NCS Toolbox

Eindhoven University of Technology

May 11, 2011

Contents

1 Installation

Unzip all the contents of NCStoolbox.zip into a folder on your computer. Add this folder and all subfolders to the MATLAB path using 'File > Set Path...'

2 Required MATLAB (2007a or later) Toolboxes

- Yalmip LMI parser (latest version)
- SeDuMi Solver (latest version)
- Linear Systems *Toolkit*
- Symbolic Toolbox
- Robust Control Toolbox
- Optimization Toolbox

3 Overview

This toolbox models, analyzes and synthesizes control of a linear time invariant plant over a network. There are two modeling frameworks taken in this toolbox, the discretized NCS model and the hybrid NCS model. At this developmental stage in the toolbox, each modeling approach differs in the network effects included in the models.

In the discretized NCS framework, stability analysis and synthesis of state feedback controllers is considered with the network including the following effects:

- varying transmission intervals $h_k \in [\underline{h}, \overline{h}]$
- varying delays $\tau_k \in [\underline{\tau}, \overline{\tau}]$ (small delay $\tau_k \leq h_k$ for all $k \in \mathbb{N}$)
- varying dropouts $\delta_k \in \{0, 1, ..., \bar{\delta}\}$.
- communication constraints $\sigma_k \in \{1, ..., N\}$

Due to the fact that the parameters h_k and τ_k vary in an infinite set, the discretized NCS approach requires to transform the system into a polytopic set which captures the dynamics of the original discretized NCS. Three methods of transformation into a polytopic set are implemented in this toolbox: the Jordan Normal Form overapproximation, the Cayley-Hamilton overapproximation and the Gridding and Norm Bounding overapproximation. Stability and synthesis can be preformed on the polytopic system representation using LMI techniques.

In the hybrid framework, stability analysis of dynamic controllers is considered with the network including the following effects:

- varying transmission intervals $h_k \in [\underline{h}, \overline{h}]$
- varying delays $\tau_k \in [0, \overline{\tau}]$ (small delay $\tau_k \leq h_k$ for all $k \in \mathbb{N}$)
- communication constraints: only a subset of input and a subset of output signals are transmitted simultaneously.

The hybrid approach has the advantage that a polytopic overapproximation is not needed and stability can be performed directly on the original model of the networked control system. This Lyapunov function has a special form suitable for the NCS dynamics.

4 NCS Editor

Before using any tools available in the NCS toolbox, an NCS object must be created. An NCS object contains the following properties:

Category	Parameter	Explanation
Control Data	`A,B,C`	`A,B,C` continuous-time plant matrices
	`ctrlType`	`'Static Feedback'`
		`'D-LTI observer-based feedback'`
		`'D-LTI dynamic feedback'`
		`'C-LTI dynamic feedback'`
	`Ac,Bc,Cc,Dc`	`A,B,C,D` controller matrices
	`K`	`K` controller matrix
	`L`	`L` observer matrix
Network Data	`h`	`[hmin,hmax]` bounds on the transmission time
	`tau`	`[taumin,taumax]` bounds on the delay
	`delta`	`delta` $\in \mathbb{N}$ bound on the number of subsequent dropouts
	`gammaU`	matrices which define nodes
	`gammaY`	matrices which define nodes
	`prot`	`'RR','TOD'` protocol

The easiest way to create a NCS object is using the NCS Editor shown in Figure 1.

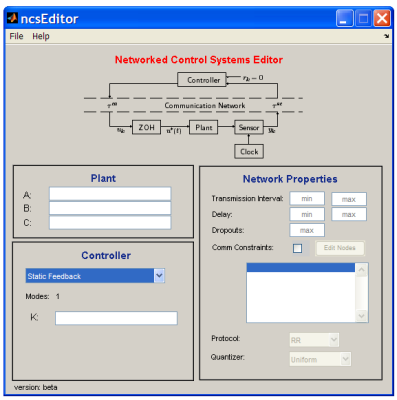

Figure 1: NCS Editor

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 by clicking 'File > Export'. If a modification to a NCS object needs to be made, by clicking on 'File > Import', NCS objects can be loaded and modified.

