A model predictive control approach to driving a car while avoiding obstacles

Christiaan Theunisse (5072379)

***Abstract*—We study a model predictive control (MPC) approach for driving a car and performing lane changing maneuvers. In this paper, we design two linear model predictive controllers: one based on state feedback and one based on output feedback. Futhermore, the state feedback controller is proven to be asymptotically stable. Simulations on the nonlinear model have been performed to show the usability of the controllers. Finally, we provide some recommendations for future research[1].**

I. Introduction

Autonomous car are the main research topic within the field of robotics and receive billions in funding each year [1]. Still, there is a lot a work to do before fully autonomous cars will be seen on the road. A autonomous car should be able to perceive its environment through different sensors types. This includes the conversion of raw sensory data into a understanding of the environment, which should at least include a mapping of free spaces and obstacles. Subsequently, the car needs to navigate through this environment in a safe and approximately optimal way.

Navigating through this environment can be done in several ways. One option is to plan a trajectory and use a controller that can follow this trajectory. Another option is to use a controller that can simultaneously find and follow this trajectory. In this paper, we propose the use of a linear model predictive controller to navigate a car through a environment, while taking into account the road boundaries and other cars.

Since the number of different environment configurations a fully autonomous car can see are nearly endless, we limit are solve the a specific situation with specific constraints, shown in figure 1. This resembles a situation where an autonomous car is changing lanes, while taking into account other cars. The constraints should be constructed in such a way that the allowed state space remains convex. The other vehicles are therefore approximated by half-spaces inspired by [2].

A. Nonlinear System

A kinematic bicycle model is used to model the car, which is a nonlinear model with the following states: $[x, y, \psi, v]$ (obtained from [3] with some modifications). x, y represent the position of the car in 2D plane, ψ is the orientation of the car and v is the forward velocity of the car. The system has 2 inputs: $[a, \delta_f]$ which are respectively the acceleration and the steering angle. The nonlinear model is given in equation 1.

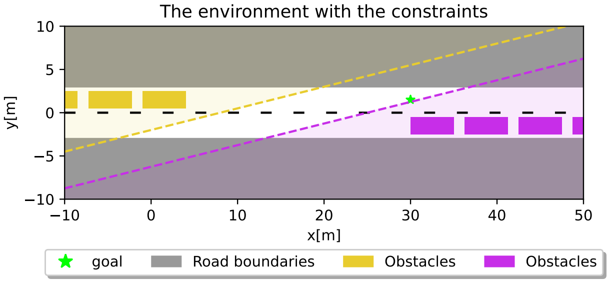

Fig. 1. The environment in which the model predictive controller is tested with the state constraints on the position visualized. The goal is not on the edge boundary, but slightly within the state space.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \cdot \cos\psi \\ v \cdot \sin\psi \\ \frac{v}{l}\tan\delta_f \\ a \end{bmatrix} \tag{1}$$

l is the length of the wheelbase and the point in the 2D describe by (x, y) is on the center of the rear axis. ψ is defined as the angle between the positive x-axis and the car where counter clockwise is positive.

B. Discrete Linearized System

Since we want to obtain a linear time-invariant system of the from described in equation 2, the nonlinear system needs to be linearized and discretized.

$$\begin{aligned} x^+ &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2}$$

First, the system is linearized around the point $[0, 0, 0, 3]^T$. When linearizing, it can be seen that the system needs to be linearized around a point with some forward velocity in order to be able to move into the y direction. Otherwise, the car can only move in a straight line, since there is no mathematical relation between steering and movement in the y-direction. The specific value for the velocity was found empirically. A too low value makes the model underestimate the heading change as a result of some steering input, while a too high value manifested the opposite behavior.

Christiaan Theunisse is a master students at TU Delft, The Netherlands. E-mail address: `C.Theunisse@student.tudelft.nl`.

