Preprints of the 3rd IFAC Workshop on
Distributed Estimation and Control in Networked Systems
September 14-15, 2012. Santa Barbara, CA, USA

Networked Control Systems Toolbox: Robust Stability Analysis Made Easy ⋆

N.W. Bauer * **S.J.L.M. van Loon** * **M.C.F. Donkers** *
N. van de Wouw * **W.P.M.H. Heemels** *

* *Department of Mechanical Engineering, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, Netherlands.*

Abstract: This paper presents the first prototype of a toolbox developed to automate stability analysis for networked control systems. Specifically, the toolbox can be employed to efficiently verify if a linear time-invariant (LTI) plant and an LTI controller interconnected with a shared network is robust to certain network imperfections. The purpose is to make the theory available in the literature readily accessible to and applicable for the general control community. Additionally, the software is structured in such a way that it is possible to incorporate custom models or use custom stability/performance analysis conditions, enabling the control community to contribute to the toolbox.

Keywords: Networked Control Systems, Stability Analysis, Numerical Toolbox

1. INTRODUCTION

In recent years, there have been many theoretical developments in the area of stability analysis for networked control systems (NCSs), however, computational tools which implement the theory are lacking. For this reason, we have developed a prototype MATLAB toolbox with the primary goal of increasing the amount of people who can interact with and use the stability theory for NCSs in a simple manner. In fact, there are multiple other reasons for creating an NCS toolbox: (i) to provide a user-friendly way to interact with the existing theoretical developments; (ii) to remove the burden of implementing some of the more complex algorithms employed in the theory; (iii) to gain feedback from the community to improve the quality and overall usefulness of both the toolbox and the theory. To appeal to the widest range of people, we have designed the toolbox such that it is useful for both the basic user and the advanced user.

To appeal to the basic user, we have completely automated the stability analysis procedure such that the user is only required to input a plant model, a controller model, and bounds on the network uncertainties (described in Section 2) and can directly start analyzing stability. Moreover, we have recognized that NCS models can contain a considerable amount of variables, which can be intimidating to someone who is not familiar with the NCS nomenclature. Therefore, we created a graphical user interface (GUI), see Fig. 3, such that the variables are displayed and can be edited in a very intuitive manner. An additional stumbling block that is encountered when learning about NCS theory is related to the fact that there exist several mathematical techniques used for analysis. To intuitively give an overview of two mathematical frameworks (based on discrete-time switched systems and continuous-time hybrid models) used for analysis and the techniques available therein, an additional analysis GUI was created, see Fig. 4.

To appeal to the advanced user, we enabled the use of functions outside the GUI to aid in analyzing more specific problem settings. First, by providing a function which can analyze stability, researchers can write their own programs which can visualize entire stability regions for their problem of interest. Second, to encourage community participation in the future development of the toolbox, we made it possible for the researcher to incorporate their own discrete-time models (provided that the network-induced uncertainties enter the model in the appropriate way, described below). In the discrete-time framework, a polytopic overapproximation of the closed-loop NCS model is necessary before linear matrix inequalities can be used to determine if stability can be guaranteed. Acquiring a polytopic overapproximation (possibly with norm-bounded uncertainty) is the most tedious aspect of implementing a discrete-time approach to stability analysis of NCSs. The toolbox in this paper automates this overapproximation procedure for a general class of models using a choice of different techniques and allows the NCS community to use the resulting overapproximation for analysis of customized stability or performance properties.

The MATLAB toolbox presented in this paper has automated the theoretical developments in Cloosterman et al. (2009); Donkers et al. (2011); Heemels et al. (2010a); Loon et al. (2012). In Cloosterman et al. (2009); Donkers et al. (2011); Loon et al. (2012) a discrete-time framework was used and in Heemels et al. (2010a) a continuous-time hybrid framework (built upon the work in Nešić and Teel (2004)) was used to analyze stability properties of NCSs. In the discrete-time framework it has been shown that the amount of conservatism when analyzing stability can be minimized, however this framework is limited to the analysis of only linear plants and (switched) linear controllers. On the other hand, the hybrid formulation has the

⋆ This work is partially supported by the Innovational Research Incentives Scheme under the VICI grant "Wireless control systems: A new frontier in automation" (no. 11382) awarded by NWO (Netherlands Organization for Scientific Research) and STW (Dutch Science Foundation), by the European Union 7th Framework Programme [FP7/2007-2013], under grant agreement no. 257462 HYCON2 Network of excellence.

