A MATLAB Toolbox for Robustness Analysis

Chung-Yao Kao
Optimization and Systems Theory
Royal Institute of Technology
Stockholm Sweden
cykao@math.kth.se

Alexandre Megretski
Laboratory for Information and Decision Systems
M.I.T
Cambridge, Massachusetts
ameg@mit.kth.se

Ulf T. Jönsson
Optimization and Systems Theory
Royal Institute of Technology
Stockholm, Sweden
ulfj@math.kth.se

Anders Rantzer
Department of Automatic Control
Lund Institute of Technology
Lund, Sweden
rantzer@control.lth.se

Abstract

Verification of control systems is often done use of extensive simulation. This gives the engineer useful qualitative information about the system performance. However, simulation cannot be used to rigorously prove stability and robustness of the system. Such properties can only be verified using mathematical analysis which for many realistic systems becomes rather involved. This paper presents the MATLAB toolbox IQCβ . It provides a user friendly interface for advanced semi-automatic systems analysis.

1 Introduction

Engineers typically base their work on mathematical models of actual processes. The mathematical models rarely describe the real processes accurately and could behave quite differently from the real process. Hence, the post-design analysis on the system's performance and robustness becomes an important stage in any engineering process. Performance analysis is often performed by computer simulation. However, for systems with infinite dimensional uncertainties (e.g. uncertain time delay), it is unlikely to explore all possible behaviors of the real systems by simulation. For engineering systems with high quality requirements, such as aircraft control systems and high speed elevators, more rigorous and efficient analysis approaches are desirable.

Robust stability and performance analysis have been active topics in the field of systems and control theory because of their practical importance. A variety of approaches have been developed since the 1940s (See [4] and references therein). Many of these analysis methods are, explicitly or implicitly, based on a concept called “Integral Quadratic Constraint“ (IQC). In this paper, we consider robustness analysis of complex systems using the IQC framework. We start with a brief introduction to the most basic ideas behind the IQC framework. Then we present a MATLAB toolbox which is built for performing analysis based upon this idea.

2 The Basic Idea

We will model real-world complex dynamical systems as the feedback interconnected block diagram shown in Figure 1. The mathematical representation of this system can be expressed as follows.

$$
\begin{aligned}
z &= \sum_{l=0}^{n} G_{0l} w_l + e_0 \\
v_k &= \sum_{l=0}^{n} G_{kl} w_l + e_k \\
w_k &= \Delta_k(v_k)
\end{aligned}
\tag{1}
$$

Each block in the diagram represents a subsystem. The G_{kl} subsystems are stable linear time-invariant (LTI) transfer functions while the Δ_k subsystems represents nonlinearities, time-varying coefficients, parametric uncertainties, and unmodelled dynamics. The exact mathematical description of (1) is in general very complicated due to the different nature of the various Δ_k blocks. To make possible an accurate and rigorous analysis of stability and performance of such a complex system it is necessary simplify the model. The basic idea behind the IQC framework is to replace each Δ_k block and every external signal e_k with an IQC characterization. In this way we embed the real system model in (1) in an IQC relaxation which contains all possible solutions of the real system. Analysis of the relaxed model is much easier and can be done using convex optimization. This approximation is often the best available for analysis and it is in some cases also exact.

To discuss the IQC framework in some more detail we introduce the space of square integrable functions $\mathbf{L}_2[0,\infty)$ and the inner product on this space

$$\langle v, w\rangle = \int_0^\infty v(t)'w(t)dt = \int_{-\infty}^{\infty} \hat{v}(j\omega)^*\hat{w}(j\omega)d\omega$$

where $\hat{v}$ and $\hat{w}$ are the Fourier transforms of v and w, respectively. There are certain mathematical assumptions on the Δ_i subsystems and on the interconnection in (1) that must be valid. The essential assumption is that all Δ_i represents causal and bounded operators

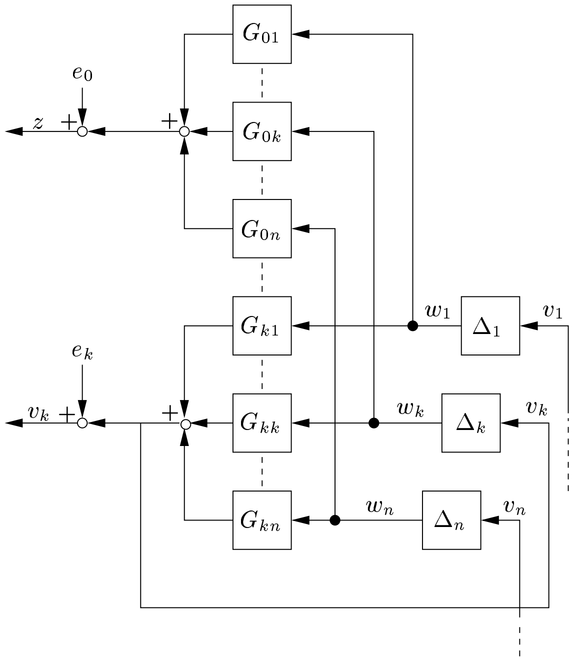

Figure 1: System under consideration

on $\mathbf{L}_2[0,\infty)$. For the exact meaning of this and other technical details below we refer to e.g. [6].

