5LMB0 – Model Predictive Control

Project Report

Mert Eyuboglu – 1599925

10.05.2021

1. State space model derivation and simulation

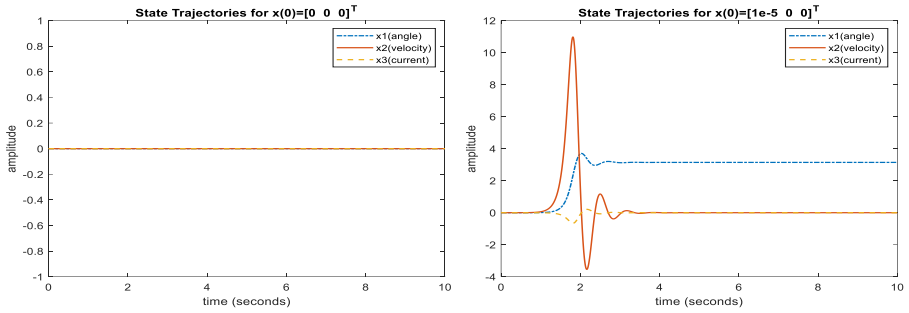
The behaviour of the electrically driven inverted pendulum system is described by the two given differential equations

$$J\frac{d\dot{q}}{dt} = mgl\,.sin\big(q(t)\big) - b\,.\;\dot{q}(t) + k\,.\;i(t) \tag{1}$$
$$L\frac{di(t)}{dt} = -k\,.\;\dot{q}(t) - R\,.i(t) + u(t)$$

and the state vector is given to be $x(t) = [x_1(t)\;\; x_2(t)\;\; x_3(t)]^T = [q(t)\;\; \dot{q}(t)\;\; i(t)]^T$. By replacing $q(t), \dot{q}(t), i(t)$ with $x_1(t), x_2(t), x_3(t)$ respectively in (1) and rearrenging the terms we get

$$\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= \frac{mgl}{J}\sin\big(x_1(t)\big) - \frac{b}{J}x_2(t) + \frac{k}{J}x_3(t)\\ \dot{x}_3(t) &= -\frac{k}{L}x_2(t) - \frac{R}{L}x_3(t) + \frac{1}{L}u(t)\end{aligned} \tag{2}$$

which is the state space model of the system. By inspecting equation (2) it can be seen that $\dot{x}(t) = 0$ for $u(t) = 0$ and $x(t) = [0\;\; 0\;\; 0]^T$ which means that this an equilibrium point of the system. To inspect the stability of this equilibrium point the system is simulated using the provided Simulink model with $u(t) = 0$ with two initial conditions $x(0) = [0\;\; 0\;\; 0]^T$ and $x(0) = [10^{-5}\;\; 0\;\; 0]^T$. The simulation results are presented in Figure 1.



(a) (b)

Figure 1. State trajectories for u(t) = 0 and (a) x(0) = $[0\;\; 0\;\; 0]^T$ (b) x(0) = $[10^{-5}\;\; 0\;\; 0]^T$

From Figure 1 it can be seen that a small perturbation of the first state from the equilibrium point, results in a different steady state value for the system which is the other equilibrium point of the system. This means that $x = [0\;\; 0\;\; 0]^T$ is an unstable equilibrium point of the system. This correlates with intuition, as we know that $x(0) = [0\;\; 0\;\; 0]^T$ is the pendulum up position and if any disturbance is applied to the system at this position the pendulum would fall down finally reaching steady state at the down position corresponding to the steady state value in (b) that is $x_{ss} = [\pi\;\; 0\;\; 0]^T$. As we also now, if we start at the pendulum down position such that $x(0) = [\pi\;\; 0\;\; 0]^T$ the system will converge to the same initial state if a disturbance is applied to the pendulum. Thus, we can claim that $x = [\pi\;\; 0\;\; 0]^T$ is the stable equilibrum point of our system. (Figure 1.a proves $[0\;\; 0\;\; 0]^T$ is a equilibrium point as all states remain at zero when there is no perturbation).

