Proc. American Control Conf., Boston, MA, June 30 - July 2, 2004, to appear.

Next generation of tools for robust control

G.J. Balas, A.K. Packard, M.G. Safonov and R.Y. Chiang

***Abstract*— This paper describes the new MATLAB Robust Control Toolbox which combines the original MATLAB Robust Control Toolbox, LMI and the μ-Analysis and Synthesis Toolboxes. The toolbox introduces an uncertain data structure and new robustness analysis and synthesis algorithms. The uncertain data structure is based on linear fractional transformations and forms the core infrastructure to model and analyze uncertain systems and matrices. The uncertain data structure is fully compatible with the Control System Toolbox data structure. In addition to the structured singular value and LMI solvers, the robust analysis commands have been expanded to include multivariable loop analysis and worst-case gain, sensitivity and gain/phase margin routines. A new H_∞ loop-shaping algorithm and automated μ-synthesis have been added to the suite of synthesis algorithms. In addition to the enhanced functionality, the new MATLAB Robust Control Toolbox provides enhanced usability of advanced control techniques for all users.**

I. Introduction

The new MATLAB Robust Control Toolbox combines the original MATLAB Robust Control [1], LMI [2] and the μ-Analysis and Synthesis [3] Toolboxes into a single, comprehensible tool set for advanced multivariable control. The core infrastructure of the new toolbox is an uncertain data structure based on linear fractional transformations that is compatible with the Control System Toolbox data structure [4]. The combined robust analysis commands have been expanded to include multivariable loop analysis and worst-case gain, sensitivity and gain/phase margin routines. A H_∞ loop-shaping and automated μ-synthesis command have been added to the suite of synthesis algorithms. This paper highlights the enhancements included in the new Robust Control Toolbox.

II. Data Structure

The uncertain data structure is based on linear fractional transformations. The data structure is implement via the following classes

All of the atoms have a `get` and `set` commands, that mimic the MATLAB Handle Graphics, and Control System Toolbox behavior. This includes tab-completion and case-insensitive, partial property name matching.

G.J. Balas is with Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis MN 55455, balas@aem.umn.edu

A.K. Packkard is with Mechanical Engineering, University of California, Berkeley CA 94720, pack@me.berkeley.edu

M.G. Safonov and R.Y. Chiang are with Electrical Engineering, University of Southern California, Los Angeles, CA 90089, msafonov@usc.edu

Name	Meaning
`ureal`	uncertain real parameter
`ultidyn`	uncertain linear, time-invariant dynamics
`ucomplex`	uncertain complex parameter
`ucomplexm`	uncertain complex matrix
`uarbop`	uncertain arbitrary operator
`atom`	parent class
`umat`	uncertain matrix
`uss`	uncertain state-space
`ufrd`	uncertain frequency response
`ic`	Interconnection object
`icsignal`	signal object for interconnection

TABLE I
Uncertain Data Structure Classes

For example, consider real parameter atoms `ureal`. The user-accessible properties of a `ureal` are

Get/ Set	Property	Meaning	Class
√	Name	name	`char`
√	NominalValue	nominal value	`double`
√	Mode	`'{PlusMinus}' | 'Range' | 'Percentage'`, invariant when nominal changed	`char`
√	PlusMinus	additive variation	scalar or 1×2 `double`
√	Range	numerical range	1×2 `double`
√	Percentage	additive variation (% of absolute value of nominal)	scalar or 1×2 `double`
√	AutoSimplify	`'none' | {'basic'} | 'full'`	`char`

The properties `Range`, `Percentage` and `PlusMinus` are all automatically synchronized. An example of an uncertain real parameter is:

```
>> b = ureal('b',2,'percentage',10)
ureal, b, NominalValue 2, percentage  +/- [-10  10]%
>> get(b)
            Name: 'b'
    NominalValue: 2
            Mode: 'Percentage'
       PlusMinus: [-0.2000 0.2000]
           Range: [1.8000 2.2000]
      Percentage: [-10.0000 10.0000]
    AutoSimplify: 'basic'
```