advantage of being able to analyze general nonlinear plants and controllers and study $\mathcal{L}_p$-gain type of performance criterion. So it is beneficial to consider both frameworks. However, due to space limitations, in this paper, we focus on the toolbox implementation of the discrete-time framework even though the analysis tools in the hybrid framework have been implemented for linear plants and controllers as well. By using this toolbox, the user is able to make multi-disciplinary design tradeoffs between control properties such as stability and performance, and network-related properties such as delays, scheduling, bandwidth limitations etc., in a user-friendly manner. Finally, we would like to emphasize here that there are many other results in the NCS literature which either use the discrete-time or hybrid framework that are currently not implemented in the the toolbox. We hope that the prototype toolbox presented here serves as a platform to which the NCS community can contribute.

The outline of this paper is as follows. In Section 2, a description of the NCS as implemented in the toolbox will be given; then in Section 3, the software structure will be presented. In Section 4, we describe how the basic user can quickly assess robust stability of an NCS model. In Section 5, we describe how the advanced user can plot stability regions and how custom models can be incorporated. In Section 6, we give an example of how the prototype toolbox can be used to investigate the conservatism introduced via one of the overapproximation techniques and demonstrate how stability regions can be plotted. Finally, conclusions and future developments are presented in Section 7.

2. DESCRIPTION OF THE NCS

The general schematic of an NCS is depicted in Fig. 1. It consists of a linear time-invariant (LTI) continuous-time plant and an LTI linear controller, which are interconnected through a communication network that induces

- uncertain time-varying transmission intervals h in the range $[h_{min}, h_{max}]$;
- uncertain time-varying network delays τ in the range $[\tau_{min}, \min\{h, \tau_{max}\}]$;
- uncertain dropouts sequences, where the number of successive dropouts is bounded by δ_{max};
- quantization;
- a shared communication medium, which prevents all sensor and actuator signals from being transmitted simultaneously.

Due to the communication medium being shared, the sensors and actuators are grouped into N nodes. At each transmission instant, one node obtains access to the network and transmits its corresponding values (implied by the red arrow shown in Fig. 1). Note that if there is only one node which contains all sensors and actuators, then the 'classical' control setup, where all sensor and actuator signals are transmitted at each transmission time, is recovered. In case of multiple nodes, a mechanism is needed that orchestrates which node is given access, when the network is available. This mechanism is called a scheduling protocol. In this toolbox, we provide analysis techniques for two well-known protocols, namely, the Round Robin (RR) protocol and the Try-Once-Discard (TOD) protocol, see Walsh et al. (2002).

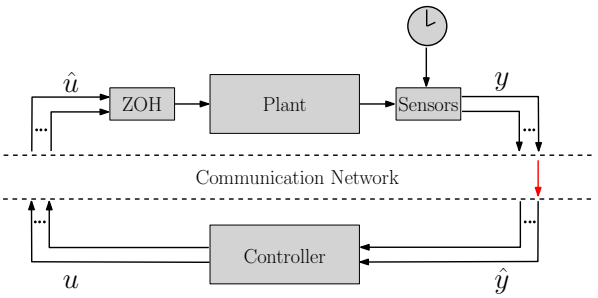


Fig. 1. Block Schematic of a networked control system.

Incorporating the communication network (including the protocol) between the plant and controller, leads to the following operation aspects of the NCS in Fig. 1. First, the sensor acts in a time-driven fashion (i.e. sends data at each transmission instant) and both the controller and actuator act in an event-driven fashion (i.e. they respond instantaneously to newly arrived data). Second, the dropouts are modeled as prolongations of the sampling interval, meaning that, if a packet is considered 'dropped' then a new packet is transmitted at the next transmission time with new data. Third, the discrete-time control commands are converted to a continuous-time control signal by using a zero-order-hold (ZOH) function, see Fig. 1. Finally, the delays are assumed to be smaller than the transmission intervals. Note that retransmissions of packets can be modeled as prolongations of the delays. More information regarding the mathematical modeling and assumptions can be found in Cloosterman et al. (2009); Donkers et al. (2011); Heemels et al. (2010a); Loon et al. (2012).

