Hierarchical Model Predictive Control of an Active Seat for Improved Passenger Comfort

Daniel Sousa Schulman, ME PhD Candidate: dschul@umich.edu
Precision Systems Design Lab, University of Michigan
AERO740: Model Predictive Control - Final Project

***Abstract*—The development of Autonomous Vehicles (AV) has pushed for new technologies that are able to improve passenger comfort and productivity. One proposed solution to the problem of improving passenger comfort is an active seat that is able to compensate for the vehicle's motion by performing roll and pitch maneuvers. In order to control such active seat, it's necessary to implement a state-space controller that performs reference tracking, handles system constraints and updates at fast computational rates. To address that, a hierarchical MPC formulation is proposed with a two-layer control architecture and shared control law. This approach nearly matched the performance of a fast-sampled long horizon MPC controller, while maintaining an update rate compatible with real-time implementation. Considerations on the tuning of the hierarchical MPC controllers is provided, as well as the limitations of this application.**

I. Introduction

A. Motivation

Motion sickness is a condition that afflicts one in three US adults traveling in passenger vehicles [1]. Symptoms include sweating, nausea, retching, etc., and may vary considerably from person to person [2]–[10]. While the deleterious effects of motion sickness on passenger comfort has been known for decades [11, 12], this problem has received renewed attention in recent years [4, 7, 9, 13, 14] given the expected adoption of self-driving or autonomous vehicles (AV). Passengers experience motion sickness more acutely compared to the driver of a traditional vehicle, who is able to make subtle preemptive corrections because they anticipate the inertial consequences of their driving actions. [1, 13, 15, 16]. In an AV, every occupant is a passive passenger, and therefore the deleterious effects of motion sickness on comfort and productivity is expected to be significant [14, 17]–[20]. Productive activities such as web surfing, texting, typing, watching video content, etc, further increase the likelihood of motion sickness [21]–[24]. Pharmaceutical solutions such as Dramamine causes drowsiness, which does not help with productivity [25]–[27]. The National Highway Traffic Safety Administration (NHTSA) estimated that in 2014 Americans spent 6.9 billion hours stuck in traffic – time that could be spent at work or leisure [28].The Morgan Stanley Research group estimated an annual increase of USD 507B to the US economy if passengers can engage in productive work in an AV during their commute [29]. However, to tap into the unprecedented opportunity to reclaim time and productivity via AVs, motion sickness is a challenge that currently remains unsolved.

Motion sickness in a moving vehicle is the consequence of the inertial forces associated with frequent acceleration (e.g. speeding, braking, turning, and bumps) and the resulting sensory conflict it creates [2]–[4].). In a traditional (manually-driven) vehicle, the passenger experiences greater motion sickness compared to the driver, who is able to make subtle preemptive corrections because they anticipate the inertial consequences of their driving actions. The passenger, on the other hand, ends up passively reacting to these driving actions and associated inertial forces. For example, the driver who initiates a right turn intentionally leans into the turn, while the passenger is swung away from the turn. In an AV, since every occupant is a passive passenger, the adverse effects of motion sickness are expected to be significant [5, 6, 30].

B. Problem Statement

One way to neutralize the impact of inertial forces on the passenger is to artificially create the corrective motion a driver performs during a turn. In order to do so, it's possible to use an active seat that can tip and tilt in response to braking, acceleration and turn events. The goal of such active seat is to minimize the lateral acceleration perceived by the passenger as seen in Fig. 1. The control of this active seat is based on the roll and pitch angles, which can be determined based on the anticipated trajectory of the vehicle with respect to the world. The roll and pitch angles are a function of the actuation of two DC motors that govern the motion of the active seat, which behaves as a parallel mechanism, shown in Fig. 2. Hence, the system can be reduced to 4 states, which correspond to each motor's angular position and velocity, $x_k = \begin{bmatrix}\theta_{1,k} & \omega_{1,k} & \theta_{2,k} & \omega_{2,k}\end{bmatrix}^T$

The diagram in Fig. 3 shows vehicle motion can be used to determine the control trajectory for the motor angles. For the purposes of this project, the vehicle motion is obtained through IMU acceleration data which has been previously collected at MCity.

The system has physical constraints that limit motor current to under 5A and the motor angles to 50^o. As the plant is a multiple input multiple output system with constraints, an MPC formulation could be used for optimal control of a reference trajectory.

The motors need to be controlled with a sampling period of 0.01s, which can be hard to achieve using online optimization for long prediction horizons. The long computational times

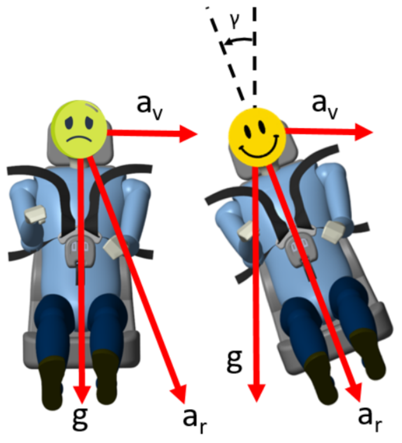


Fig. 1: Active Seat Principle of Operation. Example of roll (γ) actuation.

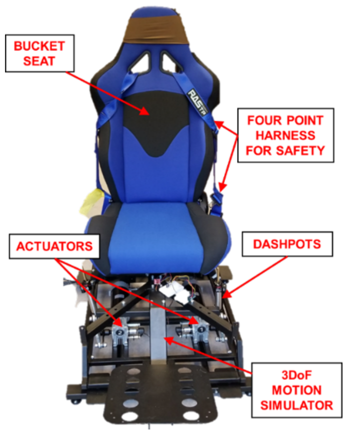


Fig. 2: Motion Simulator Hardware

required to run optimization algorithms can lead to poor performance and even instability of the system at hand.

C. Literature Review

The issue of real-time implementation of MPC has been addressed previously. Often, these implementations focus on pushing the boundaries of solver configurations. However, this is often limited by the hardware specifications of the processor running such algorithms. Researchers have recently explored new controller architectures that provide the prediction capabilities of MPC while providing real-time computations. In particular, a promising approach that hasn't been explored in depth is the concept of Hierarchical MPC. This approach consists of interfacing two or more MPC controllers with different sampling periods. The controller with the largest sampling period has a longer prediction horizon matching the problem requirements, while the controller with the smaller sampling period has a prediction horizon equal to the sampling period of the controller preceding it.

Koeln et al. [31] proposed a two-level hierarchical MPC formulation in which the output of the high level MPC is

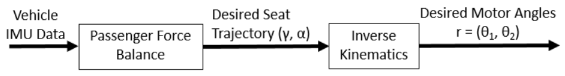

Vehicle IMU Data
Passenger Force Balance
Desired Seat Trajectory (γ, α)
Inverse Kinematics
Desired Motor Angles r = (θ1, θ2)

Fig. 3: Reference Trajectory Generation Block Diagram

an enforced constraint on the low level MPC based the predicted state at the following time step. While this approach is effective in providing strict guidance of the high level MPC, it can often be computationally intensive due to the need to reinitialize the problem formulation based on dynamic constraints. Further, the imposed constraints need to be carefully managed to ensure feasibility of the low level MPC. Further work done by Koeln et al. [32] showed the implementation of a hierarchical MPC formulation using constraint tightening as a way to ensure feasibility. The issue of feasibility of hierarchical MPC is a recurrent issue in the literature. In particular, Scattolini et al. [33] proposed a theoretical framework for a switching mechanism that ensures the feasibility of a two-level hierarchical MPC controller. However, such switching mechanism is still dependent on the dynamic allocation of constraints to the solver, which can lead to computationally intensive tasks.

D. Proposed Solution

In this paper, I propose a hierarchical MPC formulation and show its application in controlling an active seat in response to vehicle maneuvers. In such application, the control law is shared between the layers of the hierachical MPC formulation instead of being commanded by a single layer with dynamic constraints. This provides an advantageous setup for solvers that have long initialization times for constraint setup (cold start). This occurs for instance when using the MPT3 toolbox [34], which is the environment used to develop the project presented in this paper.

II. Methods

A. System Dynamics

The problem at hand can be modeled as two parallel DC motors, which behave according to the following state space continuous model.

$$\begin{aligned} x &= \begin{bmatrix}\theta_1 & \omega_1 & \theta_2 & \omega_2\end{bmatrix}^T \\ \dot{x} &= A_c x + B_c u \\ y &= C_c x + D_c u \end{aligned} \tag{1}$$

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{B_m}{J} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{B_m}{J} \end{bmatrix} B_c = \begin{bmatrix} 0 & 0 \\ \frac{KtKa}{J} & 0 \\ 0 & 0 \\ 0 & \frac{KtKa}{J} \end{bmatrix} \tag{2}$$

$$C_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} D_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3}$$

Note that for the purposes of this project, it's assumed perfect measurement of system states and no system disturbances. As a future line of work, this can be expanded through observer design and disturbance rejection formulations. However, this representation is sufficient to showcase capabilities of the hierarchical control architecture.

B. MPC Formulation

For the purposes of this project, let us consider a 2 level hierarchical MPC. Let the controller with larger sampling period be denoted as *"Upper MPC"* and the controller with smaller sampling period be denoted *"Lower MPC"*. In order to apply this framework, the system is sampled with two different sampling periods ($T_{sU} > T_{sL}$) for the Upper and Lower MPCs. This yields two state space systems defined as

$$\begin{aligned} x^U_{k+1} &= A_U x^U_k + B_U u^U_k \\ y^U_k &= C_U x^U_k + D_U u^U_k \\ T_s &= T_{sU} \end{aligned} \tag{4}$$

$$\begin{aligned} x^L_{k+1} &= A_L x^L_k + B_L u^L_k \\ y^L_k &= C_L x^L_k + D_L u^L_k \\ T_s &= T_{sL} \end{aligned} \tag{5}$$

Each of these systems is treated independently in the MPC formulation. It's necessary to establish an optimization problem for each of the controllers in the hierarchal MPC framework. Let i denote the controller instance with $i \,\epsilon\, \{U, L\}$ for this formulation. In order to provide reference tracking, an augmented state formulation was adapted from the Reference Tracking Formulation 2 from (Kolmanovsky, 2021) [35] as follows.

$$x^i_{aug|k} = \begin{bmatrix} \theta^i_{1|k} & \omega^i_{1|k} & \theta^i_{2|k} & \omega^i_{2|k} & u^i_{1|k} & u^i_{2|k} & r^i_{1|k} & r^i_{2|k} \end{bmatrix}^T \tag{6}$$

$$u^i_k = u^i_{k-1} + \delta u^i_k \tag{7}$$

This formulation was initially chosen as it allows for interfacing the hierarchical MPC with robust MPC formulations in literature. While the robust MPC was not employed in this project, it opens the possibility for this development in the future. The constrained MPC optimization problem then becomes

$$\begin{aligned} min_{\{\delta u^i\}} \quad & J^i_N = \sum_{k=0}^{N-1} {x^i_{aug|k}}^T Q x^i_{aug|k} + {\delta u^i_k}^T R \delta u^i_k \\ subject\ to \quad & x^i_{aug|k+1} = \begin{bmatrix} A_i & B_i & 0_{n_x \cdot n_r} \\ 0_{n_u \cdot n_x} & I_{n_u \cdot n_u} & 0_{n_u \cdot n_r} \\ 0_{n_r \cdot n_x} & 0_{n_r \cdot n_u} & I_{n_r \cdot n_r} \end{bmatrix} x^i_{aug|k} + \\ & \qquad + \begin{bmatrix} B_i \\ I_{n_u \cdot n_u} \\ I_{n_r \cdot n_u} \end{bmatrix} \delta u^i_k \\ & E^i_x = \begin{bmatrix} C_i & 0_{n_y \cdot n_u} & -I_{n_r \cdot n_r} \end{bmatrix} \\ & x_{min} < x^i_k < x_{max} \\ & u_{min} < u^i_k < u_{max} \\ & e^i_k = E_x x^i_{aug|k} \\ & Q^i = E^T_x Q^i_e E_x \end{aligned} \tag{8}$$

C. Hierachical MPC Architecture

Let the control effort of the Upper MPC be defined as $\bar{u_k}$ and the change in controller effort of the Lower MPC be defined as δu_k. Note $\bar{u_k}$ is only updated at every $k = j(T_{sU}/T_{sL}),\ j = \{1, 2, ...\}$. Thus, this allows for the Upper MPC to have a long prediction horizon, but only be updated at large sampling intervals. The Lower MPC has a shorter horizon but is updated at smaller sampling intervals. The horizon for the Lower MPC is chosen such that it matches the sampling period of the Upper MPC. Fig. 4 shows how the prediction horizon discretization of the Hierarchical MPC compared to that of the Classical MPC for the same horizon length.

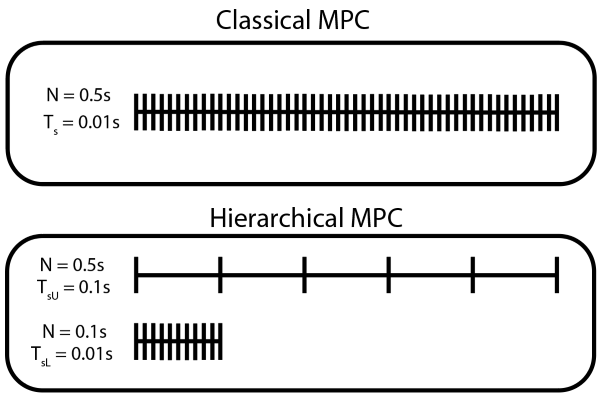


Fig. 4: Comparison of Prediction Horizons for Classical and Hierarchical MPC

The effective shared control law for the system sampled at T_{sL} can be defined as

$$u_k = \bar{u_k} + \delta u_k \tag{9}$$

The overall block diagram for the hierarchal MPC is shown in Fig. 5.