2. Linear MPC design with stability and recursive feasibility guarantees

a) The systms is linearized around the given steady state $x_{ss} = [1.5707\;\; 0\;\; -1.1445]^T$ and the corresponding input $u_{ss} = -1.1445$ by using a Taylor expansion of the nonlinear state equations i.e.

$$f(x(t),u(t)) \approx f(x_{ss},u_{ss}) + \frac{\partial f(x(t),u(t))}{\partial x}\Big|_{\substack{x_{ss}\\u_{ss}}}(x(t)-x_{ss}) + \frac{\partial f(x(t),u(t))}{\partial u}\Big|_{\substack{x_{ss}\\u_{ss}}}(u(t)-u_{ss}) \tag{3}$$

As explained in the porvided document 'Supporting Material Linearization' this method enables us to obtain a linear state space model of the system in the form of $\dot{x}(t) = Ax(t) + Bu(t)$ where

$$A = \frac{\partial f(x(t),u(t))}{\partial x}\Big|_{\substack{x_{ss}\\u_{ss}}}, \qquad B = \frac{\partial f(x(t),u(t))}{\partial u}\Big|_{\substack{x_{ss}\\u_{ss}}} \tag{4}$$

In order to perform this linearization method in MATLAB first the state equations are defined using symbolic state varibles. Then, Jacobian matrices for these set of equations are obatined using the 'jacobian' command in MATLAB. Finally by substuting the given x_{ss} and u_{ss} values into these Jacobian matrices the A and B matrices of the linearized continous time state space model are obtained as

$$A = \begin{bmatrix} 0 & 1 & 0\\ 0.0094 & -1.4286 & 85.7143\\ 0 & -60 & -1000 \end{bmatrix}, \quad B = \begin{bmatrix} 0\\ 0\\ 1000 \end{bmatrix} \tag{5}$$

Finally, the resulting continous time linear model is discretized by the 'c2d' command in MATLAB with a sample time of $0.004\,ms$. It should also be noted that an affine term is introduced due to linearization around a non-zero equilibrium point as we have not defined new stated variables as $x_{new} = x - x_{ss}$ and $u_{new} = u - u_{ss}$ but rather used the original state varibles. This afine term will be taken into account during the MPC design.

b) Now that we have our linear discrete time model, a linear MPC with stability and recursive feasibility guarantees can be designed such that the system stabilizes at x_{ss} and u_{ss}. For the given Q, R and N, it is requested that LQR controller is used as the terminal controller and the corresponding P matrix from the discrete time Ricatti equation is used as the terminal cost. The K_{lqr} and P_{lqr} corresponding to the given Q and R are obtained in MATLAB by

```
Q = [12 0 0; 0 1 0; 0 0 1];
R = 1;
[K,P_lqr,~] = dlqr(A,B,Q,R);
K_lqr = -K;
```

As LQR has stability guarantees, it is know that using P_lqr also implies the stability of our MPC controller and K_{lqr} can be used as a stabilizing terminal controller. Then the prediction matrices can be obtained as we did in the seminars throughout the course. Next the terminal set has to be calculated to be included as a terminal state constraint to guarantee recursive feasibility. The terminal set must be constraint admissable and invariant. In order to obtain this terminal set in MATLAB using the MPT Toolbox, first the state constraints are defined as a polytop.Then, a second polytop is defined such that $K_{lqr} * x$ is within the input constraints and then the constraints admissable set is obtained as an intersection of these two polytops. Next the largest possible inariant set that is a subset of the constarints admissable set is obtained which is our terminal set. The expalined procedure is implemented in MATLAB as follows:

```
% closed loop lqr
A_cl = A + B*K_lqr;
% Compute the invariant set with constraint admissible set
model = LTISystem('A', A_cl,'Ts',Ts);
% state constraints (A_x * x <= b_x)
X_set = Polyhedron([eye(n);-eye(n)],[xmax; -xmin])-x_ss;
% constraints admissible set
CA_set = Polyhedron([eye(m);-eye(m)]*K_lqr,[umax; -umin]);
% input constraint admissible set
IA_set = X_set&CA_set;
% invariant set with CA set (terminal set)
INVset_mpc = model.invariantSet('X',IA_set);
```

It should be noted that the state constarints are shifted by $-x_{ss}$ which is due to the fact that the states fed to the contorller are also shifted by $-x_{ss}$ such that the original states don't converge to the origin but instead they converge to the given x_{ss}. In order to guarantee that the original states are within the constaints the polytopic set is shifted. (The plots are shifted back by $+x_{ss}$ such that they are for the original states). The plot of the projection of feasible set of states on x_1 and x_2 for the described terminal set and $N = 4$ is presented in Figure 2. In Figure 2, the initial condition $x(0) = [5.9832\ \ 17.7\ \ -1.549]^T$ is also marked with a '*' and it can be conluded that $x(0)$ is not a feasible state as it is not inside the projection area colored by red.