3. SOFTWARE STRUCTURE

The overall structure of the MATLAB software based on the discrete-time NCS framework is given in Fig. 2. What the toolbox currently offers is the possibility to analyze three different controller structures via three different overapproximation techniques and assess stability of two standard communication protocols. By carefully considering and implementing the software structure, horizontal expansion (e.g. including different controllers/models, different overapproximation techniques and different stability analysis conditions) is straightforward since the vertical links are defined in a generalized manner. Adopting this structure, we pave the road for the addition of custom models and create the possibility for using the resulting overapproximations in customized conditions for analysis (e.g. analysis of different protocols). In this way, we hope to encourage the NCS community to use and contribute to the toolbox as well.

The structure of the toolbox reflects the goal of appealing to both the basic user and the advanced user. The basic user can simply interact with the first and last layer of the toolbox through the GUIs. However, for advanced users, the set of standard models provided with this toolbox might not include the exact model or stability analysis technique they are interested in studying. We provide the possibility for the advanced user to incorporate their own model and/or use their own stability or performance techniques, as will be discussed in Section 5.2.

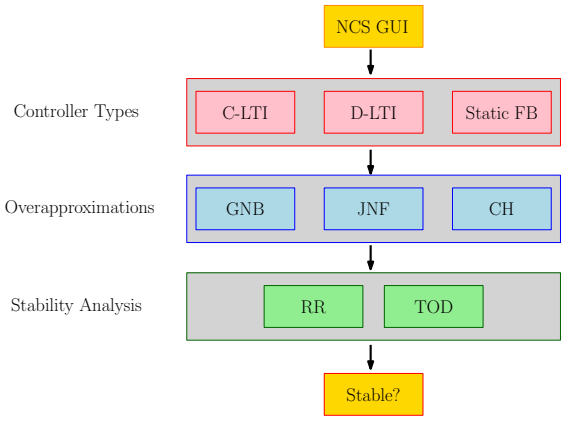


Fig. 2. The software structure for the discrete-time NCS framework.

4. BASIC FUNCTIONALITY

The NCS Toolbox contains a number of features which make it easy to efficiently verify whether an LTI plant and an LTI controller interconnected with a network are robust to the aforementioned network imperfections. This is extremely convenient to someone who designed a linear controller and wants to quickly verify these robustness properties. We provide easy model management, overapproximation automation and automated stability verification.

4.1 Model Management

An NCS class object (which describes the model covered in Section 2) consists of an LTI plant, an LTI controller and network variables (bounds on time-varying sampling intervals, bounds on time-varying delays, a bound on the maximum number of successive dropouts, the type of quantizer, the node definitions and the protocol). Creating an NCS object is easily done by using the NCS Editor (GUI) shown in Fig. 3.

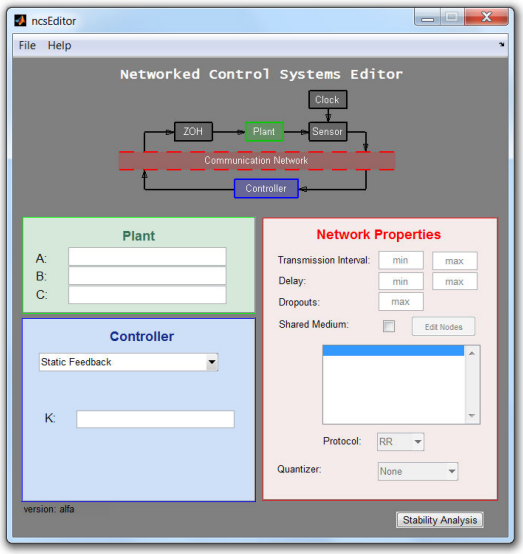


Fig. 3. Graphical user interface to define an NCS object.

To display the NCS Editor, simply type `ncsEditor` in the MATLAB command line and the GUI will appear. Here the NCS properties can be defined and an NCS object can be exported to the MATLAB workspace by clicking 'File > Export'. Modifications to an NCS object can be made by clicking on 'File > Import'. We will now briefly discuss the specific elements of the NCS object.

Plant In this toolbox the plant, as shown in Fig. 1, is a LTI continuous-time system expressed as

$$\dot{x}(t) = Ax(t) + B\hat{u}(t),$$
$$y(t) = Cx(t).$$

The matrices A, B and C are the first input parameters to define an NCS object. In the related input boxes, see Fig. 3, either matrices can be input directly or variables defined in the MATLAB workspace can be used.

