analysis M- Diagram.

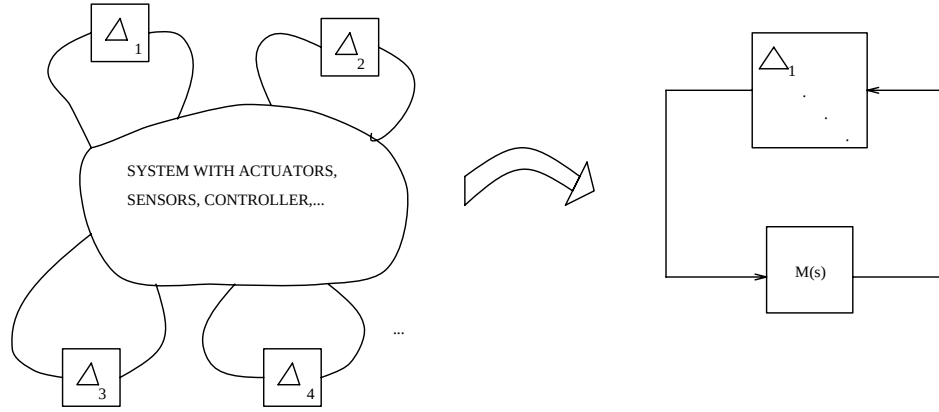


Figure 1-4: Robust Analysis M-Δ Diagram

Following are examples of modeling different types of uncertainties in the M-Δ block diagram form.

Example 1: Modeling Unstructured Uncertainty. The following plant transfer function that represents a spacecraft's rigid body dynamics and one boom structural mode (see the *Case Studies* section for more details).

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

If the nominal model is $\overline{G} = \frac{1}{s^2}$, then (see Figure 1-5, Bode Plots of Additive

$$-\Delta_A = \overline{G} - G = \frac{s^2+1}{s^2(s^2+2)};\ M(s) = -F(I+F\overline{G})^{-1}$$

and Multiplicative Uncertainty)

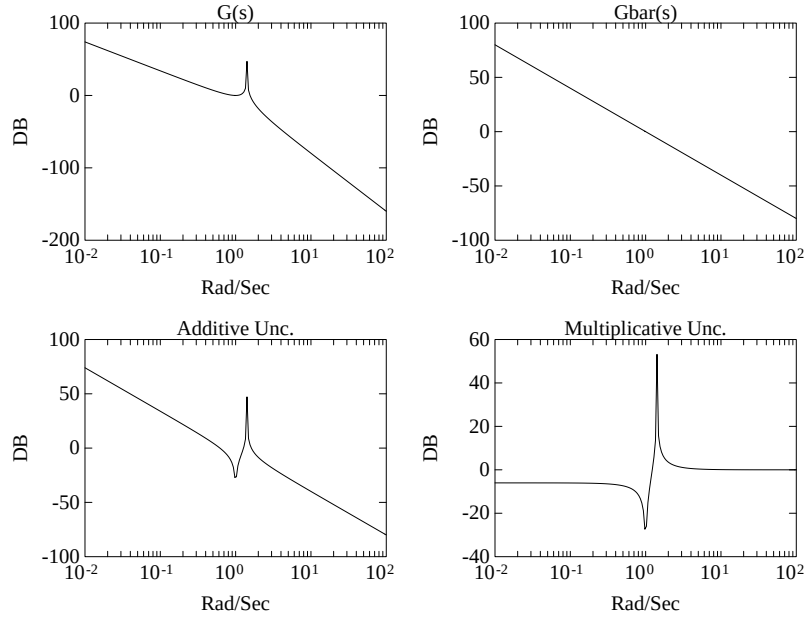


Figure 1-5: Bode Plots of Additive and Multiplicative Uncertainty

$$-\Delta_M = (\overline{G} - G)\overline{G}^{-1} = \frac{s^2 + 1}{s^2 + 2}; \qquad -M(s) = \overline{G}F(I + \overline{G}F)^{-1}$$

Example 2. Modeling Structured Uncertainty . This example shows how to pull out structured uncertainties from state-space A and B matrices.

The state-space model of the lateral-directional dynamics of a fighter aircraft is shown below [5].

$$\begin{bmatrix} \dot{p} \\ \dot{r} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} L'_p & L'_r & L'_\beta/V_T \\ N'_p & N'_r & N'_\beta/V_T \\ Y_p & Y_r & Y_\beta/V_T \end{bmatrix} \begin{bmatrix} p \\ r \\ v \end{bmatrix} + \begin{bmatrix} L'_{\delta_a} & L'_{\delta_r} \\ N'_{\delta_a} & N'_{\delta_r} \\ Y_{\delta_a} & Y_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

$$\begin{bmatrix} \beta \\ \dot{\mu}_{app} \end{bmatrix} = \frac{180}{\pi} \begin{bmatrix} 0 & 0 & 1/V_T \\ \cos\alpha & \sin\alpha & 0 \end{bmatrix} \begin{bmatrix} p \\ r \\ v \end{bmatrix}$$

where the "primed" derivatives with respect to $(p, r, \beta, \delta_a, \delta_r)$ are defined as

$$L' = \left(L + \frac{I_{xz}}{I_{xx}} N\right) \frac{I_{xx} I_{zz}}{I_{xx} I_{zz} - I_{xz}^2}$$

$$N' = \left(N + \frac{I_{xz}}{I_{xx}} L\right) \frac{I_{xx} I_{zz}}{I_{xx} I_{zz} - I_{xz}^2}$$

The aircraft is trimmed at degrees angle of attack, flying at sea level with a total velocity V_T of 334.9 ft/sec. The states to be stabilized are body-axis roll rate (p), yaw rate (r) and the velocity component along the y-axis (v). The variables to be controlled (tracked) are the roll-rate about the velocity vector ($\dot{\mu}$) and the sideslip angle (β). The control actuators whose dynamics are ignored in this analysis are aileron (δ_a)and rudder (δ_r).

The plant data is

$$\begin{bmatrix} A & B_1 \\ C_1 & D_1 \end{bmatrix} := \left[\begin{array}{ccc|cc} -1 \cdot 9953 & 0 \cdot 7513 & -0.0299 & 0.0906 & 0.0298 \\ -1 \cdot 0093 & -0 \cdot 1518 & 0.0060 & -0.0024 & -0.0204 \\ 39 \cdot 8500 & -331 \cdot 90 & -0.1673 & 0.0204 & 0.2284 \\ \hline 0 & 0 & -0.1711 & 0 & 0 \\ 56 \cdot 8927 & 6 \cdot 7840 & 0 & 0 & 0 \end{array}\right]$$

The state-space set-up for the robustness evaluation can be formulated using the block diagram in Figure 1-6, Pulling Out Parametric Uncertainties.

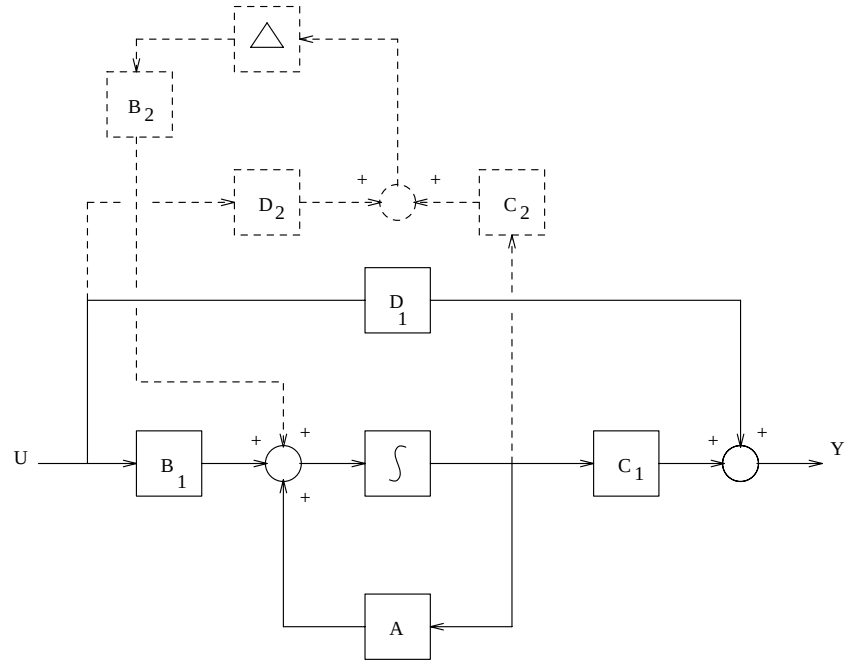


Figure 1-6: Pulling Out Parametric Uncertainties

The equations associated with the block diagram are

$$\dot{x} = Ax + B_1 u_1 + B_2 y_2$$
$$y_1 = C_1 x + D_1 u_1$$
$$y_2 = \Delta C_2 x + \Delta D_2 u_1$$

which lead to the perturbed state-space system

$$\dot{x} = (A + \underbrace{B_2 \Delta C_2}_{\Delta_A}) x + (B_1 + \underbrace{B_2 \Delta D_2}_{\Delta_B}) u_1$$

where matrices B_2, C_2 and D_2 play the roles of putting the parametric uncertainty block Δ into a diagonal structure.

If $L'_p, N'_p, L'_r, N'_r, L'_\beta, N'_\beta$ and $L'_{\delta_a}, L'_{\delta_r}, N'_{\delta_a} N'_{\delta_r}$ are the perturbations for the A and B_1 matrices respectively, then the associated Δ, B_2 and C_2 will have the following structure

$$\Delta = diag(\Delta_{L'_p}, \Delta_{N'_p}, \Delta_{L'_r}, \Delta_{N'_r}, \Delta_{L'_\beta}, \Delta_{N'_\beta}, \Delta_{L'_{\delta_a}}, \Delta_{N'_{\delta_a}}, \Delta_{L'_{\delta_r}}, \Delta_{N'_{\delta_r}})$$

$$B_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$D_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T$$

The overall augmented plant becomes

$$P(s) = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_1 & 0 \\ C_2 & D_2 & 0 \end{array}\right]$$

The linear fractional transformation `lftf` can be used to close the controller feedback loop *F(s)* around the plant from u_1 to y_1. Then, the transfer function *M(s)* "seen" by the uncertainty blocks is the transfer function from u_2 to y_2.

Example 3. Modeling Nonlinear Uncertainty. A saturation nonlinear element can be modeled as unstructured uncertainty sector bounded element inside *sector* [0,1]which, according to the nonlinear stability results of Sandberg and Zames [26, 50], may be effectively modeled as an uncertain linear time-invariant element whose Nyquist locus lies inside a complex disk of radius 0.5 centered on the real axis at 0.5 (See Figure 1-7, Modeling Nonlinearity as Unstructured Uncertainty). This uncertain linear time-invariant element may thus be decomposed as $0.5 + \Delta_A$ where the

additive uncertainty Δ_A is bounded by $\|\Delta_A\|_\infty < 0.5$. For robust stability $\|M\|\infty\|\Delta\|\infty < 1,\ \|M\|\infty < 2$.

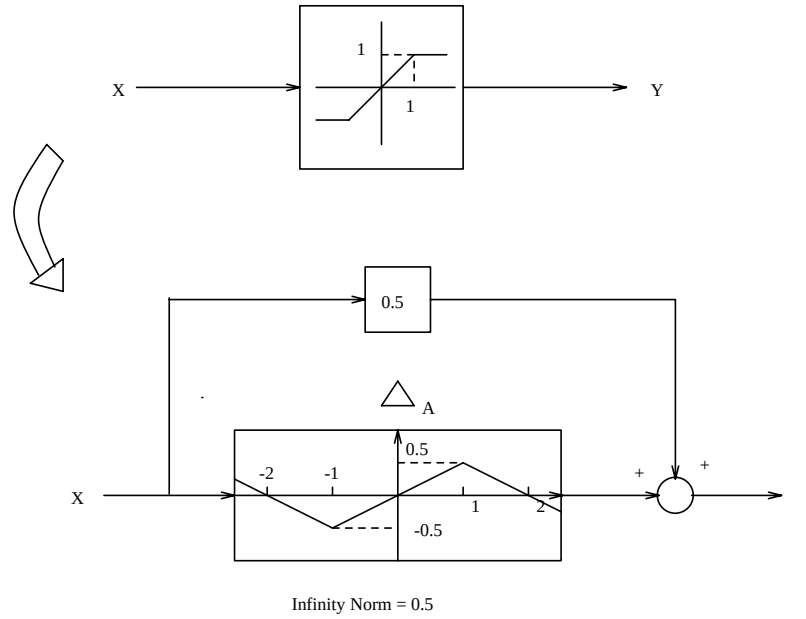


Figure 1-7: Modeling Nonlinearity as Unstructured Uncertainty

Robust Analysis — Classical Approach

First, let's recall classical definitions of SISO stability margin (robustness). Consider the following block diagram (Figure 1-8, Classical Gain/Phase Margins). The *gain margin* can be defined as the variation of real(Δ), and the *phase margin* can be defined as the variation of imag(Δ). On the Nyquist plot they are simply the intersections of loop transfer function on unit circle (phase margin) and real-axis (gain margin).

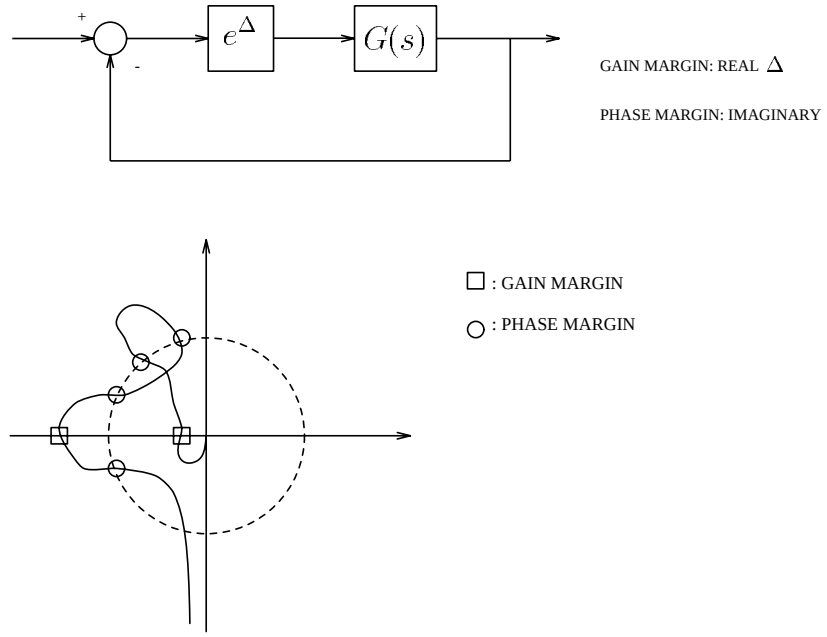


Figure 1-8: Classical Gain/Phase Margins

A simple example shown in Figure 1-9, Gain/Phase Margins ≠ Robustness reveals immediately that classical SISO gain/phase margins DO NOT represent system robustness. This is because you can have infinite gain margin and 90 degree phase margin as shown in Figure 1-9, Gain/Phase Margins ≠ Robustness but still be very close to instability (-1 critical point). In other words, classical gain/phase margins cannot capture simultaneous variations in both quantities.

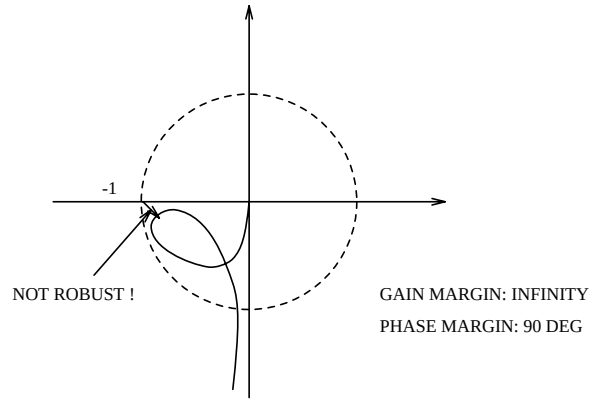


Figure 1-9: Gain/Phase Margins ≠ Robustness

For MIMO systems, classical gain/phase margins computed one-loop-at-a-time obviously ignore the effects of simultaneous variations in several loops and the cross coupling between them. In 1978, Doyle [11] described an interesting example showing how classical gain/phase margins can be dangerously optimistic in predicting system stability (robustness). Figure 1-10, MIMO Robustness vs. Gain/Phase Margin shows the system block diagram and stability hyperplanes in 2-D parameter space. Clearly, when $\Delta_1 = \Delta_2 = 0$, then the loop transfer function in each of the two feedback loops is $1/s$. So each individual loop has gain-margin $G_M = \pm\infty$ and phase-margin $\Phi_M = \pm 90$ deg. But simultaneous variations in Δ_1 and Δ_2 quickly lead to system instability.

British control theorist A. G. J. MacFarlane introduced a sophisticated analysis tool called *Characteristic Gain Loci* to measure the system robustness (e.g., [25]). The idea was to compute the gain/phase margins of each eigenvalue of the loop transfer function *L(s)*, then determine the MIMO system robust stability based on the *Generalized Nyquist Stability Theorem*:

A system is stable if and only if the Nyquist loci of the eigenvalues of the loop transfer function encircle "-1" once counterclockwise for each unstable pole of L(s).

However, characteristic gain loci may give too optimistic a result as shown in the following example.

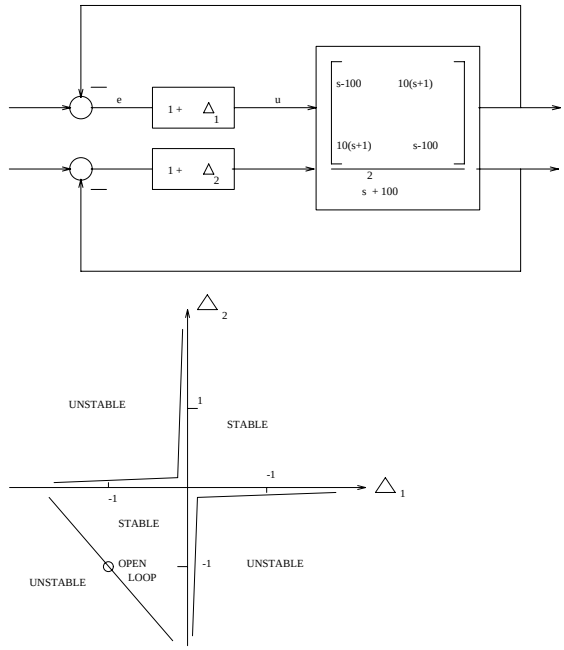


Figure 1-10: MIMO Robustness vs. Gain/Phase Margin

Example: [11]

Let's consider the 2×2 plant transfer function matrix

$$G(s) = \begin{bmatrix} \dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)} \\ \dfrac{-42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)} \end{bmatrix}$$

with modal decomposition

$$G(s) = X\Lambda(s)X^{-1} = \begin{bmatrix} 7 & -8 \\ -6 & 7 \end{bmatrix} \begin{bmatrix} \dfrac{1}{s+1} & 0 \\ 0 & \dfrac{2}{s+2} \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 6 & 7 \end{bmatrix}$$

A stabilizing feedback controller is

$$K = \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The characteristic gain loci containing in $\Lambda(s)$ *seem* to imply that the system has infinite gain margin and ± 180 degree phase margin in each feedback loop. However, if you slightly perturb the gains K_1 and K_2 to $K_1 + 0.13$ and $K_2 - 0.12$ simultaneously, the system becomes unstable.

This is not surprising from a numerical linear algebra viewpoint, because computing eigenvectors is a numerically sensitive process in that a small variation in one matrix element can result huge changes in the eigenvectors.

Robust Analysis — Modern Approach

In the late 1970's, people started to realize [29, 12] that, by using the singular value and its related robustness tests, it would be possible to substantially overcome the difficulties associated with the classical methods.

A standard Singular-Value Stability Robustness Theorem was established based on the Sandberg-Zames' Small Gain Theorem [26, 50]:

The M- Δ *system is stable for any stable* $\Delta(s)$ *satisfying*

$$\sigma(\Delta(j\omega)) < \frac{1}{\sigma[M(j\omega)]}$$

for all $\omega \in R\left(or \ \|\Delta\|_\infty < \frac{1}{\|M\|_\infty}\right)$

Several important practical considerations in applying the Theorem follow:

1. A small change in Δ never produces a large change in $\bar{\sigma}(\Delta)$ or vice versa (i.e., singular values are better than eigenvalues for robust analysis).

2. Although the theorem gives only, sufficient condition for robust stability, these conditions are also necessary for stability in the weakened sense that

there exists some Δ with $\|\Delta\|_\infty = 1/\|M\|_\infty$ such that the closed-loop system is not asymptotically stable.

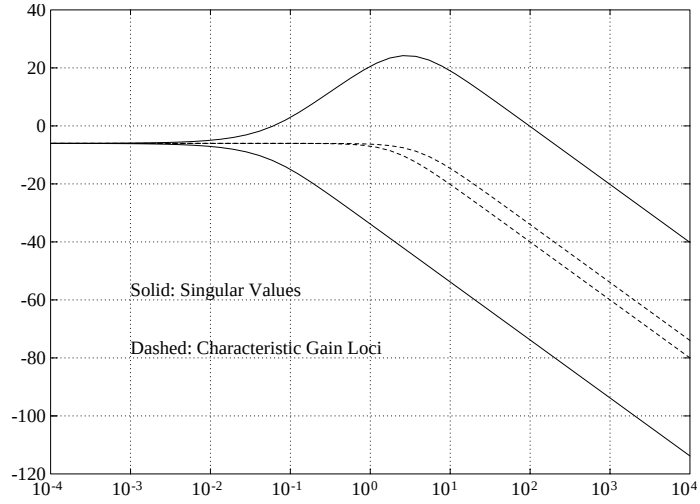


Figure 1-11: Singular Value vs. Characteristic Gain Loci

Revisiting the Characteristic Gain Loci example, you can predict system robustness accurately using this Singular Value Stability Robustness Theorem. See Figure 1-11, Singular Value vs. Characteristic Gain Loci.

Applying the Small Gain Theorem, the resonance peak of the maximum singular value $\|M\|_\infty$ (≈ 16.2695) predicts accurately that the multiplicative uncertainty can only be as large as $\frac{1}{16.2695} = 6.15\%$ before instability occurs.

Now we can formally introduce the concept of *Multivariable Stability Margin* (MSM). Recall MSM is defined as

$$K_M(M) = 1/\mu(M) = \inf_{\Delta}\{\bar{\sigma}(\Delta) | \det(I - M\Delta) = 0\}$$

where $\Delta = \mathrm{diag}(\Delta_1,\ldots,\Delta_n)$.

Properties of K_M and μ

1. K_M is the smallest $\bar{\sigma}(\Delta)$ which can destabilize the system $(I - M\Delta)^{-1}$.
2. If no Δ exists such that $\det(I - M\Delta) = 0$, $K_m = \infty$.
3. K_M is a function of M and the structure of Δ.
4. $\mu(\alpha M) = |\alpha|\mu(M)$, for any scalar α.
5. $\rho(M) \le \mu(M) \le \bar{\sigma}(M)$, where ρ is the spectral radius.
6. If $\Delta = \delta I$ for some $\delta \in C$, then $\mu(M) = \bar{\rho}(M)$.
7. If $\Delta \subset C^{n \times n}$ full matrix, $\mu(M) = \bar{\sigma}(M)$.
8. $\max_{U \in \mathrm{U}} \rho(MU) = \mu(M)$, where is U the set of all unitary matrices with the same (block) diagonal structure as Δ. This is a nonconvex optimization problem, which is impractical to solve exactly as mentioned earlier.
9. Generalized Small-Gain Theorem: If nominal *M(s)* is stable, then the perturbed system $(I - M\Delta)^{-1}$ is stable for all stable Δ_i for which $\|\Delta_i\|\infty \le 1$ if and only if $K_m(M(j\omega)) > 1$ for all $\omega \in R$.

Diagonal Scaling

In 1981, Safonov [31] introduced the concept of diagonal scaling to compute the upper bounds of MSM. See Figure 1-12, The Concept of Diagonal Scaling

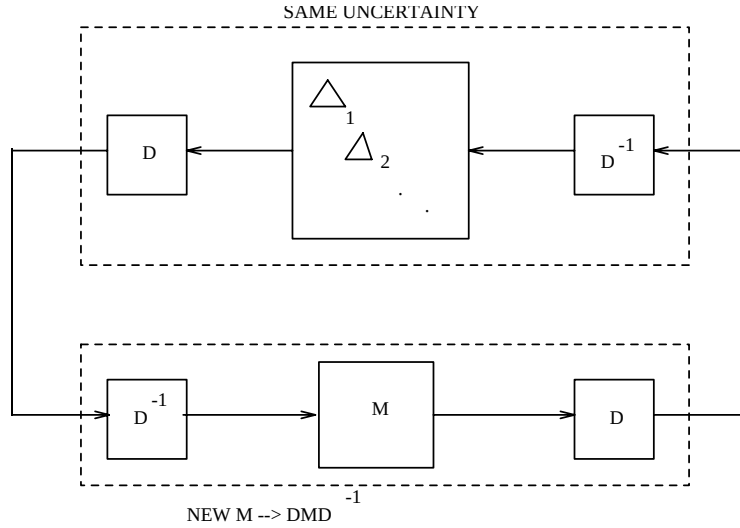


Figure 1-12: The Concept of Diagonal Scaling

The idea is as follows: If Δ and D are diagonal matrices, $\|\Delta\|_\infty = \left\|D^{-1}\Delta D\right\|_\infty$, but $\left\|DMD^{-1}\right\|_\infty$ can be much smaller than $\|M\|_\infty$. This fact leads to the following K_M upper bounds which much more accurately predict MIMO system robustness

$$\frac{1}{K_M} = \mu(M) \le \inf_{D \in D} \left\|DMD^{-1}\right\|_\infty \le \left\|D_p M D_p^{-1}\right\|_\infty$$

where $D_p \in D$ denotes the Perron optimal scaling matrix [31, 32] and $D\colon = \{\ \mathrm{diag}(d_1, I, \ldots, d_n I) \mid d_i > 0\}$. Clearly, the "unscaled" Singular Value Stability Robustness Theorem uses the most conservative upper bound on K_M to predict system MSM, whereas the scaled singular value can be much more accurate.

In the Robust Control Toolbox, several functions are provided to compute various upper bounds on a multivariable system's Structured Singular Value (SSV) $1/K_M(T_{y_1 u_1})$:

- Singular value: `sigma.m`, `dsigma.m`

- Perron diagonal scaling: `psv.m`, `ssv.m`
- Osborne diagonal scaling: `osborne.m`, `ssv.m`
- Multiplier scaling: `muopt.m`, `ssv.m`
- Characteristic gain loci: `cgloci.m` `dcgloci.m`

A comparison of the available upper bounds on the structured singular value reveals that some are much easier to compute than others and some can be more conservative than others. See Table 1-1, and the example.

Table 1-1:

Method	Property	Computation	Reference
Optimal Diagonal Scaling	$n = 3$, exact K_M $n > 3$, $\exists$ 15 % gap	`demanding`	`Doyle [11]` `Safonov [31}`
Diagonal Scaling `psv.m`, `osborne.m, ssv.m`	very close to optimal diagonal scaling	`easy`	`Safonov [31,32]`
Singular Value `sigma.m`, `dsigma.m`	can be very conservative	`easy`	`Safonov [28]` `Doyle [11]`
Multiplier Scaling `muopt.m, ssv.m`	allow mixed real and complex uncertainties	`demanding`	`Safonov et al. [44]`

The following example reveals that singular values can be excessively conservative in predicting the MSM when you know more about Δ.

Example: Given a system having nominal loop transfer function

$$G(s) = \frac{1}{s^2 + 12s + 32}\begin{bmatrix} 4s + 32 & 0 \\ 12s^2 + 64s & 8s + 32 \end{bmatrix}$$

with multiplicative uncertainty Δ at its input, find the SSV of the transfer function $G(I+G)^{-1}$ "seen" by Δ as a feedback wrapped around it.

To compute the result shown in Figure 1-13, SSV (Perron Upper Bound) vs. Singular Value, simply execute the following commands.

```
num = [0 4 32; 12 64 0; 0 0 0; 0 8 32];
den = [1 12 32]; m = 2; n = 2;
tfm = mksys(num,den,m,n,'tfm');
ssg = tfm2ss(tfm);
w = logspace(-3,3);
perron = 20*log10(ssv(ssg,w));
svmax=10*log10(max(sigma(ssg,w)));
semilogx(w,svmax,'k:',w,perron,'k-')
ylabel('DB'); xlabel('Rad/Sec');
legend('Singular Value','Perron',3)
```

More details about the algorithm associated with each method can be found in the Reference section under `psv`, `osborne` and `ssv`.

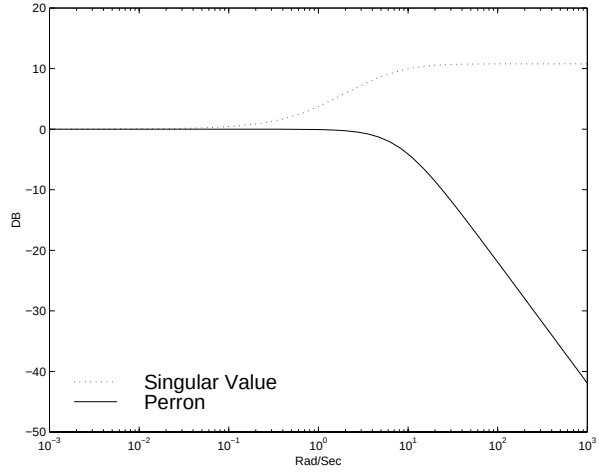


Figure 1-13: SSV (Perron Upper Bound) vs. Singular Value

Robust Control Synthesis

The recently developed $\mathbf{H}^{\infty}$, frequency-weighted LQG, LQG loop transfer recovery (LQG/LTR) and μ synthesis theories have made multivariable loop shaping a routine matter. The $\mathbf{H}^{\infty}$ theory provides a direct, reliable procedure for synthesizing a controller which optimally satisfies singular value loop shaping specifications. The frequency-weighted LQG optimal synthesis theory (also known as the "$\mathbf{H}^{2}$ theory" and "Wiener-Hopf Theory") and LQG/LTR lead to somewhat less direct, but nonetheless highly effective iterative procedures for massaging singular value Bode plots to satisfy singular value loop shaping specifications. On the other hand, the μ synthesis technique puts both "robust analysis" and "robust synthesis" problems in a single framework in which you shape the function μ (or K_M); this offers the maximum flexibility as a general robust control system design tool. Table 1-2, summarizes the techniques available in the as well as the advantages and shortcomings of each

Table 1-2:

Methods	Advantages	Disadvantages
LQR (`lqr.m`)	• Guaranteed stability margin • Pure gain controller	Need full-state feedback Need accurate model Possibly many iterations
LQG (`lqg.m`)	• Uses available noise data	No stability margin guaranteed Need accurate model Possibly many iterations
LQG/LTR (`ltru.m`, `ltry.m`)	• Guaranteed stability margin • Systematic design procedure	High gain controller Possibly many iterations Design focus on one point
$\mathbf{H}^{2}$ (`h2lqg.m`)	• Addresses stability and sensitivity • Almost exact loop shaping • Closed-loop always stable	Possibly many iterations

Table 1-2:

Methods	Advantages	Disadvantages
$\mathbf{H}^\infty$	• Addresses stability and sensitivity • Exact loop shaping • Direct one-step procedure	Requires special attention to the plant parametric robustness
μ synthesis (`musyn.m`)	• Combines structured/unstructured uncertainty in design	Problem is nonconvex. Controller size is huge (2n to 3n).

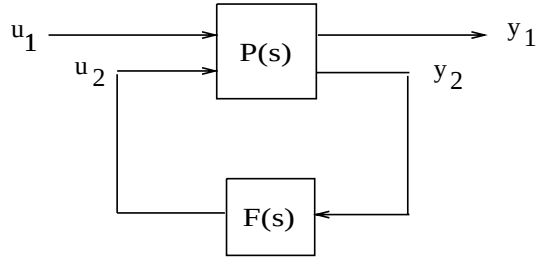


Figure 1-14: Small Gain Problem

A number of methods are available in the Robust Control Toolbox to design a robust stabilizing feedback control law such that the robustness inequality $\|T_{y_1u_1}\| < 1$ is satisfied:

- *LQG loop transfer recovery* (use `lqr`, then `ltru` or `ltry`).
- $\mathbf{H^2}$ optimal control synthesis (use `h2lqg`).
- $\mathbf{H}^\infty$ optimal control synthesis (use `hinf`, `hinfopt` or `linf`).

Figure 1-14, Small Gain Problem shows a general set-up, and the problem of making $\|T_{y_1u_1}\|_\infty \le 1$ is also called the *small-gain problem*.

LQG and Loop Transfer Recovery

The regular LQG-based design problem can be solved via the M-file `lqg` based on the *separation principle*, which is explained in the Reference section of this toolbox.

The loop transfer recovery procedure (`ltru`, `ltry`) developed by Doyle and Stein is documented in the Reference section. A multivariable fighter example is also included.

H^2 and H^∞ Synthesis

The methods of $\mathbf{H}^2$ and $\mathbf{H}^\infty$ synthesis are especially powerful tools for designing robust multivariable feedback control systems to achieve singular value loop shaping specifications. The Robust Control Toolbox functions `h2lqg`, `hinf` and `hinfopt` compute continuous-time $\mathbf{H}^2$ and $\mathbf{H}^\infty$ control laws; their discrete-time counterparts are `dh2lqg`, `dhinf` and `dhinfopt`.

The $\mathbf{H}^2$ or $\mathbf{H}^\infty$ design problem can be formulated as follows: Given a state-space realization of an "augmented plant" *P(s)* (e.g., as in Figure 1-14, Small Gain Problem)

$$P(s) := \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right],$$

find a stabilizing feedback control law

$$u_2(s) = F(s)y2(s)$$

such that the norm of the closed-loop transfer function matrix

$$T_{y1u1} = P_{11}(s) + P_{12}(s)(I - F(s)P_{22}(s))^{-1}F(s)P_{21}(s)$$

is small. Three such problems addressed by the Robust Control Toolbox are

$\mathbf{H^2}$ Optimal Control: $\min \left\| T_{y_1 u_1} \right\|_2$

$\mathbf{H}^\infty$ Optimal Control: $\min \left\| T_{y_1 u_1} \right\|_\infty$

Standard $\mathbf{H}^\infty$ Control: $\min(\left\| T_{y_1 u_1} \right\|_2 \le 1)$

The *standard* $\mathbf{H}^\infty$ *control problem* is sometimes also called the $\mathbf{H}^\infty$ *small gain problem*. Both $\mathbf{H}^2$ and $\mathbf{H}^\infty$ synthesis are often used together, with $\mathbf{H}^2$ synthesis being used as a first cut to get a sense for what level of performance is achievable. Then, an $\mathbf{H}^\infty$ performance criterion is selected based on the outcome

of the first cut $\mathbf{H}^2$ design, and the $\mathbf{H}^\infty$ synthesis theory is used to do the final design work.

The entire design procedure is simple and requires only "one-parameter" γ-iteration following path I then path II of the flow-chart in Figure 1-15, H2/H∞ γ-Iteration.

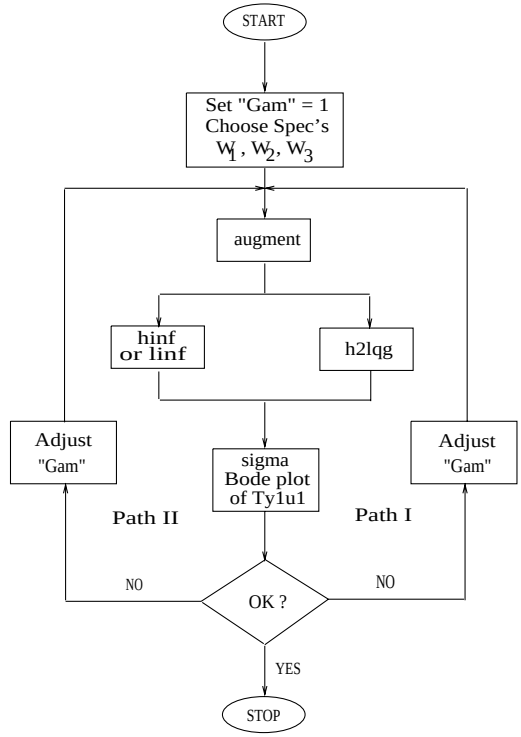


Figure 1-15: H^2/H^∞ γ-Iteration

Properties of $\mathbf{H}^\infty$ Controllers

There are several important properties of $\mathbf{H}^\infty$ controllers worth mentioning (see, for example, [3]):

Property 1: The $\mathbf{H}^\infty$ optimal control cost function $T_{y_1u_1}$ is all-pass, i.e., $\bar{\sigma}(T_{y_1u_1}) = 1$ for all $\omega \in$ R.

Property 2: An $\mathbf{H}^{\infty}$ "sub-optimal" controller produced by the standard state-space algorithm `hinf.m` has the same number of state variables as the augmented plant (n-states). An *optimal* $\mathbf{H}^{\infty}$ l controller such as produced by `hinfopt.m` has at most $(n-1)$ states — at least one of the states goes away (e.g., [20]).

Property 3: In the *weighted mixed sensitivity* problem formulation, the $\mathbf{H}^{\infty}$ controller always cancels the stable poles of the plant with its transmission zeros [4].

Property 4: In the *weighted mixed sensitivity* problem formulation [34], any unstable pole of the plant inside the specified control bandwidth will be shifted approximately to its $j\omega$ -axis mirror image once the feedback loop is closed with an $\mathbf{H}^{\infty}$ (or $\mathbf{H}^{2}$) controller. The $\mathbf{H}^{\infty}$ mixed-sensitivity problem is described in the next section.

Property 1 means that designers can ultimately achieve very precise frequency-domain loop-shaping via suitable weighting strategies. For example, you may augment the plant with frequency dependent weights W_1, and W_3 as shown in Figure 1-19. Then, if there exists a feasible controller that meets the frequency domain constraints, the software `hinf` will find one that ultimately "shapes" the signals to the inverse of the weights. This remarkable property enables $\mathbf{H}^{\infty}$ to achieve the best results — allowing much more precise manipulation of singular-value Bode plots than other loop-shaping tools such as LQG/LTR or $\mathbf{H}^{2}$.

Several design case studies are provided in the next section.

Existence of $\mathbf{H}^{\infty}$ Controllers

If you impose overly demanding design requirements then the minimal achievable $\mathbf{H}^{\infty}$ norm may be greater than one, in which case no solution exists to the standard $\mathbf{H}^{\infty}$control problem. The $\mathbf{H}^{\infty}$ theory gives the following four *necessary and sufficient* conditions for the existence of a solution to the standard $\mathbf{H}^{\infty}$ control problem [42]:

1. D_{11} Small Enough. There must exist a constant feedback control law $F(s)$ = "constant matrix" such that the closed-loop D matrix satisfies $\bar{\sigma}(D) < 1$.

2. Control Riccati $P \geq 0$. The $\mathbf{H}^{\infty}$ full-state feedback control Riccati equation must have a real, positive semidefinite solution P. The software ascertains existence of the Riccati solution by checking that the associated

Hamiltonian matrix does not have any $j\omega$-axis eigenvalues. Positive semidefiniteness of the solution is verified by checking, equivalently, that the $\mathbf{H}^{\infty}$ full-state feedback control-law is asymptotically stabilizing [42]; this circumvents numerical instabilities inherent in directly checking positive semidefiniteness.

3 Observer Riccati $S \geq 0$. The Riccati equation associated with the observer dual of the $\mathbf{H}^{\infty}$ full-state feedback control problem must have a real, positive semidefinite solution S. Again the results of [42] are used to avoid numerical instabilities.

4 $\lambda_{max}(PS) < 1$ The greatest eigenvalue of the product of the two Riccati equation solutions must be less than one.

All four conditions must hold for there to exist a feedback control law which solves the standard $\mathbf{H}^{\infty}$ control problem. The functions `hinf` and `hinfopt` automatically check each of these four conditions and produce displays indicating which, if any, of the conditions fail. Typical output displays for `hinf` and `hinfopt` *are shown in* Table 1-1, and Table 1-2, in the *Case Studies* section.

Singular-Value Loop-Shaping: Mixed-Sensitivity Approach

Consider the multivariable feedback control system shown in Figure 1-16, Block Diagram of the Multivariable Feedback Control System. In order to quantify the multivariable stability margins and performance of such systems, you can use the singular values of the closed-loop transfer function matrices from r to each of the three outputs e, u and y, viz.

$$S(s) \stackrel{def}{=} (I + L(s))^{-1}$$

$$R(s) \stackrel{def}{=} F(s)(I + L(s))^{-1}$$

$$T(s) \stackrel{def}{=} L(s)(I + L(s))^{-1} = I - S(s)$$

where $L(s) = G(s)F(s)$.

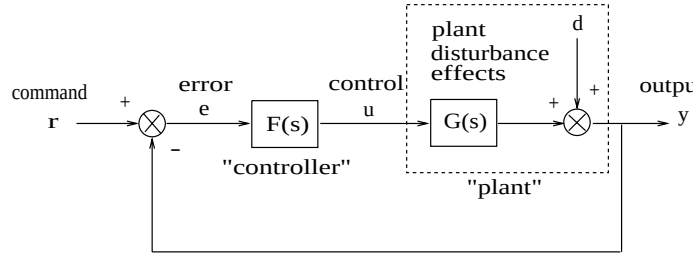


Figure 1-16: Block Diagram of the Multivariable Feedback Control System

The two matrices *S(s)* and *T(s)* are known as the *sensitivity function* and *complementary sensitivity function*, respectively. The matrix *R(s)* has no common name. The singular value Bode plots of each of the three transfer function matrices *S(s)*, *R(s)*, and *T(s)* play an important role in robust multivariable control system design. The singular values of the loop transfer function matrix *L(s)* are important because *L(s)* determines the matrices *S(s)* and *T(s)*.

The singular values of *S(jω)* determine the disturbance attenuation since *S(s)* is in fact the closed-loop transfer function from disturbance *d* to plant output *y* — *see Figure 1-16, Block Diagram of the Multivariable Feedback Control System*. Thus a disturbance attenuation performance specification may be written as

$$\bar{\sigma}(S(j\omega)) \le \left|W_1^{-1}(j\omega)\right|$$

where $\left|W_1^{-1}(j\omega)\right|$ is the desired disturbance attenuation factor. Allowing $\left|W_1(j\omega)\right|$ to depend on frequency ω enables you to specify a different attenuation factor for each frequency ω.

The singular value Bode plots of *R(s)* and of *T(s)* are used to measure the stability margins of multivariable feedback designs in the face of additive plant perturbations Δ_A and multiplicative plant perturbations Δ_M, respectively. See Figure 1-17, Additive/Multiplicative Uncertainty.

