MECH 6V29 - MPC - HW 4

Jonas Wagner

***Abstract*—In this homework assignment an active suspension system was analyzed and controlled using both an LQR and an MPC controller.**

I. Problem Definition

The system to be controlled was provided in the MATLAB file as

$$\begin{aligned} \dot{x}(t) &= \mathbf{A}_c x(t) + \mathbf{B}_c u(t) + \mathbf{V}_c d(t) \\ y(t) &= \mathbf{C} x(t) + \mathbf{D} u(t) + \mathbf{W} d(t) \end{aligned} \tag{1}$$

with states $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \in \mathbb{R}^{n_x=4}$ (tire-deflection, unsprung mass velocity, suspension deflection, and sprung mass velocity), input $u = u_1 \in \mathbb{R}^{n_u=1}$ (active suspension force), disturbances $d = \begin{bmatrix} d_1 & d_2 \end{bmatrix}^T \in \mathbb{R}^{n_d=2}$ (road height change and road height), and outputs $y = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^T \in \mathbb{R}^{n_y=3}$ (unsprung mass height, sprung mass height, and spring mass acceleration). The matrices are defined as

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K_t}{M_u} & -\frac{B_s+B_t}{M_u} & \frac{K_s}{M_u} & \frac{B_s}{M_u} \\ 0 & -1 & 0 & 1 \\ 0 & \frac{B_s}{M_s} & -\frac{K_s}{M_s} & -\frac{B_s}{M_s} \end{bmatrix} \mathbf{B}_c = \begin{bmatrix} 0 \\ \frac{-1}{M_u} \\ 0 \\ \frac{1}{M_s} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & \frac{B_s}{M_s} & -\frac{K_s}{M_s} & -\frac{B_s}{M_s} \end{bmatrix} D = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M_s} \end{bmatrix}$$

$$\mathbf{V}_c = \begin{bmatrix} -1 & 0 \\ \frac{B_t}{M_u} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} W = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

with parameters $K_s = 900$ [N/m], $K_t = 2500$ [N/m], $M_s = 2.45$ [kg], $M_u = 1$ [kg], $B_s = 7.5$ [s/m], and $B_t = 5$ [s/m].

The code derives the DT-LTI system at different discretization time steps depending on the selected velocity and the disturbance data.

In addition, bounds on the states, inputs, and disturbances were provided as the intervals:

$$x \in \mathcal{X} = \left\{ x \in \mathbb{R}^{n_x} \mid |x| \leq \begin{bmatrix} 0.01 \\ 1 \\ 0.03 \\ 1 \end{bmatrix} \right\}$$

$$u \in \mathcal{U} = \{ u \in \mathbb{R}^{n_u} \mid |u| \leq 30 \}$$

$$d \in \mathcal{D} = \left\{ d \in \mathbb{R}^{n_d} \mid |d| \leq \begin{bmatrix} 0.5 \\ 0.02 \end{bmatrix} \right\}$$

II. Open-Loop Dynamics

Initial open-loop dynamics are simulated with lsim() and the provided disturbance data resulting in the poor response dynamics shown in Fig. 1 for the realistic road profile (IRI_737b) along with the synthetic roadBumpHole road profiles. (See appendix for additional results) From these dynamics the high oscillatory (and poor dampening) effects can easily be observed. The first state, tire deflection (x_1), was often exceeding the acceptable state bounds.

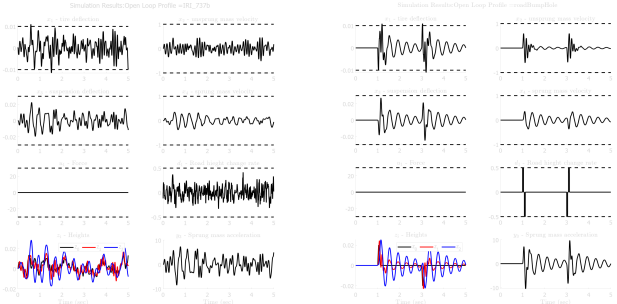

Fig. 1. Open-loop dynamics for the IRI_737b and roadBumpHole road profiles.

III. LQR Controller Design

Initial LQR controller designs were implemented and tested to provide a baseline optimization function for the MPC controller itself. The MATLAB lqr() function was used to design a controller with designed Q and R matrices.

Two primary cost matrices for state-optimization were developed: direct acceleration minimization (Q_{accel}) and bound-based optimization (Q_{bounds}).

$$Q_{accel} = \frac{10^5}{\|C^T C\|_\infty} \cdot C^T \begin{bmatrix} 0 & & \\ & 0 & \\ & & 1 \end{bmatrix} C \tag{2}$$

$$Q_{bounds} = 10^3 \cdot \begin{bmatrix} \frac{1}{0.01} & & & \\ & 1 & & \\ & & \frac{1}{0.03} & \\ & & & 1 \end{bmatrix} \tag{3}$$

These were tested individually and combined as $Q_{both} = Q_{accel} + Q_{bounds}$.