4.0.1 Plant

First we need to define the system which is to be controlled. In this toolbox the plant is a linear time-invariant system expressed as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) \end{aligned}$$

the matrices A, B and C are the first parameters input to define a NCS object. In these input boxes, either matrices can be directly input, variables defined in the workspace can be input or a mathematical expression can also be input.

4.0.2 Controller

Next, the controller needs to be specified. The type can be defined by clicking on a choice from the drop down dialog box. There are three possible controller types currently available in this toolbox. The first is 'Static Feedback' where the feedback law is $u(t) = Ky(t)$. The second controller type is 'C-LTI Dynamic Feedback' control law which is given by

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

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

$$\begin{aligned} x^c_{k+1} &= A^c x^c_k + B^c u_k, \\ y &= C^c x^c_k + D^c u_k. \end{aligned}$$

The matrices K or A_c, B_c, C_c and D_c are input into the corresponding text boxes. In these input boxes, either matrices can be directly input, variables defined in the workspace can be input or a mathematical expression can also be input.

4.0.3 Network

Lastly, the network effects need to be defined. There are roughly five recognized Networked Control System (NCS) side-effects: time-varying transmission intervals, time-varying delays, packet dropouts, quantization and communication constraints (the latter meaning that not all information can be sent over the network at once). In this toolbox, quantization is not yet implemented.

The bounds on the time-varying transmission intervals and time-varying delays are input to the corresponding input box. If the transmission intervals or delays are considered constant then the 'min' can be set equal to the 'max.'

If delays are not considered, then both 'min' and 'max' can be set equal to zero. Next, a bound on successive packet dropouts needs to be input. If dropouts are not considered, this can be set to zero.

Next there is a check box to indicate whether or not communication are present. Once this box is checked, it is possible to click the button 'Define Nodes.' Another GUI will, shown in Fig. 2, appear where the nodes can be defined by 'adding' inputs and outputs to each node.

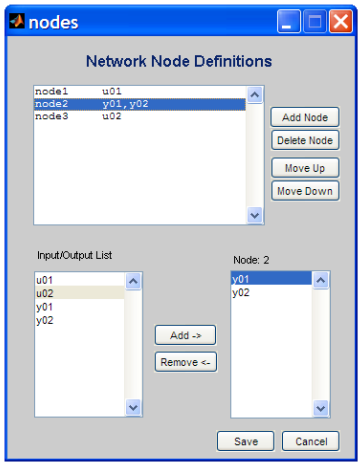

Figure 2: GUI to define nodes.

Nodes can be added or removed and moved up and down to change the ordering. After the nodes have been defined, clicking 'Save' will save the node configuration. Clicking 'Cancel' will keep the old configuration previously saved. Once saved the node GUI will closed and in the ncsEditor, the protocol needs to be chosen as 'RR' for Round Robin or 'TOD' for Try-Once-Discard.

5 Discretized NCS Approach

5.1 How to use the Discretized NCS Functions

Currently, there are two main functions which use the Jordan Normal Form (JNF) and Cayley-Hamilton (CH) overapproximation techniques, namely, robust stability verification and state feedback synthesis, which can be performed on an NCS with network effects mentioned in the Overview Section. First, stability can be verified with the following command

```
stable = ncsObj.isNcsStable(ovraprx,drpmdl,lyap,gamma)
```

where the decay rate of the system is $(1 - \texttt{gamma})$. Second, a stabilizing state feedback gain K with decay rate $(1 - \texttt{gamma})$ can be synthesized with the following command

```
[stable,K] = ncsObj.stabilizeNcs(ovraprx,drpmdl,lyap,gamma)
```

The parameters for these two functions are described as follows:

Category	**Parameter**	**Explanation**
Stability Options	`ovraprx`	`'JNF'` for Jordan Normal Form overapproximation
		`'CH'` for Caley-Hamilton overapproximation
	`drpmdl`	`'lngtrans'` for 'prolonged transmission-interval' dropout model
		`'explicit'` for explicit dropout model
	`lyap`	`'common'` for common quadratic Lyapunov function
		`'pardep'` for parameter dependent Lyapunov function
	`gamma`	$0 \leq$ `gamma` < 1 measure of the Lyapunov function decay
Output	`stable`	`0` = stability not guaranteed, `1` = stable

To see commented code in an example run the command `'doc example_dncs'` in the MATLAB command window.