Controller Next, the controller, as shown in Fig. 1, needs to be specified. The type can be defined by clicking on a choice from the drop down dialog box, see Fig. 3. There are three possible controller types currently available in this toolbox. The first is 'Static Feedback' where the feedback law is of the form $u(t) = K\hat{y}(t)$. The second controller type is a 'C-LTI Dynamic Feedback' control law, which is given by

$$\dot{x}^c(t) = A^c x^c(t) + B^c \hat{y}(t),$$
$$u(t) = C^c x^c(t) + D^c \hat{y}(t).$$

The third controller type is a 'D-LTI Dynamic Feedback' control law, which is given by

$$x^c_{k+1} = A^c x^c_k + B^c \hat{y}_k,$$
$$u(t_k) = C^c x^c_k + D^c \hat{y}(t_k),$$

where t_k is the k^{th} transmission time. The signals $\hat{y}$ and $\hat{u}$ result from transmitting y and u, respectively, through the network, as depicted in Fig. 1. The matrices K or A_c, B_c, C_c and D_c are entered into the corresponding text boxes, see Fig. 3. In these text boxes, either matrices can be input directly or variables defined in the workspace can be used. Being able to input variables from the workspace is convenient if the controller was designed using a different tool (e.g. based on loop-shaping tools).

Network Lastly, the network effects need to be defined. The bounds on the time-varying transmission intervals and time-varying delays are entered into the corresponding input box, Fig. 3. If the transmission intervals or delays are considered constant then the 'min' value can be set equal to the 'max' value. If delays are not considered, then both the 'min' value and the 'max' value can be set equal to zero. Next, a bound on successive packet dropouts needs to be specified. If dropouts do not occur, this value can be set to zero.

Next, there is a check box to indicate whether or not the communication medium is shared (restricting all actuators and sensors from communicating at the same transmission time). Once this box is checked, the button 'Define Nodes' is enabled. Another GUI will appear where the nodes can be defined by 'adding' inputs and outputs to each node. Nodes can be added or removed and moved up and down to change the ordering. Next, in the NCS Editor, the protocol needs to be chosen as 'RR' for Round Robin or 'TOD' for Try-Once-Discard.

Finally, a type of quantizer can be chosen from a drop down menu as 'none', 'uniform' or 'logarithmic'. For further information about quantizer modeling and analysis, see Loon et al. (2012).

Once all the fields are filled in, all the NCS parameters can be exported to the workspace in a single NCS object and saved for later use.

4.2 Automated Overapproximation

The discrete-time closed-loop NCS models given in Cloosterman et al. (2009); Donkers et al. (2011); Loon et al. (2012) can all be expressed in the general form

$$\bar{x}_{k+1} = \left(A_{\sigma_k} + B_{\sigma_k} \begin{bmatrix} A_{h_k} & 0 & 0 \\ 0 & E_{h_k} & 0 \\ 0 & 0 & E_{h_k-\tau_k} \end{bmatrix} C_{\sigma_k} \right) \bar{x}_k + \left(E_{\sigma_k} + D_{\sigma_k} \begin{bmatrix} A_{h_k} & 0 & 0 \\ 0 & E_{h_k} & 0 \\ 0 & 0 & E_{h_k-\tau_k} \end{bmatrix} J_{\sigma_k} \right) \omega_k \quad (1)$$

where $k \in \mathbb{N}$ is a counter related to the number of transmissions, $\sigma_k \in \{1, ..., N\}$ is the node which receives network access at transmission time t_k, $h_k \in [h_{min}, h_{max}]$ is the sampling interval at the k^{th} transmission time, $\tau_k \in [\tau_{min}, \tau_{max}]$ is the delay at the k^{th} transmission time, $\omega_k \in \mathbb{R}^{n_\omega}$ is a disturbance on the closed-loop system and $A_\rho = e^{\tilde{A}\rho}$ and $E_\rho = \int_0^\rho e^{\tilde{A}s} ds$ for some matrix $\tilde{A}$. Due to the exponential form in which the uncertainty parameters h_k and τ_k appear in (1), i.e. in the matrices $A_{h_k} \in \mathbf{\Gamma}_1$, $E_{h_k} \in \mathbf{\Gamma}_2$, and $E_{h_k-\tau_k} \in \mathbf{\Gamma}_3$, where

$$\mathbf{\Gamma}_1 := \left\{ e^{\tilde{A}h} \mid h \in [h_{min}, h_{max}] \right\},$$

