**An Introduction to Model Predictive Control**

# 1 Introduction

Model predictive control (MPC) is one of the optimization-based control strategy, which predicts the future response of the platform using a nominal process model subject to constraints. Currently, MPC has become arguably the most widespread control technique in agriculture, construction, and petrochemical industries. To explain the concept of MPC, we will take an example of overtaking a large vehicle as shown in Figure 1. The truck intends to turn right by indicting in the traffic, the driver in the small vehicle gives way to the truck by slowing down the speed as it's not safe to take over the large vehicle. In the example, the driver predicts that it's not safe to overtake. The MPC works similarly as it predicts the future control action of the vehicle and platform to produce an optimal response. The driver looks ahead at the distance from his current state on how much he needs to slow down refers as the prediction horizon. Additionally, the behaviour of slowing down the speed is the control horizon for the driver.

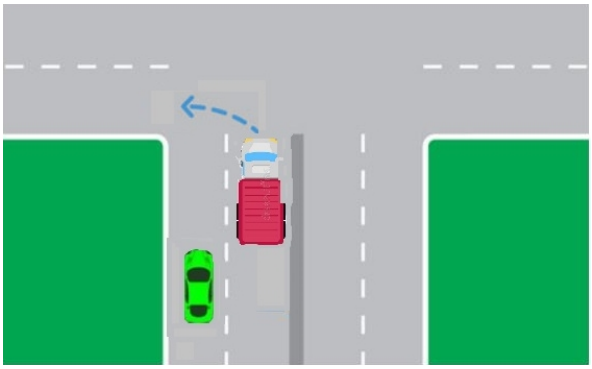

Figure 1: Example of MPC in traffic scenario

The ability of MPC in handling the constraints distinguishes it from the other class of control algorithms. Again, let's take the example of Figure 1, both the drivers are driving in a 50 $Km/h$ zone. They shouldn't exceed this speed to avoid any over speeding fine or demerit point. So, it is a limitation on the driving speed. In terms of MPC, the control signals are bounded to certain values. For instance, if we consider speed and yaw rate as a control input, then constraints on them can be set as $v = 50\,km/h$ and $\omega = 45\,deg/s$. Additionally, MPC can set constraints on the states. So, for Figure 1, both the vehicles are driving in the same line. If we consider a double sold line in the traffic, it refers to driving in the lane. So, MPC can set the values of $(x, , y, \theta)$ in a bounded limit. The control actions under the constraints are computed online in the MPC approach. The current input and output are measured at each sampling instance by the MPC to perform the following tasks:

1. Over a finite horizon, calculate the future control action that satisfies the constraints imposed on the nominal process model.

2. Apply the first control signal from the sequence of control as an input to the actual system.

In this lecture, we intend to give a short introduction to unconstrained and constrained MPC. In addition, the working of nonlinear MPC is also discussed.

## 1.1 Basic structure

To understand the theory behind MPC, we will discuss the main components involved in the design of the MPC. First, predictions based on a nominal process model are required to generate the control signals by optimizing the cost-function under the constraints. Second, an actual model (sometimes subject to external disturbance) are required to use the applied inputs. Finally, the MPC formulation shown in Figure 2 represents the basic structure of the MPC.

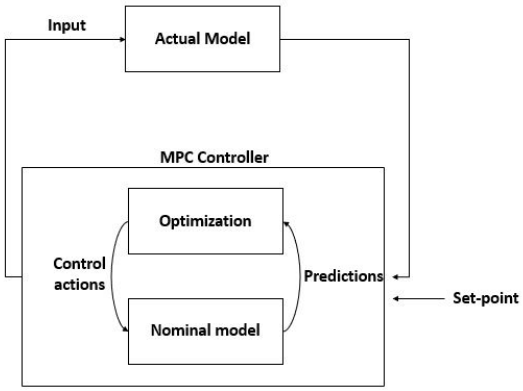


Figure 2: Basic structure of MPC

### 1.1.1 Prediction and optimization

Consider a nominal discrete-time state-space model, which will be used to predict the actual model as follows:

$$x_{k+1} = Ax_k + Bu_k, \tag{1}$$

$$y_k = Cx_k \tag{2}$$

with an initial condition of $x_{0|k} = x_k$, where $x_k$ and $u_k$ are the state and input vectors at the $k^{th}$ sampling instant. Given a predicted sequence, the corresponding sequence of state prediction is generated by simulating the model forward over the prediction horizon, of let say $N$ sampling intervals. For notation, these predicted sequences are often stacked into vectors $\mathbf{u}$, $\mathbf{x}$ defined by

$$\mathbf{u}_k = \begin{bmatrix} u_{0|k} \\ u_{1|k} \\ \vdots \\ u_{N-1|k}, \end{bmatrix} \mathbf{x}_k = \begin{bmatrix} x_{0|k} \\ x_{1|k} \\ \vdots \\ x_{N|k} \end{bmatrix} \tag{3}$$