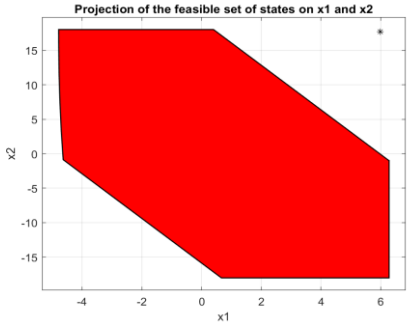
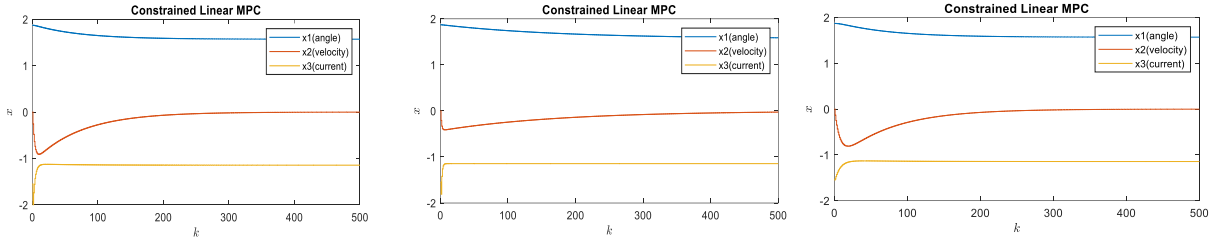


Figure 2. Projection of the set of feasible states on x_1 and x_2 with the initial condition $x(0) = [5.9832\ \ 17.7\ \ -1.549]^T$ marked by '*'

As the given initial condition $x(0) = [5.9832\ \ 17.7\ \ -1.549]^T$ is not inside the set of feasible states it leads to an infeasible problem. Thus, no simulation result for this initial condition can be presented. To verify the correctness of the MPC algorithm simulation results with feasible initial states are presented in Appendix A. By inspecting the Figures in Appendix A it can be claimed that the implemented controller is successful at stabilizing the system at the desired value for inital conditions that are within

the feasible set of states. But it also possible to see the difference between the trajectories when the linearized system or the nonlinear system in Simulink is used for simulations. When the nonlinear model is used, due to the mismatch between the prediction model and the actual model the performance of the controller drops. The amount of performance drop gets more pronounced when the states are away from the point that we linearized the system and the controller performs much better at states closer to the linearization point. This can also be verified by the Figures in Appendix A such that when simulated with the initial condition $x(0) = [1.8708 \;\; 0 \;\; -1.549]^T$ the trajectories with the nonlinear model is pretty close to the one with the linear model. However, when the initial condition is changed to $x(0) = [-4 \;\; 15 \;\; -1.549]^T$ it obvious that the difference between the trajectories are more pronounced and the settling time is much longer.

c) As no limitations are described regarding the input other than the input constraints the first action to take in order improve performance would be to remove the cost of the input from our cost function. To do so, R is updated as $R = 0$. As it canbe seen Figure 3 this immediately gives shorter settling times when compared to the Figure 6 in Appendix A.



(a) (b) (c)

Figure 3. State trajectories (a) Q=diag(12,1,1), R=0 (b) Q=diag(12,6,1), R=0 c) Q=diag(12,1,6), R=0

In order to see, individual effects of increasing the cost of x_1 and x_2 diagonal term corresponding to that state is increased to 6 while other is kept as 1 and results in Figure 3.b and 3.c are obtained repsctively. As it can be seen in Figure 3 increasing the cost of x_2 improved the trajectory of x_2 considerably but increased the settling time of x_1 as well. On the other hand, increasing the cost of x_3 results in a fast initial increase of x_3 but the trajectory of x_2 is much worse. Thus, boosting up the cost of x_1 while with respect to the cost of x_2 and x_3 will give us the best performance. Thus, we must increase $Q(1,1)$ up to the point where $x(0)$ is still inside the feasible set while $Q(2,2) = Q(3,3) = 1$. By doing so we end up with $Q = diag(5000,1,1)$ and $R = 0$. A plot of the new feasible set of states along with the initial condition is presented in Appendix B which shows that $x(0)$ is pretty close to the set boundary with the current Q and R. The corresponding simulation results with the linear and nolinear model are presented in Figure 5 and 6 respectively.