D. Simulation Setup

In order to test the model, vehicle IMU data was used to generate the desired seat trajectory in accordance with Fig. 3. A force balance equation, as a function of roll and pitch angles, was used to determine the desired motion simulator pose. The inverse kinematics of the seat were calculated to yield the desired motor angles. The two-level MPC models were defined according to their appropriate discretized model.

Note that a 0.5s prediction horizon is used in line with the nature of the data being used to generate the reference trajectory, which has maximum frequency of oscillation in the order of 1.5Hz. The 0.01s sampling period is in line with the software being used to control the DC motors, which runs at 100Hz.

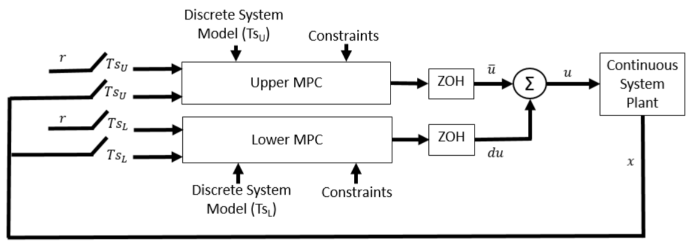


Fig. 5: Two-Level Hierarchical MPC Block Diagram

III. Results and Discussion

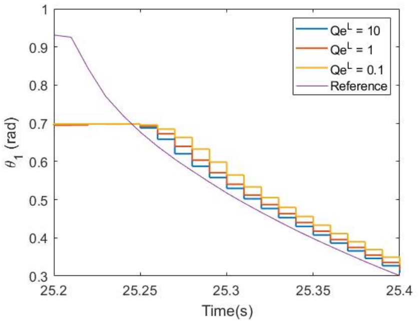
A. Cost Function Weight Analysis

In order to define the cost functions, the penalty weights for system states and control effort were for each of Upper and Lower MPCs as

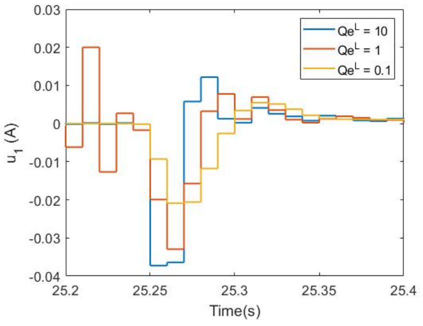
$$\begin{aligned} Q^L &= E_x^T Q_e^L E_x \\ Q^U &= E_x^T Q_e^U E_x \\ R^U &= R^L = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \end{aligned} \tag{10}$$

The coordination between cost weights of Upper and Lower MPC is a critical task in the design of hierarchical MPC controllers. In particular, it's important to consider how changing each of the costs affect the overall response of the hierarchical controller as an improper choice of these weights can lead to instability. This occurs due to the previously mentioned property that the feasibility of the Upper MPC does not ensure feasibility of the Lower MPC.In order to assess the impact of such weights on the design of the hierarchical MPC, a sensitivity analysis was performed evaluating the effect of each cost weight on the overall system response while mantaining all other parameters constant. In particular, responses of θ_1 and u_1 were analyzed in this study. These are shown in Fig. 6, 7 and 8.

As observed in Fig. 6, a value of $Q_e^L = 10$ provides the best tracking of the motion simulator motion without compromising the stability of the system or exceeding the performance parameters for the control effort. However, as seen in Fig. 6, the increase in Q_e^U from a value of 1 to 10



(a) Sensitivity of θ_1 to changes in Q_e^L



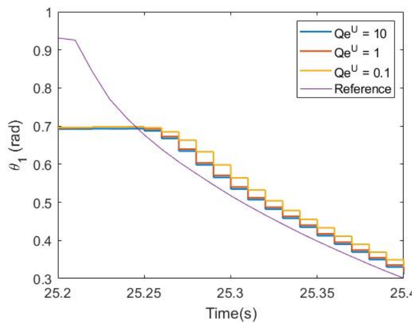
(b) Sensitivity of u_1 to changes in Q_e^L

Fig. 6: Sensitivity Analysis of Lower MPC Cost Weight with Respect to System Response

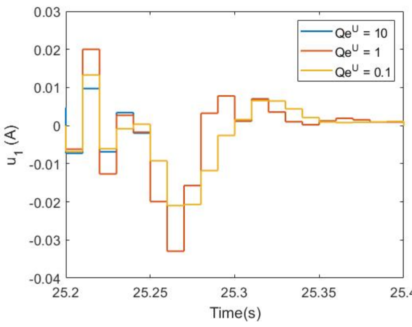
leads only to marginal improvements in state tracking. Thus, a more conservative approach in terms of maximum control effort was chosen by selecting $Q_e^U = 1$. Fig. 8 shows that a decrease in R from 0.01 to 0.001 provides better reference tracking at the cost of increasing control effort by a factor of almost 100%. The improvement in tracking by this change is not justified by the increase in control effort. Thus, the value chosen was $R = 0.01$. Note that while the values shown here guarantee feasibility of the system, this is not true for other values of Q_e^L. Thus, it's important to approach the design process conservatively, ensuring sufficient robustness of the system with respect to Q_e^L as to avoid potential issues during control implementation.

B. Reference Tracking

Fig. 9 shows the ability of the Hierarchical MPC to track a 1.5Hz sinusoidal input when the reference command exceeds the constraint limits. This is congruent with the highest frequency of motion observed in the IMU data. Hence, this demonstrates the system's ability to track the desired trajectory even when constraints become active. While the controller is able to enforce the reference trajectory, there's a lag in system response. In the future, this can be accounted for by using previously acquired IMU data to generate a feedforward trajectory for the MPC controller.

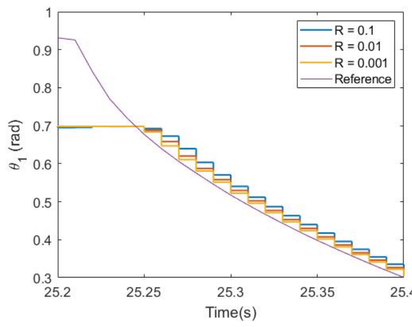


(a) Sensitivity of θ_1 to changes in R

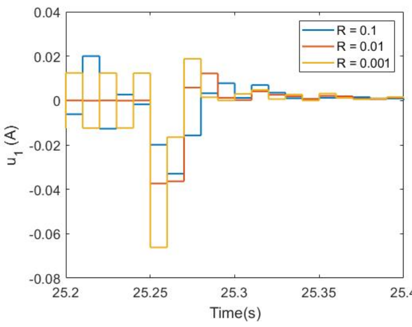


(b) Sensitivity of u_1 to changes in R

Fig. 7: Sensitivity Analysis of Upper MPC Cost Weight with Respect to System Response



(a) Sensitivity of θ_1 to changes in Q_e^U



(b) Sensitivity of u_1 to changes in Q_e^U

Fig. 8: Sensitivity Analysis of Control Effort Cost Weight with Respect to System Response

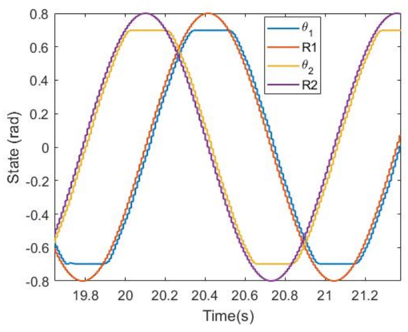


Fig. 9: Demonstration of Reference Tracking

C. Performance Comparison

Once the cost weights had been defined, the performance of the Hierarchical MPC was compared to that of Classical MPC approaches for the same cost function, which was defined as the Lower MPC cost function. Fig. 10 shows the performance tradeoff between the different controllers. While the Classical MPC with $T_s = 0.01s$ provides the lowest cost function, it also takes the longest to run. For a simulation length of 40s, this prohibitively slow as it would introduce system delays. The Classical MPC with $T_s = 0.1s$ has a much lower computational time, which is suitable for real-time applications. However, the cost function for this controller is much higher, indicating poor performance for the system at hand. In addition, the large sampling time can lead to instability in real-world applications. The Hierarchical MPC combines the advantages of each of the Classical MPC controllers. The cost function for the hierarchical MPC had a similar value to that of the Classical MPC with $T_s = 0.01s$. Further, the hierarchical MPC had a much lower computational time ($< 40s$), indicating it would be admissible for the system at hand. While the hierarchical MPC doesn't operate as fast as the Classical MPC with $T_s = 0.1s$, it provides a substantial improvement in the cost function while ensuring a sampling period that doesn't lead to system instability.

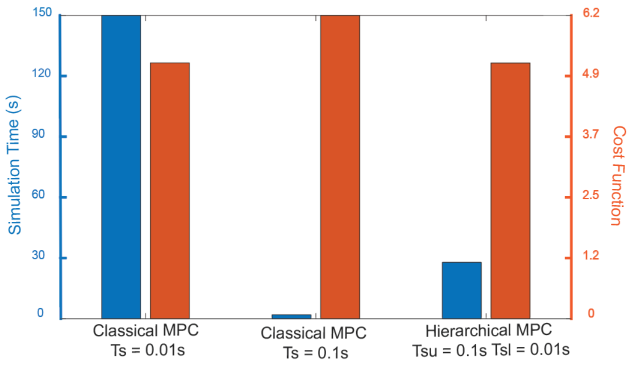


Fig. 10: MPC Controllers' Performance in Terms of Computational Time and Total Cost for a Horizon of N = 0.5s and Simulation Time of 40s.

D. Solution Limitations

As previously mentioned, one of the key limitations of this approach is that the Upper MPC feasibility does not enforce Lower MPC feasibility. It should be noted that the algorithm developed treats unfeasible solutions by returning $\delta u = [0, 0]^T$. Fig. 11 shows how the unfeasibility of Lower MPC translates to system response in the form of constraint violation ($\theta_{1max} < 0.698$). The Upper MPC enforces the constraints at every T_s^U sample interval, while the Lower MPC violates the constraint at the T_s^L intervals in between. One one T_s^U interval passes, the Upper MPC recovers the feasibility of the system if the system state is still within its feasibility region. This behavior only occurs when the constraints are active on the Upper MPC controller. One way to deal with this behavior is to tighten the constraints on the Upper MPC and tune such tightening according to the desired system performance.

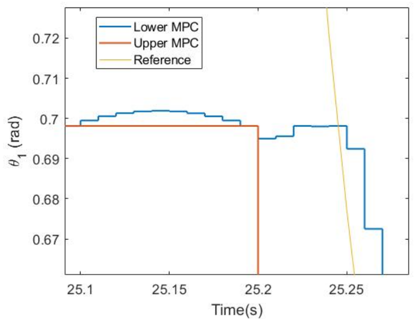


Fig. 11: Demonstration of Violation of Constraints by Lower MPC

IV. Conclusion

Overall, the hierarchical MPC formulation provided an effective solution to the real-time issue of controller implementation. It was able to match the cost performance of a fast-sampled long horizon controller, while providing a computational time compatible with real-time implementation. One of the challenges with such controller architecture is coordinating the cost weights between Lower and Upper MPC. However, once this is accounted for, the system provided satisfactory reference tracking with minimal lag. The issue of loss of Lower MPC feasibility is a challenge for this formulation, but this can be accounted for by monitoring the extent of constraint violations and enforcing constraint tightening on the Upper MPC accordingly.

While this project showed good performance with respect to reference tracking, future work to be done includes expanding the current formulation to account for disturbance rejection. The hierarchical MPC setup can be interfaced with robust MPC approaches in order to yield this characteristic. In addition to that, future work can be done on designing an observer for this system in an effort to account for sensor measurement noise.

References

[1] Genetics Home, R., 2019, “Motion Sickness,” Genet. Home Ref. [Online]. Available: https://ghr.nlm.nih.gov/condition/motion-sickness.

[2] Bles, W., Bos, J. E., de Graaf, B., Groen, E., and Wertheim, A. H., 1998, “Motion Sickness: Only One Provocative Conflict?,” Brain Res. Bull., 47(5), pp. 481–487.

[3] Bertolini, G., and Straumann, D., 2016, “Moving in a Moving World: A Review on Vestibular Motion Sickness,” Front. Neurol., 7.

[4] Green, P., 2016, “Motion Sickness and Concerns for Self-Driving Vehicles: A Literature Review,” p. 83.

[5] Iskander, J., Attia, M., Saleh, K., Nahavandi, D., Abobakr, A., Mohamed, S., Asadi, H., Khosravi, A., Lim, C. P., and Hossny, M., 2019, “From Car Sickness to Autonomous Car Sickness: A Review,” Transp. Res. Part F Traffic Psychol. Behav., 62, pp. 716–726.

[6] Salter, S., Diels, C., Herriotts, P., Kanarachos, S., and Thake, D., 2019, “Motion Sickness in Automated Vehicles with Forward and Rearward Facing Seating Orientations,” Appl. Ergon., 78, pp. 54–61.

[7] Diels, C., and Bos, J. E., 2016, “Self-Driving Carsickness,” Appl. Ergon., 53, pp. 374–382.

[8] Lackner, J. R., 2014, “Motion Sickness: More than Nausea and Vomiting,” Exp. Brain Res., 232(8), pp. 2493–2510.

