MATLAB Simulation Codes for [4, Sect. V]

These MATLAB codes reproduce the simulation results from [4, Sect. V]. In addition the scripts `ExVA.m` and `ExVB.m` the codes contain two functions `n1sim.m` and `ksol.m` to run simulations and solve kernel equations, respectively. The simulation results from [4, Sect. V] can be reproduced by running the script files, whereas the function files have an auxiliary purpose in obtaining the simulation results in the considered framework.

Description

The considered class of $n+1$ hyperbolic partial differential equations (PDEs) is of the form

$$u_t^i(t,x) + \lambda_i(x) u_x^i(t,x) = \frac{1}{n}\sum_{j=1}^{n} \sigma_{i,j}(x) u^j(t,x) + W_i(x) v(t,x), \tag{1a}$$

$$v_t(t,x) - \mu(x) v_x(t,x) = \frac{1}{n}\sum_{j=1}^{n} \theta_j(x) u^j(t,x), \tag{1b}$$

on $x \in [0,1]$, with boundary conditions

$$u^i(t,0) = q_i v(t,0), \tag{2a}$$

$$v(t,1) = \frac{1}{n}\sum_{i=1}^{n} r_i u^i(t,1) + U(t), \tag{2b}$$

where $(u^i)_{i=1}^n, v$ are the PDE states and U is the control input. In the framework of [4, Sect. V], the parameters of the system (1), (2) are given by

$$\mu(x) = 1, \tag{3a}$$

$$\lambda_i(x) = 1, \tag{3b}$$

$$\sigma_{i,j}(x) = x^3(x+1)\left(\frac{i}{n} - \frac{1}{2}\right)\left(\frac{j}{n} - \frac{1}{2}\right), \tag{3c}$$

$$W_i(x) = x(x+1)e^x\left(\frac{i}{n} - \frac{1}{2}\right), \tag{3d}$$

$$\theta_i(x) = -70e^{x\frac{35}{\pi^2}}\frac{i}{n}\left(\frac{i}{n} - 1\right), \tag{3e}$$

$$q_i = \cos\left(2\pi\frac{i}{n}\right), \tag{3f}$$

$$r_i = 0, \tag{3g}$$

for $i, j = 1, \ldots, n$, where $n \in \mathbb{N}$ can be chosen freely.

The MATLAB function call

```
[usol, Usol] = n1sim(n,T,tg,Ke)
```

simulates the system (1), (2) with parameters (3) based on a finite-difference approximation with 256 grid points in x, where `n` is the number of the u^i state components, `T` is the end time of the simulation, and `tg` is the number of temporal grid points for plotting the solution. The initial conditions

for all n are fixed according to [4, Sect. V] to $u_0^i(x) = q_i$ for $i = 1, \dots, n$ and $v_0(x) = 1$, for all $x \in [0, 1]$. Finally, the (optional) parameter `Ke` is the control gain, such that the control law $U(t)$ is of the form

$$U(t) = \int_0^1 \left[\frac{1}{n} \sum_{i=1}^{n} k^i(1, \xi) u^i(t, \xi) + k^{n+1}(1, \xi) v(t, \xi) \right] d\xi, \tag{4}$$

i.e., `Ke` (roughly) corresponds to $(k^i(1, \xi))_{i=1}^{n+1}$. The output `usol` of `n1sim` corresponds to the (finite-difference approximation of the) solution $((u^1(t), \dots, u^n(t), v(t))$ to (1), (2) evaluated at `tg` evenly distributed time instances on $[0, T]$, and `Usol` is the respective control input (4). If the gain `Ke` is not provided as an input argument, the `n1sim` function uses in (4) continuum-based, approximate gains

$$k_{\text{ca}}^i(1, \xi) = 35 \frac{i}{n} \left(\frac{i}{n} - 1 \right) \exp\left(\frac{35}{\pi^2} \xi \right), \quad i = 1, \dots, n \tag{5a}$$

$$k_{\text{ca}}^{n+1}(1, \xi) = \frac{35}{2\pi^2}, \tag{5b}$$

based on the continuum kernels from the example in [1, Sect. VII]

The MATLAB function call

```
K1 = ksol(n)
```

is provided as a method to compute the gains $(k^i(1, \xi))_{i=1}^{n+1}$ based on the exact backstepping transform for $n + 1$ PDEs [3], i.e., one can use `Ke = ksol(n)` when calling `n1sim`. The `ksol` function combines finite-differences and successive approximations (see, e.g., [2, App. F]) to compute $(k^i(1, \xi))_{i=1}^{n+1}$ by (numerically) solving the following kernel equations, for parameters given in (3),

$$\mu(x) k_x^i(x, \xi) - \lambda_i(\xi) k_\xi^i(x, \xi) = \lambda_i'(\xi) k^i(x, \xi) + \frac{1}{n} \sum_{j=1}^{n} \sigma_{j,i}(\xi) k^j(x, \xi) + \theta_i(\xi) k^{n+1}(x, \xi), \tag{6a}$$

$$\mu(x) k_x^{n+1}(x, \xi) + \mu(\xi) k_\xi^{n+1}(x, \xi) = -\mu'(\xi) k^{n+1}(x, \xi) + \frac{1}{n} \sum_{j=1}^{n} W_j(\xi) k^j(x, \xi), \tag{6b}$$

on a triangular domain $0 \le \xi \le x \le 1$ with boundary conditions

$$k^i(x, x) = -\frac{\theta_i(x)}{\lambda_i(x) + \mu(x)}, \tag{7a}$$

$$\mu(0) k^{n+1}(x, 0) = \frac{1}{n} \sum_{j=1}^{n} q_j \lambda_j(0) k^j(x, 0), \tag{7b}$$

for all $x \in [0, 1]$. The `ksol` function is set to terminate when the successive approximations for the solution differ less than 10^{-8} in norm, at which point the function displays the number of the terminating iteration. Moreover, the function is set to terminate after 400 iterations regardless of convergence, which is to avoid potential infinite loops. Regardless, in the simulations conducted in [4, Sect. V], the `ksol` function reaches the set solution tolerance of 10^{-8} in under 200 iterations for all n considered.

Examples

The simulation results for [4, Sect. V.A] can be reproduced by running the script `ExVA.m`, and the simulation results for [4, Sect. V.B] can be reproduced by running the script `ExVB.m`

References

[1] V. Alleaume and M. Krstic. Ensembles of hyperbolic PDEs: Stabilization by backstepping. *IEEE Trans. Automat. Control*, 70:905–920, 2025.

[2] H. Anfinsen and O. M. Aamo. *Adaptive Control of Hyperbolic PDEs*. Springer, 2019.

[3] F. Di Meglio, R. Vazquez, and M. Krstic. Stabilization of a system of $n + 1$ coupled first-order hyperbolic linear PDEs with a single boundary input. *IEEE Trans. Automat. Control*, 58:3097–3111, 2013.

[4] J.-P. Humaloja and N. Bekiaris-Liberis. Stabilization of a class of large-scale systems of linear hyperbolic PDEs via continuum approximation of exact backstepping kernels. *IEEE Trans. Automat. Control*, pages 1–16, 2025.