Let us consider how the singular value Bode plot of complementary sensitivity *T(s)* determines the stability margin for multiplicative perturbations Δ_M. The

multiplicative stability margin is, by definition, the "size" of the smallest stable *(s)* which destabilizes the system in Figure 1-17, Additive/Multiplicative Uncertainty with $\Delta_A = 0$.

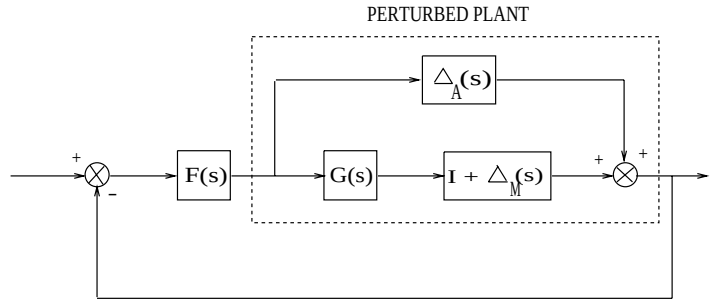


Figure 1-17: Additive/Multiplicative Uncertainty

Taking $\bar{\sigma}(\Delta_M(j\omega))$ to be the definition of the "size" of $\Delta_M(j\omega)$, you have the following stability theorem:

Robustness Theorem 1: *Suppose the system in Figure 1-17, Additive/Multiplicative Uncertainty is stable with both* Δ_A *and* Δ_M *being zero. Let* $\Delta_A = 0$. *Then the size of the smallest stable* $\Delta_M(s)$ *for which the system becomes unstable is*

$$\bar{\sigma}(\Delta_M(j\omega)) = \frac{1}{\bar{\sigma}(T(j\omega))} \qquad \textbf{(3-1)}$$

The smaller is $\bar{\sigma}(T(j\omega))$, the greater will be the size of the smallest destabilizing multiplicative perturbation, and hence the greater will be the stability margin.

A similar result is available for relating the stability margin in the face of additive plant perturbations $\Delta_A(s)$ to $R(s)$. Let us take $\bar{\sigma}(\Delta_A(j\omega))$ to be our definition of the "size" of $(\Delta_A(j\omega))$ at frequency ω. Then, you have the following stability theorem.

Robustness Theorem 2: *Suppose the system in Figure 1-17, Additive/Multiplicative Uncertainty is stable when* Δ_A *and* Δ_M *are both zero. Let* $\Delta_A = 0$. *Then the size of the smallest stable* $\Delta_A(s)$ *for which the system becomes unstable is*

$$\bar{\sigma}(\Delta_A(j\omega)) = \frac{1}{\bar{\sigma}(R(j\omega))}$$

As a consequence of Theorems 1 and 2, it is common to specify the stability margins of control systems via singular value inequalities such as

$$\bar{\sigma}(R\{j\omega\}) \le \left|W_2^{-1}(j\omega)\right| \quad \textbf{(1-1)}$$

$$\bar{\sigma}(T\{j\omega\}) \le \left|W_3^{-1}(j\omega)\right| \quad \textbf{(1-2)}$$

where $\left|W_2(j\omega)\right|$ and $\left|W_3\ (j\omega)\right|$ are the respective sizes of the largest anticipated additive and multiplicative plant perturbations.

It is common practice to lump the effects of all plant uncertainty into a single fictitious multiplicative perturbation Δ_M, so that the control design requirements may be written

$$\frac{1}{\sigma_i(S(j\omega))} \ge \left|W_1(j\omega)\right|; \quad \bar{\sigma}_i(T[j\omega]) \le \left|W_3^{-1}(j\omega)\right|$$

as shown in Figure 1-18, Singular Value Specifications on S and T.

It is interesting to note that in the upper half of Figure 1-18, Singular Value Specifications on S and T (above the zero db line)

$$\underline{\sigma}(L(j\omega)) \approx \frac{1}{\bar{\sigma}(S(j\omega))}$$

while in the lower half of Figure 1-18, Singular Value Specifications on S and T below the zero db line

$$\underline{\sigma}(L(j\omega)) \approx \bar{\sigma}(T(j\omega))$$

This results from the fact that

$$S(s) \stackrel{def}{=} (I + L(s))^{-1} \approx L(s)^{-1}, \quad \text{if } \underline{\sigma}(L(s)) \gg 1$$

$$T(s) \stackrel{def}{=} L(s)(I + L(s))^{-1} \approx L(s), \quad \text{if } \bar{\sigma}(L(s)) \ll 1$$

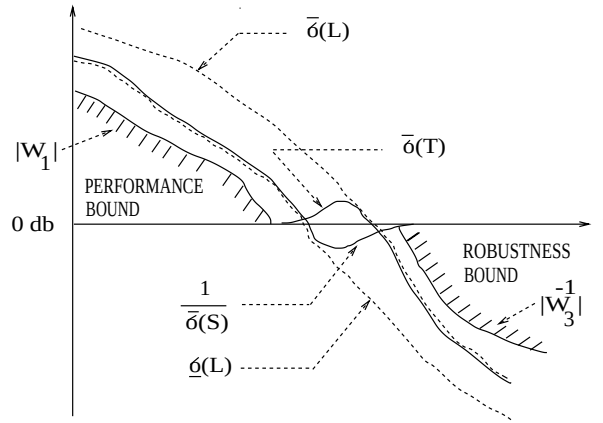


Figure 1-18: Singular Value Specifications on S and T

Thus, it is not uncommon to see specifications on disturbance attenuation and multiplicative stability margin expressed directly in terms of forbidden regions for the Bode plots of $\sigma_i(L(j\omega))$ as "singular value loop shaping" requirements. See Figure 1-18, Singular Value Specifications on S and T.

An important point to note in choosing design specifications W_1 and W_3 is that the 0 db crossover frequency the Bode plot of W_1must be sufficiently below the 0 db crossover frequency of W_3^{-1} or the performance requirements (3-1) and (1-2) will not be achievable; more precisely, we require

$$\bar{\sigma}(W_1^{-1}(j\omega)) + \bar{\sigma}(W_3^{-1}(j\omega)) > 1 \quad \forall\omega \qquad \textbf{(1-3)}$$

Guaranteed Gain/Phase Margins in MIMO Systems

For those who are more comfortable with classical single-loop concepts, there are the important connections between the multiplicative stability margins

predicted by and those predicted by $\bar{\sigma}(T)$ classical M-circles, as found on the Nichols chart. Indeed in the single-input-single-output case

$$\bar{\sigma}(T(j\omega)) = \left| \frac{L(j\omega)}{1 + L(j\omega)} \right|$$

which is precisely the quantity you obtain from Nichols chart M-circles. Thus, $\|T\|_\infty$ is a multiloop generalization of the closed-loop resonant peak magnitude which, as classical control experts will recognize, is closely related to the damping ratio of the dominant closed-loop poles. Also, it turns out that you may relate $\|T\|_\infty, \|S\|_\infty$ to the classical gain margin G_M and phase margin θ_M in each feedback loop of the multivariable feedback system of Figure 1-16, Block Diagram of the Multivariable Feedback Control System via the formulae [22]:

1 $G_M \geq 1 + \dfrac{1}{\|T\|_\infty}$

2 $G_M \geq 1 + \dfrac{1}{1 - 1/\|S\|_\infty}$

3 $\theta_M \geq 2\sin^{-1}\left(\dfrac{1}{2\|T\|_\infty}\right)$

4 $\theta_M \geq 2\sin^{-1}\left(\dfrac{1}{2\|S\|_\infty}\right)$

These formula are valid provided $\|S\|_\infty$ and $\|T\|_\infty$ are larger than one, as is normally the case. The margins apply even when the gain perturbations or phase perturbations occur simultaneously in several feedback channels.

The infinity norms of S and T also yield gain reduction tolerances. The *gain reduction tolerance* g_m is defined to be the minimal amount by which the gains in each loop would have to be *decreased* in order to destabilize the system. Upper bounds on g_m are:

5 $g_m \leq 1 - \dfrac{1}{\|T\|_\infty}$

6 $g_m \leq \dfrac{1}{1 + 1/\|S\|_\infty}$

For the Characteristic Gain Loci Example, you can compute the *guaranteed stability margins* using the formulae given above

$$Guaranteed\ \ GM = 1 \pm \frac{1}{\|T\|_\infty} = 0.94\ \ to\ \ 1.062$$

$$Guaranteed\ \ PM = \pm 2 \sin^{-1}\ \frac{1}{2\|T\|_\infty} = \pm 3.52\ \ deg$$

which clearly predict the poor robustness of the system. These guaranteed stability margins provide a tolerance such that you can vary both gain and phase simultaneously in all the feedback loops.

Significance of the Mixed-Sensitivity Approach

The *Mixed-Sensitivity* approach of the robust control system design is a direct and effective way of achieving multivariable loop shaping, although it is a special case of the *canonical robust control problem* set-up described earlier.

In the mixed-sensitivity problem formulation, *nominal* disturbance attenuation specifications and stability margin specifications equations (3-1) and (1-2) are combined into a single infinity norm specification of the form

$$\left\| T_{y1u1} \right\|_\infty \le 1 \qquad \textbf{(1-4)}$$

where

$$T_{y_1 u_1} \stackrel{def}{=} \begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix} \qquad \textbf{(1-5)}$$

The left side of equation (1-4) is the *mixed-sensitivity cost function*, so called because it penalizes both sensitivity $S(s)$ and complementary sensitivity $T(s)$.

Note that if you augment the plant $G(s)$ with the weights $W_1(s)$ and $W_3(s)$ as shown in Figure 1-19, Weighted Mixed Sensitivity Problem, then wrap the feedback $F(s)$ from output channel y_2 back to input channel u_2, the resulting nominal (i.e., $\Delta_M = 0$) closed-loop transfer function is precisely the $T_{y1u1}(s)$ given by (1-5). The Robust Control Toolbox functions `augtf` and `augss` perform this plant augmentation.

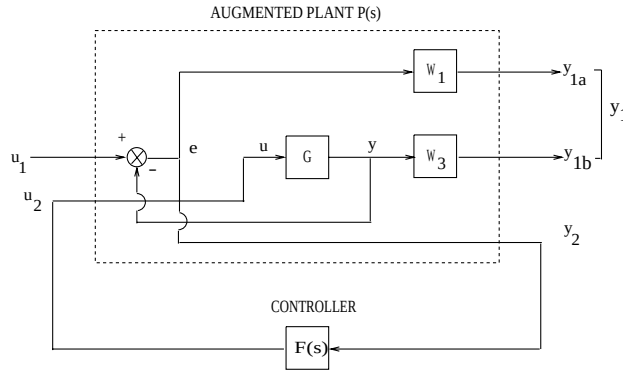


Figure 1-19: Weighted Mixed Sensitivity Problem

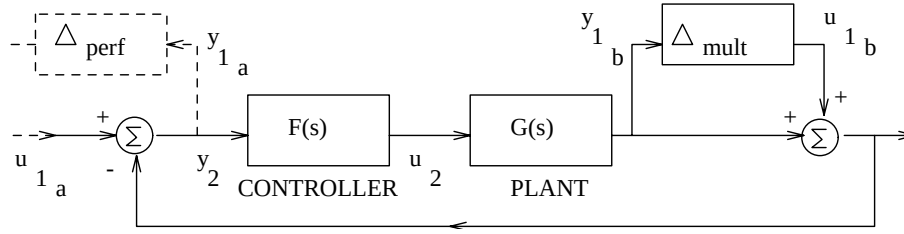


Figure 1-20: Robust Sensitivity Problem

The mixed sensitivity cost function has the attractive property that, it provides a much simplified, and nearly equivalent, alternative to the canonical robust control problem for the case of the robust sensitivity problem (cf. Figure 1-20, Robust Sensitivity Problem). It turns out that if (1-4) is strengthened very slightly to

$$\|T_{y1u1}\|_\infty \le 1/\sqrt{2}$$

then robust sensitivity performance can be guaranteed; that is,

$\frac{1}{\sigma_i(S(j\omega))} \ge |W_1(j\omega)|$ for every multiplicative uncertainty Δ_M satisfying

$$\bar{\sigma}(\Delta_M(j\omega)) \le |W_3(j\omega)|$$

This is because in this case the T_{y1u1} associated with the corresponding canonical robust control problem (cf. Figure 1-2, Canonical Robust Control Problem) becomes simply

$$T_{y1u1} = \begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix} [I, \ -I]$$

For any $S(s)$ and $T(s)$, it may be shown that [3]

$$\left\| \begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix} \right\|_\infty \leq \mu \left(\begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix} [I, \ -I] \right) \leq \sqrt{2} \left\| \begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix} \right\|_\infty$$

This relationship guarantees that $\mathbf{H}^\infty$ *synthesis for the mixed sensitivity setup depicted in Figure 1-19, Weighted Mixed Sensitivity Problem is — to within 3 db* $(\sqrt{2})$ *— the same as the* K_M *synthesis (or* μ *synthesis) for the system in Figure 1-20, Robust Sensitivity Problem.* This is a major simplification since, in general, $\mathbf{H}^\infty$ synthesis is much easier and more transparent to the designer than K_M synthesis. The numerical computations for $\mathbf{H}^\infty$ synthesis are much more straightforward too, so that the design computations can be carried out significantly more quickly than with synthesis K_M.

This relationship tremendously simplifies the robust control (K_M or μ) synthesis problem. Instead, it replaces it with an easy-to-solve, easy-to-understand m*ixed sensitivity synthesis problem*. By achieving a mixed sensitivity design with the transfer function matrix

$$T_{y_1 u_1} = \begin{bmatrix} W_1 S \\ W_3 T \end{bmatrix}$$

having its L^∞-norm less than $\frac{1}{\sqrt{2}}$, you have achieved $K_m > 1$, i.e., the "real" robust performance.

μ Synthesis

The objective of μ *synthesis* is to find a stabilizing controller *F(s)* and a diagonal scaling matrix *D(s)* such that

$$\left\| DT_{y1u1}D^{-1} \right\|_\infty < 1$$

where T_{y1u1} is as shown in Figure 1-2, Canonical Robust Control Problem. Since $\left\| T_{y1u1} \right\|_\infty$ as an upper bound of the structured singular value, $\mu \stackrel{def}{=} \frac{1}{K_{ss}}$, having the infinity norm of T_{y1u1} less than one is sufficient to ensure robust stability and/or robust performance.

A conceptual procedure for μ synthesis described in [33, 15] goes as follows (see Figure 1-21, μ Synthesis D-K Iteration):

1. Let *D(s)* = *I* and use the $\mathbf{H}^\infty$ control method (`hinf.m`) to find a *F(s)* which minimizes the cost function $\left| DT_{y1u1}D^{-1} \right|_\infty$.
2. Fix *F(s)*, then use `ssv.m` to find a cost-minimizing diagonal matrix *D(s)*.
3. Using a curve fitting method (`fitd.m`), find a low order rational approximation to the optimal *D(s)* obtained in step 2.
4. If the cost function is less than one, stop; otherwise, go to step 1.

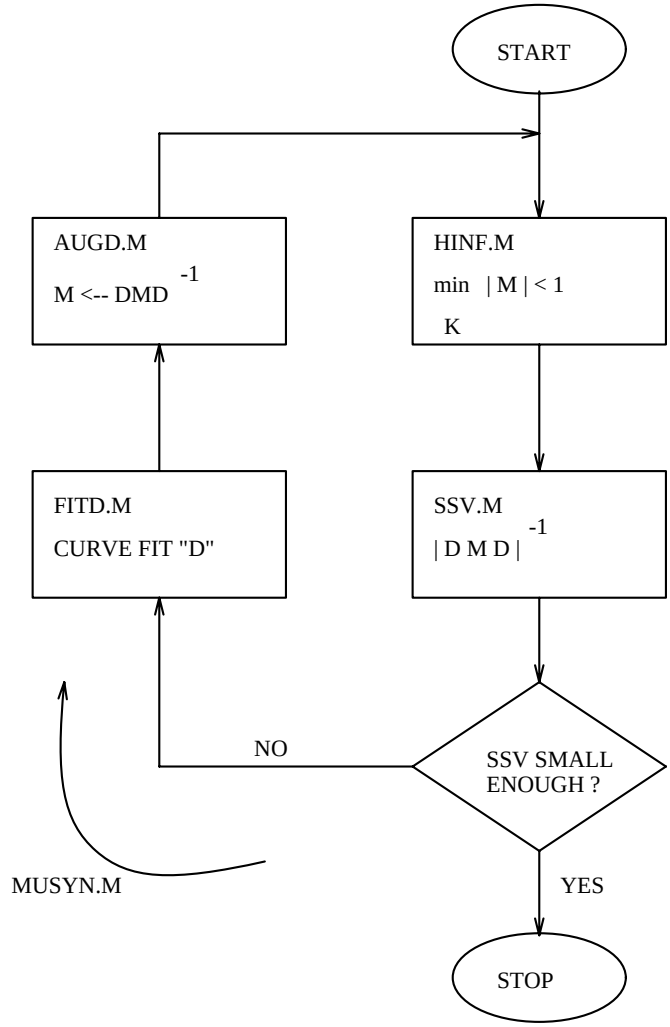


Figure 1-21: μ Synthesis D-K Iteration

This method essentially integrates two optimization problems and solves them by alternately fixing either the variable *F(s)* or the variable *D(s)*, and minimizing over the other variable until the μ bound (i.e., cost function) $\left\| DT_{y1u1}(F)d^{-1} \right\|_\infty$ is sufficiently small. For a fixed *D*, it becomes the standard

$\mathbf{H}^{\infty}$ synthesis problem solved by `hinfopt`. And, for a fixed *F(s)*, it becomes the problem suboptimally addressed by `ssv`, `psv`, `perron`, and `muopt` of finding a stable and minimum phase *D(s)* that minimizes the cost function at each frequency. This method has the potential of solving the overall Robust Control Problem.

Following is the input for a simple synthesis problem.

```
PLANT DATA:

  a=2;          b1=[.1,-1];           b2=-1;
  c1=[1;.01];   d11=[.1,.2;.01,.01]; d 12=[1; 0];
  c2=1;         d21=[0,1];            d22=3;


  tss=mksys(a,b1,b2,c1,c2,d11,d12,d21,d22,'tss');
  w=logspace(-2,1);
  H-INFINITY OPTIMAL DESIGN:
  [gam0,sscp0,sscl0]=hinfopt(tss);
  [mu0,logd0]=ssv(sscl0,w);

  MU SYNTHESIS ITERATION NO. 1 (CONSTANT D):
  [ssd1,logd1]=fitd(logd0,w);
  [gam1,sscp1,sscl1]=hinfopt(augd(tss,ssd1));
  [mu1,deltalogd]=ssv(sscl1,w);

  MU SYNTHESIS ITERATION NO. 2 (FIRST ORDER D):
  [ssd2,logd2]=fitd(logd1+deltalogd,w,1);
  [gam2,sscp2,sscl2]=hinfopt(augd(tss,ssd2));

  DISPLAY OPTIMAL SIGMA AND SSV PLOTS:
  loglog(w,max(sigma(sscl2,w))/gam2,w,..
  ssv(sscl2,w)/gam2);
```

The foregoing example illustrates the basic μ synthesis iteration. In practice, you will generally prefer to use a constant, zeroth order diagonal scaling matrix D*(s)* because it leads to a much lower order control law. It may also be necessary to experiment with the frequency range ω, adjusting it so that it coincides roughly with the frequency range over which the value of `mu` returned by `ssv` is unacceptably large.

A more detailed μ synthesis example is provided in the “Case Studies” section.

Bilinear Transform and Robust Control Synthesis

A simple bilinear transform `bilin.m` has been found to be extremely useful when used with robust control synthesis techniques. It can:

1. Remove the ill-conditioning inherent in some augment plants.
2. Provide direct control of the location of dominant closed-loop poles.

In the $\mathbf{H}^{\infty}$ mixed-sensitivity problem formulation, if the augmented plant has $j\omega$-axis poles or zeros, the $\mathbf{H}^{\infty}$ controller, if it could be reliably computed, would have marginally stable *closed-loop* poles at the corresponding $j\omega$-axis locations [35]. A similar problem arises in more general situations in which either $P_{12}(s)$ or $P_{21}(s)$ have $j\omega$-axis zeros, including zeros at ∞ that occur when you have a rank deficient matrix D_{12} or *D21* in the state-space realization of *P(s)*. In practice, the foregoing situations lead to singularities in the equations which determine the state-space realization of the $\mathbf{H}^{\infty}$ control law. The Robust Control Toolbox routines `hinf` and `dhinf` produce warning messages in these situations.

Using the bilinear transform changes this situation immediately. Problems with $j\omega$-axis poles and zeros or with rank-deficiency D_{12} or D_{21} can be removed by the transformation. Then, after the computation, you can transform back the controller using the inverse transform. The resultant control law will then be a *suboptimal* solution to the original $\mathbf{H}^{\infty}$ control problem. Combining thebilinear transform and $\mathbf{H}^{\infty}$ synthesis is also a direct and powerful way of controlling closed-loop system performance (in terms of rise time, damping ratio, settling time, etc.) [5, 6]. This will become clear after we discuss some details of the bilinear transform.

The bilinear transform can be formulated as a $j\omega$-axis pole shifting transformation

$$s = \frac{\tilde{s} + p1}{\frac{\tilde{s}}{p2} + 1} \qquad \textbf{(1-6)}$$

where the numbers $-p_1$ and $-p_2$ are the end-points of the diameter of a circle in the left s-plane (see Figure 1-22, Bilinear Transform for Axis Pole Shifting) that is mapped by (1-6) onto the $j\omega$-axis in the $\tilde{s}$-plane. The inverse of such a bilinear transform is

$$\tilde{s} = \frac{-s + p1}{\frac{s}{p2} - 1} \qquad \textbf{(1-7)}$$

Figure 1-22, Bilinear Transform for Axis Pole Shifting also shows how the various regions designated “A”, “B” and “C” in $s/\tilde{s}$ -planes are transformed by the mappings (1-6) and (1-7):

- The boundary of the s-plane circle Γ is mapped onto $j\omega$-axis in the $\tilde{s}$-plane.
- The s-plane $j\omega$-axis is mapped onto a circle Γ in the right $\tilde{s}$-plane, which is an exactly mirror image of the s-plane circle Γ.
- Areas “A”, “B” and “C” are mapped to their $\tilde{s}$-plane counterparts.

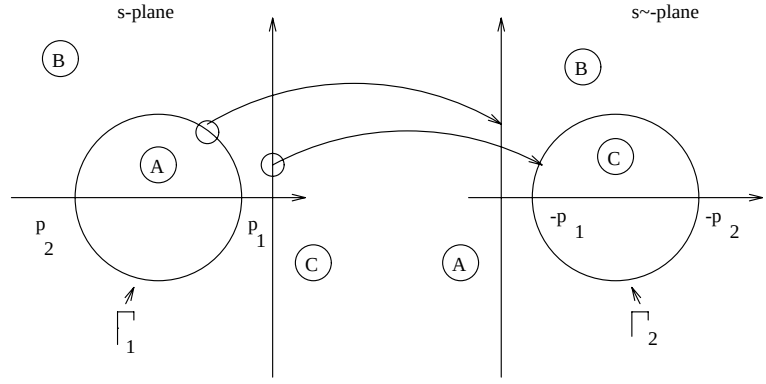


Figure 1-22: Bilinear Transform for Axis Pole Shifting

Both forward and inverse multivariable bilinear transforms are special cases of the transform $s = \frac{\alpha z + \delta}{\gamma z + \beta}$ and can be realized by the state-space formula [3]

$$\left[\begin{array}{c|c} A_b & B_b \\ \hline C_b & D_b \end{array}\right] = \left[\begin{array}{c|c} (\beta A - \delta I)(\alpha I - \gamma A)^{-1} & (\alpha\beta - \gamma\delta)(\alpha I - \gamma A)^{-1} B \\ \hline C(\alpha I - \gamma A)^{-1} & D + \gamma C(\alpha I - \gamma A)^{-1} B \end{array}\right]$$

Other important transforms that map continuous transfer functions to the discrete domain such as Tustin, prewarped Tustin, backward/forward rectangular transforms, etc. can also be handled with this state-space formula. They are discussed in the Reference section under `bilin`.

Now, if a plant has poles on the $j\omega$-axis in s-plane, the bilinear transform will map these poles onto a circle Γ in $\tilde{s}$-plane centered at

$$\frac{-(p_1+p_2)}{2}$$

Property 4 of $\mathbf{H}^\infty$ controllers will ensure the closed-loop poles are placed inside the circle "A" in -plane of Figure 1-22, Bilinear Transform for Axis Pole Shifting at positions which are the "shifted RHP" mirror images of the open-loop poles outside the region "A" in $\tilde{s}$-plane. Therefore, the parameter "p_1" in bilinear transform turns out to be the key parameter in placing the dominant closed-loop poles at the desired locations in s-plane, thereby satisfying the performance specification.

A simple design procedure can be formulated as follows:

1. Pull out the uncertainty blocks from the system block diagram to formulate an $\mathbf{H}^\infty$ robust control problem.
2. Map the plant from s-plane to $\tilde{s}$-plane via the bilinear pole shifting transform (1-6).
3. Compute the $\mathbf{H}^\infty$ optimal controller for the transformed plant $\tilde{\Gamma}\,\tilde{s}$ (i.e., solve $\min_{\tilde{F}(\tilde{s})} \left\| \tilde{T}(j\tilde{\omega}) \right\|_\infty \le 1$)
4. Map the controller $\tilde{F}(\tilde{s})$ back to s-plane via the inverse bilinear pole shifting transform (1-7).
5. Go back to step 1 and iterate the parameter p_1 of the bilinear transform until the design specifications are met.

A benchmark problem proposed by Wie and Bernstein [49] has been solved successfully via this method. See the *Case Studies* section for details.

Robustness with Mixed Real and Complex Uncertainties

One of the drawbacks associated with the complex K_M (or μ) analysis/synthesis is that it treats each uncertainty as being bounded by a complex disc, which can lead to conservativeness if the parameter is real. A generalized Popov multiplier theory overcomes this conservativeness and provides a precise analysis and synthesis treatment of the "robust control problem". In this

section, we will introduce this concept on the design and analysis of systems with mixed real and complex uncertainties.

Real K_M Analysis

In robust analysis, the idea of diagonal scaling introduced in the earlier section is to find a diagonal D matrix to scale the "size" of the transfer functions seen by the (block) diagonal complex structured uncertainties such that a much less conservative measure of the multivariable stability margin of the system can be predicted as compared to the worst-case tool singular values that assumes a complete complex uncertainty matrix with no structure at all. With the real uncertainty embedded inside the diagonal structure, one can build a "shift-and-enlarge" process on top of the existing diagonal scaling structure (as shown in Figure 1-23, Shift-and-Enlarge Process of Real Uncertainty Analysis) to capture the real parametric uncertainty. The idea of "shift-and-enlarge" is the following: If the real uncertainty is bounded inside a complex unit circle, evidently the unit circle is too conservative for a parameter that only varies on the real line. Shifting the circle along the imaginary axis and enlarging the unit circle radius to $(1+C^2)^{1/2}$ still covers the same real parametric variation on the real line, but the new multivariable stability margin seen by this "larger" uncertainty circle can be much smaller. By using convex nonlinear programming methods, the shift may be optimized to obtain less conservative bounds on the real structured singular value.

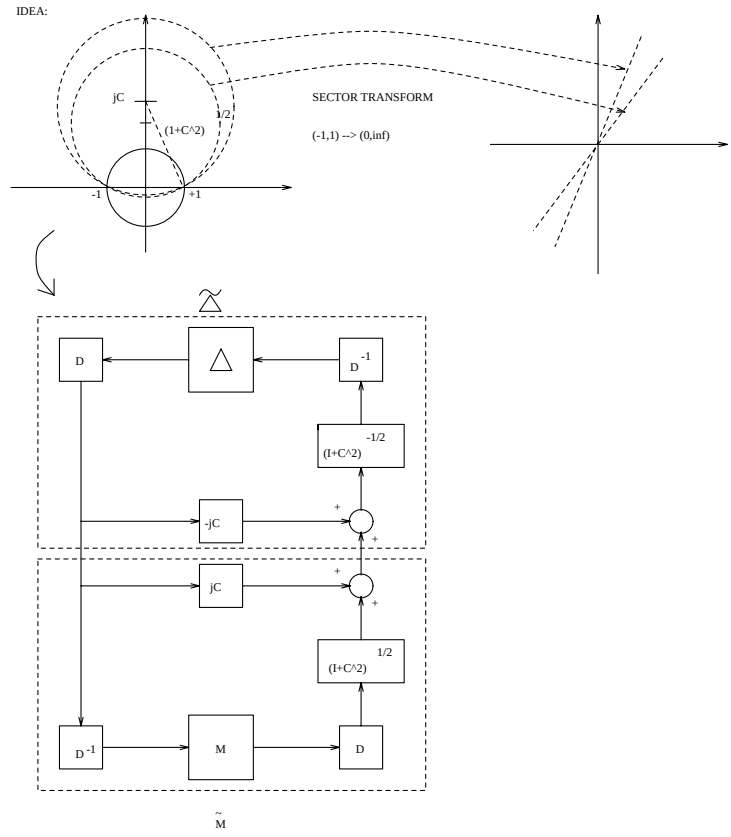


Figure 1-23: Shift-and-Enlarge Process of Real Uncertainty Analysis

Mathematically, we have the following optimization problem to solve

$$\inf_{D \in \mathrm{D}} \quad \inf_{C \in \mathrm{C}} \quad \bar{\sigma}(jC + (I + C^2)^{1/2} D T D^{-1})$$

This optimization problem can be solved via a special bilinear transform — the sector transform (`sectf.m`) that maps the circles shown in Figure 1-23, Shift-and-Enlarge Process of Real Uncertainty Analysis to halfplanes that are bounded on left by lines passing through the origin in the new domain (also, $\gamma T(s) \to T(s)$). The problem is then shifted to the following (see Figure 1-24, Real Analysis):

Find the optimal multiplier to maximize such that

$$Re(M(j\omega)\tilde{T}(j\omega)) > 0, \forall\omega, \quad \text{(i.e., } M \subset \text{sector}(0,\infty);\ \text{positive real)}$$

where the multipliers are chosen to be in the class M defined to be the set of all diagonal transfer function matrices

$$M(s) = \text{diag}(m_1(s), m_2(s), \ldots, m_n(s))$$

satisfying $\frac{1}{2}(M(j\omega) + M^*(j\omega)) \geq 0,\ \forall\omega$ *and* $m_i(j\omega) = r_i + jp_i$ *or* $\frac{-\pi}{2} < m_i < \frac{\pi}{2}$.

If such a multiplier exists, the value γ is a lower bound on the size of the smallest destabilizing real uncertainties.

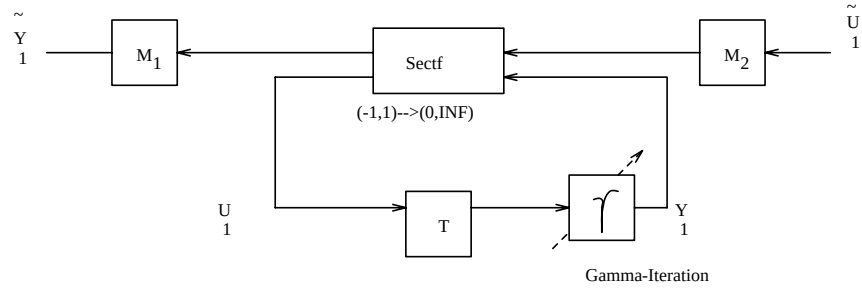


Figure 1-24: Real Analysis

Now, without loss of generality, we may restrict attention to polynomial multipliers; such multipliers $M(s) \in \mathrm{M}$; may be convexly parametrized as

$$M(s) = X(s^2) + sY(s^2)$$

where *X, Y* are diagonal polynomial matrices and $X(\omega^2) + X^*(\omega^2) \geq 0, \forall\omega$.

We call the multipliers in the class *M generalized Popov multipliers*. They have the following special properties.

Properties of the Generalized Popov Multiplier

1 Positive Real: $\text{hermitian}(M(s)) \overset{\text{de}}{=} \frac{1}{2}(M(s) + M^*(s)) \geq 0, \quad \forall s = j\omega$.

2 For real Δ, $M\Delta \subset (0,\infty)$ if and only if $\Delta \subset (0,\infty)$.

3 $M(s) = M_2(-s)^{-T} M_1(s)$, where $M_2^{-T}(-s)$, $M_1(s)$ and $M_1^*(s)$ are stable and minimum phase (i.e., no poles or zeros in the right-half complex s-plane).

Note that the classical Popov multiplier $1 + qs = 0$ is in M. It can be shown that the diagonal scaling $D(s)$ used in evaluating the *complex* structured singular value is equivalent to using *real* diagonal multipliers $M(j\omega) = M^*(j\omega) = D^*(j\omega)D(j\omega)$; that is, $M_1(j\omega) = M_2(j\omega) = D(j\omega)$. Thus, the case of mixed real and complex uncertainty is handled in the multiplier optimization framework by simply imposing the additional restriction that the multipliers corresponding to each complex uncertainty be real. The functions `muopt` and `ssv` invoke a simple nonlinear programming routine to compute the optimal multiplier. The optimization is both smooth and convex, so global convergence of the algorithm is assured.

To find such a multiplier, one needs to invoke a nonlinear programming technique, which is beyond this tutorial (see `muopt` in the reference section for more details).

Real K_M Synthesis

After understanding the analysis approach, one can formulate and solve the real K_M synthesis problem as follows (see Figure 1-25, Real KM Synthesis):

Find the greatest real number γ such that for some optimal generalized Popov multiplier M(s) the infinity norm of the cost function $T_{\tilde{z}\,\tilde{u}_1}$ is less than or equal to one, i.e.

$$\max_{(M_2^*)^{-1} M_1 \in M, \gamma} \left\| T_{\tilde{z}\,\tilde{u}_1} \right\|_\infty \le 1$$

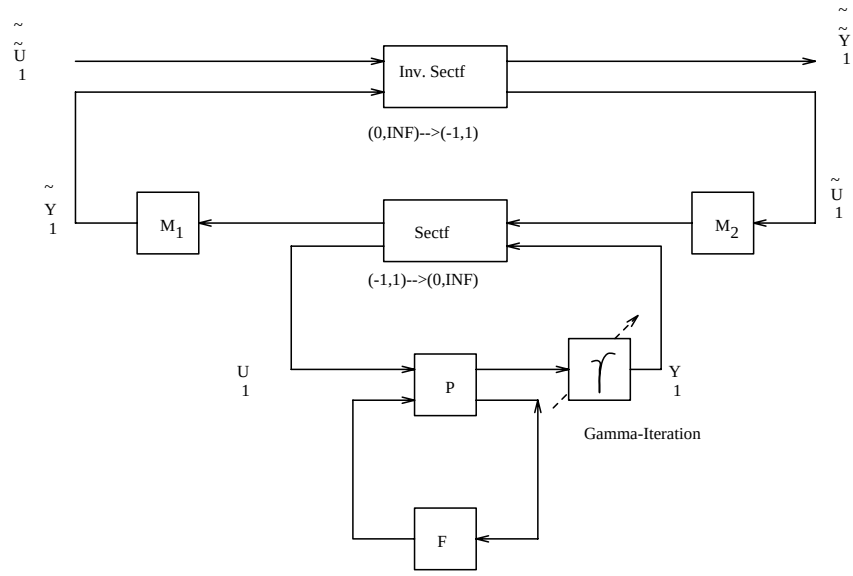


Figure 1-25: Real K_M Synthesis

A possible iterative procedure for solving the real K_M control synthesis problem is listed as follows (see Figure 1-25, Real KM Synthesis):

1. Solve the conventional $\mathbf{H}^{\infty}$ optimal control problem of finding the maximal γ such that $\min_F(\|T_{y_1u_1}\|_{\infty} \le 1)$.

2. Compute SSV to get the regular diagonal scaling $D(s)$. Initialize $M(s) = D^T(-s)D(s)$.

3. Apply the `sectf` function to transform the original sector(–1, 1) to sector (0, ∞).

4. Solve the convex optimization problem of computing an improved $X(s^2)$ and $Y(s^2)$ to maximize the margin of $MT_{\tilde{y}_1\tilde{u}_1}$ positive definiteness of the Hermitian part of, thereby ensuring that the inverse transformed $T_{\tilde{\tilde{y}}_1\tilde{\tilde{u}}_1}$ is strictly less than one with some margin so that γ can be further increased.

5. Remove $F(s)$ from $T_{\tilde{\tilde{y}}_1\tilde{\tilde{u}}_1}$ and let $T_{y_1u_1} \leftarrow T_{\tilde{\tilde{y}}_1\tilde{\tilde{u}}_1}$ with the optimal multiplier absorbed in, go back to step 1 for a larger γ.

The foregoing real K_M synthesis procedure is not optimal and has not been automated in the Robust Control Toolbox. Of course, like complex μ synthesis, it is only a suboptimal procedure, and many possibly more effective variants of the procedure can be conceived. We only mention this to provide some indication of what is possible—and of what future developments are likely to emerge as the theory of robust control advances.

Case Studies

In this section, several design case studies using $\mathbf{H^2}$, $\mathbf{H^\infty}$ and μ synthesis techniques are illustrated. Most of the designs are also included in a demo program `rctdemo`.

Classical Loop-Shaping vs. H^∞ Synthesis

Given a plant *G(s)* which is 2nd order with damping 0.05 at 20 rad/sec, find a controller to meet frequency response Bode plot specification depicted by the solid line in Figure 1-26, Second Order Open-Loop Plant (ζ: 0.05, 0.5) and the L (s) Spec: Below 50 rad/sec we wish to have the compensated loop transfer function singular values above the solid line for good disturbance attenuation. Above 200 rad/sec we wish to have the singualar values below the solid line for good stability margin.

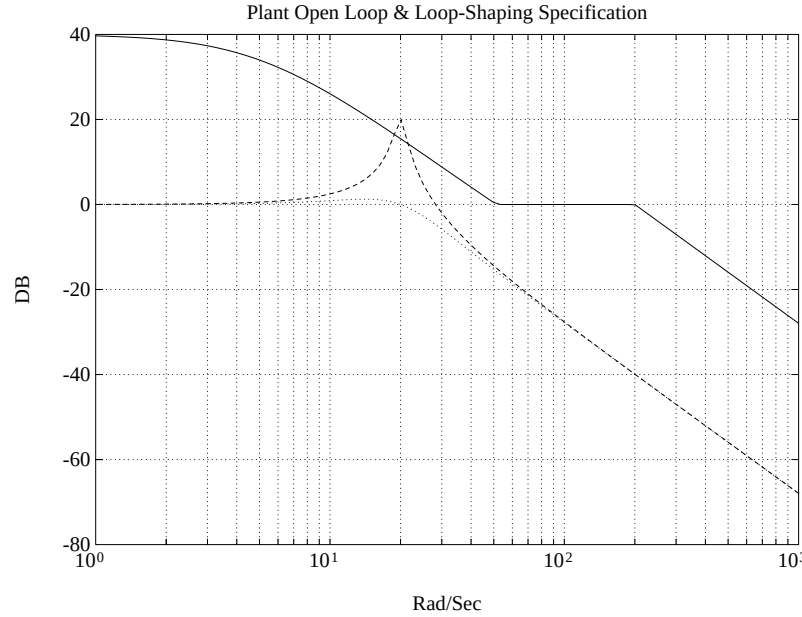


Figure 1-26: Second Order Open-Loop Plant (ζ: 0.05, 0.5) and the $L\ (s)$ Spec

A classical design might be decomposed into the following (see Figure 1-27, Classical Loop-Shaping Block Diagram):

1 Rate feedback to improve damping.

2 Design high frequency (phase margin, BW, roll-off, etc.).

3 Design low frequency (DC gain, disturbance rejection, etc.).

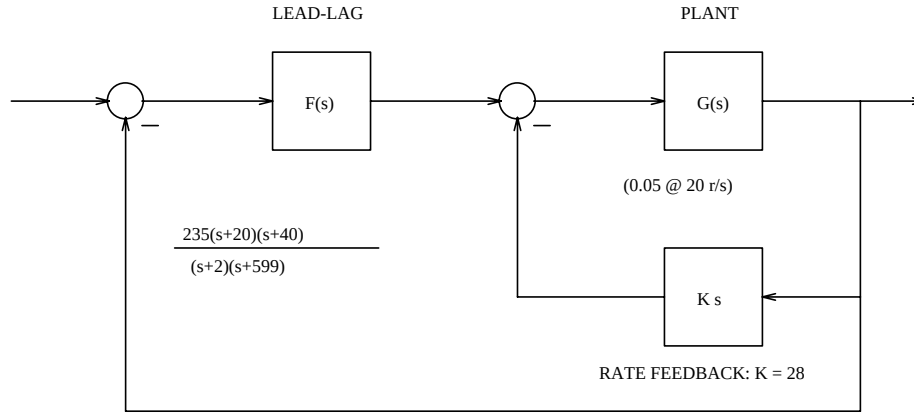


Figure 1-27: Classical Loop-Shaping Block Diagram

The classical result is shown in Figure 1-28, Classical Loop Shaping. Now, let's see how $\mathbf{H}^{\infty}$ approaches the problem.

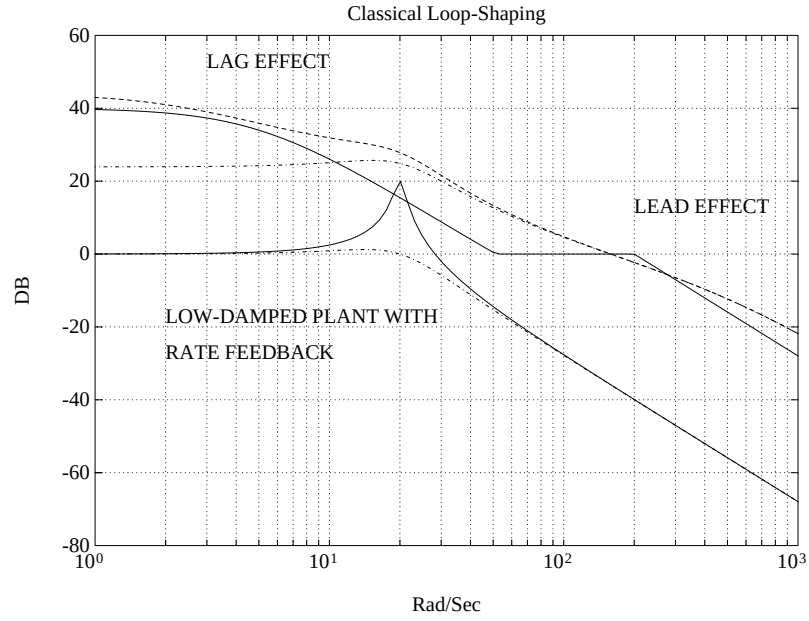


Figure 1-28: Classical Loop Shaping

H^∞ Problem Formulation

We solve the so-called $\mathbf{H}^\infty$ small-gain problem using the numerically robust descriptor 2-Riccati formulae of [42]. In our example, the frequency domain specification can be represented by the two weights

$$W_1^{-1} = \gamma^{-1}\frac{(0.2s+1)^2}{100(0.005s+1)^2}; \quad W_3^{-1} = \frac{40000}{s^2}$$

as shown (for γ=1) in Figure 1-29, H∞ Weighting Strategy for Second Order Problem.

