Cyber-Physical Control Systems - 2021/2022

Project 1 - Adaptive cruise control design

João Carvalho 49341, Tiago Rodrigues 52856

NOVA School of Science and Technology | FCT NOVA
Largo da Torre, 2825-149 Caparica
{jmfde.carvalho, tmd.rodrigues}@campus.fct.unl.pt

Abstract. This project aims at designing and implementing an adaptive cruise control for a car, that allows the car to follow a leader, maintaining the distance between them at a constant desired set-point. Two MPC controllers were implemented: one unconstrained and another with input and output limitations. The controllers were designed, tested and compared in MATLAB.

key-words: Cyber-Physical systems, Control Theory, Model Predictive Control, MATLAB, Cruise control, MPC

1 Introduction

Adaptive cruise control, or ACC, is an intelligent control system that automatically assists and adjusts a vehicle's speed in order to keep a safe distance from the vehicles ahead. With the increasing development in autonomous driving cars, where safety is a huge concern, in this report, a model predictive control (MPC) system will be designed in order to achieve the goal of a safe and robust adaptive cruise. MPC is a model-based control algorithm that allows the system to operate in an optimal way under imposed constraints. For that matter, a simple follower car model will be derived, followed by the development of both an unconstrained and a constrained MPC controller in a MATLAB based environment. The system's response and stability will be analysed and compared for both controllers, and conclusions will be drawn.

2 Follower car model

2.1 Follower car continuous-time model

Parameter	Symbol	Value
mass	m	$1500\ kg$
drag coefficient	C_d	0.3
frontal area	A	$1.8\ m^2$
air density	ρ	$1.225\ kg\,m^{-3}$
gravity accel.	g	$9.8065\ m\,s^{-2}$
typical slope	θ_e	$5\%; 0.05\ rad$
max. (engine) force	f_{max}	$2600\ N$
min. (brake) force	f_{min}	$-2000\ N$
max. force var.	Δf_{max}	$50\ N$
max. slope	θ_{max}	$10\%; 0.1\ rad$
desired distance	$\tilde{p}_r$	$10\ m$
min. distance	$\tilde{p}_{min}$	$5\ m$
typ. leader speed	v_{r_e}	$20\ m/s$
typ. wind speed	w_e	$0\ m/s$

Figure 1. Follower car's parameters

A continuous-time model for the follower car was derived considering the state vector (1).

$$x(t) = [\tilde{p}(t) \quad v(t)]^T \quad (1)$$

With F_e being the engine or break force, F_w being the air-resistance and F_g the gravitational force. According to Newton's second law the vehicle's equations of motion are the following.

$$\frac{d\tilde{p}(t)}{dt} = v_r(t) - v(t) \quad (2)$$

$$\frac{dv(t)}{dt} = \frac{1}{m}(F_e(t) - F_w(t) - F_g(t) \quad (3)$$

Now, for a space-state representation and taking into account the constants in Table 1, the state vector's derivatives can be seen in Equation 4.

$$\dot{x} = \begin{bmatrix} \dot{\tilde{p}}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v_r(t) - v(t) \\ \frac{F(t)}{m} - \frac{1}{2m}\rho A C_d \big(v(t) + w(t)\big)^2 - g\sin\theta(t) \end{bmatrix} \quad (4)$$

2.2 and 2.3 Equilibrium points and linearization

As the system's equations of motion are not linear, in order to be able to represent and control the system in a linear MPC controller a linearization around a set of defined equilibrium points is necessary.

A linear system of the follower car can be represented by the following equations.

$$\dot{x} = \begin{bmatrix} \dot{\tilde{p}}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v_r(t) - v(t) \\ a_{vv} \cdot v(t) + a_{vu} \cdot f(t) + a_{v\theta} \cdot \theta(t) + a_{vw} \cdot w(t) \end{bmatrix} \quad (5)$$

With:

$$a_{vv} = \frac{dv(t)}{dv}\left(-\frac{1}{2m}\rho A C_d \cdot \left(v(t) + w(t)\right)^2\right) \xrightarrow{v=v_e, w=w_e} \frac{\rho A C_d v_e}{m} \quad (6)$$

$$a_{vu} = \frac{dv(t)}{du}\left(\frac{f(t)}{m}\right) \xrightarrow{f=f_e} \frac{1}{m} \quad (7)$$

$$a_{v\theta} = \frac{dv(t)}{d\theta}(-g \sin\theta(t)) \xrightarrow{\theta=\theta_e} -g \quad (8)$$

$$a_{vw} = \frac{dv(t)}{dw}\left(-\frac{1}{2m}\rho A C_d \cdot \left(v(t) + w(t)\right)^2\right) \xrightarrow{v=v_e, w=w_e} -\frac{1}{2m}\rho A C_d v_e \quad (9)$$

Which results in the following linear state-space representation of the system.

$$\dot{x} = \begin{bmatrix} \dot{\tilde{p}}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & -\frac{\rho A C_d v_e}{m} \end{bmatrix} \begin{bmatrix} \tilde{p}(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f(t) + \begin{bmatrix} 1 & 0 & 0 \\ 0 & -g & -\frac{\rho A C_d v_e}{m} \end{bmatrix} \begin{bmatrix} v_r(t) \\ \theta(t) \\ w(t) \end{bmatrix} \quad (10)$$

$$y(t) = [1 \quad 0][p(t) \quad v(t)]^T$$

2.4 Discrete-time state-space model

Using Euler's method the resulting discrete-time state-space model for a sampling time, T, of 0.1s is the following.

$$A_d \approx I + AT \approx \begin{bmatrix} 1 & -0.1 \\ 0 & 1.0009 \end{bmatrix}$$

$$B_d \approx BT \approx \begin{bmatrix} 0 \\ 0.6667 \end{bmatrix} 10^{-4} \quad (11)$$

$$C_d \approx C \approx [1 \quad 0]$$

2.5 Stability, observability and controllability analysis of the discrete-time model

The system's stability can be analysed by computing the state-matrix's eigenvalues, which are equivalent to the system's poles, and verify if they're inside the complex-plane unit sphere. With at least one of eigenvalues' absolute values larger than 1, the discretized system is unstable.

$$|A_d - \lambda I| = 0 \Leftrightarrow \lambda = \{1, 1.0009\} \quad (11)$$

The system's controllability and observability are analysed through the controllability and observability matrix ranks.

$$rank\, M_c(2) = rank\, [B_d \quad A_d B_d] = rank \begin{bmatrix} 0 & -0.0667 \\ 0.6667 & 0.6673 \end{bmatrix} \cdot 10^{-4} = 2 \quad (12)$$

$$rank\, M_o(2) = rank \begin{bmatrix} C_d \\ C_d A_d \end{bmatrix} = rank \begin{bmatrix} 1 & 0 \\ 1 & -0.1 \end{bmatrix} = 2 \quad (13)$$

The ranks are equal to the number of states in the system so the resulting discrete-time system is controllable and observable.