[9] Salter, S., Diels, C., Herriotts, P., Kanarachos, S., and Thake, D., 2019, “Model to Predict Motion Sickness within Autonomous Vehicles,” Proc. Inst. Mech. Eng. Part J. Automob. Eng., p. 0954407019879785.

[10] Jones, M. L. H., Le, V. C., Ebert, S. M., Sienko, K. H., Reed, M. P., and Sayer, J. R., 2019, “Motion Sickness in Passenger Vehicles during Test Track Operations,” Ergonomics, 62(10), pp. 1357–1371.

[11] McCauley, M. E., Royal, J. W., Wylie, C. D., O’Hanlon, J. F., and Mackie, R. R., 1976, Motion Sickness Incidence: Exploratory Studies of Habituation, Pitch and Roll, and the Refinement of a Mathematical Model:, Defense Technical Information Center, Fort Belvoir, VA.

[12] Claremont, CA, 1931, “The Psychology of Sea-Sickness,” Psyche (Stuttg.), 11, pp. 86–90.

[13] Bae, I., Moon, J., and Seo, J., 2019, “Toward a Comfortable Driving Experience for a Self-Driving Shuttle Bus,” Electronics, 8(9), p. 943.

[14] Smyth, J., Jennings, P., and Birrell, S., 2020, “Are You Sitting Comfortably? How Current Self-Driving Car Concepts Overlook Motion Sickness, and the Impact It Has on Comfort and Productivity,” Advances in Human Factors of Transportation, N. Stanton, ed., Springer International Publishing, Cham, pp. 387–399.

[15] Rolnick, A., and Lubow, R. E., 1991, “Why Is the Driver Rarely Motion Sick? The Role of Controllability in Motion Sickness,” Ergonomics, 34(7), pp. 867–879.

[16] Kraft, B. A., 21 Jun 17, “Motion Sickness (Travel Sickness): Causes, Remedies, and Symptoms,” Med. News Today [Online]. Available: https://www.medicalnewstoday.com/articles/176198.php.

[17] Ingraham, C., 2019, “The Astonishing Human Potential Wasted on Commutes,” Wash. Post [Online]. Available: https://www.washingtonpost.com/news/wonk/wp/2016/02/25/how-much-of-your-life-youre-wasting-on-your-commute/.

[18] MacKay, J., 2018, “Productivity in 2017: What We Learned from Analyzing 225 Million Hours - RescueTime,” RescueTime Blog [Online]. Available: https://blog.rescuetime.com/225-million-hours-productivity/.

[19] Parida, S., Mallavarapu, S., Franz, M., and Abanteriba, S., 2019, “A Literature Review of Seating and Body Angles for Non-Driving Secondary Activities in Autonomous Driving Vehicles,” Advances in Human Aspects of Transportation, N. Stanton, ed., Springer International Publishing, Cham, pp. 398–409.

[20] Bissell, D., Birtchnell, T., Elliott, A., and Hsu, E. L., 2020, “Autonomous Automobilities: The Social Impacts of Driverless Vehicles,” Curr. Sociol., 68(1), pp. 116–134.

[21] 2018, “Self-Driving Cars Might Make People Sick to Their Stomachs,” Automot. News [Online]. Available: https://www.autonews.com/article/20181126/SHIFT/181129997/self-driving-cars-might-make-people-sick-to-their-stomachs.

[22] Sivak, M., Sivak, M., and Schoettle, B., 2015, Motion Sickness in Self-Driving Vehicles.

[23] Ekchian, J., Graves, W., Anderson, Z., Giovanardi, M., Godwin, O., Kaplan, J., Ventura, J., Lackner, J. R., and DiZio, P., 2016, “A High-Bandwidth Active Suspension for Motion Sickness Mitigation in Autonomous Vehicles,” pp. 2016-01–1555.

[24] Anderson, J., “Researchers Say We Should Start Counting Commute Time as Work. Here’s a Better Idea,” Quartz Work [Online]. Available: https://qz.com/work/1374452/how-should-i-use-my-commute-to-be-most-productive/.

[25] Graybiel, A., 1978, “Antimotion Sickness Remedy.”.

[26] Koch, A., Cascorbi, I., Westhofen, M., Dafotakis, M., Klapa, S., and Peter Kuhtz-Buschbeck, J., 2018, “The Neurophysiology and Treatment of Motion Sickness,” Dtsch. Ärztebl. Int., 115(41), pp. 687–996.

[27] Glaser, E. M., 1959, “Prevention and Treatment of Motion Sickness,” Proc. R. Soc. Med., 52(11), pp. 965–972.

[28] Lynberg, M., 2017, “Automated Vehicles for Safety,” NHTSA [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[29] 2013, “Morgan Stanley: Autonomous Cars Self-Driving the New Auto Industry Paradigm,” Morgan Stanley.

[30] Diels, C., 2014, “Will Autonomous Vehicles Make Us Sick?,” Contemporary Ergonomics and Human Factors 2014, S. Sharples, and S. Shorrock, eds., Taylor amp; Francis, pp. 301–307.

[31] Justin P. Koeln and Andrew G. Alleyne. Two-level hierarchical mission-based model predictive control. In *2018 Annual American Control Conference (ACC)*, pages 2332–2337, 2018.

[32] Justin Koeln, Vignesh Raghuraman, and Brandon Hencey. Vertical hierarchical mpc for constrained linear systems. *Automatica*, 113:108817, 2020.

[33] Riccardo Scattolini and Patrizio Colaneri. Hierarchical model predictive control. In *2007 46th IEEE Conference on Decision and Control*, pages 4803–4808, 2007.

[34] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. http://control.ee.ethz.ch/ mpt.

[35] Ilya Kolmanovsky. AERO740: Lecture Notes in MPC, 2021, University of Michigan.

Appendix

A. Code

See following pages

```
classdef Data_handle
    methods(Static)
        function [data] = dataCollect(filename)
            % Collecting Data
            delimiterIn = '\t';
            headerlinesIn = 14;
            A = importdata(filename,delimiterIn,headerlinesIn);

            rawTimeStamp = A.data(:,1);
            rawData = A.data(:,2);
            timeStamp = A.data(:,1)-A.data(1,1);
            %Normalizing data
            normTimeStamp = A.data(:,1)-A.data(1,1);
            normData = A.data(:,2)-mean(A.data(1:5,2));

            %Data Filtering
            filteredData = smoothdata(rawData,'gaussian',80);

            %Saving data to struct which is returned
            data.timeStamp = timeStamp;
            data.rawData = rawData;
            data.normTimeStamp = normTimeStamp;
            data.normData = normData;
            data.filteredData = filteredData;
        end

        function [h,v] = readtsv(dataDir)
            currentDir = pwd;
            myFiles = dir(fullfile(dataDir,'*.tsv'));
            fileNames = string({myFiles.name});
            cd(dataDir);
            % Head Data
            h.ax =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0008')==true));
            h.ay =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0009')==true));
            h.az =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0010')==true));
            h.wx =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0011')==true));
            h.wy =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0012')==true));
            h.wz =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0013')==true));

            %Vehicle Data
            v.ax =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0028')==true));
            v.ay =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0029')==true));
            v.az =
 Data_handle.dataCollect(fileNames(contains(fileNames,'C0030')==true));
```

```
            v.wx =
Data_handle.dataCollect(fileNames(contains(fileNames,'C0031')==true));
            v.wy =
Data_handle.dataCollect(fileNames(contains(fileNames,'C0032')==true));
            v.wz =
Data_handle.dataCollect(fileNames(contains(fileNames,'C0033')==true));
            cd(currentDir)
        end
        function [ax,ay,az,wx,wy,wz,timeStop] = setAcc(h)
            ax = timeseries(h.ax.filteredData, h.ax.timeStamp);
            ay = timeseries(h.ay.filteredData, h.ay.timeStamp);
            az = timeseries(h.az.filteredData, h.az.timeStamp);

            wx = timeseries(h.wx.filteredData, h.wx.timeStamp);
            wy = timeseries(h.wy.filteredData, h.wy.timeStamp);
            wz = timeseries(h.wz.filteredData, h.wz.timeStamp);
            timeStop = h.ax.timeStamp(length(h.ax.timeStamp)-1);
        end

        function [ax,ay,az,wx,wy,wz,timeStop] = setVehAcc(v)
            ax = timeseries(v.ax.filteredData, v.ax.timeStamp);
            ay = timeseries(v.ay.filteredData, v.ay.timeStamp);
            az = timeseries(v.az.filteredData, v.az.timeStamp);

            wx = timeseries(v.wx.filteredData, v.wx.timeStamp);
            wy = timeseries(v.wy.filteredData, v.wy.timeStamp);
            wz = timeseries(v.wz.filteredData, v.wz.timeStamp);
            timeStop = v.ax.timeStamp(length(v.ax.timeStamp)-1);
        end

        function [ax_vis,ay_vis,az_vis,wx_vis,wy_vis,wz_vis] =
setVisualAcc(conflict,ax,ay,az,wx,wy,wz)
            switch conflict
                case 1
                    ax_vis =
timeseries(zeros(1,length(ax.Time)),zeros(1,length(ax.Time)));
                    ay_vis =
timeseries(zeros(1,length(ax.Time)),zeros(1,length(ax.Time)));
                    az_vis =
timeseries(zeros(1,length(ax.Time)),zeros(1,length(ax.Time)));
                    wx_vis =
timeseries(zeros(1,length(ax.Time)),zeros(1,length(ax.Time)));
                    wy_vis =
timeseries(zeros(1,length(ax.Time)),zeros(1,length(ax.Time)));
                    wz_vis =
timeseries(zeros(1,length(ax.Time)),zeros(1,length(ax.Time)));
                case 0
                    ax_vis = ax;
                    ay_vis = ay;
                    az_vis = az;
                    wx_vis = wx;
                    wy_vis = wy;
                    wz_vis = wz;
            end
        end



    end
end
```

```
ans = 

  Data_handle with no properties.
```

Published with MATLAB® R2021a

Table of Contents

```
%clear all
%clc
```

Input IMU Data

```
dir = 'C:\Users\dschul\Dropbox (University of Michigan)\ENGIN-PREACT\2
 - Simulation\DS_Copy\UMTRI Mcity Data\';
subjects_mcity = ['MS019_FN'];
subject_folder = subjects_mcity(1,:);

subject = subject_folder(1:5);
condition = subject_folder(7:8);

dataDir = [dir,subject_folder];
%[h,v] = Data_handle.readtsv(dataDir);
filename_h = strcat(subject_folder,'_h','.mat');
filename_v = strcat(subject_folder,'_v','.mat');

load(filename_h,'h')
load(filename_v,'v')

[ax,ay,az,wx,wy,wz,timeStop] = Data_handle.setVehAcc(v);
```

Correct for shift in IMU coordinate frame

```
k = [mean(ax); mean(ay); mean(az)];
g = [0;0;-9.8];

k_norm = k/norm(k);
g_norm = g/norm(g);
```

```
angle = acos(dot( k_norm, g_norm ));
axis = cross( k_norm, g_norm );
axis = axis/norm(axis);

c = cos(angle);
s = sin(angle);
t = 1 - c;
x = axis(1);
y = axis(2);
z = axis(3);
R = [[c+x*x*t , x*y*t-z*s, x*z*t+y*s]; [y*x*t+z*s, c+y*y*t, y*z*t-
x*s]; [z*x*t-y*s, z*y*t+x*s, c+z*z*t]];

adata = [ax.Data(:)'; ay.Data(:)'; az.Data(:)'];
adata = R*adata;

ax.Data = adata(1,:);
ay.Data = adata(2,:);
az.Data = -adata(3,:);

wx = wx - mean(wx);
wy = wy - mean(wy);
wz = wz - mean(wz);
```

Calculate desired trajectory - alpha and gamma to motor angles

```
%These values enforce experienced accel coincident with passenger y
 axis (sagital)

alpha = atan(ay.Data./9.8);
gamma = atan(ax.Data./9.8);
theta1_d = zeros(1,length(alpha)); %desired motor angle 1 (r1)
theta2_d = zeros(1,length(alpha)); % desired motor angle 2 (r2)

%Find motor angles based on inverse kinematic equations

G = [0 0 0];                %Origin
M = [0.23 0.29 0.07];       %M Measurements [meter]
S = [0.23 0.37 0];          %S Measurements [meter]
l_gl = 0;
l_mp = 0.06;
l_ps = 0.07;
m = 1;

for i = 1:length(alpha)

mu1 = S(1)*cos(alpha(i)) + S(3)*sin(alpha(i)) - M(1);
mu2 = -S(1)*cos(alpha(i)) + S(3)*sin(alpha(i)) + M(1);

sigma1 = S(1)*sin(alpha(i))*sin(gamma(i)) + S(2)*cos(gamma(i)) -
 S(3)*cos(alpha(i))*sin(gamma(i)) - M(2);
```

```
sigma2 = -S(1)*sin(alpha(i))*cos(gamma(i)) + S(2)*sin(gamma(i)) +
 S(3)*cos(alpha(i))*cos(gamma(i)) + M(3);
sigma3 = -S(1)*sin(alpha(i))*sin(gamma(i)) + S(2)*cos(gamma(i)) -
 S(3)*cos(alpha(i))*sin(gamma(i)) - M(2);
sigma4 = S(1)*sin(alpha(i))*cos(gamma(i)) + S(2)*sin(gamma(i)) +
 S(3)*cos(alpha(i))*cos(gamma(i)) + M(3);

a1 = sigma1;
a2 = sigma3;

b1 = sigma2;
b2 = sigma4;

c1 = ((l_ps^2) - (mu1^2) - (sigma1^2) - (sigma2^2) - (l_mp^2))/
(-2*l_mp);
c2 = ((l_ps^2) - (mu2^2) - (sigma3^2) - (sigma4^2) - (l_mp^2))/
(-2*l_mp);

phi1 = atan2(a1,b1);
phi2 = atan2(a2,b2);

r1 = sqrt((a1^2) + (b1^2));
r2 = sqrt((a2^2) + (b2^2));

theta1_d(i) = real(asin(c1/r1) - phi1);
theta2_d(i) = real(asin(c2/r2) - phi2);
end
```

