Rapid Control Algorithm Validation based on the CAST Architecture

Yutong Li

Visiting Scholar, CAST, Texas A&M University
Vehicle Dynamics & Control Lab, Univ. of California, Berkeley
State Key Laboratory of Automotive Safety and Energy, Tsinghua Univ.

Center for Autonomous and Safe Technologies (CAST)
Mechanical Engineering
Texas A & M University, College Station

Feb 03, 2017

Outlines

- ✓ **Introduction**
  - *Receding horizon sliding control (RHSC) for PWA systems*
  - *CAST architecture*
- ✓ **Robustness analysis of RHSC**
  - *Control performance with no model mismatch*
  - *Parametric and Dynamics uncertainties*
- ✓ **Validation of real-time performance of RHSC based on Hardware in loop tests**
- ✓ **Conclusions and interesting extensions**

Receding horizon sliding control (RHSC) for PWA systems

✓ Why PWA?

- Theoretically, PWA provides a framework for modeling the behavior of switching between multiple modes with linear dynamics. Tough and still open.
- Practically, PWA is widely used for mechatronics systems modeling, especially in automotive engineering (powertrain, vehicle dynamics, suspensions…).

✓ Why RHSC?

	Model predictive control	Sliding mode control
Handle constraints	√	×
Predictive action	√	×
MIMO systems	√	×
Computational burden	×	√
Nonlinear system	×	√
Parameter tuning	×	√

Receding horizon sliding control (RHSC) for PWA systems

✓ **Design procedures of RHSC for PWA systems**

$$\left.\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}_{j_k}\mathbf{x}_k + \mathbf{B}_{j_k}u_k + \mathbf{f}_{j_k} \\ \mathbf{y}_k &= \mathbf{C}_{j_k}\mathbf{x}_k\end{aligned}\right\} when \quad \mathbf{x}_k \in \Omega_{j_k}$$

$$\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \qquad u \in \mathcal{U} \subseteq \mathbb{R} \qquad \mathbf{y} \in \mathbb{R}^m$$

1. **Design sliding surface for each modes in PWA systems**

$$s_{k,j_k} = \mathcal{D}_{j_k}(\varepsilon_{k,j_k}) \quad when \quad \mathbf{x}_k \in \Omega_{j_k}$$

$$\varepsilon_{k,j_k} = \mathbf{y}_{k,j_k} - \mathbf{y}_k^{des}$$

2. **Define a vector S_{k+1} that contains the variable $s_{k+1,j_{k+1}}$ over a *N*-steps prediction horizon:** $\mathrm{S}_{k+1} = [s_{k+1,j_{k+1}} \quad s_{k+2,j_{k+2}} \quad \ldots \quad s_{k+N,j_{k+N}}]^T$
3. **Solve the optimization problem in predictive control scheme:**

$$\min_{\mathbf{X}_k,\mathbf{U}_k} \quad \left\|\mathbf{MS}_k\right\|_2^2$$

$$\begin{aligned} s.t. \qquad & s_{k,j_k} = \mathcal{D}_{j_k}(\varepsilon_{k,j_k}), \quad i = k,\ldots,k+N-1 \\ & \mathbf{x}_{i+1} = \mathbf{A}_{j_i}\mathbf{x}_i + \mathbf{B}_{j_i}u_i + \mathbf{f}_{j_i}, \quad i = k,\ldots,k+N-1 \\ & y_i = \mathbf{C}_{j_i}\mathbf{x}_i, \quad i = k,\ldots,k+N \\ & \mathbf{x}_i \in \Omega_{j_k}, \quad u_i \in \mathcal{U}, \quad i = k,\ldots,k+N-1 \end{aligned}$$

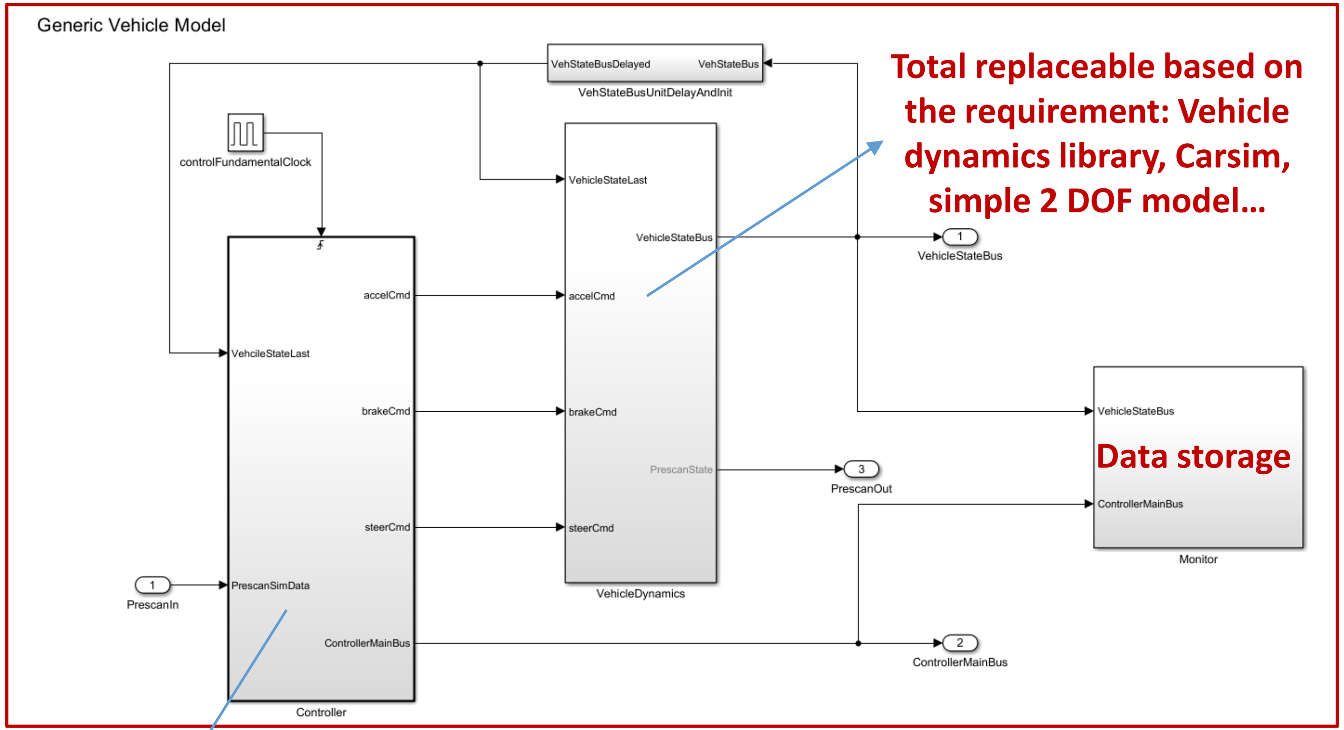
Yutong Li, Andreas Hansen, J. Karl Hedrick, Junzhi Zhang, “A receding horizon sliding control approach for electric powertrains with backlash and flexible half-shafts”, in review, *Vehicle System Dynamics*, 2016.
Yutong Li, Andreas Hansen, Chang Liu, Swaminathan Gopalswamy, J. Karl Hedrick, “Receding horizon sliding control for piecewise affine systems”, manuscript ready, 2017.

CAST control architecture

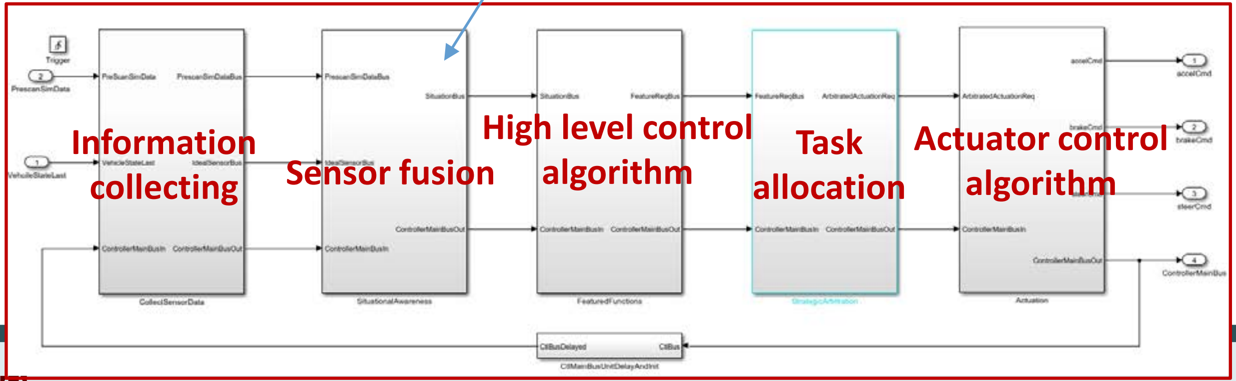
✓ Main idea and proposal

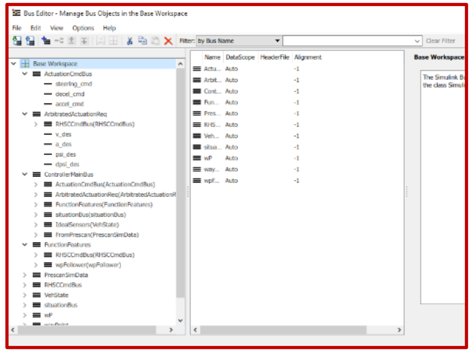
- Provide a platform based on the real vehicle control system architecture, each module provides different functions.
- The information exchange among each module is achieved by updating the bus.
- This way, newly developed control algorithm, sensor fusion algorithm, vehicle dynamics model and etc. can be easily integrated and tested within the architecture.