The results are shown in Figure 1-30, H∞ Results for Second Order System for several different γ's. Clearly, in the limit (as γ increases to the optimal value γ = 3.16) the cost function becomes "all-pass". The parameter γ of $W_1$is the only parameter on which we iterate for design; the Robust Control Toolbox script-file `hinfopt.m` automates this iteration.

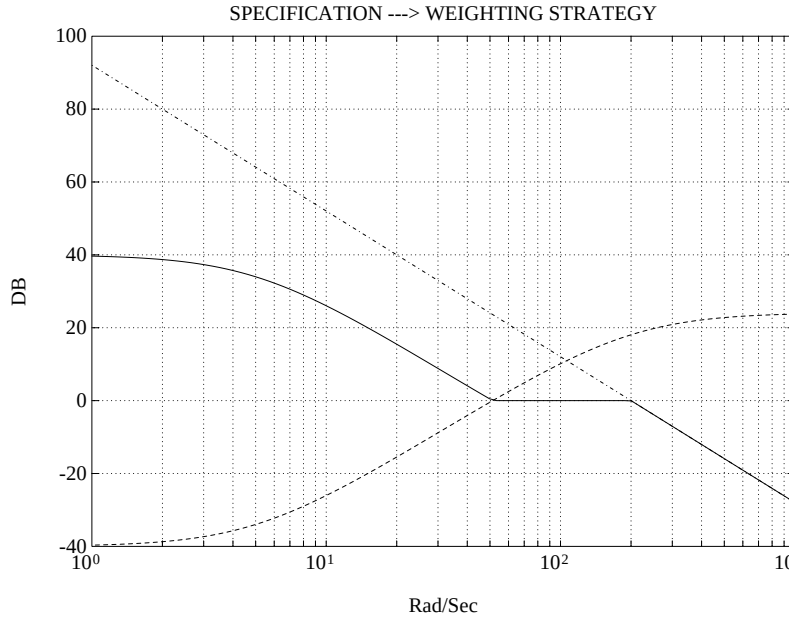


Figure 1-29: H^∞ Weighting Strategy for Second Order Problem

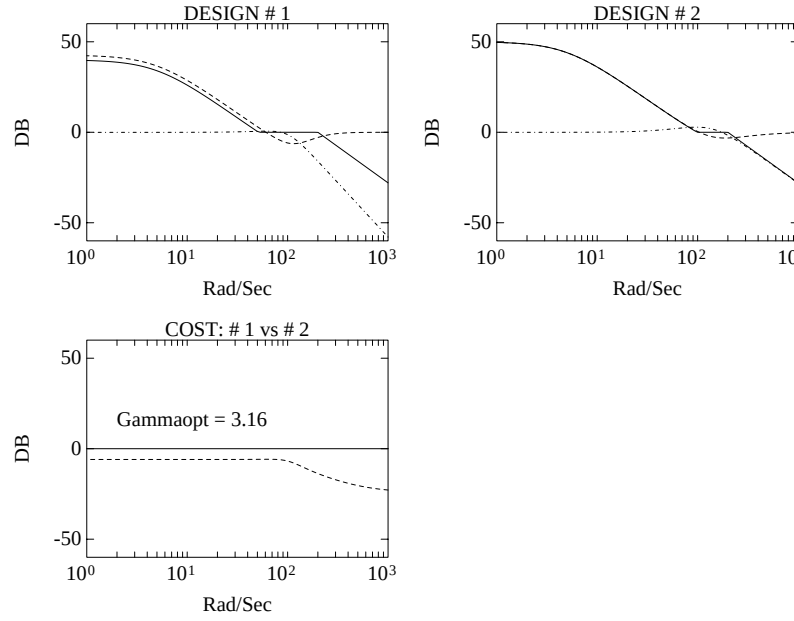


Figure 1-30: H^∞ Results for Second Order System

To do such an $\mathbf{H}^\infty$ control design with the Robust Control Toolbox is relatively simple:

```
nug = 400; dng = [1 2 400];
[ag,bg,cg,dg] = tf2ss(nug,dng);
ssg = mksys(ag,bg,cg,dg);
w1 = [2.5e-5 1.e-2 1;0.01*[4.e-2 4.e-1 1]];
w2 = [ ]; w3 = [1 0 0;0 0 40000];
TSS = augtf(ssg,w1,w2,w3);
[ssf,sscl] = hinf(TSS);
```

Table , shows the output which appears on the screen for a successful run of `hinf.m`. This corresponds to the computation of the $\mathbf{H}^{\infty}$ control law for $\gamma = 1$.

```
<< H-inf Optimal Control Synthesis >>

          Computing the 4-block H-inf optimal controller
        using the S-L-C loop-shifting/descriptor formulae

   Solving for the H-inf controller F(s) using U(s) = 0 (default)
    Solving Riccati equations and performing H-infinity
    existence tests:
       1.  Is D11 small enough?                        OK
       2.  Solving state-feedback (P) Riccati ...
            a.  No Hamiltonian jw-axis roots?          OK
            b.  A-B2*F stable (P >= 0)?                OK
       3.  Solving output-injection (S) Riccati ...
            a.  No Hamiltonian jw-axis roots?          OK
            b.  A-G*C2 stable (S >= 0)?                OK
       4.  max eig(P*S) < 1 ?                          OK
   ------------------------------------------------------
      all tests passed -- computing H-inf controller ...
                       DONE!!!
   ------------------------------------------------------
```

To compute that optimal $\mathbf{H}^{\infty}$ control law, you would replace `[ssf,sscl] = hinf(TSS);` with

```
[rhoopt,ssf,sscl] = hinfopt(TSS,1);
```

Table 1-3, shows the output which appears on the screen for the result of γ-iteration.

Table 1-3:

<< H-Infinity Optimal Control Synthesis >>

No	Gamma	D11<=1	P-Exist	P>=0	S-Exist	S>=0	lam(PS)<1	C.L.
1	1.0000e+000	OK	OK	OK	OK	OK	OK	STAB
2	2.0000e+000	OK	OK	OK	OK	OK	OK	STAB
3	4.0000e+000	OK	OK	FAIL	OK	OK	OK	UNST
4	3.0000e+000	OK	OK	OK	OK	OK	OK	STAB
5	3.5000e+000	OK	OK	FAIL	OK	OK	OK	UNST
6	3.2500e+000	OK	OK	FAIL	OK	OK	OK	UNST
7	3.1250e+000	OK	OK	OK	OK	OK	OK	STAB
8	3.1875e+000	OK	OK	FAIL	OK	OK	OK	UNST
9	3.1562e+000	OK	OK	OK	OK	OK	OK	STAB

Iteration no. 9 is your best answer under the tolerance: 0.0100 .

Fighter H^2 & H^∞ Design Example

Plant Description

The longitudinal dynamics of an aircraft trimmed at 25000 ft and 0.9 Mach are unstable and have two right half plane phugoid modes. The linear model has state-space realization $G(s) = C(Is - A)^{-1}B$ where

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] :=$$

$$\left[\begin{array}{cccccc|cc} -0.0226 & -36.6170 & -18.8970 & -32.0900 & 3.2509 & -0.7626 & 0 & 0 \\ 0.0001 & -1.8997 & 0.9831 & -0.0007 & -0.1708 & -0.0050 & 0 & 0 \\ 0.0123 & 11.7200 & -2.6316 & 0.0009 & -31.6040 & 22.3960 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -30.0000 & 0 & 30 & 0 \\ 0 & 0 & 0 & 0 & 0 & -30.0000 & 0 & 30 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right]$$

The control variables are elevon and canard actuators (δ_e and δ_c). The output variables are angle of attack (α) and attitude angle (θ).

This fighter model is the same as that in [29, 34] (see Figure 1-31, Aircraft Configuration and Vertical Plane Geometry).

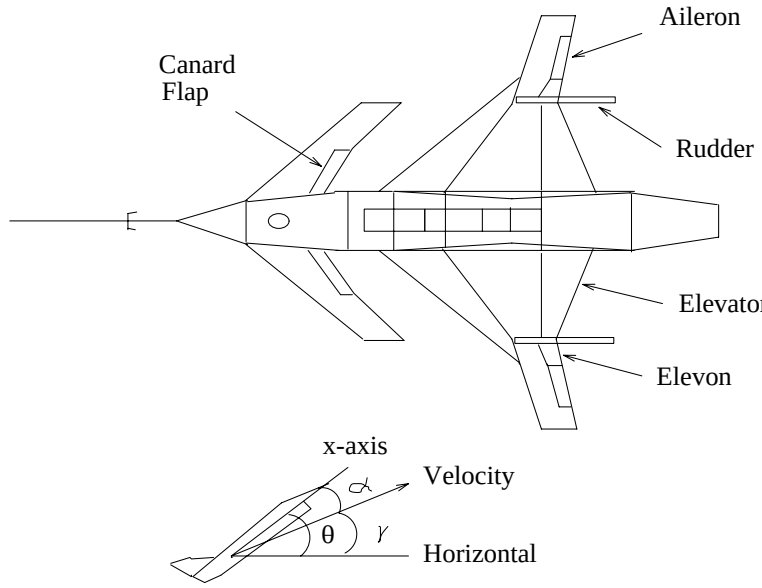


Figure 1-31: Aircraft Configuration and Vertical Plane Geometry

Design Specifications

The singular value design specifications are:

1 **Robustness Spec**.: –40 db/decade roll-off and at least –20 db at 100 rad/sec

2 **Performance Spec**.: Minimize the sensitivity function as much as possible.

Design Procedure

1 Augment the plant $G(s)$ with weighting functions $W_1(s)$ and $W_3(s)$ (design specifications) to form an "augmented plant" $P(s)$ as shown in Figure 1-19, Weighted Mixed Sensitivity Problem. Then, find a state-space realization of $P(s)$:

a Sensitivity function $S(s)$ specification:

$$W_1(s) := \gamma \begin{bmatrix} \frac{(s+100)}{(100s+1)} & 0 \\ 0 & \frac{(s+100)}{(100s+1)} \end{bmatrix}$$

γ is 1 for the first try, then can be increased or decreased accordingly later.

b Complementary sensitivity function $(I - S)$ specification:

$$W_3(s) = \begin{bmatrix} \frac{s^2}{1000} & 0 \\ 0 & \frac{s^2(rs+1)}{1000} \end{bmatrix}$$

where r = 0.5msec is selected such that both channels are penalized equally up to $1/r$ rad/sec. Note that because $W_3(s)$ is an improper transfer function (i.e., has more zeros than poles), it cannot be realized in state-space form. But, $W_3(s)G(s)$ is proper and, hence, $W_3(s)G(s)$ has a state-space realization. This ensures that the D_{12} matrix of the augmented plant $P(s)$ is full rank as required by `hinf` and `linf` [35]. (Another way to ensure a full rank would be to include a small third weight $W_2(s) = \varepsilon I$ – see the documentation for `augtf`). Because $W_3(s)$ has no state-space realization, the M-file `augtf` can be directly employed here [34].

2 Find a stabilizing controller $F(s)$ such that the infinity norm of transfer function $T_{y_1u_1}$ is minimized and is less than or equal to one (see Figure 1-19, Weighted Mixed Sensitivity Problem). We will start with $\mathbf{H}^2$ synthesis first then apply $\mathbf{H}^\infty$ to see the actual design limit (use `h2lqg`, `hinf`).

3 The $T_{y_1u_1}$ singular value Bode plot associated with each design will indicate how close the design is to the specifications. For most design problems, you need to iterate on the parameter γ in step 1 several times until a suitable design is obtained. The Robust Control Toolbox function `hinfopt` automates this iteration.

Result

The results are summarized in Figures 1-32 through 1-34.

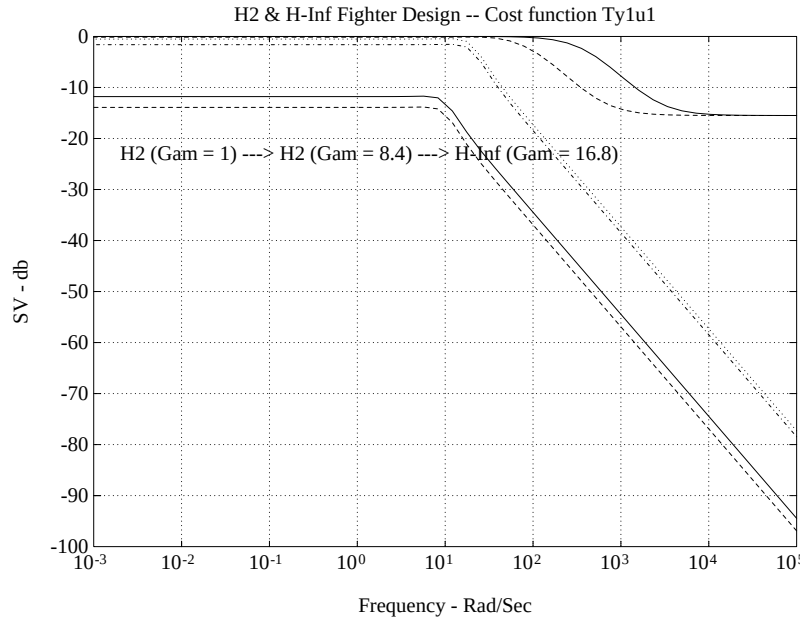


Figure 1-32: Cost Function $T_{y_1 u_1}(s)$

As shown in Figure 1-32, Cost Function, the cost function gets pushed to the "all-pass limit" (i.e., to 0 db), the sensitivity function S gets pushed down more and more, consequently the complementary sensitivity function T approaches to its associated weighting function W_3^{-1}.

The final $\mathbf{H}^\infty$ controller is stable and has 8 states which is the same as the augmented plant.

A complete flight control design case study for a supermaneuverable fighter flying the Herbst maneuver has been documented in [7]. A fixed 8-state $\mathbf{H}^\infty$ controller not only stabilizes the entire maneuver, but also maintains "robust performance" throughout in the presence of all the anticipated structured, unstructured certainties, nonlinearities.

Large Space Structure H^∞ Design Example

This design example is taken from [38, 43].

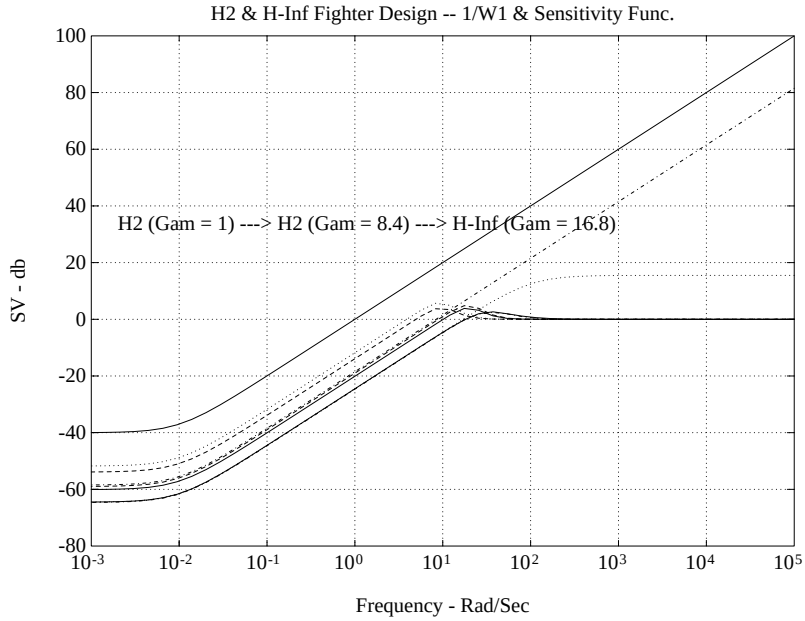


Figure 1-33: Sensitivity Function and W_1^{-1} Weighting

Plant Description

The large space structure (LSS) model was generated by the NASTRAN finite element program by TRW Space Technology Group. The model consists of 58 vibrational modes with frequencies ranging from 0.4 Hz to 477 Hz. The damping ratio is 0.3 for the first two modes and 0.002 for the rest of the modes. The structure is controlled by 18 actuators commanded by one central computer with a sampling frequency of 3000 Hz. The actuators are grouped in three locations: 6 are at the primary mirror, 6 at the secondary mirror, and 6 on structural members as shown in Figure 1-35, Large space structure.. Twelve disturbances are acting on the top and the bottom of the structure to simulate the real environmental vibration source. There are 20 sensors located at various locations in the structure. The most important sensors are the two *Line-Of-Sight* (LOS) sensors as indicated on the singular value Bode plot of the

open-loop plant shown in the Reference section. The remaining 18 sensors are collocated with the 18 control actuators.

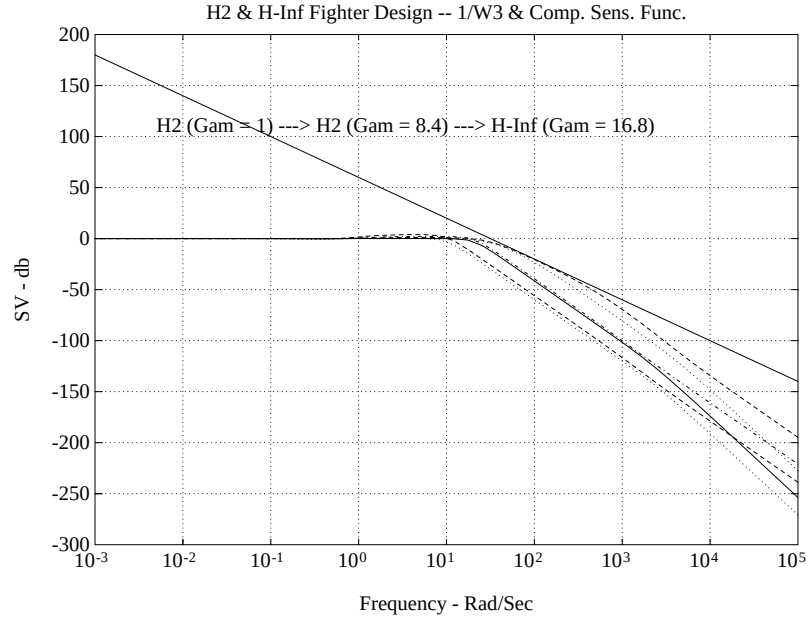


Figure 1-34: Complementary Sensitivity Function and W_3^{-1}

This leads to a state-space representation of the form

$$\dot{x} = Ax + Bu;\ y = Cx$$

where $A \in \Re^{116 \times 116}$, $B \in \Re^{116 \times 30}$, and $C \in \Re^{20 \times 116}$

Design Specifications

The LSS design specification requires the LOS error to be attenuated at least 100:1 at frequencies from 0 to 15 Hz after the feedback control loop is closed. Allowing for a 30 db per decade roll-off beyond 15 Hz places the control loop bandwidth of roughly 300 Hz (≈ 2000 rad/sec). In terms of inequalities to be satisfied by the open-loop singular value Bode plot, these specifications are as depicted in Figure 1-36, Singular Value Specifications (Weighting Functions).

For our $\mathbf{H}^{\infty}$ synthesis, these specification lead to the following weighting functions (note that they satisfy inequality (5)):

1 Robustness Spec.: –20 db/decade roll-off 2000 above rad/sec:

$$W_3(s) = \begin{bmatrix} \frac{s}{2000} & 0 \\ 0 & \frac{s}{2000} \end{bmatrix}$$

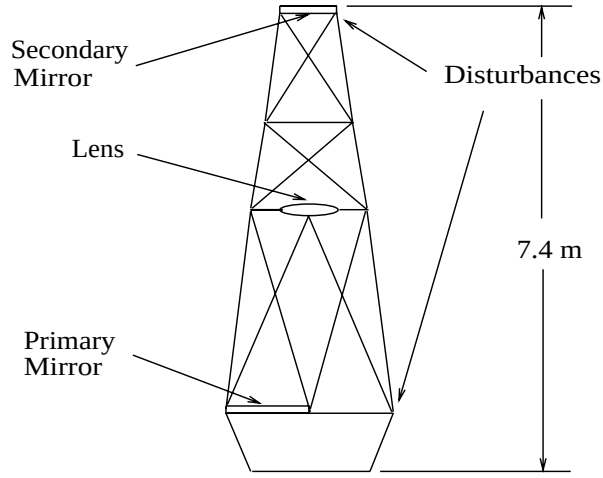


Figure 1-35: Large space structure.

2 Performance Spec.: Minimize the sensitivity function.

$$W_1(s) = \gamma \begin{bmatrix} \frac{\left(1+\frac{s}{5000}\right)^2}{0.01\left(1+\frac{s}{100}\right)^2} & 0 \\ 0 & \frac{\left(1+\frac{s}{5000}\right)^2}{0.01\left(1+\frac{s}{100}\right)^2} \end{bmatrix}$$

where in our design γ goes from one to 1.5. As with the fighter example, we use the trick of "absorbing" the improper weight W_3 into the strictly proper plant $G(s)$ so that the augmented plant has a non-zero D_{12} matrix, so that the resultant control law will be realizable in the required state-space form.

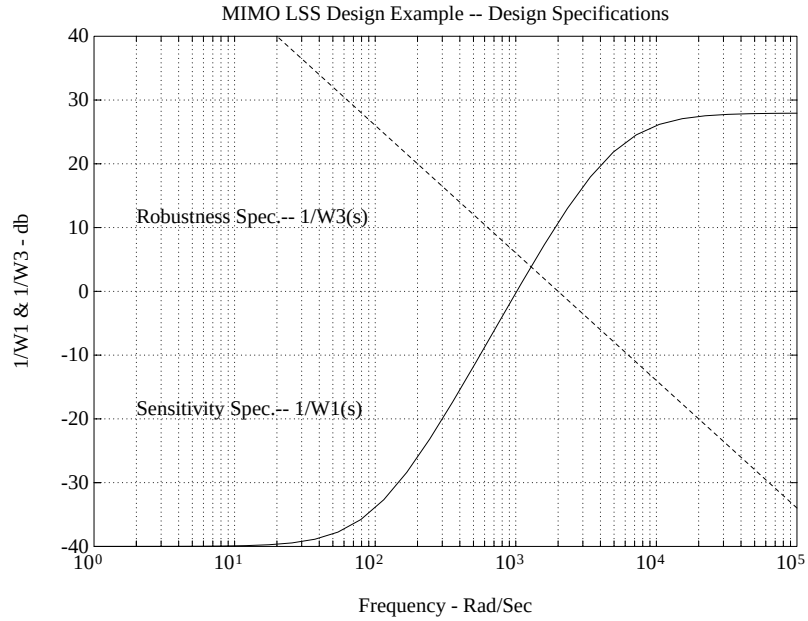


Figure 1-36: Singular Value Specifications (Weighting Functions)

Control Actions

Our design strategy for the LSS's LOS loops is as follows:

1. Use collocated rate feedback to damp out the structural modes (inner loops) and to make it possible to use the six primary mirror actuators to control the two LOS outputs (outer loops).

2. Use model reduction (aiming at a 4-state reduced order model from the 116-state original plant model).

3. Augment the plant with W_1 and W_3 as above, then use the $\mathbf{H}^{\infty}$ control design method to increase the system bandwidth and to push down the sensitivity.

4 Digitize the MIMO control law for implementation using `bilin` to compute the "shifted Tustin transform" of the controller.

Model Reduction

The model reduction algorithms in the toolbox (`bstschmr`, `slowfast`, `ohklmr`, `obalreal`) were used to find a reduced 4-state model that satisfies the "robustness criterion" (see "The Robust Control Problem" on page 1-10). The 4-state approximation of the plant with "square-down" filter is $G(s) = C(Is - A)^{-1}B$ where

$$A = \begin{bmatrix} -0.990 & 0.0005 & 0.4899 & 1.9219 \\ 0.0009 & -0.9876 & 1.9010 & -0.4918 \\ -04.961 & -1.9005 & -311.7030 & 4.9716 \\ -1.9215 & 0.4907 & -7.7879 & -398.3118 \end{bmatrix}; \quad B = \begin{bmatrix} 0.7827 & -0.6140 \\ 0.6130 & 0.7826 \\ 0.7835 & 0.5960 \\ 0.6069 & -0.7878 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.7829 & 0.6128 & -0.7816 & -0.6061 \\ -0.6144 & 0.7820 & -0.5984 & 0.7884 \end{bmatrix}$$

After augmentation with W_1 and W_3, the reduced plant model has eight states.

Results

The $\mathbf{H}^{\infty}$ controller achieves the specifications in 2 iterations on the parameter γ (`Gam`). The results are shown in Figures 1-37 to 1-39.

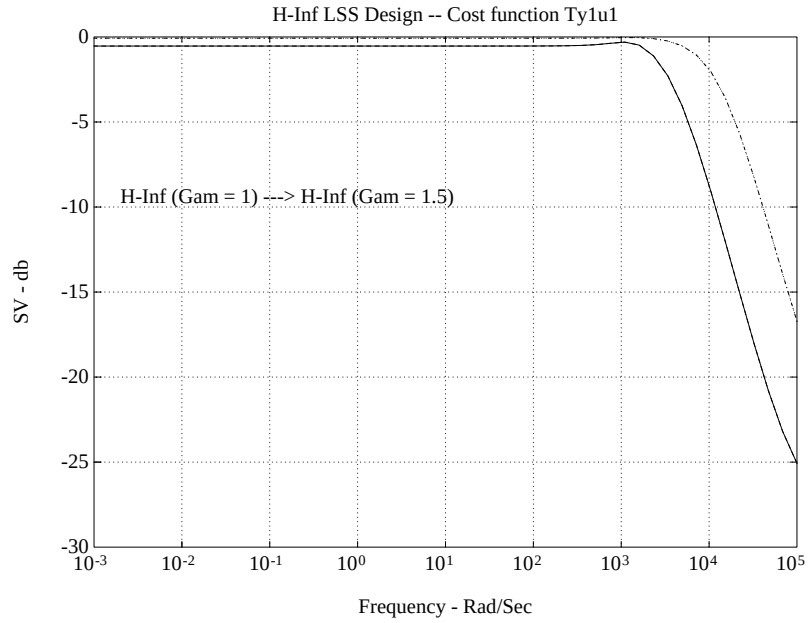


Figure 1-37: Cost Function

The final $\mathbf{H}^\infty$ controller is stable and has 8 states.

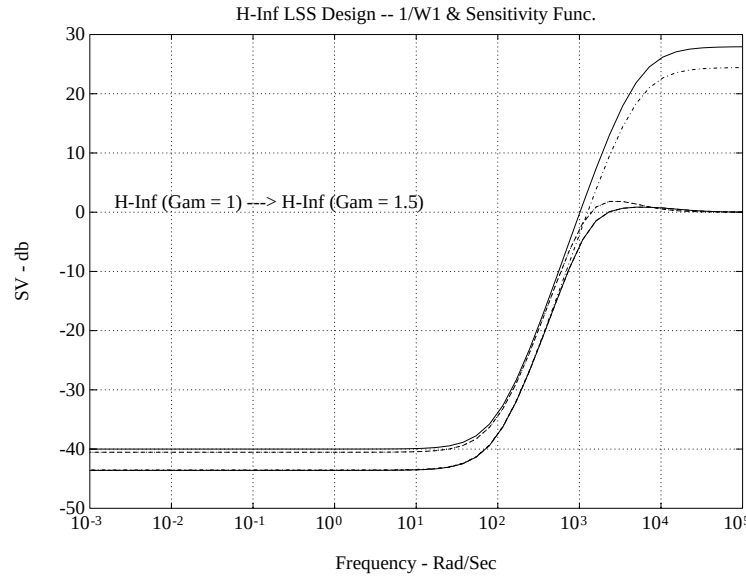


Figure 1-38: Sensitivity Function and Weighting

$\mathbf{H}^\infty$ Synthesis for a Double-Integrator Plant

A class of plants that is often encountered in modern control applications consists of those containing a double-integrator, for example:

- Rigid body dynamics of a spacecraft ignoring structural effects.
- Laser pointing device mounted on a shaft controlled by motor.

Regardless of the specific application, a double-integrator plant can be stabilized with a robust *mixed sensitivity* $\mathbf{H}^\infty$ controller. However, one special feature of the $\mathbf{H}^\infty$ *mixed sensitivity* approach prevents us from meeting our goal easily:

The plant is not allowed to have $j\omega$*-axis poles and/or zeros!*

But this does not mean we can not deal with the situation. The following design procedure circumvents the difficulty nicely:

1 Transform the double-integrator plant $G(s) = (ag,bg,cg,dg)$ via a

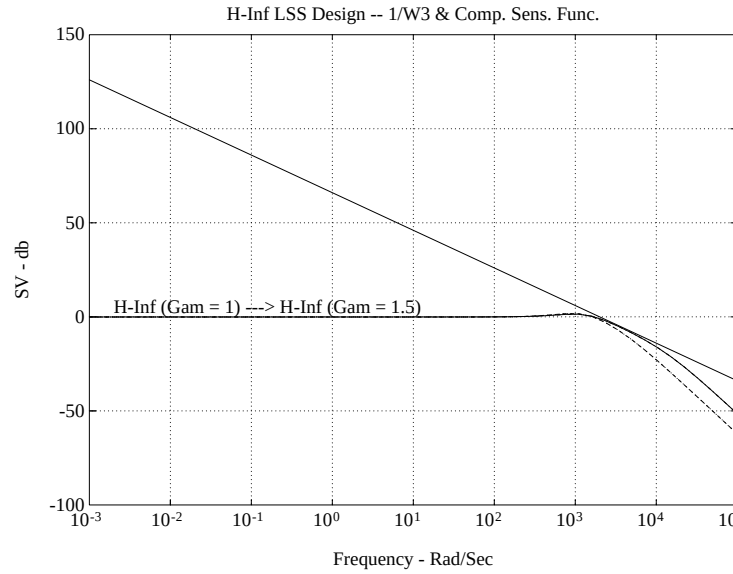


Figure 1-39: Complementary Sensitivity Function and W_3^{-1}

special bilinear transform

$$s = \frac{\tilde{s} + p_1}{\frac{\tilde{s}}{p_2} + 1}$$

where the circle points $p_2 = \infty$, and $p_1 < 0$. This is equivalent to simply shifting the $j\omega$-axis by p_1 units to the left

$$ag \leftarrow ag - p_1 * I.$$

2 Find a standard mixed-sensitivity $\mathbf{H}^\infty$ controller $F(\tilde{s})$ for the shifted problem.

3 Shift back the controller $F(\tilde{s}) = (acp, bcp, ccp, dcp)$ to $F(s)$

$$acp \leftarrow acp + p_1 * I.$$

This $j\omega$-axis shifting also guarantees that the controller is proper.

Example: The rotational dynamics of a spacecraft rigid body can be modeled as

$$G(s) = \frac{1}{Js^2}$$

where $J = 5700$ denotes the polar moment of inertia. The design goal is to find a stabilizing controller $F(s)$ that has a control loop bandwidth 10 rad/sec.

A mixed sensitivity problem can be formulated as follows

$$\min_{F(s)} \left\| \begin{bmatrix} W_1(I + GF)^{-1} \\ W_3 GF(I + GF)^{-1} \end{bmatrix} \right\|_\infty < 1$$

and the problem of $j\omega$-axis plant poles can be solved via axis shifting technique. But there are still two plant zeros at infinity, which are also on the $j\omega$-axis. This can be taken care via a clever W_3 weighting

$$W_3(s) = \frac{s^2}{100}$$

where the double-differentiator makes the plant full rank at infinity, but also serves as the complementary sensitivity weighting function and limits the control the system bandwidth to 10 rad/sec. The nonproper weight W_3 can be absorbed into the plant via routine `augtf.m`.

The parameters of a second order W_1 weighting serve as our “design knobs”

$$W_1(s) = \frac{\beta(\alpha s^2 + 2\zeta_1 \omega_c \sqrt{\alpha} s + \omega_c^2)}{(\beta s^2 + 2\zeta_2 \omega_c \sqrt{\beta} s + \omega_c^2)}$$

Suitable values for the parameters are

$\beta = 100$: DC gain of the filter (controls the disturbance rejection)

$\alpha = \frac{2}{3}$: high frequency gain (controls the response peak overshoot)

$\omega_c = 3$: filter cross-over frequency

$\zeta_1, \zeta_2 = 0.7$: damping ratios of the corner frequencies.

The following commands lead to a robust mixed-sensitivity controller for the double integrator plant (or try `dintdemo.m`).

```
[ag,bg,cg,dg] = tf2ss(1/5700,[1 0 0]);
% Shift the JW-Axis to the left by 0.1 unit:
ag0 = ag + 0.1*eye(size(ag));
w2 = []; w3 = [1 0 0;0 0 100];
beta = 100; alfa = 2/3; w1c = 3;
zeta1=0.7; zeta2=0.7;
w1 =[beta*[alfa 2*zeta1*w1c*sqrt(alfa) w1c*w1c];
[beta 2*zeta2*w1c*sqrt(beta) w1c*w1c]];
ssg = mksys(ag0,bg,cg,dg);
TSS = augtf(ssg,w1,w2,w3);
[sscp,sscl,hinfo] = hinf(TSS);
[acp,bcp,ccp,dcp] = branch(sscp);
[acl,bcl,ccl,dcl] = branch(sscl);
% Shift the JW-Axis to the right by 0.1 unit:
acp = acp - 0.1*eye(size(acp));
dinteva % computing the time and frequency responses
dintplt % plotting
```

The resulting plot shown in Figure 1-40, Results of H• Synthesis for Double-Integrator Plant

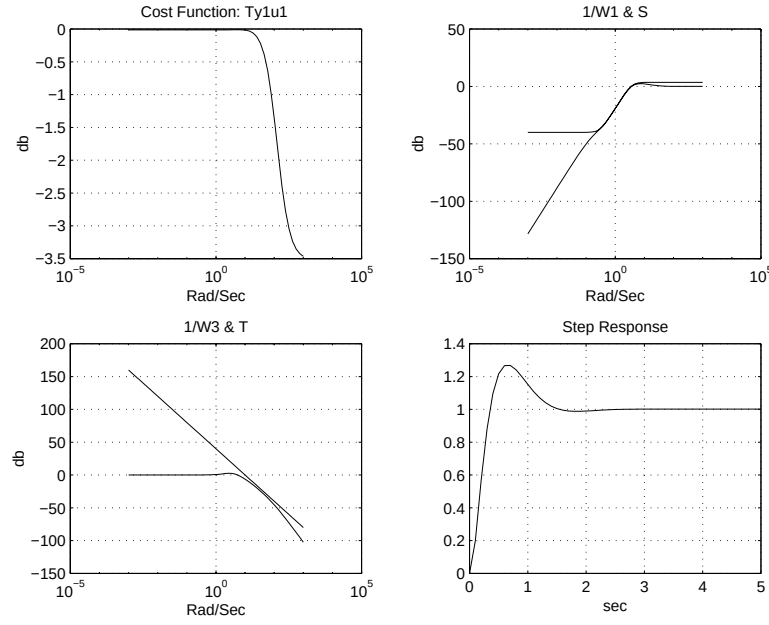


Figure 1-40: Results of H^∞ Synthesis for Double-Integrator Plant

Bilinear Transform + H^∞ on ACC Benchmark Problem

The benchmark problem described by Wie and Bernstein [49] (see Figure 1-41, The Benchmark Problem) was solved via the bilinear transform and $\mathbf{H}^\infty$technique.

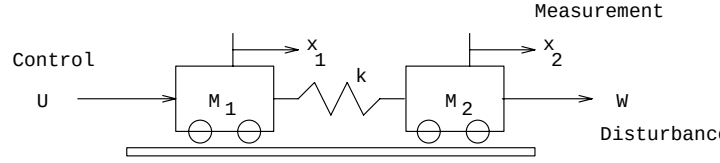


Figure 1-41: The Benchmark Problem

This undamped spring-mass system has the following transfer function

$$\frac{z}{u} = \frac{k}{m_1 s^2 \left[m_2 s^2 + \left(1 + \frac{m_2}{m_1}\right) k \right]}$$

where the masses m_1, m_2, and the spring constant k take the value 1.0 as nominal, but can be uncertain. In addition, the measurement variable z (picked up by the sensor) is not collocated with the actuator signal u, which introduces extra phase lag into the system and makes it much harder to control. If we were fortunate enough to have a collocated actuator/sensor set-up, the system would have been "passive" (i.e., not more than $\pm 90 deg$ phase shift at any frequency) and therefore guaranteed stable with at least *90deg* phase margin for any pure gain negative feedback. However, this is not the case and the plant actually has very large phase lags—this is why the problem is challenging.

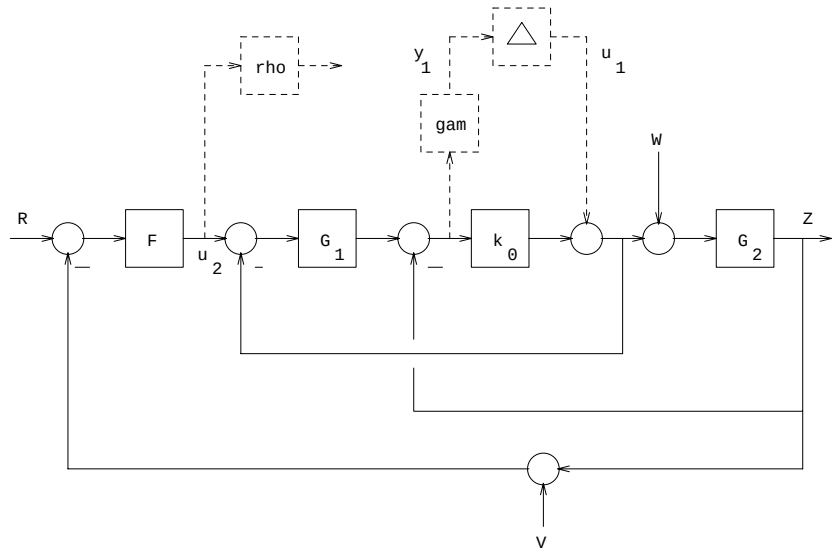


Figure 1-42: The Robust Control Problem Formulation

The design requirement is to find a controller that:

1 Stabilizes the system with an uncertain spring constant k varying between 0.5 and 2, and

2 Has an impulse response settling time (T_s) of the second mass approximately 15 sec, and

3 Has reasonable control energy.

We form the problem as shown in Figure 1-42, The Robust Control Problem Formulation. The nominal value of the spring constant k was set to 1.25 and the uncertainty ($\|\Delta\|_\infty \le 1$) is scaled by a parameter γ. In parallel, we penalize the control signal by another parameter ρ. Then, we formulate an $\mathbf{H}^\infty$ small-gain problem.

$$\left\| \begin{bmatrix} \gamma T_{y1u1} \\ \rho T_{u2u1} \end{bmatrix} \right\|_\infty < 1$$

The objective here is to maximum the robustness level γ with minimum control energy 1/ρ. As control energy is allowed to increase by decreasing 1/ρ, the maximal achievable robustness level γ increases. The trade-off is shown inFigure 1-43, Trade-off Between Robustness and Control Energy

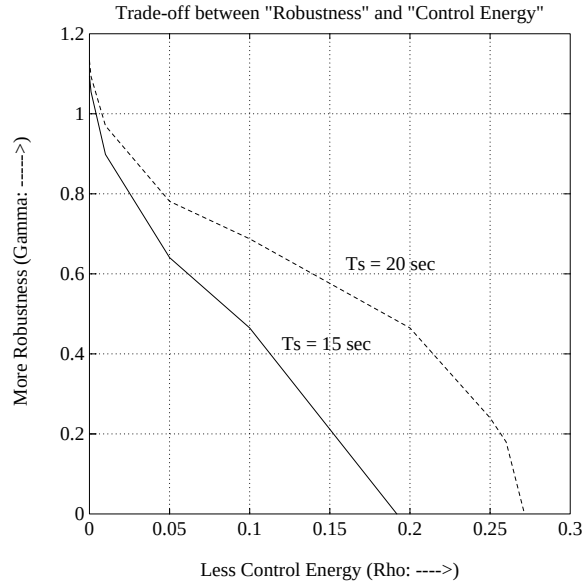


Figure 1-43: Trade-off Between Robustness and Control Energy

The system settling time T_s was controlled naturally by the bilinear transform parameter p_1. For this design, the parameter p_1 of bilinear transform was selected via a simple rule-of-thumb in classical control [8]

$$T_{settling} \approx \frac{4}{\zeta\omega_n}$$

where $T_{settling} = 15\ sec$ and the real part of the dominant closed-loop poles is $p_1 = \zeta\omega_n \approx -0.3$. Parameter $p_2 = -100$ was selected to have large magnitude (i.e., $|p_2|$ much greater than the control bandwidth).

Figures 1-44 through 1-46 show a design with $p_1 = -0.35,\ p_2 = \infty,\ \gamma = 0$ and $\rho = 0.01$. As indicated, this design has met all three requirements.

Notice that $\gamma = 0$ in this case is all we need to satisfy the robustness require "real" parameter uncertainty.

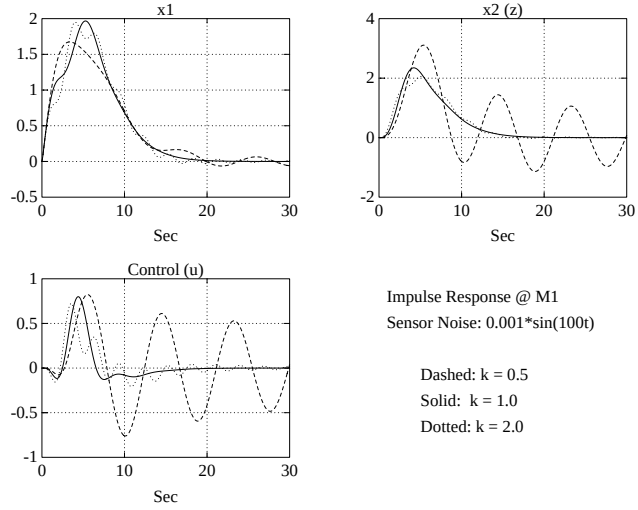


Figure 1-44: Impulse Response (Settling Time ≈ 15 Seconds)

μ Synthesis Design on ACC Benchmark Problem ACC Benchmark Problem

The benchmark problem proposed by Wie and Bernstein [49] was solved for a different requirement, viz., to maximize the MSM on simultaneous (M_1, M_2, k)

variations subject to a settling time constraint $T_{settling} = 15\,sec$. This falls naturally into the framework of μ synthesis.