In the IQC framework, integral quadratic constraints are used to characterize external disturbances e_k and operators Δ_k. An operator Δ is said to satisfy the IQC defined by Π_λ (notation: $\Delta \in \mathrm{IQC}(\Pi_\lambda)$), if for all input-output pairs $w = \Delta(v)$, the integral inequality

$$\left\langle \begin{bmatrix} v \\ w \end{bmatrix}, \Pi_\lambda \begin{bmatrix} v \\ w \end{bmatrix} \right\rangle \geq 0 \tag{2}$$

is valid. The operator Π_λ is often referred to as the multiplier of the IQC. Here, we consider multipliers which are linearly parameterized by λ, where λ is a finite dimensional vector belonging to a *convex* cone Λ. The subscript λ is used to reflect this linear parameterization.

Similarly, for a given external disturbance e belonging to some set $\mathcal{E}$, we say that e satisfies the integral quadratic constraint defined by a bounded, self-adjoint operator Ψ_λ on $\mathbf{L}_2[0,\infty)$ (and similar notation $\mathcal{E} \in \mathrm{IQC}(\Psi_\lambda)$ applies) if

$$\langle e, \Psi_\lambda e \rangle \geq 0 \tag{3}$$

for all $e \in \mathcal{E}$. This allows us to describe the spectral characteristics of the uncertainty.

To start with the IQC analysis, one assumes that the operators Δ_k and the external disturbances e_k are known to satisfy integral quadratic constraints defined by $\Pi_{k,\lambda}$ and $\Psi_{k,\lambda}$, respectively. More comments on how to find IQCs for Δ_k and e_k, as well as the parameterization of IQCs, are given in the examples. The idea is then to replace these operators and sets by arbitrary signals that satisfy these IQCs. To illustrate this idea we let[1] $\mathcal{D}_{k,\lambda} : \mathbf{L}_2^{m_k}[0,\infty) \to \mathcal{P}(\mathbf{L}_2^{m_k}[0,\infty))$ be the set-valued function $w_k \in \mathcal{D}_{k,\lambda}(v_k)$, where

$$\mathcal{D}_{k,\lambda}(v_k) = \Bigl\{ w_k \in \mathbf{L}_2^{m_k}[0,\infty) \mid \left\langle \begin{bmatrix} v_k \\ w_k \end{bmatrix}, \Pi_{k,\lambda} \begin{bmatrix} v_k \\ w_k \end{bmatrix} \right\rangle \geq 0 \Bigr\}$$

We also introduce the sets $\mathcal{E}_{k,\lambda}$

$$\mathcal{E}_{k,\lambda} = \Bigl\{ e_k \in \mathbf{L}_2^{l_k}[0,\infty) \mid \langle e, \Psi_{k,\lambda} e \rangle \geq 0 \Bigr\}$$

The essential idea of IQC analysis is to replace the operator Δ_k by the set-valued function $\mathcal{D}_{k,\lambda}$ and the external disturbance e_k by any signal from the set $\mathcal{E}_{k,\lambda}$. More specifically, we now consider the system which can be described as in Figure 2. All possible solutions (in $\mathbf{L}_2[0,\infty)$) of the original system are also valid solutions of the new system. This follows since, by definition,

- every $w_k = \Delta_k(v_k)$ belongs to $\mathcal{D}_{k,\lambda}(v_k)$
- every e_k belongs to $\mathcal{E}_{k,\lambda}$

The original system is now embedded in the IQC relaxation in Figure 2. It is sufficient to analyze the new system, and stability and worst-case performance of the new system implies the same properties hold for the original system.

Now let G_0 and G be the operators

$$\left[\begin{array}{c} G_0 \\ \hline G \end{array} \right] := \left[\begin{array}{ccc} G_{01} & \cdots & G_{0n} \\ \hline G_{11} & \cdots & G_{1n} \\ \vdots & \ddots & \vdots \\ G_{n1} & \cdots & G_{nn} \end{array} \right]$$

and let $v' = \begin{bmatrix} v_1' & \cdots & v_n' \end{bmatrix}$, $w' = \begin{bmatrix} w_1' & \cdots & w_n' \end{bmatrix}$, $e' = \begin{bmatrix} e_1' & \cdots & e_n' \end{bmatrix}$, and $e_t' = \begin{bmatrix} e_0' & e' \end{bmatrix}$. Also, define the following quadratic forms

$$\sigma_\Pi(v,w) = \sum_{k=1}^{n} \left\langle \begin{bmatrix} v_k \\ w_k \end{bmatrix}, \Pi_{k,\lambda} \begin{bmatrix} v_k \\ w_k \end{bmatrix} \right\rangle$$

$$\sigma_\Psi(e_t) = \sum_{k=0}^{n} \langle e_k, \Psi_{k,\lambda} e_k \rangle$$

Note that the quadratic forms σ_Π and σ_Ψ are linearly parameterized by λ.