CAST control architecture



Information generated from new controller can be easily added by Buseditor



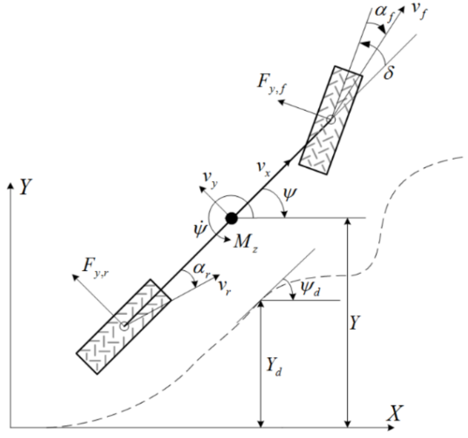


Outlines

- ✓ **Introduction**
  - – *Receding horizon sliding control (RHSC) for PWA systems*
  - – *CAST architecture*
- ✓ **Robustness analysis of RHSC**
  - – *Control performance with no model mismatch*
  - – *Parametric and Dynamics uncertainties*
- ✓ **Validation of real-time performance of RHSC based on Hardware in loop tests**
- ✓ **Conclusions and interesting extensions**

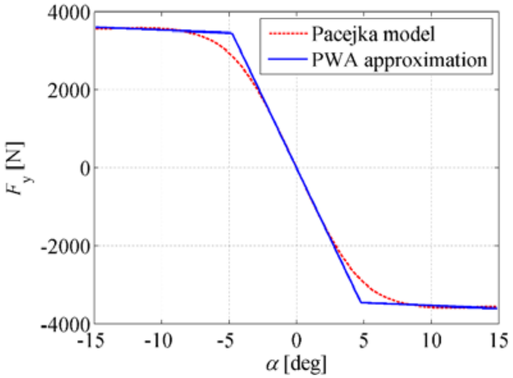
Control algorithm design for path following

✓ **Apply RHSC for PWA systems to autonomous vehicle path following problems**



$$\begin{cases} \dot{Y} = v_y + v_x\psi \\ \dot{v}_y = \dfrac{F_f + F_r}{m} - v_x\dot{\psi} \\ \dot{\psi} = \dot{\psi} \\ \ddot{\psi} = \dfrac{F_f l_f - F_r l_r}{I_z} \end{cases}$$

+



$$F_{y*} = \begin{cases} K_{*,2}\alpha_f + b_{*,2} & \alpha_* \le -\alpha_{*\_the1} \\ K_{*,1} & \alpha_{*\_the1} < \alpha_* < \alpha_{*\_the2} \quad (* = i, j) \\ K_{*,3}\alpha_f + b_{*,3} & \alpha_{*\_the2} \le \alpha_* \end{cases}$$

$$\begin{cases} \dot{Y} = v_y + v_x\psi \\ \dot{v}_y = \dfrac{K_f + K_r}{mv_x} v_y + \left(\dfrac{K_f l_f - K_r l_r}{mv_x} - v_x\right)\dot{\psi} - \dfrac{K_f}{m}\delta + \dfrac{b_f + b_r}{m} \\ \dot{\psi} = \dot{\psi} \\ \ddot{\psi} = \dfrac{K_f l_f - K_r l_r}{I_z v_x} v_y + \left(1 + \dfrac{K_f l_f^2 + K_r l_r^2}{I_z v_x}\right)\dot{\psi} - \dfrac{K_f l_f}{I_z}\delta + \dfrac{1}{I_z} M_z + \dfrac{b_f l_f - b_r l_r}{I_z} \end{cases}$$

Control algorithm design for path following

✓ Formulate the RHSC problem

- **Different sliding surface for different states:**

$$s^{Y}_{k,j_k} = (Y_{k+1} - Y^{des}_{k+1}) - \alpha_Y (Y_k - Y^{des}_k)$$

$$s^{\psi}_{k,j_k} = (\psi_{k+1} - \psi^{des}_{k+1}) - \alpha_\psi (\psi_k - \psi^{des}_k)$$

- **RHSC optimization problem for MIMO systems:**

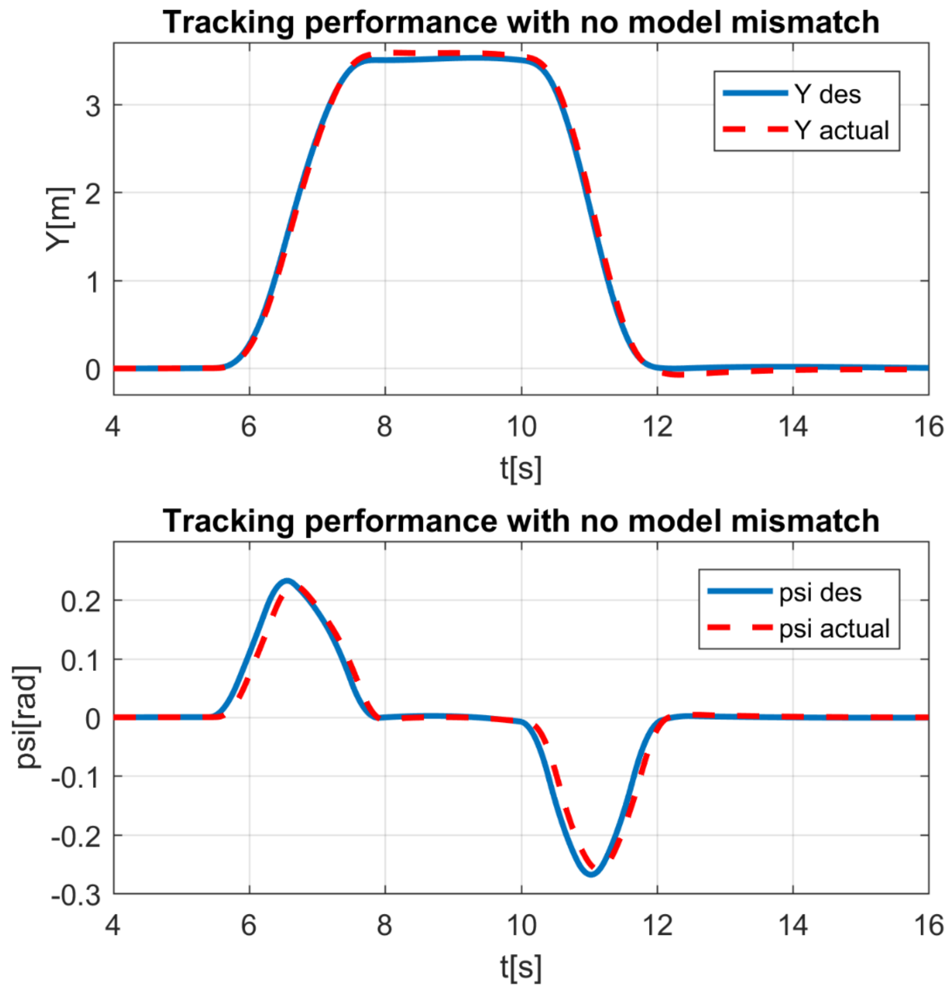
$$\min_{\mathbf{X}_k, \mathbf{U}_k} \quad \left\| \mathbf{M}\mathbf{S}_{k+1} \right\|_F^2$$

$$\begin{aligned}
s.t. \quad & \mathbf{s}_{i+1, j_{i+1}} = \mathcal{D}_{j_{i+1}}(\boldsymbol{\varepsilon}_{i+1, j_{i+1}}) \quad i = k, \ldots, k+N-1, \quad j_i \in \{1, 2, \ldots, 9\} \\
& \mathbf{x}_{i+1} = \mathbf{A}_{j_i}\mathbf{x}_i + \mathbf{B}_{j_i}\mathbf{u}_i + \mathbf{f}_{j_i}, \quad i = k, \ldots, k+N_{j_i} \\
& \mathbf{y}_{i+1} = \mathbf{C}_{j_i}\mathbf{x}_i, \quad i = k, \ldots, k+N_{j_i} \\
& |\mathbf{u}_k| \le \mathbf{u}_{\max}, \quad i = k, \ldots, k+N_{j_i} \\
& |\Delta\mathbf{u}_k| \le \Delta\mathbf{u}_{\max}, \quad i = k, \ldots, k+N_{j_i}
\end{aligned}$$

- **The optimization problem is solved by Yalmip in conjunction with Mosek solver. N=5, Ts = 0.05s, 9 modes.**

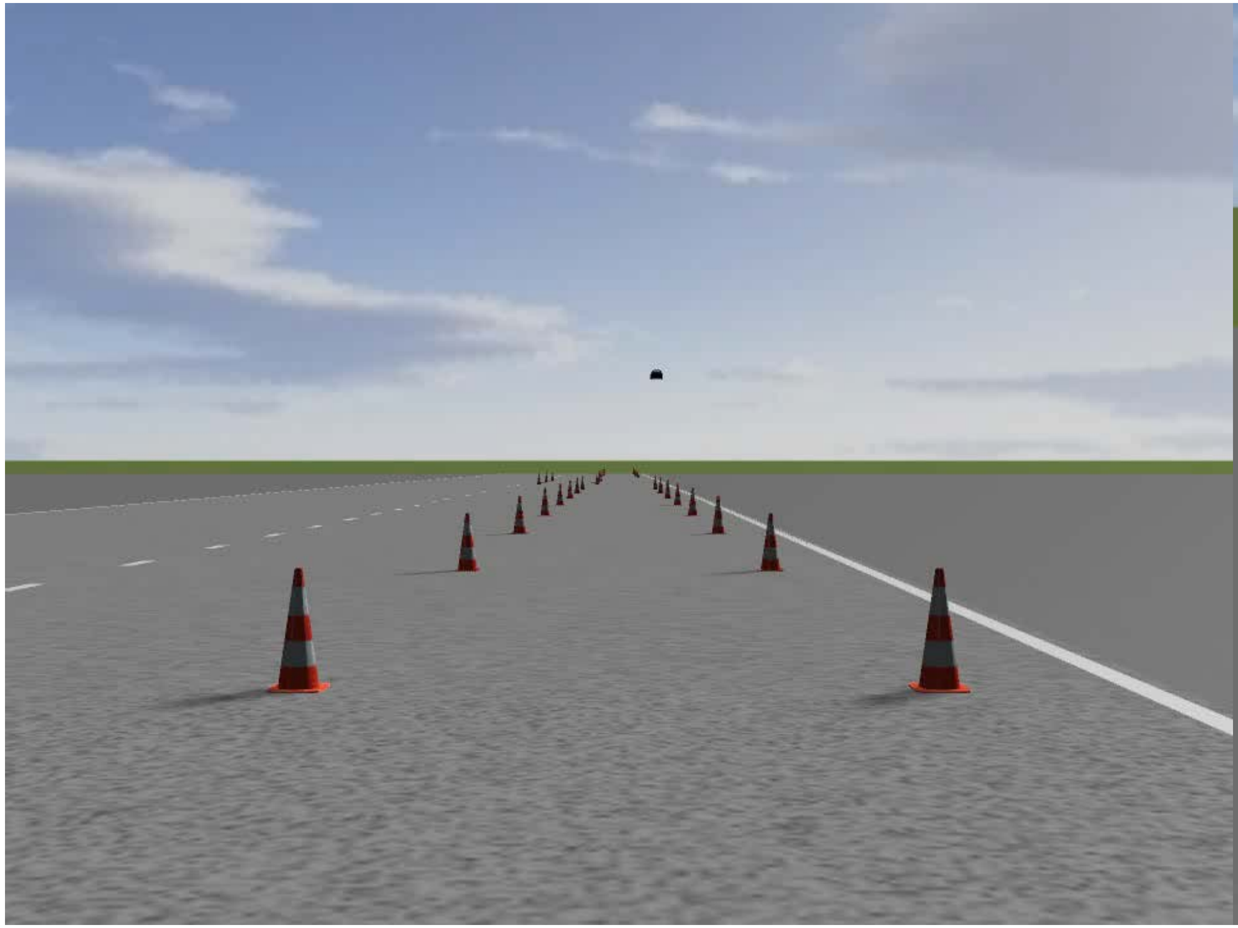
Simulation results with no model mismatch