2.6 Open-loop simulation

Using the MATLAB function *lsim*, the system's step response, with an amplitude equal to 30 N, was simulated in open-loop, with the car starting at the equilibrium position $x_0 = [0 \ 0]'$.

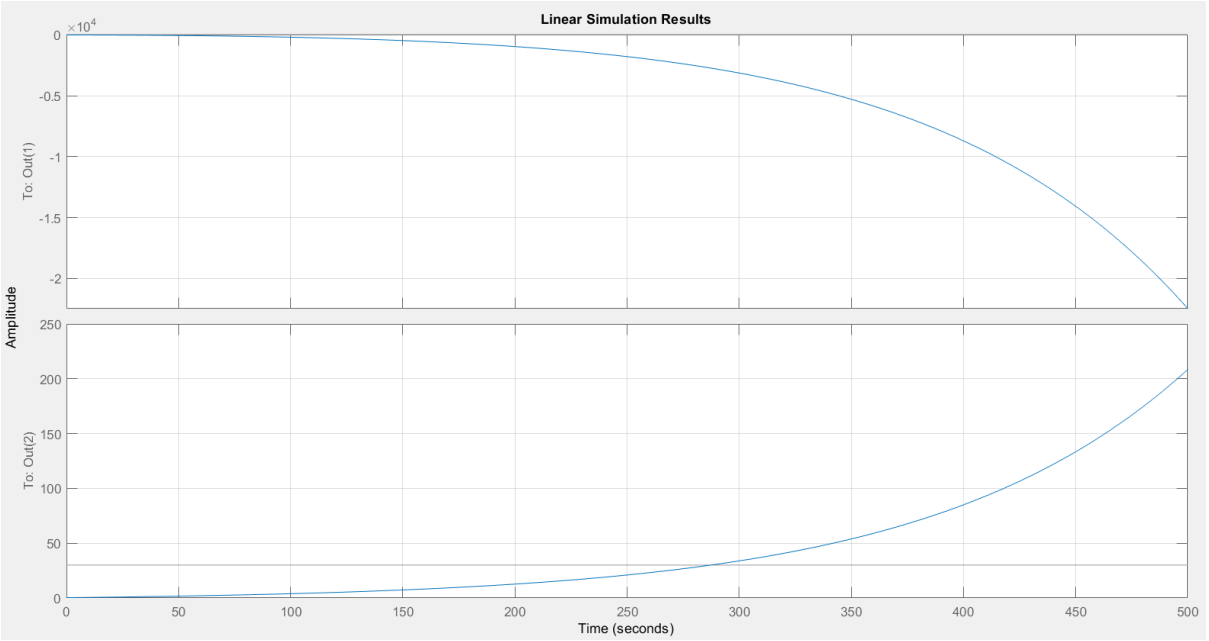

Figure 2. Open loop step response.

The system is clearly unstable as both outputs diverge from the reference and tend to infinity.

3 Unconstrained predictive control

In this section a model-based predictive controller, MPC, will be designed in order to implement the desired cruise control.

3.1 Augmented model with embedded integrators

The system model was augmented in order to eliminate steady state error. With a new state vector defined as:

$$x(k) = \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix} \quad (14)$$

And new augmented A, B and C matrixes.

$$A = \begin{bmatrix} A_d & 0 \\ C_d A_d & I \end{bmatrix} = \begin{bmatrix} 1 & -0.1 & 0 \\ 0 & 1.0009 & 0 \\ 1 & -0.1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} B_d \\ C_d B_d \end{bmatrix} = \begin{bmatrix} 0 \\ 0.6667 \\ 0 \end{bmatrix},$$
$$C = [0 \quad I] = [0 \quad 0 \quad I] \quad (15)$$

In this new system, the number of states increased from 2 to 3.

3.2 Stability, observability and controllability analysis of the augmented system

Similar to the work done in 2.4, the augmented systems stability, observability and controllability were analysed.

$$|A_d - \lambda I| = 0 \Leftrightarrow \lambda = \{1, 1, 1.0009\} \quad (16)$$

With poles at the frontier and outside the unit circle, the system remains unstable.

$$rank\, M_c(3) = rank\,[B \quad AB \quad A^2B] = rank \begin{bmatrix} 0 & -0.0667 & -0.1334 \\ 0.6667 & 0.6673 & 0.6678 \\ 0 & -0.0667 & -0.2001 \end{bmatrix} \cdot 10^{-4} = 3 \quad (17)$$

$$rank\, M_o(2) = rank \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} = rank \begin{bmatrix} 0 & 0 & 1 \\ 1 & -0.1 & 1 \\ 2 & -0.3001 & 1 \end{bmatrix} = 3 \quad (18)$$

With the ranks of the observability matrix and controllability equal to 3, or the number of states, the system is both observable and controllable.

3.3 Augmented model's matrix form

The matrix form of the augmented model is represented by the following equation.

$$\bar{Y} = \bar{F}x(0) + \bar{G}\Delta U \quad (19)$$

With:

$$\bar{F} = H \cdot F = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix}, \qquad \bar{G} = H \cdot G = \begin{bmatrix} 0 & 0 & 0 \\ CB & 0 & 0 \\ CAB & CB & 0 \\ CA^2B & CAB & CB \end{bmatrix} \quad (20)$$

3.4 Quadratic cost function for the augmented incremental control

The quadratic cost function can be defined as.

$$\min_{\Delta u_0, \dots, \Delta u_{N-1}} J_0 = (y_N - \bar{y}_N)^T P (y_N - \bar{y}_N) + \sum_{k=0}^{N-1} (y_k - \bar{y}_k)^T Q (y_k - \bar{y}_k) + \Delta u_k^T R \Delta u_k \quad (21)$$

Such that:

$$\begin{aligned} x_{k+1} &= Ax_k + B\Delta u_k \\ y_k &= Cx_k \\ x_0 &= z_0 \end{aligned} \quad (22)$$

In the batch form, taking into account the matrices computed in 3.3, the following matrices are obtained:

$$\tilde{R} = \bar{G}^T \bar{Q}\, \bar{G} + \bar{R} \quad , \quad \tilde{S} = \bar{G}^T \bar{Q} \quad (23)$$

With:

$$\bar{R} = diag(R, R, \dots, R) \quad (24)$$

As such, the gains can be computed:

$$K_y = \tilde{R}^{-1}\,\tilde{S} \quad , \quad K = \tilde{R}^{-1}\,\tilde{S}\,\bar{F} \quad (25)$$