If verification of robust stability via the gridding and norm bounding (GNB) overapproximation technique is desired, robust stability analysis can be performed with the following command

```
stable = ncsObj.isNcsStable(ovraprx,gridpntM,epsilon_u,uWired)
```

The parameters are described as follows:

Category	Parameter	Explanation
Stability Options	`ovraprx`	`'GNB'` for Gridding and Norm Bounding overapproximation
	`gridpntM`	`gridpntM` $\in \mathbb{N}$ maximum number of gridpoints tolerated
	`epsilon_u`	`epsilon_u` $\in \mathbb{R}$ user defined tightness of overapproximation
	`uWired`	`1` if u is continuously known by controller
		`0` if u is sampled and sent over network
Output	`stable`	`0` = stability not guaranteed, `1` = stable

5.2 JNF & CH Implementation

We consider controlling the following plant

$$\dot{x}(t) = Ax(t) + Bu(t),$$

by using the static control law $u(t) = Kx(t)$ subject to

- time-varying transmission intervals $h_k \in [\underline{h}, \overline{h}]$
- time-varying delays (small) $\tau_k \in [\underline{\tau}, \overline{\tau}]$
- varying dropouts $\delta_k \leq \bar{\delta}$

To simplify the explanation of how we modeled the NCS in the discretized NCS framework, we consider the NCS that only incorporates time-varying sampling intervals and time-varying communication delays. The case that dropouts are present can be handled analogously in the toolbox and is discussed in Section 5.2.5. The general NCS model is given by

$$\xi_{k+1} = \underbrace{\begin{bmatrix} e^{Ah_k} & \int_{h_k-\tau_k}^{h_k} e^{As} ds B \\ 0 & 0 \end{bmatrix}}_{\tilde{A}} \xi_k + \underbrace{\begin{bmatrix} \int_0^{h_k-\tau_k} e^{As} ds B \\ I \end{bmatrix}}_{\tilde{B}} u_k \tag{1}$$

where the lifted state vector $\xi_k = [x_k \; u_{k-1}]$.

5.2.1 Polytopic Overapproximation

The first step in the overapproximation process is to separate all Jordan blocks in **JJ = JordanBlocks(J)** (where JJ is a cell that contains all separated Jordan blocks) by checking for ones on the upper diagonal (in case of a real eigenvalue) or non-zero elements on the lower diagonal and ones on the second upper diagonal (in case of a complex eigenvalue pair). Now that all Jordan blocks are separated, the exponential terms in (1) can be determined symbolically by Matlab, using the expressions

$$e^{Ah_k} = Q \; \mathrm{diag}(e^{J_1 h_k}, e^{J_2 h_k}, \ldots, e^{J_p h_k}) Q^{-1}. \tag{2}$$

With this, the matrix exponential $e^{J_i h_k}$ can be written as the sum of varying functions α_j and constant matrices S_j for all $i = 1, 2, \ldots, p$, i.e.

$$e^{J_i h_k} = \sum_{j=1}^{\nu} \bar{\alpha}_{i,j}(h_k) \bar{S}_{i,j}, \tag{3}$$

with $\nu \leq n$. The values of $\bar{\alpha}_{i,j}(h_k)$ and $\bar{S}_{i,j}$ for the Jordan Normal Form are determined in the file **[S alfa] = exponent(Ji)**. Then the expression

$$e^{Ah_k} = \sum_{j=1}^{\nu} \alpha_{1,j}(h_k) S_{1,j}, \tag{4}$$

can be formed where $\alpha_{1,j}$ are all the $\bar{\alpha}_{i,j}$ terms and $S_{1,j}$ are matrices consisting of $\bar{S}_{i,j}$ matrices which are placed corresponding to the Jordan Block and pre- and post-multiplied by Q and Q^{-1} respectively. This concludes how the Jordan Normal Form is computed.