For the input cost, a simple inverse bound cost was used:

$$R = \begin{bmatrix} \frac{1}{0.5} & \\ & \frac{1}{0.02} \end{bmatrix} \tag{4}$$

The simulations are provided in Fig. 2, Fig. 3, and Fig. 4 respectively.

Each of these lqr control schemes had a noticeable improvement in dampening transient response. The demonstrative Q matrices are weighted against the R matrices by multiple orders of magnitude as iteratively testing yielded these weights to have a reasonable balance.

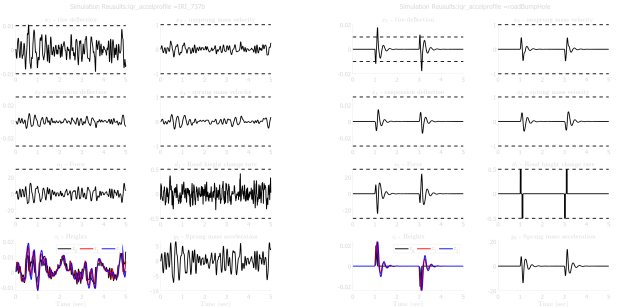

Fig. 2. LQR closed-loop dynamics with Q_{accel} and R for the IRI_737b and roadBumpHole road profiles.

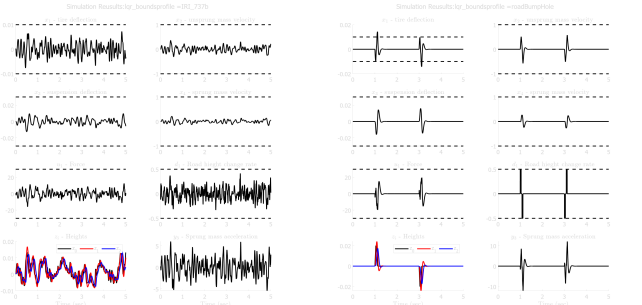

Fig. 3. LQR closed-loop dynamics with Q_bounds and R for the IRI_737b and roadBumpHole road profiles.

Although it does have a noticeable improvement in response, it does not meet the ideal specs; specifically, the state bounds are still often exceeded and the oscillations still don't seem like an enjoyable ride.

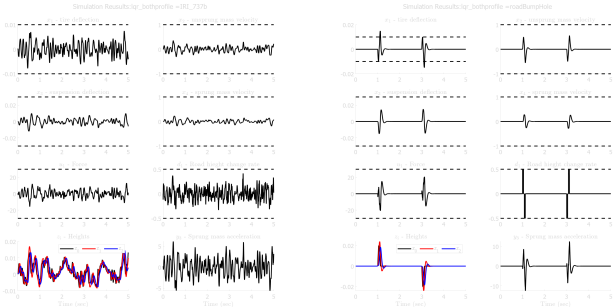

Fig. 4. LQR closed-loop dynamics with Q_both and R for the IRI_737b and roadBumpHole road profiles.

IV. Simple MPC Controller Formulation

An initial implementation of the MPC controller was constructed using the provided code. The same Q and R matrices from before were used and tested iteratively to attempt good responses. (See MATLAB code with rFlag = 0 for the simple formulation). For demonstrative purposes only the Q_{both} version of cost are provided in this report.

A. Update Parameter Selection

The time-horizon and time-step selection were tested for multiple values and ultimately the decision to select a specific value for this was pretty arbitrary. The demonstrative plots are using $N = 15$, but this selection was mostly just an initial guess and since no explicit hardware was selected the computational limit is pretty arbitrary. The integer multiplier was selected mainly by comparing a few different values. As can be seen in Fig. 5, the case where the prediction forward is done every-single time-step the response becomes very oscillatory and the bounds are greatly exceeded.

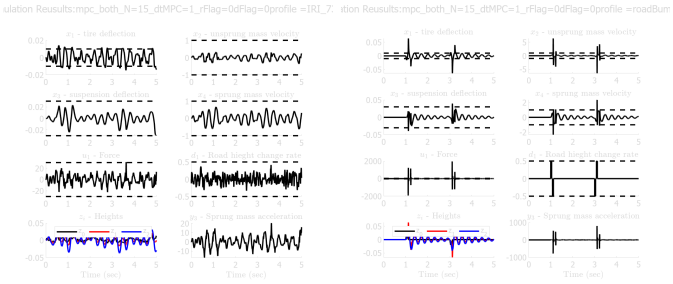

Fig. 5. Closed-loop dynamics with no disturbance information for standard MPC using Q_both and R updating every time-step with $N = 15$ for the IRI_737b and roadBumpHole road profiles.