3.5 Unconstrained optimal control problem for the augmented model

The control action to be used is defined as:

$$u(k) = \Delta u(k) + u(k-1) \quad (26)$$

And defining ΔU, for an horizon N:

$$\Delta U = [\Delta u(0)^T \dots \Delta u(N-1)^T]^T \quad , \quad X = Fx(0) + G\Delta U \quad (27)$$

With the gains defined in 3.4, the optimal control sequence is:

$$\Delta U^* = -Kx(0) + K_y\bar{Y} = \{\Delta u_0^*, \dots, \Delta u_{N-1}^*\} \quad (28)$$

And the respective receding horizon control policy:

$$\Delta u(t_k) = \Delta u_0^* = -[I\ 0 \dots 0]\left(Kx(0) + K_y\bar{Y}\right) = -K_0 x(0) + K_{y_0}\bar{y}(0) \quad (29)$$

3.6 Unconstrained MPC control policy and system simulation

The unconstrained MPC control policy can be defined as the following control action:

$$u(t_k) = u(t_{k-1}) + \Delta u(t_k) \quad (30)$$

The system and a closed-loop unconstrained MPC controller were simulated for a square-wave with 50% duty cycle with amplitudes between -10 and 10 during 500 time intervals. The MPC cost function parameters N, R, Q and P are, respectively, valued 50, 0.1, 0.5 and 500, which are the input cost, output cost and final output cost, respectively. To solve the optimization problem, the MATLAB function *quadprog()* was used.

-

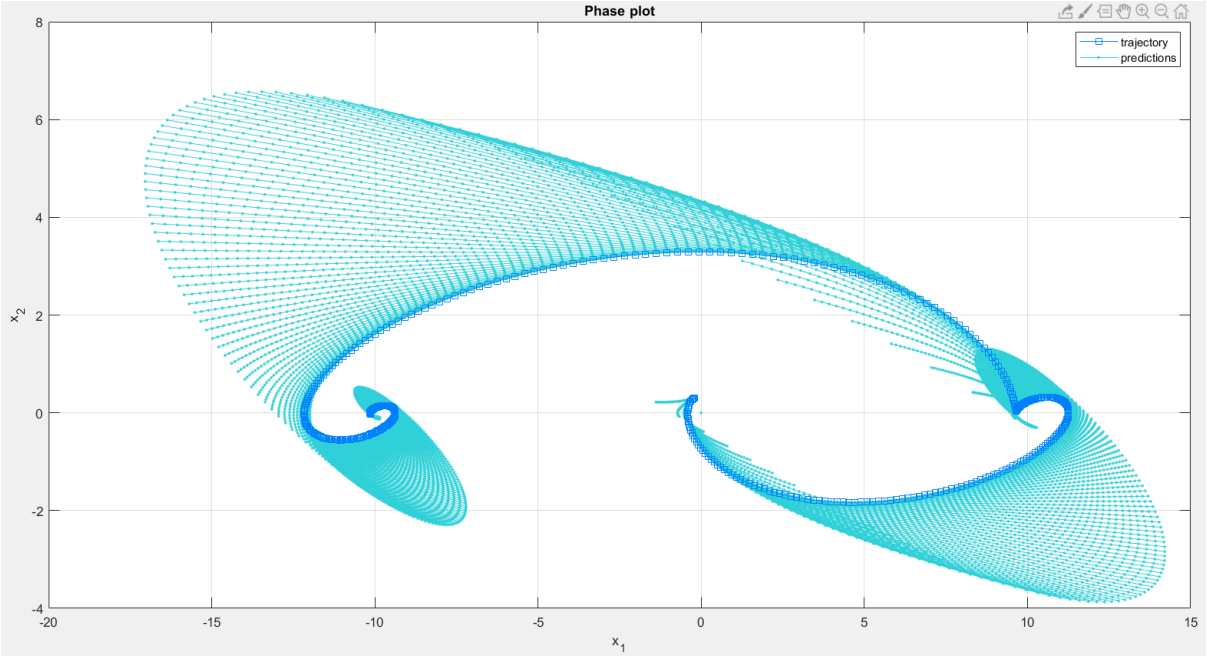

Figure 3. Unconstrained MPC controller: simulation phase plot.

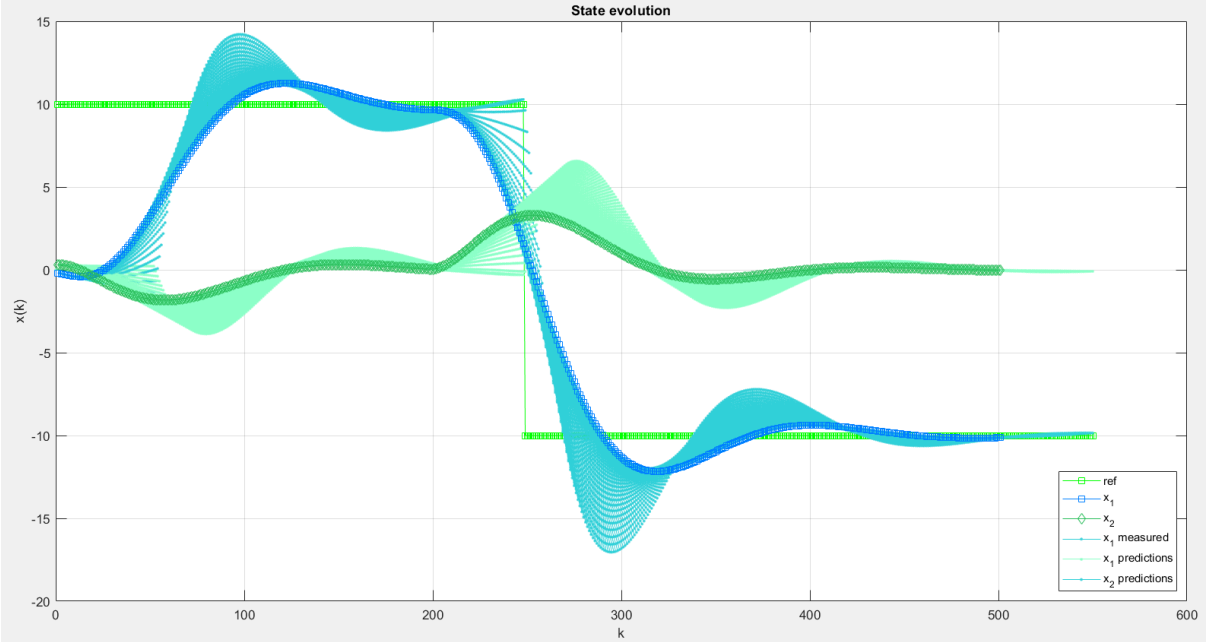

Figure 4. Unconstrained MPC controller: simulation state evolution.