First, a direct $\mathbf{H}^{\infty}$ design was formulated with additive uncertainties in M_1, M_2 and k pulled out and penalized as an $\mathbf{H}^{\infty}$ Small-Gain problem. Using the γ-iteration procedure `hinfopt`, an "optimal" $\mathbf{H}^{\infty}$ controller was found. The structured-singular-value routine `ssv` then computed the cost function (MSM) in $s/\tilde{s}$ -plane as well as the 3×3 diagonal scaling $D(s)$ (see Figures 1-47 and 1-48). The second step is to curve-fit the diagonal scaling $D(s)$ using the toolbox function `fitd` and absorb it into the plant for another $\mathbf{H}^{\infty}$ design iteration (see Figure 1-48, Diagonal Scaling D(s) and Curve Fitting for the curve-fitting). The design was pushed to the limit after one cycle μ of synthesis, i.e., no more improvement can be squeezed out of the problem formulation.

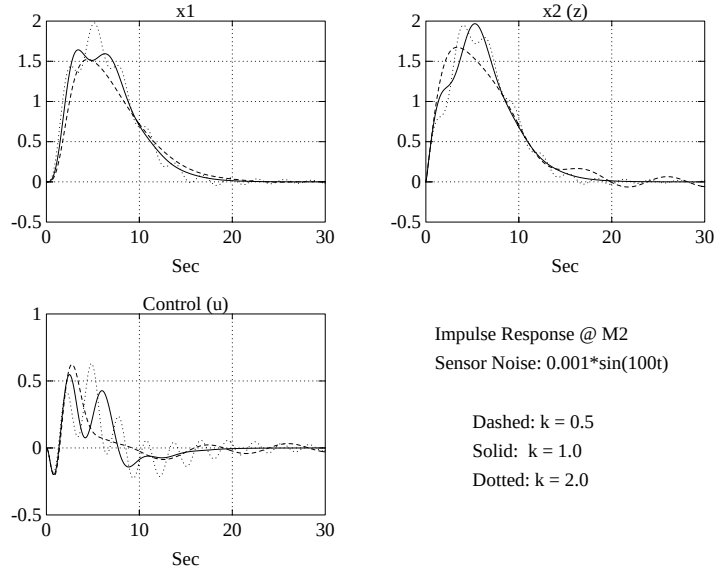


Figure 1-45: Impulse Response (Settling Time ≈ 15 Seconds)

Figure 1-49, Impulse Response (Settling Time Seconds) shows the impulse response excited at Mass 1. The design parameters were chosen to be $p_1 = -0.4$ and $p_2 = -100$. The settling time specification of Mass 2 is met (≈ 15 sec). The system is robust against ±23% simultaneous plant variations in (M_1, M_2, k) as well as high frequency sensor noise ($0.001sin(100t)$).

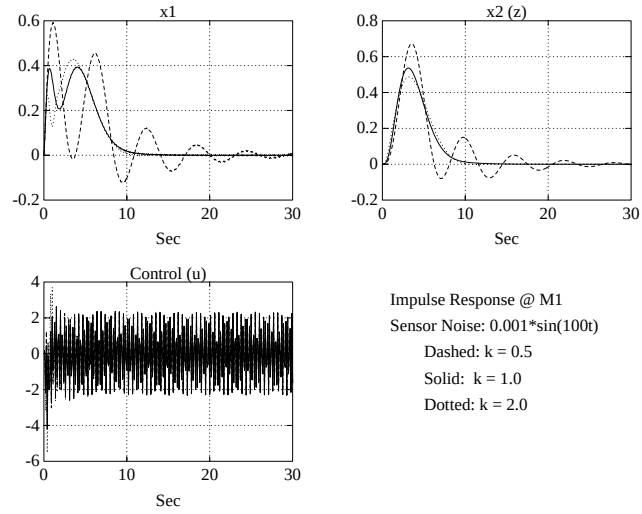


Figure 1-46: Controller and Loop Transfer Function

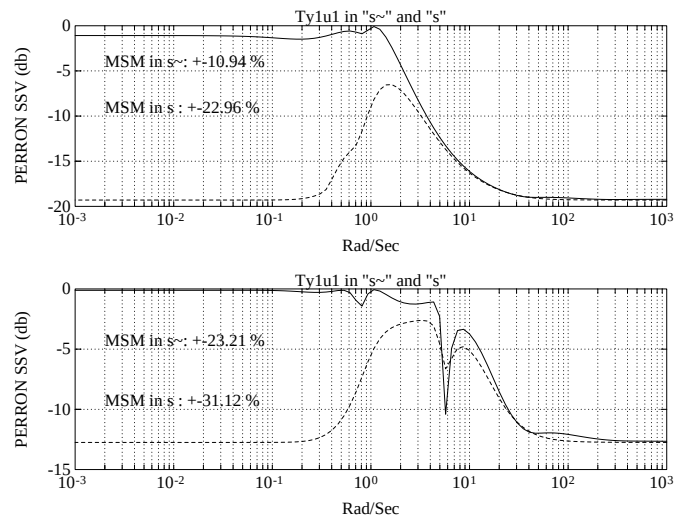


Figure 1-47: Structured Singular Values (2 iterations)

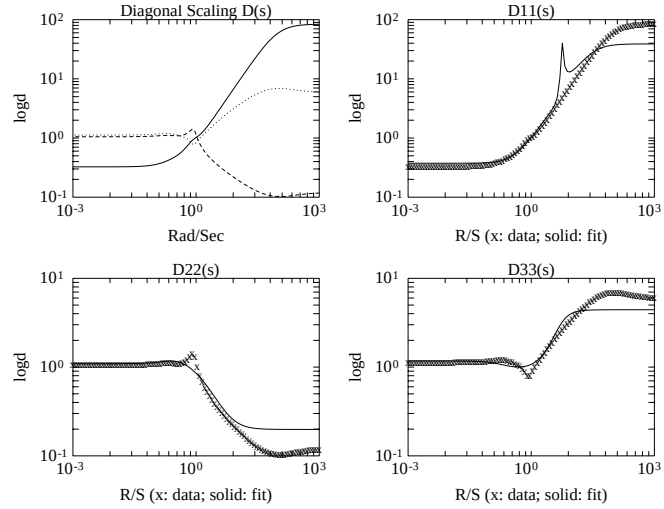


Figure 1-48: Diagonal Scaling $D(s)$ and Curve Fitting

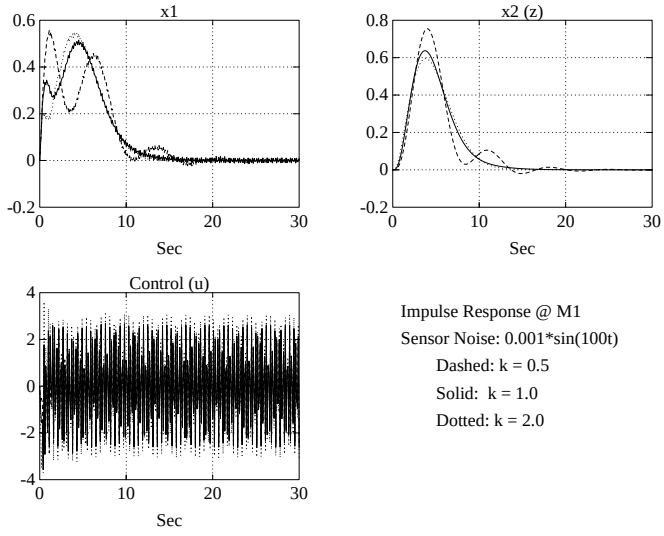


Figure 1-49: Impulse Response (Settling Time ≈ 15 Seconds)

Model Reduction for Robust Control

In the design of controllers for complicated systems, model reduction arises in several places:

1. It is desirable to simplify the best available model in light of the purpose to which the model is to be used — namely, to design a control system to meet certain specifications.

2. In using certain design methods (including the $\mathbf{H}^{\infty}$ method implemented in `linf`), fictitious unobservable/uncontrollable states are generated by the algorithms which must be stripped away by a reliable model reduction algorithm.

3. Finally, if a modern control method such as LQG or $\mathbf{H}^{\infty}$ (see `h2lqg` or `hinf`) is employed for which the complexity of the control law is not explicitly constrained, the order of the resultant control law is likely to be considerably greater than is truly needed. A good model reduction algorithm applied to the control law can sometimes significantly reduce control law complexity with little change in control system performance.

However, a good model reduction routine has to be both numerically robust and be able to address the closed-loop "robustness" issues.

The Robust Control Toolbox model reduction routines `schmr`, `ohklmr` and `reschmr` all possess the following special features:

1. They bypass the ill-conditioned balancing transformation `balreal`, hence, they can easily deal with the "nonminimal" systems.

2. They employ Schur decomposition to robustly compute the orthogonal bases for eigenspaces required in intermediate steps.

3. They have an $\mathbf{H}^{\infty}$-norm error bound. The infinity norm of either the *relative error* or the *absolute error* of the reduced order model is bounded by a precomputable positive real number for all frequency.

Achievable Bandwidth vs. H^{∞} Modeling Error

In modeling, the validity of a given model always depends on the intended use for the model. In doing model reduction for control purposes, singular value

Bode plots of the reduced plant model and the model's error provide the information needed to assure a given reduced model is sufficiently accurate to be used in the design of a control system with a prescribed bandwidth. Using the $\mathbf{H}^{\infty}$ error-bound, it is possible to associate a "robust frequency" (ω_{r_A} or ω_{r_M}) with a reduced model such that the model may be reliably used for any control design whose bandwidth does not exceed the robust frequency. The remainder of this section elaborates on this point.

Additive Model Reduction

Let $G(s),\ \tilde{G}(s) \in C^{n \times n}$ be a "true" plant transfer function matrix and its reduced model, respectively. Then, the *additive modeling error* is defined as (see Figure 1-50, Additive Plant Uncertainty)

$$\tilde{\Delta}_A := G - \tilde{G}$$

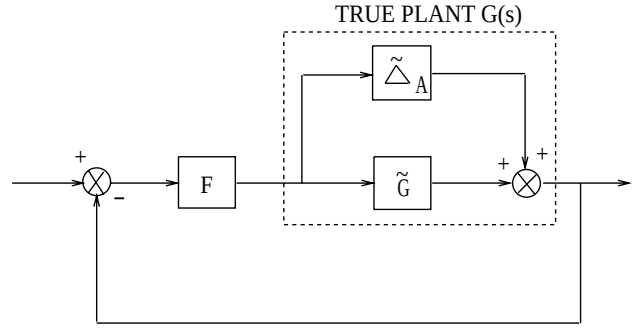


Figure 1-50: Additive Plant Uncertainty

Definition: Given G, $\tilde{G}$ and $\tilde{G}_A$ as above, the *additive robust frequency* is

$$\omega_{r_A} := max\left\{ \omega \ \middle|\ \underline{\sigma}(\tilde{G}(j\omega)) \geq \bar{\sigma}(\tilde{\Delta}_A(j\omega)) \right\}$$

Loosely speaking, the bandwidth ω_B of a control system is the frequency range where the loop transfer function is "big", i.e.,

The bandwidth is also the frequency range over which $\bar{\sigma}(S) \ll 1$ and, hence, over which the feedback is effective in attenuating disturbances. The

$$\underline{\sigma}(\tilde{G}F) \gg 1 \quad \forall \omega < \omega_B$$

significance of the robust frequency in the context of model reduction is that, if $\bar{\sigma}(\Delta_A(j\omega))$ were the only information available about the modeling error, then the robust frequency ω_{r_A} is an upper bound on the bandwidth of any multivariable control system that can be designed without causing the sufficient condition for stability

$$\frac{\bar{\sigma}(\tilde{\Delta A})}{\underline{\sigma}(\tilde{G})} \bar{\sigma}\tilde{G}F(I + \tilde{G}F)^{-1} < 1 \tag{1-8}$$

to be violated at some frequency within the bandwidth. Notice that for $\omega < \omega_B$, you have

$$\bar{\sigma}(\tilde{G}F(I + \tilde{G}F)^{-1}) \approx 1$$

Thus you have the following Additive Error robustness Criterion: [38]

If $\bar{\sigma}(\Delta A) \le \underline{\sigma}(\tilde{G})$ *for* $\omega \le \omega_{f_A}$ *and if* ΔA *is open-loop stable, then the closed-loop system will not be destabilized by* ΔA *provided that the control bandwidth is less than* ω_{r_A}. *(Note: This is only sufficient, not necessary.)*

Of course, since the bandwidth is only defined in terms of the fuzzy "much greater" ($\gg$) inequality, this is not a theorem in a rigorous mathematical sense.

The implication is that to achieve the bandwidth of, say *2000* rad/sec, it suffices to use any reduced model whose robustness frequency $\omega_{r_A} > 2000$ rad/sec. Note that our "robustness criterion" requires that *G* be square; this assumption cannot be relaxed. Accordingly, the nonsquare plant must be "squared down" with a suitable pre-compensator before we can apply the "robustness criterion".

Additive Model Reduction Methods

Four methods are available to do the additive error model reduction:

1. Ordered balanced realization (`obalreal`).
2. Truncated balanced model reduction (`balmr`).
3. Schur balanced model reduction (`schmr`).

4 Optimal Hankel approximation without balancing (`ohklmr`).

The regular balanced realization (`obalreal`) is known to be ill-conditioned when the model is nonminimal. The popular truncated version of square-root balancing (`balmr`) can be ill-conditioned when the system has some modes that are strongly controllable but weakly observable, or vice versa. The most numerically robust methods are `schmr` and `ohklmr`. All four methods are discussed further in the Reference section.

Each of the above methods possess the same infinity-norm error bound for a k-th order reduced order model $G(s)$ of an m-th order system $G(s)$:

$$\bar{\sigma}(G(j\omega) - \tilde{G}(j\omega)) \leq 2 \sum_{i=k+1}^{m} \sigma_i \quad \forall\omega$$

where σ_i, called the *Hankel singular values*, are computed from the observability-reachability grammians of $G(s)$ as described in the Reference section under `obalreal`.

Multiplicative Model Reduction

The quantity $\frac{\bar{\sigma}(\tilde{\Delta A})}{\underline{\sigma}\tilde{G}}$ in equation (9) is a crude measure of the *relative*

(multiplicative) error of a plant G defined as (see Figure 1-51, Multiplicative Plant Uncertainty)

$$\tilde{\Delta}_M \stackrel{\Delta}{=} (G - \tilde{G})\tilde{G}^{-1}$$

A slightly less conservative sufficient condition for stability than (9) is

$$\bar{\sigma}(\Delta_M) < 1/\bar{\sigma}(GF(I+GF)^{-1}) \qquad \textbf{(1-9)}$$

Definition: Given G, $\tilde{G}$ and Δ_M as, the *multiplicative robust frequency is*

$$\omega_{r_M} := max\left\{\omega \mid \bar{\sigma}(\tilde{\Delta}_M) \leq 1\right\}$$

Loosely speaking, the bandwidth ω_B of a control system is the frequency range where the loop transfer function is "big", i.e.,

$$\underline{\sigma}(\tilde{G}F) \gg 1 \quad \forall \omega < \omega_B$$

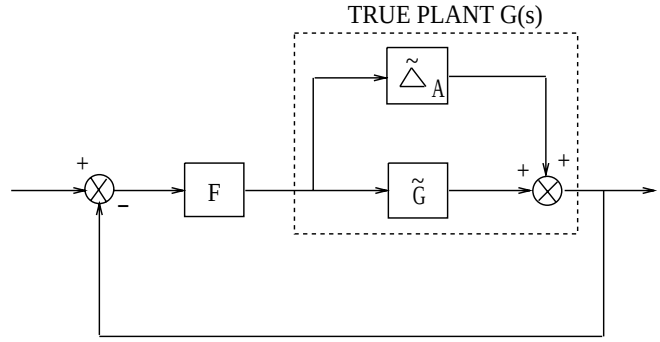


Figure 1-51: Multiplicative Plant Uncertainty

The significance of the robust frequency in the context of model reduction is that, if $\bar{\sigma}(\Delta_M(j\omega))$ were the only information available about the modeling error, then the robust frequency ω_{r_M} is an upper bound on the bandwidth of any multivariable control system that can be designed without causing the sufficient condition for stability to be violated at any frequency within the bandwidth. Notice that for $\omega < \omega_B$ where the inequality (1-9) is satisfied, you have

$$\bar{\sigma}(\tilde{G}F(I + \tilde{G}F)^{-1}) \approx 1 \qquad \textbf{(1-10)}$$

Thus when a bound on the relative error $\bar{\sigma}(\Delta_M)$ is available, you can strengthen the Additive Error Robustness Criterion to the following Multiplicative Error Robustness Criterion:

If $\bar{\sigma}(\tilde{\Delta M}) \le 1$ *for* $\omega \le \omega_{r_M}$ *and if* $\tilde{\Delta M}$ *is open-loop stable, then the closed-loop system will not be destabilized by* ΔM *provided that the control bandwidth is less than* ω_{r_M}. *(Note: This is only sufficient, not necessary.)*

As before, since the bandwidth is only defined in terms of the fuzzy inequality (1-10), this is not a theorem in a rigorous mathematical sense.

While (1-10) implies that the size of the relative error (viz., $\bar{\sigma}(\tilde{\Delta_M})$) must be less than one throughout the specified control loop bandwidth (viz., $\omega \le \omega_B$), it is interesting to note that it is not very important how much less than one the relative error bound is. It makes little difference whether the size of the relative error is *0.5* or *0.0005*; it only matters that the relative error is less than one.

Note that you can easily show using the properties of singular values that

$$\bar{\sigma}(\Delta_M) \ge \frac{\bar{\sigma}(\Delta_A)}{\underline{\sigma}(G)}$$

Hence, you always have that $\omega_{r_M} \ge \omega_{r_A}$; that is, the multiplicative error robustness criterion is *always* less conservative than the additive error robustness

Multiplicative Model Reduction Method

A relevant result of *balanced stochastic truncation* (BST) combined with *relative error bound* (REM) has achieved the "optimal" solution for robust model reduction. `Reschmr` implements the Schur version of the BST-REM theory and enjoys the following "relative-error" and "multiplicative-error" bounds [47, 48]:

$$\left\| G^{-1}(G - \tilde{G}) \right\|_\infty \le \text{err}$$

and

$$\left\| \tilde{G}^{-1}(G - \tilde{G}) \right\|_\infty \le \text{err}$$

where

$$\text{err} = \sum_{i=k+1}^{n} \frac{2\sigma_i}{1-\sigma_i} \qquad \textbf{(1-11)}$$

Example: Let's compare the model reduction methods via the plant

$$G(s) = \frac{0.05(s^7 + 801s^6 + 1024s^5 + 599s^4 + 451s^3 + 119s^2 + 49s + 5.55)}{s^7 + 12.6s^6 + 53.48s^5 + 90.94s^4 + 71.83s^3 + 27.22s^2 + 4.75s + 0.3}$$

If we truncate the system to three states, the results show that the Schur-BST algorithm (`bstschmr`) is much superior to optimal Hankel model reduction (`ohklmr`) and Schur balanced truncation (`balmr`, `schmr`) — see Figure 1-52, Schur BST-REM vs. Schur BT and Hankel. The relative error bound (1-11) equals `20.781`.

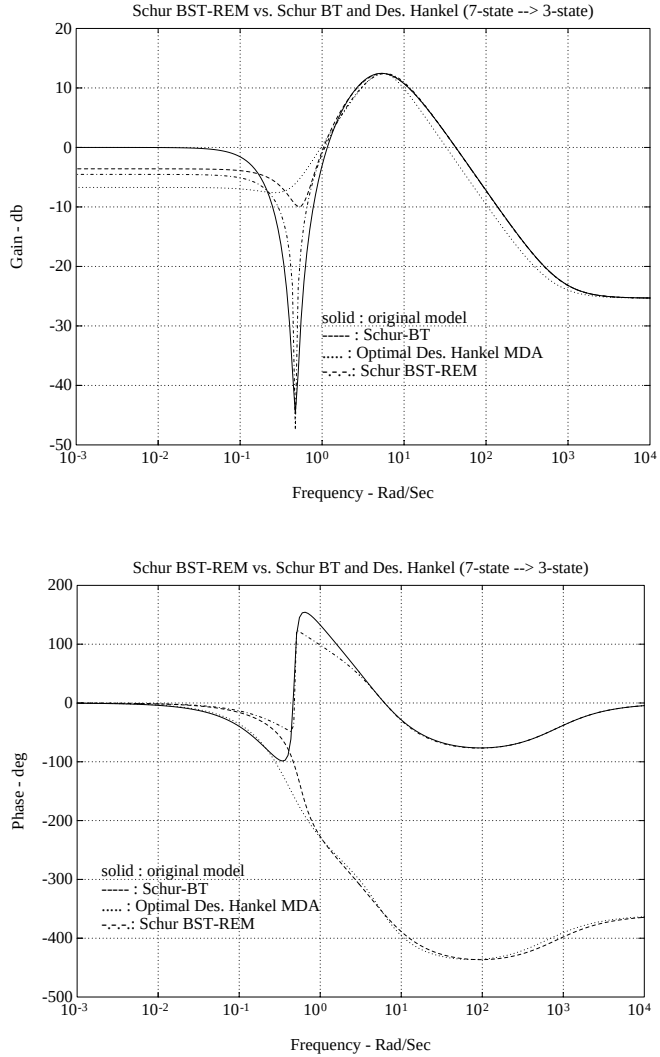


Figure 1-52: Schur BST-REM vs. Schur BT and Hankel

Sampled-Data Robust Control

Nowadays, sampled-data control systems dominate the control industry. The Robust Control Toolbox includes the function `bilin` to support "singular-value loop shaping" and "robust model reduction" techniques for sampled-data control systems.

The frequency-domain *bilinear transform*

$$s \to \frac{\alpha s + \delta}{\gamma s + \beta}$$

plays a role of bridging the gap between continuous $\mathbf{H}^\infty$ control and $\mathbf{H}^\infty$ model reduction theories and their discrete counterparts. The bilinear transform, which includes the popular *Tustin transform*

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

as a special case, can be used to transform discrete systems into equivalent continuous systems and back again. A key property of the Tustin transform is that it preserves the $\mathbf{H}^\infty$ norm.

This toolbox implements a multivariable state-space version of the bilinear transform in bilin as [35]

$$\left[\begin{array}{c|c} A_b & B_b \\ \hline C_b & D_b \end{array}\right] = \left[\begin{array}{c|c} (\beta A - \delta I)(\alpha I - \gamma A)^{-1} & (\alpha\beta - \gamma\delta)(\alpha I - \gamma A)^{-1}B \\ \hline C(\alpha I - \gamma A)^{-1} & D + \gamma C(\alpha I - \gamma A)^{-1}B \end{array}\right]$$

Robust Control Synthesis

For stable discrete-time transfer functions matrices $G(z) \in C^{m \times n}$, $p = \min\{m,n\}$, the $\mathbf{H}^2$-norm and the $\mathbf{H}^\infty$-norm are defined in terms of the frequency-dependent singular values $\sigma_i(j\omega)$ of $G(j\omega)$:

Discrete H^2-norm

$$\|G\|_2 \stackrel{\Delta}{=} \left[\int_{-\pi}^{\pi} \sum_{i=1}^{P} (\sigma_i(e^{j\omega}))^2 d\omega \right]^{\frac{1}{2}}$$

Discrete H^∞-norm

$$\|G\|_\infty \stackrel{\Delta}{=} \sup_{\omega} \bar{\sigma}(G(e^{j\omega})) \quad (sup\text{: } the \ least \ upper \ bound)$$

To design a digital $\mathbf{H}^\infty$ control system, you can approach it in either of the following two ways (see Figure 1-53, Sample-Data Robust Control Synthesis):

1. Design a control law $F(s)$ in the s-domain using the synthesis methods supported by this toolbox, then map the controller into the z-plane via the Tustin transform

   $$F(z) \leftarrow F(s)\Big|_{s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

   using `bilin`. Provided that the sampling frequency is several times higher than the design bandwidth, the resulting sampled-data system will perform about the same as the continuous s-plane design.

2. Add the zero-order-hold and sampler to the sampled plant $G(z)$, then map $G(z)$ into the w-plane and proceed with the design methods in the toolbox as if it were in the s-plane (watch out for the extra nonminimum phase zero added by the Z.O.H). After the design is done, map the controller back into the z-plane via the inverse w-transform (use `bilin`). Then the controller can be implemented in the z-domain directly. The $\mathbf{H}^\infty$ norm remains invariant under the Tustin transform, so no degradation of $\mathbf{H}^\infty$ performance measures results in this case.

The Robust Control Toolbox functions `dhinf` and `dhinfopt` automate this procedure.

A discussion of how the function `bilin` was used in the design of a sampled-data $\mathbf{H}^\infty$ controller for a large space structure may be found in [38, 43].

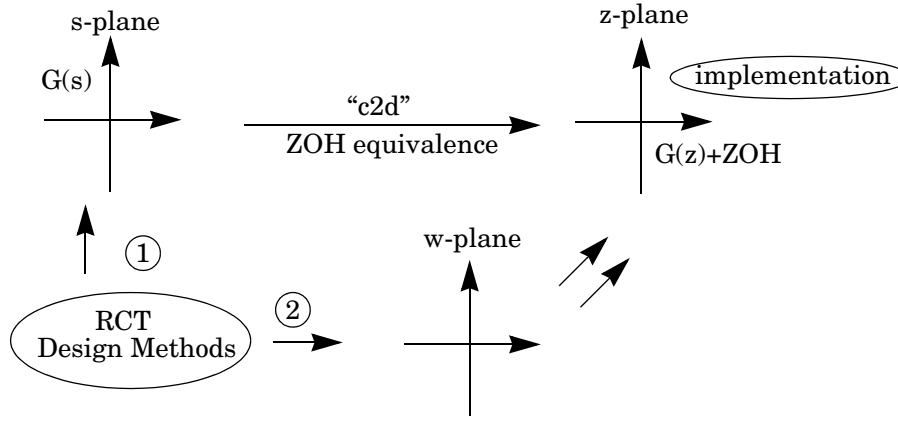


Figure 1-53: Sample-Data Robust Control Synthesis

The bilinear transform (`bilin`) can be used equally well in "classical" digital control systems design, which we will leave to the users.

Miscellaneous Algorithms

A number of algorithms are worth mentioning here:

1. Ordered Schur Decomposition (`rschur`, `cschur`).
2. Descriptor to state-space form (`des2ss`) model conversion.
3. Sector transformation (`sectf`).

Ordered Schur Decomposition

Schur decomposition plays a crucial role in numerical linear algebra, because of its special way of orthogonally triangularizing a square -dimensional real matrix as:

$$U^T A U = T = \begin{bmatrix} \lambda_{11} & \dots & \dots & * \\ 0 & \lambda_{22} & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \lambda_{nn} \end{bmatrix}$$

On the main diagonal are the eigenvalues of matrix A which may be ordered, e.g., by real part or by modulus in ascending or descending order (six types of ordering are available in `cschur`). Moreover, for any integer $1 \le k < n$, the first k columns of the U matrix form an orthonormal basis for the span of the eigenvectors associated with the first k eigenvalues $_{11},\dots,\lambda_k$. Comparing to the regular eigenspace decomposition (`eig`), this method is numerically much more robust, especially when A is near to a matrix having Jordan blocks of size greater than one.

red Schur decomposition arises as a subroutine used by many of the functions in the Robust Control Toolbox:

1. Stable/unstable projection (`stabproj`).
2. Slow/fast decomposition (`slowfast`).
3. Algebraic Riccati solver (`aresolv`, `lqrc`).
4. Model reduction via Schur balancing (`schmr`).

5 Model reduction via Schur stochastic balancing (`bstschml`).

6 Optimal control synthesis (`h2lqg`, `hinf`, `hinfopt`).

Descriptor System

In "modern" robust control computations, the descriptor system representation

$$E\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

has become increasingly important because it enables you to circumvent numerical ill-conditioning problems that are often unavoidable when you employ the less flexible state-space representation where E = I. However, even though the descriptor form may be preferable for numerical robustness inside algorithms, most users still prefer to have the final answer in state-space form. The function `des2ss` provides the conversion from descriptor form to state-space form. You can say that in the next few years, descriptor system type of operations will arise more and more often in control theory derivations and conceptual problem formulations.

Several functions in the Robust Control Toolbox make use of a descriptor form system representation internally then make use of `des2ss` as a final step to return answers in the more familiar state-space form. For example:

1 Optimal Hankel approximation without balancing (`ohklmr`).

2 The "2-Riccati" $\mathbf{H}^{\infty}$ control solution (`hinf`, `hinfopt`); these routines use `des2ss` to convert the final all-solution controller from the descriptor given in [42] to state-space form.

Sector Transform

In robust control theory, conic sector theory plays a key role in transforming one problem to another [50]. The function `sectf` enables you to convert from *sector[a, b]* to *sector[0, ∞]*, or to *sector[–1, 1]*, etc. These kinds of transformations can convert a *positive real problem* into a *small-gain problem*, or vice versa [36]. The *small-gain* problem can then be readily solved using the toolbox M-files `hinf`, `linf`, or `h2lqg`, etc.

SVD System Realization

A system identification method proposed by Kung in 1978 [21] is coded in function `imp2ss`. This function takes a discrete impulse response and forms the Hankel matrix Γ. Based on singular value decomposition of Γ, a discrete state-space realization can be created. Details are documented in the reference section under `imp2ss`.

Closing Remarks

This toolbox is the result of many years of research work, evolving into its present form as the authors were involved in several aerospace and industrial design studies using modern theories such as $\mathbf{H}^{\infty}$, μ synthesis and balanced stochastic truncation model reduction for which commercial software did not previously exist. At the core of the toolbox are the robust model reduction algorithms, the structured singular values routines and the singular value loop shaping synthesis methods, e.g., loop transfer recovery, $\mathbf{H}^2$ synthesis and $\mathbf{H}^{\infty}$ synthesis and curve fitting routines like `fitd` and `fitgain` which enable you to extend the $\mathbf{H}^{\infty}$ methodology to do μ synthesis.

We have tried our best to ensure that the algorithms are numerically robust and are based on the best theory available. However, as with any other software package, mistakes and oversights are possible. We intend to update this toolbox periodically and welcome user feedback.

There are a number of people to whom we are deeply grateful. David Limebeer and Michael Green at Imperial College and Keith Glover at Cambridge University, and Wes Wang at MathWorks, Inc., and J. C. (Jerry) Juang at National Cheng Kung University we thank for their encouragement and suggestions. Jack Little at MathWorks, Inc., we thank for his foresight and patience while waiting to see this toolbox released.

For technical questions or suggestions, we can be reached at

University of Southern California
Dept. of Electrical Engineering — Systems
University Park
Los Angeles, CA 90089-2563
E-mail: `richiang@ampere.usc.edu`, `msafonov@usc.edu`

References

[1] M. Athans, "The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design," I*EEE Trans. on Automatic Contro*l, AC-16, 6, pp. 529-551, December 1971.

[2] R.Y. Chiang and M. G. Safonov, "The LINF Computer Program for $\mathbf{L}^{\infty}$ Controller Design," USC report EECG-0785-1, ver. 1.0 2.0, July, 1986 and 1987.

[3] R. Y. Chiang, *Modern Robust Control Theory*. Ph. D. Dissertation: USC, 1988.

[4] R. Y. Chiang, M. G. Safonov and J. A. Tekawy, "$\mathbf{H}^{\infty}$ Flight Control Design with Large Parametric Robustness," *Proc. of American Contr. Conf.*, San Diego, CA, May, 1989.

[5] R. Y. Chiang and M. G. Safonov, "$\mathbf{H}^{\infty}$ synthesis using a Bilinear Pole Shifting Transform," *AIAA, J. Guidance, Control and Dynamics*, vol. 15, no. 5, pp. 1111-1117, September–October 1992.

[6] R. Y. Chiang and M. G. Safonov, "Real $\mathbf{K}_m$ Synthesis Using Generalized Popov Multiplier," *Proc. of American Control Conf.*, pp. 2417-2418, Chicago, IL, June 24-26, 1992.

[7] R. Y. Chiang, M. G. Safonov, K. Haiges, K. Madden, and J. Tekawy, "A Fixed $\mathbf{H}^{\infty}$ Controller for a Supermaneuverable Fighter Performing the Herbst Maneuver," *Automatica (Special Issues on Robust Control)*, vol. 29, no. 1, pp. 111-127, January 1993.

[8] R. C. Dorf, *Modern Control Systems (5th ed.)*. Reading, MA: Addison-Wesley, 1989.

[9] P. Dorato (editor), *Robust Control*. New York: IEEE Press, 1987.

[10] P. Dorato and R. K. Yedavalli (editors), *Recent Advances in Robust Control*. New York: IEEE Press, 1990.

[11] J. C. Doyle, "Robustness of Multiloop Linear Feedback Systems," *Proc. of 1978 IEEE Conf. on Decision and Control*, pp. 12-18, San Diego, CA, January 8-10, 1979.

[12] J. C. Doyle and G. Stein, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," *IEEE Trans. on Automat. Contr.*, AC-26, 1, pp. 4-16, February 1981.

[13] J. C. Doyle, "Analysis of Feedback Systems with Structured Uncertainties," *IEE Proc.*, 129 (Pt. D), No. 6, pp. 242-250 November 1982.

[14] J. C. Doyle, J. E. Wall and G. Stein, "Performance and Robustness Analysis for Structured Uncertainty," *Proc. IEEE Conf. on Decision and Control*, pp. 629-636, Orlando, FL, December 1982.

[15] J. C. Doyle, "Synthesis of Robust Controllers and Filters with Structured Plant Uncertainty," *Proc. IEEE Conf. on Decision and Control*, pp. 109-114, San Antonio, TX, December 14-16, 1983.

[16] J. C. Doyle, *Advances in Multivariable Contro*l. Lecture Notes at ONR/Honeywell Workshop. Minneapolis, MN, Oct. 8-10, 1984.

[17] J. C. Doyle, K. Glover, P. Khargonekar and B. Francis, "State Space Solution to Standard $\mathbf{H}^2$ and $\mathbf{H}^\infty$ Control Problems," *IEEE Trans. on Automatic Contro*l, AC-34, 8, pp. 832-847, August 1989.

[18] B. A. Francis, *A Course in* $\mathbf{H}^\infty$ *Control Theory*, New York: Springer-Verlag, 1987.

[19] K. Glover and J. C. Doyle, "State Space Formulae for All Stabilizing Controllers that Satisfy an $\mathbf{H}^\infty$-Norm Bound and Relations to Risk Sensitivity", *Systems and Control Letters*, 11, pp. 167-172, 1988.

[20] K. Glover, D. J. N. Limebeer, J. C. Doyle, E. M. Kasenally and M. G. Safonov, "A Characterization of All Solutions to the Four Block General Distance Problem," *SIAM J. Control*, 29, 2, pp. 283-324, March 1991.

[21] S. Y. Kung, "A New Identification and Model Reduction Algorithm via Singular Value Decompositions," *Proc.Twelth Asilomar Conf. on Circuits, Systems and Computers*., pp. 705-714, November 6-8, 1978.

[22] N. A. Lehtomaki, N. R. Sandell, Jr., and M. Athans, "Robustness Results in Linear-Quadratic Gaussian Based Multivariable Control Designs," *IEEE Trans. on Automat. Contr*., vol. AC-26, No. 1, pp. 75-92, Feb. 1981.

[23] D. J. N. Limebeer and E. Kasenally, unpublished notes, 1987.

[24] D. J. N. Limebeer, E. M. Kasenally, I. Jaimouka and M. G. Safonov, "All Solutions to the Four Block General Distance Problem", *Proc. IEEE Conf. on Decision and Control*, Austin, TX, December 7-9, 1988.

[25] I. Postlethwaite and A. G. J. MacFarlane, *A Complex Variable Approach to the Analysis of Linear Multivariable Feedback Systems*. New York: Springer-Verlag, 1979.

[26] I. W. Sandberg, "On the $\mathbf{L}^2$-Boundedness of Solutions of Nonlinear Functional Equations," *Bell Syst. Tech*. J., vol. 43, pp.1581-1599, July, 1964.

[27] I. W. Sandberg, "A Frequency-Domain Condition for the Stability of Feedback Systems Containing a Single Time-Varying Nonlinear Element," *Bell Syst. Tech. J*., vol. 43, pp.1601-1608, July, 1964.

[28] M. G. *Safonov, Stability and Robustness of Multivariable Feedback Systems*. Cambridge, MA: MIT Press, 1980. Also, "Robustness and Stability Aspects of Multivariable Feedback System Design," PhD Thesis, MIT, Cambridge, MA, 1977.

[29] M. G. Safonov, A. J. Laub, and G. Hartmann, "Feedback Properties of Multivariable Systems: The Role and Use of Return Difference Matrix," *IEEE Trans. of Automat. Contr*., AC-26, 1, pp. 47-65, February 1981.

[30] M. G. Safonov and M. Athans, "A Multiloop Generalization of the Circle Criterion for Stability Margin Analysis," *IEEE Trans. on Automatic Control*, AC-26, 2, pp. 415-422, April 1981.

[31] M. G. Safonov, "Stability Margins of Diagonally Perturbed Multivariable Feedback Systems," *Proc. IEEE Conf. on Decision and Control*, San Diego, CA December 16-18, 1981.

[32] M. G. Safonov, "Stability Margins of Diagonally Perturbed Multivariable Feedback Systems," *IEE Proc*., 129 (Pt. D), 2, pp. 252-255, November 1982

[33] M. G. Safonov, "$\mathbf{L}^{\infty}$ Optimization vs. Stability Margin," *Proc. IEEE Conf. on Decision and Control*, San Antonio, TX, December 14-16, 1983.

[34] M. G. Safonov and R. Y. Chiang, "CACSD Using the State-Space $\mathbf{L}^{\infty}$ Theory — A Design Example", *Proc. IEEE Conf. on CACSD*, Washington D. C., Sep. 24-26, 1986, also *IEEE Trans. on Automat. Contr*., AC-33, No. 5, pp. 477-479, May 1988.

[35] M. G. Safonov, "Imaginary-Axis Zeros in Multivariable $\mathbf{H}^{\infty}$ Optimal Control", *Proc. NATO Advanced Research Workshop on Modeling, Robustness and Sensitivity Reduction in Control Systems*, Groningen, The Netherlands, Dec. 1-5, 1986.

[36] M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, `"Synthesis of Positive Real Multivariable Feedback Systems", *Int. J. Contro*l, vol. 45, no. 3, pp. 817-842, 1987.

[37] M. G. Safonov, R. Y. Chiang, and D. J. N. Limebeer, "Hankel Model Reduction without Balancing — A Descriptor Approach," *Proc. IEEE Conf. on Decision and Contro*l, Los Angeles, CA, Dec. 9-11, 1987.

[38] M. G. Safonov, R. Y. Chiang and H. Flashner, "$\mathbf{H}^{\infty}$ Control Synthesis for a Large Space Structure," *Proc. of American Contr. Conf.*, Atlanta, GA. June 15-17, 1988.

[39] M. G. Safonov and R. Y. Chiang, "A Schur Method for Balanced Model Reduction," *IEEE Trans. on Automat. Contr.*, vol. AC-34, no. 7, pp. 729-733, July 1989.

[40] M. G. Safonov and R. Y. Chiang, "Model Reduction for Robust Control: A Schur Relative-Error Method," *Proc. American Contr. Conf.*, June 15-17, 1988.

[41] M. G. Safonov and R. Y. Chiang, "CACSD Using the State-Space $\mathbf{L}^{\infty}$ Theory— A Design Example," *IEEE Trans. on Automatic Contro*l, AC-33, 5, pp. 477-479, 1988.

[42] M. G. Safonov, D. J. N. Limebeer and R. Y. Chiang, "Simplifying the $\mathbf{H}^{\infty}$ Theory via Loop Shifting, Matrix Pencil and Descriptor Concepts," *Int. J. Control,* 50, 6, pp. 2467-2488, 1989.

[43] M. G. Safonov, R. Y. Chiang and H. Flashner, "$\mathbf{H}^{\infty}$ Control Synthesis for a Large Space Structure," *AIAA J. Guidance, Control and Dynamics*, 14, 3, pp. 513-520, May/June 1991.

[44] M. G. Safonov and Peng-Hin Lee, "A Multiplier Method for Computing Real Multivariable Stability Margins," in *Proc. IFAC World Congress*, Sydney, Australia, July 1993.

[45] D. C. Youla, J. J. Bongiorno and H. A. Jabr, "Modern Wiener-Hopf Design of Optimal Controllers — Part I: The Single-Input-Output Case," *IEEE Trans. on Automatic Contro*l, AC-21, 1, pp. 3-13, 1976.

[46] D. C. Youla, H. A. Jabr and J. J. Bongiorno, "Modern Wiener-Hopf Design of Optimal Controllers — Part II: The Multivariable Case," *IEEE Trans. on Automatic Contro*l, AC-21, 3, pp. 319-338, 1976.

[47] W. Wang and M. G. Safonov, "A Tighter Relative-Error Bound for Balanced Stochastic Truncation," *Systems and Control Letters*, 14, pp. 307-317, 1990.

[48] W. Wang and M. G. Safonov, "Multiplicative-Error Bound for Balanced Stochastic Truncation," *IEEE Trans. on Automatic Control*, AC-37 no 8, pp. 1265-1267, August 1992.

[49] B. Wie and D. S. Bernstein, “A Benchmark Problem for Robust Controller Design,” *Proc. American Control Conf.*, San Diego, CA May 23-25, 1990; also Boston, MA, June 26-28, 1991.

[50] G. Zames, “On the Input-Output Stability of Time-Varying Nonlinear Feedback Systems,” Parts I and II, *IEEE Trans. on Automatic Contro*l, AC-11, 2 3, pp. 228-238 465-476, 1966.

2

Function Reference

Functions — By Category

This section contains detailed descriptions of the main functions in the Robust Control Toolbox. It begins with a listing of entries grouped by subject area and continues with the reference entries in alphabetical order. Information is also available through the on-line help facility.

For easy use, most functions have several default arguments. In a reference entry under Synopsis, we first list the function with all necessary input arguments and then with all possible input arguments. The functions can be used with any number of arguments between these extremes, the rule being that missing, trailing arguments are given default values, as defined in the manual.

As always in MATLAB, all output arguments of functions do not have to be specified, and are then not returned to the user. For functions with several output arguments, missing arguments are as a rule not computed in order to save time.