If the Cayley-Hamilton overapproximation is desired, then the coefficients $\alpha_j(h_k)$ and matrices S_j in (4) are changed to $f_j(h_k)$ and A^{j-1}, respectively, where **[Ai,f] = Cayley(A,S,alfa)** determines the new constants f_j where

$$f_j(h_k) = \sum_{l=1}^{\nu} \beta_{j,l} \alpha_{1,l}(h_k), \quad \sum_{l=1}^{\nu} \beta_{j,l} A^{l-1} = S_{1,j}, \tag{5}$$

for all $l = 1, 2, \ldots, \nu$.

To compute the integral terms of the NCS model, we simply use the exponential terms computed above. If A is invertible we have that

$$\begin{aligned}
\int_a^b e^{As} ds &= A^{-1} e^{Ab} - A^{-1} e^{Aa}, \\
&= A^{-1} \sum_i \alpha_i(b) S_i - A^{-1} \sum_i \alpha_i(a) S_i \\
&= \sum_i \left[\alpha_i(b) A^{-1} S_i \right] + \sum_i \left[\alpha_i(a) [-A^{-1} S_i] \right].
\end{aligned}$$

If A is not invertible we use

$$\begin{aligned}
\int_a^b e^{As} ds &= \int_a^b \sum_i \alpha_i(s) S_i ds, \\
&= \sum_i \int_a^b \alpha_i(s) ds S_i = \sum_i (\Lambda_i(b) - \Lambda_i(a)) S_i \\
&= \sum_i \left[\Lambda_i(b) S_i \right] + \sum_i \left[\Lambda_i(a) [-S_i] \right]
\end{aligned}$$

where

$$\int_a^b \alpha_i(s) ds = \Lambda_i(b) - \Lambda_i(a)$$

is computed symbolically. Hence, we can express

$$\int_{h_k}^{h_k-\tau_k} e^{As} ds = \sum_{l=1}^{2\nu} \alpha_{2,l}(h_k, \tau_k) S_{2,l}, \tag{6}$$

$$\int_{0}^{h_k-\tau_k} e^{As} ds = \sum_{l=1}^{2\nu} \alpha_{3,l}(h_k, \tau_k) S_{3,l}. \tag{7}$$

Now that each term of the NCS model (1) is written as the sum of varying functions $\alpha_{i,j}$ and constant matrices $S_{i,j}$, it is possible to write the model by substituting (3), (6) and (7) into (1), resulting in

$$\xi_{k+1} = \left(F_0 + \sum_{i=1}^{2\nu} \alpha_i(h_k, \tau_k) F_i\right)\xi_k + \left(G_0 + \sum_{i=1}^{2\nu} \alpha_i(h_k, \tau_k) G_i\right)u_k, \tag{8}$$

with $\{\alpha_1, \ldots, \alpha_{2\nu}\}$ is the total set of varying functions that contains all different functions from $\alpha_{1,j}$, $\alpha_{2,l}$ and $\alpha_{3,j}$, where the objective is that each distinct varying function only appears once in $\{\alpha_1, \ldots, \alpha_{2\nu}\}$. This is done in the routine **[alfa_ht,F_ht,G_ht] = check_alpha(alfa11,alfa22,alfa33,S11,S22,S33,Bc)** that puts all functions from $\alpha_{1,j}$ (*alfa11*), $\alpha_{2,l}$ (*alfa22*) and $\alpha_{3,j}$ (*alfa33*) in one set α_i (*alfa_ht*) and then checks each function on multiplicity by using a symbolic subtraction.

Since the uncertainties h_k and τ_k can take infinitely many values, the model in (8) needs to be embedded in a convex hull having a finite number of vertices, i.e.

$$\mathcal{H}_{\mathcal{S}} = \left\{ \underbrace{\left(F_0 + \sum_{i=1}^{2\nu} \alpha_i F_i\right)}_{H_{F,j}}, \underbrace{\left(G_0 + \sum_{i=1}^{2\nu} \alpha_i G_i\right)}_{H_{G,j}} \mid \alpha_i \in \{\underline{\alpha}_i, \overline{\alpha}_i\}, \;\; i = 1, 2, \ldots, 2\nu \right\}, \tag{9}$$

with F_0, F_i, G_0 and G_i are constant matrices and α_i are time-varying functions with minimal values $\underline{\alpha}_i$ and maximal values $\overline{\alpha}_i$ for $i = 1, 2, \ldots, 2\nu$.

