Technische Universiteit **Eindhoven** University of Technology

Department of Electrical Engineering

Model predictive Control (5LMB0)

Homework Assignment

Student name:	**Student ID:**
Job Meijer	1268155
Marcel van Wensveen	1253085

Version 1

Eindhoven, May 2019

1 Part 1

1.1 Assignment 1.1

In this part of the project the "Van der Pol" oscillator is examined. The continuous-time model of the oscillator is given by equation 1.1.

$$\ddot{p}(t) = \mu(1 - p^2(t))\ddot{p}(t) - p(t) + u(t) \tag{1.1}$$

The given continuous-time model of equation 1.1 is rewritten to the nonlinear state space as model shown in equation 1.2. This is done by defining the states $[p(t), \dot{p}(t)]^T$ and rewriting the equation.

$$\begin{bmatrix}\dot{p}(t)\\ \ddot{p}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -1 & \mu(1-p^2)\end{bmatrix}\begin{bmatrix}p(t)\\ \dot{p}(t)\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}u(t) \tag{1.2}$$

Here $p(t)$ denotes the position, $\dot{p}(t)$ the velocity, $\ddot{p}(t)$ the acceleration, $u(t)$ the input and parameter $\mu = 2$.
The state and input constraints are given as:

$$\begin{bmatrix}-2\\ -5\end{bmatrix} \leq p(t) \leq \begin{bmatrix}2\\ 5\end{bmatrix}, \quad [-1 \leq u(t) \leq 1] \tag{1.3}$$

This model is simulated in MATLAB Simulink with initial conditions $x(0) = [1\ 0]^T$ and zero input. The results of this simulation are shown in Figure 1.1. In those plots it can be seen that the system ends up in a limit cycle.

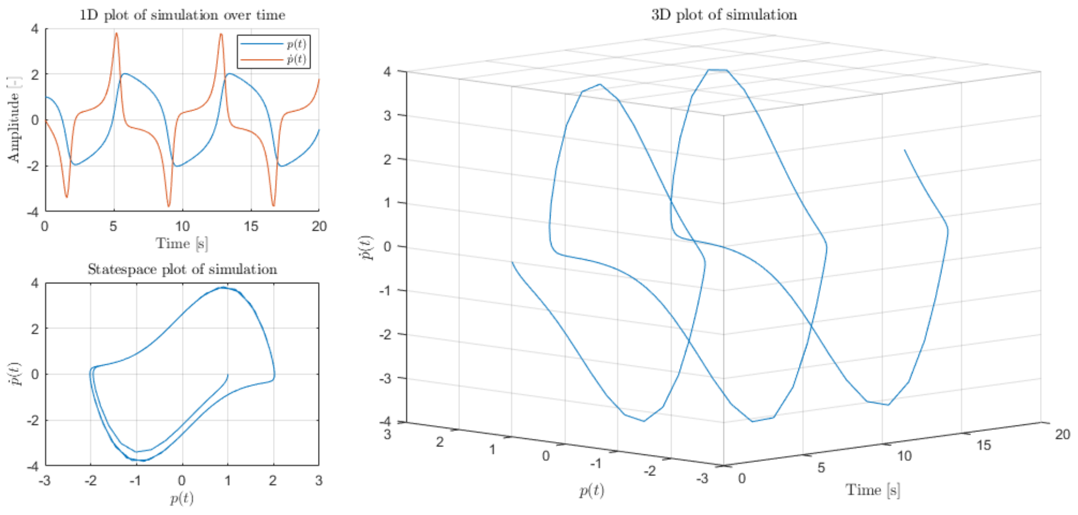


Figure 1.1: Simulation of nonlinear model

1.2 Assignment 1.2

The goal in this section is to design MPC controllers to regulate system 1.2 to zero with zero input ($\mathbf{x} = u = 0$).
An MPC controller with the guarantee of stability and recursive feasibility is designed using the following settings:

$$N = 50, Q = \begin{bmatrix}1 & 0\\ 0 & 4\end{bmatrix}, R = 0.1. \tag{1.4}$$

Where N is the prediction horizon, Q is the cost matrix for states and R is the cost for inputs. An LQR controller is used as terminal controller and the P from the discrete time Riccati equation is