Define continuous system dynamics (command voltage to motor angle)

```
J = 8.5e-6;
Bm = 4.2e-6;
Kt = 0.022945;
Ka = 2;

A = [0 1 0 0; 0 -Bm/J 0 0; 0 0 0 1; 0 0 0 -Bm/J];
B = [0 0; Kt*Ka/J 0; 0 0; 0 Kt*Ka/J];
C = [1 0 0 0; 0 0 1 0];
D = zeros(2,2);

sys_c = ss(A,B,C,D);

Ts =  0.01;

sys_d_fast = c2d(sys_c,Ts);

[Ad,Bd,Cd,Dd] = ssdata(sys_d_fast);

nx = 4;
nu = 2;
nr = 2;
```

```
ny = 2;
```

Augment States for reference tracking formulation 2

```
%x = [theta1; omega1; theta2; omega2; u1; u2; r1; r2]

Ax = [Ad                Bd           zeros(nx,nr)
      zeros(nu,nx)   eye(nu)         zeros(nu,nr)
      zeros(nr,nx)   zeros(nr,nu)       eye(nr)];

Bx = [Bd;eye(nu);zeros(nr,nu)];

Ex = [Cd zeros(ny, nu) -eye(nr)];

Cx = eye(8);

Dx = zeros(8,2);

model = LTISystem('A', Ax , 'B', Bx, 'C', Cx, 'D', Dx, 'Ts', Ts);
```

Define constraints

```
% Control constraints
    model.u.min = [-1,-1]; %+-5V each motor to account for current
 limit (driver gain = 2);
    model.u.max= -model.u.min;

% State constraints
    model.x.min=[-deg2rad(40), -inf, -deg2rad(40), -inf, -5, -5, -inf,
 -inf];
    %model.x.min=[-inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf];
    model.x.max=-model.x.min;

% Output constraints
    model.y.min = [-deg2rad(40), -inf, -deg2rad(40), -inf, -5, -5, -
inf, -inf];
    %model.y.min=[-inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf];
    model.y.max = -model.y.min;
```

Define penalties

```
Qy = 0.1;
Q = Ex'*Qy*Ex;
R = diag([0.1, 0.1]);
model.u.penalty = QuadFunction(R);
model.x.penalty = QuadFunction(Q);
%model.x.with('terminalPenalty');
%model.x.terminalPenalty = QuadFunction(Q/2);
```

Slow Controller M=1

```
Ts_slow =  0.1;

sys_d_slow = c2d(sys_c,Ts_slow);

[Ad_s,Bd_s,Cd_s,Dd_s] = ssdata(sys_d_slow);

Ax_s = [Ad_s                Bd_s          zeros(nx,nr)
      zeros(nu,nx)   eye(nu)        zeros(nu,nr)
      zeros(nr,nx)   zeros(nr,nu)      eye(nr)];

Bx_s = [Bd_s;eye(nu);zeros(nr,nu)];

Ex_s = [Cd_s zeros(ny, nu) -eye(nr)];

Cx_s = eye(8);

Dx_s = zeros(8,2);

model_s = LTISystem('A', Ax_s , 'B', Bx_s, 'C', Cx_s, 'D', Dx_s, 'Ts',
 Ts_slow);

% Control constraints
    model_s.u.min= [-1,-1]; %+-5V each motor to account for current
 limit (driver gain = 2);
    model_s.u.max= -model.u.min;

% State constraints
    model_s.x.min=[-deg2rad(40), -inf, -deg2rad(40), -inf, -5, -5, -
inf, -inf];
    %model.x.min=[-inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf];
    model_s.x.max=-model.x.min;

% Output constraints
    model_s.y.min = [-deg2rad(40), -inf, -deg2rad(40), -inf, -5, -5, -
inf, -inf];
    %model.y.min=[-inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf];
    model_s.y.max = -model.y.min;

Qy = 1;
Qn = Ex'*Qy*Ex;
Rn = diag([0.1, 0.1]);
model_s.u.penalty = QuadFunction(Rn);
model_s.x.penalty = QuadFunction(Qn);
%model_s.x.with('terminalPenalty');
%model_s.x.terminalPenalty = QuadFunction(Q/2);

N_s = 0.5/Ts_slow;
ctrl_s = MPCController(model_s,N_s);
```

Fast Controller M=2

```
N = Ts_slow/Ts;
ctrl = MPCController(model,N);
```

Simulation setup

```
x0 = [0;0;0;0];
u0  = [0; 0];
r0 = [0; 0];

SimTime = floor(timeStop);

SimTime = 29;

Nsim = SimTime/Ts; %Simulation steps for entire IMU data ~20min

Nsim_s = SimTime/Ts_slow;

steps_ratio = Nsim/Nsim_s;

x = x0;
u = u0;
r =[0;0];
y =[0;0];
J = 0;

data.Time = ([1:Nsim]-1)*Ts;
data.Time_s = ([1:Nsim_s]-1)*Ts_slow;
data.X=zeros(nx,Nsim);
data.X_s=zeros(nx,Nsim_s);
data.U=zeros(nu,Nsim);
data.U_s=zeros(nu,Nsim_s);
data.R=zeros(nr,Nsim);
data.Y = zeros(2,Nsim);
data.Iter = zeros(1,Nsim);
data.J = zeros(1,Nsim);
```

Simulation

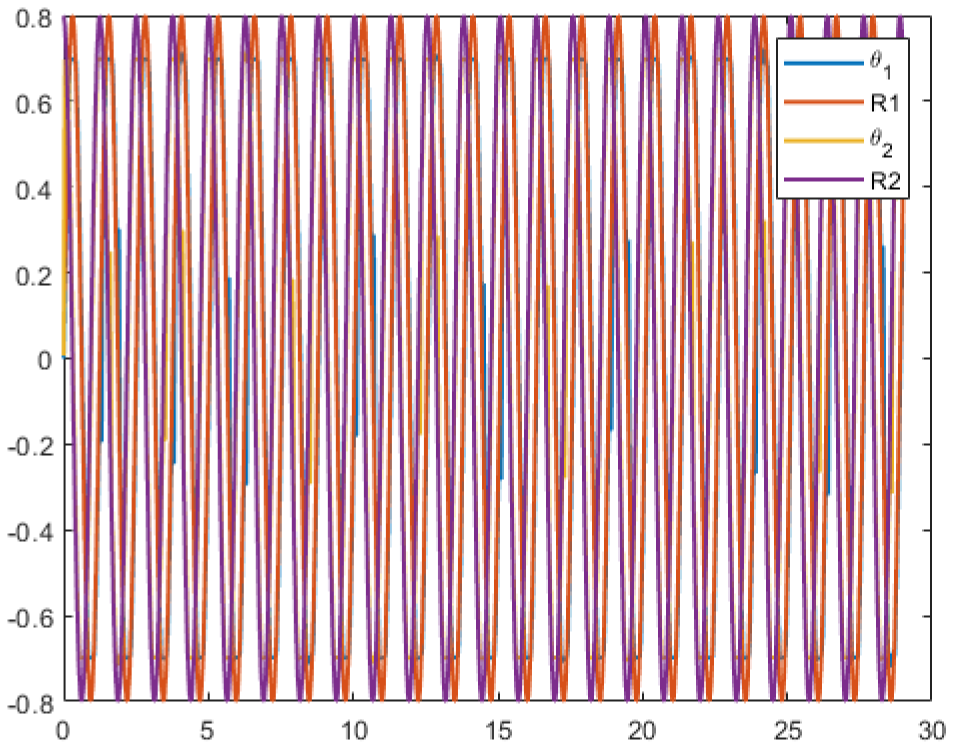
```
j = 0;
i = 1;
du_s = [0;0];
du_f = [0;0];
while j<Nsim_s
    i = 1;
    while i<=steps_ratio
    tic
        r(1) = theta1_d(i+j*steps_ratio); %desired motor angles
        r(2) = theta2_d(i+j*steps_ratio);
        r(1) = 0.8*sin((i+j*steps_ratio)/20); %desired motor angles
        r(2) = 0.8*cos((i+j*steps_ratio)/20);
        %try
            if i==1 %|| i==steps_ratio
                %r(1) = theta1_d(i+(j+1)*steps_ratio); %desired motor
 angles
                %r(2) = theta2_d(i+(j+1)*steps_ratio);
                %r(1) = sin((i+(j+1)*steps_ratio)/10); %desired motor
 angles
                %r(2) = cos((i+(j+1)*steps_ratio)/10);
                [du_s,feasible,openloop] =
 ctrl_s.evaluate([x(:);u;r]);
                if isnan(du_s)
                    du_s = [0;0];%-u;%-0.1*sign(u).*u;
                    %disp(j);
                end
                du_s = du_s; %change ratio
                %du = du_s;
                %u = u+du_s;
                du = du_s;
                data.U_s(:,j+1) = u+du_s;
                data.X_s(:,j+1) = Ad_s*x + Bd_s*(u+du_s); %predicted
 next state by slow controller
                %r(1) = theta1_d(i+j*steps_ratio); %desired motor
 angles
                %r(2) = theta2_d(i+j*steps_ratio);
                %r(1) = sin((i+j*steps_ratio)/10); %desired motor
 angles
                %r(2) = cos((i+j*steps_ratio)/10);
            else
                du_s = [0;0];
                [du,feasible,openloop] = ctrl.evaluate([x(:);u
+du_s;r]);
            end
        %catch
            %du = 0;
        %end
        %[du,feasible,openloop] = ctrl.evaluate([x(:);u;r]);

        if isnan(du)
            du = [0;0];
            %du = -u;%-0.1*sign(u).*u;
            %disp(j);
        end

        loop_time = toc;
        delay_steps = round((loop_time - Ts)/Ts);
        %disp(loop_time);

        u = u + du; % + du_s;
        x = Ad*x + Bd*u;
        y = Cd*x + Dd*u;
        xaug = [x; u; r];
        J = J + (xaug'*Q*xaug + du'*R*du);

        data.X(:,i+j*steps_ratio)=x;
```

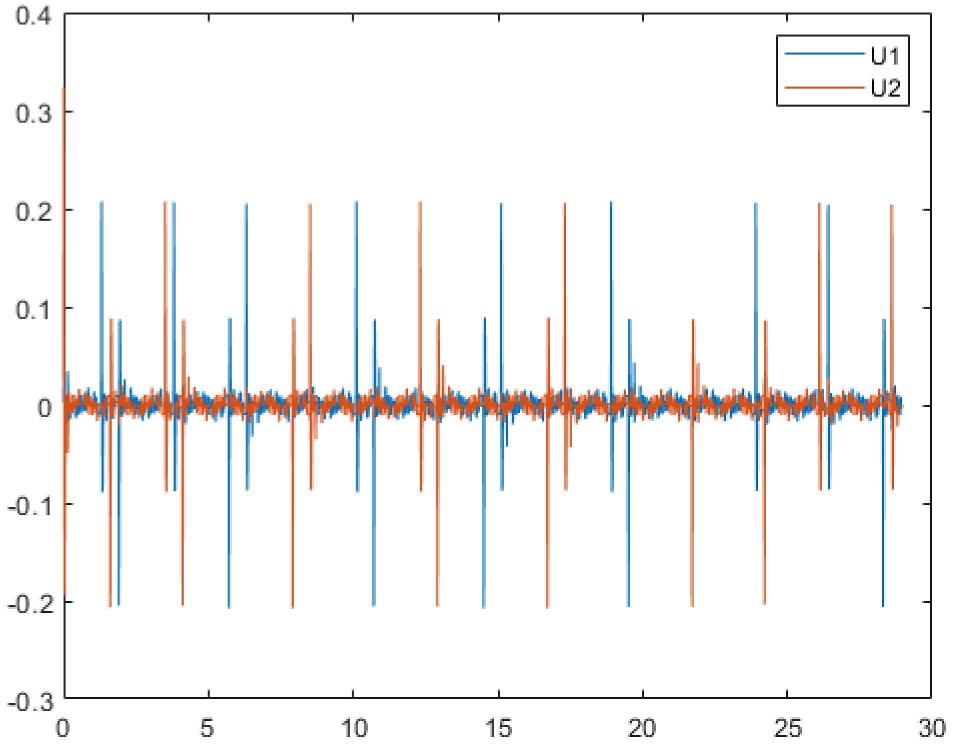
```
                data.U(:,i+j*steps_ratio)=u;
                data.R(:,i+j*steps_ratio)=r;
                data.Y(:,i+j*steps_ratio)=y;
                data.J(:,i+j*steps_ratio)=J;
                data.Iter(:,i+j*steps_ratio) = loop_time;

                i = i+1;

                if delay_steps>0 && i>10 && 0
                    for j = 1:delay_steps
                        x = Ad*x + Bd*u;
                        y = Cd*x + Dd*u;
                        data.X(:,i*j)=x;
                        data.U(:,i*j)=u;
                        data.R(:,i*j)=r;
                        data.Y(:,i*j)=y;
                        i = i+1;
                        disp('hi');
                    end
                end

            end

            j = j+1;
        end
```