Uncertain matrices (class `umat`) are built from doubles, and uncertain atoms.

```
>> a = ureal('a',3);
>> b = ureal('b',10,'pe',20);
>> m = [-a 1/b;b a+1/b;1 3]
```

```
UMAT: 3 Rows, 2 Columns
2 copies, ureal, a, nominal 3, additive  +/- [-1  1]
3 copies, ureal, b, nominal 10, percentage  +/- [-20  20]%
```

The user-accessible properties of a `umat` are its `NominalValue` and `Uncertainty`. The `NominalValue` of an uncertain matrix corresponds to the constant matrix resulting from all uncertain paramters set to their nominal values. The `Uncertainty` property is a gateway that contains the uncertain parameters from of an `umat` object.

Uncertain systems (class `uss`) are built from of certain and/or uncertain state-space matrices. For example

```
>> p1 = ureal('p1',10,'pe',50);
>> p2 = ureal('p2',3,'plusm',[-.5 1.2]);
>> A = [-p1 p2;0 -p1];
>> B = [-p2;p2];
>> C = [1 0;1 1];
>> D = [0;0];
>> sys = uss(A,B,C,D)

USS: 2 States, 2 Outputs, 1 Input Continuous System
2 copies, ureal, p1, nominal 10, percentage  +/- [-50  50]%
2 copies, ureal, p2, nominal 3, additive  +/- [-0.5 1.2]
```

Uncertain frequency response objects (class `ufrd`) are typically generated in two manners: combining frequency reponse data (`frd`) objects with uncertain matrices (`umat`), or taking the frequency response on an uncertain state-space (`uss`) system. An example of a `ufrd` follows.

```
>> sys

USS: 2 States, 2 Outputs, 1 Input Continuous System
2 copies, ureal, p1, nominal 10, percentage  +/- [-50  50]%
2 copies, ureal, p2, nominal 3, additive  +/- [-0.5 1.2]

>> sysg = frd(sys,logspace(-2,2,100));

UFRD: 2 Outputs, 1 Input Cont System, 100 Frequency points
ureal, p1, nominal 10, percentage  +/- [-50  50]%, 2 copies
ureal, p2, nominal 3, additive  +/- [-0.5  1.2], 2 copies
```

Consider the following example to illustrate the new Robust Control Toolbox data structure. Two uncertain real parameters `gamma` and `tau` are constructed from which an uncertain transfer function `P` is formed.

```
>> gamma = ureal('gamma',4)
Uncertain Real Parameter: Name gamma, NominalValue 4,
  variability = [-1  1]
>> tau = ureal('tau',.5,'Percentage',30);
Uncertain Real Parameter: Name tau, NominalValue 0.5,
  variability = [-30  30]%
>>P = tf(gamma,[tau 1])
USS: 1 State, 1 Output, 1 Input, Continuous System
gamma: real, nominal = 4, variability = [-1  1]
  tau: real, nominal = 0.5, variability = [-30  30]%
```

An integral controller, `C`, is synthesized for the uncertain plant `P` based on nominal plant parameters. The closed-loop system `CLP` is formed from `C` and `P`.

```
>>KI = 1/(2*tau.Nominal*gamma.Nominal);
>>C = tf(KI,[1 0]);
>>CLP = feedback(P*C,1);
```

The uncertain parameters `gamma` and `tau` are randomly sampled 20 times and a unit step response of the corresponding plant models and closed-loop systems are plotted in Figure 1.

```
>>subplot(2,1,1); step(usample(P,20))
>>subplot(2,1,2); step(usample(CLP,20))
```