[1] Github: https://github.com/christiaantheunisse/CarMPC.git

Subsequently, the model should be discretized to be able to use it in the model predictive controller. The forward Euler method is used to discretize the system, where the next state x^+ is estimated by the current state x and the derivative at that point $\dot{x}$ (equation 3).

$$x^+ = x + f(x,u)h \tag{3}$$

h is the time step used to integrate, so the smaller h, the more accurate the model is. However, a smaller h will also mean that the MPC needs a bigger horizon to be able to look ahead the same amount of time. The matrices A, B of the resulting system are given in equation 4. The matrix C depends on the controller that is used, a state feedback controller or an output feedback controller.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0.2 \\ 0 & 1 & 0.6 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0.171 \\ 0.2 & 0 \end{bmatrix} \tag{4}$$

II. Model predictive control design

In this section, the designed model predictive controllers are described. The task is to travel from the starting state to a certain reference state while taking into account the state and input constraints. We designed two types of controllers to perform this task:

- State feedback controller: It is assumed that the controller can observe the whole state. C from equation 2 is therefore equal to the identity matrix.
- Output feedback controller: The controller can only observe certain states, namely $\{x, y, v\}$, which means that the ψ is invisible. This is a realistic scenario because the velocity and the displacement in longitude direction can be easily obtained from the odometry. However, the movement and orientation in lateral direction are relatively difficult to measure accurately with the odometry. C from equation 2 is defined as:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

A. State Feedback MPC

The prediction horizon N was set to 20 with a sampling time of $0.2s$, which means a horizon of 4 seconds. The numbers were tuned empirically and are a trade-off between stability and computational effort, which is fairly low since algorithm runs at least twice as fast as real time with the visualization. A horizon much shorter than 4 seconds results in an infeasible problem, since the last state of the prediction horizon cannot be in the terminal set.

The system is subject to several constraints. Firstly, there are state constraints to make sure that the linearized systems remains valid. This are constraint on the states ψ and v, since other states have a nonlinear relation with these states, therefore deviating too much from the linearized state will hurt the stability. The constraints are given in equation 6.

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} x <= \begin{bmatrix} \frac{\pi}{8} \\ \frac{\pi}{8} \\ 5 \\ 1 \end{bmatrix} \tag{6}$$

Furthermore, there are some state constraints induced by the environment, namely the road and other cars. The constraints apply to a point on the car, so in practice it will be necessary to set the constraint with some offset from the road boundaries and other obstacles. The constraints are given in equation 7, namely, two road boundaries and two constraints resembling other vehicles.

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ -0.25 & 1 & 0 & 0 \\ 0.25 & -1 & 0 & 0 \end{bmatrix} x <= \begin{bmatrix} 3 \\ 3 \\ -2 \\ 6.25 \end{bmatrix} \tag{7}$$

The input of the system is also restricted in order to keep its application close to the linearized state and to keep them feasible from a physical perspective. The constraints are given in equation 8.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} x <= \begin{bmatrix} 2 \\ \frac{\pi}{8} \\ 2 \\ \frac{\pi}{8} \end{bmatrix} \tag{8}$$

The quadratic cost function (eq. 9) that is optimized in every time step, consist of a stage cost $l(x(k), u(k))$ and a terminal cost $V_f(x(N)$. The stage cost has the following values for $Q = \text{diag}(5, 5, 10, 10)$ and $R = \text{diag}(10, 100)$. The values are obtain by tuning an LQR controller with these matrices. First, the Q matrix was tuned in order to reach stable behavior and subsequently the R matrix to reduce the input values. The terminal cost is given by equation 9 where P is the solution of the discrete algebraic Riccati equation.

$$\begin{aligned} \ell(x(k), u(k)) &= x(k)^T Q x(k) + u(k)^T R u(k) \\ V_f(x(N) &= x(N)^T P x(N) \end{aligned} \tag{9}$$

The optimal control problem $P(x0, t)$ solved in every time step is given in equation 10. $\mathbf{u}_N$ is the sequence of inputs for the prediction horizon, which is optimized and $x0$ is the current state. $\mathbb{X}$ is defined by concatenating the constraints in equations 6 and 7 and $\mathbb{U}$ by the constraints in equation 8. $\mathbb{X}_f$ is the terminal set and is discussed in depth in section III-A.