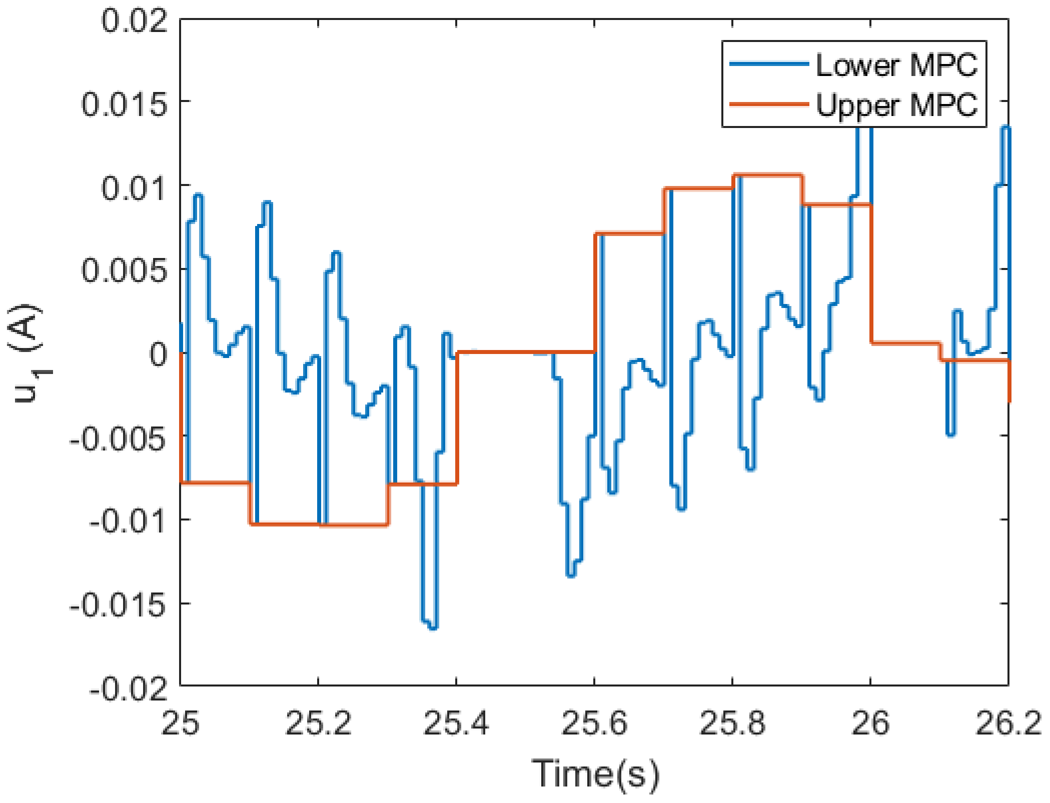
Plot

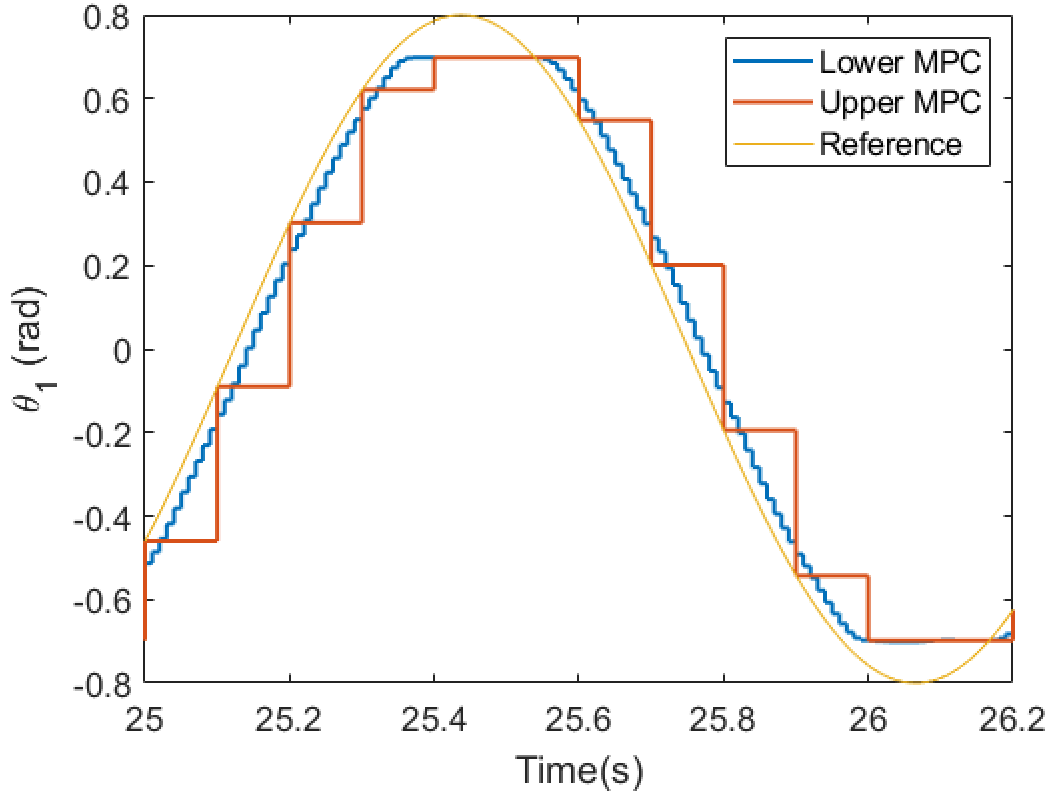
```
figure(2)
stairs(data.Time,data.X(1,:),'LineWidth',1.5);
hold on
stairs(data.Time,data.R(1,:),'LineWidth',1.5);
hold on
stairs(data.Time,data.X(3,:),'LineWidth',1.5);
hold on
stairs(data.Time,data.R(2,:),'LineWidth',1.5);
legend('\theta_1','R1','\theta_2','R2');

figure(3)
plot(data.Time,data.U(1,:));
hold on
plot(data.Time,data.U(2,:));
legend('U1','U2');

figure(4)
stairs(data.Time,data.U(1,:),'LineWidth',1.5);
hold on
stairs(data.Time_s,data.U_s(1,:),'LineWidth',1.5);
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('u_1 (A)');
legend('Lower MPC', 'Upper MPC')
```

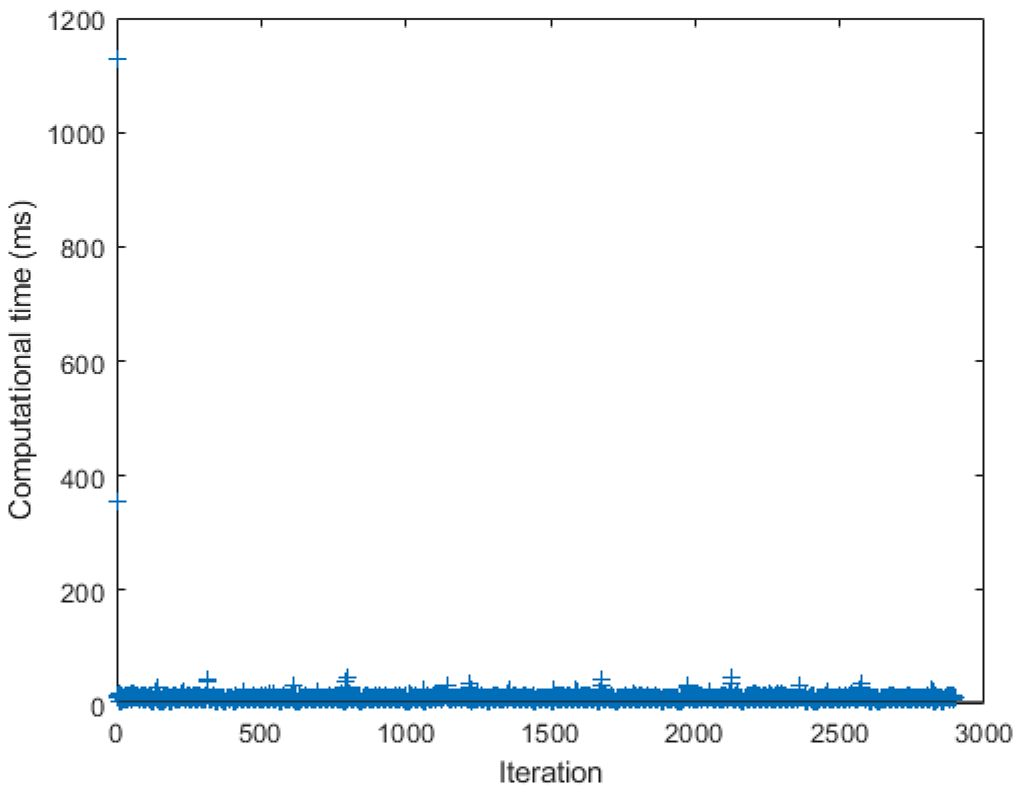
```
xlim([25,26.2])

figure(5)
stairs(data.Time,data.X(1,:),'LineWidth',1.5);
hold on
stairs(data.Time_s,data.X_s(1,:),'LineWidth',1.5);
hold on
plot(data.Time,data.R(1,:));
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('\theta_1 (rad)');
legend('Lower MPC', 'Upper MPC','Reference')
xlim([25,26.2])

figure(6)
plot([1:Nsim],data.Iter(1,:).*1000,'+');
xlabel('Iteration');
ylabel('Computational time (ms)')
yline(5);
```




U1
U2
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
5
10
15
20
25
30


Lower MPC
Upper MPC
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
u_1 (A)
25
25.2
25.4
25.6
25.8
26
26.2
Time(s)


Lower MPC
Upper MPC
Reference
θ_1 (rad)
Time(s)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
25
25.2
25.4
25.6
25.8
26
26.2


Computational time (ms)
Iteration
1200
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
3000

Save values

```
time_hier = sum(data.Iter(1,:));
cost_hier = data.J(end);
save('hier_qyL100_qyU10_R001_sim29.mat','data');
```

Find forward kinematics using lookup table. Ie, Recover alpha and gamma

```
%{
load('lookup_angles.mat')
for i = 1:length(data.X(1,:))
    theta1 = data.X(1,i);
    theta2 = data.X(3,i);
    poss_ind = intersect(find(abs(lookup_angles(:,1)-
theta1)<0.5),find(abs(lookup_angles(:,2)-theta2)<0.5));
    index = round(median(poss_ind));
    alpha_f = lookup_angles(index,3);
    gamma_f = lookup_angles(index,4);
end
%}
```

Published with MATLAB® R2021a

Table of Contents

```
%clear all
%clc
```

Input IMU Data

```
dir = 'C:\Users\dschul\Dropbox (University of Michigan)\ENGIN-PREACT\2
 - Simulation\Monica Jones Dataset\DS_Copy\UMTRI Mcity Data\';
subjects_mcity = ['MS019_FN'];
subject_folder = subjects_mcity(1,:);

subject = subject_folder(1:5);
condition = subject_folder(7:8);

dataDir = [dir,subject_folder];
%[h,v] = Data_handle.readtsv(dataDir);
filename_h = strcat(subject_folder,'_h','.mat');
filename_v = strcat(subject_folder,'_v','.mat');

load(filename_h,'h')
load(filename_v,'v')

[ax,ay,az,wx,wy,wz,timeStop] = Data_handle.setVehAcc(v);
```

Correct for shift in IMU coordinate frame

```
k = [mean(ax); mean(ay); mean(az)];
g = [0;0;-9.8];

k_norm = k/norm(k);
g_norm = g/norm(g);
```

```
angle = acos(dot( k_norm, g_norm ));
axis = cross( k_norm, g_norm );
axis = axis/norm(axis);

c = cos(angle);
s = sin(angle);
t = 1 - c;
x = axis(1);
y = axis(2);
z = axis(3);
R = [[c+x*x*t , x*y*t-z*s, x*z*t+y*s]; [y*x*t+z*s, c+y*y*t, y*z*t-
x*s]; [z*x*t-y*s, z*y*t+x*s, c+z*z*t]];

adata = [ax.Data(:)'; ay.Data(:)'; az.Data(:)'];
adata = R*adata;

ax.Data = adata(1,:);
ay.Data = adata(2,:);
az.Data = -adata(3,:);

wx = wx - mean(wx);
wy = wy - mean(wy);
wz = wz - mean(wz);
```

Calculate desired trajectory - alpha and gamma to motor angles

```
%These values enforce experienced accel coincident with passenger y
 axis (sagital)

alpha = atan(ay.Data./9.8);
gamma = atan(ax.Data./9.8);
theta1_d = zeros(1,length(alpha)); %desired motor angle 1 (r1)
theta2_d = zeros(1,length(alpha)); % desired motor angle 2 (r2)

%Find motor angles based on inverse kinematic equations

G = [0 0 0];                %Origin
M = [0.23 0.29 0.07];       %M Measurements [meter]
S = [0.23 0.37 0];          %S Measurements [meter]
l_gl = 0;
l_mp = 0.06;
l_ps = 0.07;
m = 1;

for i = 1:length(alpha)

mu1 = S(1)*cos(alpha(i)) + S(3)*sin(alpha(i)) - M(1);
mu2 = -S(1)*cos(alpha(i)) + S(3)*sin(alpha(i)) + M(1);

sigma1 = S(1)*sin(alpha(i))*sin(gamma(i)) + S(2)*cos(gamma(i)) -
 S(3)*cos(alpha(i))*sin(gamma(i)) - M(2);
```

```
sigma2 = -S(1)*sin(alpha(i))*cos(gamma(i)) + S(2)*sin(gamma(i)) +
 S(3)*cos(alpha(i))*cos(gamma(i)) + M(3);
sigma3 = -S(1)*sin(alpha(i))*sin(gamma(i)) + S(2)*cos(gamma(i)) -
 S(3)*cos(alpha(i))*sin(gamma(i)) - M(2);
sigma4 = S(1)*sin(alpha(i))*cos(gamma(i)) + S(2)*sin(gamma(i)) +
 S(3)*cos(alpha(i))*cos(gamma(i)) + M(3);

a1 = sigma1;
a2 = sigma3;

b1 = sigma2;
b2 = sigma4;

c1 = ((l_ps^2) - (mu1^2) - (sigma1^2) - (sigma2^2) - (l_mp^2))/
(-2*l_mp);
c2 = ((l_ps^2) - (mu2^2) - (sigma3^2) - (sigma4^2) - (l_mp^2))/
(-2*l_mp);

phi1 = atan2(a1,b1);
phi2 = atan2(a2,b2);

r1 = sqrt((a1^2) + (b1^2));
r2 = sqrt((a2^2) + (b2^2));

theta1_d(i) = real(asin(c1/r1) - phi1);
theta2_d(i) = real(asin(c2/r2) - phi2);
end
```