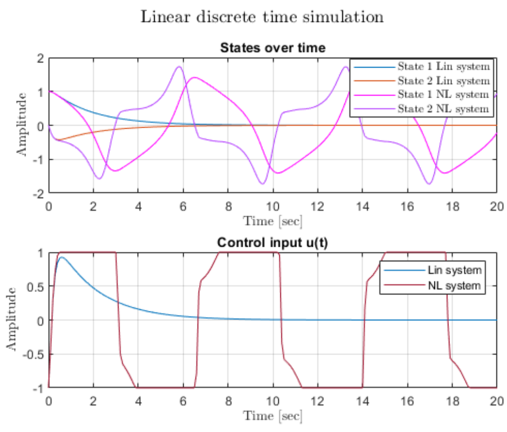
used as the terminal cost to guarantee stability. Also the invariant set is calculated and used as terminal constraint for the controller to guarantee recursive feasibility of the solution.
First the model is linearized around $p = [1; 0]$, resulting in the following state-space model.

$$\begin{bmatrix} \dot{p}(t) \\ \ddot{p}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} p(t) \\ \dot{p}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{1.5}$$

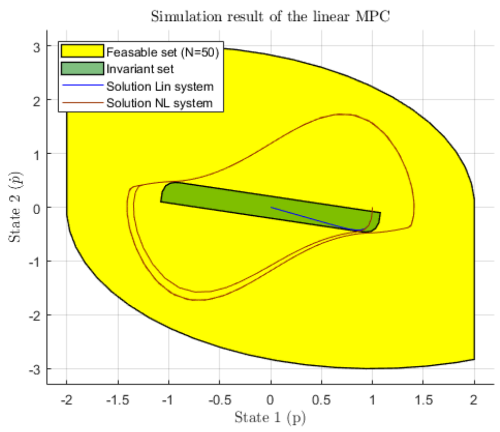
and discretized using *c2d.m* in MATLAB. Simulating this model with a linear MPC controller will result in a stable system, ending up in the origin as desired. When simulating the linear MPC controller with the nonlinear model will result in the same limit cycle as found in assignment 1.1. This is because the linear model differs too much from the non-linear model and does not generate a strong enough input signal to force the system to the origin. Both of those simulations are found in figure 1.2.
By re-tuning the cost matrices to

$$Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}, R = 0.01. \tag{1.6}$$

the nonlinear system goes to the origin as shown in figure 1.3. In this case the velocity is kept lower (due to a different Q, resulting in a higher cost on velocity compared to the position and control input) and as a result of that the system does not end in the limit cycle and the input is strong enough to force the system to the origin. When linearizing the model around $x = [0\ 0]$ the system becomes unfeasible since x(0) lies outside the feasible set, as shown in figure 1.4. Therefore it is not possible to calculate a solution.