[1] $\mathcal{P}(\mathbf{L}_2^{m_k}[0,\infty))$ denotes the set of all subsets of $\mathbf{L}_2^{m_k}[0,\infty)$

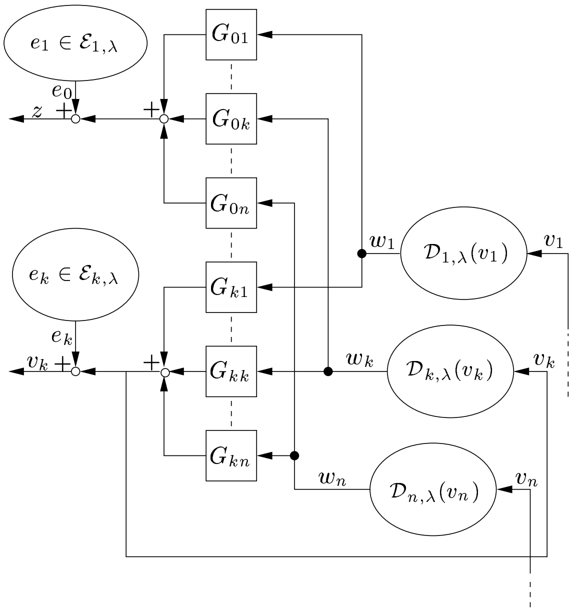

Figure 2: IQC relaxation of the system in Figure 1

In the IQCβ toolbox, checking stability is never the only objective of the analysis. Instead, stability is inferred from a robust performance problem. Typically, we consider the induced gain from some disturbance e_0 to a particular performance output z. If there exists a finite positive γ such that

$$\int_0^\infty (|z|^2 - \gamma|e_0|^2)dt \leq 0$$

for all possible solutions of the IQC relaxation, then under some weak conditions the system is stable and the induced gain is $\sqrt{\gamma}$. More generally, we can consider quadratic performance constraints of the following form

$$\sigma_\Theta(z, e_t) = \left\langle \begin{bmatrix} z \\ e_t \end{bmatrix}, \Theta \begin{bmatrix} z \\ e_t \end{bmatrix} \right\rangle$$

where Θ is a bounded, self-adjoint operator defined on $\mathbf{L}_2[0,\infty)$ space. What we would like to do is to check whether $\sigma_\Theta(z,e) \leq 0$ for all solutions of the system when $e_t \in \mathcal{E}_{0,\lambda} \times \cdots \times \mathcal{E}_{n,\lambda}$. This can be carried out by solving a constrained feasibility problem

$$\begin{aligned} &\text{Find } \lambda \in \Lambda, \quad \text{such that} \\ &\sigma_T(G_0 w + e_0, Gw + e, e_t, w) \leq 0, \ \forall (e_t, w) \in \mathbf{L}_2[0,\infty). \end{aligned}$$

where $\sigma_T(z, v, e_t, w) = \sigma_\Theta(z, e_t) + \sigma_\Pi(v, w) + \sigma_\Psi(e_t)$. This is a convex optimization problem that can be solved efficiently. Here, due to limitation of space, we will not discuss more specifically how the computational problem arises.

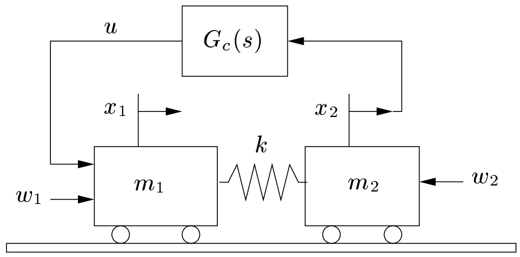

Figure 3: Two-mass-one-spring system

3 The Toolbox IQCβ

IQCβ is a collection of MATLAB functions designed to streamline rigorous computer-aided analysis of stability and performance of nonlinear, uncertain, and time-varying dynamical systems. Both the theoretical foundation and the language of the toolbox are based on the method of integral quadratic constraints, which was briefly introduced in the previous section. The toolbox is constructed based upon the principles that one can describe the system to be analyzed and the integral quadratic constraints necessary for analysis using MATLAB language in an straightforward and user-friendly fashion. The software supports handling of various abstract objects, such as signals, parameters, inequalities, or frequency-dependent quadratic forms, as MATLAB variables, using standard operations of addition, concatenation, sub-referencing, and so on. In this section, a simple example is used to illustrate how IQCβ works.