Define continuous system dynamics (command voltage to motor angle)

```
J = 8.5e-6;
Bm = 4.2e-6;
Kt = 0.022945;
Ka = 2;

A = [0 1 0 0; 0 -Bm/J 0 0; 0 0 0 1; 0 0 0 -Bm/J];
B = [0 0; Kt*Ka/J 0; 0 0; 0 Kt*Ka/J];
C = [1 0 0 0; 0 0 1 0];
D = zeros(2,2);

sys_c = ss(A,B,C,D);

Ts =  0.01;

sys_d_fast = c2d(sys_c,Ts);

[Ad,Bd,Cd,Dd] = ssdata(sys_d_fast);

nx = 4;
nu = 2;
nr = 2;
```

```
ny = 2;
```

System fast

```
Ts_f =  0.01;

sys_d_fast = c2d(sys_c,Ts_f);

[Ad_f,Bd_f,Cd_f,Dd_f] = ssdata(sys_d_fast);

nx = 4;
nu = 2;
nr = 2;
ny = 2;
```

Augment States for reference tracking formulation 2

```
%x = [theta1; omega1; theta2; omega2; u1; u2; r1; r2]

Ax = [Ad                Bd           zeros(nx,nr)
      zeros(nu,nx)   eye(nu)         zeros(nu,nr)
      zeros(nr,nx)   zeros(nr,nu)       eye(nr)];

Bx = [Bd;eye(nu);zeros(nr,nu)];

Ex = [Cd zeros(ny, nu) -eye(nr)];

Cx = eye(8);

Dx = zeros(8,2);

model = LTISystem('A', Ax , 'B', Bx, 'C', Cx, 'D', Dx, 'Ts', Ts);
```

Define constraints

```
% Control constraints
    model.u.min = [-1,-1]; %+-5V each motor to account for current
 limit (driver gain = 2);
    model.u.max= -model.u.min;

% State constraints
    model.x.min=[-deg2rad(40), -inf, -deg2rad(40), -inf, -5, -5, -inf,
 -inf];
    %model.x.min=[-inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf];
    model.x.max=-model.x.min;

% Output constraints
    model.y.min = [-deg2rad(40), -inf, -deg2rad(40), -inf, -5, -5, -
inf, -inf];
```

```
    %model.y.min=[-inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf];
    model.y.max = -model.y.min;
```

Define penalties

```
Qy = 0.1;
Q = Ex'*Qy*Ex;
R = diag([0.1, 0.1]);
model.u.penalty = QuadFunction(R);
model.x.penalty = QuadFunction(Q);
%model.x.with('terminalPenalty');
%model.x.terminalPenalty = QuadFunction(Q/2);
```

Fast Controller M=2

```
N = 0.5/Ts;
N = 5;
ratio = Ts/0.01;
ctrl = MPCController(model,N);
```

Simulation setup

```
x0 = [0;0;0;0];
u0  = [0; 0];
r0 = [0; 0];

SimTime = floor(timeStop);

SimTime = 29;

Nsim = SimTime/Ts; %Simulation steps for entire IMU data ~20min


x = x0;
u = u0;
r =[0;0];
y =[0;0];

data.Time = ([1:Nsim]-1)*Ts;
data.X=zeros(nx,Nsim);
data.U=zeros(nu,Nsim);
data.R=zeros(nr,Nsim);
data.Y = zeros(2,Nsim);
data.Iter = zeros(1,Nsim);
data.J = zeros(1,Nsim);
```

Simulation

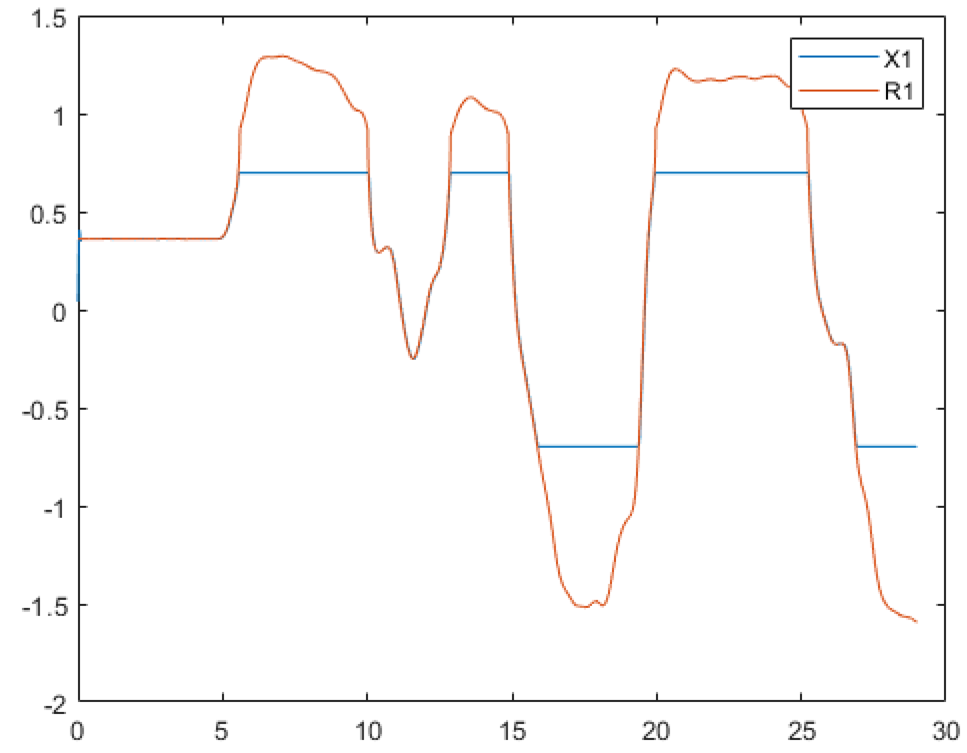
```
j = 0;
du_s = [0;0];
```

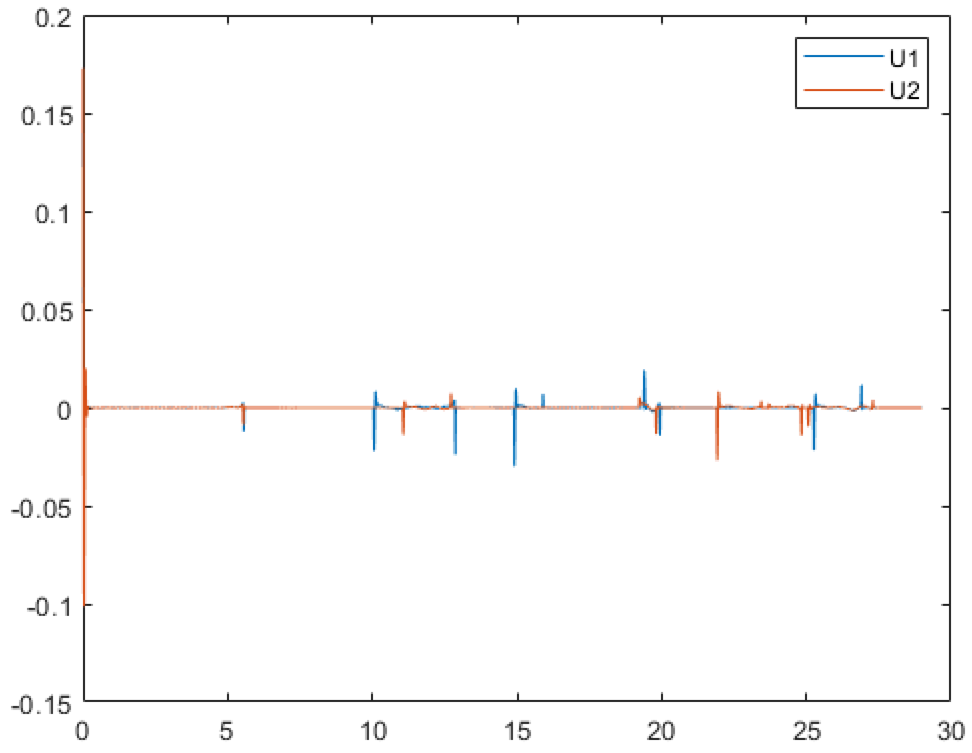
```
du_f = [0;0];
J = 0;

for i = 1:Nsim
    tic
    r(1) = theta1_d(i*ratio); %desired motor angles
    r(2) = theta2_d(i*ratio);

    [du,feasible,openloop] = ctrl.evaluate([x(:);u;r]);

    if isnan(du)
        du = [0;0];
        %du = -u;%-0.1*sign(u).*u;
    end

    loop_time = toc;
    delay_steps = round((loop_time - Ts)/Ts);
    %disp(loop_time);

    u = u + du;

    for j = 1:ratio
        x = Ad_f*x + Bd_f*u;
        y = Cd_f*x + Dd_f*u;
        xaug = [x; u; r];
        J = J + (xaug'*Q*xaug + du'*R*du);
    end


    %xaug = [x; u; r];
    %J = J + (xaug'*Q*xaug + du'*R*du)*ratio;

    data.X(:,i)=x;
    data.U(:,i)=u;
    data.R(:,i)=r;
    data.Y(:,i)=y;
    data.Iter(1,i) = loop_time;
    data.J(1,i) = J;



    if delay_steps>0 && i>10 && 0
        for j = 1:delay_steps
            x = Ad*x + Bd*u;
            y = Cd*x + Dd*u;
            data.X(:,i*j)=x;
            data.U(:,i*j)=u;
            data.R(:,i*j)=r;
            data.Y(:,i*j)=y;
            i = i+1;
            disp('hi');
        end
    end
```