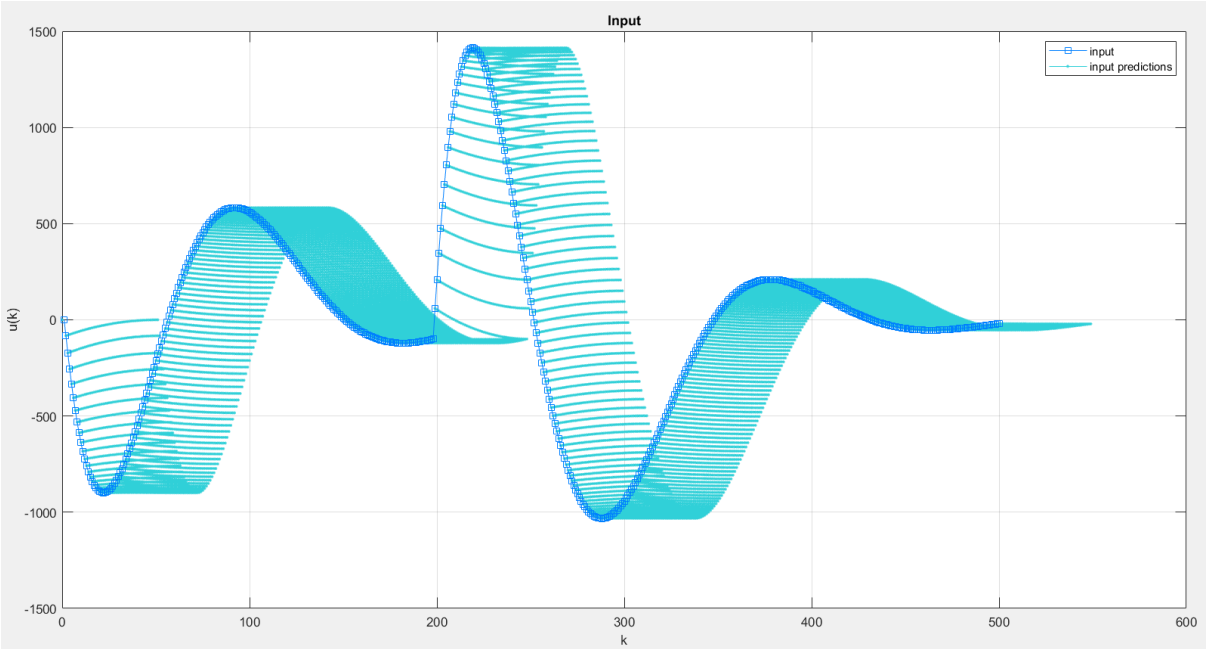

Figure 5. Unconstrained MPC controller: simulation input.

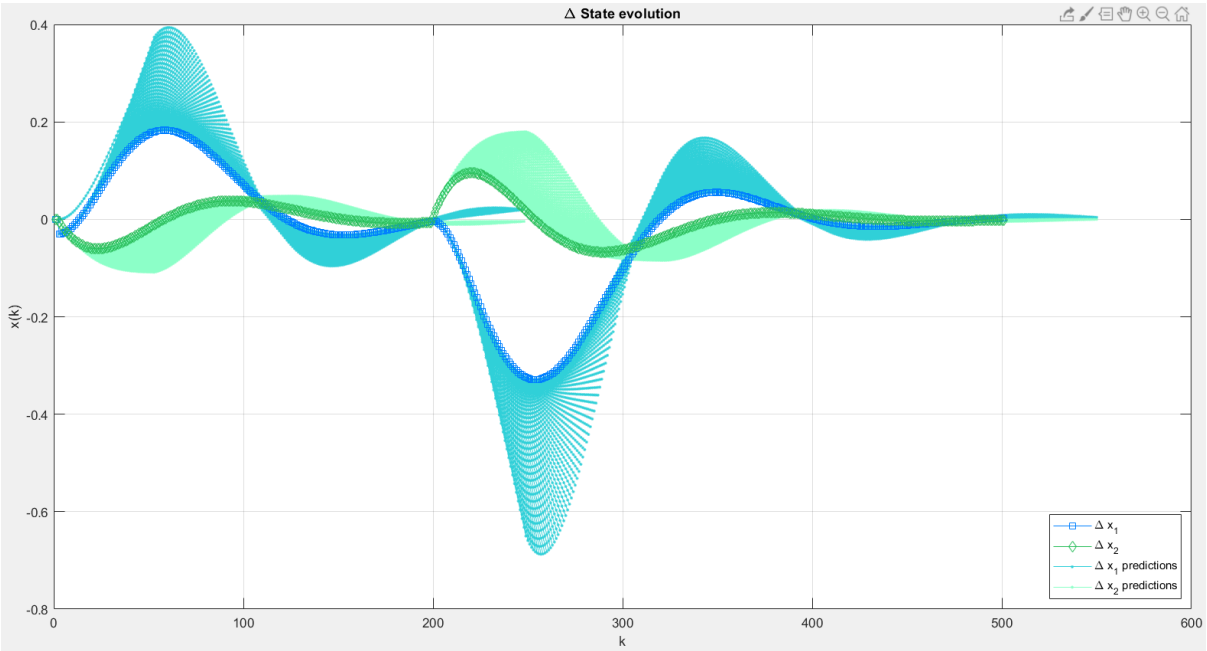

Figure 6. Unconstrained MPC controller: state variation.

3.7 Unconstrained MPC control policy and system simulation: different parameters

Given other cost function parameters, we can achieve a faster response and a reduced over-shoot. Using N, R, Q and P as 70, 0.0001, 1 and 1000, respectively, we have:

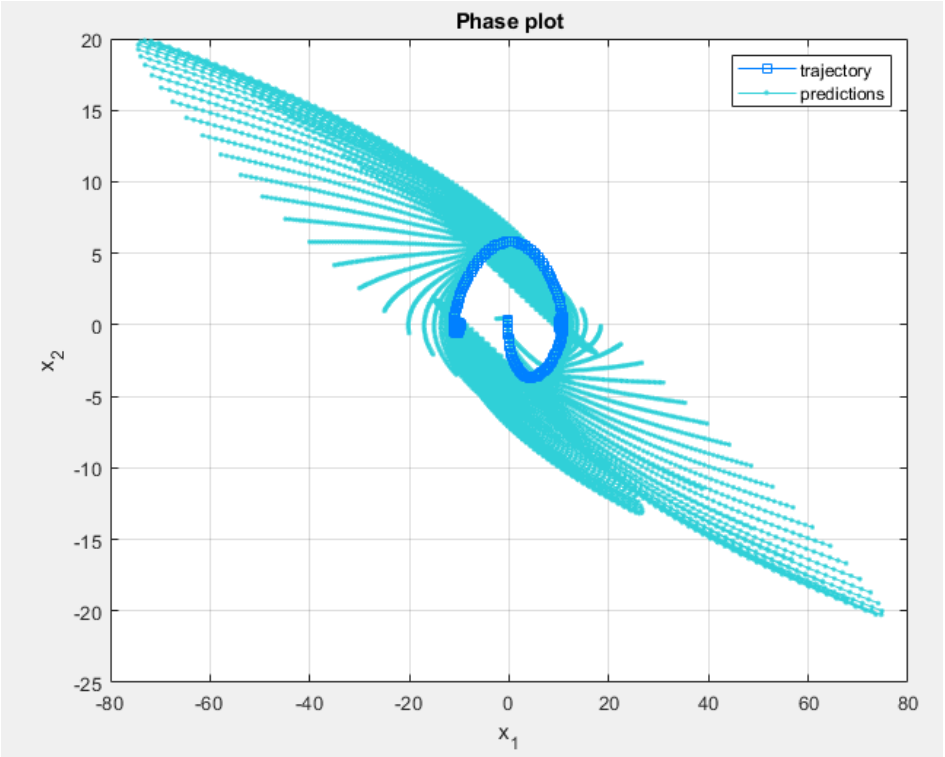

Figure 7. Unconstrained MPC controller for different parameters: simulation phase plot.

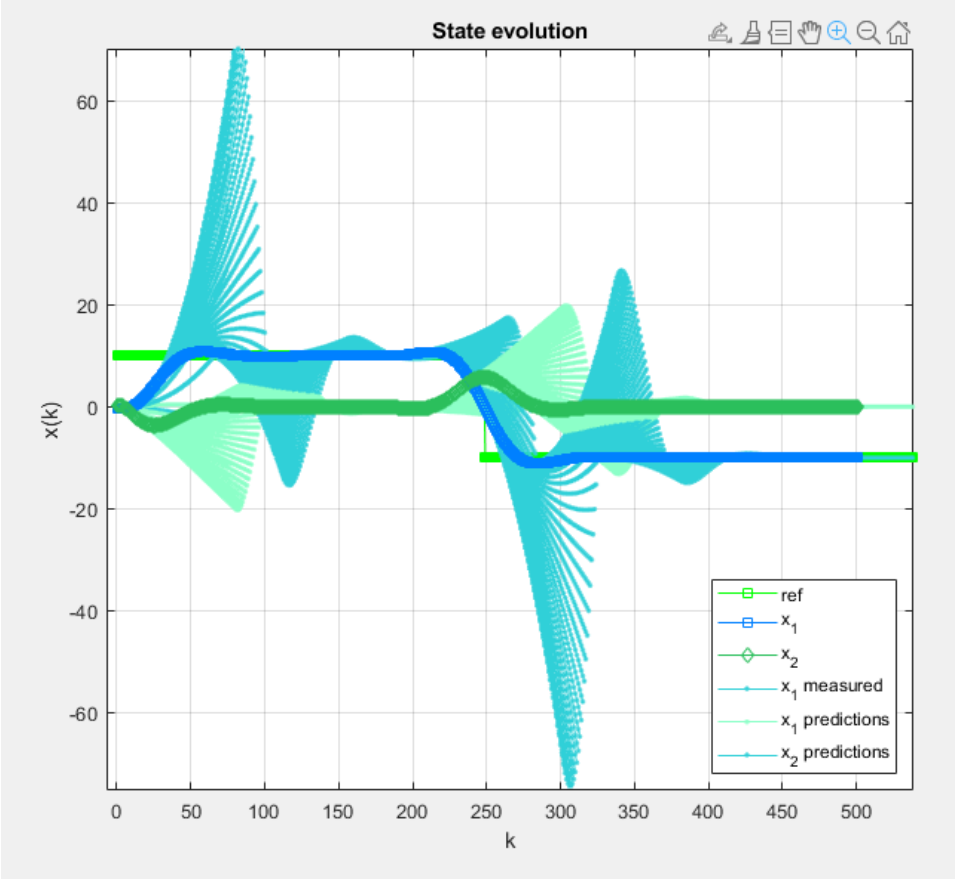

Figure 8. Unconstrained MPC controller for different parameters: simulation state evolution.

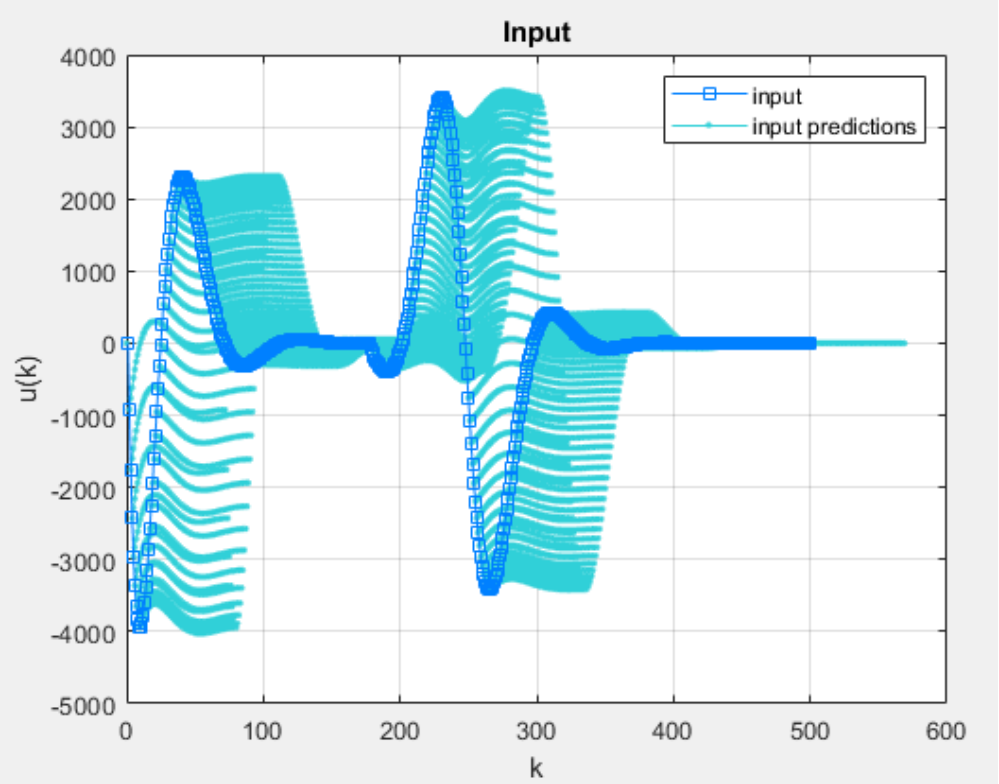

Figure 9. Unconstrained MPC controller for different parameters: simulation input.