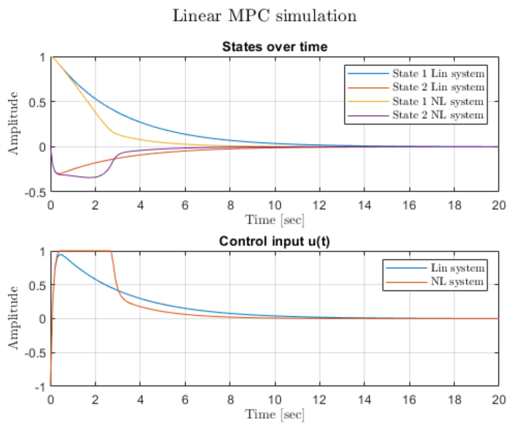


(a) Time domain plots

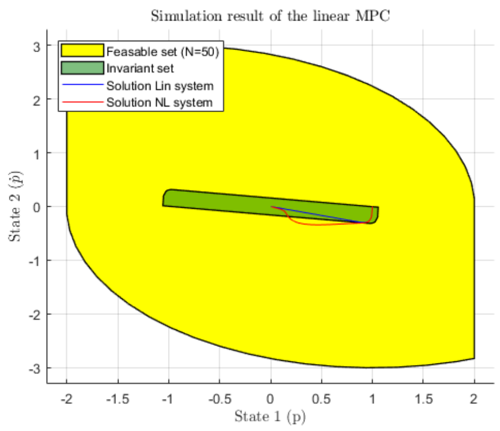


(b) State space plots

Figure 1.2: Linear MPC with given cost

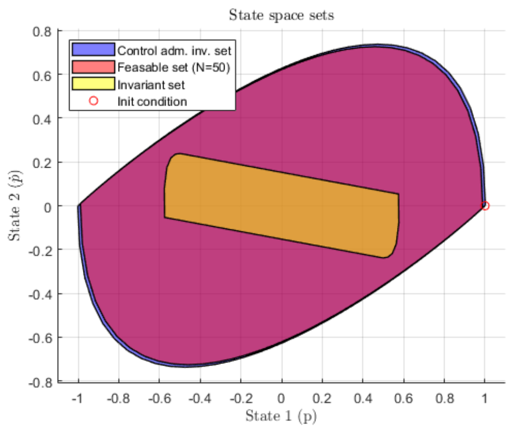


(a) Time domain plots

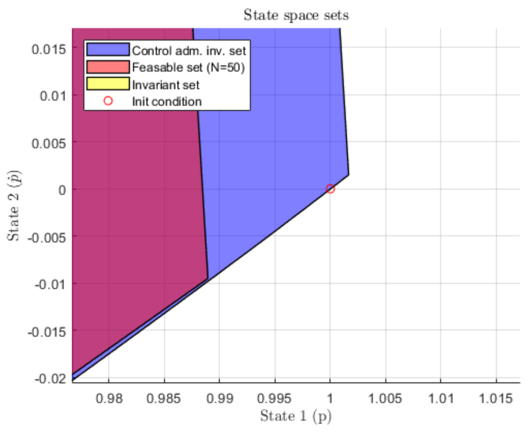


(b) State space plots

Figure 1.3: Linear MPC with re-tuned cost



(a) MPC sets of system linearized around x = 0,0



(b) Zoomed around initial condition

Figure 1.4: Invariant, Feasible and control admissible sets with init conditions

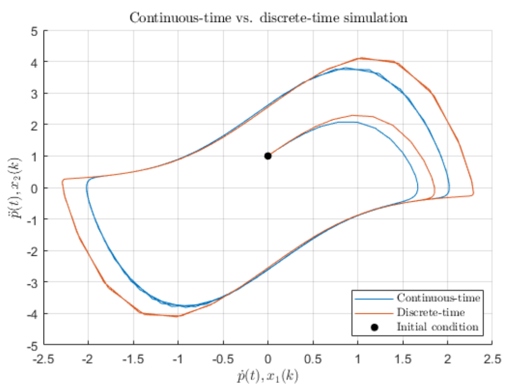
1.3 Assignment 1.3

Now a nonlinear MPC controller is designed to regulate the system which is based on the discrete-time approximation of the nonlinear system with $T_s = 0.1$, resulting in 1.7.