To obtain this set of matrices $\mathcal{H}_{\mathcal{S}}$, all minima and maxima of α_i, $\underline{\alpha}_i$ and $\overline{\alpha}_i$ respectively, need to be determined by the Matlab toolbox. This is done in

[F,F0,G,G0,alfaMinMaxCombos, alfaMinMax] =
getAlfaMinMax(alfa,Ah,Bh,h,tau).

In this file, all combinations of the minimal and maximal numerical values of all functions α_i are captured in *alfaMinMaxCombos* with the corresponding constant matrices in the cells *F* and *G*. The matrices *F0* and *G0* correspond to constant *alfa* terms and thus do not need to be overapproximated. The output *alfaMinMax* is a matrix of the min and max for each *alfa*. This step concludes the overapproximation process: all vertices $\mathcal{H}_{F,j}$ and $\mathcal{H}_{G,j}$ of the convex embedding are contained in the finite set $\mathcal{H}_{\mathcal{S}}$.

5.2.2 Controller Synthesis

In case that no controller matrix K is available, the Matlab toolbox has a state feedback synthesis function such that a state feedback of the form $u_k = -Kx_k$ can be obtained. One can choose for a controller K that is synthesised using a common quadratic Lyapunov function (**K = CQLF(A,B,gamma)**, where A is a cell containing matrices $H_{F,j}$, B is a cell containing matrices $H_{G,j}$ and *gamma* is the decay rate of the Lyapunov function) or a parameter-dependent Lyapunov function (**K = PDLF(A,B,gamma)**). The controller synthesis that is based on a parameter-dependent LF is implemented in the Matlab toolbox in **PDLF.m**, i.e.

$$\begin{bmatrix} X_j + X_j^T - Y_j & X_j^T H_{F,j}^T - \begin{bmatrix} \bar{Z} & 0 \end{bmatrix}^T H_{G,j}^T \\ H_{F,j}X_j - H_{G,j}\begin{bmatrix} \bar{Z} & 0 \end{bmatrix} & (1-\gamma)Y_l \end{bmatrix} > 0, \tag{10}$$

where $H_{F,j}$ and $H_{G,j}$ are the sets of vertices that are contained in $\mathcal{H}_{\mathcal{S}}$, and the state feedback $K = \bar{Z}X_1^{-1}$. If a controller synthesis is desired that is based on a common quadratic LF, we set $Y_j = Y_l = Y$. This is implemented in **CQLF.m**.

5.2.3 Apply Available Controller

Using the control law $u_k = -Kx_k$, the vertices of the closed-loop overapproximated system is given by

$$\mathcal{H}_{CL,j} = H_{F,j} - [K \;\; 0]H_{G,j}, \tag{11}$$

for all $j = 1, 2, ..., 2^{2\nu}$

5.2.4 Stability Analysis in the Matlab Toolbox

With the set of vertices (11), the stability of the overapproximated closed-loop NCS model can be analysed. Because dropouts are not considered, no dropout-dependent LMIs can be constructed and the following set of LMIs is obtained

$$\begin{aligned} P &> \epsilon I, \\ \mathcal{H}_{CL,j}^T P \mathcal{H}_{CL,j} - (1-\gamma)P &< 0, \end{aligned} \tag{12}$$

where $0 < \epsilon \leq 1$ is a small scalar that avoids numerical problems in Matlab, $0 \leq \gamma < 1$ ensures the Lyapunov function to decrease with rate $1-\gamma$ and $\mathcal{H}_{CL,j}$ is the set of vertices (11) for all $j = 1, 2, \ldots, 2^{2\nu}$. Note that, in this thesis we only considered dropout-dependent LFs and we did not consider dependencies of delay and sampling interval. The LMIs from (12) are implemented in the Matlab toolbox in the file **[P, LMIs] = LMI_common(Dt,epsilon,gamma)** (where P is the Lyapunov matrix, the cell *LMIs* contains all solved LMIs and the cell *Dt* contains all vertices of the overapproximation), solved by the SeDuMi (Yalmip)

LMI solver and verified if they satisfy the matrix inequality constraints. If they do, the overapproximated closed-loop NCS model (and consequently the original NCS) is stable; if they do not, no conclusions can be drawn about the stability of the NCS.