$$\mathbf{\Gamma}_2 := \left\{ \int_0^h e^{\tilde{A}s} ds \mid h \in [h_{min}, h_{max}] \right\},$$

$$\mathbf{\Gamma}_3 := \left\{ \int_0^{h-\tau} e^{\tilde{A}s} ds \mid h \in [h_{min}, h_{max}], \tau \in [\tau_{min}, \tau_{max}] \right\},$$

the discrete-time model (1) is not directly suitable to construct LMI conditions for stability verification. To make the model amendable for LMI-based stability analysis, we aim to overapproximate the set of matrices $\mathbf{\Gamma}_i$, $i = 1, 2, 3$, as

$$\mathbf{\Gamma}_i \subseteq \left\{ \sum_{l=1}^{L} \alpha_l \bar{F}_{i,l} + \bar{G}_i \Delta_i \bar{H}_i \mid \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_L \end{bmatrix} \in \mathcal{A},\ \Delta_i \in \mathbf{\Delta} \right\},$$

where $F_{i,l}$, G_i, H_i, $l = 1, ..., L$, $i = 1, 2, 3$, are suitably constructed matrices with L the number of vertices in the polytopic overapproximation. In addition, $\mathbf{\Delta}$ is a specific set of structured matrices (e.g. with a norm bound $||\Delta|| := \sqrt{\lambda_{max}(\Delta^\top \Delta)} \leq 1$) and

$$\mathcal{A} = \left\{ \alpha \in \mathbb{R}^L \mid \alpha_l \geq 0,\ l = 1, ..., L \text{ and } \sum_{l=1}^{L} \alpha_l = 1 \right\}.$$

In the toolbox, we provide techniques to overapproximate the matrix sets $\mathbf{\Gamma}_i$ leading to the transformation of the closed-loop system (1) into

$$\bar{x}_{k+1} = \left(\sum_{l=1}^{L} \alpha_k^l \bar{A}_{\sigma_k,l} + \bar{B}_{\sigma_k} \bar{\Delta}_k \bar{C}_{\sigma_k} \right) \bar{x}_k + \left(\sum_{l=1}^{L} \alpha_k^l \bar{E}_{\sigma_k,l} + \bar{D}_{\sigma_k} \bar{\Delta}_k \bar{J}_{\sigma_k} \right) \omega_k \quad (2)$$

for $l \in \{1, ..., L\}$ and $\sigma \in \{1, ..., N\}$ with $\alpha_k^l \in \mathcal{A}$ and $\bar{\Delta}_k \in \mathbf{\Delta}$, $k \in \mathbb{N}$. Three overapproximation techniques are automated in this toolbox: an approach based on gridding and norm bounding (GNB), see, e.g., Donkers et al. (2011), an approach based on the Jordan normal form (JNF), see, e.g., Cloosterman et al. (2009) and an approach based on the Cayley-Hamilton theorem (CH), see, e.g., Gielen et al. (2010). For a theoretical comparison between these three methods, the reader is referred to Heemels et al. (2010b).

Although each of these three techniques are mathematically interesting, a fairly strong familiarity with the notation is required in order to implement (and actually use) these techniques in software. So for the control engineer who would like to determine if their specific closed-loop system is robust to network-induced effects, it will cost him or her a significant amount of time understanding these techniques and implementing them. The NCS toolbox allows the control engineer to quickly verify if his control setup possesses robustness properties without having to know the all the (mathematical) details about a polytopic overapproximation besides the basic idea and the fact that they may introduce some conservatism.

4.3 Automated Stability Verification

To analyze stability of the NCS, linear matrix inequality (LMI) conditions are verified on the polytopic overapproximation described in the previous section. The LMI conditions are verified using the YALMIP interface, Löfberg (2004), with the SeDuMi solver, Sturm (1999). Stability of an NCS configuration can be assessed directly from the NCS Editor by clicking on the button 'Analyze Stability' in the lower left corner of the NCS Editor shown in Fig. 3 which opens the NCS Analyzer (GUI) shown in Fig. 4.