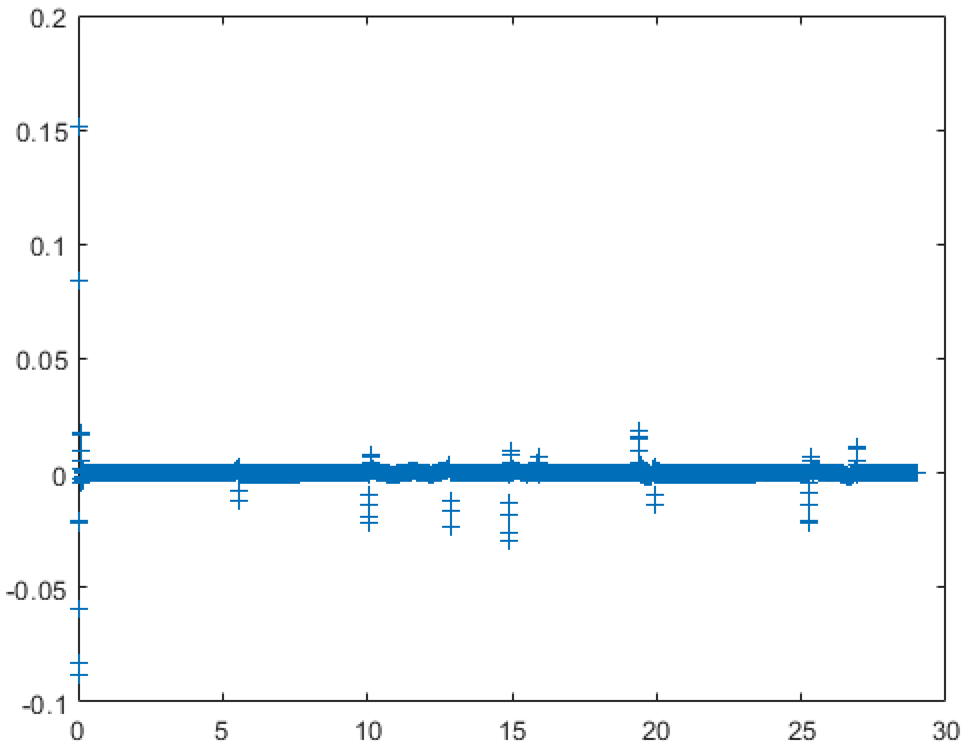
```
end
```

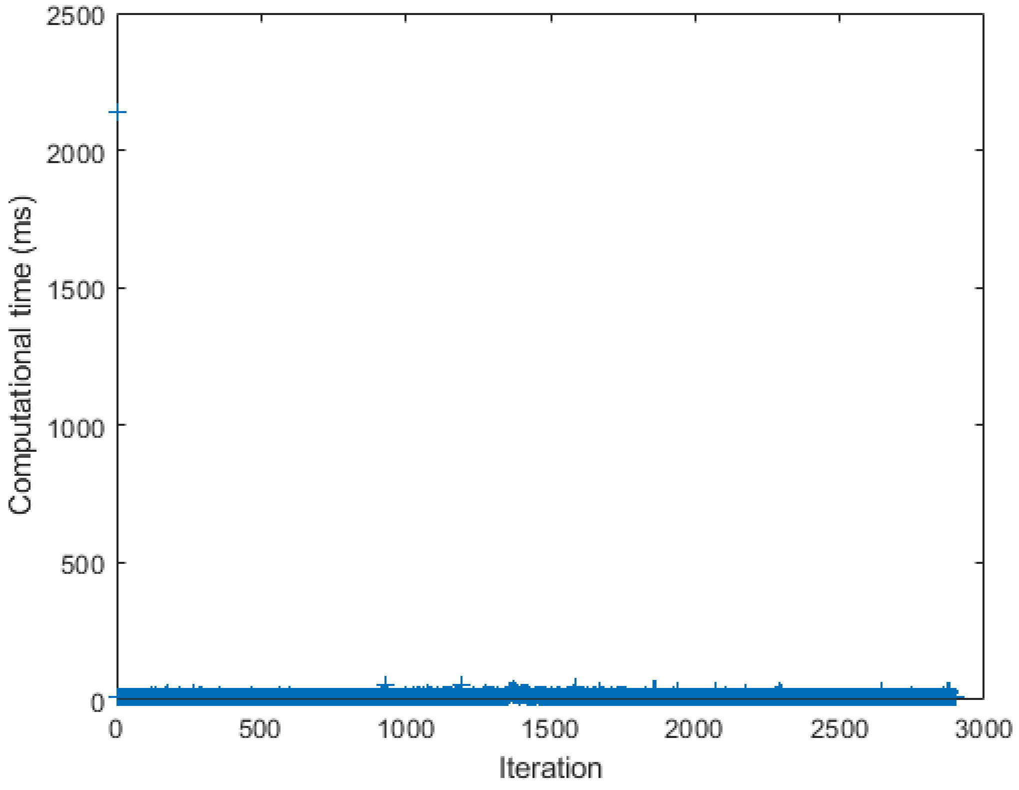
Plot

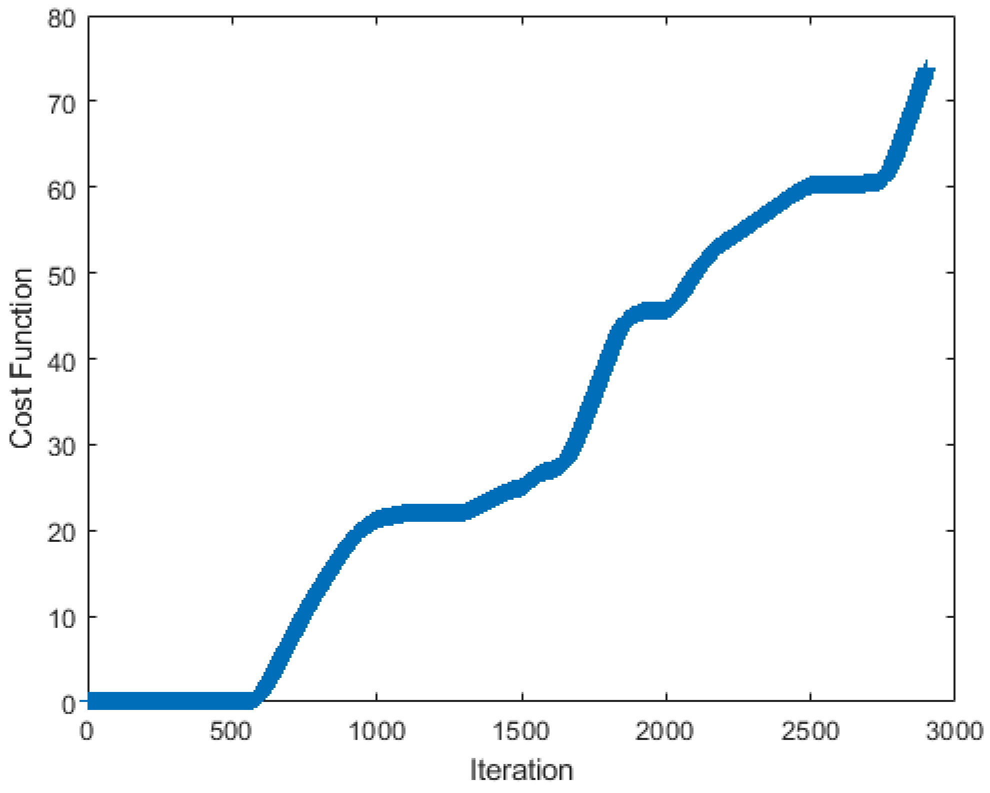
```
figure(2)
plot(data.Time,data.X(1,:));
hold on
plot(data.Time,data.R(1,:));
legend('X1','R1');

figure(3)
plot(data.Time,data.U(1,:));
hold on
plot(data.Time,data.U(2,:));
legend('U1','U2');

figure(4)
plot(data.Time,data.U(1,:),'+');
hold on
%plot(data.Time_s,data.U_s(1,:),'o');


figure(5)
plot([1:Nsim],data.Iter(1,:).*1000,'+');
xlabel('Iteration');
ylabel('Computational time (ms)')
yline(5);

figure(6)
plot([1:Nsim],data.J(1,:),'+');
xlabel('Iteration');
ylabel('Cost Function')
```


X1
R1


U1
U2


0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
5
10
15
20
25
30


2500
2000
1500
1000
500
0
Computational time (ms)
0
500
1000
1500
2000
2500
3000
Iteration



Save values

```
time_01_N5 = sum(data.Iter(1,:));
cost_01_N5 = data.J(end);
```

Find forward kinematics using lookup table. Ie, Recover alpha and gamma

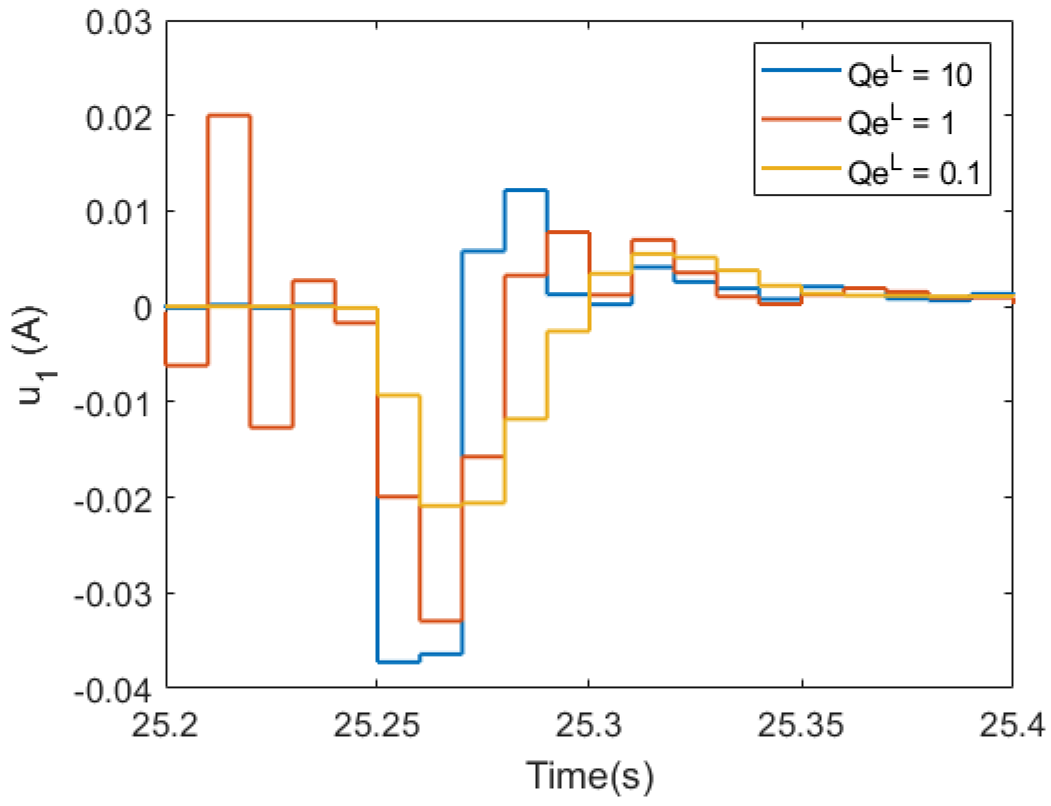
```
%{
load('lookup_angles.mat')
for i = 1:length(data.X(1,:))
    theta1 = data.X(1,i);
    theta2 = data.X(3,i);
    poss_ind = intersect(find(abs(lookup_angles(:,1)-
theta1)<0.5),find(abs(lookup_angles(:,2)-theta2)<0.5));
    index = round(median(poss_ind));
    alpha_f = lookup_angles(index,3);
    gamma_f = lookup_angles(index,4);
end
%}
```

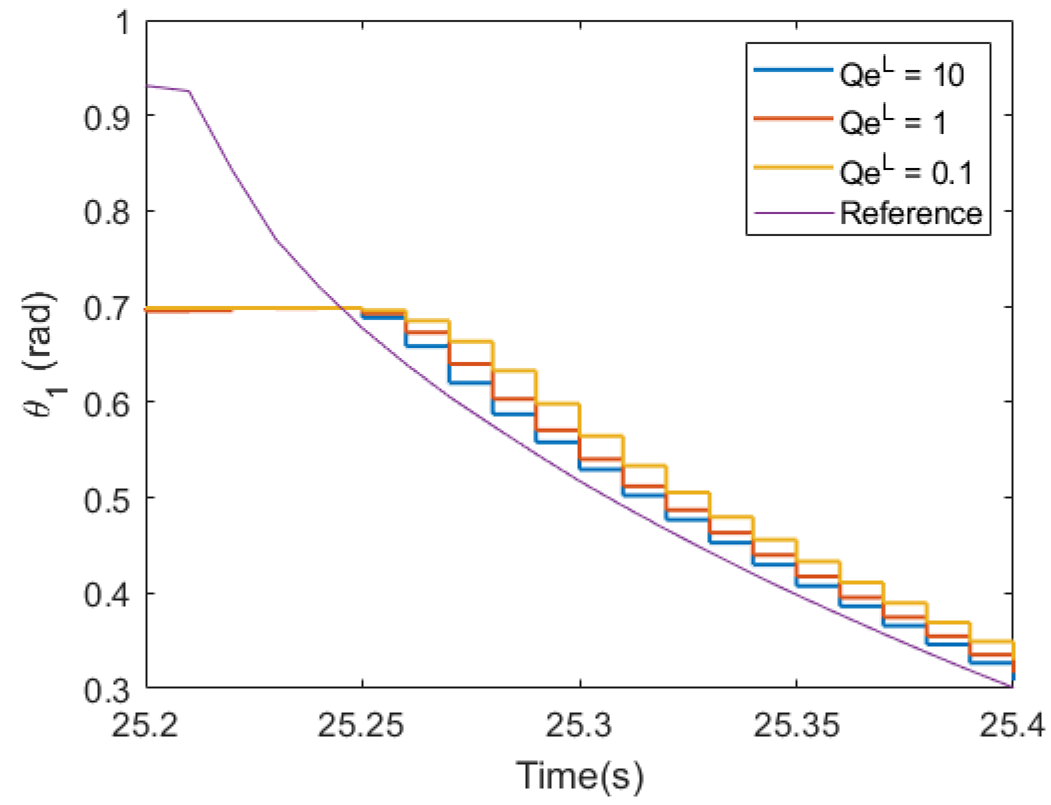
Published with MATLAB® R2021a

Table of Contents

Qy_L

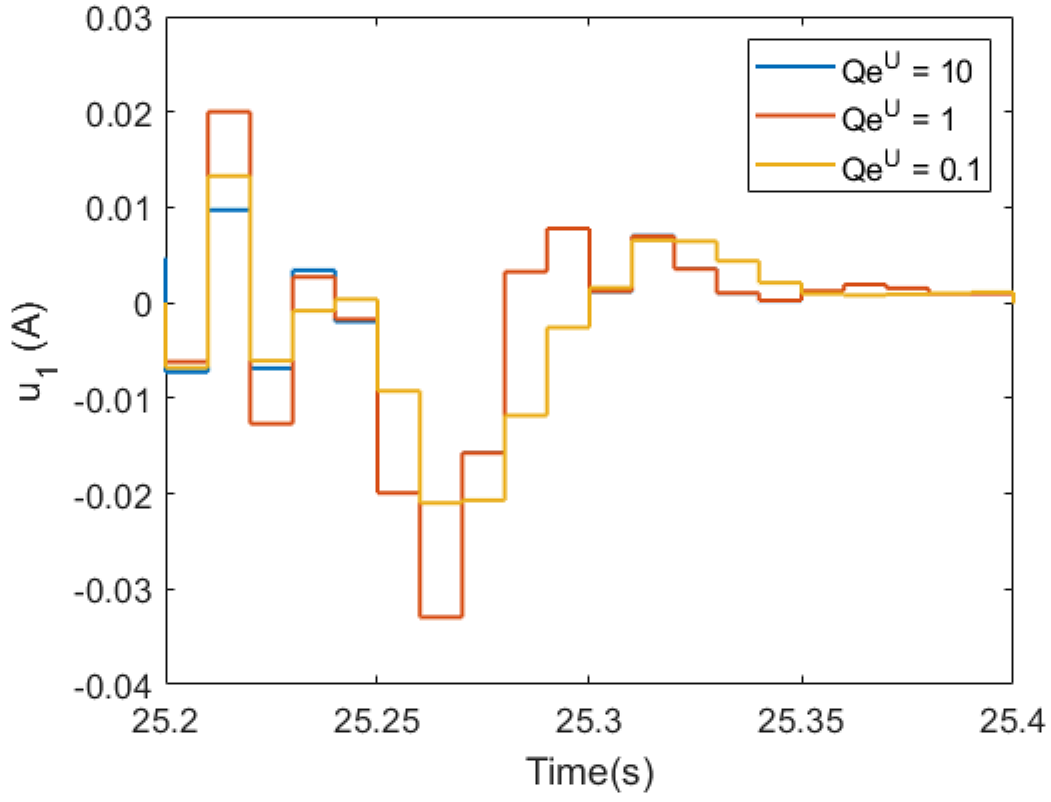
```
QyL100 = load('hier_qyL100_qyU10_R01_sim29.mat','data');
QyL10 = load('hier_qyL10_qyU10_R01_sim29.mat','data');
QyL01 = load('hier_qyL01_qyU10_R01_sim29.mat','data');

figure(1)
stairs(QyL100.data.Time,QyL100.data.U(1,:),'LineWidth',1.5);
hold on
stairs(QyL10.data.Time,QyL10.data.U(1,:),'LineWidth',1.5);
hold on
stairs(QyL01.data.Time,QyL01.data.U(1,:),'LineWidth',1.5);
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('u_1 (A)');
legend('Qe^L = 10', 'Qe^L = 1', 'Qe^L = 0.1');
xlim([25.2,25.4])

figure(2)
stairs(QyL100.data.Time,QyL100.data.X(1,:),'LineWidth',1.5);
hold on
stairs(QyL10.data.Time,QyL10.data.X(1,:),'LineWidth',1.5);
hold on
stairs(QyL01.data.Time,QyL01.data.X(1,:),'LineWidth',1.5);
hold on
plot(QyL100.data.Time,QyL100.data.R(1,:));
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('\theta_1 (rad)');
legend('Qe^L = 10', 'Qe^L = 1', 'Qe^L = 0.1','Reference');
xlim([25.2,25.4])
```