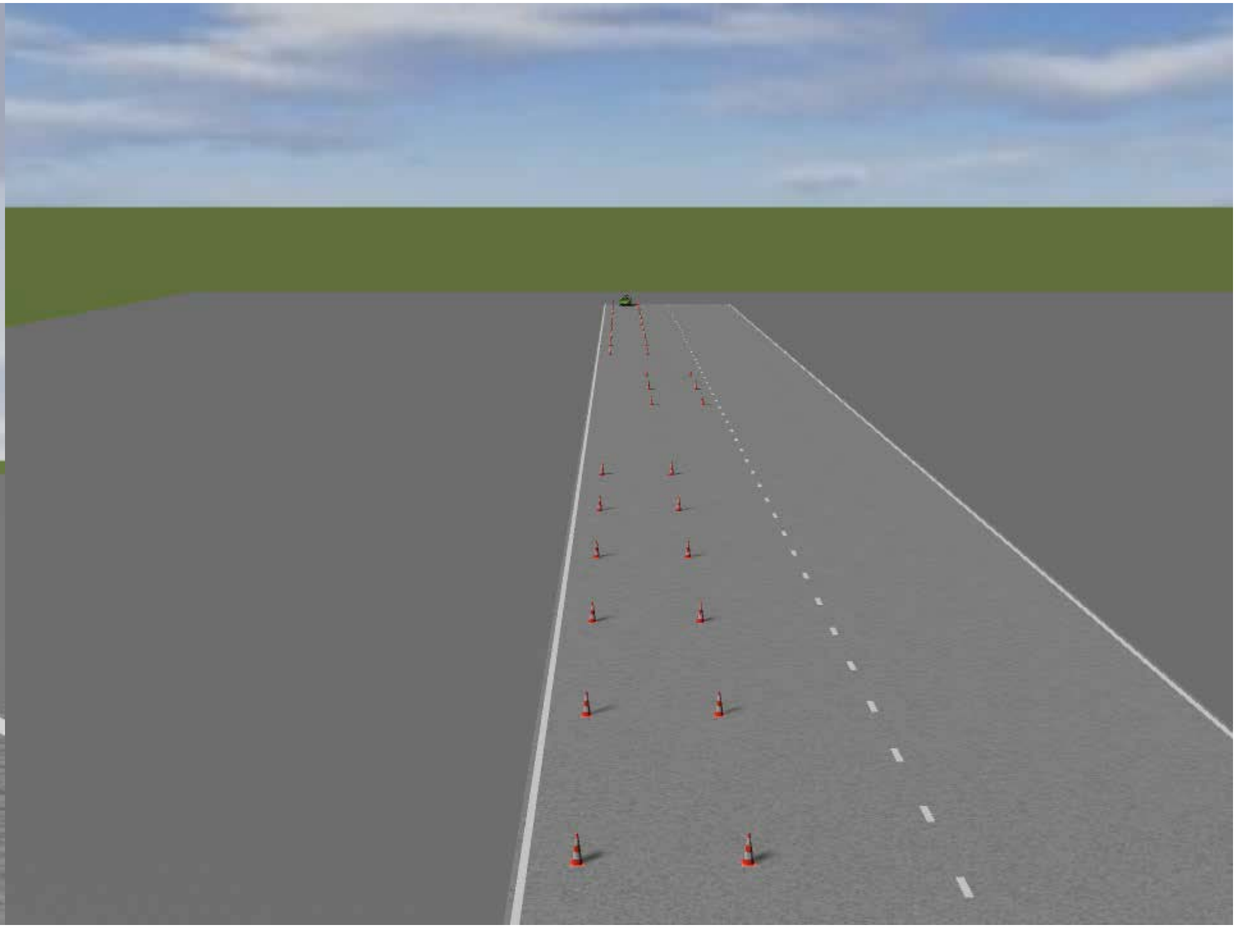
✓ **Follow the predefined ISO double lane change trajectory (high miu)**



Simulation results with no model mismatch

✓ **Follow the predefined ISO double lane change trajectory (with PreScan)**





Robustness analysis of RHSC

✓ **Analysis the robustness of RHSC against model parameter uncertainties**

- **Change the parameters in RHSC model, for example, tire lateral stiffness, mass, inertia of moment…**
- **Obtain important tips for choosing the parameters of the model we use for controller design.**

✓ **Analysis the robustness of RHSC against dynamics uncertainties (take this case as an example)**

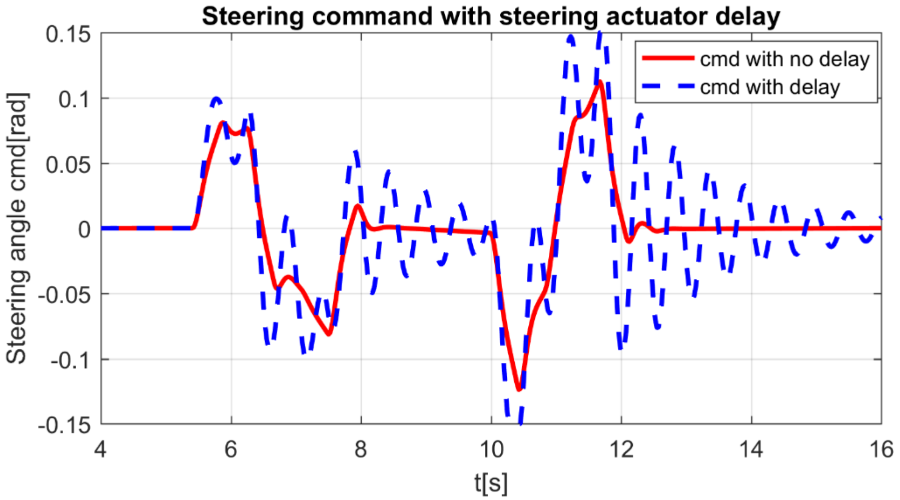
- **Dynamics of the actuator: in our case, we consider the time delay of the steering servo in RC car. The dynamics of the steering system is as follows, where $f_a = 2Hz$ which is the frequency value of the RC car's servo.**

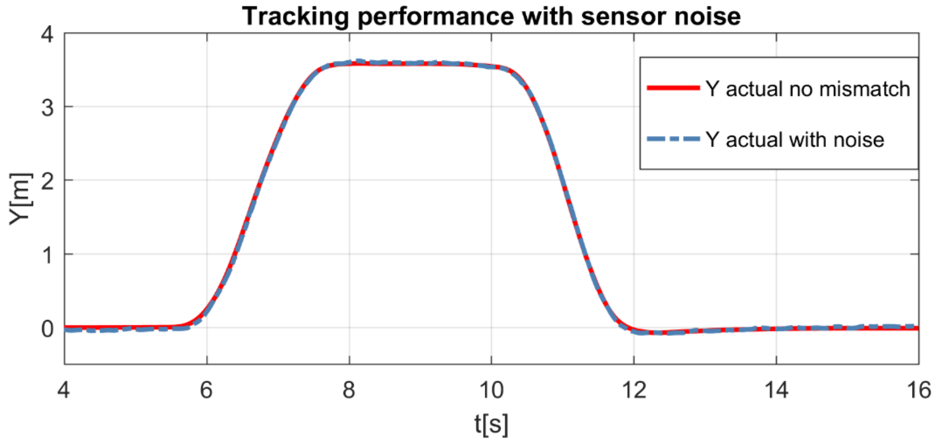
$$\dot{\delta} = -\frac{1}{f_a}\delta + \frac{1}{f_a}\delta_{cmd}$$

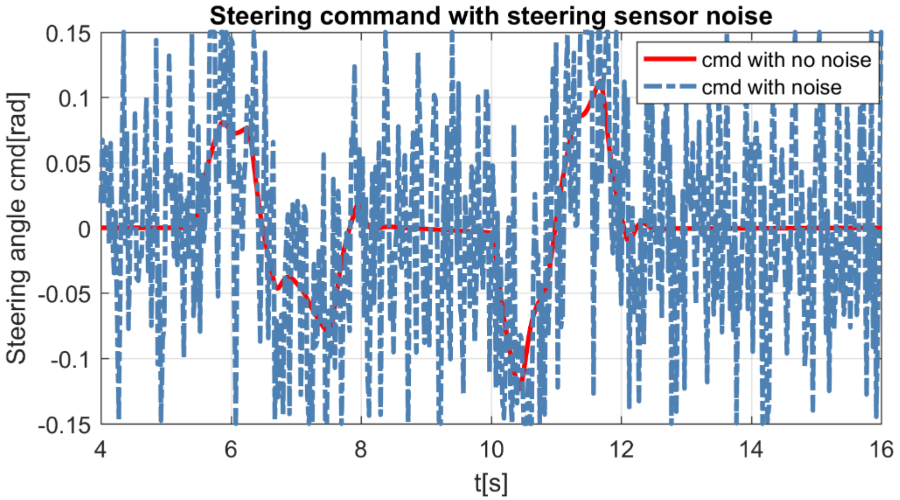
- **Sensor noise: we model the sensor noise as the Gaussian white noise, where the signal noise ratio *SNR* = 40dB, which is the same value as the IMU in the RC car.**

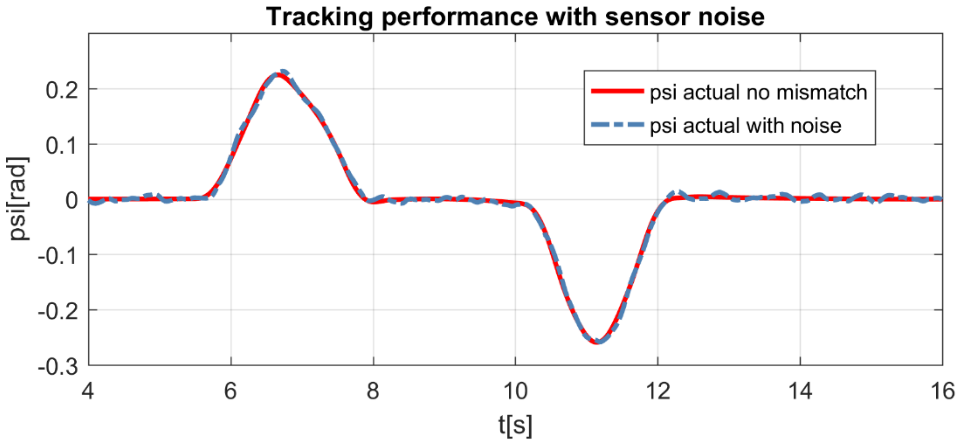
Yutong Li, Andreas Hansen, Swaminathan Gopalswamy, J. Karl Hedrick, "Path following control for driverless vehicles: a multi-input multi-output receding horizon sliding control approach", manuscript in preparation, 2017.