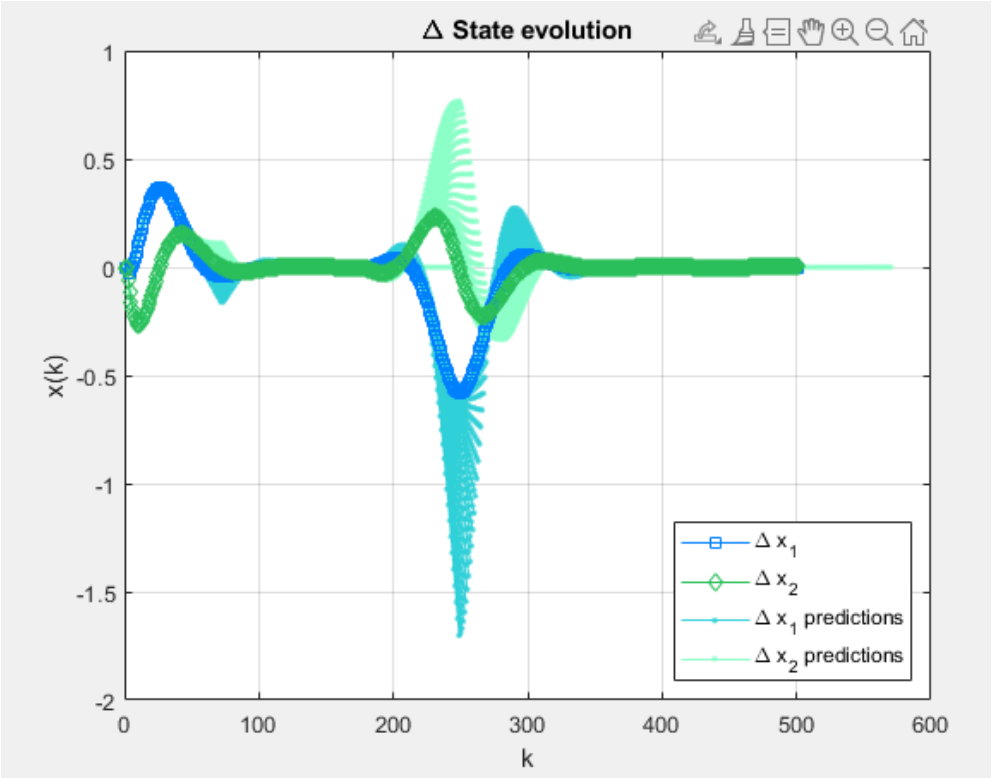

Figure 10. Unconstrained MPC controller for different parameters: simulation state variation.

4 Constrained predictive control

A constrained MPC controller was built in this section based on the previously built unconstrained version. Input limitations on the engine's force and force variation were applied, together with a restriction on the minimum distance to the lead car. In a first approach, we will try to use a saturator in the unconstrained MPC, in order to reach the problem limitations, and then design a new MPC controller taking into account the restrictions that are given in the problem. In section 4.3 we will present the constrained MPC controller and then, in section 4.4 we will discuss the differences between the saturated unconstrained MPC controller and the constrained MPC controller.

4.1 Constrained MPC problem definition

Defining the constrained tracking predictive control problem given by (21), we also consider the constraints:

$$x_{k+1} = Ax_k + Bu_k, \qquad y_k = Cx_k, \forall_{k=0,\dots,N-1}, x_0 = x_{t_k} \quad (31)$$

$$x_k \in \chi, u_k \in U, \forall_{k=0,\dots,N-1}, x_N \in \chi_f$$

Or in the batch form:

$$M\Delta U \leq \omega \quad (32)$$

For this problem we used as constraints:

$$Engine\ force:\ f_{min} \leq f(k) \leq f_{max}$$

$$Engine\ force\ variation:\ -\Delta f_{max} \leq \Delta f(k) \leq \Delta f_{max} \quad (33)$$

$$Distance\ to\ leader\ car:\ \tilde{p}(k) \geq \tilde{p}_{min}$$

Where the parameters are defined in table 1. In the matrix form we have:

$$\begin{bmatrix} M_{\Delta u} \\ M_u \\ M_y \end{bmatrix} \Delta U \leq \begin{bmatrix} \omega_{\Delta u} \\ \omega_u \\ w_y \end{bmatrix} \quad (34)$$

With:

$$M_u = \begin{bmatrix} -M_3 \\ M_3 \end{bmatrix}$$

$$M_{\Delta u} = \begin{bmatrix} -I_N \\ I_N \end{bmatrix} \quad (35)$$

$$M_y = \begin{bmatrix} -\bar{G} \\ \bar{G} \end{bmatrix}$$

And:

$$\omega_u = \begin{bmatrix} U_{max} + M_4 u_1 \\ U_{max} - M_4 u_1 \end{bmatrix}$$

$$\omega_y = \begin{bmatrix} -Y_{min} + \bar{F}x(k) \\ Y_{max} - \bar{F}x(k) \end{bmatrix} \quad (36)$$

$$\omega_{\Delta u} = \begin{bmatrix} -\Delta u_{min} \\ \Delta u_{max} \end{bmatrix}$$

$$\omega = \begin{bmatrix} \omega_u \\ \omega_y \\ \omega_{\Delta u} \end{bmatrix}$$

Here, we need to pay attention to the fact that the constrained variables are incremental, not the actual physical variable [1], so we have to subtract for each constraint the linearization point. Therefore, the new constraints are:

$$f_{max} = 2467{,}7$$

$$f_{min} = -2132{,}3 \quad (37)$$

$$\tilde{p}_{min} = -5$$

4.2 Constrained MPC controller and results

The controller was implemented, and after several parameter adjustments, the following results were obtained. A horizon of N equal to 50 was used. The utilized reference signal is the same as the one used in 3.6, a square wave with amplitude between -10 and 10. The MPC controller parameters N, R, Q and P are, respectively, valued 50, 0.1, 0.5 and 300.
To solve the optimization problem the Matlab function *quadprog()* was used.