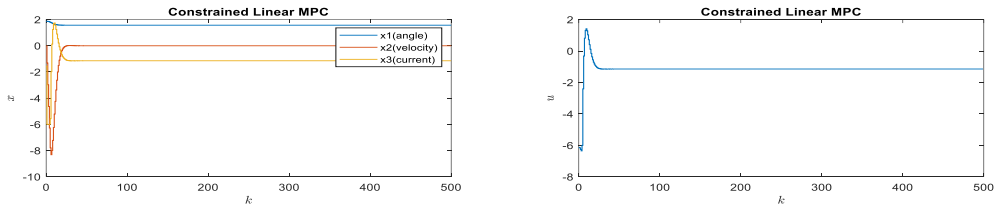


Figure 4.a Linear MPC simulation with linearized model

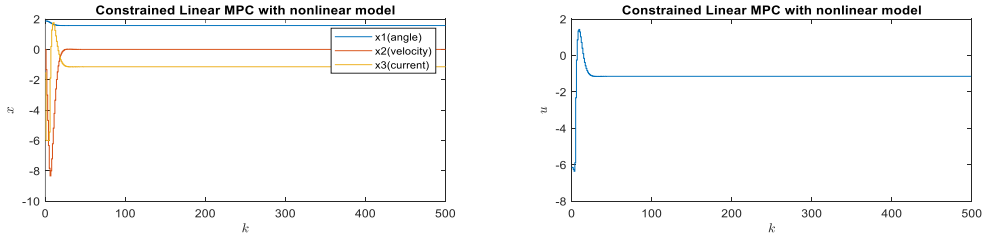


Figure 4.b Linear MPC simulation with linearized model

By comparing Figure 4 with Figure 6 in Appendix it can be seen that the states converge much faster to their steady state values while a much more aggressive control action is applied.

3. Nonlinear MPC design using an LPV approach

The nonlinear state space equations in (2) are discretized with the approximation $x(k+1) = x(k) + T_s \,.\, f(x(t), u(t))$ which results in

$$\begin{aligned} x_1(k+1) &= x_1(k) + T_s x_2(k) \\ x_2(k+1) &= x_2(k) + T_s \frac{mgl}{J}\sin\big(x_1(k)\big) - T_s \frac{b}{J} x_2(k) + T_s \frac{k}{J} x_3(k) \\ x_3(k+1) &= x_3(k) - T_s \frac{k}{L} x_2(k) - T_s \frac{R}{L} x_3(k) + T_s \frac{1}{L} u(k) \end{aligned} \tag{6}$$

By extracting the nonlinear sin term and including it as an LPV affine term we obtain the following LPV system with an affine term

$$x(k+1) = \begin{bmatrix} 1 & T_s & 0 \\ 0 & 1 - T_s \frac{b}{J} & T_s \frac{k}{J} \\ 0 & -T_s \frac{k}{L} & 1 - T_s \frac{R}{L} \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ \frac{T_s}{L} \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ T_s \frac{mgl}{J}\sin\big(x_1(k)\big) \\ 0 \end{bmatrix} \tag{7}$$

To design a controller for the system given by (7) the LPV MPC using iterative quadratic programming method covered in Lecture 6 is combined with the reference tracking MPC method covered in Lecture 7. This time as we have parameter varying model, the coordinate shift approach that we have implemented in the previous question is replaced with the modified cost function presented in Lecture 7 slide 8. By using this cost function and the model in (7) the prediciton matrices and constraints are formulized as in the 'Tracking MPC Hints' document. Due to its parameter varying varying structure the affine term has to be updated in an iterative manner at each time step. Affine terms are computed, according those the x_{ss} and u_{ss} terms throughout the prediction horizon are calulated by the equation in Lecture 7 slide 6. To mantain the invertability of the matrix in this expression C matrix is chosen to be $[1 \;\; 0 \;\; 0]$ and the reference is selected as $r = 1.5707$. After this procedure the reulting quadratic problem is solved but instead of applying the calculated input directly, first the affine term is updated with the predicted trajectory and the optimization problem is solved again according to the new affine term. This is repeated until the infinity norm of the difference of two consecuitve input trajectories get smaller than 10^{-5}. Then the first element of the predicted input sequence is applied and a time step is taken. The simulation result of this controller via the nonlinear Simulink model for $x(0) = [4.7124 \quad 17.7 \quad -1.549]^T$ is presented in Figure 5.