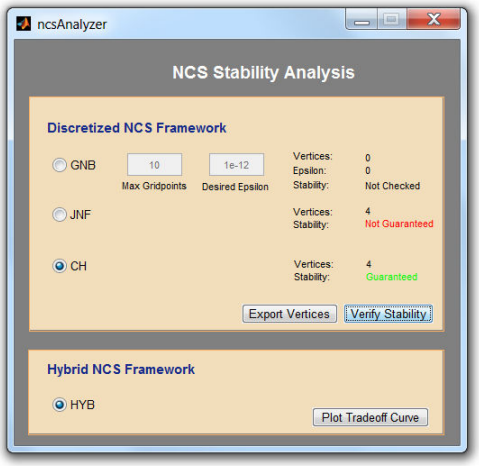


Fig. 4. Graphical user interface to determine stability.

The NCS Analyzer was designed such that a user who is not familiar with the theory can quickly glance at the layout and intuitively understand that there are two main frameworks and that each framework has multiple options to choose from. In this way, a basic user can easily understand the overview of modeling and stability techniques of an NCS without having to first understand any of the mathematical details. Verifying closed-loop stability of the NCS can be done by clicking on the 'Verify Stability' button. After the stability verification is complete, the result is displayed on the rightmost column (see Fig. 4). By making it easy to test stability with various options, the user can understand that some methods fail at proving stability while others succeed due to different levels of conservatism introduced by the different methods.

5. ADVANCED FUNCTIONALITY

Of course, having a GUI to interact with the NCS theory is useful to people who want to quickly verify stability properties of the provided NCS setups; however, it is also possible to plot entire regions for which robust stability can be guaranteed by iteratively calling the function to assess robust stability. By providing a simple function to assess stability, the advanced user can write their own programs to plot robust stability regions for different parameters of interest. Furthermore, we recognize that the NCS configurations included in the toolbox might not suit the needs of a particular problem. This is why we made it possible for advanced users to incorporate their own models and use their own stability or performance conditions.

5.1 Plotting Robust Stability Regions

With the capability to assess stability of a single set of conditions, it is also possible to vary certain parameters to discover how robust the closed-loop system is and gain an understanding of where certain design tradeoffs lie. Assessing stability is achieved by executing the function

```
>> ncs.isNcsStable(ovrAprxType,quantVars,eu,M)
```

where `ovrAprxType` is either 'GNB', 'JNF' or 'CH' to indicate the type of overapproximation, `quantVars` is a structure containing data regarding quantization (if applicable), and `eu` and `M` are optional parameters used for the GNB algorithm. By iteratively using the above command, the NCS toolbox presented here can be used to produce tradeoff curves between networked properties (e.g. delays) and control properties (e.g. $\mathcal{L}_2$ gain) such as the ones given in previously published papers (e.g. in Donkers et al. (2011); Heemels et al. (2010a,b); Loon et al. (2012)). In Section 6, we will provide simple example code which demonstrates how to plot stability regions.

5.2 Automated Overapproximation for Custom Models

To encourage community development, the NCS toolbox allows users to incorporate custom models for robust stability to be assessed using the discrete-time NCS framework. For example, one might be interested in an NCS setup where a different type of hold mechanism is used (besides the zero order hold) or where a multi-hop network needs to be modeled or where specific inputs and outputs are directly wired to the controller and others are connected via a network. Creating a discrete-time model (of the form (1)) in these situations is not a very time-consuming task. However, the implementation of an overapproximation technique is generally a very time consuming task, which is why we have automated the overapproximation procedure for closed-loop models that can be expressed in the form (1). By providing, the matrices A_σ, B_σ, C_σ, D_σ, E_σ, J_σ, $\sigma \in \{1, ..., N\}$ and $\tilde{A}$ then the procedure to create a polytopic model (2) for $l \in \{1, ..., L\}$ and $\sigma \in \{1, ..., N\}$ is automated by using either the GNB, JNF or CH overapproximation technique. To create an overapproximation (2) of a model (1), the user just needs to execute the command

```
>> ovrAprx = genPolyOvrAprx(mdlVars,ovrAprxType,eu,M)
```

where `mdlVars` is a variable containing the matrices needed to express a custom NCS model in the form (1), `ovrAprxType` is either 'GNB', 'JNF' or 'CH', and `eu` and `M` are optional parameters used for the GNB algorithm. The output of this function is a variable called `ovrAprx`, which contains the matrices in (2) and other information about the overapproximation technique. The data in `ovrAprx` is suitable to use for any type of performance or stability analysis conditions based on LMIs.