Optional System Data Structures

`branch`	Extract branches from a tree
`graft`	Add a branch to a tree
issystem	Identify a system variable
istree	Identify a tree variable
mksys	Build tree variable for system
tree	Build a tree variable
vrsys	Returns standard system variable names

Model Building

`augss, augtf`	Plant augmentation (weights on ‘e’, ‘u’ and ‘y’)
`interc`	General multivariable interconnected system

Model Conversions

`bilin`	Multivariable bilinear transform of frequency (s or z) — 7 options
`des2ss`	Convert descriptor system to state-space via SVD
`lftf`	Linear fractional transformation
`sectf`	Sector transformation
stabproj	Stable/antistable projection
slowfast	Slow/fast decomposition
`tfm2ss`	Convert transfer function matrix (MIMO) into state-space block-controller form

Utility

`aresolv`, `daresolv`	Generalized continuous./discrete Riccati solver
`riccond`, `driccond`	Continuous/discrete Riccati condition number
blkrsch	Block ordered real Schur form via `cschur`
cschur	Ordered complex Schur form via complex Givens rotation

Multivariable Bode Plots

cgloci	Continuous characteristic gain loci
dcgloci	Discrete characteristic gain loci
dsigma	Discrete singular-value Bode plot
`muopt`	Multiplier scaling
`osborne`	SSV upper bound via Osborne method
`perron, psv`	Perron eigenstructure SSV
sigma	Continuous singular-value Bode plot
`ssv`	structured singular value Bode plot

Factorization Techniques

iofc	Inner-outer factorization (column type)
iofr	Inner-outer factorization (row type)
sfl	Left spectral factorization
sfr	Right spectral factorization

Model Reduction Methods

balmr	Truncated balanced model reduction
`bstschml, bstschmr`	Relative error Schur model reduction
`imp2ss`	Impulse response to state-space realization
`obalreal`	Ordered balanced realization
`ohkapp, ohklmr`	Optimal Hankel minimum degree approximation
schmr	Schur model reduction

Robust Control Synthesis Methods

`h2lqg, dh2lqg`	Continuous/discrete H^2 synthesis
`hinf, dhinf, linf`	Continuous/discrete H^∞ synthesis
`hinfopt`	γ-iteration of H^∞ synthesis
`normhinf, normh2`	H^∞ and H^2 norm
`lqg`	LQG optimal control synthesis
`ltru, ltry`	LQG loop transfer recovery
`musyn, fitd, augd`	μ-synthesis
`youla`	Youla parametrization

Demonstration

`accdemo`	Spring-mass benchmark problem
`dintdemo`	H^∞ design for double-integrator plant
`hinfdemo`	H^∞ & H^2 design examples – fighter and large space structure
`ltrdemo`	LQG/LTR design examples – fighter

`mudemo, mudemo1`	μ-synthesis examples
`mrdemo`	Robust model reduction examples
`rctdemo`	Robust Control Toolbox demo – main menu

Functions — Alphabetical List

Purpose

Generalized continuous algebraic Riccati Solver.

Syntax

```
[p1,p2,lamp,perr,wellposed,p] = aresolv(a,q,r)
[p1,p2,lamp,perr,wellposed,p] = aresolv(a,q,r,Type)
```

Description

`aresolv` solves the continuous algebraic Riccati equation

$$A^T P + PA - PRP + Q = 0$$

where P=`p=p1/p2` is the solution for which the eigenvalues of $A - RP$ have negative real parts. This solution exists and is unique provided that the associated Hamiltonian matrix has no $j\omega$-axis eigenvalues; otherwise, the flag `wellposed` is set to the string value `'FALSE'`.

Two algorithms are available:

`Type = 'eigen'` — eigenvector approach

`Type = 'Schur'` — Schur vector approach

`Type 'eigen'` is selected by default, when the input argument `Type` is missing, provided that the associated Hamiltonian matrix is not defective (i.e., does not have a full set eigenvectors); otherwise the algorithm defaults to `Type 'Schur'`. If `Type = 'Schur'`, then the Schur approach is taken directly.

The residual and closed loop eigenvalues are also returned in variables `perr` and `lamp`.

In game theory and $\mathbf{H}^\infty$ applications, the weighting matrix R is usually indefinite. The program is coded to permit such cases.

Algorithm

The eigenvector approach (`Type = 'eigen'`) uses `reig` to find a real basis V_1 for the stable eigenspace of the Hamiltonian H [1]

$$H = \begin{bmatrix} A & -R \\ -Q & -A^T \end{bmatrix} = V \begin{bmatrix} -\Lambda & 0 \\ 0 & \Lambda \end{bmatrix} V^{-1}$$

where $\Lambda = diag(\lambda_1,\lambda_2,\ldots,\lambda_n)$ and

$$V = \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}$$

The first n columns (V_1) of the matrix V that form the stable eigenspace of H also provide the desired solution of ARE:

$$P = V_{21}V_{11}^{-1}$$

where V_{21} = `p2` and V_{11} = `p1` respectively. This algorithm requires about $120n^3$ flops.

The eigenvector approach can be numerically unstable when the Hamiltonian matrix is close to *defective*, as can occur some cases in which H is close to a matrix whose Jordan form has ones above its main diagonal. In this case, the matrix V_{11} will be ill-conditioned. However, the ill-conditioning of V_{11} is independent of the conditioning of the Riccati equation (ref. `riccond`, where six types of Riccati condition numbers are provided).

To circumvent the ill-conditioning problems associated with a defective Hamiltonian matrix, you can span the same stable eigenspace of Hamiltonian with Schur vectors [2]. In this approach the Hamiltonian H is orthogonally transformed into the ordered Schur form instead of modal form:

$$H = U\begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}U^T$$

where eigenvalues of T_{11} are stable and those of T_{22} are unstable.

The orthogonal matrix U can be partitioned as

$$U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}$$

where the first n column vectors span the same stable eigenspace as V_1 matrix. It is proved in [2] that the desired solution of ARE is $P = U_{21}U_{11}^{-1}$.

The Schur algorithm coded in `aresolv` first puts H in the ordered complex Schur form using `cschur`, then projects the complex basis into a real basis using the QR algorithm. The entire ordered Schur method takes about $75n^3$ flops, which is less than the eigenvector approach.

However, the Schur method itself can also become numerically unstable, if the norm of certain matrices involved in the Riccati equation are much smaller than the norms of others [3]. Our experience has been that more often than not, the eigenvector method performs more reliably than the Schur method, with the notable exception of the case of a defective Hamiltonian mentioned above. In difficult cases in which both eigenvector and Schur methods produce a large residual error, Newton methods may be used to refine the result.

If A is stable, and weighting matrix Q is zero, $P2 = 0$ and $P1 = I$ are returned without going through the eigenspace computations.

If the sixth output P is not included, `aresolv` ignores its computation as well. This can avoid some possible singular cases occurring in computing the "optimal" $\mathbf{H}^\infty$ controller. If $P1$ is singular or nearly so, the rank deficient $P1$ matrix will be used to form the $\mathbf{H}^\infty$ controller, but the null space of the descriptor is removed by `des2ss` at the final stage of the computation (see the reference pages `des2ss` and `hinf` for details).

Limitations

The Riccati equation is ill-posed if the Hamiltonian H has imaginary axis eigenvalues. In cases in which $Q, R \le 0$, it suffices for the system $((A, R, Q))$ to be both stabilizable and detectable to avoid imaginary axis eigenvalues; in this case the unique stabilizing Riccati solution will be positive semidefinite. In any case, the output variable `wellposed` is returned with the value `TRUE` or `FALSE` to reflect the well-posedness of the problem.

See Also

`are, cschur, daresolv, lqrc, reig, riccond`

References

[1] J. E. Potter, "Matrix Quadratic Solutions," *SIAM J. Appl. Mat*h., Vol. 14, pp. 496-501, 1966.

[2] A. J. Laub, "A Schur Method for Solving Algebraic Riccati Equations," I*EEE Trans. Autom. Control*, AC-24, pp. 913-921, 1979.

[3] P. Hr. Petkov, N. D. Christov, and M. M. Konstantinov, "On the Numerical Properties of Schur Approach for Solving the Matrix Riccati Equation," *Systems and Control Letters*, 9, pp. 197-201, 1987.

augd

Purpose

Augment the two-port plant with diagonal scaling.

Syntax

```
[AD,BD1,BD2,CD1,CD2,DD11,DD12,DD21,DD22] = ...
   augd(a,b1,b2,c1,c2,d11,d12,d21,d22,ad,bd,cd,dd)
[TSSD] = augd(TSS,ssd)
```

Description

Augd augments the original two-port plant $P(S)$ used in $\mathbf{H}^{\infty}$ synthesis with the diagonal scaling matrix $D(s)$ (see Figure 2-1, Augment with Diagonal Scaling). ssd is a state-space realization of the optimal diagonal scaling that computes the structured singular value upper bound of *P(s)*

The two-port plant $P(s)$ after scaling becomes

$$D(s)P(s)D(s)^{-1} = \begin{bmatrix} DP_{11}D^{-1} & DP_{12} \\ P_{21}D^{-1} & P_{22} \end{bmatrix}$$

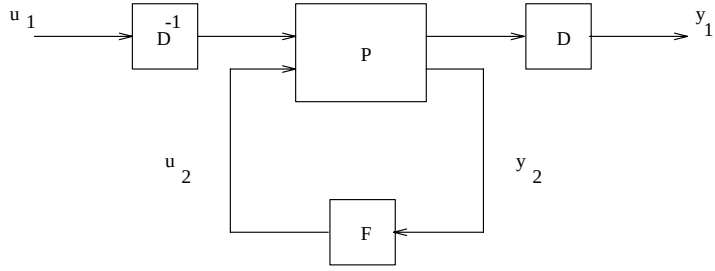


Figure 2-1: Augment with Diagonal Scaling

See Also

fitd, hinf, musyn, ssv

Purpose

State-space or transfer function plant augmentation for use in weighted mixed-sensitivity $\mathbf{H}^{\infty}$ and $\mathbf{H}^{2}$ design.

Syntax

```
[a,b1,b2,c1,c2,d11,d12,d21,d22] = ...
     augss(ag,bg, ,aw1,bw1, ,aw2,bw2, ,aw3,bw3, )
[a,b1,b2,c1,c2,d11,d12,d21,d22] = ...
     augss(ag,bg, ,aw1,bw1, ,aw2,bw2, ,aw3,bw3, ,w3poly)
[a,b1,b2,c1,c2,d11,d12,d21,d22] = ...
     augtf(ag,bg,cg,dg,w1,w2,w3)
[tss] = augss(ssg,ssw1,ssw2,ssw3,w3poly)
[tss] = augtf(ssg,w1,w2,w3)
[tss] = augss(ssg,ssw1,ssw2,ssw3)
```

`augss` computes a state-space model of an augmented plant $P(s)$ with weighting functions $W_1(s)$, $W_2(s)$, and $W_3(s)$ penalizing the error signal, control signal and output signal respectively (see block diagram) so that the closed-loop transfer function matrix is the weighted mixed sensitivity

$$T_{y_1 u_1} \triangleq \begin{bmatrix} W_1 S \\ W_2 R \\ W_3 T \end{bmatrix}$$

where S, R and T are given by

$$\begin{aligned} S &= (I + GF)^{-1} \\ R &= F(I + GF)^{-1} \\ T &= GF(I + GF)^{-1} \end{aligned} \quad \cdot$$

The matrices S(s) and *T(s)* are the sensitivity and complementary sensitivity, respectively

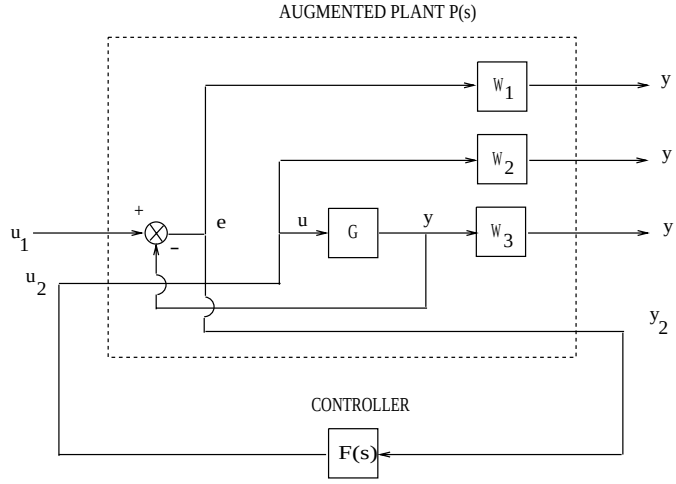


Figure 2-2: Plant Augmentation.

The transfer functions $G(s)$, $W_1(s)$ and $W_3(s)G(s)$ must be proper, i.e., bounded as $s \to \infty$. However, $W_3(s)$ may be improper. Input data of augss are the state-space system matrices of $G(s)$, $W_1(s)$, $W_2(s)$and $W_3(s)$:

$$ssg := \left[\begin{array}{c|c} A_g & B_g \\ \hline C_g & D_g \end{array}\right] = mksys(ag,\ bg,\ cg,\ dg);$$

$$ssw1 := \left[\begin{array}{c|c} A_{W_1} & B_{W_1} \\ \hline C_{W_1} & D_{W_1} \end{array}\right] = mksys(aw1, bw1, cw1, dw1);...etc.$$

The possibly improper transfer function matrix

$$W_3(s) = C_{W_3}(Is - A_{W_3})^{-1}B_{W_3} + D_{W_3} + P_n s^n + \ldots + P_1 s + P_0$$

is input to augss as

$$ssw3 = \left[\begin{array}{c|c} A_{W_3} & B_{W_3} \\ \hline C_{W_3} & D_{W_3} \end{array}\right] = mksys(aw3, bw3, cw3, dw3);$$

$$w3poly = [P_n,\ldots,P_1,P_0]$$

If one of the weighting functions is absent, e.g., $W_2(s) = 0$, then simply assign `ssw2 = [ ]`.

augtf does the plant augmentation as well, except the weighting functions W_1, W_2 and W_3 are input as *diagonal* transfer function matrices. The numerator and denominator of each diagonal entry of the transfer functions are entered as rows of the weighting matrix. For example, the weightings

$$W_1(s) = \begin{bmatrix} \dfrac{(s+100)}{(100s+1)} & 0 \\ 0 & \dfrac{(s+100)}{(100s+1)} \end{bmatrix};\ W_2 = [\ ];\ W_3(s) = \begin{bmatrix} \dfrac{s^3}{1000} & 0 \\ 0 & \dfrac{s^3(\tau s+1)}{1000} \end{bmatrix}$$

used in the $\mathbf{H}^\infty$fighter design presented in the *Tutorial* chapter are entered as

$$W1 = \begin{bmatrix} 1 & 100 \\ 100 & 1 \\ 1 & 100 \\ 100 & 1 \end{bmatrix};\ W2 = [\ ];\ W3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 \\ \text{tau} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix};$$

Algorithm

The augmented plant *P(s)* produced by augss is

$$P(s) = \left[\begin{array}{c|c} W_1 & -W_1G \\ 0 & W_2 \\ 0 & W_3G \\ \hline I & -G \end{array}\right]$$

with state-space realization

$$P(s) := \left[\begin{array}{cc|c} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ \hline C_2 & D_{21} & D_{22} \end{array}\right]$$

$$= \left[\begin{array}{ccccc|c} A_G & 0 & 0 & 0 & 0 & B_G \\ -B_{W_1}C_G & A_{W_1} & 0 & 0 & B_{W_1} & -B_{W_1}D_G \\ 0 & 0 & A_{W_2} & 0 & 0 & B_{W_2} \\ B_{W_3}C_G & 0 & 0 & A_{W_3} & 0 & B_{W_3}D_G \\ \hline -D_{W_1}D_G & C_{W_1} & 0 & 0 & D_{W_1} & -D_{W_1}D_G \\ 0 & 0 & C_{W_2} & 0 & 0 & D_{W_2} \\ \tilde{C}_G + D_{W_3}C_G & 0 & 0 & C_{W_3} & 0 & \tilde{D}_G + D_{W_3}D_G \\ \hline -C_G & 0 & 0 & 0 & I & -D_G \end{array}\right]$$

where

$$\tilde{C}_G = P_0 D_G + P_1 C_G A_G + \ldots + P_n C_G A_G^n B_G$$

$$\tilde{D}_G = P_0 D_G + P_1 C_G B_G + \ldots + P_n C_G A_G^{n-1} B_G$$

`augtf` calls `augss` internally after a series of transfer function to state-space transformations on *W*1, *W*2, and *W*3.

Limitations

Note that if the augmented plant is to be used for $\mathbf{H}^\infty$ synthesis via `hinf` or `linf`, then it is essential that the weights W_1, W_2, and W_3 be selected so that the augmented plant has a "D_{12}" matrix of full column rank. An easy way to ensure that this is the case is to choose a $W_2(s)$ with an invertible "*D*-matrix", e.g., $W_2(s) = \varepsilon I$ where ε is any non-zero number.

See Also

h2lqg, hinf, hinfdemo, linf, linfdemo, dh2lqg, dhinf

Purpose

Balanced model reduction via truncated and Schur methods.

Syntax

```
[am,bm,cm,dm,totbnd,svh] = balmr(a,b,c,d,Type)
[am,bm,cm,dm,totbnd,svh] = balmr(a,b,c,d,Type,aug)
[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type)
[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type,aug)
[ssm,totbnd,svh] = balmr(ss,Type,aug)
[ssm,totbnd,svh] = schmr(ss,Type,aug)
```

Description

Both `balmr` and `schmr` compute a k^{th} order reduced model

$$G_m(s) = C_m(Is + A_m)^{-1}B_m + D_m$$

of a possibly non-minimal and not necessarily stable, n^{th} order system

$$G(s) = C(Is - A)^{-1}B + D$$

such that

$$\left\| G(j\omega) - G_m(j\omega) \right\|_\infty \le totbnd,$$

$$totbnd = 2 \sum_{i=k+1}^{n} svh(i)$$

The n-vector `svh` contains the Hankel singular values of the stable and antistable projections of $G(j\omega)$, i.e., the square-roots of eigenvalues of their reachability and observability grammians.

Three options are provided for both functions:

1. `Type = 1, aug = k`, size of reduced order model.
2. `Type = 2, aug = tol`, find a k^{th} order reduced model such that the total error `totbnd` is less than `tol`.
3. `Type = 3`, display `svh` and prompt for `k`. In this case, there is no need to assign a value for `aug`.

Both `balmr` and `schmr` produce state-space realizations of the same transfer function G_m. The realization (A_m, B_m, C_m, D_m) produced by `balmr` is balanced (see `balreal` or `obalreal`), whereas that produced by `schmr` algorithm is not. The `schmr` algorithm is numerically *more robust* than `balmr` and is thus to be preferred when the reduced model need not be balanced.

Algorithm

`Balmr` and `schmr` employ the algorithms described in [3] for implementing the balanced truncation model reduction procedure of [2], but unlike the original Moore algorithm, these algorithms bypass the numerically delicate preliminary step of computing a balanced minimal realization of $G(s)$.

Unstable systems are handled via the M-function `stabproj` which splits $G(s)$ into the sum of stable and antistable parts.

See Also

balreal, mrdemo, obalreal, ohklmr, bstschmr, bstschml

[1] A. J. Laub, M. T. Heath, C. C. Page, and R. C. Ward, "Computation of balancing transformations and other applications of simultaneous diagonalization algorithms," IEEE Trans. on Automat. Contr., AC-32, pp. 115-122, 1987.

[2] B. C. Moore, "Principal component analysis in linear systems: controllability, observability, and model reduction," IEEE Trans. on Automat. Contr., AC-26, pp. 17-31, 1981.

[3] M. G. Safonov and R. Y. Chiang, "A Schur Method for Balanced Model Reduction," *IEEE Trans. on Automat. Contr.*, vol. AC-34, no. 7, pp. 729-733, July 1989.

Purpose

Multivariable bilinear transform of frequency (s or z).

Syntax

```
[ab,bb,cb,db] = bilin(a,b,c,d,ver,Type,aug)
[ssb] = bilin(ss,ver,Type,aug)
```

Description

`Bilin` computes the effect on a system of the frequency-variable substitution,

$$s = \frac{\alpha z + \delta}{\gamma z + \beta}$$

The variable `Ver` denotes the transformation direction:

`Ver= 1`, forward transform $(s \to z)$.

`Ver=-1`, inverse transform $(z \to s)$.

This transformation maps lines and circles to circles and lines in the complex plane. People often use this transformation to do sampled-data control system design [1] or, in general, to do shifting of $j\omega$ modes [2], [3], [4].

`Bilin` computes several state-space bilinear transformations such as Tustin, prewarped Tustin, etc., based on the `Type` you select:

1 `Type = 'Tustin'`, Tustin transform:

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

`aug` = T, the sampling period.

2 `Type = 'P_Tust'`, prewarped Tustin:

$$s = \frac{\omega_0}{\tan((\omega_0 T)/2)}\left(\frac{z-1}{z+1}\right)$$

`aug = [`T ω_0`]`, ω_0 is the prewarped frequency.

3 `Type = 'BwdRec'`, backward rectangular:

$$s = \frac{z-1}{Tz}$$

`aug` = T, the sampling period.

4 `Type = 'FwdRec'`, forward rectangular:

$$s = \frac{z-1}{T}$$

`aug` = T, the sampling period.

5 `Type = 'S_Tust'`, shifted Tustin:

$$s = \frac{2}{T}\left(\frac{z-1}{\frac{z}{h}+1}\right)$$

`aug` = $[T\ h]$, is the "shift" coefficient.

6 `Type = 'S_ftjw'`, shifted $j\omega$-axis bilinear:

$$s = \frac{z+p_1}{1+z/p_2}$$

`aug` = $[p_2\ p_1]$.

7 `Type = 'G_Bilin'`, general bilinear:

$$s = \frac{\alpha z+\delta}{\gamma z+\beta}$$

`aug` = $[\alpha\ \beta\ \gamma\ \delta]$.

Examples

Consider the following continuous-time plant (sampled at 20 Hz)

$$A = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix},\ B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Following is an example of four common “continuous to discrete” `bilin` transformations for the sampled plant:

```
ss = mksys(a,b,c,d); %use system data structure
[sst] = bilin(ss,1,'Tustin',0.05);
[ssp] = bilin(ss,1,'P_Tust',[0.05 40]);
[ssb] = bilin(ss,1,'BwdRec',0.05);
[ssf] = bilin(ss,1,'FwdRec',0.05);
w = logspace(-2,3,100) %frequency
svt = dsigma(sst,0.05,w);
svp = dsigma(ssp,0.05,w);
svb = dsigma(ssb,0.05,w);
svf = dsigma(ssf,0.05,w);
```

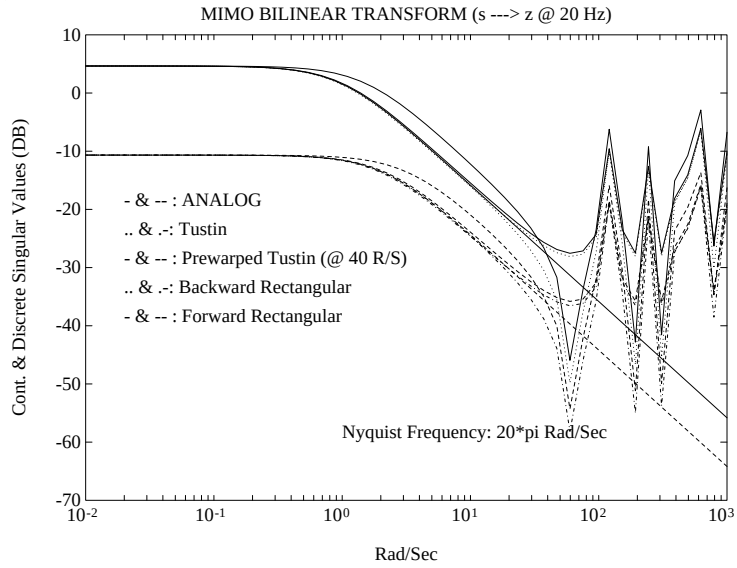


Figure 2-3: Comparison of 4 Bilinear Transforms.

you can generate the continuous and discrete singular value Bode plots as shown in the Figure 2-3, Comparison of 4 Bilinear Transforms..

Note that the Nyquist frequency is at 20π rad/sec.

Algorithm

`bilin` employs the state-space formulae in [3]:

$$\left[\begin{array}{c|c} A_b & B_b \\ \hline C_b & D_b \end{array}\right] = \left[\begin{array}{c|c} (\beta A - \delta I)(\alpha I + \gamma A)^{-1} & (\alpha\beta - \gamma\delta)(\alpha I - \gamma A)^{-1} B \\ \hline C(\alpha I - \gamma A)^{-1} & D + \gamma C(\alpha I - \gamma A)^{-1} B \end{array}\right]$$

See Also

lftf, sectf

[1] G. F. Franklin and J. D. Powell, *Digital Control of Dynamics System*, Addison-Wesley, 1980.

[2]M. G. Safonov, R. Y. Chiang and H. Flashner, "**H**∞ Control Synthesis for a Large Space Structure," *AIAA J. Guidance, Control and Dynamics*, 14, 3, pp. 513-520, May/June 1991.

[3] M. G. Safonov, "Imaginary-Axis Zeros in Multivariable **H**∞Optimal Control", in R. F. Curtain (editor), *Modelling, Robustness and Sensitivity Reduction in Control Systems*, pp. 71-81, Springer-Verlag, Berlin, 1987. *Proc. NATO Advanced Research Workshop on Modeling, Robustness and Sensitivity Reduction in Control Systems*, Groningen, The Netherlands, Dec. 1-5, 1986.

[4] R. Y. Chiang and M. G. Safonov, "**H**∞ Synthesis using a Bilinear Pole Shifting Transform," *AIAA, J. Guidance, Control and Dynamics*, vol. 15, no. 5, pp. 1111-1117, September–October 1992.

Purpose

Block ordered real Schur form.

Ordered complex Schur form via complex Givens rotation.

Syntax

```
[v,t,m] = blkrsch(a,Type,cut)
[v,t,m,swap] = cschur(a,Type)
```

Description

Cschur computes a unitary similarity transformation V and a complex upper triangular matrix T for a real or complex matrix A such that

$$V^T A V = T = \begin{bmatrix} T_1 & T_{12} \\ 0 & T_2 \end{bmatrix}$$

where T has the eigenvalues $\lambda_i(A)$ ordered on the diagonal according to the value of the variable `Type`:

1. `Type = 1` — $Re(\lambda_i(T_1)) < 0,\ Re(\lambda_i(T_2)) > 0$.
2. `Type = 2` — $Re(\lambda_i(T_1)) > 0,\ Re(\lambda_i(T_2)) < 0$.
3. `Type = 3` — eigenvalue real parts in descending order.
4. `Type = 4` — eigenvalue real parts in ascending order.
5. `Type = 5` — modulus of eigenvalues in descending order.
6. `Type = 6` — modulus of eigenvalues in ascending order.

Variable `swap` records the number of Givens rotations swaps it takes and variable `m` returns the number of "stable" eigenvalues of A (see `rsf2csf` or `cgivens`).

`Blkrsch` computes a block ordered real Schur form such that the resulting T matrix has four blocks

$$V^T A V = T = \begin{bmatrix} B_1 & B_{12} \\ 0 & B_2 \end{bmatrix}$$

The input variable `cut` is the dimension of the square block B_1. If `Type` is 1, `cut` is automatically set to `m` — the number of eigenvalues of A with negative real parts.

Six options are available:

1. `Type = 1` — $Re(\lambda_i(B_1)) < 0,\ Re(\lambda_i(B_2)) > 0$.
2. `Type = 2` — $Re(\lambda_i(B_1)) > 0,\ Re(\lambda_i(B_2)) < 0$.
3. `Type = 3` — $Re(\lambda_i(B_1)) > 0,\ Re(\lambda_i(B_2))$.
4. `Type = 4` — $Re(\lambda_i(B_1)) < 0,\ Re(\lambda_i(B_2))$.
5. `Type = 5` — $|\lambda_i(B_1)| > |\lambda_i(B_2)|$.
6. `Type = 6` — $|\lambda_i(B_1)| < |\lambda_i(B_2)|$

Algorithm

`blkrsch` and `cschur`, are M-files in the *Robust Control Toolbox*. `cschur` uses `schur`, `rsf2csf` and the complex Givens rotation [1] to iteratively swap two adjacent diagonal terms according the style you select. `blkrsch` projects the resulting complex subspace onto the real.

Limitations

For `blkrsch`, the matrix A must have zero imaginary part.

See Also

cgivens, rsf2csf, schur

References

[1] Golub G. H. and C. F. Van Loan, *Matrix Computations*. Baltimore: Johns Hopkins University Press, 1983.

Purpose

This function recovers the matrices packed in a `mksys` or `tree variable` selectively.

Syntax

```
[b1,b2, ,bn] = branch(tr,PATH1,PATH2, ,PATHN)
```

Description

Branch returns N sub-branches of a tree variable `tr`, if `nargin = 1`, the root branches are returned in sequence by numerical index; otherwise, the branches returned are determined by the paths `PATH1`, `PATH2`, , `PATHN`. Each path is normally a string of the form

```
PATH = '/name1/name2/ /namen';
```

where `name1`, `name2`, et cetera are the string names of the branches that define the path from the tree root to the sub-branch of interest.

Alternatively, you may substitute for any `PATH` a row vector containing the integer indices of the branches that define the `PATH`. For example, if `S = tree('a,b,c','foo',[49 50],'bar')`, then `branch(S,'c')` and `branch(S,3)` both return the value "`bar`".

Examples

See reference pages for `tree` and `mksys`.

See Also

tree, mksys, graft, istree, issystem, vrsys

bstschml, bstschmr

Purpose

Relative error model reduction via Schur balanced stochastic truncation.

Syntax

```
[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type)
[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type,no)
[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type,no,info)
[ssred,aug,svh] = bstschmr(SS,Type)
[ssred,aug,svh] = bstschmr(SS,Type,no)
[ssred,aug,svh] = bstschmr(SS,Type,no,info)
```

The same syntax applies to `"bstschml"`

Description

Given an n^{th} order stable plant

$$G(s) := \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

`bstschmr` computes a k^{th} order reduced model

$$\overline{G}(s) := \left[\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline C & D \end{array}\right]$$

such that the *multiplicative error* between $G(s)$ and $\overline{G}(s)$ is bounded as follows [9]

$$\left\| \overline{G}^{-1}(G - \overline{G}) \right\|_\infty \le 2 \sum_{i=k+1}^{n} \frac{\sigma_i}{1-\sigma_i}$$

and the *relative error* of $G(s)$ and $\overline{G}(s)$ also enjoys the same error bound [6]:

$$\left\| G^{-1}(G - \overline{G}) \right\|_\infty \le 2 \sum_{i=k+1}^{n} \frac{\sigma_i}{1-\sigma_i}$$

where σ_i are the Hankel singular values of the *all-pass* phase matrix $(W^*(s))^{-1}G(s)$, and

$$W(s) := \left[\begin{array}{c|c} A_W & B_W \\ \hline C_W & D_W \end{array}\right]$$

is a *minimum phase* left spectral factor of $\Phi = G(s)G^T(-s) = W^T(-s)W(s)$.

Three options are provided:

1. `Type = 1, no =` k, size of reduced order model.
2. `Type = 2, no =` tol, a relative tolerance band in db such that the k^{th} order reduced model $\overline{G}(j\omega) \subset G(j\omega) \pm tol$ for all frequencies.
3. `Type = 3`, display `svh` and prompt for `k`. In this case, no need to assign a value for `no`.

Variable `aug(1,1)` returns the number of states that have been removed, while `aug(1,2)` returns the relative error bound.

`Bstschml` solves the "dual" problem of `bstschmr` with the same error bound

$$\text{Relative Error: } \left\|(G-\overline{G})G^{-1}\right\|_\infty \le \quad 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$$

$$\text{Multiplicative Error: } \left\|(G-\overline{G})\overline{G}^{-1}\right\|_\infty \le \quad 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$$

For a given discrete $G(z)$, you can still apply the functions `bstschmr` and `bstschml` via bilinear transform `bilin` to get a reduced order $\overline{G}(z)$ [8], i.e.,

$$G(z) \xrightarrow{bilin} G(w) \xrightarrow{bstschmr} \overline{G}(w) \xrightarrow{bilin} \overline{G}(z)$$

The resultant reduced order model $\overline{G}(z)$ enjoys the same relative and multiplicative error bound as the continuous case [7, 9]. A direct discrete BST/REM algorithm without using the bilinear transform is not available at this time.

Algorithm

`bstschmr` implements the BST model reduction algorithm of [1], but using the Schur method of [4] to bypass the numerical sensitive balancing step. The BST relative error bound is due to Wang and Safonov [6, 9]. The complete algorithm of `bstschml` and `bstschmr` is presented in [5].

`bstschmr` computes the reachability grammian P of $G(s)$ and the observability grammian Q of $W(s)$ via the equations

$$AP + PA^T + BB^T = 0$$

$$B_W = PC^T + BD^T$$

$$QA + A^TQ + (QB_W - C^T)(-DD^T)^{-1}(QB_W - C^T)^T = 0$$

A Schur algorithm, analogous to that in [4], is then applied to the product of the grammians PQ to reliably compute the BST reduced model $\overline{G}(s)$. Note that the particular realization of $\overline{G}(s)$, viz. (`Ared`, `Bred`, `Cred`, `Dred`), will not in general be stochastically balanced.

The BST model reduction procedure produces similar relative error bounds and is closely related to the optimal Hankel norm phase matching model results of [2] and [3].

`Bstschml` is completely analogous and simply applies the "dual" BST/REM theory. It can also be called by `bstschmr` with an additional input variable `info= "left"`.

Limitations

The BST model reduction theory requires that D be full rank, for otherwise the Riccati solver fails. For any problem with strictly proper plant, you can shift the $j\omega$-axis via `bilin` such that BST/REM approximation can be achieved up to a particular frequency range of interest. Alternatively, you can attach a small but full rank D matrix to the original problem but remove the matrix of the reduced order model afterwards. As long as the size of D matrix is insignificant inside the control bandwidth, the reduced order model should be fairly close to the true model.

See Also

balmr, mrdemo, ohklmr, schmr

References

[1] U. B. Desai and D. Pal, "A Transformation Approach to Stochastic Model Reduction," *IEEE Trans. on Automat. Contr.*, AC-29, 12, 1984.

[2] K. Glover, "Multiplicative Approximation of Linear Systems with L^∞ Error Bounds," *Proc. American Contr. Conf.*, Seattle, WA, June 18-20, 1986.

[3] E. A. Jonckheere and R. Li, "L$^\infty$ Error Bound for Phase Matching Approximation — The One-Step-At-A-Time Hankel Norm Model Reduction," *Int. J. Control*, Vol. 46, no. 4, pp. 1343-1354, 1987.

[4] M. G. Safonov and R. Y. Chiang, "A Schur Method for Balanced Model Reduction," *IEEE Trans. on Automat. Contr.*, vol. AC-34, no. 7, pp. 729-733, July 1989.

[5] M. G. Safonov and R. Y. Chiang, "Model Reduction for Robust Control: A Schur Relative-Error Method," *Proc. American Contr. Conf.*, June 15-17, 1988.

[6] W. Wang and M. G. Safonov, "A Tighter Relative-Error Bound for Balanced Stochastic Truncation," *Systems and Control Letters*, 14, No. 4, pp. 307-317, 1990.

[7] W. Wang and M. G. Safonov, "A Relative Error Bound for Discrete Balanced Stochastic Truncation," *Int. J. of Contro*l, Vol. 54, No. 3, 1990.

[8] W. Wang and M. G. Safonov, "Comparison between Continuous and Discrete Balanced Stochastic Truncation Model Reduction," *Proc. of Contr. and Decision Conf.*, Honolulu, Hawaii, 1990.

[9] W. Wang and M. G. Safonov, "Multiplicative-Error Bound for Balanced Stochastic Truncation Model Reduction," *IEEE Trans. on Automat. Contr,* vol. AC-37, no. 8, pp1265-1267, August 1992.

Purpose

Continuous characteristic gain loci frequency response.

Discrete characteristic gain loci frequency response.

Syntax

```
[cg,ph,w] = (d)cgloci(a,b,c,d(,Ts))
[cg,ph,w] = (d)cgloci(a,b,c,d(,Ts),'inv')
[cg,ph,w] = (d)cgloci(a,b,c,d(,Ts),w)
[cg,ph,w] = (d)cgloci(a,b,c,d(,Ts),w,'inv')
[cg,ph,w] = (d)cgloci(ss, )
```

Description

`cgloci` computes the matrices `cg` and `ph` containing the characteristic gain and phase of the frequency response matrix $G(j\omega) = C(j\omega I - A)^{-1}B + D$ as a function of frequency, ω. The characteristic gain `cg` and phase `ph` vectors are defined as

$$cg := \text{abs}(\text{eig}(G(j\omega)))$$

$$ph := \left(\frac{180^{\circ}}{\pi}\right)\angle\text{eig}(G(j\omega))$$

When invoked without lefthand arguments, `cgloci` produces a characteristic gain and phase Bode plot on the screen. The frequency range is chosen automatically and incorporates more points where the plot is changing rapidly.

`cgloci(a,b,c,d,'inv')` plots the characteristic gain and phase loci of the inverse system $G(s)^{-1}$.

When the frequency vector `w` is supplied, the vector `w` specifies the frequencies in radians/sec at which the char. loci will be computed. See `logspace` to generate frequency vectors that are equally spaced logarithmically in frequency.

When invoked with lefthand arguments,

```
[ga,ph,w] = cgloci( )
```

returns the gain, phase matrices and the frequency point in the vector `w`.

`dcgloci` computes the discrete version of the characteristic loci by replacing $G(j\omega)$ by $G(e^{j\omega T_s})$. The variable `Ts` is the sampling period.

Cautionary Note

Nyquist loci and Bode plots of characteristic gain loci can produce a misleadingly optimistic picture of the stability margins multiloop systems. For this reason, it is usually a good idea when using `cgloci` to also examine the singular value Bode plots in assessing stability robustness. See `sigma` and `dsigma`.

Examples

Let's consider a 2 by 2 transfer function matrix of the plant [1]

$$G(s) = \begin{bmatrix} \frac{-47s+2}{(s+1)(s+2)} & \frac{56s}{(s+1)(s+2)} \\ \frac{-42s}{(s+1)(s+2)} & \frac{50s+2}{(s+1)(s+2)} \end{bmatrix}$$

with model decomposition

$$G(s) = X\Lambda(s)X^{-1} = \begin{bmatrix} 7 & -8 \\ -6 & 7 \end{bmatrix} \begin{bmatrix} \frac{1}{s+1} & 0 \\ 0 & \frac{2}{s+2} \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 6 & 7 \end{bmatrix}$$

The controller is a diagonal constant gain matrix

$$K = \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The characteristic gain loci containing in $\Lambda(s)$ *seem* to imply that the system has infinity gain margin and ± 180 degree phase margin in each of the feedback loop. However, if you slightly perturb the gains K_1 and K_2 to K_1 + 0.13 and K_2 – 0.12 simultaneously, the system becomes unstable!

On the other hand, the singular value Bode plot of the complementary sensitivity $T(s) = G(s)(I + G(s))^{-1}$ (MATLAB command: `sigma`) easily predicts the system robustness (see *Tutorial*). See Figure 2-4, Singular Value vs. Characteristic Gain Loci..

The resonance peak of the maximum singular value (≈ 16.2695) predicts that the multiplicative uncertainty can only be as large as $\frac{1}{16.2695} = 6.15\%$ before instability occurs.

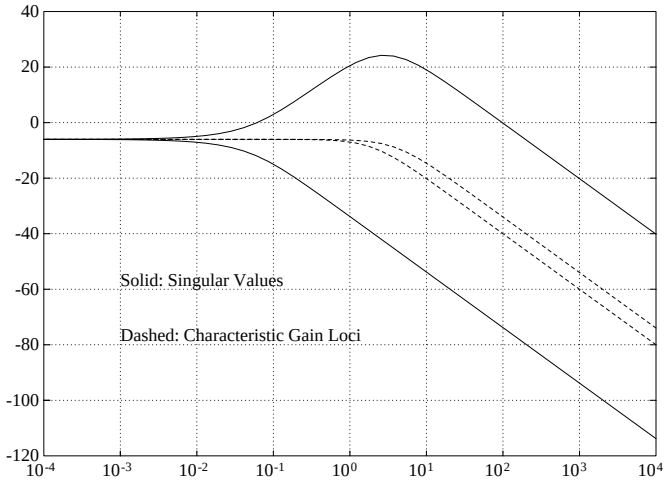


Figure 2-4: Singular Value vs. Characteristic Gain Loci.

Alternatively, you can compute *guaranteed stability margins* using the formulae given in *Singular-Value Loop-Sharing: The Mixed-Sensitivity Approach* [2]

$$\text{Guaranteed GM} = 1 \pm \frac{1}{\|T\|_\infty} = 0.94 \;\; to \;\; 1.062$$

$$\text{Guaranteed PM} = \pm 2 \sin^{-1}\left(\frac{1}{2\|T\|_\infty}\right) \pm 3.52 \;\; \text{deg}$$

which clearly predict the poor robustness of the system. These guaranteed stability margins provide a tolerance such that you can vary both gain and phase simultaneously in all the feedback loops.

See Also

bode, dsigma, dbode, logspace, sigma

[1] J. C. Doyle, “Robustness of Multiloop Linear Feedback Systems,” *Proc. IEEE Conf. on Decision and Control*, San Diego, CA, Jan. 10-12, 1979

[2] N. A. Lehtomaki, N. R. Sandell, Jr., and M. Athans, “Robustness Results in Linear-Quadratic Gaussian Based Multivariable Control Designs,” *IEEE Trans. on Automat. Contr.*, vol. AC-26, No. 1, pp. 75-92, Feb. 1981.

[3] I. Postlethwaite and A. G. J. MacFarlane, *A Complex Variable Approach to the Analysis of Linear Multivariable Feedback Systems*, Springer-Verlag, 1979.

daresolv

Purpose

Generalized discrete algebraic Riccati solver.

Syntax

```
[p1,p2,lamp,perr,wellposed,p] = daresolv(a,b,q,r)
[p1,p2,lamp,perr,wellposed,p] = daresolv(a,b,q,r,Type)
```

Description

Daresolv solves the discrete algebraic Riccati equation

$$A^TPA - P - A^TPB(R + B^TPB)^{-1}B^TPA + Q = 0$$

where $P = P2/P1$ is the solution for which the eigenvalues of $A - RP$ are inside the unit disk. This solution exists and is unique provided that the associated discrete Hamiltonian matrix

$$H = \begin{bmatrix} A + BR^{-1}B^TA^{-T}Q & -BR^{-1}B^TA^{-T} \\ -A^{-T}Q & A^{-T} \end{bmatrix}$$

has no eigenvalues on the unit circle; otherwise, the flag `wellposed` is set to the string value `'FALSE'`.

