Simultaneous compensation of input delay and state/input quantization for linear systems via switched predictor feedback

Description

This document provides an overview of **QuantizerP**, a project developed to illustrate the concepts presented in the paper *Simultaneous Compensation of Input Delay and State/Input Quantization for Linear Systems via Switched Predictor Feedback*. **QuantizerP** simulates a linear time-delay system represented by the following differential equation:

$$\dot{X}(t) = AX(t) + BU(t-D), \tag{1}$$

This system can alternatively be represented by an ODE-PDE cascade system over $x \in [0, D]$:

$$\begin{aligned} \dot{X}(t) &= AX(t) + Bu(0,t), && (2)\\ u_t(x,t) &= u_x(x,t), && (3)\\ u(D,t) &= U(t), && (4) \end{aligned}$$

where $D > 0$ is the constant input delay, $t \geq 0$ is the time variable, $X \in \mathbb{R}^n$ represents the ODE state, u denotes the transport PDE state with initial conditions $u(x,0) = u_0(x)$, and U is the scalar control input.

The hybrid predictor-feedback law is a quantized version of the predictor-feedback controller given by:

$$U(t) = \begin{cases} 0, & 0 \leq t \leq t_1^*, \\ KP_{\mu(t)}\left(X(t), u(\cdot,t)\right), & t > t_1^*, \end{cases} \tag{5}$$

where

$$P_\mu(X,u) = \mathrm{e}^{AD} q_{1\mu}(X) + \int_0^D \mathrm{e}^{A(D-y)} B q_{2\mu}(u(y))\,dy, \tag{6}$$

and the dynamic quantizer depends on the piecewise constant signal μ defined as:

$$\mu(t) = \begin{cases} \overline{M}_1 \mathrm{e}^{2|A|(j+1)\tau}\mu_0, & (j-1)\tau \leq t \leq j\tau + \bar{\tau}\delta_j, \\ 1 \leq j \leq \left\lfloor \frac{t_1^*}{\tau} \right\rfloor, & \\ \mu(t_1^*), & t \in (t_1^*, t_1^* + T], \\ \Omega\mu\left(t_1^* + (i-1)T\right), & t \in \left(t_1^* + (i-1)T, t_1^* + iT\right], \quad i = 2,3,\ldots \end{cases} \tag{7}$$

Here, t_1^* is the first time instant satisfying:

$$\left|\mu(t_1^*) q_1\left(\frac{X(t_1^*)}{\mu(t_1^*)}\right)\right| + \left\|\mu(t_1^*) q_2\left(\frac{u(\cdot,t_1^*)}{\mu(t_1^*)}\right)\right\|_\infty \leq (M\overline{M} - \Delta)\mu(t_1^*), \tag{8}$$

with constant parameters defined as:

$$
\begin{aligned}
M_1 &= |K|\mathrm{e}^{|A|D}\max\{1, D|B|\}+1, && (9)\\
M_2 &= \frac{1}{|K|\mathrm{e}^{|A+BK|D}\max\{1, D|B|\}+1}, && (10)\\
M_3 &= |K|\mathrm{e}^{|A|D}(1+|B|D), && (11)\\
\overline{M} &= \frac{M_2}{M_1(1+M_0)}, && (12)\\
\overline{M}_1 &= 1+D|B|, && (13)\\
\Omega &= \frac{(1+\lambda)(1+M_0)^2\Delta M_3}{M_2 M}, && (14)\\
T &= -\frac{\ln\left(\frac{\Omega}{1+M_0}\right)}{\delta}. && (15)
\end{aligned}
$$

The error Δ and the quantizer range M must satisfy:

$$
\frac{\Delta}{M} < \frac{M_2}{(1+M_0)\max\{M_3(1+\lambda)(1+M_0), 2M_1\}}. \tag{16}
$$

The parameters δ, λ, and M_0 are defined as follows:
- The constant $\delta \in (0, \min\{\sigma, \nu\})$, for some $\nu, \sigma > 0$, satisfies:

$$
\left|e^{(A+BK)t}\right| \leq M_\sigma e^{-\sigma t}, \tag{17}
$$

for some $M_\sigma > 1$.
- λ is selected large enough to satisfy the small-gain condition:

$$
\frac{e^D}{1+\lambda}\left(\frac{M_\sigma}{\sigma}|B|+1\right) < 1. \tag{18}
$$