6. EXAMPLE

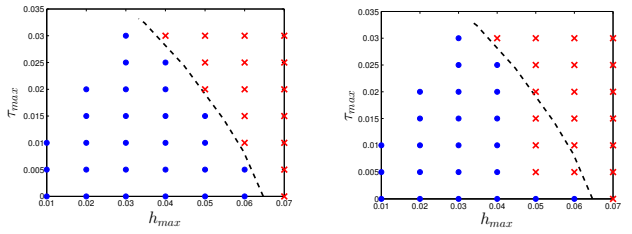
In this example, we will analyze the batch reactor system, studied in Donkers et al. (2011); Nešić and Teel (2004); Walsh et al. (2002) and others, to demonstrate how stability regions can be visualized, as described in Section 5.1. In particular, we will investigate the conservatism introduced by using the GNB technique. The GNB algorithm iteratively tightens the overapproximation by iteratively adding grid points to the overapproximated model at the location of the worst-case approximation error, until either the user-specified maximum number of grid points are reached or the user-specified desired tightness of the overapproximation is achieved, see Donkers et al. (2011). The resulting tightness of the overapproximation that is obtained, denoted ϵ, is a norm related to the overapproximation error. This is the fundamental idea behind the GNB algorithm, and this is all the user needs to know before they can start investigating the technique through numerical examples.

In order to investigate the conservatism, we will use the following simple MATLAB script:

```
for taumax=0:0.005:0.03
    for hmax=0.01:0.01:0.07
        if hmax >= taumax
            ncs.tau = [0 taumax];
            ncs.h = [1e-3 hmax];
            stable =
                ncs.isNcsStable('GNB',[],eu,M);
            if stable==1
                plot(ncs.h(2),ncs.tau(2),'b.');
            else
                plot(ncs.h(2),ncs.tau(2),'rx');
            end
        end
    end
end
```

In this script, `ncs` is an NCS object containing all the parameters specified in Donkers et al. (2011), which was created using the NCS Editor shown in Fig. 3. This script tests combinations of maximal transmission intervals and delays, (`hmax`,`taumax`), by modifying the parameters `ncs.h` and `ncs.tau` and then calling the function `isNcsStable`. If the NCS is determined stable, then we specify to plot a blue dot, otherwise if stability cannot be guaranteed, we specify to plot a red 'x'. The input parameters to `isNcsStable` are the `ncs` variable, the string `'GNB'` (to specify the overapproximation technique), `[]` to indicate that a quantizer is not used, and two additional parameters that need to be specified: the maximum number of grid points allowed, denoted `m`, and the desired approximation tightness, denoted `eu`. We will investigate the GNB technique by varying these parameters and observing the resulting conservatism introduced.

For the first case, denoted GNB1, we take `M`=50 and `eu`=2. For this example, 50 grid points is never reached, so the overapproximation tightness $\epsilon \leq 2$ is guaranteed.



(a) (b)

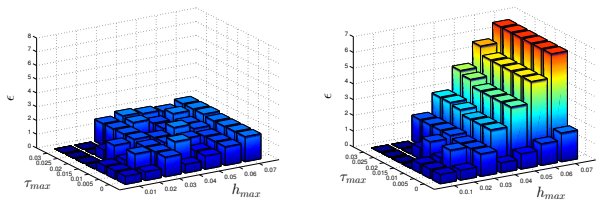
Fig. 5. Batch reactor stability plot. (a) is associated with GNB1 and (b) is associated with GNB2.

To compare with a second case, denoted GNB2, we set `M`=10 and keep `eu`=2. With these parameters, the overapproximation technique is limited to using a maximum of 10 grid points or less, depending on if $\epsilon \leq 2$ can be obtained within 10 grid points. This situation can be convenient if the user would prefer a reduction in computational time at the cost of a possible increase in conservatism. These results, along with the curve plotted in Donkers et al. (2011), are shown in Fig. 5.

From Fig. 5(a), we can see that using GNB1, we have successfully approximated the results in Donkers et al. (2011) since all the blue dots lie to the left of the dashed line and all the red x's lie to the right of the dashed line. However, limiting the maximum number of grid points to 10 introduces conservatism, as some points in Fig. 5(b) are not able to be proven stable by GNB2 but were proven stable by GNB1.