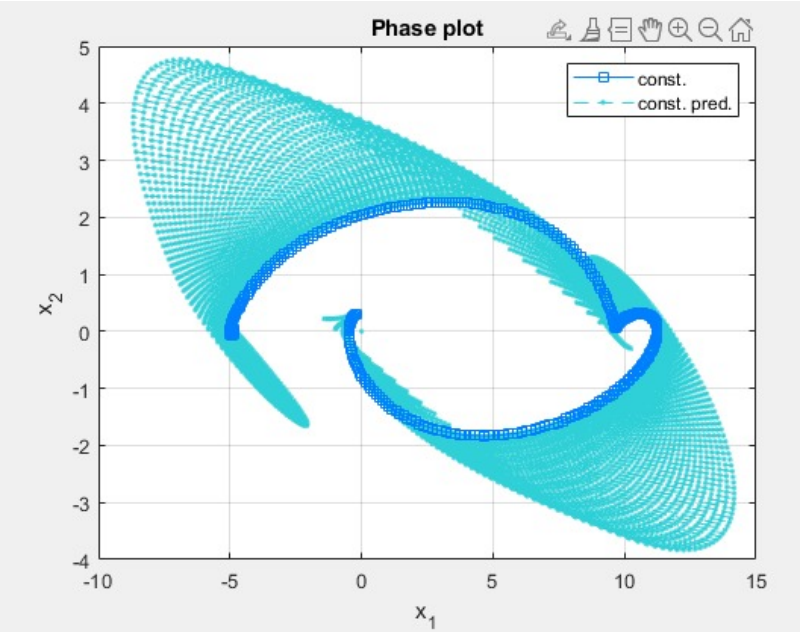

Figure 11. Constrained MPC controller: simulation phase plot

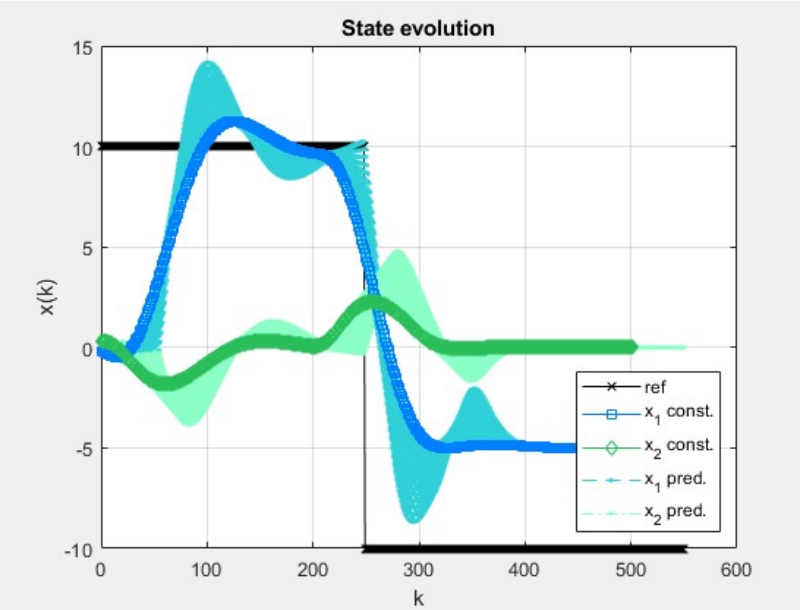

Figure 12. Constrained MPC controller: simulation state evolution.

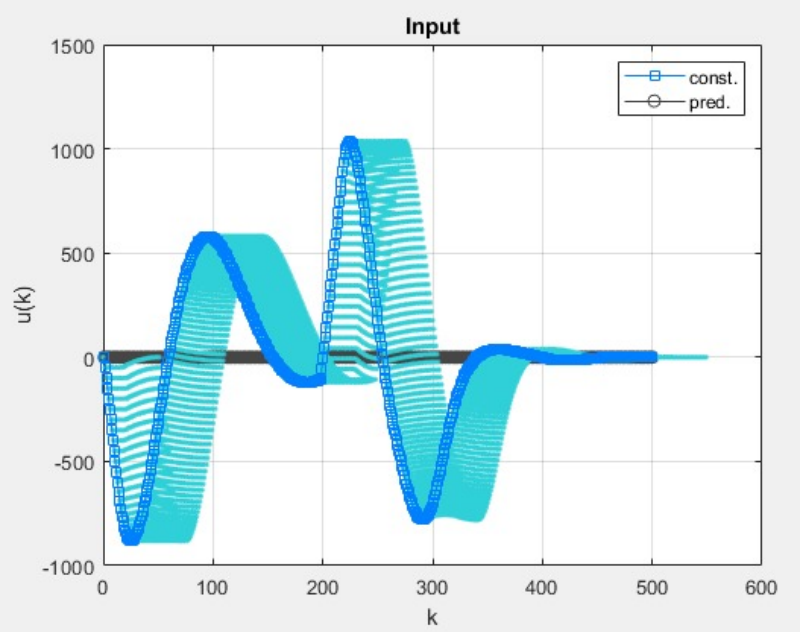

Figure 13. Constrained MPC controller: simulation input.

4.3 Comparison between unconstrained and constrained MPC controllers

Given the results, we can say that the unconstrained MPC controller can achieve a better performance, due to its faster response time, when compared to the constrained MC controller. Although, because the physical system has limitations, we can go for that solution in this case.
As mentioned above in this chapter, we tried to introduce the problem limitations on the MPC controller by saturation. As we can see in the figures below, the system was driven to instability, so in this case a constrained MPC is mandatory.

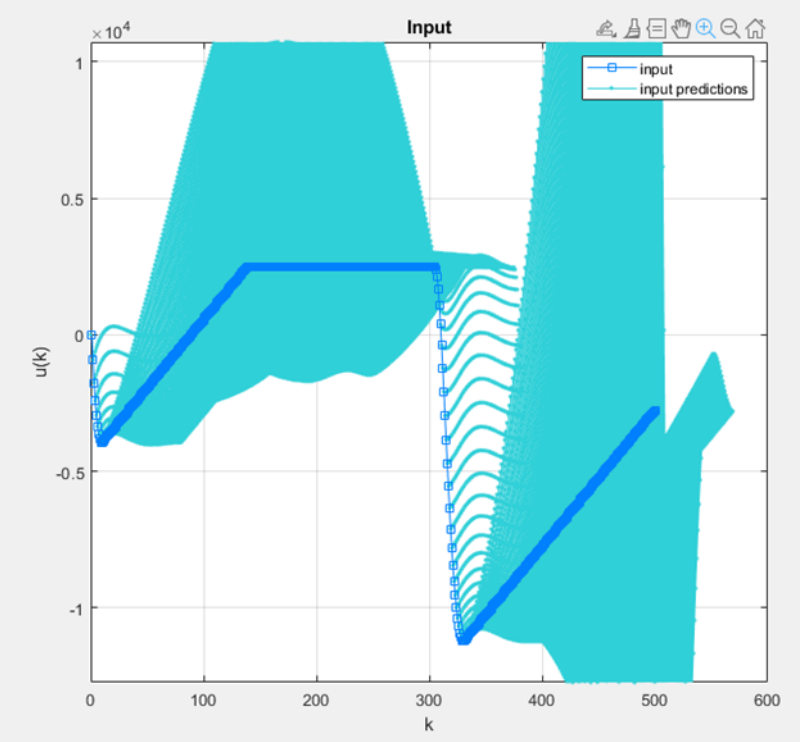

Figure 14. Control action of the unconstrained MPC through saturation of the input