Consider the controlled two-mass-one-spring system in Figure 3. This example is adopted from the manual of MATLAB LMI Control Toolbox. The problem is to design an output-feedback law $u = G_c(s)x_2$ that adequately rejects the disturbances w_1, w_2, and guarantees stability for values of the stiffness parameter k in a certain given range.

The dynamics of the system can be described by a fourth order state-space model

$$\dot{x} = Ax + B_1(u + w_1) + B_2 w_2.$$

where the state vector x consists of the position and the velocity of masses m_1 and m_2. The matrices B_1 and B_2 are constant. Matrix A, which depends affinely on k, can be expressed as $A = A_0 + A_1 \bar{k} A_2'$, where A_0, A_1, A_2 are again constant matrices and $\bar{k}$ is an unknown constant in the range of $[-a, a]$. A fourth-order stabilizing controller $u = C_c(sI - A_c)^{-1} B_c x_2$ is designed for this system. The mass-spring control system can be expressed as the block diagram shown in Figure 4.

In the following, we will show how to verify stability of the control system using IQCβ . This is done by computing an estimate of the energy gain from $[w_1, w_2]$ to

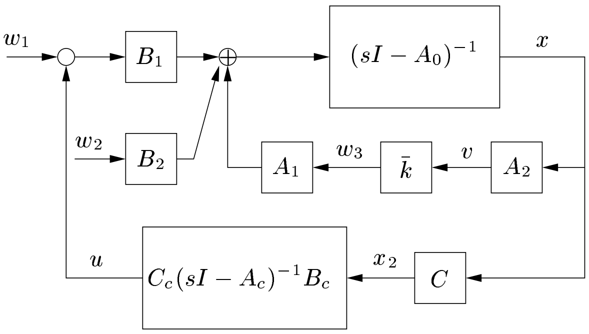

Figure 4: Block diagram of the mass-spring system

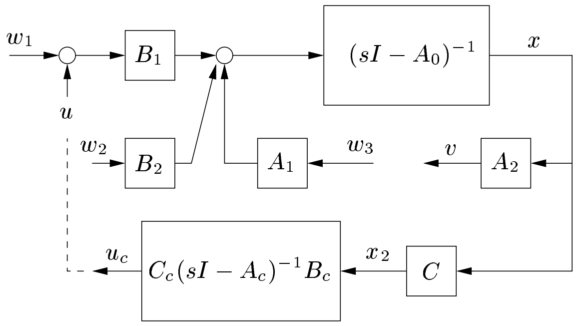

Figure 5: Constant LTI part of the mass-spring system

x_2. At current stage, checking stability is always performed by estimating the gain from some external disturbance to any internal signal in the feedback system. A finite gain implies stability.

We assume that matrices and parameters A_0, A_1, A_2, B_1, B_2, C, a, as well as the LTI systems $G_p(s)$ and $G_c(s)$, are already defined in the MATLAB workspace. We will also assume that the readers have experience of working with MATLAB Control Systems Toolbox.

The first step is to initialize the abstract IQC-environment. This is done by typing

```
>> abst_init_iqc;
```

The M-function `abst_init_iqc.m` tells MATLAB the "rules" of the abstract IQC-environment: which types of abstract data can be defined, which type of external data is allowed, and which operations can be done in the abstract environment.

The next step is to describe the system, i.e. to define how the signals in the system are related. First of all, we need to identify the blocks which correspond to the Δ_k blocks mentioned in the previous section. These blocks should be excluded at this stage, and later be replaced by set-valued functions defined using IQCs. In this example, $\bar{k}$ is uncertain and hence the $\bar{k}$ block is the one which should be removed. Secondly, we have to identify the so-called "basic signals" of the system. A basic signal is a signal that cannot be derived from any other signals through a constant LTI transformation. The external disturbances and the outputs of uncertain blocks are usually the sources of the basic signals. In this example, w_1, w_2 and w_3 are basic signals.

Sometimes, in order to facilitate the process of describing the system, extra basic signals are defined as dummies for signals which can be represented as LTI transformations of the other signals. Then, these signals are declared to be the same as their substitutes by using "==". For example, the two-mass-one-spring system in Figure 4 has a loop in which every signal is a LTI transformation of the others. One way to describe this loop is to break the loop as shown in Figure 5 and introduce an extra basic signal u. We can then define x, x_2, and u_c as LTI transformations of basic signals including u. In the end, we declare that u is the same as u_c by using "==". Therefore, we need 4 basic signals in this example: w_1, w_2, w_3, and u. To define these basic signals, one uses the M-function `signal.m`

```
>> w1=signal;
>> w2=signal;
>> w3=signal;
>> u=signal;
```

After defining basic signals, we can go forward and relate the remaining signals (x, x_2, v, and u_c) in the system to the basic signals we just defined.

```
>> x=Gp*(A1*w3+B1*(u+w1)+B2*w2);
>> v=A2*x;
>> x2=C*x;
>> uc=Gc*x2;
```

To close the control loop, we need to tell MATLAB that u and u_c are the same signals. This is done by the operation "=="

```
>> u==u_c;
```

The meaning of "==" is very different from the meaning of "=". If "=" would have been used instead, the result would simply be *re-assigning* u as an identifier to the signal identified by u_c, while the signal originally identified by u still remains in the memory (and has no identifier now!). With "==", the command `u==u_c` tells MATLAB that u and u_c are identical. This operation is called *linking* and should be used with caution. Wrong usage of this function will result in error.

