Stabilization of Linear Switched Systems with Long Constant Input Delay via Average or Averaging Predictor Feedbacks

1 Purpose of this Document

This document describes the MATLAB code that was used to produce the results in Section 4 of the paper *"Stabilization of Linear Switched Systems with Long Constant Input Delay via Average or Averaging Predictor Feedbacks"*. It explains:

- the system setup and the main equations implemented in the scripts,
- the structure of the repository and the role of each folder,
- how to run all simulations in order to reproduce the figures and numerical values,
- how the theoretical constants ϵ, $\bar{\epsilon}$, $\epsilon^\star$, $\bar{\epsilon}^\star$ and dwell-time bounds $\tau_d^\star$, $\bar{\tau}_d^\star$ are computed.

2 System Setup

We consider a continuous–time switched linear system with input delay $D > 0$ of the form

$$\dot{X}(t) = A_{\sigma(t)}\, X(t) + B_{\sigma(t)}\, U(t - D), \tag{1}$$

where

- $X(t) \in \mathbb{R}^2$ is the state,
- $U(t) \in \mathbb{R}$ is the control input,
- $\sigma(t) \in \{1, 2, 3\}$ is the switching signal indicating the active mode,
- for each $i \in \{1, 2, 3\}$, (A_i, B_i) is a stabilizable pair.

In the numerical examples, we use the specific matrices

$$A_1 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0.97 & 1.15 \\ 1.06 & 2.09 \end{bmatrix}, \qquad A_3 = \begin{bmatrix} 1.08 & 1.2 \\ 1.14 & 2.13 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 1.05 \end{bmatrix}, \qquad B_3 = \begin{bmatrix} 0 \\ 1.1 \end{bmatrix}.$$

For each subsystem we choose a state–feedback gain $K_i \in \mathbb{R}^{1\times 2}$ such that $A_i + B_i K_i$ is Hurwitz. Throughout the code this is done by pole placement:

$$K_i = -\operatorname{place}(A_i, B_i, \{\lambda_1, \lambda_2\}), \quad \lambda_1 = -3,\ \lambda_2 = -2, \quad i = 1, 2, 3,$$

via the auxiliary function `computeK.m`.

In the simulations, we use:

- input/nominal delay $D = 1$ s,
- minimum dwell time $\tau_d = 0.9$ s, maximum dwell time $\bar{\tau}_d = 3$ s
- total simulation time $T = 10$ s,
- sampling period $\Delta t = 10^{-3}$ s.

The switching signal $\sigma(\cdot)$ is a random signal in $\{1, 2, 3\}$ respecting the minimum and maximum dwell time conditions. The specific realization used in the paper is stored in the file `swsig_values.mat` as a vector `swsig_values`. It was generated once using `generateSwSignal.m` and is reused in all simulations for exact reproducibility.

3 Feedback Laws

The paper considers three control laws:

- Controller U_1, as defined in equation (4) in [KBL25]
- Controller U_2, as defined in equation (10) in [KBL25]
- The exact predictor U_{exact}, as defined in equation (140) in [KBL25]

3.1 Average Predictor-Based Controller U_1

The average predictor feedback builds a single "average" pair $(\bar{A}, \bar{B})$ and an associated gain $\bar{K}$. They are computed by solving convex problems of the form

$$\min_{\bar{Y}} R \quad \text{subject to} \quad \left|\bar{Y} - Y_i\right|_2 \le R, \qquad i = 1, 2, 3, \tag{2}$$

for $Y = A, B, K$, using the auxiliary function `optimizeL2.m`. That is,

$$\bar{A} = \text{optimizeL2}(A_1, A_2, A_3), \quad \bar{B} = \text{optimizeL2}(B_1, B_2, B_3), \quad \bar{K} = \text{optimizeL2}(K_1, K_2, K_3).$$

The average predictor controller U_1 is then constructed as follows. Let $\tau_0(t)$ denote the last switching instant and let $\tau(t)$ be the "remaining dwell-time" variable satisfying

$$\tau(t) = \max\{0,\ \tau_0(t) + \tau_d - t\} \in [0, \tau_d].$$

Define

$$X\big(t + \tau(t)\big) = e^{A_{\sigma(\tau_0(t))}\tau(t)} X(t) + \int_{t-D}^{t+\tau(t)-D} e^{A_{\sigma(\tau_0(t))}(t+\tau(t)-D-\theta)} B_{\sigma(\tau_0(t))} U(\theta)\, d\theta. \tag{3}$$

Then U_1 control is given by

$$U_1(t) = \bar{K} \left(e^{\bar{A}(D-\tau(t))} X(t + \tau(t)) + \int_{t+\tau(t)-D}^{t} e^{\bar{A}(t-s)} \bar{B} U(s)\, ds \right). \tag{4}$$

In the code, equations (3)–(4) are implemented using functions that compute numerically (via discretization) the matrix exponentials and the integrals are solved with the left end point Rule:

$$e^{A_i \Delta t} \approx \text{expm}(A_i \Delta t), \qquad e^{\bar{A}\Delta t} \approx \text{expm}(\bar{A}\Delta t).$$

- `U1_dwell.m` implements (1) under (4) under dwell-time knowledge.
- `U1_nodwell.m` implements the same law but with $\tau(t) \equiv 0\ \forall t \ge 0$, i.e. without using dwell time information.