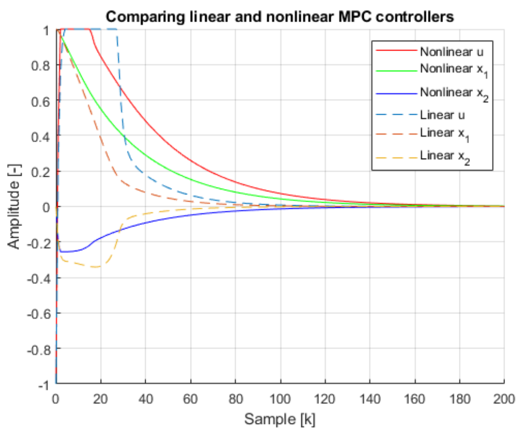
$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ -0.1 & 1.2-\rho \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u(k) \tag{1.7}$$

where $\rho = 0.2x_1^2(k)$.

First the discrete-time model is compared to the continuous-time model with zero input and an initial condition of $[0, 1]^T$. This results in the trajectories of 1.5a which show that there are differences after discretization. This is due to the fact that the discretization is only a first order approximation and therefor does not approximate the continuous time model in a fully correct way. For the Nonlinear MPC the YALMIP nonlinear MPC solver is used to compute a suitable input (u(k)) for each time step k. This input is applied to the continuous time nonlinear Simulink model. The new state obtained from the continuous time simulation is then used for the next time step k. The YALMIP solver uses the nonlinear discrete model described in 1.7 for the computation. The result of both the linear and nonlinear MPC is shown in figure 1.5b. In this figure the difference is clearly visible in the first part of the simulation. The nonlinear MPC has a much smoother behaviour compared to the linear MPC. this is due to the fact that the nonlinear MPC can predict and control the system better because of the predictable nonlinear behaviour. The fact that the



(a) Continuous-time vs. discrete-time simulation



(b) Nonlinear and linear simuation

Figure 1.5: Nonlinear and linear MPC simulations

linear MPC goes to the origin is due to the correctly picked Q and R as in assignment 1.2. The fact that the nonlinear MPC goes to the origin is less dependent of the chosen Q and R but for those simulations the same Q and R are used.

1.4 Assignment 1.4

For the quasi LPV controller the system described in equation 1.7 with the ρ is used. The quasi LPV controller calculates for every time step the optimal control input u(k) and after every iteration the expected new state x(k) is determined. Those new states are used to recalculate the ρ for every future state and the A matrix in the cost function is changed accordingly. Then the controller recalculates the optimal u(k). This repeats until the 2 norm of the difference $(u_{previous} - u_{last})$ is smaller than 10^{-5}. When this is the case this means that the expected state trajectory is close to the nonlinear behaviour, while using a linear model. The calculated u(k) is applied to the system and for the next time step the process is repeated.
In figure 1.6 the Simulation for the linear MPC and nonlinear MPC are shown as found in assignment 1.3. Also the trajectory of the quasi LPV is shown as a dotted line. The trajectory of the quasi LPV is almost similar to the nonlinear MPC, which is as expected. From this it can be concluded that the nonlinear MPC and the quasi LPV are leading to approximately the same behaviour. For these simulation the same Q, R, P and N = 10 are used as in assignment 1.2 and 1.3