The input variables and output variables are otherwise identical to the continuous time case—see `aresolv`. If `Type = 1` the eigenvector approach is used. If `Type = 2` the Schur vector approach is used. The eigenvector approach is selected by default when no `Type` is specified, unless the Hamiltonian matrix is defective in which case the algorithm defaults to the Schur approach. The residual and closed loop eigenvalues are returned in variables `perr` and `lamp`.

Algorithm

The algorithm is essentially the same as that employed for the continuous version of this function, save for the difference in the discrete Hamiltonian. The matrices *P1* and *P2* are computed such that the columns of

$$\begin{bmatrix} P1 \\ P2 \end{bmatrix}$$

form a basis for the stable eigenspace of the discrete Hamiltonian, i.e., the space spanned by the eigenvectors corresponding to eigenvalues in the unit disk. See `aresolv`.

Limitations

The Riccati equation is ill-posed if the Hamiltonian H has eigenvalues on the unit circle. In cases in which $Q, R \geq 0$, it suffices for the system (A,R,Q) to be both stabilizable and detectable to avoid eigenvalues on the unit circle; in this case, the unique stabilizing Riccati solution will be positive semidefinite.

See Also

are, cschur, aresolv, lqrc, reig, driccond

References

1 A. J. Laub, "A Schur Method for Solving Algebraic Riccati Equations," *IEEE Trans. Autom. Control*, AC-24, pp. 913-921, 1979.

Purpose

Convert descriptor system into SVD state-space form.

Syntax

```
[aa,bb,cc,dd] = des2ss(a,b,c,d,E,k)
[ss1] = des2ss(ss,E,k)
```

Description

`des2ss` converts a descriptor system [1]

$$G_1 := \left[\begin{array}{c|c} -Es + A & B \\ \hline C & D \end{array}\right]$$

into state-space form $G_2(s)$:

$$\left[\begin{array}{c|c} -Is + \Sigma^{-1/2}(A_{11} - A_{12}A_{22}^{-1}A_{21})\Sigma^{-1/2} & \Sigma^{-1/2}(B_1 - A_{12}A_{22}^{-1}B_2) \\ \hline (C_1 - C_2A_{22}^{-1}A_{21})\Sigma^{-1/2} & D - C_2A_{22}^{-1}B_2 \end{array}\right]$$

via the *singular value decomposition* (SVD) of the matrix E (E may be singular with n – *rank(E)* = < n)

$$E = U\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}V^T = [U_1 \;\; U_2]\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}V_1^T\\ V_2^T\end{bmatrix}$$

where

$$k := dimension\ of\ null\ space\ of\ the\ matrix\ E$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} A [V_1 \ \ V_2]$$

$$\begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} B$$

$$[C_1 \ \ C_2] = C[V_1 \ \ V_2]$$

Algorithm

Converting descriptor system into SVD coordinates can be carried out easily by applying a series of strict system equivalence (s.s.e.) transformations to the Rosenbrock system matrix [2].

See Also

ohklmr, hinf, linf

References

[1] D. G. Luenberger, “Dynamic Equations in Descriptor Form”, *IEEE Trans. on Automat. Contr.*, AC-22, No. 3, Jun. 1977.

[2] M. G. Safonov, R. Y. Chiang, and D. J. N. Limebeer, “Hankel Model Reduction without Balancing -- A Descriptor Approach,” *Proc. IEEE Conf. on Decision and Control*, Los Angeles, CA, Dec. 9-11, 1987.

driccond

Purpose

Condition numbers of discrete algebraic Riccati equation.

Syntax

```
[tot] = driccond(a,b,q,r,p1,p2)
```

Description

`driccond` provides the condition numbers of discrete Riccati equation

$$A^T PA - P + Q - A^T PB(R + B^T PB)^{-1} B^T PA = 0$$

where $P = P2/P1$ is the positive definite solution of ARE, and [$P2$; $P1$] spans the stable eigenspace of the Hamiltonian

$$H = \begin{bmatrix} A + SA^{-T}Q & -SA^{-T} \\ -A^T Q & A^{-T} \end{bmatrix}$$

where $S = BR^{-1}B^T$.

Several measurements are provided:

1. Frobenius norm of matrices A, Q, and B$R^{-1}B^T$ (`norA`, `norQ`, `norRc`).
2. condition number of R (`conR`).
3. condition number of $P1$ (`conP1`).
4. Byers' condition number (`conBey`) [1].
5. residual of Riccati equation (`res`).

The output variable `tot` puts the above measurements in a column vector

```
tot= [norA,norQ,norRc,conR,conP1,conBey,res]'
```

For an ill-conditioned problem, one or more of the above measurements could become large. Together, they should give a general information of the Riccati problem.m

Algorithm

Byers' Riccati condition number is computed as [1]

$$\text{conBey} = \frac{\|Q_c\|_F + 2\|A_c\|_F^2\|P\|_F + \|A\|_F^2\|S\|_F\|P\|_F}{\|P\|_F \ \text{sep}(A'_{cl},\ A_{cl})}$$

where $A_{cl} = (I_n + SP)^{-1} A$ and

$$\text{sep}(A_{cl}^T,\ A_{cl}) = \min_i \ \sigma_i[A_c^T \otimes A_c^T - I_{n^2}]$$

See Also

are, aresolv, daresolv, riccond

[1] R. Byers, "Hamiltonian and Symplectic Algorithms for the Algebraic Riccati Equation," Ph.D. dissertation, Dept. of Comp. Sci., Cornell University, Ithaca, NY, 1983.

dsigma, sigma

Purpose

Discrete singular value frequency response.

Continuous singular value frequency response.

Syntax

```
[sv,w] = (d)sigma(a,b,c,d(,Ts))
[sv,w] = (d)sigma(a,b,c,d(,Ts),'inv')
[sv,w] = (d)sigma(a,b,c,d(,Ts),w)
[sv,w] = (d)sigma(a,b,c,d(,Ts),w,'inv')
[sv,w] = (d)sigma(ss,..)
```

Description

`sigma` computes the singular values of the complex matrix $C(j\omega I - A)^{-1}B + D$ as a function of frequency, ω. The singular values are an extension of the Bode magnitude response for MIMO systems. When invoked without lefthand arguments, `sigma` produces a singular value Bode plot on the screen. The frequency range is chosen automatically and incorporates more points where the plot is changing rapidly.

For square systems, `sigma(A, B, C, D,' inv')` plots the singular values of the inverse complex matrix:

$$G^{-1}(\omega) = D^{-1}C[j\omega I - (A - BD^{-1}C)]^{-1}BD^{-1} + D^{-1}$$

When supplied by the user, the vector `w` specifies the frequencies in radians/sec at which the singular value Bode plot will be calculated. See `logspace` to generate frequency vectors that are equally spaced logarithmically in frequency.

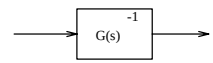
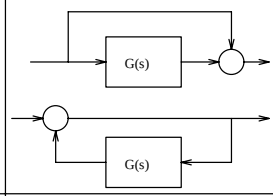
When invoked with lefthand arguments,

```
[sv,w] = sigma(A,B,C,D,..)
```

returns the magnitude of the singular values in matrix `sv` and the frequency points in `w`. No plot is drawn on the screen. Each column of the matrix `sv` contains the singular values, in decreasing order, for the corresponding frequency point in the vector `w`.

`Dsigma` computes the discrete version of the singular value Bode plot by substituting $G(e^{j\omega T_s})$ for $G(j\omega)$. The variable `Ts` is the sampling period.

For robustness analysis, the singular values of particular transfer matrices are analyzed. The table below shows which input-output combinations and associated MATLAB commands achieve the desired transfer matrices:

TF Matrix	Block Diagram	MATLAB Commands
G(jw)		sigma(a,b,c,d)
$G(jw)^{-1}$		sigma(a,b,c,d,'inv')
(I+G(jw))		[a,b,c,d] = parallel(a,b,c,d,[],[],[],eye(d)) sigma(a,b,c,d) [a,b,c,d] = feedback([],[],[],eye(d),a,b,c,d) sigma(a,b,c,d,'inv')
$(I+G(jw))^{-1}$		[a,b,c,d] = feedback(a,b,c,d,[],[],[],eye(d)) sigma(a,b,c,d,'inv')

The singular value response of a SISO system is identical to the Bode magnitude response of that system.

Examples

Figure 2-5, Singular Value Bode Plot of LSS. shows the singular value Bode plot of the open loop large space structure in [1]. There are 58 vibrational

modes (116 states) with damping ratio 0.3 to 0.002 scattered in the frequency range from 0.4 **H**z to 477 **H**z, 18 control actuators and 20 sensors.

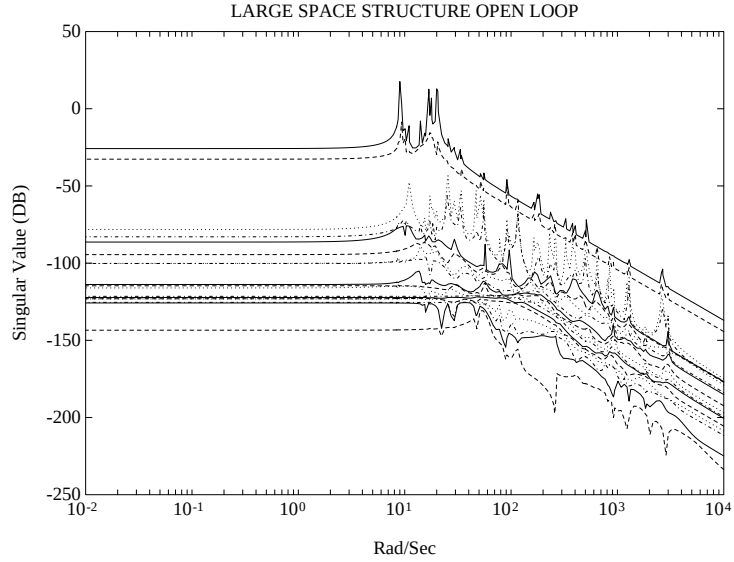


Figure 2-5: Singular Value Bode Plot of LSS.

Algorithm

`Sigma` and `dsigma` use the `svd` function in MATLAB.

See Also

bode, cgloci, dcgloci, dbode, logspace

References

[1] M. G. Safonov, R. Y. Chiang and H. Flashner, "**H**∞ Control Synthesis for a Large Space Structure," *AIAA J. Guidance, Control and Dynamics*, 14, 3, pp. 513-520, May/June 1991.

Purpose

State space realization for a given magnitude Bode plot.

Syntax

```
[ad,bd,cd,dd,logdfit] = fitd(logd,w)
[ad,bd,cd,dd,logdfit] = fitd(logd,w,n)
[ad,bd,cd,dd,logdfit] = fitd(logd,w,n,blksz)
[ad,bd,cd,dd,logdfit] = fitd(logd,w,n,blksz,flag)
[ssd,logdfit] = fitd( )
```

Description

`fitd` produces a continuous stable minimum-phase state-space realization `ssd` of a diagonal transfer function matrix such that the diagonal elements' magnitude Bode plots approximately fit Bode magnitude plot data given in the rows of the matrix `logd`.

Input variable `logd` is a matrix whose rows are logarithmic magnitude Bode plots evaluated at frequency vector `w`.

Optional input variables:

`n` — vector containing orders of the state-space approximants of the diagonal scalings (default = 0)

`blksz` — a vector of the size of the diagonal blocks (default = 1 for each block).
`flag` — set to "1" to display a Bode plot (default = 1).

`fitd` uses the stable routine `yulewalk` in the Signal Processing Toolbox to fit the continuous magnitude Bode plot. `Bilin` and `polyfit` are also used to pre-process the frequency axis shift from s-domain to z-domain. The final discrete realization is transformed back to s-domain via `bilin`.

`fitd` plays a crucial role in μ-synthesis design technique. The success of "*D* – *F*" iterations in μ-synthesis depends on the result of `fitd`.

See Also

bilin, fitgain, invfreqs, musyn, polyfit, yulewalk

fitgain

Purpose

State space realization for a given magnitude Bode plot.

Syntax

```
[ad,bd,cd,dd,logfit] = fitgain(logd,w)
[ad,bd,cd,dd,logfit] = fitgain(logd,w,n)
[ad,bd,cd,dd,logfit] = fitgain(logd,w,n,wt)
[ad,bd,cd,dd,logfit] = fitgain(logd,w,n,wt,flag)
[ssd,logfit] = fitgain( )
```

Description

`fitgain` produces a stable and minimum-phase state-space realization of the given magnitude Bode plot `logd`.

The input variable `logd` is a vector containing the logarithmic magnitude Bode plot evaluated at frequency vector `w`.

Optional input variables:

`n` — the size of the desired state-space approximant (default = 0)

`wt` — a weighting vector contains the weight at each frequency point

`flag` — set to "1" to display a Bode plot (default = 1).

`fitgain` uses the routine `invfreqs` in Signal Processing Toolbox to fit the continuous magnitude Bode plot. Three steps are involved:

1. Take the *power spectrum density* of the given magnitude data, i.e., $|G(s)|^2 = G(s)G(-s)$.
2. Use the error correction method (routine `invfreqs`) to fit the PSD data with a rational transfer function.
3. Extract the stable and minimum phase part of the realization.

This method is not as numerically stable as `fitd`.

See Also

fitd, invfreqs, yulewalk

Purpose

Adds root branch onto a tree.

Syntax

```
TR = graft(TR1,B)
TR = graft(TR1,B,NM)
```

Description

`graft` adds root branch `B` onto a tree variable `TR1` (previously created by `tree` or `mksys`). If `TR1` has N branches, then the numerical index of the new branch is N+1; and the numerical indices of other root branches are unchanged.

The string name `NM`, if present, becomes the name of the new root branch.

See Also

tree, mksys, branch, istree, issystem, vrsys

Purpose

$\mathbf{H}^2$ optimal control synthesis (continuous and discrete).

Syntax

```
[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = (d)h2lqg(A,B1,B2, ,D22)
[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = (d)h2lqg(A,B1,B2, ,D22,aretype)
[sscp,sscl] = (d)h2lqg(TSS)
[sscp,sscl] = (d)h2lqg(TSS,aretype)
```

Description

`h2lqg` solves $\mathbf{H}^2$ optimal control problem; i.e., find a stabilizing *positive-feedback* controller for an "augmented" system

$$P(s) := \left[\begin{array}{cc|c} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

such that the $\mathbf{H}^2$-norm of the closed-loop transfer function matrix $T_{y_1u_1}(s)$ is minimized:

$$\min_{F(s)} \left\| T_{y_1u_1} \right\|_2 := \min_{F(s)} \left(\frac{1}{\pi} \int_0^\infty \mathrm{trace}(T^*_{y_1u_1}(j\omega) T_{y_1u_1}(j\omega)) d\omega \right)^{1/2}$$

The stabilizing feedback law $F(s)$ and the closed-loop transfer function $T_{y_1u_1}(s)$ are returned as

$$F(s) := (acp,\ bcp,\ ccp,\ dcp)$$

$$T_{y_1u_1}(s) := (acl,\ bcl,\ ccl,\ dcl)$$

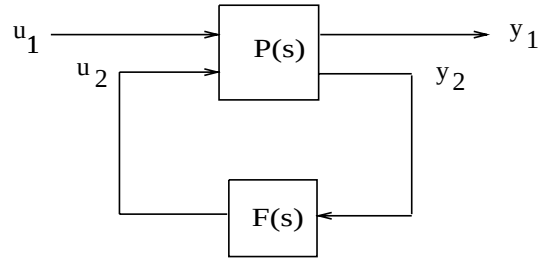


Figure 2-6: H^2 Control Synthesis.

The optional input `aretype` determines the method used by ARE solver `aresolv`. It can be either `"eigen"` (default), or `"Schur"`.

`dh2lqg` solves the discrete counterpart of the problem by directly forming two discrete ARE's and solve them via `daresolv`. Note that in contrast to the $\mathbf{H}^\infty$ case, the bilinear transform technique does not apply in the $\mathbf{H}^2$ case. This is because the $\mathbf{H}^2$ norm, unlike the $\mathbf{H}^\infty$ norm, is not invariant under bilinear transformation.

Examples

See the *Tutorial* chapter for design examples and demonstrations. Especially, see the comparison between $\mathbf{H}^2$ synthesis and $\mathbf{H}^\infty$synthesis in the *Fighter* $\mathbf{H}^2$ and $\mathbf{H}^\infty$*Design Example* in the Tutorial section.

Algorithm

`H2lqg` solves the $\mathbf{H}^2$-*norm optimal control problem* by observing that it is equivalent to a conventional *Linear-Quadratic Gaussian optimal control problem* involving cost

$$
\begin{aligned}
J_{LQG} &= \lim_{T\to\infty} E\left\{\frac{1}{T}\int_0^T y_1^T y_1 dt\right\} \\
&= \lim_{T\to\infty} E\left\{\frac{1}{T}\int_0^T [x^T \;\; u_2^T]\begin{bmatrix} Q & N_c \\ N_c^T & R\end{bmatrix}\begin{bmatrix} x \\ u_2\end{bmatrix} dt\right\} \\
&= \lim_{T\to\infty} E\left\{\frac{1}{T}\int_0^T [x^T \;\; u_2^T]\begin{bmatrix} C_1^T \\ D_{12}^T\end{bmatrix}[C_1 \;\; D_{12}]\begin{bmatrix} x \\ u_2\end{bmatrix} dt\right\}
\end{aligned}
$$

with correlated white plant noise ξ and white measurement noise θ entering the system via the channel $[B_1\ D_{21}]^T$ and having joint correlation function

$$E\left\{\begin{bmatrix}\xi(t)\\ \theta(t)\end{bmatrix}[\xi(\tau)\ \ \theta(\tau)]^T\right\} = \begin{bmatrix}\Xi & N_f\\ N_f^T & \Theta\end{bmatrix} = \begin{bmatrix}B_1\\ D_{21}\end{bmatrix}\begin{bmatrix}B_1^T\ D_{21}^T\end{bmatrix}\delta(t-\tau)$$

$$= \begin{bmatrix}B_1B_1^T & B_1D_{21}^T\\ D_{21}B_1^T & D_{21}D_{21}^T\end{bmatrix}\delta(t-\tau)$$

The $\mathbf{H}^2$ optimal controller $F(s)$ is thus realizable in the usual LQG manner as a full-state feedback K_c and a Kalman filter with residual gain matrix K_f.

1 Kalman Filter

$$\hat{x} = A\hat{x} + B_2u_2 + K_f(y_2 - C_2\hat{x} - D_{22}u_2)$$

$$K_f = (\Sigma C_2^T + N_f)\Theta^{-1} = (\Sigma C_2^T + B_1D_{21}^T)(D_{21}D_{21}^T)^{-1}$$

where $\Sigma = \Sigma^T$and satisfies ARE

$$\Sigma A^T + A\Sigma - (\Sigma C_2^T + N_f)\Theta^{-1}(C_2\Sigma + N_f^T) + \Xi = 0$$

2 Full-State Feedback

$$u_2 = K_c\hat{x}$$

$$K_c = R^{-1}(B_2^TP + N_c^T) = (D_{12}^TD_{12})^{-1}(B_2^TP + D_{12}^TC_1)$$

where $P = \mathrm{P}^T$ and satisfies ARE

$$A^TP + PA - (PB_2 + N_c)R^{-1}(B_2^TP + N_c^T) + Q = 0$$

The final *positive*-feedback $\mathbf{H}^2$ optimal controller $u_2 = F(s)y_2$ has a familiar closed-form

$$F(s) := \left[\begin{array}{c|c} A - K_f C_2 - B_2 K_c + K_f D_{22} K_c & K_f \\ \hline -K_c & 0 \end{array}\right];$$

It can be easily shown that by letting $D_{21} \to 0$ the $\mathbf{H}^2$-optimal LQG problem is essentially equivalent to LQ *full-state feedback loop transfer recovery* (see `ltru`). Dually, as $D_{12} \to 0$ you obtain *Kalman filter loop transfer recovery* [1] (see `ltry`).

Limitations

1. (A, B_2, C_2) must be *stabilizable and detectable*.
2. D_{11} must be zero, otherwise the $\mathbf{H}^2$ optimal control problem is ill-posed. If a nonzero D_{11} is given, the algorithm ignores it and computes the $\mathbf{H}^2$ optimal control as if D_{11} were zero.
3. D_{12} and D_{21}^T must both have full column rank.

See Also

`hinf, dhinf, lqg, ltru, ltry, aresolv, daresolv`

References

[1] J. Doyle and G. Stein, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," *IEEE Trans. on Automat. Contr*., AC-26, pp. 4-16, 1981.

[2] J. Doyle, *Advances in Multivariable Control*. Lecture Notes at ONR/Honeywell Workshop. Minneapolis, MN, Oct. 8-10, 1984.

[3] M. G. Safonov, A. J. Laub, and G. Hartmann, "Feedback Properties of Multivariable Systems: The Role and Use of Return Difference Matrix," *IEEE Trans. of Automat. Contr*., AC-26, pp. 47-65, 1981.

[4] G. Stein and M. Athans, "The LQG/LTR Procedure for Multivariable Feedback Control Design," *IEEE Trans. on Automat. Contr*., AC-32, pp. 105-114, 1987.

Purpose

$\mathbf{H}^\infty$ optimal control synthesis (continuous and discrete)

Syntax

```
linf
    Inputs: A, B1, B2, C1, C2, D11, D12, D21, D22
    Outputs: acp, bcp, ccp, dcp, acl, bcl, ccl, dcl
[acp, ,acl, ,hinfo,ak, ,dk22] = (d)hinf(A, ,D22)
[acp, ,acl, ,hinfo,ak, ,dk22] = (d)hinf(A, ,D22,au, ,du)
[acp, ,acl, ,hinfo,ak, ,dk22] = ...
  (d)hinf(A, ,D22,au, ,du,verbose)
[sscp,sscl,hinfo,tssk] = (d)hinf(TSSP)
[sscp,sscl,hinfo,tssk] = (d)hinf(TSSP,ssu,)
[sscp,sscl,hinfo,tssk] = (d)hinf(TSSP,ssu,verbose)
```

Description

`linf` and `hinf` solve the small-gain infinity-norm robust control problem; i.e., find a stabilizing controller $F(s)$ for a system

$$P(s) := \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

such that the closed-loop transfer function satisfies the infinity-norm inequality

$$\left\| T_{y_1 u_1} \right\|_\infty \stackrel{\Delta}{=} \sup_\omega \sigma_{max}(T_{y_1 u_1}(j\omega)) < 1$$

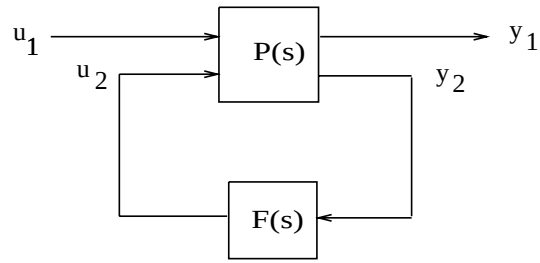


Figure 2-7: Particular $F(s)$

State-space matrices for a particular solution $F(s)$ and the corresponding $T_{y\ u_i}(s)$ are returned as

$$F(s) := (acp,\ \ bcp,\ \ ccp,\ \ dcp) or\ \ sscp$$

$$T_{y_1 u_1}(s) := (acl,\ \ bcl,\ \ ccl,\ \ dcl) or\ \ sscl$$

`linf` finds the solution $F(s)$ using the *Hankel approximation* algorithm outlined in [15], whereas `hinf` uses a more recent *two-Riccati* algorithm [18, 8, 9, 3, 5, 6, 17].

In general, the solution to the infinity-norm optimal control problem is non-unique. Whereas `linf` computes only a particular $F(s)$, `hinf` computes in addition the all- solution controller parameterization $K(s)$ such that all solutions to the infinity-norm control problem are parameterized by a free stable contraction map $U(s)$ constrained by $(\|U(s)\|_\infty < 1)$ (see Figure 2-8, All-solution F(s)). By default `hinf` will set $U(s) = 0$, if no $U(s)$ is supplied. But if you specify

$U(s) := (au, bu, cu, du)$ in advance, `hinf` will compute the corresponding $F(s)$ as shown in Figure 2-8, All-solution F(s).

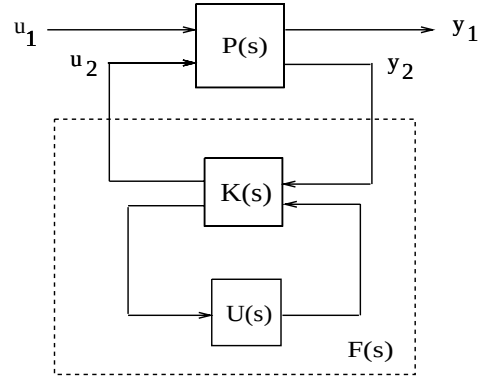


Figure 2-8: All-solution $F(s)$

An important use of the infinity-norm control theory is for direct shaping of closed-loop singular value Bode plots of control systems. In such cases, the system $P(s)$ will typically be the plant augmented with suitable loop-shaping filters — see `augss` and `augtf`.

`dhinf` solves the discrete counterpart of the problem using bilinear transform `bilin`, since the infinity norm of a given discrete problem is preserved under such a transformation in the continuous domain. The resulting continuous $\mathbf{H}^\infty$ controller is then transformed back to the discrete domain via the inverse bilinear transform inside program `dhinf`. The discrete time $\mathbf{H}^\infty$ theory is documented nicely in [10].

Examples

See the *Tutorial* chapter for $\mathbf{H}^\infty$ design examples and their demonstrations in `rctdemo`. Also see the examples in `hinfopt`.

Algorithm

`linf` implements the first generation state-space $\mathbf{H}^\infty$ theory developed in 1984 to 1987 ([2, 14, 1, 4]), which splits the $\mathbf{H}^\infty$ problem into three phases:

1. Plant augmentation (see `augtf`, `augss`).
2. Youla parametrization (see `youla`).
3. Interpolation via optimal anticausal descriptor Hankel approximation (see `ohklmr`).

The bottleneck of `linf` algorithm is the lengthy model reduction work required in step 3. However, successful results using `linf` have been reported in [12, 16].

The 2-Riccati $\mathbf{H}^\infty$ controller has been derived via two distinct approaches — *game theory (in time domain)* and *all-pass embedding (in frequency domain)*. The game theory approach is conceptually abstract but leads to a much simpler derivation. In 1977 Mageirou and Ho [11] solved the full-state feedback case via game theory and others [7] later "rediscovered" it. Subsequently, Doyle et al.[3] extended the full-state feedback result into its observer dual and established a separation principle for $\mathbf{H}^\infty$ (a counterpart of the LQG). A frequency domain derivation for the dynamical output feedback case was initiated by Limebeer, et al. [8, 9, 6] using Parrott's all-pass embedding technique and the optimal Hankel norm theorem. Both approaches require a large effort and much algebraic manipulation to handle all but very special cases in which the plant "C" and "D" matrices satisfied certain conditions. Safonov et al. [17] developed a loop-shifting technique to simplify the derivations, introduced a descriptor matrix-pencil representation and improved existence conditions. The latter eliminated numerical instabilities that had plagued previous formulations of the $\mathbf{H}^\infty$ control theory. A matrix pencil is an s-dependent matrix of the form $As + B$. The generalized eigenvalues of a regular square matrix pencil, denoted $\lambda i(As + B)$, are the values of $s \in C$ at

which the determinant of the pencil vanishes; numerically robust algorithms exist for computing the generalized eigenvalues and associated eigenspaces.

`hinf` implements the loop-shifting "two-Riccati" formulae for the infinity-norm control [17]. The chief advantage of `hinf` over `linf` is that the lengthy numerically sensitive model reduction work is completely eliminated. Instead, `hinf` produces an $\mathbf{H}^\infty$ controller with the same state dimension as the augmented plant $P(s)$.

Limitations

In contrast to the $\mathbf{H}^2$ problem, a solution to the infinity-norm control problem does not exist for every $P(s)$. Error messages such as "*Riccati solver fails*" are possible indications that the augmented system $P(s)$ arises from singular-value Bode plot specifications that are infeasible or, possibly, that certain other well-posedness conditions have been violated. In particular, the algorithms `linf` and `hinf` require that the following conditions hold: [13]

$$\text{rank}(D_{12}) = \dim(u_2) \le \dim(y_1)$$

that is, D_{12} must be a "tall" matrix with full "column" rank.

$$\text{rank}(D_{21}) = \dim(y_2) \le \dim(u_1)$$

that is, D_{21} must be a "fat" matrix with full "row" rank.

Careful problem formulations can avoid some numerically or physically ill-posed $\mathbf{H}^\infty$ problems. For example,

1. Always include a proper control weighting W_2 to ensure that D_{12} is a full column rank (ref. `augtf`)
2. Form a standard mix-sensitivity problem with D_{21} square. This formulation has solved a lot of real world control problems (flight control, large space structure, spacecraft attitude control, etc.).
3. Use some classical loop-shaping judgments to penalize your physical variables.
4. If still no solution is found, try γ-Iteration `hinfopt` on the problem.

If you have exhausted all the above possibilities, you can always call the authors for help.

See Also

augss, augtf, h2lqg, hinfdemo, linfdemo, lqg, ltru, ltry

References

[1] R. Y. Chiang and M. G. Safonov, "The LINF Computer Program for L∞ Controller Design," USC report EECG-0785-1, ver. 1.0 2.0, July, 1986 and 1987.

[2] J. Doyle, *Advances in Multivariable Control*. Lecture Notes at ONR/Honeywell Workshop. Minneapolis, MN, Oct. 8-10, 1984.

[3] J. Doyle, K. Glover, P. Khargonekar, and B. Francis, "State-space solutions to standard $\mathbf{H}^2$ and $\mathbf{H}^\infty$ control problems," *IEEE Trans. Automat. Contr*., AC-34, no. 8, pp. 831-847, Aug. 1989.

[4] B. A. Francis, *A Course i*n $\mathbf{H}^\infty$ *Control Theory*, Springer-Verlag: 1987.

[5] K. Glover and J. C. Doyle, "State Space Formulae for All Stabilizing Controllers that Satisfy an $\mathbf{H}^\infty$-Norm Bound and Relations to Risk Sensitivity", *Systems and Control Letters*, 1988.

[6] K. Glover, D. J. N. Limebeer, J. C. Doyle, E. M. Kasenally and M. G. Safonov, "A Characterization of All Solutions to the Four Block General Distance Problem", *SIAM J. Control and Opt*., vol. 27, pp. 283-324, 1991.

[7] P. P. Khargonekar, I. R. Petersen, and M. A. Rotea, "$\mathbf{H}^\infty$ optimal control with state feedback," *IEEE Trans. Automat. Contr*., AC-33, pp. 783-786, 1988.

[8] D.J.N. Limebeer and E. Kasenally, unpublished notes, 1987.

[9] D.J.N. Limebeer, E. M. Kasenally, E. Jaimouka, and M. G. Safonov, "A Characterization of All Solutions to the Four Block General Distance Problem," *Proc. 27th IEEE Conf. Decision Contr*., Austin, TX, 1988.

[10] D.J.N. Limebeer, M. Green, and D. Walker, "Discrete Time $\mathbf{H}^\infty$ control," *Proc. of Conf. on Decision and Contro*l, Tampa, FL., Dec. 1989.

[11] E. F. Mageirou and Y. C. Ho, "Decentralized Stabilization via Game Theoretic Methods," *Automatica*, vol. 13, pp. 393-399, 1977.

[12] M. G. Safonov and R. Y. Chiang, "CACSD Using the State-Space L∞ Theory — A Design Example", *Proc. IEEE Conf. on CACSD*, Washington D. C., Sep. 24-26, 1986, also I*EEE Trans. on Automat. Contr*., AC-33, No. 5, pp. 477-479, May 1988.

[13]M. G. Safonov, “Imaginary-Axis Zeros in Multivariable **H**∞Optimal Control”, in R. F. Curtain (editor), *Modelling, Robustness and Sensitivity Reduction in Control Systems*, pp. 71-81, Springer-Verlag, Berlin, 1987. *Proc. NATO Advanced Research Workshop on Modeling, Robustness and Sensitivity Reduction in Control Systems*, Groningen, The Netherlands, Dec. 1-5, 1986.

[14] M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, “Synthesis of Positive Real Multivariable Feedback Systems”, I*nt. J. Contro*l, vol. 45, no. 3, pp. 817-842, 1987.

[15] M. G. Safonov, R. Y. Chiang, and D. J. N. Limebeer, “Hankel Model Reduction without Balancing – A Descriptor Approach,” P*roc. IEEE Conf. on Decision and Control*, Los Angeles, CA, Dec. 9-11, 1987.

[16] M. G. Safonov, R. Y. Chiang and H. Flashner, “**H**∞ Control Synthesis for a Large Space Structure,” *AIAA J. Guidance, Control and Dynamics*, 14, 3, pp. 513-520, May/June 1991.

[17] M. G. Safonov, D. J. N. Limebeer and R. Y. Chiang, “Simplifying the **H**∞ Theory via Loop Shifting, Matrix Pencil and Descriptor Concepts”, *Int. J. Contr.*, vol. 50, no. 6, pp. 2467-2488, 1989.

[18] G. Stein, *Lecture Notes, Tutorial Workshop on* **H**∞ *Control Theory*, Los Angeles, CA, Dec. 7-8, 1987.

hinfopt

Purpose

H∞ optimal control synthesis via γ-iteration.

Syntax

```
[gamopt,acp, ,dcp,acl, ,dcl] = hinfopt(A, ,D22)
[gamopt,acp, ,dcp,acl, ,dcl] = hinfopt(A, ,D22,gamind)
[gamopt,acp, ,dcp,acl, ,dcl] = hinfopt(A, ,D22,gamind,aux)
[gamopt,sscp,sscl] = hinfopt(tss)
[gamopt,sscp,sscl] = hinfopt(tss,gamind)
[gamopt,sscp,sscl] = hinfopt(tss,gamind,aux)
```

Description

`hinfopt` does **H**∞ "γ-iteration" to compute the optimal **H**∞ controller using the loop-shifting two-Riccati formulae of `hinf`. The output `gamopt` is the optimal "γ" for which the cost function $T_{y\ u_i}$ can achieve under a preset tolerance

$$\left\| \begin{bmatrix} \gamma T_{y_1 u_1}(gamind,:) \\ T_{y_1 u_1}(otherind,:) \end{bmatrix} \right\|_\infty \le 1$$

An optional input variable `aux` specifies the tolerance that stops the γ-iteration

$$aux = [tol \ \ maxgam \ \ mingam]$$

where *maxgam* and *mingam* denotes a range for γ-iteration to be carried out. The default value of `aux` is `[0.01 1 0]`. Another optional input variable `gamind` specifies the indices of the cost function output channels (i.e.,rows) to be scaled by γ. Default for `gamind` is to scale all the output channels ($gamind = 1{:}\ n$).

Algorithm

A binary search algorithm is coded in `hinfopt` to achieve the required γ-iteration. The iteration logic is based on the **H**∞ existence tests performed inside `hinf` and recorded in the output variable `hinfo` of `hinf`. The search of optimal γ stops whenever the γ relative error between two adjacent stable solutions is less than the tolerance specified. For most practical purposes, the tolerance can be set at 0.01 or 0.001.

Examples

Following are three simple problems solved via `hinfopt` with the SISO plant

$$G(s) = \frac{s-1}{s-2}$$

Problem 1: Mixed-Sensitivity $W_1 = \frac{0.1(s+1000)}{100s+1}$, $W_2 = 0.1$, no W_3.

```
[ag,bg,cg,dg] = tf2ss([1 -1],[1 -2]);
ssg = mksys(ag,bg,cg,dg);
w1 = [0.1*[1 100];[100 1]]; w2 = [0.1;1]; w3 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,[1:2],[0.001,1,0]);
```

In this case, $\gamma_{opt} = 1.5146$.

Problem 2: W_1 is removed.

```
w1 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,1,[0.001,1,0]);
```

In this case, $\gamma_{opt} = 2.5$, $F(s) = -4/3$.

Problem 3: $W_1 = \frac{s+1}{10s+1}$, W_2 is removed.

```
w1 = [1 1;10 1]; w2 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,1,[0.001,1,0]);
```

For this problem, $F(s) \approx \infty$, $\gamma_{opt} = 11/6$.

These three very simple problems can be also solved analytically using the interpolation technique of [1], augmented in the case of problem 1 with the "two-block to one-block" embedding technique of [2]. The results of `hinfopt` match the exact solutions very well.

See Also

augss, augtf, hinf, linf

References

[1] G. Zames and B. A. Francis, "Feedback, Minimax Sensitivity, and Optimal Robustness," *IEEE Trans. on Autom. Contro*l, AC-28, 5, pp. 585-601, May 1983.

[2] M. Verma and E. A. Jonckheere, "L$^\infty$-Compensation with Mixed Sensitivity as a Broadband Matching Problem," *Systems and Control Letters,* 4, pp. 125-129, May 1984.

imp2ss

Purpose

System realization via Hankel singular value decomposition.

Syntax

```
[a,b,c,d,totbnd,svh] = imp2ss(y)
[a,b,c,d,totbnd,svh] = imp2ss(y,ts,nu,ny,tol)
[ss,totbnd,svh] = imp2ss(imp)
[ss,totbnd,svh] = imp2ss(imp,tol)
```

Description

The function `imp2ss` produces an approximate state-space realization of a given impulse response

```
imp=mksys(y,t,nu,ny,'imp');
```

using the Hankel SVD method proposed by S. Kung [2]. A continuous-time realization is computed via the inverse Tustin transform (using `bilin`) if *t* is positive; otherwise a discrete-time realization is returned. In the SISO case the variable *y* is the impulse response vector; in the MIMO case *y* is a *N+1*-column matrix containing $N + 1$ time samples of the matrix-valued impulse response $H_0, \ldots, H_N$ of an `nu`-input, `ny`-output system stored row wise:

$$y = [H_0(:)';H_2(:)';H_3(:)';\ldots;H_N(:)']$$

The variable *tol* bounds the $\mathbf{H}^\infty$ norm of the error between the approximate realization *(a, b, c, d)* and an exact realization of *y*; the order, say *n*, of the realization *(a, b, c, d)* is determined by the infinity norm error bound specified by the input variable *to*l. The inputs `ts`, `nu`, `ny`, `tol` are optional; if not present they default to the values `ts` = 0, `nu` = 1, `ny` = (no. of rows of y)/`nu`, tol $= 0.01\bar{\sigma}_1$. The output svh $= [\bar{\sigma}_1, \bar{\sigma}_2, \ldots]'$ returns the singular values (arranged in descending order of magnitude) of the Hankel matrix:

$$\Gamma = \begin{bmatrix} H_1 & H_2 & H_3 & \ldots & H_N \\ H_2 & H_3 & H_4 & \ldots & 0 \\ H_3 & H_4 & H_5 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_N & 0 & \ldots & \ldots & 0s \end{bmatrix}$$

Denoting by G_N a high-order exact realization of y, the low-order approximate model G enjoys the $\mathbf{H}^\infty$ norm bound

$$\left\| G - G_N \right\|_\infty \le \text{totbnd}$$

where

$$\text{totbnd} = 2 \sum_{i=n+1}^{N} \bar{\sigma}_i$$

Algorithm

The realization *(a, b, c, d)* is computed using the Hankel SVD procedure proposed by Kung [2] as a method for approximately implementing the classical Hankel factorization realization algorithm. Kung's SVD realization procedure was subsequently shown to be equivalent to doing balanced truncation (`balmr`) on an exact state space realization of the finite impulse response $\{y(1),\ldots y(N)\}$ [3]. The infinity norm error bound for discrete balanced truncation was later derived by Al-Saggaf and Franklin [1]. The algorithm is as follows:

1 Form the Hankel matrix Γ from the data y.

2 Perform SVD on the Hankel matrix

$$\Gamma = U\Sigma V^* = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V^*_1 \\ V^*_2 \end{bmatrix} = U_1 \Sigma_1 V^*_1$$

where Σ_1 has dimension $n \times n$ and the entries of Σ_2 are nearly zero. U_1 and V_1 have ny and nu columns, respectively.

3 Partition the matrices U_1 and V_1 into three matrix blocks:

$$U_1 = \begin{bmatrix} U_{11} \\ U_{12} \\ U_{13} \end{bmatrix}; \quad \begin{bmatrix} V_{11} \\ V_{12} \\ V_{13} \end{bmatrix}$$

where $U_{11},\ U_{13} \in C^{ny \times n}$ and $V_{11},\ V_{13} \in C^{nu \times n}$.

4 A discrete state-space realization is computed as

$$A = \Sigma_1^{-1/2}\overline{U}\Sigma_1^{1/2}$$

$$B = \Sigma_1^{-1/2}V^*{}_{11}$$

$$C = U_{11}\Sigma_1^{-1/2}$$

$$D = H_0$$

where

$$\overline{U} = \begin{bmatrix} U_{11} \\ U_{12} \end{bmatrix}' \begin{bmatrix} U_{12} \\ U_{13} \end{bmatrix}$$

5 If the sampling time t is greater than zero, then the realization is converted to continuous time via the inverse of the Tustin transform

$$s = \frac{2}{t}\frac{z-1}{z+1};$$

otherwise, this step is omitted and the discrete-time realization calculated in Step 4 is returned.

See Also

ohklmr, schmr, balmr, bstschmr

References

[1] U. M. Al-Saggaf and G. F. Franklin, "An Error Bound for a Discrete Reduced Order Model of a Linear Multivariable System," *IEEE Trans. on Autom. Contr.*, AC-32, pp. 815-819, 1987.

[2] S. Y. Kung, "A New Identification and Model Reduction Algorithm via Singular Value Decompositions," *Proc.Twelth Asilomar Conf. on Circuits, Systems and Computers.*, pp. 705-714, November 6-8, 1978.

[3] L. M. Silverman and M. Bettayeb, "Optimal Approximation of Linear Systems," *Proc. American Control Conf.*, San Francisco, CA, 1980.

Purpose

General multivariable interconnected system.

Syntax

```
[acl,bcl,ccl,dcl] = interc(a,b,c,d,m,n,f)
```
[sscl] = interc(ss,m,n,f)

Description

`interc` computes state-space realization of a multivariable interconnected system closed loop, given system $P(s) := C(Is - A)^{-1}B + D$ and constant blocks M, N, and F representing the interconnections (see block diagram).

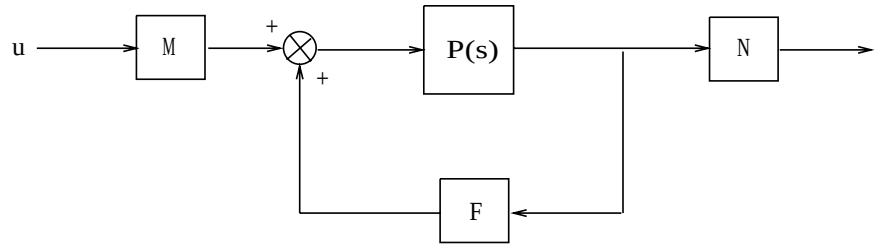


Figure 2-9: General MIMO Interconnection.