By comparison, as seen in Fig. 6, predicting forward in increments of 3 time-steps is far less oscillatory; however this is also dependent on what N is.

B. Disturbance knowledge Comparison

The closed-loop dynamics were also tested for when the MPC is given different levels of disturbance information:

1) The nominal case (dFlag = 0) sets $d_1 = d_2 = 0$ for the entire time-horizon.

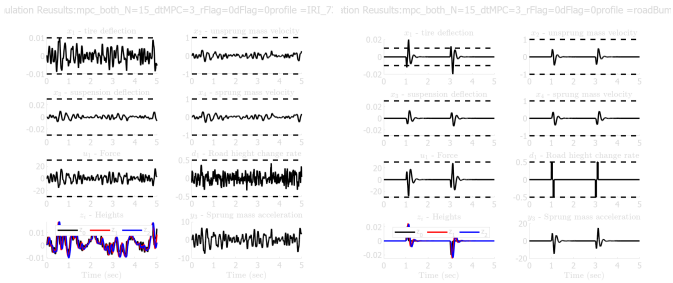

Fig. 6. Closed-loop dynamics with no disturbance information for standard MPC using $Q_b oth$ and R updating every 3 time-steps with $N = 15$ for the IRI_737b and roadBumpHole road profiles.

2) The current state-known case (dFlag = 1) assumes that d_1 and d_2 are known at k. Although also tested when the rest of the testing-horizon is fully nominal ($d_1 = d_2 = 0$) and $d_1 = 0$ for the rest of the horizon, the better approach was for d_2 to be derived from having d_1 held constant for the time-horizon.
3) The entire knowledge case (dFlag = 2) has both d_1 and d_2 directly inputed into the MPC problem.

A direct comparison can be seen in Fig. 7,Fig. 7, and Fig. 8.

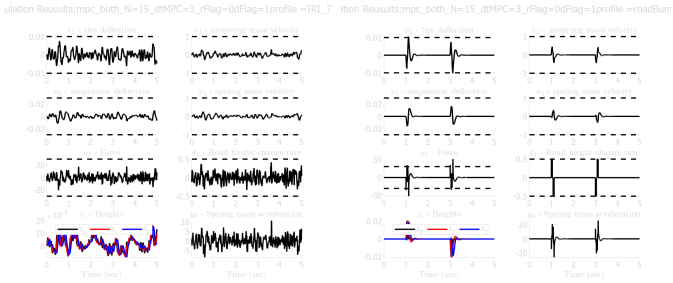

Fig. 7. Closed-loop dynamics with partial disturbance information for standard MPC using $Q_b oth$ and R updating every 3 time-steps with $N = 15$ for the IRI_737b and roadBumpHole road profiles.

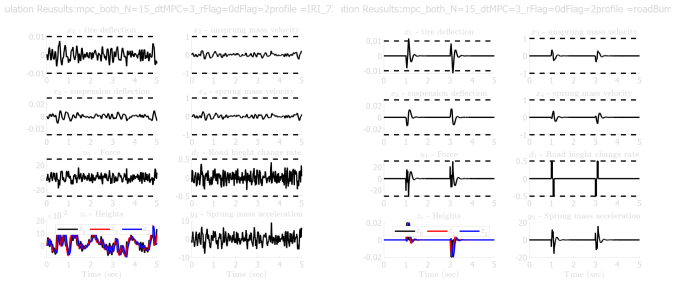

Fig. 8. Closed-loop dynamics with full disturbance information for standard MPC using $Q_b oth$ and R updating every 3 time-steps with $N = 15$ for the IRI_737b and roadBumpHole road profiles.

V. Robust MPC Controller Formulation

The biggest issue with all of the previous implementations has been that the bounds (specifically the tire deflection and input; and even more-so when no disturbance information is known). In order to address this, additional constraints can be implimented to fix this.

A naive approach (something I have done prior to taking this course) is to directly attach these as hard constraints in the optimization problem (see appendix... well really the associated ones in the folder on my github) for these results; but essentially this causes an abundance of feasibility issues.

To fix this issue (something I did do prior to taking this class) would be to use those constraints instead as what I referred to as "soft" constraints within the objective function. This was also tested, although disregarded, since we had already learned better solutions (namely the constraint tightening and tube MPC approaches).

Although I intuitively understand the constraint tightening approach more, the coding/implementation for the RPI sets is simpler[1], thus this was implemented.

[1]not a great reason, but it's why I made the choice I did