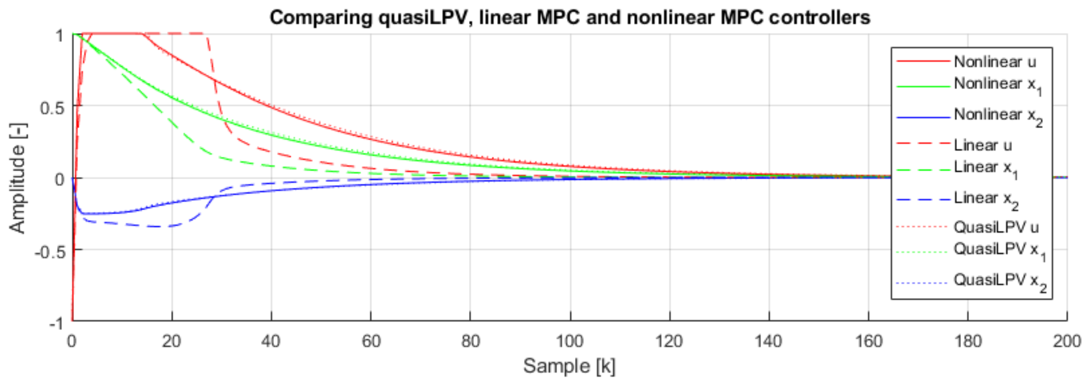


Figure 1.6: quasi LPV, linear MPC and Nonlinear MPC simulations

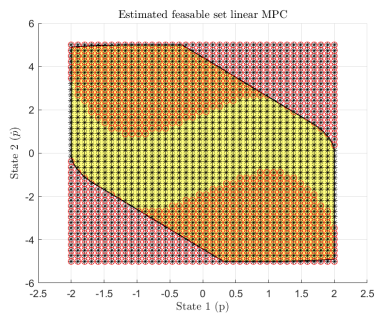
1.5 Assignment 1.5

Because it is not possible to calculate a feasible set for the nonlinear system an estimation is made by simulating the system over grid point in the constrained state space. In figure 1.7 the feasible sets

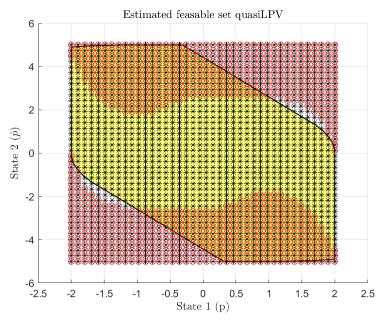
Controller	ODE45	Simulink
Linear MPC	0.0024 [sec]	0.0763 [sec]
Quasi LPV	0.0139 [sec]	0.8531 [sec]
Nonlinear MPC	0.0464 [sec]	0.1119 [sec]

Table 1.1: Average computation time for different controllers

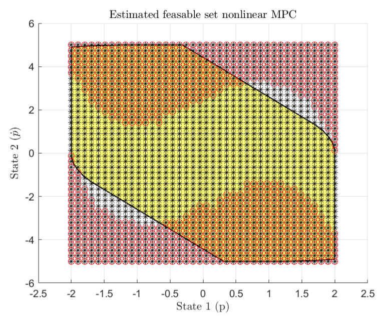
of the linear MPC, quasi LPV and nonlinear MPC simulated with the nonlinear model. The same Q, R and P as described in (1.6) and assignment 1.2, 1.3 and 1.4 are used. The used horizon is N = 5. In table 1.1 the average computation time per time step per control strategy is shown. From this table it can be clearly seen that the simulation time with the Simulink model is significant larger compared to the ODE45 solver. Since the feasible sets contains 40x40 = 1600 grid point to simulate the ODE45 method saves significant amounts of computation time. Also it can be seen that the quasi LPV method is faster than the nonlinear MPC with the ODE45 method, while the Simulink method is slower. This is not desired because the quasi LPV method should be a faster method to solve a nonlinear MPC problem. This difference can be explained by the fact that the Simulink model must be simulated multiple times for the quasi LPV, while the nonlinear MPC must be simulated only once. The Linear MPC is in both cases faster, but has a smaller feasible set.
If the feasible sets are compared it can be concluded that the nonlinear MPC and quasi LPV have a feasible set larger than the linear MPC polytopic feasible set. This can be explained by the fact that the nonlinear system behaviour is exploited in those options and in the linear MPC option this is not used. The polytopic feasible set is larger than the found feasible set for the linear MPC. This can be explained by the fact that the nonlinear behaviour of the system causes the state to move outside the feasible set, while the linearized system does not have this behaviour. The quasi LPV method and the nonlinear method have approximately the same feasible set as expected.