$$\begin{aligned} \min_{\mathbf{u}_N} \quad & \sum_{k=0}^{N-1} \ell(x(k), u(k)) + V_f(x(N) \\ \text{s.t.} \quad & x(0) = x0 \\ & x(k+1) = Ax(k) + Bu(k) \forall k \\ & x(k) \in \mathbb{X} \quad \forall k \in \{0, 1, ..., N\} \\ & u(k) \in \mathbb{U} \quad \forall k \in \{0, 1, ..., N-1\} \\ & x(N) \in \mathbb{X}_f \end{aligned} \tag{10}$$

B. Output FeedBack MPC

The prediction horizon for the output based feedback controller is also set to $N = 20$ with $dt = 0.2s$. The same constraints on the state and input apply and no additional output constraints have been applied on the output. The model is different, see equation 11.

$$\begin{aligned} x^{+} &= Ax + Bu \\ y &= Cu \end{aligned} \tag{11}$$

Since the state cannot be observed, a observer is needed to estimate the state x. To ensure that the system is observable, the original system (A, C) needs to be observable. A Luenberger observer was tuned to observe the immeasurable state(eq. 12). The matrix L was tuned with the help of the pole placement method from [4].

$$\hat{x}^{+} = A\hat{x} + Bu + L(Y - C\hat{x}) \tag{12}$$

Since the reference point is now given by an certain output y_{ref}, the reference state and input (x_{ref}, u_{ref}) have to be determined. Besides the state constraints, it is know that the reference point should be a equilibrium point and the corresponding output should of course satisfy $y_{ref} = Cx_{ref}$. The optimal target selection (OTS) in equation **??** is solved once and the solution can be used for every time step. The stage cost is used as the objective.

$$(x_{ref}, u_{ref})(\hat{d}, y_{ref}) \in \\ \begin{cases} \underset{x_{ref}, u_{ref}}{\text{argmin}} & \ell(x(k), u(x)) \\ s.t. & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_{ref} \\ u_{ref} \end{bmatrix} = \begin{bmatrix} 0 \\ y_{ref} \end{bmatrix} \\ & (x_{ref}, u_{ref}) \in \mathbb{Z} \end{cases} \tag{13}$$

Nearly the same cost function is used as in the state feedback MPC (eq. 9. The only difference is that the MPC now tracks a set point (x_{ref}, u_{ref}) determine by the OTS, so the following changes were made: $x(\cdot) \rightarrow (x(\cdot) - x_{ref})$ and $u(\cdot) \rightarrow (u(\cdot) - u_{ref})$. The matrices Q, R and P remain the same.

III. Asymptotic stability

In this section, we show that the designed state based feedback MPC asymptotically stabilizes the linearized closed-loop system. Section IV will be used to verify that the designed linear MPC is able to stably control the nonlinear system. With this aim, we verify the Assumptions of 2.2, 2.3 and 2.14 and the fact that the origin lies within $\mathbb{X}$. According to Theorems 2.19 and 2.21 from [5], the origin is exponentially stable in $\mathcal{X}_N$, if the aforementioned assumptions are met. Before we will discuss each assumption separately, some basic assumptions need to be verified and the terminal set $\mathbb{X}_f$ has to be defined.

First of all, the model derived in equation 4 is controllable. This can be verified by observing that the following matrix is full rank: $C = [B, AB, A^2B, A^3B]$. Furthermore, the matrices Q and R in the stage cost are both positive definite and P in the terminal cost is the solution of the discrete ARE equation (both eq. 9).