Simulation results with actuator delay and sensor noise









- **Tracking performances of Y and psi are degenerated.**
- **Control input signal becomes oscillating, not acceptable!**

Control algorithm design for actuator delay compensation

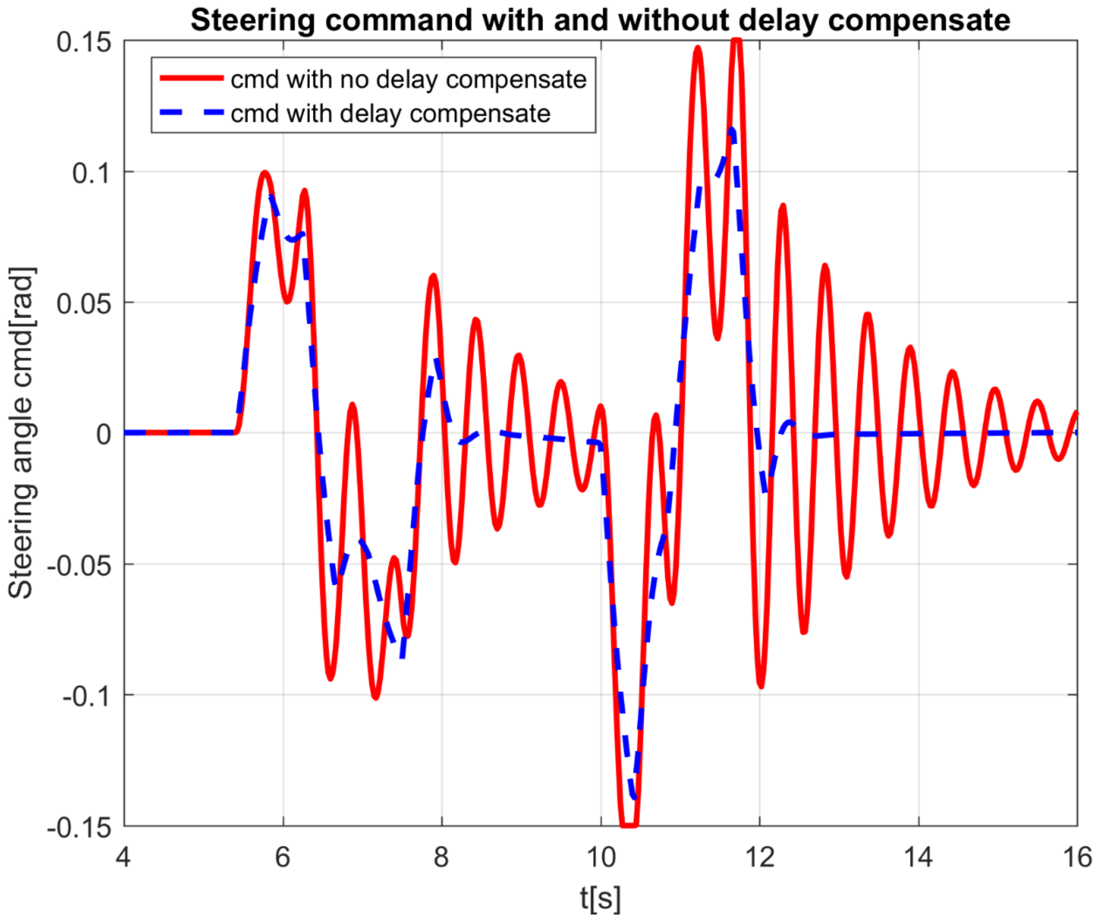
- ✓ **Main idea: explicitly consider the steering system's dynamics in the path following vehicle dynamics model.**

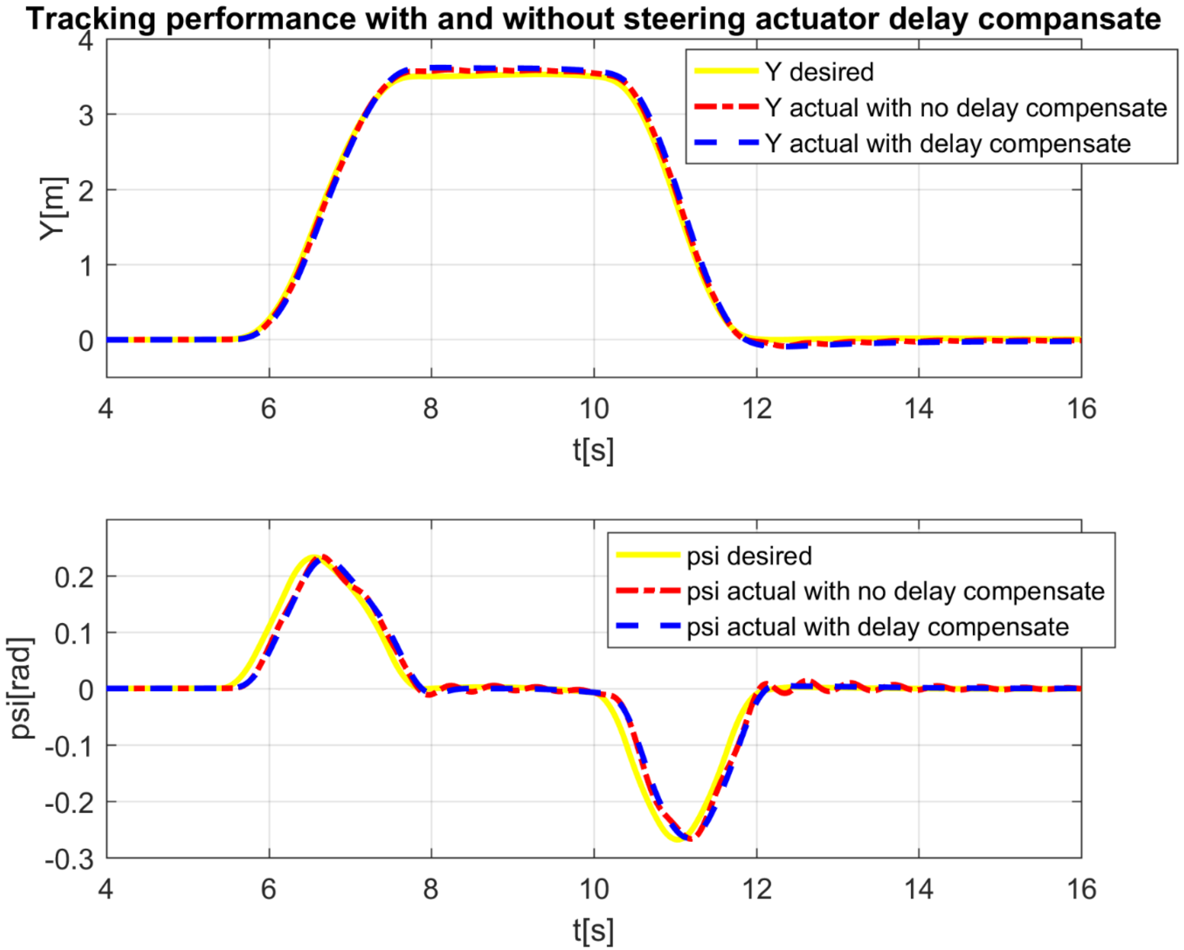
$$\begin{cases} \dot{Y} = v_y + v_x \psi \\ \dot{v}_y = \dfrac{K_f + K_r}{m v_x} v_y + \left(\dfrac{K_f l_f - K_r l_r}{m v_x} - v_x\right)\dot{\psi} - \dfrac{K_f}{m}\delta + \dfrac{b_f + b_r}{m} \\ \dot{\psi} = \dot{\psi} \\ \ddot{\psi} = \dfrac{K_f l_f - K_r l_r}{I_z v_x} v_y + \left(1 + \dfrac{K_f l_f^2 + K_r l_r^2}{I_z v_x}\right)\dot{\psi} - \dfrac{K_f l_f}{I_z}\delta + \dfrac{b_f l_f - b_r l_r}{I_z} \end{cases}$$

$$\Rightarrow$$

$$\begin{cases} \dot{Y} = v_y + v_x \psi \\ \dot{v}_y = \dfrac{K_f + K_r}{m v_x} v_y + \left(\dfrac{K_f l_f - K_r l_r}{m v_x} - v_x\right)\dot{\psi} - \dfrac{K_f}{m}\delta + \dfrac{b_f + b_r}{m} \\ \dot{\psi} = \dot{\psi} \\ \ddot{\psi} = \dfrac{K_f l_f - K_r l_r}{I_z v_x} v_y + \left(1 + \dfrac{K_f l_f^2 + K_r l_r^2}{I_z v_x}\right)\dot{\psi} - \dfrac{K_f l_f}{I_z}\delta + \dfrac{b_f l_f - b_r l_r}{I_z} \\ \dot{\delta} = -\dfrac{1}{f_a}\delta + \dfrac{1}{f_a}\delta_{cmd} \end{cases}$$