5.2.5 Stability Analysis With Dropouts

So far, we discussed NCSs that do not incorporate packet dropouts in this chapter. As a consequence, a general model (1) is used to describe NCSs and this model can only be analysed on stability using common quadratic Lyapunov functions (discarding the dependence on the overapproximation vertices).

From here, we consider an NCS model that incorporates dropouts as well as time-varying delays and time-varying sampling intervals. Two modeling approaches are implemented, namely modeling dropouts as a prolongation of the sampling intervals and the explicit dropout modeling using hybrid automata. With this extension of the NCS model with dropouts, the only difference with Section 5.3 (no dropouts) is the implementation of the stability analysis in the Matlab toolbox. This is due to the fact that in case of dropouts, the dimension of the set of vertices is depending on the maximum number of subsequent dropouts $\bar{\delta}$, i.e.

$$\mathcal{H}_{CL,i,g} = H_{F,i,g} - \begin{bmatrix} K & 0 \end{bmatrix} H_{G,i,g}, \tag{13}$$

for the dropout variable $i = 0, 1, \ldots, \bar{\delta}$ and the number of vertices $g = 1, 2, \ldots, 2^{2\nu}$ (in case of modeling dropouts as prolongation of the sampling intervals) or $i = 0, 1$, with $g_0 = 0, 1, \ldots, 2^{2\nu}$ for $i = 0$ and $g_1 = 0, 1, \ldots, 2^{\nu}$ for $i = 1$ (in case of modeling dropouts explicitly using hybrid automata). Using this, dropout-dependent LFs can be constructed to analyze the stability of the NCS model.

If we use the explicit dropout modeling approach in combination with a PDLF based stability analysis, the following LMIs are used:

$$\begin{aligned} P_l &> \epsilon I, \\ \mathcal{H}_{CL,0,g_0}^T P_0 \mathcal{H}_{CL,0,g_0} - (1-\gamma) P_l &< 0, \\ \mathcal{H}_{CL,1,g_1}^T P_{j+1} \mathcal{H}_{CL,1,g_1} - (1-\gamma) P_j &< 0, \end{aligned} \tag{14}$$

for $j = 0, 1, \ldots, \bar{\delta} - 1$ and $l = 0, 1, \ldots, \bar{\delta}$, where the input $deltaBar = \bar{\delta}$ and the input Dt is a cell containing all vertices $\mathcal{H}_{CL,i,g}$ of the overapproximated NCS model, which are implemented in **[P, LMIs]=LMI(Dt,deltaBar,epsilon)**.

If we use the dropout as prolongation of the sampling intervals modeling approach in combination with a PDLF based stability analysis, the following LMIs are used:

$$\begin{aligned} P_i &> \epsilon I, \\ \mathcal{H}_{CL,i,g}^T P_j \mathcal{H}_{CL,i,g} - (1-\gamma) P_i &< 0, \end{aligned} \tag{15}$$

for all $i, j \in \{0, 1, \ldots, \bar{\delta}\}$ and $g = 1, 2, \ldots, 2^{2\nu}$, which are implemented in **[P,LMIs] = LMI_ dependent(Dt,deltaBar,epsilon)**.

If all conditions in (14) or (15) are satisfied, the analyzed NCS is said to be asymptotically stable. If not, no conclusions about the stability of the NCS can be drawn. More details can be found in [3], [6], [4].

5.3 GNB Implementation

We consider controlling the following plant

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) \end{aligned}$$

by using a continuous-time dynamic control law

$$\begin{aligned} \dot{x}^c(t) &= A^c x^c(t) + B^c u(t), \\ y(t) &= C^c x^c(t) + D^c u(t) \end{aligned}$$

or a discrete-time dynamic control law

$$\begin{aligned} x^c_{k+1} &= A^c x^c_k + B^c u_k, \\ y &= C^c x^c_k + D^c u_k \end{aligned}$$

subject to

- time-varying transmission intervals $h_k \in [\underline{h}, \overline{h}]$,
- time-varying delays (small) $\tau \in [\underline{\tau}, \overline{\tau}]$,
- communication constraints $\sigma_k \in \{1, ..., N\}$
- varying dropouts $\delta_k \leq \bar{\delta}$.