- M_0 is defined by:

$$
\begin{aligned}
M_0 = \max&\left\{(1-\phi)^{-1}(1-\varphi_1)^{-1}e^{D(\nu+1)}; (1-\phi)^{-1}(1-\varphi_1)^{-1}\phi M_\sigma\right\} + \max\left\{(1-\varphi_1)^{-1}M_\sigma;\right.\\
&\left.(1+\varepsilon)(1-\phi)^{-1}(1-\varphi_1)^{-1}e^{D(\nu+1)}\frac{M_\sigma}{\sigma}|B|\right\},
\end{aligned} \tag{19}
$$

where $0 < \phi < 1$ and $0 < \varphi_1 < 1$ with:

$$
\begin{aligned}
\phi &= \frac{1+\varepsilon}{1+\lambda}e^{D(\nu+1)}, && (20)\\
\varphi_1 &= (1+\varepsilon)(1-\phi)^{-1}\phi\frac{M_\sigma}{\sigma}|B|, && (21)
\end{aligned}
$$

for some $\varepsilon > 0$. The choice of ν and ε ensures $\phi < 1$ and $\varphi_1 < 1$, which is always possible given (18). For the example, the quantizer is defined component-wise for each $x \in [0, D]$ as:

$$
q_\mu\left(\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, u\right) = \left(\begin{bmatrix}\mu q\left(\frac{X_1}{\mu}\right) & \mu q\left(\frac{X_2}{\mu}\right)\end{bmatrix}^T, \mu q\left(\frac{u}{\mu}\right)\right), \tag{22}
$$

where:

$$
q\left(\frac{u(x)}{\mu}\right) = \begin{cases} M, & \frac{u(x)}{\mu} > M \\ -M, & \frac{u(x)}{\mu} < -M \\ \Delta\left\lfloor \frac{u(x)}{\mu\Delta}+\frac{1}{2}\right\rfloor, & -M \leq \frac{u(x)}{\mu} \leq M \end{cases}. \tag{23}
$$

Requirements

To run this project, you will need:

- MATLAB R2023b or later.

Installation

Follow these steps to set up the project:

1. Download the project files from `https://github.com/flo3221/quantizerp`.
2. Extract the contents to a directory of your choice.
3. Open MATLAB and navigate to the project directory using the `cd` command:

```
cd /path/to/quantizerp
```

Usage

To use **QuantizerP**, follow these steps:

1. Open MATLAB and ensure you are in the project directory.
2. Run the main script or function:

```
Quantized_predictor_feedback.m
```

3. Ensure that the `private` folder is in the same directory. This folder is used in [2] to solve initial-boundary value problems for first-order systems of hyperbolic partial differential equations (PDEs).

Functions

QuantizerP includes the following key functions:

- `hpde.m` and `setup.m`: These functions are used to solve the transport PDE described by equation (3).
- `mu`: Implements the switching parameter $\mu(t)$ as defined in equation (7).
- `quantizer`: Implements the quantizer function described in equation (23).

Examples

Refer to the following script for examples of how to use **QuantizerP**:

`Quantized_predictor_feedback.m`

and

`fixed_mu_quantized_predictor_feedback.m`

for fixed switching parameter case.

Contributing

To contribute to **QuantizerP**, please follow these steps:

- Fork the repository on GitHub.
- Create a new branch for your feature or fix.
- Make your changes and commit them.
- Submit a pull request with a detailed description of your changes.

License

Contact

For questions or feedback, please contact fkoudohode@tuc.gr.

References

[1] F. Koudohode and N. Bekiaris-Liberis, "Simultaneous compensation of input delay and state/input quantization for linear systems via switched predictor feedback," Systems & Control Letters, vol. 192, pp.105912, 2024.

[2] L. F. Shampine, "Solving hyperbolic PDEs in MATLAB," *Applied Numerical Analysis & Computational Mathematics*, vol. 2, no. 3, pp. 346–358, 2005.