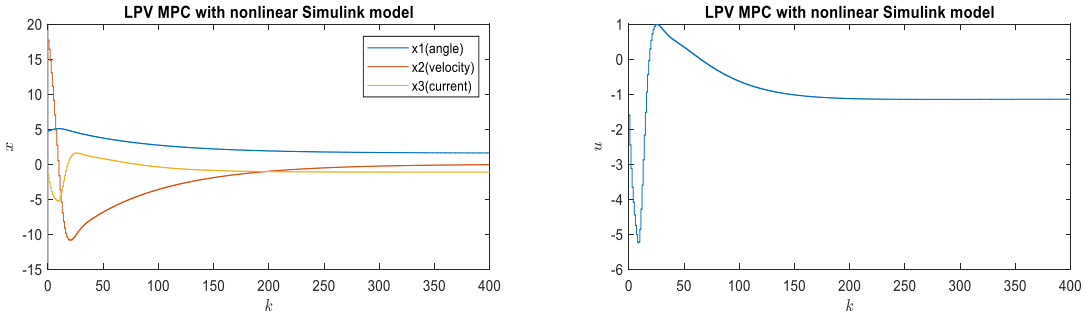


Figure 5. LPV based MPC simulation with nonlinear model and $x(0) = [4.7124 \quad 17.7 \quad -1.549]^T$

a) When simulated with the nonlinear model, the average computation tme of the LPV based MPC controller is found to be 0.0073 seconds while the avarage computation time of the linear MPC is 0.028 seconds per time step. This is the expected result and it is due to the fact that at each time step a few iterations are done in the LPV approach until the parameters are estimated accurate inof and then the input is applied. This means that at each iteration a quadratic program is solved while in the linear MPC there are no iteration at time steps thus only one quadratic program is solved per time step.

Appendix A: Simulation results of the linear MPC with feasible initial states

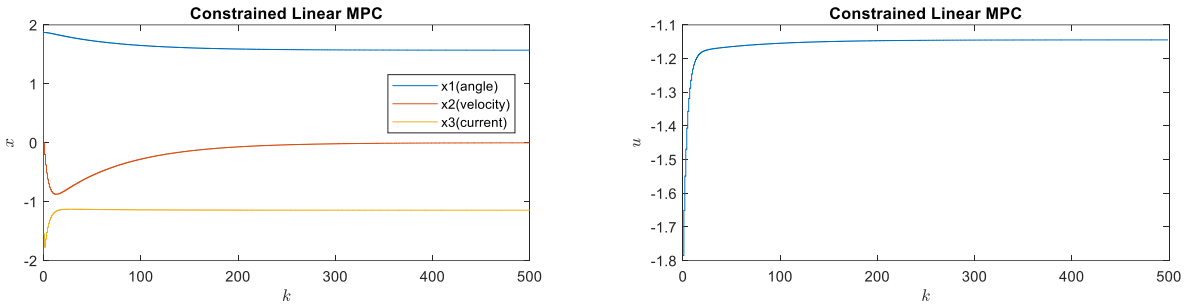


Figure 6. Linear MPC simulation with linearized model and $\mathrm{x}(0) = [1.8708 \;\; 0 \;\; -1.549]^{\mathrm{T}}$