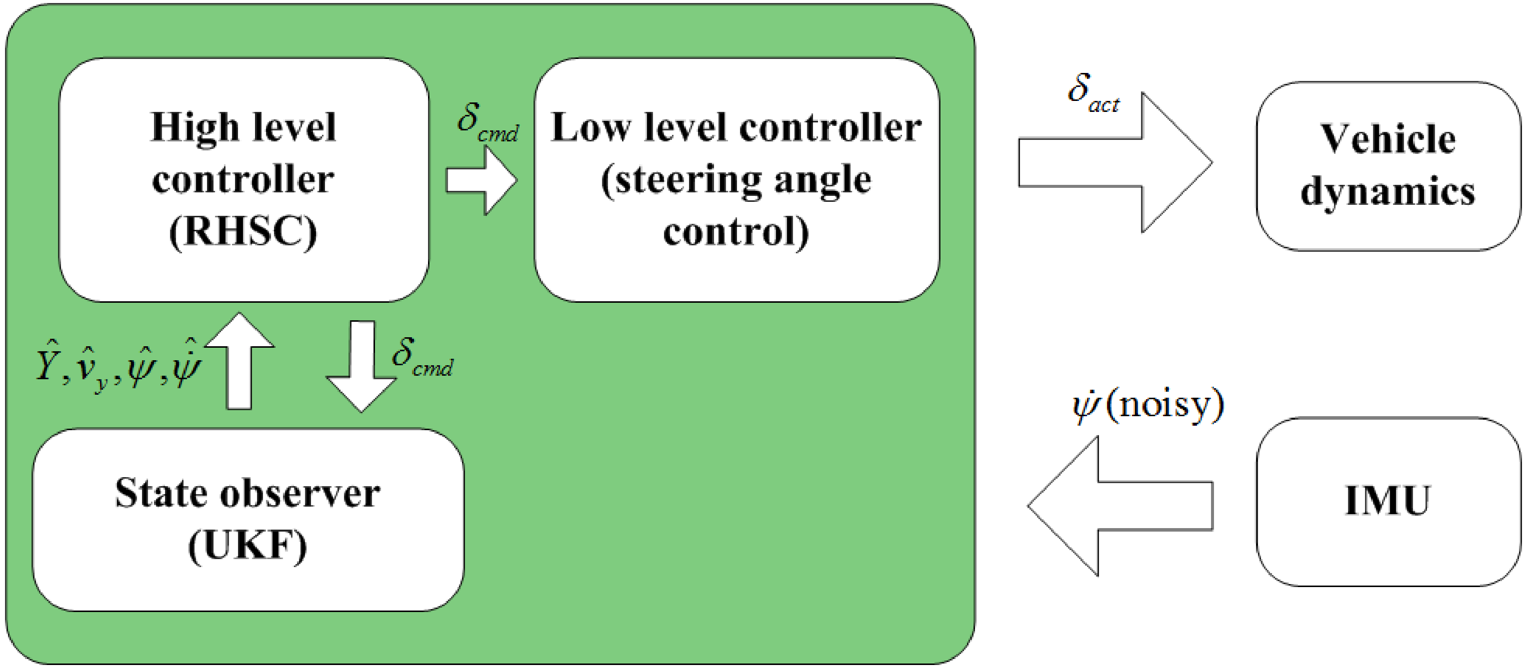
Control algorithm design for actuator delay compensation



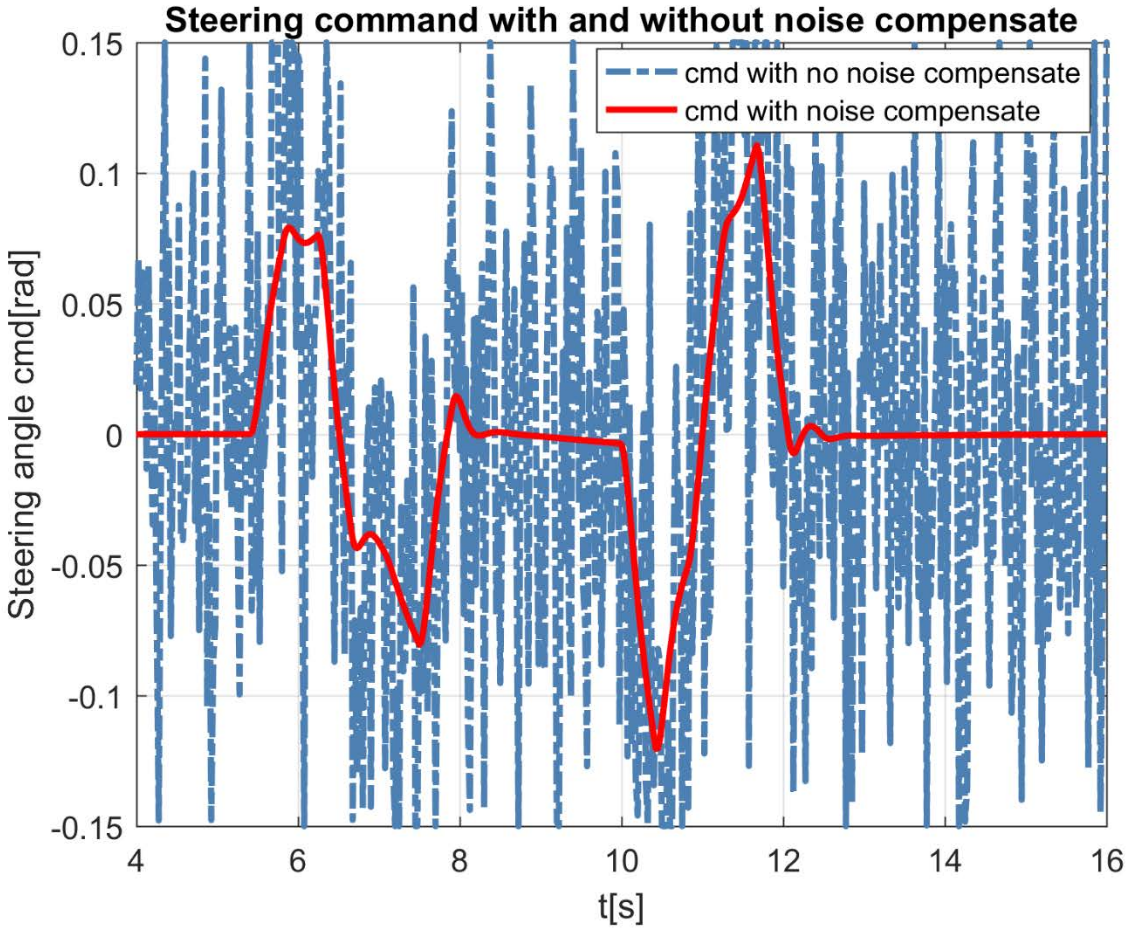


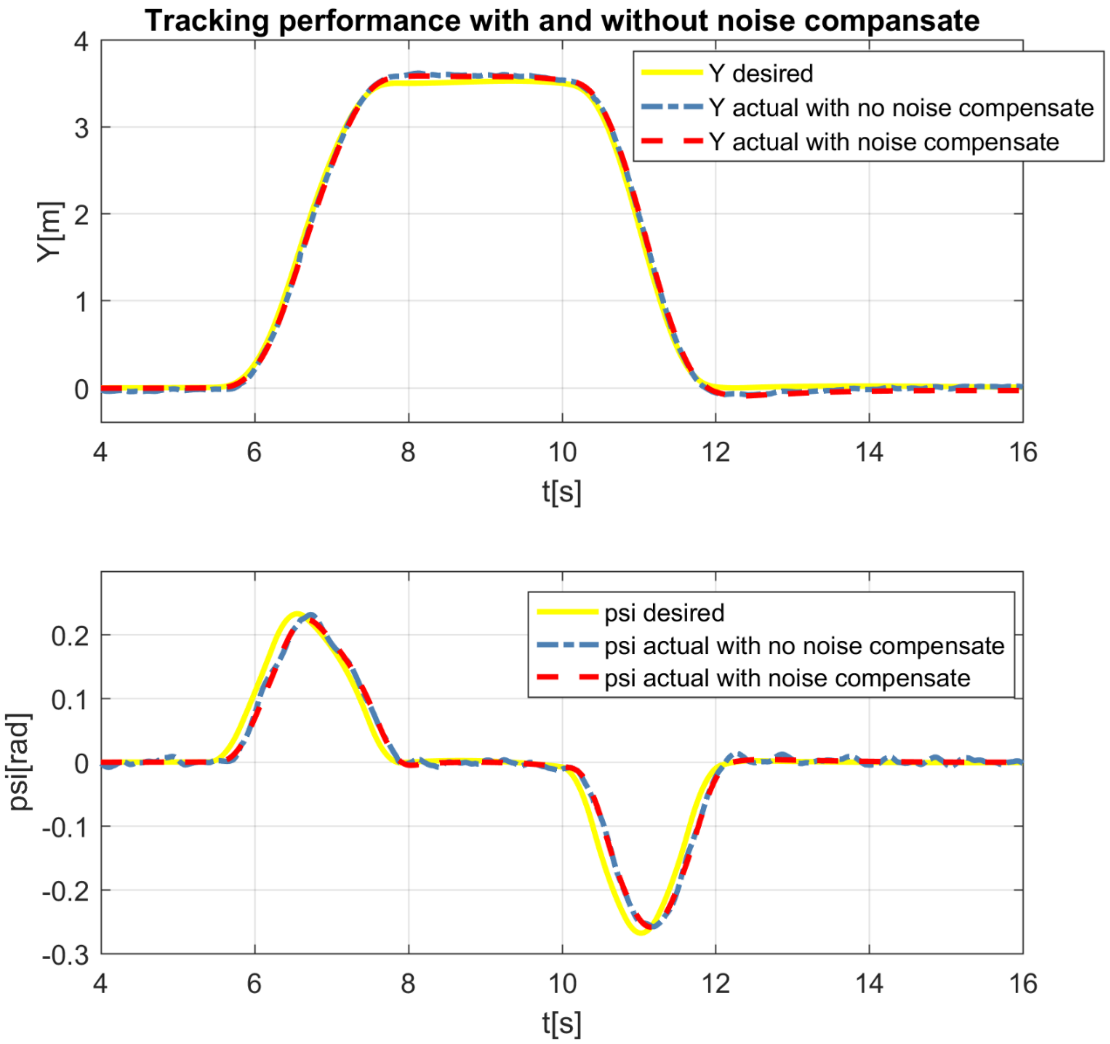
Control algorithm design for sensor noise compensation