A. Terminal set $\mathbb{X}_f$

$\mathbb{X}_f$ is the control invariant set for the linear system. The unconstrained control law K can be used within this set, because it is constructed to satisfy all the state and input constraints when using $u = Kx$, where $K = -(R + B^TPB)^{-1}B^TPA$. $\mathbb{X}_f$ was obtained by taking the intersection of the following two set:

1) $\{x \in \mathbb{X} \mid x \in \mathbb{X}_f \implies A_kx \in \mathbb{X}_f\}$ which is the set of all states within the state constraints that stay within $\mathbb{X}_f$ when the unconstrained control law $u = Kx$ is used. This set is obtained by using algorithm 1, which is a modified version of the algorithm proposed by [6]. The state constraints have been translated in order for the reference state to lie on the origin, since the reference state does not lie on the origin (see figure 1). The algorithm gives a set of 42 constraints of the form $Ax <= b$. Since this set does not satisfy the input constraints, the following set is needed to provide some additional constraints.

2) $\{x \in \mathbb{X} \mid Kx \in \mathbb{U}\}$ which is the set of states that satisfies the input constraints $\mathbb{U}$ when the unconstrained control law $u = Kx$ is used. This is fairly easy and requires some rewriting after which the following constraints are found that describe this set: $A_uKx <= b_u$ (A_u and b_u from eq. 8).

The intersection of the sets can be easily obtained by concatenating the constraint matrices. Besides, it has be verified that the origin is an interior point of the terminal set.

Algorithm 1: Algorithm to compute the invariant set

```
Input:
    $A_k = A + BK$
    $f(x) = A_xx - b_x$ (from $A_xx <= b_x$ describing $\mathbb{X}$)
For $i$ = 0, 1, ..., s
    $x_i^* = \begin{cases} \underset{x}{\arg\max} & f_i(A_K^{k+1}x) \\ \text{s.t.} & f_j(A_K^kx) \leq 0, \\ & \forall j \in \{1, 2, ..., s\}, \\ & \forall t \in \{0, 1, ..., k\} \end{cases}$
End
If $f_i(A_K^{k+1}x_i^*) \leq 0 \forall i \in \{1, 2, ..., s\}$ then
    $k^* = k$
    $\mathbb{X}_f = \{x \in \mathbb{R}^n \mid f_i(A_K^t) \leq 0,$
                     $\forall \in \{1, 2, ..., s\},$
                     $\forall t \in \{0, 1, ..., k*\}\}$
    Break
Else
    $k = k + 1$
```

B. Assumption 2.2

The function f, defined as $x^{+} = f(x,u) = Ax + Bu$, is continuous and satisfies $f(0,0) = 0$. The same is true for the stage cost ℓ and the terminal cost V_f, which are both continuous and zero at the origin.

C. Assumption 2.3

The set $\mathbb{Z} = (\mathbb{X} \times \mathbb{U})$ is closed since all constraints are defined as less or equal to. $\mathbb{X}_f$ and $\mathbb{U}$ are both compact and contain the origin.

D. Assumption 2.14

a) Since $\mathbb{X}_f$ is computed by algorithm 1, it must be invariant. To verify the working of the algorithm, the control invariance of the terminal set is verified a prior. Several runs are started at a point in the terminal set, whereby it is checked that the linear model converges towards to origin without leaving the terminal set meanwhile.

Furthermore, the Lyapunov decrease condition should apply to the terminal cost. When the terminal cost and stage cost $V_f(x(N))$ are chosen as in equation 9 and P is the solution of the discrete Riccati equation, the Lyapunov decrease condition is met as shown by [5] in section 2.5.4. Given that $P = A_K^T P A_K + Q_K$, $A_K = A + BK$, $Q_K = Q + K^T R K$ and $u = Kx$, the proof is shown in equation 14 This constraint is also verified a posteriori and the result of one of the runs is plotted in figure 2. It might not be visible in the plot but the lines are exactly plotted over each other. These results are obtained from a test run within the terminal set $\mathbb{X}_F$. Different start positions within the terminal set and different horizon settings show the same results. This is not suprising since this was already shown in equation 14

$$\begin{aligned}
\textit{Assumption: } V_f(f(x,u)) &\leq V_f(x) - \ell(x,u) \\
V_f(A_K x) &\leq V_f(x) - \ell(x,u) \\
x^T A_K^T P A_K x &\leq x^T P x - (x^T Q x + x^T K^T R K x) \\
x^T A_K^T P A_K x &\leq x^T P x - x^T Q_K x \\
x^T (A_K^T P A_K + Q_K) x &\leq x^T P x \\
x^T P x &\leq x^T P x
\end{aligned} \tag{14}$$

b) This can be shown mathematically as is done in equation 15. The inequality in the second line holds, because R is positive definite.