The resulting system closed-loop is

$$\left[\begin{array}{c|c} A_{cl} & B_{cl} \\ \hline C_{cl} & D_{cl} \end{array}\right] := \left[\begin{array}{c|c} A + BFXC & B(M + FXDM) \\ \hline NXC & NXDM \end{array}\right]$$

where $X = (I - DF)^{-1}$.

Examples

Consider a system with three subsystems (G_1, G_2, G_3) where each of the subsystems has its own state-space representation (A_x, B_x, C_x, D_x). If the overall system is interconnected as shown in the Figure 2-10, Example of MIMO Interconnection., then

$$P(s) = \begin{pmatrix} G_1(s) & 0 & 0 \\ 0 & G_2(s) & 0 \\ 0 & 0 & G_3(s) \end{pmatrix}$$

and the associated constant blocks M, N, F and for this problem are

$$M = \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$$

$$N = \begin{bmatrix} I & 0 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & 0 & I \\ 0 & 0 & I \\ I & I & 0 \end{bmatrix}$$

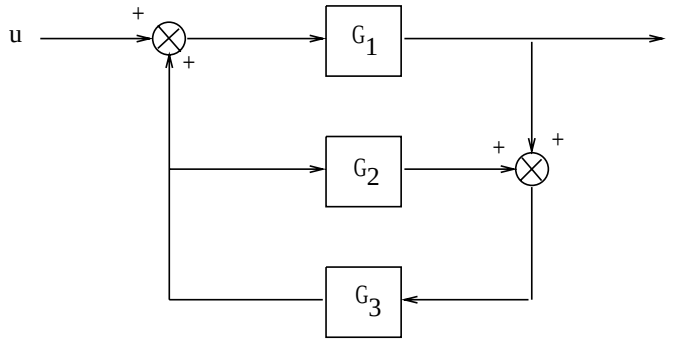


Figure 2-10: Example of MIMO Interconnection.

Using the following MATLAB commands, you can easily get a state-space realization of $P(s)$ as

```
[AA,BB,CC,DD] = append(A1,B1,C1,D1,A2,B2,C2,D2);
[AA,BB,CC,DD] = append(AA,BB,CC,DD,A3,B3,C3,D3);
```

Then, the state-space representation (Acl, Bcl, Ccl, Dcl) of the whole system from $u \rightarrow$ y is

```
[Acl,Bcl,Ccl,Dcl] = interc(AA,BB,CC,DD,M,N,F);
```

Note that the resulting system is *not necessarily minimal*; for example, pole-zero cancellations that arise as a result of the interconnection lead to such a nonminimal state-space realization. Model reduction routines such as

`minreal`, `schmr`, `ohklmr`, or `bstschmr` can be helpful in removing any uncontrollable and/or unobservable modes.

See Also

append, feedback, lftf, sectf, tfm2ss

Purpose

Inner-outer factorization (row type).

Inner-outer factorization (column type).

Syntax

```
[ain, ,ainp, ,aout, ] = iofr(c)(a,bcd)
[ssin,ssinp,ssout] = iofr(c)(ss)
```

Description

A square transfer function $M(s)$ is *outer* if it is proper and stable and has an inverse that is also proper and stable. A transfer function $\theta(s)$ of dimension m by n is *inner* if it is stable and satisfies

$$\theta^T(-s)\theta(s) = I,\ \forall s,\ if\ \ m \geq n,\ (row\ \ type)$$

or

$$\theta(s)\theta^T(-s) = I,\ \forall s,\ if\ \ m \geq n,\ (column\ \ type)$$

When $m \neq n,\ \theta(s)$ has a *complementary inner* (or *all-pass extension*) $\theta^{\perp}(s)$ such that $[\theta(s)|\theta^{\perp}(s)]$ or $\begin{bmatrix}\theta(s)\\ \theta^{\perp}(s)\end{bmatrix}$ is square and inner.

`Iofr` computes an inner-outer factorization for a stable transfer function $G(s) := (A,\ B,\ C,\ D)$ for which $m \geq n$ such that

$$G = [\theta\ \ \theta^{\perp}]\ \begin{bmatrix}M\\ 0\end{bmatrix}$$

The output variables are defined as

$$\text{ssin} \triangleq \theta(s) := \begin{bmatrix} A_\theta & B_\theta \\ C_\theta & D_\theta \end{bmatrix}$$

$$\text{ssimp} \triangleq \theta^{\perp}(s) := \begin{bmatrix} A_{\theta\perp} & B_{\perp\theta} \\ C_{\theta\perp} & D_{\theta\perp} \end{bmatrix}$$

$$\text{ssout} \triangleq \begin{bmatrix} M(s) \\ 0 \end{bmatrix} := \begin{bmatrix} A_M & B_M \\ C_M & D_M \end{bmatrix}$$

`iofc` computes an inner-outer factorization for the case of $m < n$ via duality by applying `iofr` to $G^{\mathrm{T}}(s)$, then transposing the result.

Algorithm

`iofr` implements the algorithm described in [1], where it is shown that inner-outer factorization relates closely to the standard *optimal LQ control problem* as follows:

Given a transfer function $G(s) := (A, B, C, D)$ of dimension $m \times n$ $(m \geq n)$, the LQR optimal control $u = -Fx = -R^{-1}(XB + N)^{\mathrm{T}}x$ stabilizes the system and minimizes the quadratic cost function

$$J = \frac{1}{2}\left(x^T(t_f)P_1x(t_f) + \int_{t_0}^{t_f} [x^T u^T] \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt \right)$$

as $t_f \to \infty,$ where $X = X^T > 0$, satisfies the algebraic Riccati equation

$$A^TX + XA - (XB + N)R^{-1}(XB + N)^T + Q = 0$$

Moreover, the optimal return difference $I + L(s) = I + F(Is - A)^{-1}B$ satisfies the *optimal LQ return difference equality*:

$$(I + L)^*R(I + L) = [\Phi^*I] \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} \Phi \\ I \end{bmatrix}$$

where $\Phi(s) = (Is - A)^{-1}B$, $\Phi^*(s) = \Phi^T(-s)$, and

$$\begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} = \begin{bmatrix} C^T \\ D^T \end{bmatrix} [C \;\; D]$$

It may be easily shown [1] that the return difference equality implies that an inner-outer factorization of $G(s)$ is given by

$$[\theta(s) | \theta^{\perp}(s)] = \left[\begin{array}{c|cc} A - BF & BR^{-1/2} & -X^{-1}C^T D^{\perp} \\ \hline C - DF & DR^{-1/2} & D^{\perp} \end{array}\right]$$

and

$$M^{-1}(s) = \left[\begin{array}{c|c} A - BF & BR^{-1/2} \\ \hline -F & R^{-1/2} \end{array}\right]$$

The variables X and F are computed via the MATLAB command:

```
[F,X] = lqr(A,B,Q,R,N) = lqr(A,B,C'*C,D'*D,C'*D).
```

The matrix X^{-1} is a generalized inverse (e.g., a pseudoinverse). Although X may be singular, $X^{-1}C^T D^{\perp}$ is well defined since the left null-space of $C^T D^{\perp}$ includes the left null-space of X [1].

`Iofc` applies `iofr` to $G^{\mathrm{T}}(s)$, then transposes the result.

Limitations

The inner-outer factorization requires the system $G(s)$ to be stable and to have neither poles nor transmission zeros on the $j\omega$-axis or at ∞. In particular D must have full column rank for `iofr` or full row rank for `iofc`.

See Also

sfl, sfr

References

[1] J. Doyle, *Advances in Multivariable Control*. Lecture Notes at ONR/Honeywell Workshop. Minneapolis, MN, Oct. 8-10, 1984.

[2] M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, "Synthesis of Positive Real Multivariable Feedback Systems", I*nt. J. Contro*l, vol. 45, no. 3, pp. 817-842, 1987.

Purpose

Two-port or one-port state-space linear fractional transformation.

Syntax

```
[a,b1,b2,c1,c2,d11,d12,d21,d22] =
lftf(A,B1,B2, ,a,b1,b2, )
[aa,bb,cc,dd] =
lftf(a,b1,b2,c1,c2,d11,d12,d21,d22,aw,bw,cw,dw)
[aa,bb,cc,dd] =
lftf(aw,bw,cw,dw,a,b1,b2,c1,c2,d11,d12,d21,d22)
tss = lftf(tss1,tss2)
ss = lftf(tss1,ss2)
ss = lftf(ss1,tss2)
```

Description

`lftf` computes a state-space closed loop transfer function from input u_1 to output (see Figure 2-11, Two-Port Linear Fractional Transformation.), given the open loop transfer function from u_1 to y_1

$$tss1 := \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

and the transfer function $tss2$ from u_2 to y_2

$$tss2 := \left[\begin{array}{c|cc} a & b_1 & b_2 \\ \hline c_1 & d_{11} & d_{12} \\ d_2 & d_{21} & d_{22} \end{array}\right]$$

Either of the systems ($tss1$ or $tss2$) can be "one-port" state space or "two-port." `lftf` also handles the case when some of the A, B or C matrices are empty.

The output variables will be returned in state-space form or, if the inputs are in the optional `mksys` form, then the returned outputs will likewise be in `mksys` form.

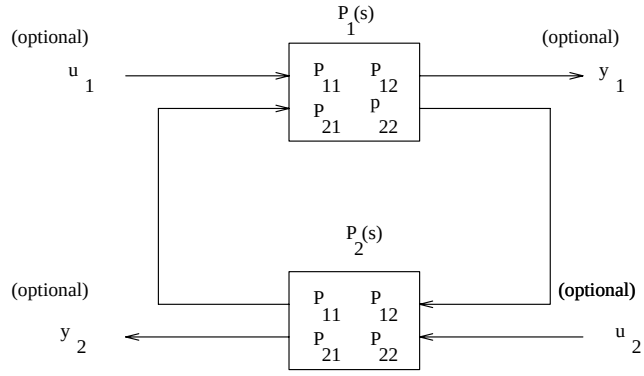


Figure 2-11: Two-Port Linear Fractional Transformation.

Algorithm

`lftf` implements the formulae in [1] for (`aa`, `bb`, `cc`, `dd`), when the second input system is a "one-port" $A,\ B,\ C,\ D$:

$$\left[\begin{array}{cc|c} A+B_2\tilde{D}XC_2 & B_2(\tilde{C}+\tilde{D}XD_{22}\tilde{C}) & B_1+B_2\tilde{D}XD_{21} \\ \tilde{B}XC_2 & A+\tilde{B}XD_{22}\tilde{C} & \tilde{B}XD_{21} \\ \hline C_1+D_{12}\tilde{D}XC_2 & d_{12}(\tilde{C}+\tilde{D}XD_{22}\tilde{C}) & D_{11}+D_{12}\tilde{D}XD_{21} \end{array}\right]$$

where $X = (I - D_{22}\tilde{D})^{-1}$.

The formula for the other cases are similar.

See Also

interc, sectf

References

1 M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, "Synthesis of Positive Real Multivariable Feedback Systems," *Int. J. Contro*l, vol. 45, no. 3, pp. 817-842, 1987.

Purpose

LQG optimal control synthesis.

Syntax

```
[af,bf,cf,df] = lqg(A,B,C,D,W,V)
[ssf] = lqg(ss,w,v)
```

Description

`lqg` computes an optimal computes an optimal controller to stabilize the plant $G(s)$

$$\dot{x} = Ax + Bu + \xi$$

$$y = Cx + Du + \theta$$

and minimize the quadratic cost function

$$J_{LQG} = \lim_{T \to \infty} E\left\{ \int_0^T [x^T u^T] \begin{bmatrix} Q & N_c \\ N_c^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt \right\}$$

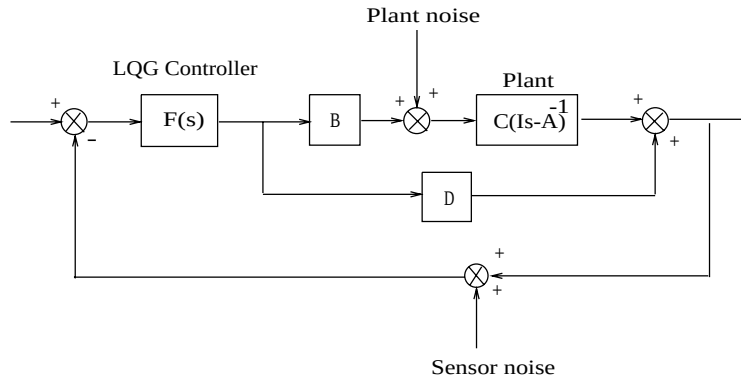


Figure 2-12: LQG Synthesis.

The plant noise ξ and measurement noise θ are white and Gaussian with joint correlation function

$$E\left\{ \begin{bmatrix} \xi(t) \\ \theta(r) \end{bmatrix} [\xi(t)\theta(r)]^T \right\} = \begin{bmatrix} \Xi & N_f \\ N_f^T & \Theta \end{bmatrix} \delta(t-r)$$

The input variables W and V are

$$W = \begin{bmatrix} Q & N_c \\ N_c^T & R \end{bmatrix};\ V = \begin{bmatrix} \Xi & N_f \\ N_f^T & \Theta \end{bmatrix}$$

The LQG controller is returned as *F(s)* := (*af*, *bf*, *cf*, *df)*.

Algorithm

The solution of the LQG problem is a combination of the solutions of *Kalman filtering and full-state feedback* problems based on the so-called *separation principl*e. The individual problems are explained under `lqe` and `lqr` in the *Control System Toolbox*. The final *negative*-feedback controller $u = -F(s)y$ has the form (e.g. [1])

$$F(s) := \left[\begin{array}{c|c} A - K_f C_2 - B_2 K_c + K_f D_{22} K_c & K_f \\ \hline K_c & 0 \end{array}\right]$$

Note that the sign of $F(s)$ is minus that in the function `h2lqg`; this is because by convention `lqg` feedback is negative (i.e, $u = -F(s)y$) while the `h2lqg` is positive (i.e, $u = F(s)y$). The `lqg` feedback can also be realized as a full-state feedback and Kalman filter:

$$u = -K_c\hat{x} \quad \text{(full-state feedback)}$$

$$\dot{\hat{x}} = A\hat{x} + Bu + K_f(y - C\hat{x} - Du) \quad \text{(Kalman feedback)}$$

See Also

h2lqg, hinf, hinfdemo, linf, linfdemo, ltru, ltry

References

[1] M. Athans, “The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design,” *IEEE Trans. Automat. Contr*., AC-16, pp. 529-552, Dec. 1971.

Purpose

LQG loop transfer recovery.

Syntax

```
[af,bf,cf,df,svl] = ltru(A,B,C,D,Kc,Xi,Th,r,w)
[af,bf,cf,df,svl] = ltry(A,B,C,D,Kf,Q,R,q,w)
[ssf,svl] = ltru(ss,Kc,Xi,Th,r,w,svk)
[ssf,svl] = ltry(ss,Kf,Q,R,q,w,svk)
```

Description

Given a plant with transfer function $G(s) = D + C(Is - A)^{-1}B$, `ltru` implements the Doyle-Stein procedure for recovering full-state-feedback loop transfer function

$$L(s) = K_c(Is - A)^{-1}B$$

via a Kalman-filter-reconstructed state feedback with fictitious plant noise. Singular-value Bode plots of the reconstructed-state loop transfer function matrix

$$L_r(s) = F(s)G(s) = [K_c(Is - A + BK_c + K_fC - K_fDK_c)^{-1}K_f]G(s)$$

are computed and displayed for each value of the fictitious noise intensity parameter `r`, so that you can observe the loop-transfer recovery as `r` increases,

$$\lim_{r \to \infty} L_r(j\omega) = L(j\omega)$$

Input variables are:

`A,B,C,D` := $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$, (the plant)

`ss` = optional system matrix form

`Kc` — full-state feedback gain matrix

`Xi` — nominal plant noise intensity

`Th` — measurement noise intensity

`r` — row vector of intensities of fictitious plant noise

ω — frequencies for Bode plot

The row vector `r` contains a set of "recovery" gains (e.g., $[1, 1.e5 \ldots, 1.e15]$). `ltru` will iteratively compute the singular-value Bode plot of the loop gain $F(s)G(s)$ and plot the associated curves in front of the user, until the gain vector `r` runs out of its entries.

The controller $F(s)$ corresponding to the last value of `r` is returned in state-space (`af`,`bf`,`cf`,`df`). All the singular values (MIMO case) or the Nyquist loci (SISO case) will be returned in variable `svl`. The frequency response of $L(j\omega)$ is stored in `svk` which can be passed into `ltru` as an optional input and displayed superimposed on the frequency plots of $L_r(j\omega)$.

`ltry` does the "dual" problem, i.e., recovery of the observer loop transfer function

$$L_q(s) = C(Is - A)^{-1}K_f$$

Examples

Consider the fighter design example in [2]. Apply the LTR procedure `ltry` to the plant model, and let the observer be a Kalman-Bucy filter.

The initial cost and recovery gains used in this example are

$$\Xi = BB^T,\ \Theta = I,\ Q = qC^TC,\ R = I,$$

$$q = [1, 1e5, 1e10, 1e15]$$

The singular value Bode plot is shown in Figure 2-13, Example of LQG/LTR at Plant Output..

The LQG/LTR loop transfer function converges in the limit (as q increases to ∞) to $C\,(Is - A)^{-1}\,K_f$, which is the KBF loop transfer function.

Algorithm

The controller F(s) is computed as

$$F(s) = K_c(Is - A + BK_c + K_fC - K_fDK_c)^{-1}K_f$$

where, in `ltru`, the Kalman filter gain is $K_f = \Sigma C^T\Theta^{-1}$ and Σ satisfies the Kalman filter Riccati equation

$$0 = \Sigma A^T + A\Sigma - \Sigma C^T\Theta^{-1}C\Sigma + \Xi + rBB^T$$

In `ltry` gain `Kc` is computed as $K_c = R^{-1}B^TP$ where P satisfies the full-state Riccati equation

$$0 = PA + A^TP - PBR^{-1}B^TP + Q + qC^TC$$

The theory is documented in [1].

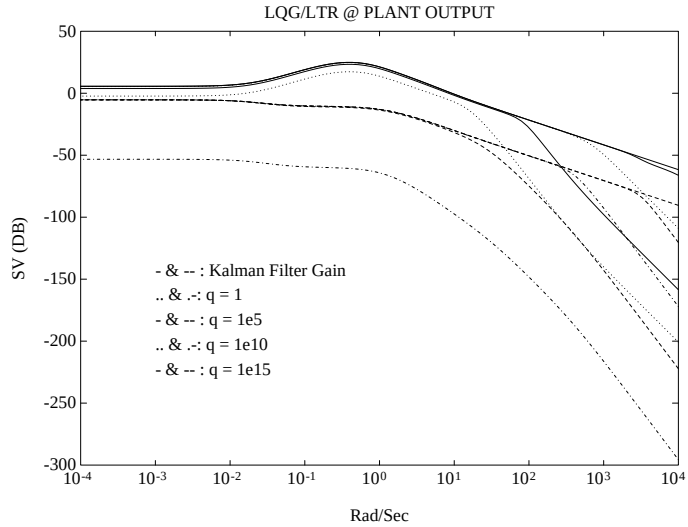


Figure 2-13: Example of LQG/LTR at Plant Output.

Limitations

The `ltru` procedure may fail for nonminimum phase plants or for plants with number of control inputs exceeds the number of measurement outputs. The dual procedure `ltry` may fail for nonminimum phase plants or for plants with fewer inputs than outputs.

See Also

h2lqg, hinf, hinfdemo, lqg, ltrdemo

References

[1] J. Doyle and G. Stein "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," *IEEE Trans. on Automat. Contr.*, AC-26, pp. 4-16, 1981.

[2] M. G. Safonov, A. J. Laub, and G. Hartmann, "Feedback Properties of Multivariable Systems: The Role and Use of Return Difference Matrix," *IEEE Trans. of Automat. Contr.*, AC-26, pp. 47-65, 1981.

Purpose

Create a single MATLAB variable containing all the matrices describing a system, their dimensions and their "standard" names (depending on the type of system). `mksys` implements the *Robust Control Toolbox* system data structure used to simplify user interaction with functions whose input or output variables include state-space systems, transfer function matrices, or other types of systems.

Syntax

```
S = mksys(a,b,c,d)
S = mksys(v1,v2,v3, vn, TY)
[VARS,N] = vrsys(NAM)
[I,TY,N] = issystem(S)
```

Description

`mksys` packs several matrices describing a system of type *TY* into a MATLAB variable *S*, under "standard" variable names determined by the value of the string `TY` as follows:

Table 1-1 Table of System Names

Type	$V_1, V_2, \ldots, V_n$	Description
`'ss'`	`(a,b,c,d,ty)`	Standard state-space (default)
`'des'`	`(a,b,c,d,e,ty)`	Descriptor system
`'tss'`	`(a,b1,b2,c1,c2,d11,d12,d21,d22,e,ty)`	Two port state-space
`'tdes'`	`(a,b1,b2,c1,c2,d11,d12,d21,d22,e,ty)`	Two-port descriptor
`'gss'`	`(sm,dimx,dimu,dimy,ty)`	General state-space
`'gdes'`	`(e,sm,dimx,dimu,dimy,ty)`	General descriptor
`'gpsm'`	`(psm,deg,dimx,dimu,dimy,ty)`	General polynomial system matrix
`'tf'`	`(num,den,ty)`	Transfer function
`'tfm'`	`(num,den,m,n,ty)`	Transfer function matrix
`'imp'`	`(y,ts,nu,ny)`	Impulse response

The value of `TY` is packed into `S` under the name `ty`.

The function `branch` recovers the individual matrices packed into the system `S`; e.g.,

```
[a,b,c,d]=branch(ssg);
```

See `branch` and `tree` for further details.

`vrsys` returns a string `VARS` and an integer `N` where `VARS` contains the list (separated by commas) of the `N` names of the matrices associated with a system described by the string name *NAM*. Valid values for the string `NAM` are strings of the form

```
[TY '_' SUF]
```

where `SUF` is a suffix string which is appended to the standard variable names determined from the table above by the string `TY`. For example, the MATLAB command `[var,n] = vrsys('ss_g');` returns the following:

```
var
   = 'ag,bg,cg,dg'
n
   = 4.
```

`issystem` returns a value for `i` of either 1 (true) or 0 (false) depending on whether or not the variable `S` is a system created by the function `mksys`. Also returned is the type of system `TY` and the number `N` of variable names associated with a system of type `TY`, except that if `S` is not a system then `TY = [ ]`; and `N = 0`.

Examples

The following MATLAB commands pack and recover the state-space matrices of any system easily.

```
% Pack the state-space (ag,bg,cg,dg) into ssg:
% (no need to specify 'ss' in this case)
ssg = mksys(ag,bg,cg,dg);
% Pack a massive two-port state-space into tss:
tss = mksys(A,B1,B2,C1,C2,D11,D12,D21,D22,'tss'');
% Pack the descriptor state-space (ag,bg,cg,dg,eg)
% into desg:
desg = mksys(ag,bg,cg,dg,eg,'des');
```

Now, you can extract any matrix or matrices out of the system variable using branch:

```
% Extract  ag, dg  out of system variable
%  ss_g :
[ag,dg] = branch(ssg,'ag,dg');

% Extract  D22,C1,C2  out of system variable
%  tss_ :
[D22,C1,C2] = branch(tss,'D22,C1,C2');
% Extract  ag,eg  out of system variable
%  des_g :
[ag,eg] = branch(desg,'ag,eg');
```

See Also tree, branch, graft, issystem, istree, vrsys

Purpose

Compute an upper bound on the structured singular value using multiplier approach.

Syntax

```
[mu,ascaled,logm,x] = muopt(a)
[mu,ascaled,logm,x] = muopt(a,k)
```

Description

`muopt(A)` produces the scalar upper bound `mu` on the structured singular value (ssv) of a $p \times q$ matrix A having n real or complex uncertainty blocks using the optimal multiplier method.

The optional input `k` records the uncertainty block sizes with default value $k = ones(p, 2)$. `k` can be an n by *1* or n by *2* matrix whose rows are the uncertainty block sizes for which the SSV is to be evaluated. If only the first column of k is given then each of the individual uncertainty blocks is taken to be square, as if $k(:, 1) = k(:, 2)$. Real uncertainty (must be scalar) is indicated by multiplying the corresponding row of K by minus one, e.g., if the second uncertainty block is real then set K(2)=-1.

`mu` returns an upper bound on the real/complex structured singular value of A. The output `ascaled` returns the multiplier-scaled A-matrix

$$A_{scaled} = \mu(I - \tilde{A}_{scaled})(I + \tilde{A}_{scaled})^{-1}$$

where $A_{scaled} = M^{1/2}(\mu I - A)(\mu I + A)^{-1}M^{-1/2*}$ and M is the optimal diagonal generalized Popov multiplier scaling matrix. The output `logm` returns $log(diag(M^{1/2}))$, a complex vector of length p. The multiplier matrix M is related to the D,G–scales of [3] by $M = D + jG$.

`x` returns a normalized eigenvector associated with the smallest eigenvalue of the positive semidefinite matrix $A_{scaled} + \tilde{A}^{*}_{scaled}$.

Algorithm

`muopt` is based on the optimal generalized Popov multiplier theory of Safonov and Lee [1] and uses the computational algorithm of Fan and Nekooie [2]. The upper bound of μ returned is found as the solution to the optimization

$$\min_{M \in \mathrm{M}} \ \mu$$

subject to

$$(\tilde{A}_{scaled} + \tilde{A}^{*}_{scaled} \geq 0)$$

where A_{scaled}= *M½(μI – A)(μI + A)-1M-½** and M is the set of block diagonal generalized Popov multiplier for the uncertainty structure determined by k. This results in the returned value of A_{scaled} satisfying $\mu = \bar{\sigma}(A_{scaled})$. When μ=1, the Popov scaling matrix M is related to the *D,G*–scales of [3] by $M = D + jG$.

Note that in the case in which all uncertainties are complex, the diagonal multiplier matrix M is real and A_{scaled} becomes simply $_{scaled} = M^{½} A M^{-½}$. In this case the optimal μ is computed via the diagonally scaled singular value optimization $= \min_{M \in \mathrm{M}} \bar{\sigma}(M^{½} A M^{-½}$.

Limitations

The algorithm in general produces a smaller upper bound on μ than `perron`, `psv` and `osborne`, but `muopt` requires significantly greater computation time than these other functions.

See Also

perron, psv, osborne, ssv

References

[1] M. G. Safonov, and Peng-Hin Lee, "A Multiplier Method for Computing Real Multivariable Stability Margins," *Proc. IFAC World Congress*, Sydney, Australia, July 1993.

[2] M.K.H. Fan and B. Nekooie, "An Interior Point Method for Solving Linear Matrix Inequality Problems," *SIAM J. Contr. and Optim.*, to appear.

[3] M.K.H. Fan, A. Tits and J. Doyle, Robustness in the Present of Mixed Parametric Uncertainty and Unmodelled Dynamics, IEEE Trans. on Autom. Contr., vol. AC-36, no. 1, pp. 25-38, January 1991.

Purpose

μ synthesis procedure.

Syntax

```
[acp, ,dcp,mu,logd,ad, ,dd,gam] = musyn(A,B1,B2, ,D22,w)
[acp, ,dcp,mu,logd,ad, ,dd,gam] = ...
     musyn(A,B1,B2, ,D22,w,gammaind,aux,logd0,n,blksz,flag)
[sscp,mu,logd,ssd,gam] = musyn(tss,w)
[sscp,mu,logd,ssd,gam] = ...
     musyn(tss,w,gammaind,aux,logd0,n,blksz,flag)
```

Description

Given a two-port plant state space (in its regular form `mksys` data form `tss`):

$$P(s) := \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

`musyn` automates the μ synthesis $D-F$ iteration procedure that iteratively applies `hinfopt` and `fitd` to find a control law

$$F(s) := \left[\begin{array}{cc} A_{cp} & B_{cp} \\ C_{cp} & D_{cp} \end{array}\right]$$

and a diagonal scaling matrix$D(s) = diag(d_1(s)I_{k_1},\ldots,d_n(s)I_{k_n})$ that attempts to satisfy the *robust performance* objective

$$\left\| DT_{y_1u_1}D^{-1} \right\|_\infty < 1$$

Here the identity matrices $I_{k_1},\ldots,I_{k_n}$ are of dimensions determined by the input variable `blksz` described below.

The resulting structured singular value upper bound μ is returned together with the control law $F(s)$ (`sscp`). The variable `logd` returns as its rows the log magnitude frequency response of the diagonal entries of the diagonal scaling matrix $D(s)$.

Also returned is a state-space realization of the $D(s)$ used in the `hinfopt` portion of the last $D-F$ iteration along with the corresponding optimal value

of γ (gam) from the hinfopt γ-iteration. See the documentation of hinfopt for further details.

The input variable w contains the frequency at which the structured singular value μ is to be evaluated. The remaining input variables gammaind, aux, logd0, n, blksz, and flag are optional. The variable logd0 allows you to specify an initial guess for logd (default $D(s) = I$). See the documentation for fitd for an explanation of n, blksz, and flag and their default values. The documentation for hinfopt explains the uses and defaults for the optional input variables gammaind and aux. If an optional variable is given as the empty matrix [ ], then it assumes its default value.

Examples

Following are the MATLAB input commands for a simple μ–synthesis problem:

```
% PLANT DATA:
a=2;         b1=[.1,-1];         b2=-1;
c1=[1;.01];  d11=[.1,.2;.01,.01];d12=[1; 0];
c2=1;        d21=[0,1];          d22=3;

tss=mksys(a,b1,b2,c1,c2,d11,d12,d21,d22,'tss');
w = logspace(-2,1); % FREQUENCY VECTOR

% Starting μ Synthesis D-F Iterations:
[sscp,mu,logd0] = musyn(tss,w);

% DISPLAY OPTIMAL MU PLOTS:
loglog(w,mu);

% Now improve using frequency dependent D(s):
[sscp,mu1,logd1] = musyn(tss,w,[ ],[ ],logd0,1);

% DISPLAY OPTIMAL MU PLOTS:
loglog(w,mu,w,mu1);
```

The foregoing example illustrates the basic μ-synthesis iteration. In practice, you will generally prefer to use a constant ($n = 0$) diagonal scaling matrix $D(s)$ because it leads to a much lower order control law. It may also be necessary to experiment with the frequency range w, adjusting it so that it coincides roughly with the frequency range over which the value of μ returned by ssv is

unacceptably large. In "Design Case Studies" of the *Tutorial* a more detailed μ-synthesis example is provided.

Algorithm

The $D - F$ iteration procedure is as follows [1, 2]:

Initialize: If the input variable `logd0` is present, go to Step 3; otherwise set $D(s) = I$ and continue.

1. Use the $\mathbf{H}^\infty$ control method (`hinf`) to find an $F(s)$ which minimizes the cost function $\left\| DT_{y_1u_1}D^{-1} \right\|_\infty$.
2. Use `ssv` to estimate the structured singular value Bode plot and the corresponding frequency response of `logd`. The function `ssv` computes an upper bound on the structured singular value μ and produces the corresponding $D(s)$ by attempting, at each frequency ω, to solve the minimization $\mu = \min_{D(j\omega)} \bar{\sigma}(D(j\omega)T_{y_1u_1}(j\omega)D^{-1}(j\omega))$.
3. If the cost is small enough stop; otherwise continue.
4. Using `fitd`, curve fit an order `n` rational approximation to each of the diagonal elements of the $D(s)$ obtained in Step 2 and, using `augd`, augment the plant `tss` with the fitted $D(s)$. Go to Step 1.

See the *Tutorial* chapter "Design Case Studies" for further discussion.

The order of the μ-synthesis controller can be large when a frequency dependent $D(s)$ is employed. The order in general is equal to the order of the plant plus twice the order of $D(s)$. For example, if the plant `tss` has six states and $D(s)$ has six states, then the order of the μ-synthesis control law will be18, i.e., three times the order of the original plant. This highly limits the potential of practical applications and hardware implementations. Therefore, it is desirable to use as low an order $D(s)$ as is possible; preferably a constant $D(s)$. The combined $D - F$ iteration procedure is not *convex*, so in general the μ synthesis controller resulting from the $D - F$ iteration is suboptimal.

See Also

hinf, augd, fitd, fitgain, ssv

References

[1] M. G. Safonov, "L∞ Optimization vs. Stability Margin," *Proc. IEEE Conf. on Decision and Control*, San Antonio, TX, December 14-16, 1983.

[2] J. C. Doyle, "Synthesis of Robust Controllers and Filters," *Proc. IEEE Conf. on Decision and Control,* San Antonio, TX, December 14-16, 1983.

normhinf, normh2

Purpose

Compute the $\mathbf{H}^\infty$ norm and $\mathbf{H}^2$ norm of a system.

Syntax

```
[h2n] = normh2(a,b,c,d)
[hinfn] = normhinf(a,b,c,d,aux)
[hinfn] = normhinf(a,b,c,d)
[h2n] = normh2(ss)
[hinfn] = normhinf(ss,aux)
[hinfn] = normhinf(ss)
```

Description

Given a stable system $G(s) := (A, B, C, D)$, `normh2` computes its $\mathbf{H}^2$ norm and `normhinf` computes its $\mathbf{H}^\infty$ norm.

The computation of $\|G\|_\infty$ requires a search, therefore an optional input variable of `aux` overrides default values for initializing the search

$$aux = [tol \;\; gammax \;\; gammin]$$

where *tol* terminates the search process (default=0.001), and *gammax* and *gammin* are initial guesses for upper and lower bounds on $\|G\|_\infty$. Defaults for *gammax* and *gammin* are

$$gammin = \max[\bar{\sigma}(D), \bar{\sigma}_H(G)]$$

$$gammax = \bar{\sigma}(D) + 2\sum_{i=1}^{n} \sigma_{H_i}(G)$$

where the $\sigma_{H_i}(G)$'s are the Hankel singular values of G(s). The bounds may be found among the results in [1, 2].

Algorithm

Consider a strictly proper, stable $G(s) := (A, B, C, 0)$. The two norm of $G(s)$ is

$$\|G\|_2 = \text{trace}(CPC^T) = \text{trace}(B^TQB)$$

where P is the controllability grammian of (A, B) and Q is the observability grammian of (C, A) computed by `gram`.

For computing the $\mathbf{H}^\infty$ norm, consider the following fact:

Given a $\gamma > 0$, $\|G\|_\infty < \gamma$*, if and only if the right spectral factorization (cf.* `sfr.m`*) Hamiltonian matrix*

$$H_\gamma = \begin{bmatrix} A + BR^{-1}D^T C & -BR^{-1}B^T \\ C^T(I + DR^{-1}D^T)C & -(A + BR^{-1}D^T C)^T \end{bmatrix}$$

has no imaginary eigenvalues; here $R = \gamma^2 I - D^T D > 0$.

`normhinf` uses a standard binary search to find the optimal γ similar to the algorithm used in `hinfopt`.

See Also

gram, hinf, hinfopt

References

[1] K. Glover, "All Optimal Hankel Norm Approximations of Linear Multivariable Systems, and Their L^∞ -Error Bounds," *Int. J. Control*, vol. 39, no. 6, pp. 1145-1193, 1984.

[2] S. Boyd, V. Balakrishnan, and P. Kabamba, "In Computing the $\mathbf{H}^\infty$ Norm of a Transfer Matrix," *Mathematics of Control, Signals, and Systems*, 1988.

obalreal

Purpose

Balanced realization via B. C. Moore's algorithm.

Syntax

```
[abal,bbal,cbal,g,t] = obalreal(a,b,c)
```

Description

This M-file does functionally the same thing as `balreal`, but the balanced reachability and observability grammians (P and Q) are *ordered* and $P = Q = diag(g)$. The similarity transformations are accumulated in the variable `t`. Moore's [1] k^{th}-order reduced model $G_k(s)$ can be simply extracted from the balanced state-space

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D \end{array}\right] := \left[\begin{array}{c|c} Abal(1{:}k,\,1{:}k) & Bbal(1{:}k,\,:) \\ \hline Cbal(:,1{:}k) & D \end{array}\right]$$

`obalreal` is an M-file that implements the algorithm of [1]. `Balreal` uses the Cholesky decomposition to find the associated left and right eigenspaces of PQ. `Obalreal` is superior to the existing `balreal` M-file in two ways:

1. Grammians are ordered.
2. Transformations are carried out using reliable SVD's.

What makes balanced realization important is not only its structure but also $L\infty$ the norm error bound associated with its k^{th} order reduced model ([2] and [3]):

$$\left\| G(s) - G_k(s) \right\|_\infty \le 2\left(\sum_{i=k+1}^{n} \sigma_i \right)$$

Therefore, you can anticipate how big an error the reduced model will have before actually doing the model reduction.

Limitations

The original system (A, B, C, D) has to be *minimal*, otherwise the balancing algorithm in either `obalreal` [1] or `balreal` [4] breaks down. See `schmr` and `balmr` for robust methods for computing G_k without balancing.

See Also

balreal, balmr, schmr, schbal, ohklmr, ohkapp, reschmr

References

[1] B. C. Moore, “Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction,” *IEEE Trans. on Automat. Contr.*, AC-26, pp. 17-31, February 1981.

[2] D. F. Enns, “Model Reduction with Balanced Realizations: An Error Bound and Frequency-Weighted Generalization,” *Proc. IEEE Conf. on Decision and Control*, Las Vegas, NV, Dec. 12-14, 1984.

[3] K. Glover, “All Optimal Hankel Norm Approximations of Linear Multivariable Systems, and Their $\mathbf{L}^{\infty}$ -error Bounds,” *Int. J. Control*, vol. 39, no. 6, pp. 1145-1193, 1984.

[4] A. J. Laub, M. T. Heath, C. C. Page, and R. C. Ward, “Computation of balancing transformations and other applications of simultaneous diagonalization algorithms,” *IEEE Trans. on Automat. Contr.*, AC-32, pp. 115-122, 1987.

[5] M. G. Safonov and R. Y. Chiang, “A Schur Method for Balanced Model Reduction,” *IEEE Trans. on Automat. Contr.*, vol. AC-34, no. 7, pp. 729-733, July 1989.

Purpose

Optimal Hankel minimum degree approximation without balancing.

Syntax

```
[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,Type)
[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,Type,in)
[am,bm,cm,dm,totbnd,svh] = ohklmr(a,b,c,d,Type)
[am,bm,cm,dm,totbnd,svh] = ohklmr(a,b,c,d,Type,in)
[ssx,ssy, ] = ohkapp(ss, )
[ssm, ] = ohklmr(ss,..)
```

Description

`ohkapp` computes the k^{th} order optimal *Hankel minimum degree approximation* (OHMDA)

$$G_x(s) = C_x(Is - A_x)^{-1}B_x + D_x$$

of a possibly non-minimal n^{th} order stable system

$$G(s) = C(Is - A)^{-1}B + D$$

such that G_x is stable and

$$\|G - G_x\|_\infty \le totbnd,$$

$$totbnd = 2\sum_{i=k+1}^{n} \sigma_i$$

where $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_n$ denote Hankel singular values of $G(s)$, i.e., the square roots of eigenvalues of PQ, where P and Q are the reachability and observability grammians of (A, B, C, D).

An anticausal $G_y(s)$ is also returned in (A_y, B_y, C_y, D_y). Together $G_x(s)$ and $G_y(s)$ possess the following important property

$$\|G - G_x - G_y\|_\infty \le \sigma_{k+1}$$

`[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,1,0)` computes the *zeroth order* OHMDA, i.e., the *anticausal* OHMDA of a stable system, which is an important intermediate step of the $\mathbf{L}^\infty$ control synthesis.

Variable aug contains the following information:

```
aug(1,1) = σ1
aug(1,2) = number of states removed
aug(1,3) = totbnd
aug(4:4+n-1) = [σ1, σ2, ..., σn].
```

ohklmr also computes the k_{th} order OHMDA, but allows the system to be unstable. It works by applying ohkapp to the stable and antistable parts of $G(s)$ (obtained via stabproj), then applying addss. Totbnd returns the $\mathbf{L}^\infty$ norm error bound of the approximation. Variable svh returns the Hankel singular values σ_i^- of $[G(s)]-$ (stable part) and σ_i^+ of $[G(-s)]+$ (reversed antistable part), i.e.,

$$\text{svh} = [\sigma_1^- ,\sigma_2^- ,\ldots,\sigma_m^- ,\sigma_1^+ ,\sigma_2^+ , \sigma_{n-m}^+]^T$$

where m denotes the number of stable roots, $n-m$ denotes the number of unstable roots.

Both ohkapp and ohklmr provide three options:

1. Type = 1, in = k, size of reduced order model.
2. Type = 2, in = *tol*, find a k^{th} order reduced model such that the total error totbnd is less than *tol*.
3. Type = 3, display svh and prompt for $k + 1$. In this case, there is no need to assign a value for in.

Algorithm

ohkapp and ohklmr employ the algorithm described in [3], which is a "basis-free" descriptor system implementation of the OHMDA. The descriptor formulae bypass the numerically ill-conditioned balanced realization step required by the earlier state-space formulae [1, 2].

ohklmr uses the M-function stabproj to split $G(s)$ into the sum of stable and antistable parts, then applies ohkapp to each part.

See Also

balmr, mrdemo, obalreal, bstschmr, schmr, stabproj

References

[1] M. Bettayeb, L. M. Silverman and M. G. Safonov, “Optimal Approximation of Continuous Time Systems,” *IEEE Conf. on Decision and Control*, Albuquerque, NM, Dec. 10-12, 1981.

[2] K. Glover, “All Optimal Hankel Norm Approximation of Linear Multivariable Systems, and Their $\mathbf{L}^\infty$-error Bounds,” *Int. J. Control*, vol. 39, no. 6, pp. 1145-1193, 1984.

[3] M. G. Safonov, R. Y. Chiang, and D. J. N. Limebeer, “Hankel Model Reduction without Balancing — A Descriptor Approach,” *Proc. IEEE Conf. on Decision and Control*, Los Angeles, CA, Dec. 9-11, 1987, also “Optimal Hankel Model Reduction for Nonminimal Systems,” *IEEE Trans. on Automat. Contr.*, vol. 34, no. 4, pp. 496-502, 1990.

Purpose

Compute an upper bound on the structured singular value via the Osborne method.

Syntax

```
[mu,ascaled,logd] = osborne(a)
[mu,ascaled,logd] = osborne(a,k)
```

Description

`osborne` computes a block-diagonal scaling that minimizes the Frobenius norm of a p by q matrix `a`. The maximum singular value of the scaled matrix is returned as `mu`; it is a scalar upper bound on the Structured Singular Value (SSV) of the matrix `a`.

Also returned are the diagonally scaled matrix `ascaled` and the logarithm of the Osborne diagonal scaling `logd`.