- ✓ **Main idea: as the sensor's information is too noisy to be used in RHSC prediction, we introduce a Unscented Kalman Filter (UKF) as an observer.**

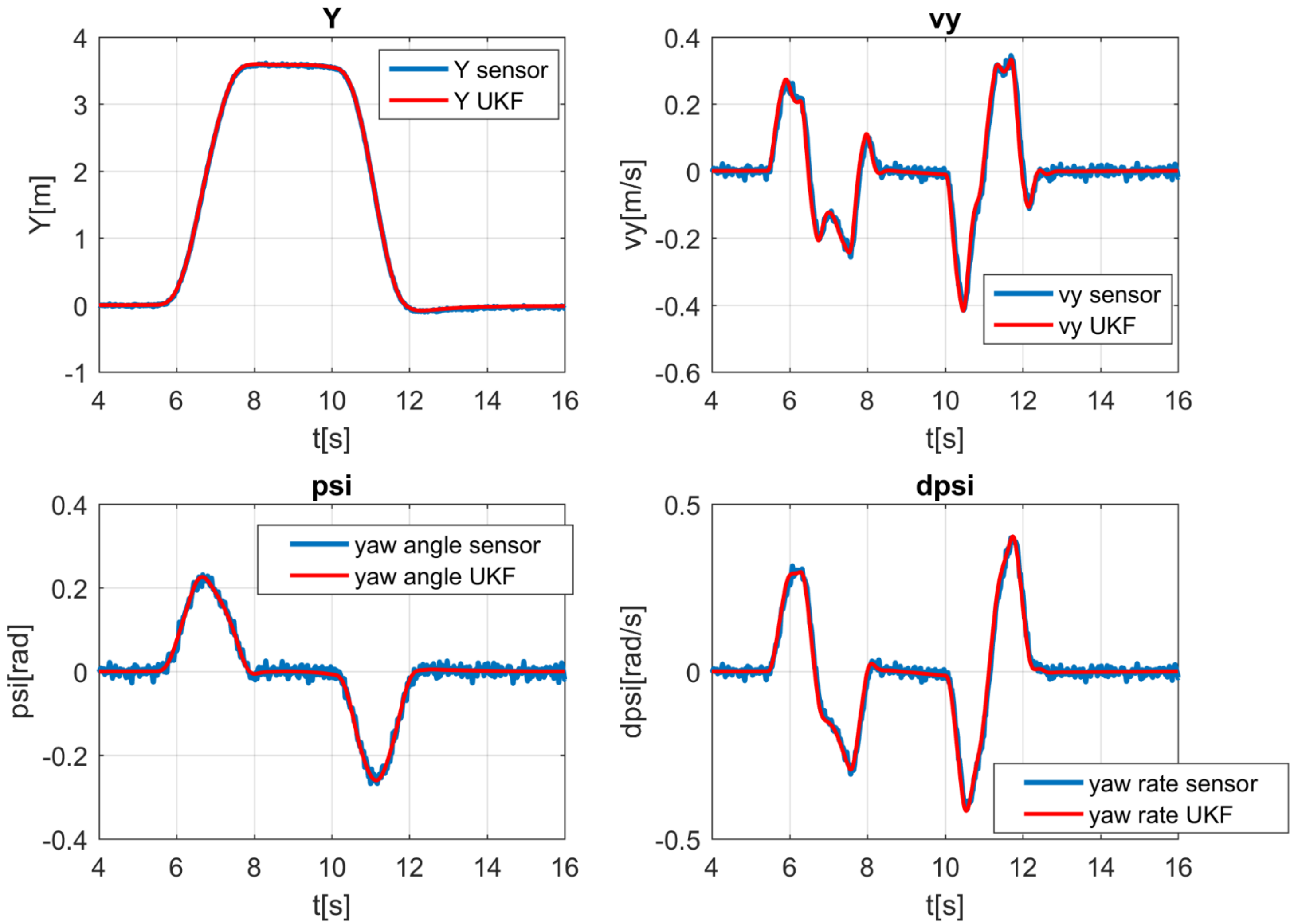


Control algorithm design for sensor noise compensation





Control algorithm design for sensor noise compensation



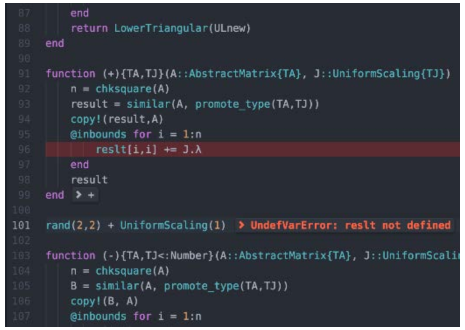
Outlines

- ✓ **Introduction**
  - *Receding horizon sliding control (RHSC) for PWA systems*
  - *CAST architecture*
- ✓ **Robustness analysis of RHSC**
  - *Control performance with no model mismatch*
  - *Parametric and Dynamics uncertainties*
- ✓ **Validation of real-time performance of RHSC based on Hardware in loop tests**
- ✓ **Conclusions and interesting extensions**

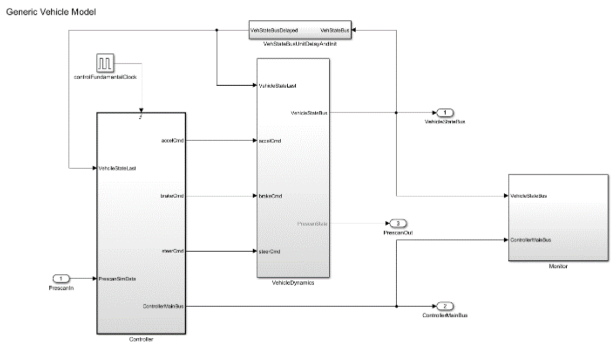
HIL tests for real-time performance of RHSC

- ✓ **Main purpose: to test if the mixed-integer programming can be solved on-line.**
- ✓ **Main idea: run vehicle dynamics model on Simulink, run RHSC on Intel nuc, use ROS to communicate.**

RHSC in Julia language

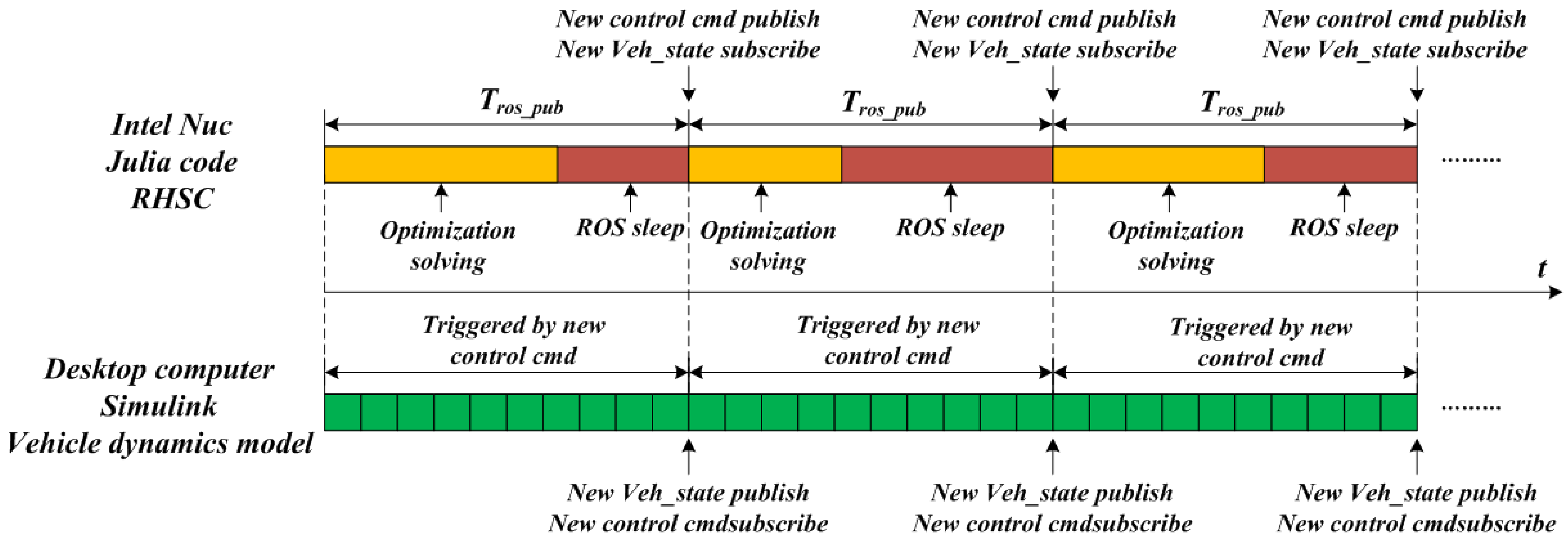


ROS



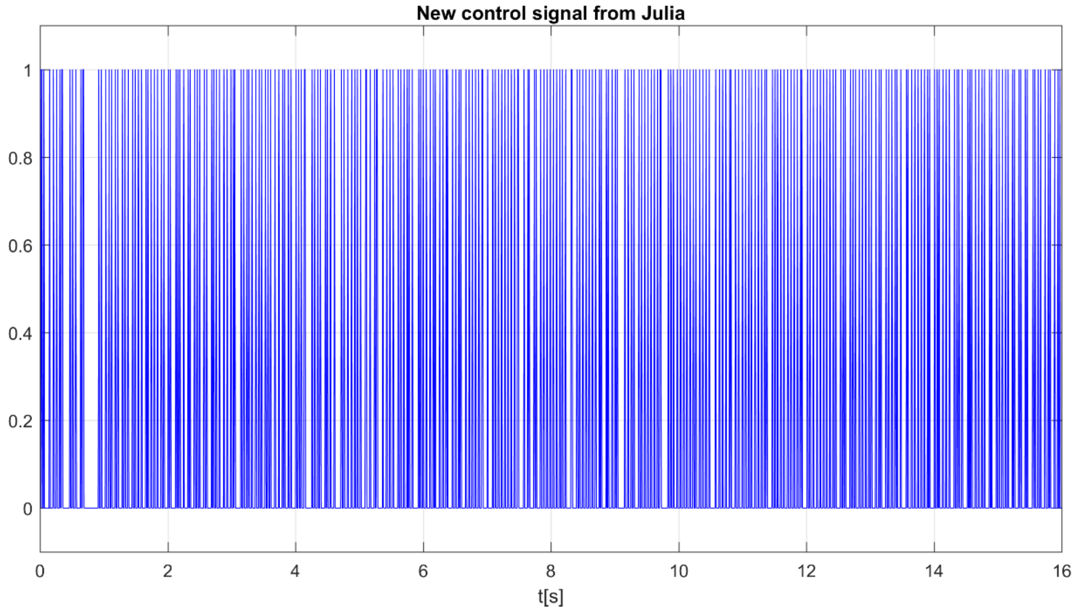
HIL tests for real-time performance of RHSC