$$\begin{gathered}
\textit{Assumption: } \quad \ell(x,u) \geq \alpha_1(|x|) \\
x(k)^T Q x(k) + u(k)^T R u(k) \geq x(k)^T Q x(k) = \alpha_1(|x|)
\end{gathered} \tag{15}$$

The second assumption can also be verified mathematically. The terminal cost can just be multiplied by a factor ≥ 1 since P is positive semi-definite, see equation 16.

$$\begin{aligned}
\textit{Assumption: } \quad V_f(x) &\leq \alpha_f(|x|) \\
x^T P x &\leq \lambda * x^T P x = \alpha_f(|x|) \quad \text{where} \quad \lambda \geq 1
\end{aligned} \tag{16}$$

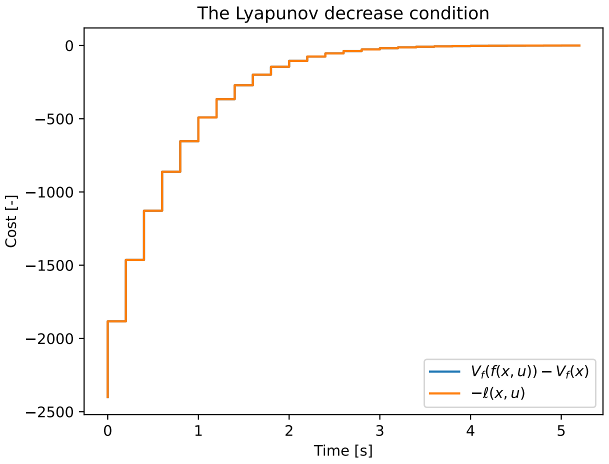

Fig. 2. The Lyapunov decrease condition is verified for the terminal cost. In the figure $V_f(f(x,u)) - V_f(x)$ is plotted against $\ell(x,u)$, the stage cost.

E. Region of attraction $\mathcal{X}_N$

The region of attraction can be obtained empirically by performing a grid search over all the dimension and subsequently converting it to a polyhedron. However, in order to have 10 points per dimension the total number of calculations will be $10^4 = 10000$. Although, this is doable when one really needs this set, we will restrict ourselves to a grid search over 2 states to get a impression what the region of attraction looks like and how it can be computed. We will vary the initial x-coordinate of the position x and velocity v. The other states y and ψ are fixed at 0. When the simulation converges to the origin, the point lies within $\mathcal{X}_N$ and otherwise it lies outside the set of controllable points. The resulting set is shown in figure 3.

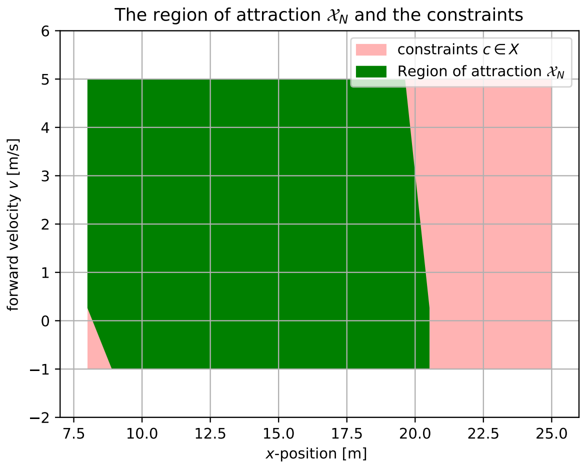

Fig. 3. A visualization of the region of attraction $\mathcal{X}_N$ (green) and all the points within the state constraints (red). The sets are show for the states x and v.