(a) Linear MPC



(b) QuasiLPV



(c) nonlinear MPC

Figure 1.7: Estimated feasible sets

2 Part 2

A longitudinal model of an aircraft flight dynamics is now considered where the state $x_1 = \alpha$ is the angle of elevation and $x_2 = q$ is the pitch angular rate. The control input $u = \delta$ is the elevator control surface deflection. The linearized and discretized model is described by the following 2D model:

$$x(k+1) = \begin{bmatrix} 0.9719 & 0.0155 \\ 0.2097 & 0.9705 \end{bmatrix} x(k) + \begin{bmatrix} 0.0071 \\ 0.3263 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

The initial condition is $[8\ 0]^T$, only the output can be measured and the model has the following input and state constraints:

$$\begin{bmatrix} -15 \\ -100 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 30 \\ 100 \end{bmatrix}, -25 \leq u(k) \leq 25$$

2.1 Assignment 2.1

Now an MPC controller is designed that should achieve small overshoot, a fast response and zero tracking error with an constant reference. The given reference is 10 for $k \in [0\ 600]$, -10 for $k \in [601\ 1200]$ and 10 for $k \in [1201\ 1800]$. The settings of the MPC controller are:

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 0.001 \end{bmatrix},\ R = 1,\ N = 18$$

These settings for the MPC controller are defined after tuning by hand until the closed-loop output showed the desired tracking performance. The reference and closed-loop output (left) and states compared to the estimated states (right) are shown in Figure 2.1.
A Kalman filter is used to estimate the second state, because only the first state is available from the output. The implemented Kalman filter has the following structure:

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) + LC^T(y(k) - \hat{y}(k) + Bu(k) \\ \hat{y}(k) &= C\hat{x}(k) + Du(k) \end{aligned}$$

Where L is the solution of the discrete Riccati equation with $Q_{est} = 0.1 \cdot I_2$ and $R_{est} = 0.5$, which are determined after tuning by hand.
In order to track a reference, a reference preview matrix Y is added to the cost function: $c = F\hat{x}(k) + Yr(k)$. The reference preview matrix is computed beforehand by:
$Y = -2(\Gamma^T \Omega \Psi)(0_{(n+m)N \times n}\ I_{(n+m)N}\ 0_{(n+m)N \times m})H$, where the matrix H is obtained from:

$$(S\ H) = \begin{bmatrix} A & -I_n & & 0 & B & & 0 & 0 \\ & \ddots & \ddots & & & \ddots & & \\ 0 & & A & -I_n & 0 & & B & 0 \\ 0 & & 0 & A - I_n & 0 & & 0 & B \\ C & & 0 & 0 & 0 & & 0 & 0 \\ & \ddots & & & & \ddots & & \\ 0 & & 0 & C & 0 & & 0 & 0 \end{bmatrix}^{-1}$$

This implementation is chosen because the reference preview matrix improves performance, can be computed beforehand and the simplicity of implementing it in the cost function that was already available from the linear MPC framework.