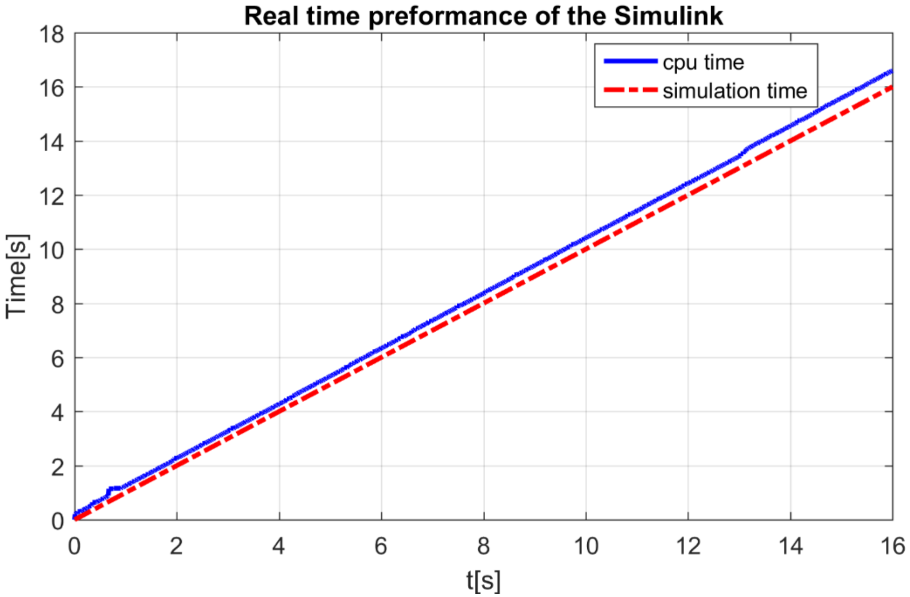
- ✓ **Scheduling and synchronization scheme: vehicle dynamics model is triggered by RHSC.**



HIL tests for real-time performance of RHSC

- ✓ Not hard real-time, but soft real-time.
- ✓ Still can satisfy the test requirements, as we just want to validate the real-time performance of RHSC.





Soft real-time on Julia side.

Soft real-time on Simulink side.

HIL tests for real-time performance of RHSC

✓ Formulate the RHSC problem

- **Different sliding surface for different states:**

$$s^{Y}_{k,j_k} = (Y_{k+1} - Y^{des}_{k+1}) - \alpha_Y (Y_k - Y^{des}_k)$$

$$s^{\psi}_{k,j_k} = (\psi_{k+1} - \psi^{des}_{k+1}) - \alpha_\psi (\psi_k - \psi^{des}_k)$$

- **RHSC optimization problem for MIMO systems:**

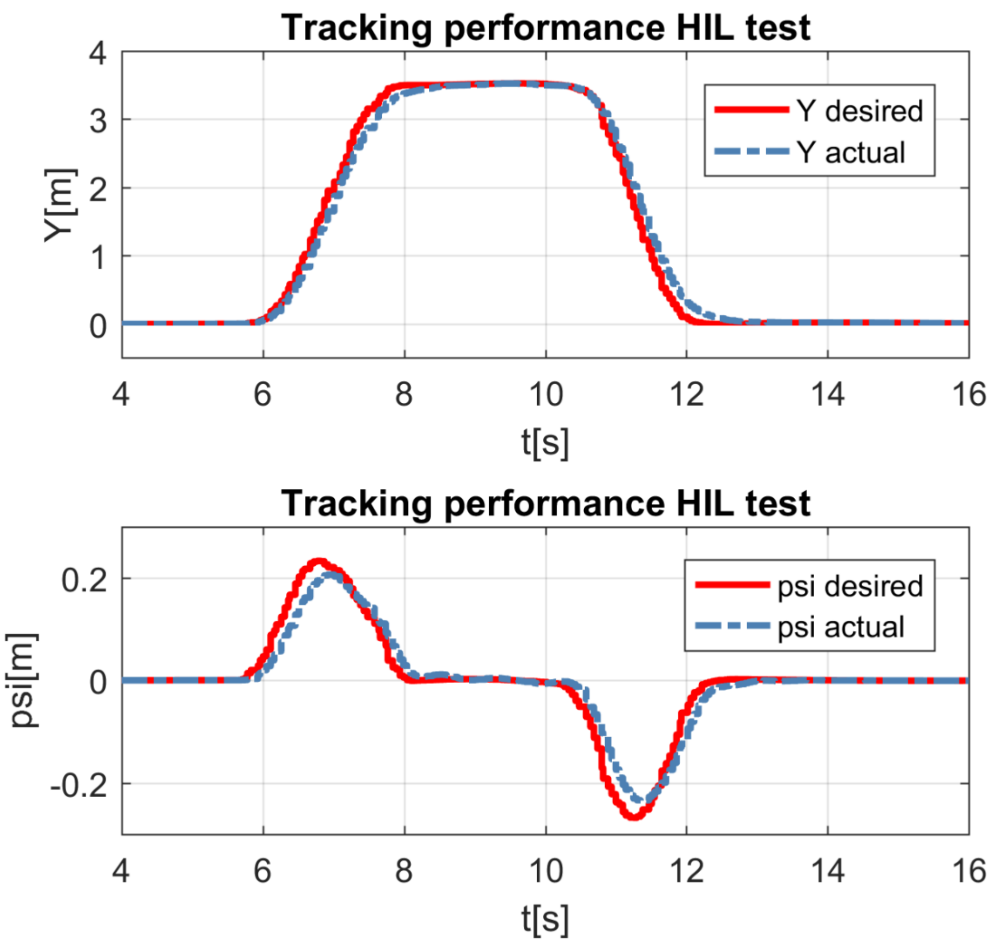
$$\min_{\mathbf{X}_k, \mathbf{U}_k} \quad \left\| \mathbf{MS}_{k+1} \right\|_F^2$$

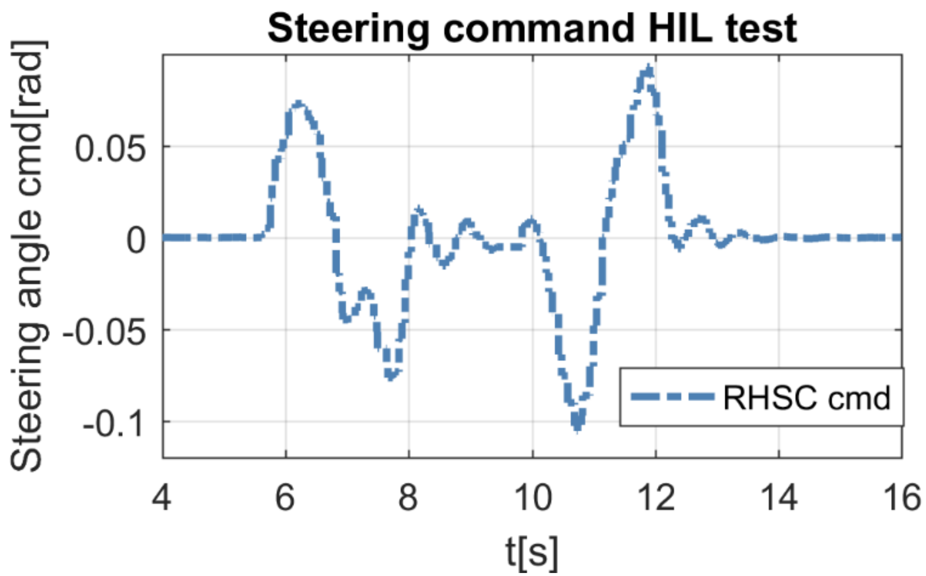
$$\begin{aligned}
s.t. \quad & \mathbf{s}_{i+1, j_{i+1}} = \mathcal{D}_{j_{i+1}}(\boldsymbol{\varepsilon}_{i+1, j_{i+1}}) \quad i = k, \dots, k+N-1, \quad j_i \in \{1, 2, \dots, 9\} \\
& \mathbf{x}_{i+1} = \mathbf{A}_{j_i}\mathbf{x}_i + \mathbf{B}_{j_i}\mathbf{u}_i + \mathbf{f}_{j_i}, \quad i = k, \dots, k+N_{j_i} \\
& \mathbf{y}_{i+1} = \mathbf{C}_{j_i}\mathbf{x}_i, \quad i = k, \dots, k+N_{j_i} \\
& |\mathbf{u}_k| \le \mathbf{u}_{\max}, \quad i = k, \dots, k+N_{j_i} \\
& |\Delta\mathbf{u}_k| \le \Delta\mathbf{u}_{\max}, \quad i = k, \dots, k+N_{j_i}
\end{aligned}$$

- **The optimization problem is solved by Yalmip in conjunction with Mosek solver. N=4, Ts = 0.05s, 3 modes.**