The next step is to relate w_3 and v with a set-valued function which is defined using integral quadratic constraints. Since $|\bar{k}| \le a$, we know that

$$\int_0^\infty (a^2|v(t)|^2 - |w_3(t)|^2)dt \ge 0.$$

This inequality remains valid after multiplication by any scalar larger than zero. Hence, the following parameterized integral inequality holds for all pairs of (w_3, v) which satisfy $w_3 = \bar{k}v$

$$\int_0^\infty M \cdot (a^2 \cdot v^2 - w_3^2)dt \geq 0, \qquad (4)$$

where $M \geq 0$ is the corresponding λ parameter mentioned in the previous section.

In MATLAB, the analog to the set-valued function defined in the previous section is a sub-function in the main program. For the integral quadratic constraint in (4), the sub-function can be written as follows

```
function w=iqc_gainbounded(v,a)
w=signal(size(v,1));                            % (1)
M=variable;                                     % (2)
M>0;                                            % (3)
v'*a^2*M*v>w'*M*w;                              % (4)
```

where the first line defines a basic signal w which has the same size as signal v. The second line defines the parameter M. The M-function `variable.m` gives an analog of type 3 declaration in the LMI Control Toolbox, and its syntax follows exactly the convention of the LMI Control Toolbox. The third and fourth lines give the restriction on M and the integral quadratic constraint (4). Note that, for the sake of simplicity, "$>$" and "$<$" are used to replace "$\geq$" and "$\leq$" in IQCβ . Now, one can relate w_3 to signal v by

```
>> w3==iqc_gainbounded(v);                      % (5)
```

Note the operation `==` must be used to declare that w_3 is identical to the output of the sub-function, the basic signal w defined inside `iqc_gainbounded`. If instead the operation `=` was used, then an error would occur in the later stage of analysis.

The last step of the analysis is to execute the optimizer to estimate the L_2-gain from $[w_1; w_2]$ to x_2. This is done by the M-function `iqc_gain_tbx.m`.

```
>> g=iqc_gain_tbx([w1;w2],x2)
```

The execution of `iqc_gain_tbx` trigger the following events: first, the script with IQCβ input between the commands `g=abst_init_iqc` and `g=iqc_gain_tbx([w1;w2],x2)` are collected and processed to form an optimization problem which corresponds to the worst-case $\mathbf{L}_2$-gain estimation problem. The optimization problem is in the form of the semi-definite program (SDP). Secondly, the SDP solver of the MATLAB LMI Control Toolbox is called to solve the optimization problem.

More precisely, an attempt will be made to choose the variable parameters (M and g in this case, where g is defined inside the M-function `iqc_gain_tbx.m`) so that

$$\int_0^\infty (g(w_1^2 + w_2^2) - x_2^2)dt > \int_0^\infty M(a^2v^2 - w_3^2)dt$$

holds for all non-zero $\mathbf{L}_2$ signals w_1, w_2, w_3. In fact, the attempt to minimize g will also be made, and the SDP solver will try to find the smallest possible value of g.

If the optimization problem is found feasible, an value of g will be returned. Otherwise, g is empty. For more details about IQCβ , we refer to the manual [5].

`abst_init_iqc;`	initialization

`w1=signal;`
`w2=signal;`
`w3=signal;`
`u=signal;`
	define basic inputs

`x=Gp*(A1*w3+B1*(u+w1)+B2*w2);`
`v=A2*x;`
`x2=C*x;`
`uc=Gc*x2;`
`u==uc;`
	describe the LTI part of the system

`w3==iqc_gainbounded(v,a);`	define IQC for the uncertainty $\bar{k}$
`g=iqc_gain_tbx([w1;w2],x2)`	compile and solve

`function w=iqc_gainbounded(v,a)`
`w=signal(size(v,1));`
`M=variable;`
`M>0;`
`v'*a*a*M*v>w'*M*w;`
	the sub-function that defines IQC

Figure 6: IQCβ scripts for the mass-spring example

4 The Scope of IQCβ

From an implementation point of view, the most essential idea in IQCβ is the introduction of a new MATLAB class "abst". In IQCβ , the variables from the "abst" class are used to define abstract entities such as signals, variables, quadratic forms, integral quadratic constraints, and so on, which are the essential parts in IQC analysis. However, the "abst" class is not limited to be used only for IQC analysis; it can also be used to describe a variety of other mathematical structures which are not associated with systems analysis at all.