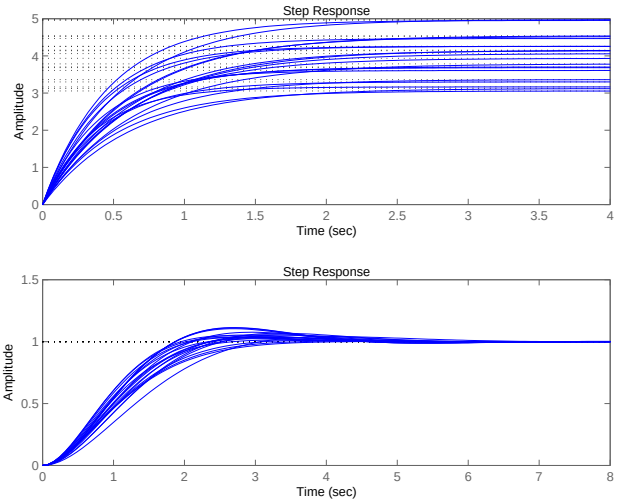

Fig. 1. Plots of the uncertain plant and closed-loop system for 20 random values of `gamma` and `tau`.

The uncertain data structure allows the construction of complicated interconnections of uncertain components. The uncertain interconnection is represented as an uncertain state-space (`uss`) object in the user's workspace. Internally, the uncertain system is represented as a linear fractional transformation.

III. Robustness Analysis

The suite of robust analysis tools have been expanded to include multivariable analysis for linear systems (`loopsens, loopmargin`) and worst-case gain, gain/phase margin and sensitivity (`wcgain, wcmargin, wcsens`) analysis.

The worst-case analysis commands are based on algorithms described in Reference [7]. The command `wcgain` calculates upper and lower bounds on the worst-case gain of an uncertain system. In addition to the upper and lower bounds, the algorithms provide corresponding information about the frequency and uncertainty values associated with the worst-case bounds. A text string description of the worst-case results is provided to aid novice users in interpreting the numerical data.

Similarly `wcmargin` calculates the combined worst-case input and output loop-at-a-time gain/phase margins of a plant and controller feedback loop. The worst-case margin algorithm calculates the minimum "distance" to the -1 point of the perturbed open-loop

gain in terms of the Nyquist diagram. This value is translated into a guaranteed bound on the worst-case, single-loop gain and phase margin of the channel. As in `loopmargin`, the guaranteed bounds on the combined gain/phase margins are calculated based on a balanced sensitivity function [6]. A text string description of the worst-case margins results is also provided. `wcsens` calculates the worst-case sensitivity and complementary sensitivity functions for the plant and controller feedback loop. The frequency range of importance for the worst-case sensitivity analysis can be influenced with the addition of a scaling function. The default is `'Absolute'` which involves the worst-case unweighted sensitivity function. Alternatively, a `'Relative'` scaling or any scaling realized as a LTI object could be used to scaled the sensitivity function.

All the worst-case functions have an option to provide sensitivity analysis information on the worst-case bounds to the individual uncertainties in the problem. The full suite of robustness analysis commands available are shown in Table II.

Command	Meaning
`gapmetric`	Computes the gap and Vinnicombe gap metric.
`loopmargin`	Comprehensive analysis of feedback loop.
`loopsens`	Sensitivity functions of feedback loop.
`mussv`	Structured singular value bounds (μ).
`ncfmargin`	Normalized coprime stability margin of feedback loop.
`robustperf`	Robust performance of uncertain systems.
`robuststab`	Stability margins of uncertain systems.
`wcgain`	Calculates worst-case gain of an uncertain system.
`wcmargin`	Worst-case gain/phase margins for feedback loop.
`wcsens`	Worst-case sensitivity functions of feedback loop.

TABLE II
ROBUST ANALYSIS COMMANDS

IV. Controller Synthesis

Two new synthesis commands have been added to the Robust Control Toolbox: `loopsyn` and `dksyn`. The command `loopsyn` is a H_∞ loopshaping control design routine and `dksyn` is an automated μ-synthesis command.

A. Loop Shaping

A new loop-shaping algorithm has been developed, denoted `loopsyn`, that optimally synthesizes a stabilizing compensator by fitting the compensated loop transfer function's sigma plot to any desired sigma plot as precisely as possible. The algorithm combines a novel all-pass squaring-down compensator technique, together with optimal Balanced, Stochastic Truncation (BST) minimal realization techniques and normalized-coprime optimal H_∞ synthesis.