The closed-loop NCS can be expressed as a switched discrete-time system

$$\bar{x}_{k+1} = \underbrace{\begin{bmatrix} A_{h_k} + E_{h_k} BDC & E_{h_k} BD - E_{h_k - \tau_k} B\Gamma_{\sigma_k} \\ C(I - A_{h_k} - E_{h_k} BDC) & I - D^{-1}\Gamma_{\sigma_k} + C(E_{h_k - \tau_k} B\Gamma_{\sigma_k} - E_{h_k} BD) \end{bmatrix}}_{=: \bar{A}_{\sigma_k, h_k, \tau_k}} \bar{x}_k, \qquad (16)$$

where the matrices are defined in [2]. This model will be overapproximated by the GNB overapproximation method and stability will be assessed via LMI conditions.

5.3.1 Overapproximation

The closed-loop model (16) will be expressed as a polytopic system with norm-bounded additive uncertainty, *i.e.*,

$$\bar{x}_{k+1} = \Big(\sum_{l=1}^{L} \alpha_k^l \hat{A}_{\sigma_k,l} + \hat{B}_l \Delta_k \hat{C}_{\sigma_k} \Big) \bar{x}_k. \tag{17}$$

where $\hat{A}_{\sigma_k,l}$ is $\tilde{A}_{\sigma_k,h_k,\tau_k}$ evaluated at M points and the l-th gridpoint is (h_l, τ_l) and $\hat{B}\Delta_k\hat{C}_{\sigma_k}$ is defined such that

$$\Big\{ \tilde{A}_{j,h,\tau} \mid h \in [\underline{h}, \overline{h}], \tau \in [\underline{\tau}, \overline{\tau}] \Big\} \subseteq \Big\{ \sum_{l=1}^{M} \alpha^l \left(\hat{A}_{j,l} + \hat{B}_l \Delta \hat{C}_j \right) | \, \alpha \in \mathbf{\Omega}, \Delta \in \mathbf{\Delta} \Big\}. \tag{18}$$

The grid points are chosen via an iterative algorithm which minimizes the error between (16) and (17). The function **CreateGridPoints.m** creates an initial set of gridpoints. The function **GNB.m** calculates the norm-bounded uncertainty resulting from a set of gridpoints. More details concerning the overapproximation can be found in [2].

5.3.2 Stability Analysis

Stability is guaranteed if there exists symmetric matrices P_ℓ and $R_{\ell,j}$ for $\ell = 1, ..., N$ and $j = 1, ..., M$ such that

$$\begin{bmatrix} \bar{A}_{\sigma_\ell,j}^\top P_{\ell+1} \bar{A}_{\sigma_\ell,j} - P_\ell + \bar{C}_{\sigma_\ell}^\top R_{\ell,j} \bar{C}_{\sigma_\ell} & \bar{A}_{\sigma_\ell,j}^\top P_{\ell+1} \bar{B}_j \\ \bar{B}_j^\top P_{\ell+1} \bar{A}_{\sigma_\ell,j} & \bar{B}_j^\top P_{\ell+1} \bar{B}_j - R_{\ell,j} \end{bmatrix} \prec 0. \tag{19}$$

for $\ell = 1, ..., N$ and $j = 1, ..., M$. Creating the overapproximated system and performing stability analysis is done in the main function **dncs_GNB.m**. More details concerning stability analysis can be found in [2].

6 Hybrid NCS Approach

6.1 How to use the Hybrid NCS Function

There is one main function which provide the data to plot tradeoff curves between varying delays and varying transmission intervals of the NCS including the given the effects mentioned in the Overview Section. This function is called with the following command

```
[Hmati,Tmad] = ncsObj.findNcsStablilityTradeoff(Sy,Su)
```

where

`Sy`	`[y1 y2 .. yN]`,	`yi` is `1` if the i-th output is sampled and sent over the network,
		`yi` is `0` if the i-th output is continuously known by controller
`Su`	`[u1 u2 .. uM]`,	`ui` is `1` if the i-th input is sampled and sent over the network,
		`ui` is `0` if the i-th input is continuously known by controller
	`[1 1 .. 1]`,	required for state feedback

Unlike the DLPV approach, this function outputs the data needed to plot the stability region for $\tau_k \in [0 \;\; \tau_{max}]$ and $h_k \in [h_{min} \;\; h_{max}]$ where $\tau_k \leq h_k$ for all k. A stability region comparison between the RR protocol and TOD protocol can be easily done with this function.