The main purpose of introducing the "abst" class is to allow accumulation of mathematical information *without* actually performing any operations. When asked to perform a certain operation on "abst" variables, MATLAB checks whether this operation is allowed and, if the operation is allowed, writes a record of the operation into a global matrix variable "ABST". No other action is performed. For example, if x and y belong to the "abst" class, the statement $z = x + y$ defines z as the sum of x and y. However, no addition is actually performed, only a record of addition is made in the global variable "ABST".

As a data entry, each variable of the "abst" class contains a single field, which is an integer number referring to a row in the global variable "ABST". The true information is accumulated in "ABST", which users of IQCβ are not supposed to access directly. What users may do is to *initialize* a particular abstract environment, to *define* and/or to *derive* relevant abstract variables, and finally to *compile* the total set of "abst" operations performed by certain M-functions provided by IQCβ .

Three different abstract environments, which are built for handling different mathematical objects and structures, can be initialized.

- The `lmi` environment: this environment is designed to handle standard semi-definite programs. See [5] for details.
- The `fdlmi` environment: this environment is designed to handle optimization problems with frequency-dependent affine matrix inequalities. See [5] for details.
- The `iqc` environment: this environment is designed to handle stability and performance analysis in the IQC framework. This is the abstract environment which IQCβ is designed for.

5 The IQC Library and Other Extensions

One of the most important features of the IQC approach is its ability to localize the analysis of complex dynamical systems to their elementary building blocks. In essence, what we do is to discover/derive integral quadratic constraints for the non-LTI part of the system to be analyzed. These IQCs are then used to build an optimization problem for system analysis, which can be solved numerically. The beauty of this approach is that the IQCs for the building blocks of a system can be stored and reused for other systems which have the same building blocks.

IQCβ provides a library of pre-defined IQC descriptions of many typical blocks commonly seen in practical engineering applications, such as saturation, unmodelled LTI dynamics, constant uncertain delay, slowly/fast time-varying parameters, and so on. This library is written using the language of IQCβ and stored as M-functions. Users can of course add more elements to the library, or build one on their own. A listing and descriptions of the available block descriptions can be found in [5].

There is also a graphical user interface (GUI) to IQCβ , which is based on SIMULINK2 or higher versions. To use the GUI of IQCβ , one first opens the SIMULINK library `iqc_lib`, which contains some standard LTI SIMULINK blocks and a number of special IQC blocks. Using these blocks, users can create a SIMULINK block diagram. After completing the SIMULINK diagram, save the block diagram and run

```
>> iqc_gui('name_of_the_diagram')
```

where "`name_of_the_diagram`" is the name of the block diagram without the `.mdl` extension. The program `iqc_gui.m` will process the block diagram, create and run a set of IQCβ scripts for analyzing stability and performance of the system.

Due to its convenient interface to the LMI Control Toolbox, some researchers have written programs for their research using IQCβ language. One such example is Jorge Gonçalves' *global analysis of relay, on/off, and saturation feedback systems* [1]. Gonçalves' toolbox, which extends IQCβ 's power in complex system analysis to certain piecewise linear systems, can be found in IQCβ under the subdirectory `applications`.

IQCβ will only run under MATLAB version 5.2 or higher. Using IQCβ also requires the Control Systems Toolbox. At this moment, it also requires the LMI Control Toolbox as the major supported optimization engine. Alternative optimization options are being developed. Specialized solvers based on cutting plane type of algorithms [4] can be obtained by e-mailing the authors and will soon be a part of IQCβ . Another specialized fast solver for IQC optimization problems, developed by A. Hansson, L. Vandenberghe and their associates [2], is also being interfaced with IQCβ .

6 Example

This example is about robustness analysis of the Space Shuttle lateral axis flight control system during the reentry. The system is a simplified model of the space shuttle, in the final stage of landing, as it transitions from supersonic to subsonic speeds. This example is adapted from the manual of the MATLAB μ-Analysis and Synthesis Toolbox.

Figure 7 shows the Space Shuttle lateral axis flight control system. The rigid body model for the aircraft at

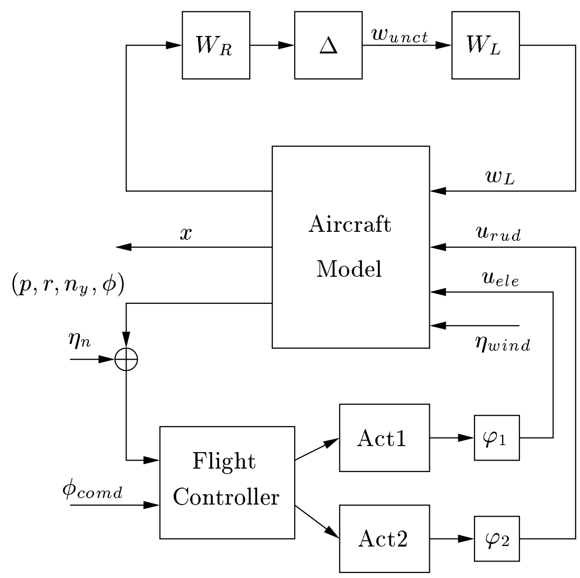

Figure 7: Schematic diagram of the lateral flight control system of the Space Shuttle.