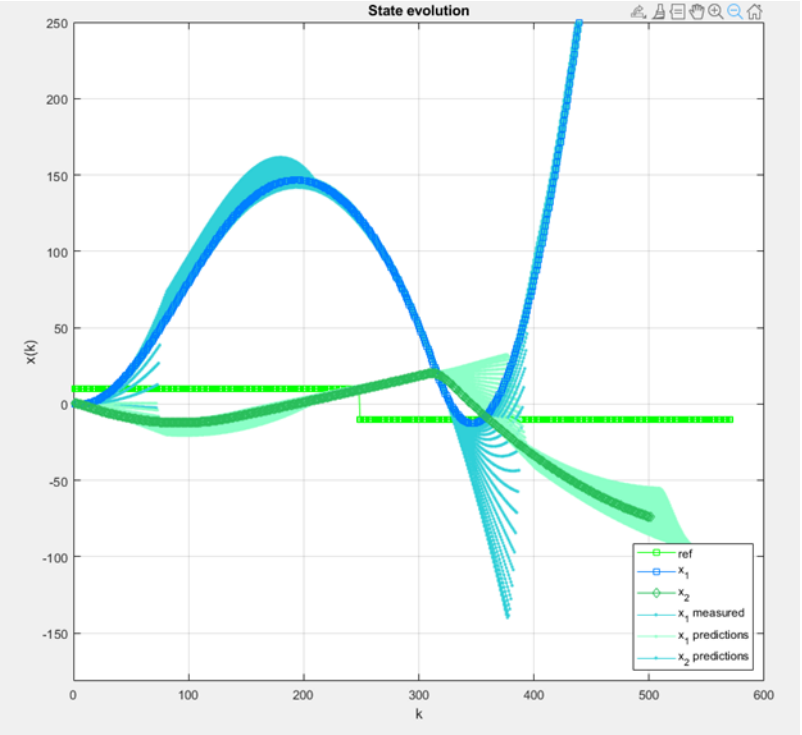

Figure 15. State variables of the unconstrained MPC through saturation of the input

As another test, we saturated the output value, and as we can see is that the control action reached enormous values, called wind-up phenoma. We can see that also drove the system to instability.

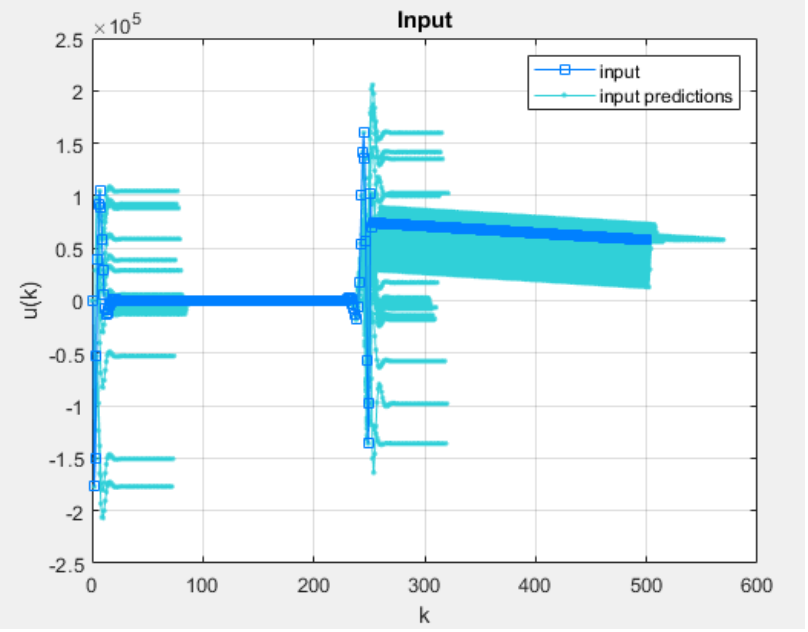

Figure 16. Wind-up phenoma due to output saturation

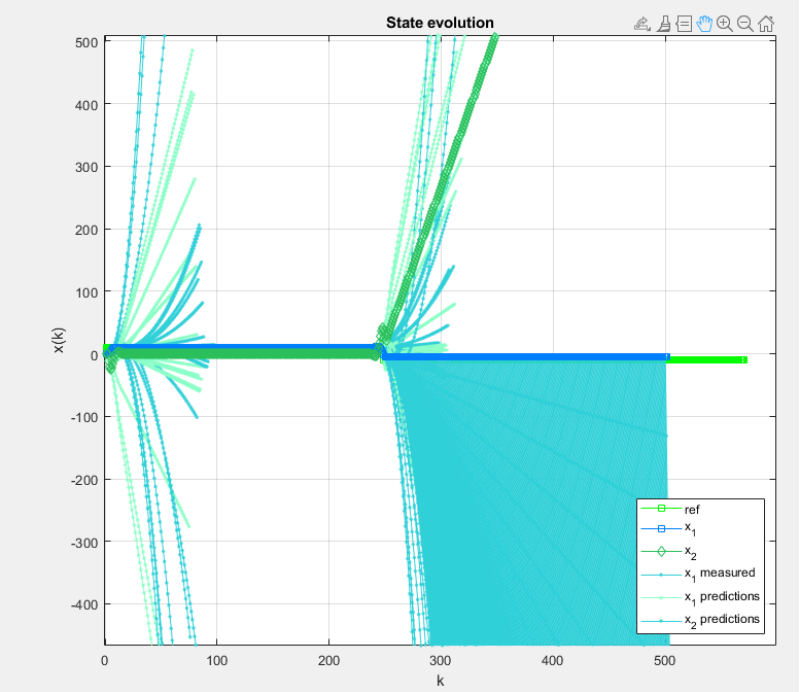

Figure 16. System driven to instability due to output saturation

4 Conclusion

The proposed controllers were successfully implemented and allowed to control an otherwise unstable system.

It should be reiterated that the unconstrained controller is much faster than the constrained version, however it cannot be utilized in real-life applications as the system requires constraints and limitations. On top of that, it's optimal and recommended to utilize the constrained controller opposed to an input-saturated unconstrained version.

It's also noteworthy to mention that the system works best around the specified equilibrium points, and as such, for a real implementation multiple equilibrium points and respective conditioning should be used. Otherwise, when facing abrupt changes the system can be driven to instability or suboptimal behavior.

References

1. Liuping Wang. Model Predictive Control System Design and Implem. Using MAT- LAB. Springer, 2009.