To see commented code in an example run '`doc example_hncs`' in the MATLAB command window.

6.2 Hybrid Implementation

We consider controlling the following plant

$$\begin{aligned} \dot{x}_p &= A_p x_p + B_p \hat{u}, \\ y &= C_p x_p \end{aligned}$$

by using the linear dynamic controller given by

$$\begin{aligned} \dot{x}_c &= A_c x_c + B_c \hat{y}, \\ u &= C_c x_c + D_c \hat{y} \end{aligned}$$

subject to

- time-varying transmission intervals $h_k \in [\underline{h}, \overline{h}]$,
- time-varying delays (small) $\tau_k \in [\underline{\tau}, \overline{\tau}]$,

- communication constraints $\sigma_k \in \{1, ..., N\}$

We can model this system as a hybrid system in the following form

$$\begin{aligned} \dot{\xi} &= F(\xi), \quad \xi \in C \\ \xi^+ &= G(\xi), \quad \xi \in D \end{aligned}$$

where the continuous system experiences abrupt jumps when a transmission event occurs `[hmin,hmax]` and an update event occurs `[tmin,tmax]`.

Stability of this system can be proven directly on the hybrid model. We consider using a Lyapunov function candidate of the form

$$U(\xi) = V(x) + \phi(\tau)W(\xi)$$

where $V(x)$ is a Lyapunov function of the closed loop system without any network effects, ϕ is a differentiable function of only the state τ which determines when the system experiences jumps and $W(\xi)$ is a Lyapunov function for the protocol being used. Since we have a hybrid model, this Lyapunov function needs to have a negative derivative during flows $\xi \in C$ and decreasing when the system experiences a jump $\xi \in D$, i.e.

$$\begin{aligned} \tfrac{d}{d\xi}U(\xi)F(\xi) < 0 \qquad & \xi \in C \\ U(G(\xi)) - U(\xi) < 0 \qquad & \xi \in D \end{aligned}$$

Then by varying the initial conditions of the function ϕ, the tradeoff plot is constructed. More details can be found in [1].

References

[1] W.P.M.H. Heemels, A.R. Teel, N. van de Wouw, and D. Nešić. Networked Control Systems With Communication Constraints: Tradeoffs Between Transmission Intervals, Delays and Performance. *IEEE Transactions on Automatic Control*, 55(8):1781-1796, August 2010.

[2] M.C.F. Donkers, W.P.M.H. Heemels, N. van de Wouw, and L. Hetel. Stability Analysis of Networked Control Systems using a Switched Linear Systems Approach. *IEEE Transactions on Automatic Control*, to appear.

[3] M.B.G. Cloosterman, N. van de Wouw, W.P.M.H. Heemels, and H. Nijmeijer. Stability of Networked Control Systems With Uncertain Time-Varying Delays. *IEEE Transactions on Automatic Control*, 57(7):1575-1580, July 2009.

[4] M.B.G. Cloosterman, L. Hetel, N. van de Wouw, W.P.M.H. Heemels, J. Daafouz and H. Nijmeijer. Controller synthesis for networked control systems. *Automatica*, 46(10):1584-1594, 2010.

[5] W.P.M.H. Heemels, N. van de Wouw, R.H. Gielen, M.C.F. Donkers, L. Hetel, S. Olaru, M. Lazar, J. Daafouz, and S. Niculescu. Comparison of overapproximation methods for stability analysis of networked control systems. *HSCC '10: Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 181-190, New York, NY, USA, 2010. ACM.

[6] J.J.C. van Schendel, M.C.F. Donkers, W.P.M.H. Heemels, and N. van de Wouw. On dropout modelling for stability analysis of networked control systems. *Proceedings of the American Control Conference*, pages 555-561, July 2010.