IV. Numerical simulations

In this section we will perform several numerical simulations with our developed MPC controllers. The controllers will be tested on the nonlinear model, where zero order hold is assumed. This means that the control input remains constant for $0.2s$ until the MPC controller updates the control input. We will study the influence of the horizon length N. Afterwards we will compare the state feedback based MPC to a unconstrained LQR controller. Finally, we will verify the performance on the observer in the output based feedback controller.

When we compare the performance of the state feedback controller for different horizons, there is not difference when the terminal constraint is enforced. This is to be expected, because a longer horizon will just be able to look further into the terminal set. However, since the MPC always performs optimally within the terminal set, this does not influence the performance. This has to do with the reason that the terminal set is constructed in such a way that the unconstrained LQR control law can be used inside this set. However, when the terminal constraint and the terminal cost is removed, the horizon can be shorter and a clear difference is seen. In figure 4 it can be seen that the models with a longer horizon have a lower overall cost at the end. High control inputs are costly so the model with the horizon $N = 4$ avoids this and has a very low and smooth input. The models with the longer horizons $N = 8, 12$ can already 'see' that the high control input will be rewarded by a lower state cost in the future.

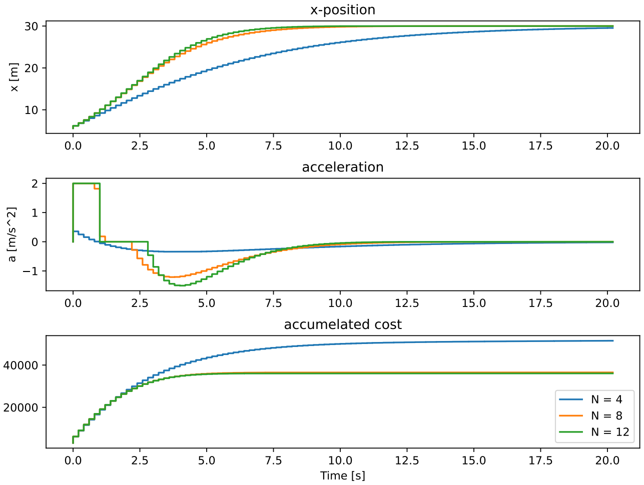

Fig. 4. A visualisation of a state, an input and the accumulated stage cost for models with different horizons.

Since the LQR cannot handle constraints, a situation was chosen where the model stayed within the state constraints. The control output of the LQR was clipped to satisfy the control input constraints. When the accumulated cost is compared, it can be seen that the MPC performs as good as the LQR which uses the unconstrained control law (figure 5). So the MPC, finite horizon, performs as good as the LQR, infinite horizon. The advantage of having the same performance for a finite horizon approach is that constraints can be taken into account.

As described in section II the output feedback controller uses a observer to estimate the values of the states. In figure 6 to estimated value is compared to the real value. For most states, the estimation is considerably well except for the state ψ. This is easy to explain since all the states are directly in the output vector except for ψ. Besides, since the system is linearized it has trouble controlling the steering angle in a stable way.

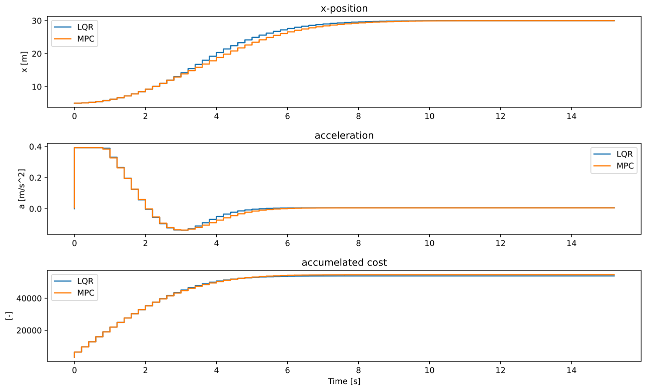

Fig. 5. The MPC and a unconstrained LQR are compared, showing that the MPC performs as good as the LQR.