3.2 Averaging Predictors Controller U_2

The averaging of predictors method constructs three separate exact predictors, one per subsystem (A_i, B_i). Their dynamics are based on the fixed matrices A_i, B_i and the same structure as in (3). The corresponding controls are

$$U_i(t) = K_i \left(e^{A_i(D-\tau(t))} X(t+\tau(t)) + \int_{t+\tau(t)-D}^{t} e^{A_i(t-s)} B_i U(s)\, ds \right), \quad i = 1, 2, 3,$$

and the implemented control is the arithmetic average

$$U(t) = \frac{1}{3}\big(U_1(t) + U_2(t) + U_3(t)\big). \tag{5}$$

In the repository:

- `U2_dwell.m` implements (1) with (5) under dwell-time knowledge.
- `U2_nodwell.m` implements the same law but with $\tau(t) \equiv 0\ \forall t \geq 0$, i.e. without using dwell time information.

3.3 Exact Predictor U_{exact}

The exact predictor uses for each mode $i = \sigma(t)$ the nominal delay $D = 1$ and reconstructs the future state $P(t) = X(t + D)$ under the exact mode-dependent dynamics. The corresponding predictor state (for a fixed interval) satisfies equation (18) in [KBL25]. The exact predictor feedback is then of the form

$$U_{\text{exact}}(t) = K_{\sigma(t+D)}\, X(t + D), \tag{6}$$

It is not implementable in practice because it requires full knowledge of future modes.

In the repository, the routine `ExactPredictor.m` implements (6) along the stored switching signal `swsig_values`. It stores the resulting trajectories in `U_ex.mat`.

3.4 Robustness to Delay Perturbations

To test robustness to delay mismatch, we consider the case where the actual delay $D_{\text{actual}} = D$ differs from the delay $D_{\text{controller}}$ used inside the two proposed control laws.

The script `U1_robust_0_95.m` implements the proposed U_1 controller, but with $D_{\text{actual}} = D = 0.95$ in the delayed input and using $D_{\text{controller}} = 1$ in the prediction-based controller.

The script `U1_robust_1_05.m` implements the proposed U_1 controller, but with $D_{\text{actual}} = D = 1.05$ in the delayed input and using $D_{\text{controller}} = 1$ in the prediction-based controller.

The script `U2_robust_0_95.m` implements the proposed U_2 controller, but with $D_{\text{actual}} = D = 0.95$ in the delayed input and using $D_{\text{controller}} = 1$ in the prediction-based controller.

The script `U1_robust_1_05.m` implements the proposed U_2 controller, but with $D_{\text{actual}} = D = 0.95$ in the delayed input and using $D_{\text{controller}} = 1$ in the prediction-based controller.

4 Computation of Parameters

Theorems 1 and 2 in the paper ([KBL25]) define two main parameters:

- ϵ (equation (14) in paper), which is the maximal deviation between the triples (A_i, B_i, K_i) and their averaged counterparts $(\bar{A}, \bar{B}, \bar{K})$,
- $\bar{\epsilon}$ (equation (16) in paper), which is the maximal deviation between subsystems (A_i, B_i, K_i) amongst themselves.

They determine admissible ranges $\epsilon^\star$, $\bar{\epsilon}^\star$ and corresponding dwell time bounds $\tau_d^\star$, $\bar{\tau}_d^\star$.

The script `epsilon_star.m` computes ϵ and then evaluates the rest of the constants defined in Theorem 1 (following the notation in the paper). Eventually, it prints the numerical values of $\epsilon, \epsilon^\star, \tau_d^\star$. The dwell time used in the simulations ($\tau_d = 0.9$ s) can then be compared with $\tau_d^\star$ to determine how conservative the bound is.

In the same manner, script `barepsilon_star.m` computes $\bar{\epsilon}$ and then evaluates the rest of the constants defined in Theorem 2 (following the notation in the paper). Eventually, it prints the numerical values of $\bar{\epsilon}, \bar{\epsilon}^\star, \bar{\tau}_d^\star$ so that the user can compare them with the simulation dwell time τ_d.