Mach 0.9 is a four-state system with states

$$x = \begin{bmatrix} \beta \\ p \\ r \\ \phi \end{bmatrix} = \begin{bmatrix} \text{sideslip angle} \\ \text{roll rate} \\ \text{yaw rate} \\ \text{bank angle} \end{bmatrix}.$$

There are three inputs to the aircraft: the actual angular deflection of the elevon surface, the actual deflection of the rudder surface, and a lateral wind gust disturbance input η_{wind}, due to the winds that occur at this altitude. The wind disturbance is assumed to have dominating harmonics between $\underline{m}$ Hz and $\overline{m}$ Hz; i.e., the spectrum of the signal is concentrated in the frequency band $[\underline{m},\overline{m}]$ Hz.

The major source of uncertainty in the aircraft model is in the aerodynamic coefficients. There are standard aerodynamic parameters which express incremental forces and torques generated by incremental changes in sideslip, elevon, and rudder angles. This is a linear relationship, expressed as

$$\begin{bmatrix} \text{side force} \\ \text{yawing moment} \\ \text{rolling moment} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix} \begin{bmatrix} \beta \\ \theta_{ele} \\ \theta_{rud} \end{bmatrix}.$$

Uncertainty in these coefficients is modelled as a nominal value, plus a perturbation term

$$\begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix} = \bar{C} + W_L \Delta W_R,$$

where W_R and W_L are weighting matrices of dimension 3×9 and 9×3, respectively. Δ is a diagonal uncertainty; i.e., $\Delta = \mathrm{diag}(\delta_1, \cdots, \delta_9)$, and the each δ_i is assumed to be a fixed, unknown real parameter with $|\delta_i| \leq 1$.

The controller has four noisy sensor inputs and one command input. Three of the sensor inputs are states: (p, r, ϕ). The fourth sensor input is the lateral acceleration at the pilot's location, denoted by n_y. The command input is the band-angle command. There are two outputs of the controller. One is the rudder command, and the other is the elevon command. In this example, the controller design is based on weighted $\mathcal{H}_\infty$ optimization and then Hankel model order reduction is used to remove states which have little influence on the dynamical behavior of the controller. The resulting controller has 19 states.

The control signals are modified by two identical nonlinear operators

$$\phi_i(x) = \begin{cases} x, & \text{if } |x| \leq 1 \\ 1, & \text{if } x > 1 \\ -1, & \text{if } x < -1 \end{cases}$$

before they enter the aircraft model. The purpose of these nonlinearities is to model the effect of actuator saturation. Note that in the manual of the μ-Analysis and Synthesis toolbox where this example is taken, the effect of actuator saturation is not considered. Here we add these nonlinearities to make the setup more realistic.

In this example, we are interested in verifying stability of the flight control system subject to the uncertainty in aerodynamic coefficients and actuator saturation. Furthermore, we also want to compute an upper bound of the $\mathbf{L}_2$-gain from the disturbance η_{wind} to the state vector x. In this example, the non-LTI part of the systems consists of 9 unknown constants and two identical saturation nonlinearities, which we shall characterize using IQCs. Furthermore, the disturbance η_{wind} have dominating harmonics between $\underline{m}$ Hz and $\overline{m}$ Hz. One can also characterize this property using IQCs.

There are numbers of IQCs one can choose from to characterized the uncertain parameters δ_i, the saturation nonlinearities $\phi_i(\cdot)$, and the wind disturbance η_{wind}. For each uncertain parameters, IQCs defined by multipliers of the following form

$$\Pi(j\omega) = \begin{bmatrix} \hat{x}(j\omega) & \hat{y}(j\omega) \\ \hat{y}(j\omega)^* & -\hat{x}(j\omega) \end{bmatrix}$$

can be used, where $\hat{x}(j\omega)$ and $\hat{y}(j\omega)$ are rational transfer functions satisfying $\hat{x}(j\omega) = \hat{x}(j\omega)^* > 0$, $\hat{y}(j\omega) + \hat{y}(j\omega)^* = 0$, $\forall\omega$. The saturation nonlinearities $\phi_i(\cdot)$ can be characterized using the so-called Zames-Falb's IQCs [6]. Furthermore, since $\phi_1(\cdot)$ and $\phi_2(\cdot)$ are identical, one can exploit this feature and construct more sophisticated IQCs, see e.g. [3]. The Zames-Falb's IQCs have multipliers of the form

$$\Pi(j\omega) = \begin{bmatrix} 0 & h_0 - \hat{h}(j\omega) \\ h_0 - \hat{h}(j\omega)^* & -2(h_0 - \mathbf{Re}\hat{h}(j\omega)) \end{bmatrix}$$

where h_0 is a positive constant and $\hat{h}(j\omega)$ is a rational transfer function whose impulse response $h(t)$ satisfies

$$\int_{-\infty}^{\infty} |h(t)| dt \leq h_0$$

For the form of the more sophisticated IQCs for identical saturation nonlinearities, readers are referred to [3].