MPC runs by minimizing a predictive cost function, which defines the predicted sequences of $u$, $x$. Our focus would be the quadratic cost in this document. Consider the following quadratic cost function:

$$J(x_k, u_k) = \sum_{i=0}^{N-1} \|x_{i|k}\|_Q^2 + \|u_{i|k}\|_R^2 + \|x_{N|k}\|_P^2 \tag{4}$$

where $\|x_{i|k}\|_Q^2 = x^T Q x$, $Q$ is the state penalty symmetric and positive definite matrix, $R$ is input penalty symmetric and positive definite matrix, $N$ is the prediction horizon, and It is important to include the terminal predictive state $x_{N|k}$ to get the final response of the system. The matrix $P$ is final state penalty symmetric and positive semi-definite matrix. Now, we can rewrite the Eqn (1) as follows:

$$X = S_x x_0 + S_u u, S_x = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, S_u = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & & & \\ AB & B & & \\ \vdots & \vdots & \ddots & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \tag{5}$$

The cost function in (4) can be re-written by ignoring the terminal cost as follows:

$$J = x_k^T Q x_k + u_k^T R u_k \tag{6}$$

with

$$\bar{Q} = \begin{bmatrix} Q_1 & 0 & \cdots 0 \\ 0 & Q_2 & \cdots 0 \\ 0 & 0 & \ddots Q_N \end{bmatrix}, \qquad \bar{R} = \begin{bmatrix} R_1 & 0 & \cdots 0 \\ 0 & R_2 & \cdots 0 \\ 0 & 0 & \ddots R_N \end{bmatrix} \tag{7}$$

Substituting for $x_k$ using (4)

$$J = x_0^T Q x_0 + X^T \bar{Q} X + u^T \bar{R} u \tag{8}$$

where Note that the matrices $H$, $F$, and $G$ can be computed offline. It is important to mention it here that we intend to apply the first element from our predicted sequence of input $u_k^*$ to the system:

$$u_k = u_{0|k}^* \tag{9}$$

at each sampling instant $k = 0, , 1, \ldots,$ the process of computing $u_k^*$ by minimizing the predicted cost and implementing the first element of $u_k^*$ is then repeated. Thus, we use online optimization. In Figure 3, we express the prediction horizon over the future response of the system. It predicts the future state of the system with predicted control input (control horizon). The main idea is to learn from the past behaviour of the system and apply predictions to get the future response. To enhance our knowledge of MPC, we will now focus on two basic types of models such as linear and nonlinear models.

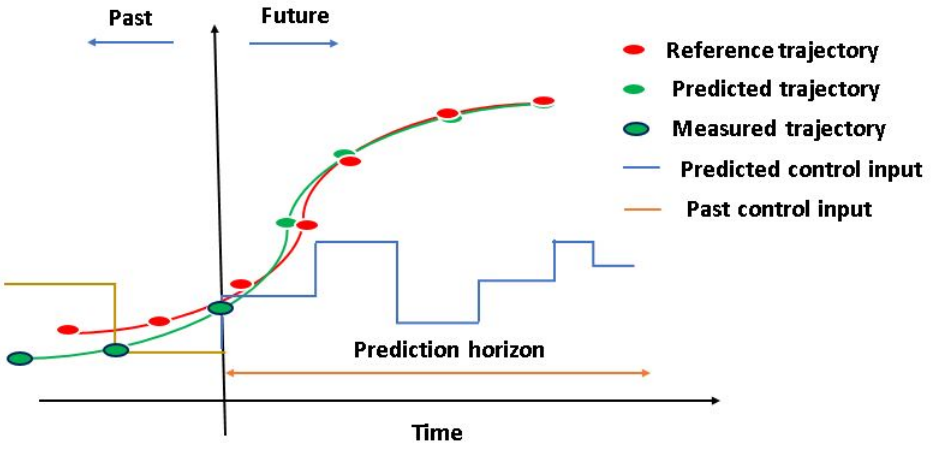


Figure 3: Principle of MPC

### 1.1.2 Linear process model

For linear time-invariant (LTI) systems, predictions on $x_k$ on $u_k$ are linearly distributed. The cost-function expressed for $u$ is as follows:

$$J(u_k) = u_k^T G u_K + 2f^T u_k + g \tag{10}$$

where $G$ is a constant positive definite matrix, $f = f(x_k)$ and $g = g(x_k)$ are vector and scalar, respectively. Additionally, constraints imposed on $u_k$ would be linear as

$$A_c u_k \leq b_c \tag{11}$$

where $A_c$ is a constant matrix and the vector $b_c = b_c(x_k)$ is a function of $x_k$. Hence, the optimization of quadratic cost-function subject to linear constraints are given below:

$$\textbf{min}_u \qquad u^T G u + 2f^t u \tag{12}$$

$$\textbf{subject to} \qquad A_c u \leq b_c \tag{13}$$

To solve the above optimization problem, we use quadratic programming (QP), given G is positive definite matrix and linear constraints are imposed on the cost-function. The optimization problem in Eqn (10) and (11) is a convex optimization problem, which can be solved using various optimization techniques. Computationally, QP can be solved using Matlab's QP solver (quadprog). Consider that we intend to run a convex problem using quadprog on a 2.3 GHz processor with 16GB RAM, the processing time would me 2 $ms$ each iteration to solve 100 variables subject to 100 constraints.