The optional input `k` records the uncertainty block sizes with default value $k = ones(p, 2)$. The variable `k` can be an n by 1 or n by 2 matrix whose rows are the uncertainty block sizes for which the SSV is to be evaluated. If only the first column of k is given then each individual uncertainty block is taken to be square, as if $k(:, 1) = k(:, 2)$.

Algorithm

The Osborne iteration algorithm is as follows:

1. Partition the given matrix a according to its pre-determined uncertainty block size `k`.
2. Form the n by n matrix F whose elements are the largest singular values of the blocks of the matrix A.
3. Compute the diagonal scaling D that minimizes the Frobenius norm of DFD^{-1} via the following algorithm:

```
% Initialize D scaling
D = eye(n);
for i = 1 : n
   while abs(D(i)   1) < 1.e   6
      offrow(i) = sum(norm(off-diagonal terms of ith row));
      offcol(i) = sum(norm(off-diagonal terms of ith column));
      D(i) = offrow (i)/o f f col(i);
      F(i, :) = F(i, :)/D(i);
      F(:, i) = F(:, i * D(i);
   end
end
```

Limitations

The Osborne algorithm is ill-posed when the matrix F is reducible [1]; as sometimes is the case when some of the matrix' entries are zero. This problem is solved in `osborne` by slightly perturbing the zero entries, to create a nearby irreducible matrix.

Examples

Following are some simple examples of problems involving the difficult reducible case that are easily handled by the *Robust Control Toolbox* implementation of the `osborne` command:

```
% A reducible case as compared to sigma
A = eye(10); A(1,10) = 100000;
[mu,Ascaled,logd] = osborne(A);
mu % Display answer mu

% Another reducible case as compared to sigma
A = eye(8);
A(1,3) = 100000; A(4,8) = 500000;
[mu,Ascaled,logd] = osborne(A);
mu % Display answer mu
```

See Also

muopt, psv, perron, ssv, sigma

References

[1] E. E. Osborne, "On Preconditioning of Matrices," *J. of Assoc. of Computing Machinery*, vol. 7, pp. 338-345, March, 1960.

Purpose

Compute an upper bound on the structured singular value via the Perron eigenvector method.

Syntax

```
[mu] = perron(a)
[mu] = perron(a,k)
[mu,ascaled,logd] = psv(a)
[mu,ascaled,logd] = psv(a,k)
```

Description

`perron` produces the Perron eigenvalue for a given real or complex p by q matrix. This value serves as a scalar upper bound "`mu`" on the Structured Singular Value (SSV).

`psv` computes a tighter SSV upper bound `mu` via the formula

$$mu = \bar{\sigma}[A_{scaled}] = \bar{\sigma}[D_p A D_p^{-1}]$$

where D_p = d*iag*(*exp(logd)*) is the Perron optimal diagonal scaling. In addition, `psv` returns the log magnitude of the optimal diagonal scaling `logd` in a column vector, and the scaled matrix A_{scaled} is returned in `ascaled`.

The optional input `k` records the uncertainty block sizes with default value k = *ones*(q, 2) corresponding to 1 by 1 uncertainty blocks. `k` can be an n by 1 or n by 2 matrix whose rows are the uncertainty block sizes for which the SSV is to be evaluated. If only the first column of k is given, then each individual uncertainty block is taken to be square, as if k(:, 1) = k(:, 2).

Algorithm

The values of `mu` and `logd` are found by examining the eigenvalues and eigenvectors of the n by n nonnegative square matrix F formed from by A replacing each block of A (as defined by the partitioning `k`) by its greatest singular value. For any given positive square matrix (i.e., matrix with positive entries), there exists a positive real eigenvalue λ_p of multiplicity one whose magnitude is greater than the real part of any other eigenvalue,:

$$\lambda_p = \max_i \ Re(\lambda_i(F))$$

This real eigenvalue λ_p is called the *Perron eigenvalue* of F, and its left and right eigenvectors, denoted as y_p and x_p respectively, are called *Perron eigenvectors*.

In 1982, Safonov [1] showed that the Perron eigenvalue is a good upper bound on the structured singular value μ, i.e.,

$$\mu(A) \le \inf_{D \in D} \left\| DAD^{-1} \right\|_\infty \le \left\| D_p A D_p^{-1} \right\|_\infty \le \lambda_p(F)$$

$D_p \in D$ is the Perron optimal scaling matrix

$$D_p = diag(d_1,\ d_2,\ \ldots,\ d_n)$$

and

$$d_i = \left| \sqrt{\frac{y_{p_i}}{x_{p_i}}} \right|$$

Moreover, the above inequalities become equalities when $A = F$ so that, in the case in which A is a positive matrix and the uncertainty blocks are scalar, the Perron eigenvalue bound on μ is tight.

For reducible matrices the Perron optimal scaling matrix D_p can be singular, which would lead to numerical instability if corrective action were not taken. This problem is solved in `psv` (in the same fashion as it is in the function `osborne`) by very slightly perturbing the matrix F to a nearby irreducible matrix which has a slightly greater Perron eigenvalue. See `osborne` for further details and examples. Perturbation is not required with the `perron` function since D_p is not computed.

As compared to Osborne or nonlinear programming techniques, Perron eigenvector algorithms implemented by `perron` and `psv` require no iteration and so tend to be faster.

Examples

Several problems can be solved via the following psv commands, where most careless algorithms fail:

```
% An reducible case as compared to sigma
A = eye(10); A(1,10) = 100000;
[mu,Ascaled,logd] = psv(A);
s1 = max(svd(A)); [s1, mu],

% Another reducible case as compared to sigma
A = eye(8);
A(1,3) = 100000; A(4,8) = 500000;
[mu,Ascaled,logd] = psv(A);
s1 = max(svd(A)); [s1, mu],
```

See Also

muopt, osborne, ssv, sigma, dsigma

References

[1] M. G. Safonov, "Stability Margins for Diagonally Perturbed Multivariable Feedback Systems," *IEE Proc.*, vol. 129, Part D, pp. 251-256, 1982.

Purpose

Real and ordered eigenstructure decomposition.

Syntax

```
[xr,d] = reig(a)
```
[xr,d] = reig(a,Opt)

Description

`Reig` produces a "real" and "ordered" eigenstructure decomposition such that

$$X_r^{-1} A X_r = D$$

where X_r is a set of "real" eigenvectors that span the same eigenspace as the complex ones. D is a real block diagonal matrix with real eigenvalue appearing as 1×1 block and/or complex eigenvalue $a + jb$ appearing as a 2×2 block

$$\begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

Two types of ordering are available

- `Opt = 1` — eigenvalues are ordered by real parts (default).
- `Opt = 2` — eigenvalues are ordered by their magnitudes.

Algorithm

The k^{th} real eigenvector pair $xr(:, k{:}k{+}1)$ is

$$xr(:,\ k{:}k+1) = [\ \text{real}(xr(:,\ k))\quad \text{imag}(xr(:,\ k))\ ]$$

See Also

eig, cschur

riccond

Purpose

Condition numbers of continuous algebraic Riccati equation.

Syntax

```
[tot] = riccond(a,b,qrn,p1,p2)
```

Description

`Riccond` provides the condition numbers of continuous Riccati equation. The input variable `qrn` contains the weighting matrix

$$qrn = \begin{bmatrix} Q & N \\ N' & R \end{bmatrix}$$

for the Riccati equation

$$A'P + PA - (PB + N)R^{-1}(B'P + N') + Q = 0$$

where $P = P2/P1$ is the positive definite solution of ARE, and $[P2; P1]$ spans the stable eigenspace of the Hamiltonian

$$H = \begin{bmatrix} A_c & -R_c \\ -Q_c & -A'_c \end{bmatrix} = \begin{bmatrix} (A - BR^{-1}N') & -BR^{-1}B' \\ -(Q - NR^{-1}N') & -(A - BR^{-1}N') \end{bmatrix}$$

Several measurements are provided:

- Frobenius norms `norAc`, `norQc`, `norRc` of matrices A_c, Q_c, and Rc, respectively.
- condition number `conR` of R.
- condition number `conP1` of $P1$.
- Arnold and Laub's Riccati condition number (`conArn`) [1].
- Byers condition number (`conBey`) [2].
- residual of Riccati equation (`res`).

The output variable `tot` puts the above measurements in a column vector

```
tot= [norA,norQ,norRc,conR,conP1,conBey,res]'
```

For an ill-conditioned problem, one or more of the above measurements could become large. Together, these measurements give a general sense of the Riccati problem conditioning issues.

Algorithm

Arnold and Laub's Riccati condition number is computed as follows [1]:

$$conArn = \frac{\|Q_c\|_F}{\|P\|_F sep(A'_{cl'} - A_{cl})}$$

where $A_{cl} = A_c - R_c P$ and

$$sep(A'_{cl'} - A_{cl}) = \min_i \sigma_i [I_n \otimes A'_{cl} + A'_{cl} \otimes I_n]$$

Byers' Riccati condition number is computed as [2]

$$conBye = \frac{\|Q_c\|_F + 2\|A_c\|_F \|P\|_F + \|R_c\|_F \|P\|_F^2}{\|P\|_F sep(A'_{cl'} - A_{cl})}$$

See Also

are, aresolv, daresolv, driccond

References

[1] W. F. Arnold, III and A. Laub, "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations," *Proceedings of the IEEE*, Vol. 72, No. 12, Dec. 1984.

[2] R. Byers, "Hamiltonian and Symplectic Algorithms for the Algebraic Riccati Equation," Ph.D. dissertation, Dept. of Comp. Sci., Cornell University, Ithaca, NY, 1983.

Purpose

State-space sector bilinear transformation.

Syntax

```
[ag,bg1, ,dg22,at,bt1, ,dt21,dt22] = sectf(af,bf1, ,df22,secf,secg)
[ag,bg,cg,dg,at,bt1, ,dt21,dt22] = sectf(af,bf,cf,df,secf,secg)
[tssg,tsst] = sectf(tssf,secf,secg)
[ssg,tsst] = sectf(ssf,secf,secg)
```

Description

`sectf` may be used to transform conic-sector control system performance specifications into equivalent $\mathbf{H}\infty$-norm performance specifications. Given a two-port state-space system $F(s)$:= *tssf*, `sectf` computes a linear-fractionally-transformed two-port state-space system $G(s)$:= *tssg* such that the channel-one Input-Output (I/O) pairs (u_{g1}, y_{g1}) of $G(s)$ are in sector *secg i*f and only if the corresponding I/O pairs of $F(s)$ are in *secf*. Also computed is a two-port system $T(s)$ such that $G(s)$ is obtained via the MATLAB command `tssg=lftf(tsst,tssf)`.

Input variables are:

The open loop plant $F(s)$

`tssf`	`mksys(af,bf1,bf2,cf1,cf2,df11,df12,df21,df22,'tss'),`
or	
`ssf`	`mksys(af,bf,cf,df)`

Conic sector specifications for $F(s)$ and $G(s)$, respectively, in one of the following forms:

`secg, secf`	`secg, secf`	Sector inequality:
	`[-1,1]` or `[-1;1]`	$\|y\|^2 \le \|u\|^2$
	`[0,Inf]` or `[0;Inf]`	$0 \le Re[y^* u]$
	`[A,B]` or `[A;B]`	$0 \ge \mathrm{Re}[(y - Au)^*(y - Bu)]$
	`[a,b]` or `[a;b]`	$0 \ge \mathrm{Re}[(y - diag(a)u)^*(y - diag(b)u)]$

`S`	$0 \geq \mathrm{Re}[(S_{11}u + S_{12}y)^*(S_{21}u + S_{22}y)]$
`tsss`	$0 \geq \mathrm{Re}[(S_{11}u + S_{12}y)^*(S_{21}u + S_{22}y)]$

where `A,B` are scalars in [–∞, ∞] or square matrices; `a,b` are vectors; `S=[S11 S12;S21,S22]` is a square matrix whose blocks `S11,S12,S21,S22` are either scalars or square matrices; `tsss` is a two-port system `tsss=mksys(a,b1,b2, ,'tss')` with transfer function

$$S(s) = \begin{bmatrix} S_{11}(s) & S_{12}(s) \\ S_{21}(s) & S_{22}(s) \end{bmatrix}$$

Output variables `are:`

	The transformed plant *G(s)*:
`tssg`	`mksys(ag,bg1,bg2,cg1,cg2,dg11,dg12,dg21,dg22,'tss'),`
	or
`ssg`	`mksys(ag,bg,cg,dg)`
	The linear fractional transformation *T*(*s*):
`tsst`	`mksys(at,bt1,bt2,ct1,ct2,dt11,dt12,dt21,dt22,'tss')`

`Here tssf, tsst,` and `tssg` are two-port state-space representations of *F*(*s*), *T*(*s*), and *G*(*s*).

If the input *F*(*s*) is specified as a standard state-space system `ssf`, then the sector transformation is performed on *all* channels of *F*(*s*), so that the output *G*(*s*) will likewise be returned in standard state-space form `ssg`.

Examples

The statement G(jω) inside *sector*[–1, 1] is equivalent to the $\mathbf{H}^\infty$ inequality

$$\sup_{\omega} \bar{\sigma}(G(j\omega)) = \|G\|_\infty \leq 1$$

Given a two-port open-loop plant *P*(*s*) := *tssp1*, the command `tssp1 = sectf(tssp,[0,Inf],[-1,1])` computes a transformed *P*(*s*) := *tssp*1 such that an $\mathbf{H}^\infty$ feedback *K*(*s*), which places the closed-loop transformed

system inside *sector*[–1, 1], also places the original system inside *sector*[0, ∞]. See Figure 2-14, Sector Transform Block Diagram..

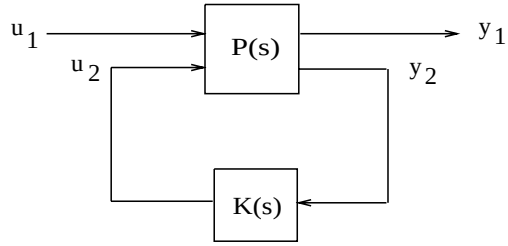


Figure 2-14: Sector Transform Block Diagram.

Here is a simple example of the sector transform.

$$P(s) = \frac{1}{s+1} \in \text{sec}tor[-1, 1] \rightarrow P_1(s) = \frac{s+2}{s} \in \text{sec}tor[0,\infty].$$

You can compute this by simply executing the following commands:

```
[A,B,C,D] = tf2ss(1,[1 1]);
[a,b,c,d] = sectf(A,B,C,D,[-1,1],[0,Inf]);
```

The Nyquist plots for this transformation are depicted in Figure 2-15, Example of Sector Transform.. The condition $P_1(s)$ inside [0, ∞] implies that $P_1(s)$ is stable and $P_1(j\omega)$ is *positive real*, i.e.,

$$P_1^*(j\omega) + P_1(j\omega) \geq 0 \quad \forall\omega$$

`sectf` is a M-file in the *Robust Control Toolbox* that uses the generalization of the sector concept of [3] described by [1]. First the sector input data `Sf= secf` and `Sg=secg` is converted to two-port state-space form; non-dynamical sectors are handled with empty *a*, *b1*, *b2*, *c1*, *c2* matrices. Next the equation

$$S_g(s)\begin{bmatrix} u_{g_1} \\ y_{g_1} \end{bmatrix} = S_f(s)\begin{bmatrix} u_{f_1} \\ y_{f_1} \end{bmatrix}$$

is solved for the two-port transfer function $T(s)$ from $u_{g_1} y_{f_1}$ to $u_{f_1} y_{g_1}$. Finally, the function `lftf` is used to compute $G(s)$ via one of the following:

```
tssg=lftf(tsst,tssg)
ssg=lftf(tsst,ssg).
```

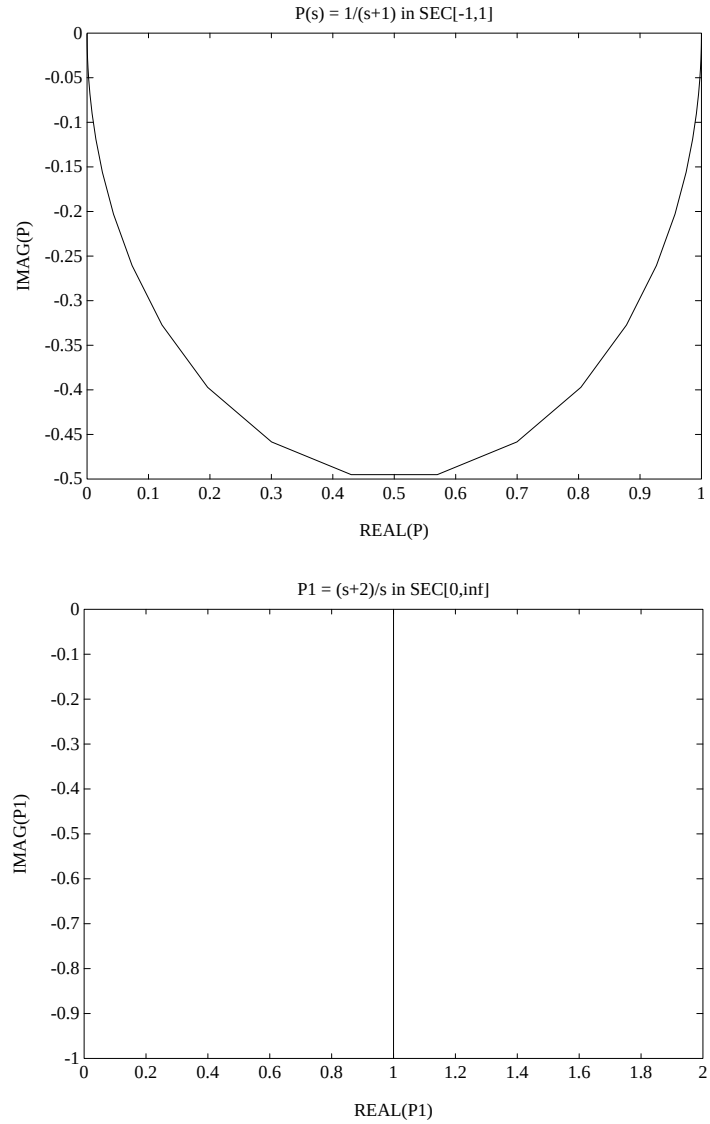


Figure 2-15: Example of Sector Transform.

Limitations

A well-posed conic sector must have $\det(B - A) \neq 0$ or $\det\left(\begin{bmatrix} s_{11} & s_{12}; \\ s_{21} & s_{22} \end{bmatrix}\right) \neq 0$.

Also, you must have $dim(u_{f_1}) = dim(y_{f_1})$ since sectors are only defined for square systems.

See Also

lftf, hinf, system

References

[1] M. G. Safonov, *Stability and Robustness of Multivariable Feedback Systems*. Cambridge, MA: MIT Press, 1980.

[2] M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, "Synthesis of Positive Real Multivariable Feedback Systems," *Int. J. Control*, vol. 45, no. 3, pp. 817-842, 1987.

[3] G. Zames, "On the Input-Output Stability of Time-Varying Nonlinear Feedback Systems ≥— Part I: Conditions Using Concepts of Loop Gain, Conicity, and Positivity," *IEEE Trans. on Automat. Contr.*, AC-11, pp. 228-238, 1966.

Purpose

Left and right spectral factorization.

Syntax

```
[am,bm,cm,dm] = sfl(a,b,c,d)
[am,bm,cm,dm] = sfr(a,b,c,d)
[ssm] = sfl(ss)
[ssm] = sfr(ss)
```

Description

Given a stabilizable realization of a transfer function $G(s) := (A, B, C, D)$ with $\|G\|_\infty < 1$, `sfl` computes a left spectral factor $M(s)$ such that

$$M^*(s)M(s) = I - G^*(s)G(s)$$

where $M(s) := (A_M, B_M, C_M, D_M)$ is *outer* (i.e., stable and minimum-phase).

`Sfr` computes a right spectral factor $M(s)$ of $G(s)$ such that

$$M(s)M^*(s) = I - G(s)G^*(s)$$

Algorithm

Given a transfer function $G(s) := (A, B, C, D)$, the LQR optimal control $u = -Fx = -R^{-1}(XB + N)^T x$ stabilizes the system and minimize the quadratic cost function

$$J = \frac{1}{2}\left(x^T(t_f)P_1x(t_f) + \int_{t_0}^{t_f} \begin{bmatrix} x^T & u^T \end{bmatrix} \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt \right)$$

as $t_f \to \infty$, where $X = X^T$ satisfies the algebraic Riccati equation

$$A^TX + XA - (XB + N)R^{-1}(XB + N)^T + Q = 0$$

Moreover, the optimal return difference $I + L(s) = I + F(Is - A)^{-1}B$ satisfies the *optimal LQ return difference equality*:

$$(I + L)^*R(I + L) = \begin{bmatrix} \Phi^* & I \end{bmatrix} \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} \Phi \\ I \end{bmatrix}$$

where $\Phi(s) = (Is - A)^{-1}B$, and $\Phi^*(s) = \Phi^T(-s)$. Taking

$$\begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} C^T \\ D^T \end{bmatrix} \begin{bmatrix} C & D \end{bmatrix},$$

the return difference equality reduces to

$$(I + L)^* R (I + L) = I - G^* G$$

so that a minimum phase, but not necessarily stable, spectral factor is

$$\tilde{M}(s) := R^{\frac{1}{2}}(I + L) = R^{\frac{1}{2}}(I + F(Is - A)^{-1}B)$$

where X and F can simply be obtained by the command:

```
[F,X] = lqr(A,B,Q,R,N) =
lqr(A,B,-C'*C,(I-D'*D),-C'*D).
```

Finally, to get the stable spectral factor, we take $M(s)$ to be the inverse of the outer factor of $\tilde{M}^{-1}(s)$. The routine `iofr` is used to compute the outer factor.

Limitations

The spectral factorization algorithm employed in `sfl` and `sfr` requires the system $G(s)$ to have $\|G\|_\infty < 1$ and to have no $j\omega$-axis poles. If the condition $\|G\|_\infty < 1$ fails to hold, the Riccati subroutine (`aresolv`) will normally produce the message

```
WARNING: THERE ARE jω-AXIS POLES...
RESULTS MAY BE INCORRECT !!
```

This happens because the Hamiltonian matrix associated with the LQR optimal control problem has jω-axis eigenvalues if and only if $\|G\|_\infty < 1$. An interesting implication is that you could use `sfl` or `sfr` to check whether $\|G\|_\infty < 1$ without the need to actually compute the singular value Bode plot of G(jω).

See Also

`iofc`, `iofr`

Purpose

Compute the structured singular value (multivariable stability margin) Bode plot.

Syntax

```
[mu,logd] = ssv(a,b,c,d,w)
[mu,logd] = ssv(a,b,c,d,w,k)
[mu,logd] = ssv(a,b,c,d,w,k,opt)
[mu,logd] = ssv(ss, )
```

Description

ssv produces the row vector `mu` containing an upper bound on the structured singular value (SSV) of $p \times q$ a transfer function matrix

$$G(j\omega) = C(j\omega I - A)^{-1}B + DB$$

evaluated at frequency points in vector ω.

Several methods are included via the following input options:

`opt` — options for method used in computing SSV:

'`osborne`' Osborne method

'`psv`' optimal diagonal scaled Perron eigenvalue (default)

'`perron`' Perron eigenvalue (if only `mu` output specified).

'`muopt`' Multiplier approach for pure real, pure complex and mixed real/complex uncertainties. If there exists real uncertainty, '`muopt`' will be used (default for systems with mixed uncertainty).

`k` — uncertainty block sizes (default: `k=ones(q,2)`); `k` can be an $n \times 1$ or $n \times 2$ matrix whose rows are the uncertainty block sizes for which the SSV is to be evaluated. If only the first column of `k` is given, then the uncertainty blocks are taken to be square, as if `k(:,1) = k(:,2)`. If a 1×1 uncertainty block is real (say, the i^{th} block), then you should assign

```
k(i,:) = [-1, -1];
```

and set the input argument `opt` to '`muopt`' to invoke the multiplier nonlinear programming algorithm to compute a less conservative SSV upper bound.

The output variables are:

`mu` — the Bode plot of the SSV

`logd` — the log Bode plot of the optimal diagonal scaling $D(j\omega)$. When the uncertainties are all complex, then $D(j\omega)$ is purely real; this is always the case for the `opt = 'psv'` or `opt = 'muopt'` options. If `opt = 'muopt'` and the uncertainty is of the mixed real/complex type, `logd` in general will be complex and will contain the log Bode plot of the squareroots of the optimal multiplier scalings.

Algorithm

`ssv` performs its computation using either `perron`, `psv`, `osborne`, or `muopt` depending of the value of option. All of them can handle the *irreducible* special case where the standard algorithms fail. See the documentation for these functions for further details.

Examples

This example compares all the methods available in the *Robust Control Toolbox* for a less conservative multivariable stability margin prediction. The transfer function T_{y_u} seen by the real uncertainties is the one established in ACC Benchmark problem [4], which can be extracted from the demo `accdemo.m`. As shown in Figure 2-16, Comparison of Robust Analysis Methods., the multiplier solution provides the least conservative result, as compared to Perron and Osborne.

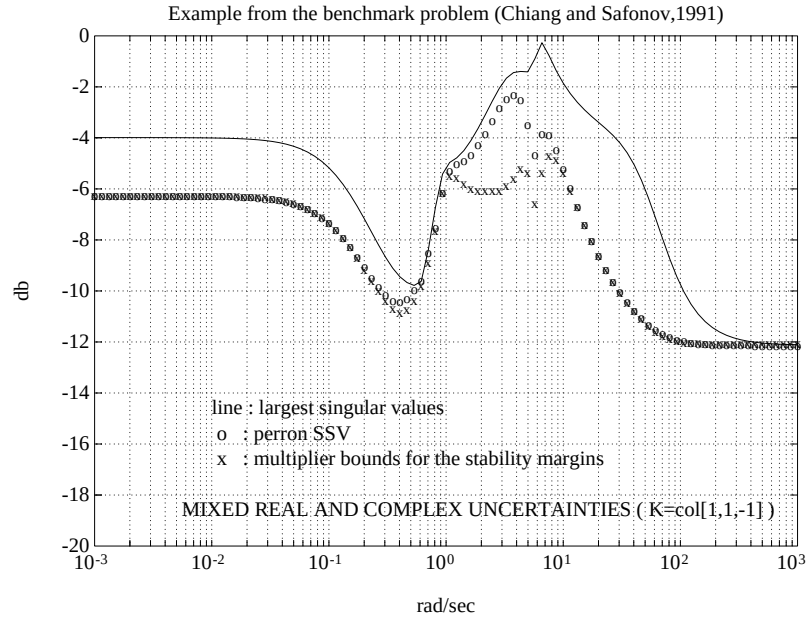


Figure 2-16: Comparison of Robust Analysis Methods.

See Also

muopt, perron, psv, osborne

References

[1] E. E. Osborne, “On Preconditioning of Matrices,” *J. of Assoc. of Computing Machinery*, vol. 7, pp. 338-345, March 1960.

[2] M. G. Safonov, “Stability Margins for Diagonally Perturbed Multivariable Feedback Systems,” *IEE Proc.*, vol. 129, Part D, pp. 251-256, 1982.

[3] M. G. Safonov, and Peng-Hin Lee, “A Multiplier Method for Computing Real Multivariable Stability Margins,” unpublished report, Electrical Eng. Dept., University of Southern Calif., Los Angeles, CA 90089-2563, July 1992; submitted to *1993 IFAC World Congress*, Sydney, Australia.

[4] B. Wie and D. S. Bernstein, “A Benchmark Problem for Robust Controller Design,” *Proc. American Control Conf.*, San Diego, CA May 23-25, 1990; also Boston, MA, June 26-28, 1991.

Purpose

Stable and antistable projection.

Slow and fast modes decomposition.

Syntax

```
[a1,b1,c1,d1,a2,b2,c2,d2,m] = stabproj(a,b,c,d)
[a1,b1,c1,d1,a2,b2,c2,d2] = slowfast(a,b,c,d,cut)
[ss1,ss2,m] = stabproj(ss)
[ss1,ss2] = slowfast(ss,cut)
```

Description

`stabproj` computes the stable and antistable projections of a minimal realization $G(s)$ such that

$$G(s) = [G(s)]_- + [G(s)]_+$$

where $[G(s)]_- := (\hat{A}_{11},\ \hat{B}_1,\ \hat{C}_1,\ \hat{D}_1)$ denotes the stable part of $G(s)$, and $[G(s)]_+ := (\hat{A}_{22},\ \hat{B}_2,\ \hat{C}_2,\ \hat{D}_2)$ denotes the antistable part. The variable `m` returns the number of stable eigenvalues of A.

`Slowfast` computes the slow and fast modes decompositions of a system $G(s)$ such that

$$G(s) = [G(s)]_s + [G(s)]_f$$

where $[G(s)]_s := (\hat{A}_{11},\ \hat{B}_1,\ \hat{C}_1,\ \hat{D}_1)$ denotes the slow part of $G(s)$, and $[G(s)]_f := (\hat{A}_{22},\ \hat{B}_2,\ \hat{C}_2,\ \hat{D}_2)$ denotes the fast part. The variable cut denotes the index where the modes will be split.

Algorithm

Both `stabproj` and `slowfast` employ the algorithm in [1] as follows:

Find an unitary matrix V via the ordered Schur decomposition routines `blksch` or `rschur` such that

$$A = V^T A V = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ 0 & \hat{A}_{22} \end{bmatrix}$$

Based on the style of ordered Schur form, you can get a stable $\hat{A}_{11}$ and an antistable $\hat{A}_{22}$ for the case of `stabproj`; $|\lambda_i(\hat{A}_{11})| < |\lambda_i(\hat{A}_{22})|$ for the case of `slowfast`.

Finally solving the matrix equation for X

$$\hat{A}_{11}X - X\hat{A}_{22} + \hat{A}_{12} = 0$$

you get the state-space projections

$$[G(s)]_- \quad or \quad [G(s)]_s := \left[\begin{array}{c|c} \hat{A}_{11} & \hat{B}_1 \\ \hline C_1 & 0 \end{array}\right]$$

$$[G(s)]_+ \quad or \quad [G(s)]_f := \left[\begin{array}{c|c} \hat{A}_{22} & \hat{B}_2 \\ \hline C_2 & D \end{array}\right]$$

where

$$\begin{bmatrix} \hat{B}_1 \\ \hat{B}_2 \end{bmatrix} := \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} VB$$

and

$$\left[\hat{C}_1 \middle| \hat{C}_2\right] := CV^T \begin{bmatrix} I & X \\ 0 & I \end{bmatrix}$$

See Also

`blkrsch`, `cschur`, `rschur`, `schur`

References

[1] M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, "Synthesis of Positive Real Multivariable Feedback Systems", *Int. J. Control*, vol. 45, no. 3, pp. 817-842, 1987.

tfm2ss

Purpose

Convert a transfer function matrix (MIMO) into state-space form.

Syntax

```
[a,b,c,d] = tfm2ss(num,den,r,c)
[ss] = tfm2ss(tf,r,c)
```

Description

`Tfm2ss` converts a transfer function matrix $G(s)$ of dimension r by c into the *block-controller* form [1], where

$$G(s) = \frac{1}{d(s)} N(s)_{r \times c}$$

and $d(s)$ is the least common multiple of the denominators of the entries of $G(s)$

$$d(s) = \alpha_0 s^n + \alpha_1 s^{n-1} + \alpha_2 s^{n-2} + \ldots + \alpha_n$$

and the entry in the i^{th} row, j^{th} column of the numerator matrix

$$[N(s)]_{ij} = \beta_{ij_0} s^n + \beta_{ij_1} s^{n-1} + \ldots + \beta_{ij_n}$$

The input variables `num` and `den` are of the following forms

$$num := \begin{bmatrix} N_{11} \\ \vdots \\ N_{r1} \\ \vdots \\ N_{1c} \\ \vdots \\ N_{rc} \end{bmatrix}$$

$$den := [\alpha_0 \alpha_1 \alpha_2 \ldots \alpha_n]$$

where $N_{ij} = [\beta_{ij_0}, \beta_{ij_1}, \ldots, \beta_{ij_n}]$.

A dual realization *lock-observer* form can simply be obtained by applying `tfm2ss` to $G^T(s)$, then taking the transpose of the resulting state-space system.

Note that the resulting system has n by c states and is *not necessarily minimal*. Model reduction routines such as `minreal`, `schmr`, `ohklmr`, or `bstschmr` can be helpful in removing any uncontrollable and unobservable modes.

See Also

`minreal`, `obalreal`, `ohklmr`, `bstschmr`, `schmr`, `ss2tf`, `tf2ss`

References

[1] T. Kailath, *Linear Systems*, Prentice-Hall, 1980, pp. 346-349.

Purpose

Pack all information and data from several matrices, vectors and/or strings into a single 'tree' variable. `tree` implements the *Robust Control Toolbox* hierarchical tree data structure.

Syntax

```
T = tree(nm,b1,b2, ,bn)
[i] = istree(T)
[i,b] = istree(T,path)
```

Description

`tree` creates a data structure `T`, called a tree containing several variables and their names. This single variable contains all the data and dimension information from its branches `b1`, `b2`, `b3`,..., `bn` along with a numerical index and a string name for each branch.

The input argument `nm` is a string containing a list of names (separated by commas) to be assigned to the respective branches `b1`,`b2`,...,`bn`

```
nm = 'name1,name2,name3, ,nameN';
```

The names may be any valid variable names and must be separated by commas. If no names are to be assigned, set `"nm = "`;.

The input arguments `b1,b2, ,bn` (called the root branches of the tree) may be matrices, vectors, strings, or even trees themselves, thus enabling the creation of a hierarchical tree data structure within the MATLAB framework.

`istree` checks whether a variable `T` is a tree or not.

```
I = istree(T);
```

returns `I=1` (true) if "`T`" is a tree variable created by the function `tree`; otherwise it returns `I=0` (false). When the second input argument `PATH` is present, the function `istree` checks the existence of the branch specified by `PATH`. For example,

```
[I,B] = istree(T,PATH);
```

returns "`I = 1`", if both "`T`" is a tree variable and "`PATH`" is a valid path to a branch in the tree `T`. Otherwise, it returns "`I = 0`". If the optional output argument "`B`" is present, then `B` returns the value of the branch specified by `PATH`, provided `T` is a tree and `PATH` is a valid path in `T`.

Related functions include the following:

`branch`: returns branches of a tree.
`branch(T,0)`: returns `nm`, the list of branch names.
`T(1)`: returns the number `N` of the root branches.

Examples

A hierarchical tree structure shown inFigure 2-17, Example of Tree Structure. can be built as follows

```
tree1 = tree('a,b,c',a,b,c);
tree3 = tree('w,x',w,x);
tree2 = tree('tree3,y,z',tree3,y,z);
bigtree = tree('tree1,tree2',tree1,tree2);
```

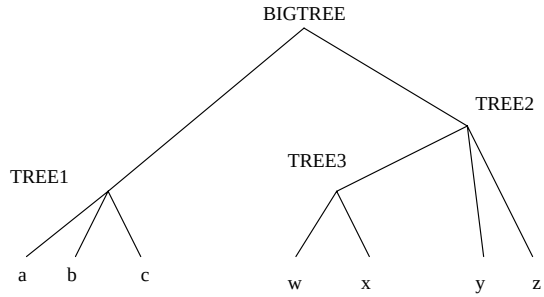


Figure 2-17: Example of Tree Structure.

To extract a variable from a tree, simply use the branch function:

```
[tree1,tree2] = branch(bigtree);
% getting variable w:
w = branch(bigtree,'tree2/tree3/w');
% getting several variables:
[w,b] = branch(bigtree,'tree2/tree3/w,tree1/b');
```

Paths in a tree are also determined by the numerical indices of the branches which lead to the branch of interest. For example, the last line above is equivalent to

```
[w,b] = branch(bigtree,'2/1/1,1/2');
```

See branch for further details.

See Also

branch, mksys, graft, istree, issystem, vrsys

youla

Purpose

Parametrization of all realizable stable closed-loop systems for use in infinity-norm controller synthesis.

Syntax

```
youla
  Inputs: A, B1, B2, C1, C2, D11, D12, D21, D22
  Outputs: at11, bt11, ct11, dt11
     at12, bt12, ct12, dt12, at1p, bt1p, ct1p, dt1p
     at21, bt21, ct21, dt21, at2p, bt2p, ct2p, dt2p
     kx, x, ky, y, f, h
```

Description

`youla` is a script M-file used as a subroutine by the script M-file `linf`. Given an "augmented plant" $P(s)$ having state-space matrices

$$\left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

`youla` computes an LQG controller $K(s)$ such that the closed-loop system $T(s)$ shown in Figure 2-18, Youla Parametrization. has the form

$$T(s) := \begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & T_{22}(s) \end{bmatrix} \equiv \begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & 0 \end{bmatrix}$$

with T_{12} and T_{21} *inner*, i.e.

$$T_{12}^T(-s)T_{12}(s) = I, \quad T_{21}(s)T_{21}^T(-s) = I$$

`Youla` also computes complementary inner-factors $T_{12}^{\perp}$ and $T_{21}^{\perp}$ such that $T_{12}(s)$, $T_{21}^{\perp}(s)$ and $\begin{bmatrix} T_{21}(s) \\ T_{21}^{\perp}(s) \end{bmatrix}$ are *square* and inner. A realization for

T_{11}, T_{12}, $T_{12}^{\perp}$, T_{21}, $T_{21}^{\perp}$ is returned as

$$T_{11}(s) := ss11, \; T_{12}(s) := ss12, \; T_{12}^{\perp}(s) := ss12p,$$

$$T_{21}(s) := ss21, \; T_{21}^{\perp}(s) := ss21p$$

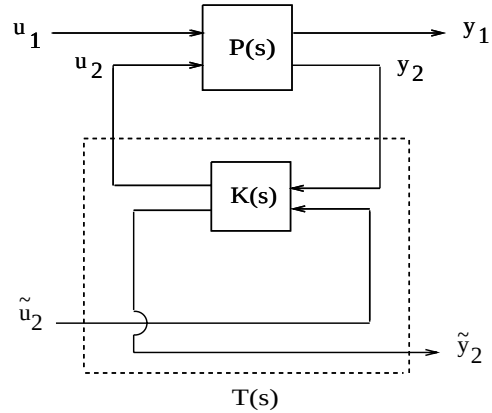


Figure 2-18: Youla Parametrization.

The LQG controller has realization

$$\dot{\hat{x}} = A\hat{x} + B_2 u_2 - H\tilde{y}_2$$
$$u_2 = F\hat{x} + \tilde{u}_2$$
$$\tilde{y}_2 = y_2 - C_2\hat{x} + D_{22}u_2$$

The state-feedback Riccati solution and the Kalman-Bucy filter Riccati equation are returned as `x` and `y` respectively. Also returned are the associated gain matrices `f` and `h`.

As shown by [3], the closed-loop transfer function of the system in Figure 2-19, Q-Parametrization. is the Youla parameterization of the set of realizable stable closed-loop transfer functions, viz.,

$$T_{y_1 u_1} = T_{11}(s) + T_{12}(s)Q(s)T_{21}(s)$$

where $Q(s)$ is any stable transfer function matrix.

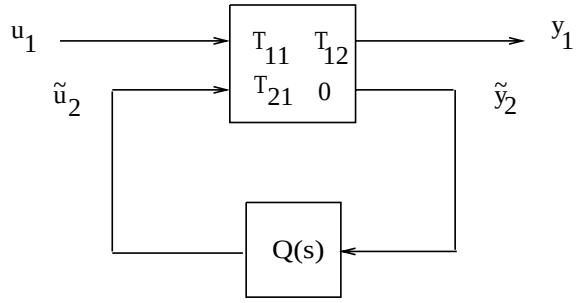


Figure 2-19: Q-Parametrization.

Algorithm

We employ the formulae of [2], as reported in the paper [4]:

$$\begin{bmatrix} T_{11} & T_{12} & T_{12}^{\perp} \\ T_{21} & 0 & 0 \\ T_{21}^{\perp} & 0 & 0 \end{bmatrix} :=$$

$$\left[\begin{array}{cc|ccc} A+\tilde{B}_2F & -B_2F & B_1 & \tilde{B}_2 & -X^{-1}C_1^TC_{12}^{\perp} \\ 0 & A+H\tilde{C}_2 & B_1+H\tilde{D}_{21} & 0 & 0 \\ \hline C_1+\tilde{D}_{12}F & -\tilde{D}_{12}F & D_{11} & \tilde{D}_{12} & \tilde{D}_{12}^{\perp} \\ 0 & -\tilde{C}_2 & -\tilde{D}_{21} & 0 & 0 \\ 0 & D_{21}^{\perp}B_1^TY^{-1} & -D_{21}^{\perp} & 0 & 0 \end{array}\right]$$

where X^{-1} and Y^{-1} are pseudo inverses, and

$$\tilde{D}_{12} = D_{12}(D_{12}^TD_{12})^{-½}, \quad \tilde{D}_{21} = (D_{21}D_{21}^T)^{-½}D_{21}$$
$$\tilde{B}_2 = B_2(D_{12}^TD_{12})^{-½}, \quad \tilde{C}_2 = (D_{21}D_{21}^T)^{-½}C_2$$

and $D_{12}^{\perp}$ and $D_{21}^{\perp}$ are computed using `ortc` and `ortr` such that

$$\begin{bmatrix} D_{12} & D_{12}^{\perp} \end{bmatrix} \quad and \quad \begin{bmatrix} D_{21} \\ D_{21}^{\perp} \end{bmatrix}$$

are both unitary. The variables `f`, `x`, `h`, `y` are computed as the solution of LQ optimal control problems via the MATLAB commands:

```
[kx,x] = lqrc(A,B2,C1'*C1,D12'*D12,C1'*D12);
[ky,y] = lqrc(A',C2',B1*B1',D21*D21',B1*D21');
f = -kx;
h = -ky';
```

See Also

h2lqg, hinf, hinfdemo, linf, linfdemo

References

[1] C. A. Desoer, R. W. Liu, J. Murray and R. Saeks, “Feedback System Design: The Fractional Representation Approach to Analysis and Synthesis,” *IEEE Trans. on Automat. Control*, June, 1980.

[2] J. Doyle, *Advances in Multivariable Control*. Lecture Notes at ONR/Honeywell Workshop. Minneapolis, MN, Oct. 8-10, 1984.

[3] C. N. Nett, C. A. Jacobson, and M. J. Balas, “A Connection Between State-Space and Doubly Coprime Fractional Representations,” *IEEE Trans. on Automat. Control*, AC-29, Sep. 1984.

[4] M. G. Safonov, E. A. Jonckheere, M. Verma and D. J. N. Limebeer, “Synthesis of Positive Real Multivariable Feedback Systems”, *Int. J. Contro*l, vol. 45, no. 3, pp. 817-842, 1987.

Index

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Y