The disturbance η_{wind} satisfies the IQC defined by the multiplier

$$\Psi = \lambda_1 H(j\omega)^* H(j\omega) - \lambda_2$$

where H is a Butterworth bandpass filter with cut-off frequencies at $\omega_1 = a/n$ and $\omega_2 = nb$. Here n is a user-defined parameter. Parameters λ_1 and λ_2 must satisfy the following inequalities

$$\lambda_1 \mathbf{Re}(H(ja)) \geq \lambda_2, \ \lambda_1 \mathbf{Re}(H(jb)) \geq \lambda_2, \ \lambda_1, \lambda_2 \geq 0$$

To define the IQCs mentioned above, one can use the following M-functions from the IQC library

- `iqc_ltvnorm.m` or `iqc_ltigain.m` for uncertain parameters δ_i.
- `iqc_monotonic.m` or `iqc_d_slope_odd.m` for the two identical saturation nonlinearities.
- `iqc_domharmonic.m` for wind disturbance with dominating harmonics.

The $\mathbf{L}_2$-gain estimation problem can now be easily formulated in IQCβ language. Figure 8 shows such an example, where the transfer matrices for the aircraft (`SC`), the flight controller (`CR`), actuators (`AC1` and `AC2`), and matrices `WL`, `WR` are assumed to be defined in the MATLAB workspace already.

Without the saturation nonlinearities, we are able to verified stability of the control system for $|\delta_i| \leq 1$. The $\mathbf{L}_2$-gain from η_{wind} to x is estimated to be 0.0014. With the the saturation nonlinearities, stability of the control system can only be verified for $|\delta_i| \leq 0.174$, where the $\mathbf{L}_2$-gain from η_{wind} to x is estimated to be 0.48. In this case, we use the IQC derived in [3] for the saturation nonlinearities and simple gained bounded IQCs for the uncertain parameters (i.e., $\hat{x}(j\omega) = \lambda > 0$, and $\hat{y}(j\omega) = 0$). Notice, however, that this does not necessarily mean that the control system is unstable for $|\delta_i| > 0.15$. It only means that the IQCs we used was not good enough for proving stability. Indeed, the success of the analysis is dependent on the choice of dynamic basis for the IQCs. The script in Figure 8 uses the default non-dynamic multipliers for the uncertain gains but nontrivial IQCs for the saturation nonlinearities and the disturbance signal. We refer to [5] for details about the individual IQC block descriptions.

```
abst_init_iqc;

w_unct = signal(9);
w_satu = signal(2);
w_wind = signal;

w_L = WL*w_unct;
output_SS = SC*[w_L;w_satu;w_wind];
v_1 = WR*output_SS(1:3);
v_2 = output_SS(4:7);
output_ctrl = CR*v_2;
output_ac1 = AC1*output_ctrl(1);
output_ac2 = AC1*output_ctrl(2);

for flcnt =1:9
  w_unct(flcnt) == iqc_ltigain(v_1(flcnt));
end
w_satu == iqc_d_slope_odd([output_ac1;...
   output_ac2],[1.2],[0.1];[0.1],[1.2],0,1);
w_wind == iqc_domharmonic(1,1/10,10,2,2);

iqc_gain_tbx(w_wind,v_2);
```

Figure 8: IQCβ scripts for the space shuttle example

References

[1] Jorge Gonçalves. *Constructive Global Analysis of Hybrid Systems*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[2] A. Hansson and L. Vandenberghe. A primial-dual potential reduction method for integral quadratic constraints. In *Proceedings of the 2001 American Control Conference*, pages 3013–3017, Arlington, VA, June 2001.

[3] F. J. D'Amato, M. A. Rotea, A. Megretski, and U. Jönsson. New results for analysis of systems with repeated nonlinearities. *Automatica*, 37(5):739–747, 2001.

[4] Chung-Yao Kao. *Efficient Computational Algorithms for Robustness Analysis*. PhD thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, 2002.

[5] A. Megretski, C.-Y. Kao, U. Jönsson, and A. Rantzer. A guide to IQCβ: Software for robust analysis. August 1999. Manual of IQC Toolbox.

[6] A. Megretski and A. Rantzer. System analysis via Integral Quadratic Constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, June 1997.