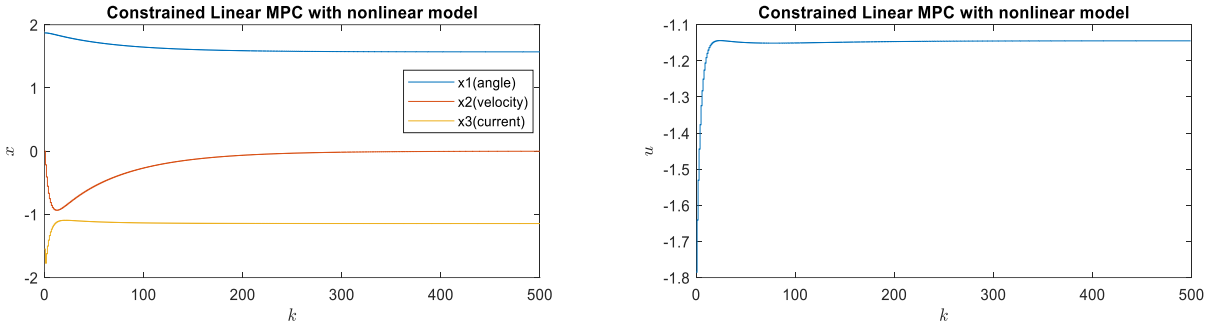


Figure 7. Linear MPC simulation with nonlinear model and $\mathrm{x}(0) = [1.8708 \;\; 0 \;\; -1.549]^{\mathrm{T}}$

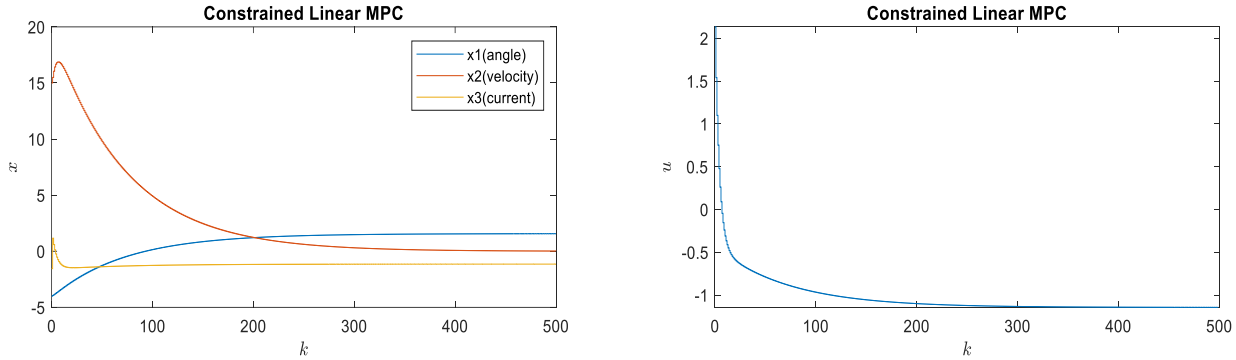


Figure 8. Linear MPC simulation with linearized model and $\mathrm{x}(0) = [-4 \;\; 15 \;\; -1.549]^{\mathrm{T}}$

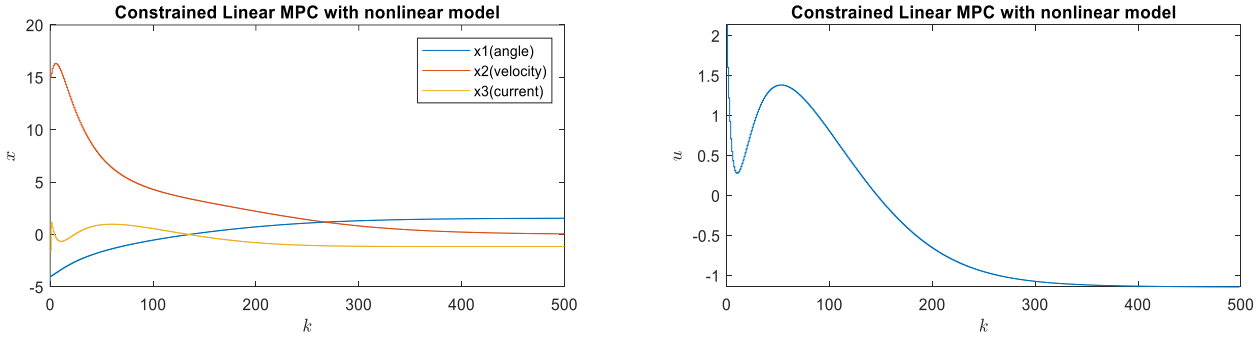


Figure 9. Linear MPC simulation with nonlinear model and $\mathrm{x}(0) = [-4 \;\; 15 \;\; -1.549]^{\mathrm{T}}$