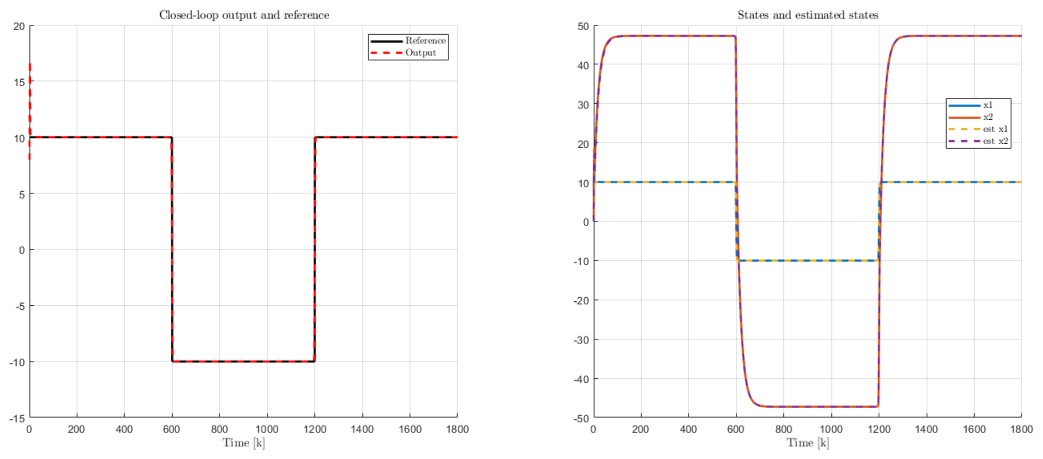


Figure 2.1: Reference tracking

2.2 Assignment 2.2

Due to air turbulence there is now a disturbance acting on the aircraft based on the following model:

$$x(k+1) = \begin{bmatrix} 0.9719 & 0.0155 \\ 0.2097 & 0.9705 \end{bmatrix} x(k) + \begin{bmatrix} 0.0071 \\ 0.3263 \end{bmatrix} u(k) + \begin{bmatrix} 0.5 \\ 0.2 \end{bmatrix} d(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

Where $d(k)$ is 0.1 for $k \in [0\ 600]$, 0.5 for $k \in [601\ 1200]$ and -0.5 for $k \in [1201\ 1800]$. This resulted in Figure 2.2 where the left plot shows the reference tracking and the right plot shows the disturbance estimation compared to the actual disturbance.

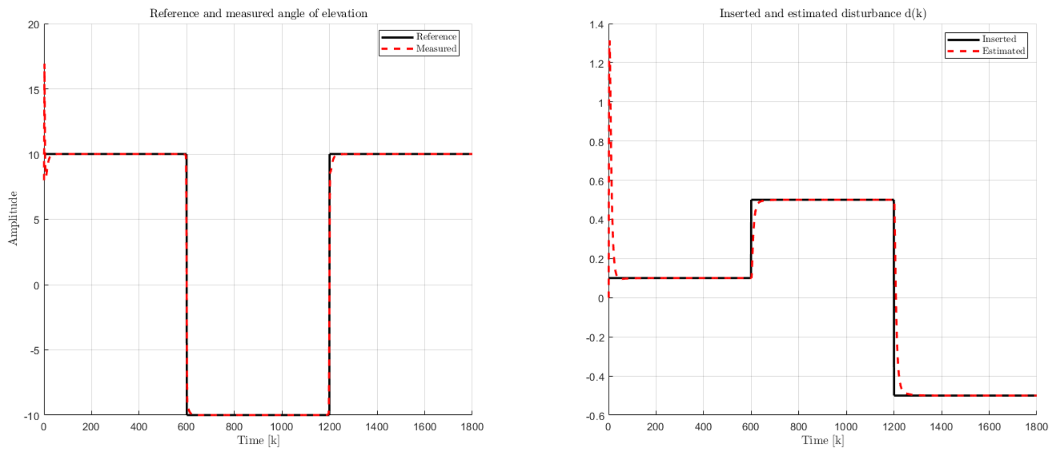


Figure 2.2: Reference tracking with disturbances

The disturbance rejection is done by estimating the value of the disturbance, $\hat{d}(k)$, which also includes estimating the states of the system, $\hat{x}(k)$. Also the steady state values $x^{ss}(k)$ and $u^{ss}(k)$ are computed, which are used for correcting the cost, constraints and input. These values are computed as follows:

$$\begin{bmatrix} x^{ss}(k) \\ u^{ss}(k) \end{bmatrix} = \begin{bmatrix} 0 & C^{-1} \\ B^{-1} & (-A+I)/(BC) \end{bmatrix} \cdot \begin{bmatrix} -B_d\hat{d}(k) \\ 0 \end{bmatrix}$$

Compared to the situation without disturbance rejection, the cost is now computed by $c = F(\hat{x}(k) - x^{ss}(k)) + Yr(k)$, the constraints by $b = W(\hat{x}(k) - x^{ss}(k))$ and the input by $u(k) = u^{MPC}(k) + u^{ss}(k)$.