5 Performance Index

To compare the performance of different controllers U_1, U_2 and the exact predictor, we use the quadratic performance index

$$J = \int_0^T \left(|X(t)|^2 + |U(t)|^2 \right) dt, \tag{7}$$

approximated via the trapezoidal rule:

$$J \approx \int_0^T |X(t)|^2 dt + \int_0^T |U(t)|^2 dt \approx \text{trapz}(T, |X(t)|^2) + \text{trapz}(T, |U(t)|^2).$$

The script `Performance_Index.m`:

- loads the files `U1_dwell.mat`, `U1_nodwell.mat`, `U2_dwell.mat`, `U2_nodwell.mat`, and `U_ex.mat`,
- extracts the trajectories (X, U) in each case,
- computes J according to (7),
- prints the resulting values in a table.

6 Repository Structure

The repository is organized as follows:

- `Auxiliary Functions/`
  - `computeK.m` (feedback gain design for each mode)
  - `optimizeL2.m` (CVX-based computation of $\bar{A}, \bar{B}, \bar{K}$)
  - `generateSwSignal.m` (generation of random dwell-time switching signal)
  - `swsig_values.mat` (fixed switching signal used in all simulations)
- `Section_4.1/` (System (1) under U_1 and U_2 with dwell time knowledge)
  - `U1_dwell_time.m` (controller U_1)
  - `U2_dwell_time.m` (controller U_2)
  - `epsilon_star.m` (computes $\epsilon, \epsilon^\star, \tau_d^\star$)
  - `barepsilon_star.m` (computes $\bar{\epsilon}, \bar{\epsilon}^\star, \bar{\tau}_d^\star$)
- `Section_4.2/` (Comparison with exact predictor and no dwell time knowledge cases)

  - `U1_nodwell_time.m` (U_1 without dwell time)
  - `U2_nodwell_time.m` (U_2 without dwell time)
  - `ExactPredictor_Uex.m` (exact predictor simulation)
  - `Performance_Index.m` (computes J performance index for all controllers)
- `Section_4.3/` (Delay perturbation / robustness)
  - `U1_robust_0_95.m` (U_1 with $D = 0.95$ and $D_{\text{controller}} = 1$)
  - `U1_robust_1_05.m` (U_1 with $D = 1.05$ and $D_{\text{controller}} = 1$)
  - `U2_robust_0_95.m` (U_2 with $D = 0.95$ and $D_{\text{controller}} = 1$)
  - `U2_robust_1_05.m` (U_2 with $D = 1.05$ and $D_{\text{controller}} = 1$)
- `READMEcod.pdf`
- `README.md`
- `LICENSE`

7 How to Run the Simulations

Requirements

- MATLAB
- CVX toolbox (for `optimizeL2.m`), with a compatible SDP solver (e.g. SeDuMi, SDPT3, MOSEK).
- Symbolic Math Toolbox (for the computation of theoretical constants in `epsilon_star.m` and `barepsilon_star.m`).

General Steps

1. Open MATLAB and set the repository root as the current folder.
2. Ensure that `swsig_values.mat` is present in `Auxiliary Functions/` and that CVX is correctly installed and set up.

Section 4.1: Implement $U_1(t)$ and $U_2(t)$

1. Navigate to `Section_4.1` in MATLAB.
2. Run

```
U1_dwell_time.m
U2_dwell_time.m
epsilon_star.m
barepsilon_star.m
```

3. Scripts `U1_dwell_time.m` and `U2_dwell_time.m` will produce plots of $X(t)$, $U(t)$ and $\sigma(t)$, and save the trajectories as `U1_dwell.mat` and `U2_dwell.mat` if needed.
4. Scripts `epsilon_star.m` and `barepsilon_star.m` will print the theoretical constants derived from Theorem 1 and 2, respectively.

Section 4.2: Comparison with Exact Predictor and Performance Index

1. Navigate to `Section_4.2`.
2. Run the following scripts in any order:

   ```
   U1_nodwell_time.m
   U2_nodwell_time.m
   ExactPredictor.m
   ```

   They will create files `U1_nodwell.mat`, `U2_nodwell.mat`, and `U_ex.mat`.
3. Then run

   ```
   Performance_Index.m
   ```

   to compute and display the values of the performance index J in (7) for all controllers.

Section 4.3: Delay Perturbations

1. Navigate to `Section_4.3`.
2. Run, in any order,

   ```
   U1_robust_0_95.m
   U1_robust_1_05.m
   U2_robust_0_95.m
   U2_robust_1_05.m
   ```

   The scripts generate plots of $X(t)$, $U(t)$, and $\sigma(t)$ for the perturbed-delay cases.

References

[KBL25] Andreas Katsanikakis and Nikolaos Bekiaris-Liberis. Stabilization of linear switched systems with long constant input delay via average or averaging predictor feedbacks. *Revised version submitted in Systems and Control Letters*, 2025.