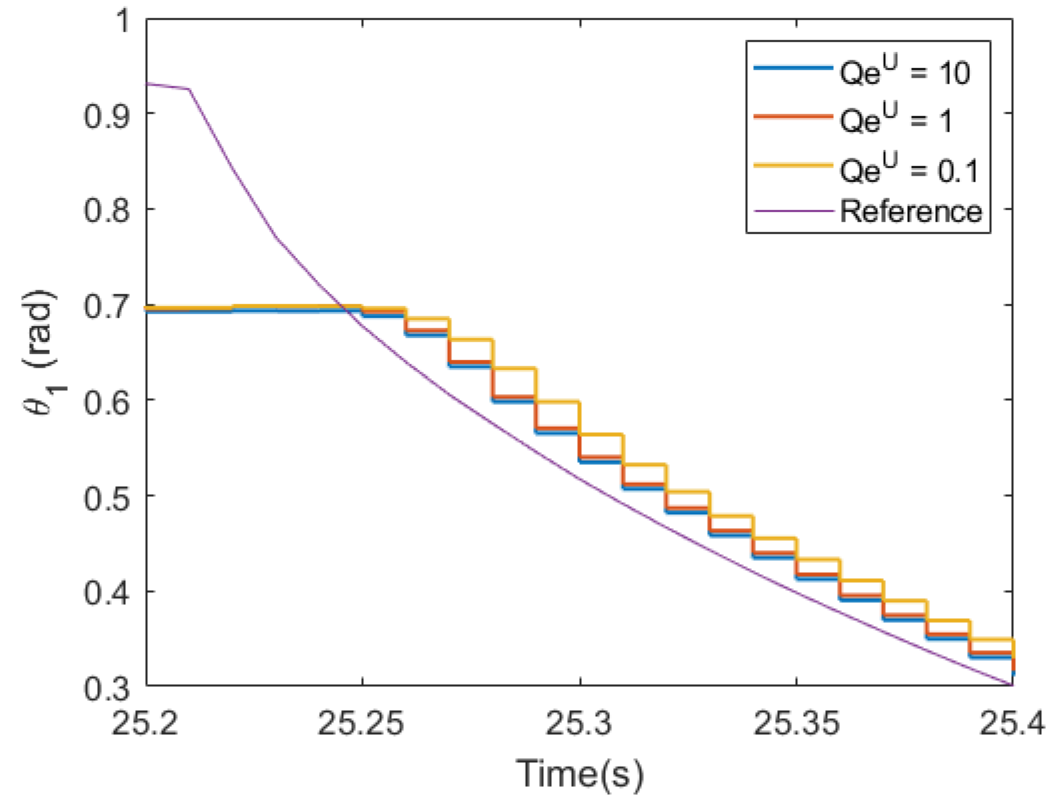

0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
25.2
25.25
25.3
25.35
25.4
Time(s)
u_1 (A)
$Qe^L = 10$
$Qe^L = 1$
$Qe^L = 0.1$


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
25.2
25.25
25.3
25.35
25.4
Time(s)
θ_1 (rad)
$Qe^L = 10$
$Qe^L = 1$
$Qe^L = 0.1$
Reference

Qy_U

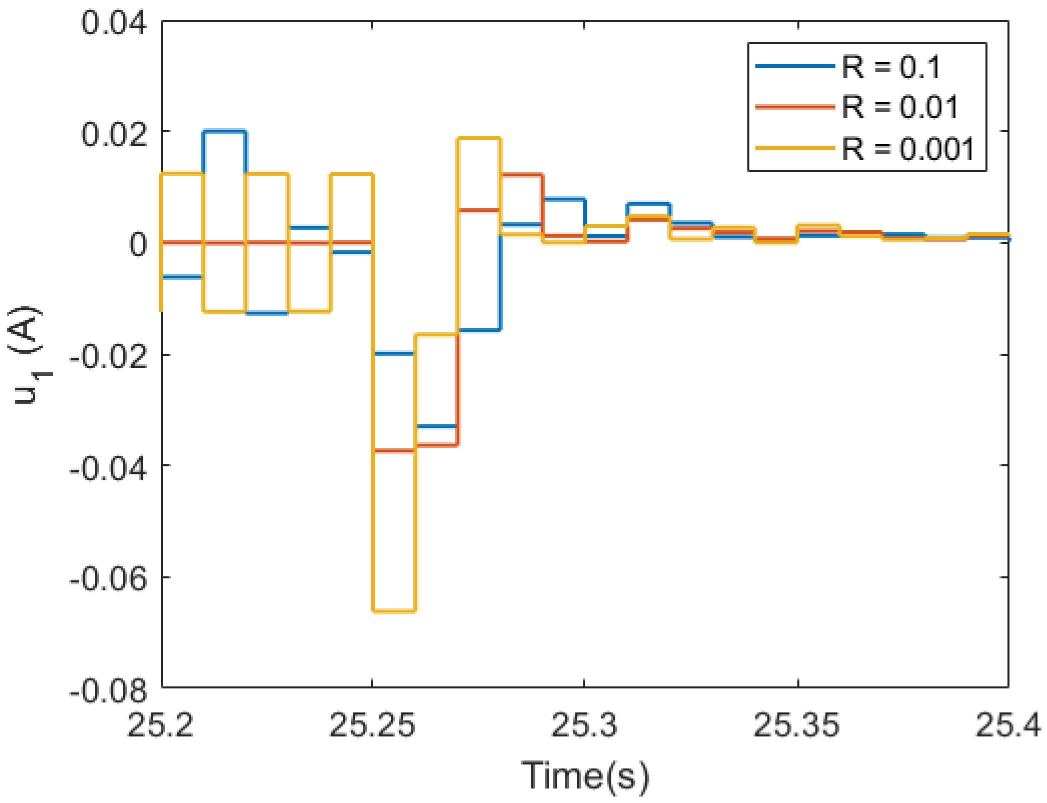
```
QyU100 = load('hier_qyL10_qyU100_R01_sim29.mat','data');
QyU10 = load('hier_qyL10_qyU10_R01_sim29.mat','data');
QyU01 = load('hier_qyL10_qyU01_R01_sim29.mat','data');

figure(3)
stairs(QyU100.data.Time,QyU100.data.U(1,:),'LineWidth',1.5);
hold on
stairs(QyU10.data.Time,QyU10.data.U(1,:),'LineWidth',1.5);
hold on
stairs(QyU01.data.Time,QyU01.data.U(1,:),'LineWidth',1.5);
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('u_1 (A)');
legend('Qe^U = 10', 'Qe^U = 1', 'Qe^U = 0.1');
xlim([25.2,25.4])

figure(4)
stairs(QyU100.data.Time,QyU100.data.X(1,:)-0.005,'LineWidth',1.5);
hold on
stairs(QyU10.data.Time,QyU10.data.X(1,:),'LineWidth',1.5);
hold on
stairs(QyU01.data.Time,QyU01.data.X(1,:),'LineWidth',1.5);
hold on
plot(QyU100.data.Time,QyU100.data.R(1,:));
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('\theta_1 (rad)');
legend('Qe^U = 10', 'Qe^U = 1', 'Qe^U = 0.1','Reference');
xlim([25.2,25.4])
```


$Qe^U = 10$
$Qe^U = 1$
$Qe^U = 0.1$
u_1 (A)
Time(s)

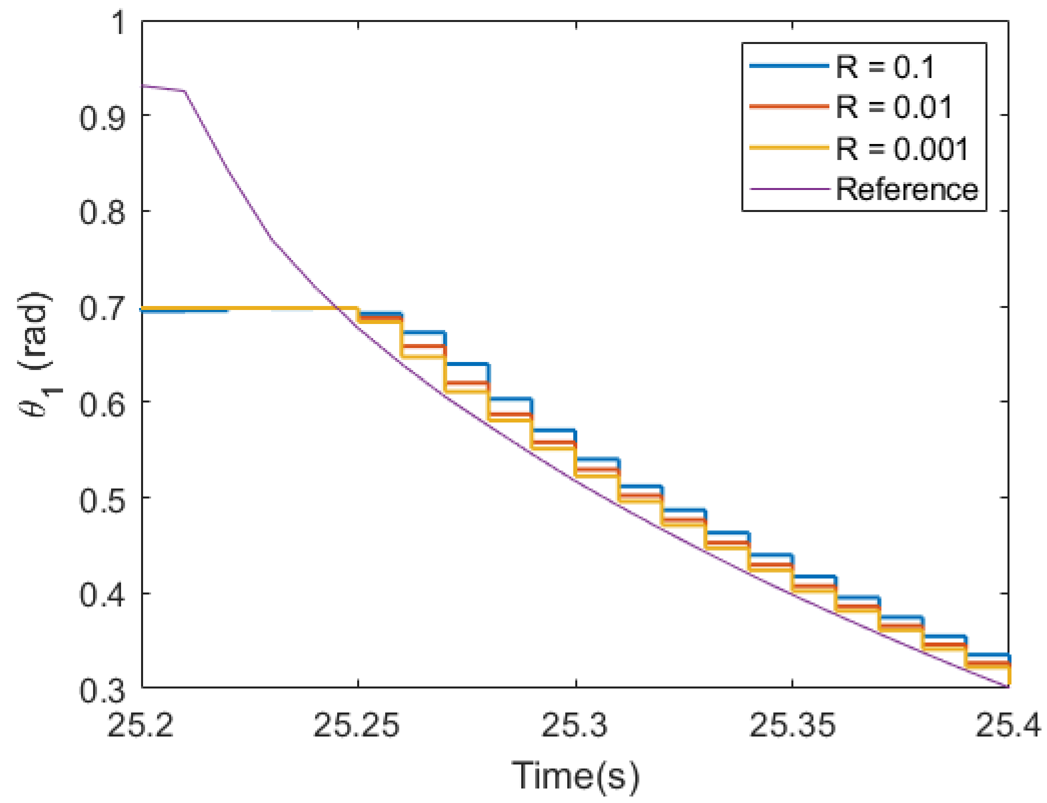

$Qe^U = 10$
$Qe^U = 1$
$Qe^U = 0.1$
Reference
θ_1 (rad)
Time(s)

R

```
R01 = load('hier_qyL10_qyU10_R01_sim29.mat','data');
R001 = load('hier_qyL10_qyU10_R001_sim29.mat','data');
R0001 = load('hier_qyL10_qyU10_R0001_sim29.mat','data');

figure(5)
stairs(R01.data.Time,R01.data.U(1,:),'LineWidth',1.5);
hold on
stairs(R001.data.Time,R001.data.U(1,:),'LineWidth',1.5);
hold on
stairs(R0001.data.Time,R0001.data.U(1,:),'LineWidth',1.5);
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('u_1 (A)');
legend('R = 0.1', 'R = 0.01', 'R = 0.001');
xlim([25.2,25.4])

figure(6)
stairs(R01.data.Time,R01.data.X(1,:),'LineWidth',1.5);
hold on
stairs(R001.data.Time,R001.data.X(1,:),'LineWidth',1.5);
hold on
stairs(R0001.data.Time,R0001.data.X(1,:),'LineWidth',1.5);
hold on
plot(R01.data.Time,R01.data.R(1,:));
axi=gca;
axi.FontSize = 13;
xlabel('Time(s)');
ylabel('\theta_1 (rad)');
legend('R = 0.1', 'R = 0.01', 'R = 0.001','Reference');
xlim([25.2,25.4])
```



R = 0.1
R = 0.01
R = 0.001
u_1 (A)
Time(s)



R = 0.1
R = 0.01
R = 0.001
Reference
θ_1 (rad)
Time(s)

Published with MATLAB® R2021a