To further investigate the GNB technique and the resulting conservatism which was observed in Fig. 5(b), the toolbox enables the user to access the parameter ϵ for each point (`hmax`, `taumax`) by using the second output from the `isNcsStable` function. The second output is a structure which contains useful information about the overapproximation used for verifying stability. So, the algorithm above can be modified such that the `isNcsStable` outputs `[stable, ovrAprx]` instead of just `stable` and the variable `ovrAprx` contains a property `ovrAprx.MaxEps` which is the ϵ obtained from the GNB algorithm. Plotting this value for each (`hmax`, `taumax`) results in the figures given in Fig. 6. As can be seen from Fig. 6, GNB1 is able to guarantee $\epsilon \leq 2$ while the ϵ associated with GNB2 is higher due to the 10 grid point limit specified.

This example shows that by using the functions provided in the toolbox within simple iterative procedures, the control community can quickly begin experimenting with these techniques to gain insights which are not readily available from reading the literature.



(a) (b)

Fig. 6. Batch reactor overapproximation tightness, ϵ, plot. (a) shows ϵ for each (h_{max}, τ_{max}) associated with GNB1 and (b) shows ϵ for each (h_{max}, τ_{max}) associated with GNB2.

7. CONCLUSIONS AND FUTURE DEVELOPMENT

In this paper, we introduced a prototype of a toolbox that automates stability analysis of NCSs based on the theoretical contributions in Cloosterman et al. (2009); Donkers et al. (2011); Heemels et al. (2010a); Loon et al. (2012). The toolbox was designed such that the general control engineer can immediately start assessing robust stability of LTI systems interconnected via a network. Moreover, the software was coded in such a way that the NCS community can easily apply the overapproximation theory on any closed-loop model which is able to be written in the general form (1). Finally, the resulting overapproximation can be used in custom conditions for performance or stability analysis, further promoting community development.

This first NCS toolbox prototype focuses only on analysis; however, in future releases, we plan to include automation of the controller synthesis techniques given in Bauer et al. (2012); Cloosterman et al. (2010) and the stochastic results given in Donkers et al. (2012).

REFERENCES

Bauer, N., Donkers, M., van de Wouw, N., and Heemels, W. (2012). Decentralized static output-feedback control via networked communication. In *Proc. IEEE American Control Conf.*

Cloosterman, M., Hetel, L., van de Wouw, N., Heemels, W., Daafouz, J., and Nijmeijer, H. (2010). Controller synthesis for networked control systems. *Automatica*, 46(10), 1584–1594.

Cloosterman, M., van de Wouw, N., Heemels, W., and Nijmeijer, H. (2009). Stability of networked control systems with uncertain time-varying delays. *IEEE Trans. Autom. Control*, 54(7), 1575–1580.

Donkers, M., Heemels, W., Bernardini, D., Bemporad, A., and Shneer, V. (2012). Stability analysis of stochastic networked control systems. *Automatica*.

Donkers, M., Heemels, W., van de Wouw, N., and Hetel, L. (2011). Stability analysis of networked control systems using a switched linear systems approach. *Trans. Autom. Control*, 56(9), 2101–2115.

Gielen, R., Olaru, S., Lazar, M., Heemels, W., van de Wouw, N., and Niculescu, S.I. (2010). On polytopic inclusions as a modeling framework for systems with time-varying delays. *Automatica*, 46(3), 615 – 619.

Heemels, W., Teel, A., van de Wouw, N., and Nešić, D. (2010a). Networked control systems with communication constraints: Tradeoffs between transmission intervals, delays and performance. *IEEE Trans. Autom. Control*, 55(8), 1781–1796.

Heemels, W., van de Wouw, N., Gielen, R., Donkers, M., Hetel, L., Olaru, S., Lazar, M., Daafouz, J., and Niculescu, S. (2010b). Comparison of overapproximation methods for stability analysis of networked control systems. In *HSCC 2010: Proc. 13th ACM Int. Conf. on Hybrid systems: Computation and control*, 181–190.

Löfberg, J. (2004). YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proc. CACSD Conf.*

Loon, S., Donkers, M., Bauer, N., van de Wouw, N., and Heemels, W. (2012). Stability analysis of networked and quantized control systems: A switched linear systems approach. *Trans. on Industrial Informatics*.

Nešić, D. and Teel, A. (2004). Input-output stability properties of networked control systems. *IEEE Trans. Autom. Control*, 49(10), 1650–1667.

Sturm, J. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12, 625–653.

Walsh, G., Ye, H., and Bushnell, L. (2002). Stability analysis of networked control systems. *IEEE Trans. Control Systems Technology*, 10(3), 438–446.