HIL results of RHSC real-time performance

✓ **Follow the predefined ISO double lane change trajectory (high miu)**

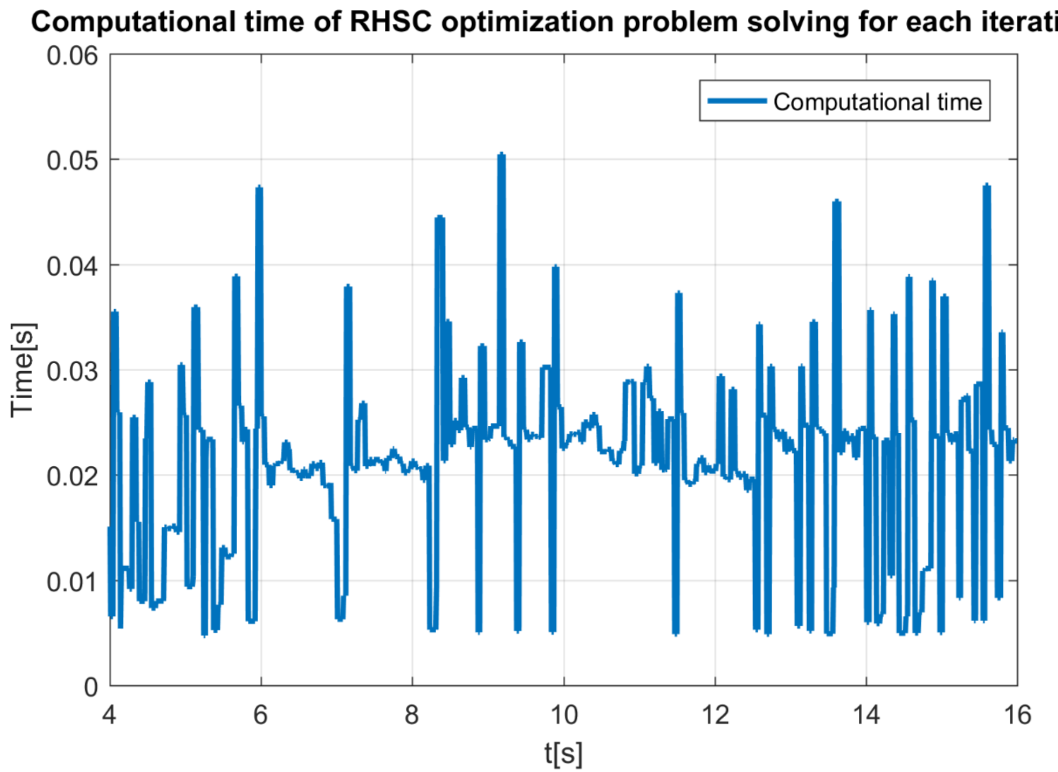




- **Tracking performances of Y and psi are good.**
- **As is on the high frictional road, there is no mode change happens.**

HIL results of RHSC real-time performance

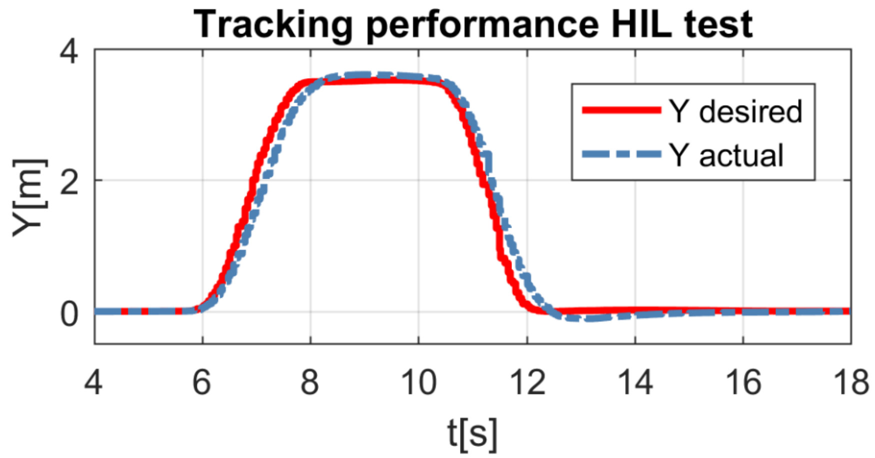
✓ **Computational time (high miu)**

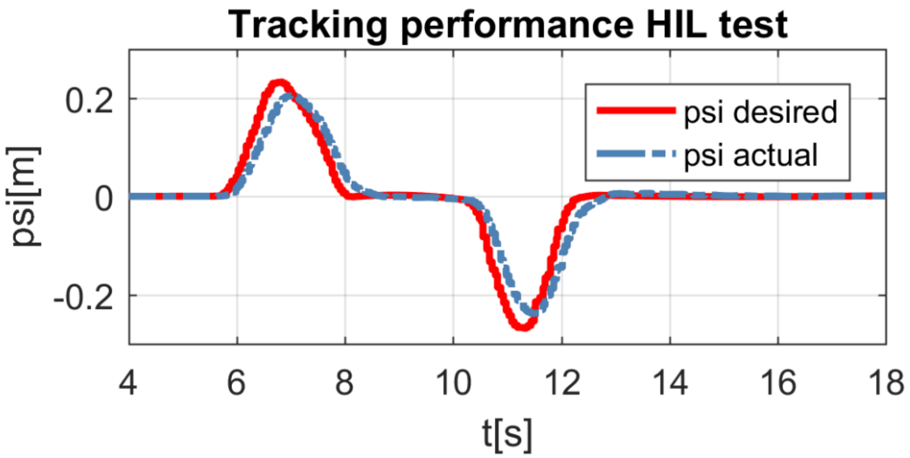


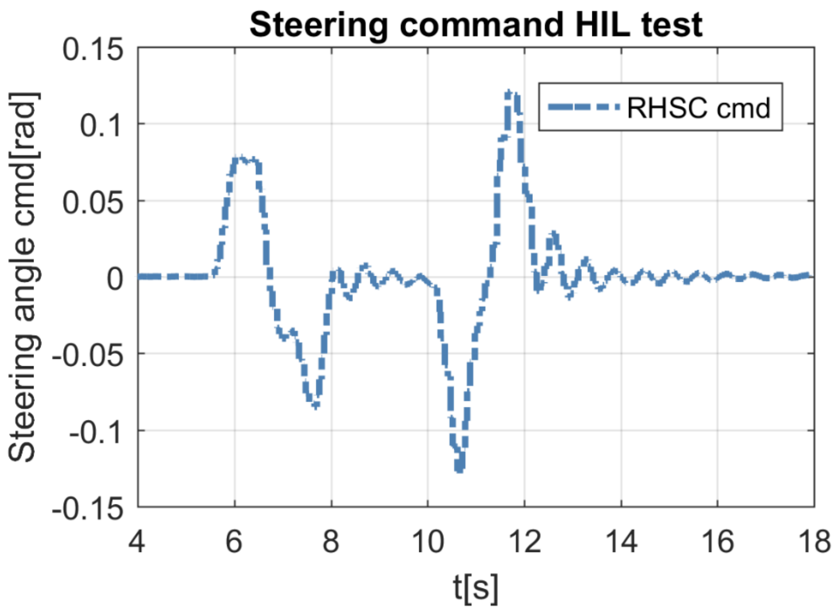
- **Mixed-integer programming is solvable on-line.**
- **It is to be noted that in this case there is no mode change happens, so it is "pseudo mixed-integer programming".**

HIL results of RHSC real-time performance

✓ **Follow the predefined ISO double lane change trajectory (low miu)**



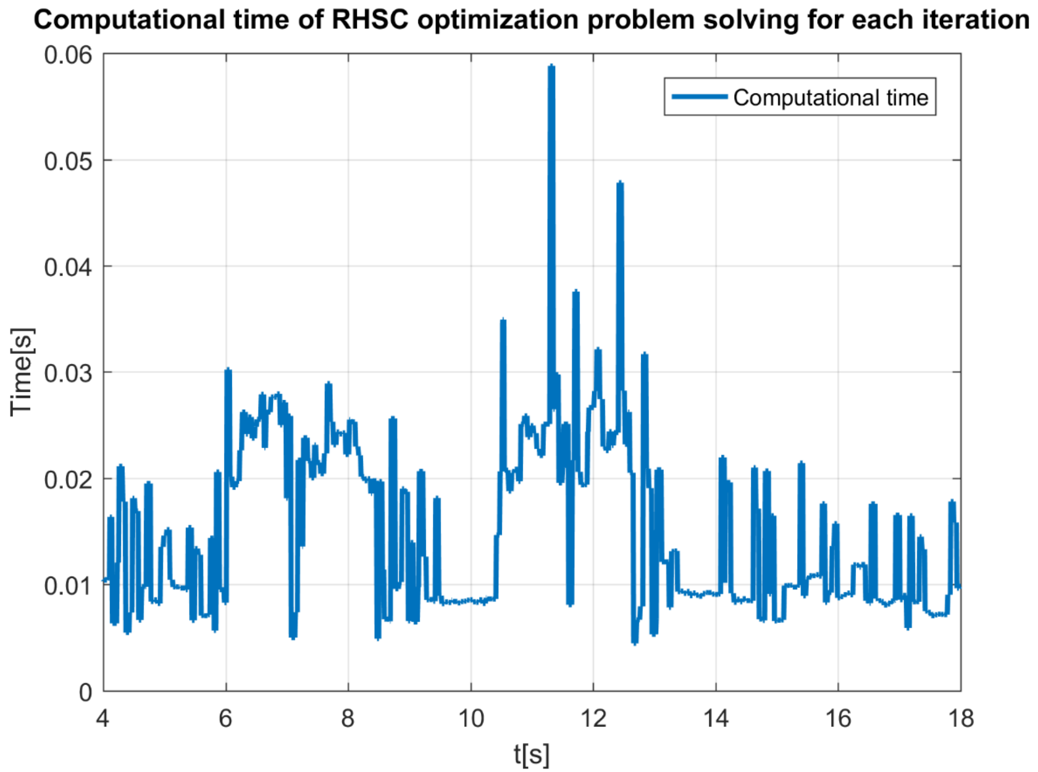




- **Tracking performances of Y and psi are good.**
- **As is on the low frictional road, there is mode change happens.**

HIL results of RHSC real-time performance

✓ **Computational time (low miu)**



- **Mixed-integer programming is solvable on-line.**
- **It is to be noted that in this case there is mode change happens, so it is real mixed-integer programming.**

Outlines

- ✓ **Introduction**
  - *Receding horizon sliding control (RHSC) for PWA systems*
  - *CAST architecture*
- ✓ **Robustness analysis of RHSC**
  - *Control performance with no model mismatch*
  - *Parametric and Dynamics uncertainties*
- ✓ **Validation of real-time performance of RHSC based on Hardware in loop tests**
- ✓ **Conclusions and interesting extensions**

Conclusions and interesting extensions

- CAST architecture is powerful and easy to use to rapidly validate the newly developed control algorithm, observer, sensor fusion algorithm…

- RHSC is promising control algorithm:
  - Compared with the conventional MPC, as the idea of a sliding surface is embedded, the design and tuning of desired error dynamics becomes easy. Output tracking problems are easier to be solved.
  - Compared with conventional SMC, system's constraints can be directly considered. Chattering problems are mitigated by the prediction scheme used in RHSC.

- Tests will be conducted on RC car. Real lane change, include decision making and path planning.