For modern H_∞ robust control to be as easy to use as classical design methods, it needs to be simple and intuitive. Indeed, it could be even easier to use than classical methods, since the H_∞ optimization methods can automate the chore of compensator design, once the designer has given the specification. Traditional H_∞ design methods like mixed-sensitivity fall short in this regard because the several weights $W_1(s)$, $W_2(s)$ and $W_3(s)$ in S, KS and $T = I - S$ *over-specify* the design: There are three weights to choose but only one loop-transfer function to shape. The designer becomes bogged down with extra care required to ensure that fundamental identities like $S + T = I$ and $T = GKS$ are not violated by the implied loop-shape constraints, viz. $\bar{\sigma}(S) < 1/|W_1|$, $\bar{\sigma}(KS) < 1/|W_2|$, $\bar{\sigma}(T) < 1/|W_3|$. Further, direct shaping S, KS and T may seem unnecessarily awkward to a designer familiar with classical specifications that are directly related to the loop shape $L := GK$, e.g., crossover-frequency ω_c, etc.

The Glover-McFarlane loop-shaping technique has proved useful in this respect, since it allows one to directly focus on loop-shape and crossover frequency. However, the designer is still burdened with manually designing pre-filter (or 'weight') W for the plant G, so that the compensated plant GW has the desired loop-shape, which can be a non-trivial task if attempted for a MIMO plant.

The H_∞ loopshaping routine, `loopsyn`, makes use of a novel all-pass squaring-down algorithm of Safonov and Le to compute an optimal stable minimum-phase Glover-McFarlane pre-filter $W(s)$ that causes the sigma plots for all singular values to assume any specified loop-shape $|G_d(j\omega)|$ [5]. Further, the Safonov-Le pre-filter has the important property that plant RHP zeros are left invariant; i.e., no performance-limiting RHP zeros or poles are introduced. The result is that the designer is completely relieved of the task of manually computing the weight $W(s)$. Designing an optimal loop shaping controller K for a MIMO plant G with the new `loopsyn` technique is as simple as specifying the desired loopshape G_d.

```
>> K=loopsyn(G,Gd)
```

Figure 2 contains a frequency magnitude plot of the individual weights, controller and plant model.

B. μ-Synthesis Control Design

The design of a robust controller via μ-synthesis has been automated in the new Robust Control Toolbox with the command `dksyn`. `dksyn` takes as input arguments an uncertain plant (`uss`), the number of measurements and control inputs and an optional options object. The outputs of the `dksyn` command is a robust controller, closed-loop uncertain system, bounds on the achieved μ value and a structure containing information about each design iteration. The options

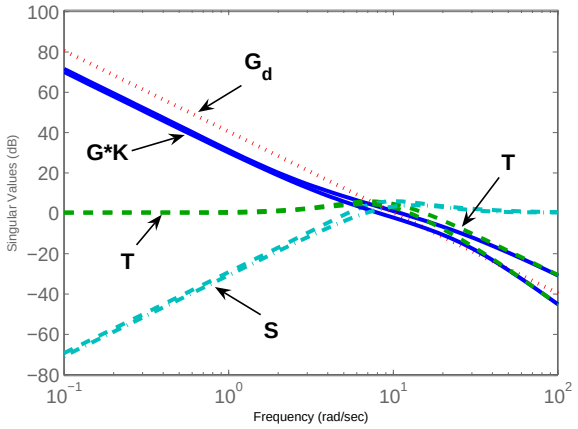

Fig. 2. Plots of GK, S, G_d, and T for a 2×3 model of the NASA HiMAT aircraft.

input argument is created via the `dkitopt` command. The option properties include the frequency vector used for analysis, an initial controller, an automatic iteration flag, display flag, starting iteration number and total number of iterations. The user can also specify the state order for fitting D-scaling data, an automated termination procedure and the termination tolerance. Default values for these parameters are used if the options object is not included in the command call.