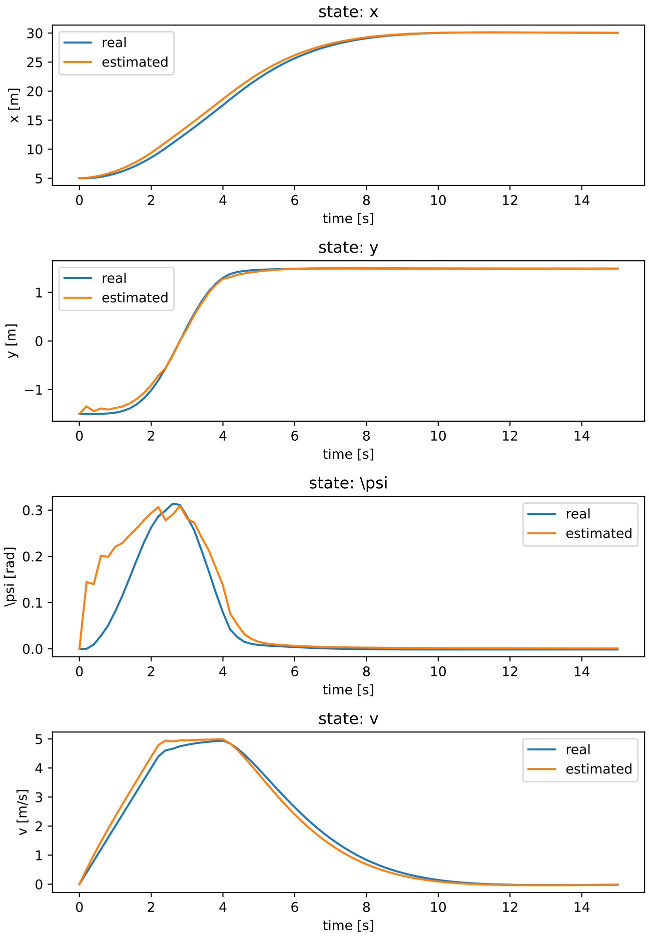

Fig. 6. The estimated value for the states in the output feedback controller is shown for each state in this figure together with the real values.

During the numerical simulations it was observed that the system stabilized well with a nearly optimal path, when the terminal constraint was lifted. It should therefore be fairly easy to empirically show stability without a terminal constraint while the terminal cost is probably increased by a constant. This would be an recommendation for future research, since the computational cost of the controller can be reduced by removing this constraint.

Another phenomena that was observed during the simulations on the nonlinear model, is that the linear controller tended to steer to close to the boundaries of the environment, thereby making the problem infeasible. This has to do with the fact that the model is linearized around a certain speed. When the actual speed is below the linearization speed, the model overestimates the heading change as a result of it steering input. When the speed is higher then the linearization speed, the opposite would happen. This would result in the controller planning to take a optimal path close to a boundary, but in reality steering too much toward the boundary. The next time step the model has already steered too much, rendering the problem infeasible. This problem could probably be solved by adding an extra cost for traveling to close to the boundaries and thereby creating an offset between the optimal path and the boundaries. This is also a recommendation for future research.

References

[1] L. Cloak, "Autonomous Driving Startups Raise Record Funding As The Push For Commercialization Begins," 10 2021. [Online]. Available: https://www.cbinsights.com/research/autonomous-vehicle-tech-funding-trends/#:~:text=Autonomous%20driving%20tech%20companies%20have,than%20double%20last%20year's%20total.

[2] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A distributed multi-robot coordination algorithm for navigation in tight environments," *arXiv preprint arXiv:2006.11492*, 2020.

[3] C. Lundquist, W. Reinelt, and O. Enqvist, "Back driving assistant for passenger cars with trailer," *ZF Lenksysteme GmbH, Schwäbisch Gmünd, Germany*, pp. 1–8, 2006.

[4] A. Varga, "Robust pole assignment via sylvester equation based state feedback parametrization," in *CACSD. Conference Proceedings. IEEE International Symposium on Computer-Aided Control System Design (Cat. No. 00TH8537)*. IEEE, 2000, pp. 13–18.

[5] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2008.

[6] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: the theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control*, vol. 36, pp. 1008–1020, 1991.