The controller synthesis commands available in the package are shown in Table III.

Command	Meaning
`dksyn`	Synthesis of a robust controller via μ-synthesis.
`h2hinfsyn`	Mixed H_2/ H_∞ controller synthesis.
`h2syn`	H_2 controller synthesis.
`hinfsyn`	H_∞ controller synthesis.
`loopsyn`	H_∞ loop shaping controller synthesis.
`ltrsyn`	Loop-transfer recovery controller synthesis
`mixsyn`	H_∞ mixed-sensitivity controller synthesis.
`ncfsyn`	H_∞ normalized coprime factor controller synthesis.
`sdhfsyn`	Sample-data H_∞ controller synthesis.

TABLE III
CONTROLLER SYNTHESIS COMMANDS

V. Model Reduction

Model reduction techniques in the new Robust Control Toolbox have been consolidated into Hankel Singular Value based routines and a modal decomposition routine. All the Hankel SV based routines provide error bounds in additive or multiplicative form, whereas modal decomposition (modreal) can handle in particular large sized problems (greater than 200 states). The stable/antistable projection routine (`stabproj`) still plays a crucial role in toolbox to handle unstable and/or discrete systems.

Higher level routines (e.g., `reduce, balancmr, schurmr, hankelmr, bstmr,` and `ncfmr`, etc.) have been created for novice users and batch job reduced model generations. They make use of text string input augments such that the input can be provided via a series of `'String Name'` and `'Value'` in random order; the reduced order model is returned as an LTI array. The system to be reduced can be either continuous or discrete-time. Multi-model reduction is now made possible by providing a vector of desired size of the models. The new routines will generate a series of reduced order models accordingly as one wished.

The model approximation and reduction commands available in the package are shown in Table IV.

Command	Meaning
`balancmr`	Balanced truncation model reduction.
`bstmr`	Balanced stochastic truncation model reduction.
`drawmag`	Interactive mouse-based sketching and fitting tool.
`hankelmr`	Optimal Hankel norm model approximation.
`modreal`	State-space modal truncation/realization.
`ncfmr`	Balanced normalized coprime factor model reduction.
`reduce`	High level model reduction.
`schurmr`	Schur balanced truncation model reduction.

TABLE IV
MODEL APPROXIMATION AND REDUCTION COMMANDS

VI. Summary

This paper describes the new MATLAB Robust Control Toolbox which combines the original MATLAB Robust Control Toolbox, LMI and the μ-Analysis and Synthesis Toolboxes. The new toolbox provides an integrated and expanded suite of routines for the multivariable control area.

References

[1] Chiang, R.Y. and Safonox, M.G. (1996). "Robust Control Toolbox," *The Mathworks Inc.*, Natick, MA.

[2] Gahinet, P., Nemirovski, A., Laub, A.J. and Chilali, M. (1996). "LMI Control Toolbox," *The Mathworks Inc.*, Natick, MA.

[3] Balas, G.J., Doyle, J.C., Glover, K., Packard, A.K. and Smith, R. (1996). "μ-Analysis and Synthesis Toolbox," *The Mathworks Inc.*, Natick, MA.

[4] (2002). "Control System Toolbox," *The Mathworks Inc.*, Natick, MA.

[5] Le, V.X. and Safonox, M.G. (1992). "Rational matrix GCD's and the design of squaring-down compensators – s state-space theory," *IEEE Transactions on Automatic Control*, vol. 37, p. 384-92.

[6] Blight, J.D., Dailey, R.L. and Gangsass, D. (1994). "Practical control law design for aircraft using multivariable techniques," *International Journal of Control*, Vol. 59, No. 1, 93-137.

[7] A.K. Packard, G.J. Balas, R. Lui and J. Shin (2000). "Results on worst-case performance assessment," *American Control Conference*, Chicago, IL, 1862-6.