### 1.1.3 Nonlinear process model

The MPC optimization is significantly challenging if a nonlinear process model is employed. This is mainly due to nonlinear dependence of the states $x_k$ on $u_k$. The cost function would become a non-convex function of $u_k$, which makes the following optimization problem as nonlinear programming (NLP)

$$\textbf{min}_u \qquad (x_k, u) \tag{14}$$

$$\textbf{subject to} \qquad g(x_k, u) \leq 0 \tag{15}$$

In contrast to the convex optimization problem, non-convex optimization is complex and doesn't guarantee that a solver will converge to a global minimum of the cost function. In terms of processing NLP, we use fmincon in Matlab. Consider, the same specs as per the previous example in QP, the processing time would be 100 $ms$ per iteration. The NLP's are non-convex and that's why they take a longer time to optimize.

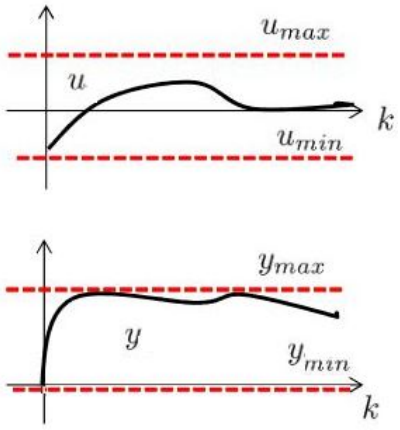


Figure 4: Constraints imposed on a plant

### 1.1.4 Constraints handling

In addition to obvious equality constraints, MPC can encounter inequality constraints on both the states and input. In online optimization, inequality constraints are imposed explicitly; while the equality constraints are dealt with implicitly. For this lecture, we will focus on input and state constraints.

**Input Constraints**: They occur as a result of the actuator limits (torque saturation in D.C motors, restricting speed and yaw rate in unmanned ground vehicles (UGVs), and motor voltage control of pendulum systems). In optimization problem, we can define them as follows:

$$u_{min} \leq u_k \leq u_{max} \tag{16}$$

Additionally, we can set a rate of change constraints as follows:

$$\Delta u_{min} \leq u_k - u_{k-1} \leq \Delta u_{max} \tag{17}$$

**State constraints** State constraints may be active during transients (aircraft stall speed or in steady-state operation of UGVs) while respecting the bounds of the path. The state constraints are defined as

$$y_{min} \leq y_k \leq y_{max} \tag{18}$$

**Hard/soft constraints**: The optimization problem becomes unfeasible if they are not met. For instance, a vehicle must have a maximum speed of $3\,m/s$. In contrast, soft constraints can be violated if necessary to avoid in-feasibility. For example, the upper and lower bounds of the path can have some small deviation. In Figure 4, an example of constraints imposed on a platform is shown, which clarifies the concept of upper and lower bounds on input and states.

# 2 Unconstrained MPC

To solve the unconstrained MPC we must have a cost function (4), prediction model (1), and parameters $(Q, R, N, \bar{Q}$ used in solving MPC. The idea behind unconstrained MPC is to compute control sequence $u_k$ that minimize the cost function without considering the input and state constraints. We will consider the following solutions to solve this problem:

**Predicted future states**: We will first group the future control inputs and predicted/future states into vectors as per (3) as follows

$$U = \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}, \qquad X = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \tag{19}$$

Now, we will compute $x_k$ as per (5) and apply it to cost function in (6).

**Compute control actions that optimize future states**: For this procedure, we will first apply the equality constraints (9-11) into the cost function (8) and then minimize the U that yields zero gradient

$$\min_{J(u)} = x_0^T H x_0 + 2U^T F x_0 + U^T G U \tag{20}$$

where $H = Q + S_x^T \bar{Q} S_x$, $F = S_u^T \bar{Q} S_x$, and $G = \bar{R} + S_u^T \bar{Q} S_u$ can be computed offline.

$$\nabla J(U) = 0 \in U^* = -G^{-1} F x_0 \tag{21}$$

The above expression is an analytical solution. Now, we can extract the first element of the optimal

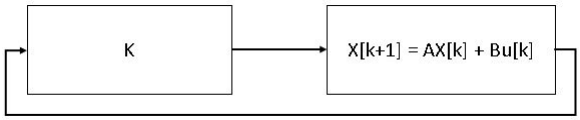


Figure 5: Closed loop configuration of unconstrained MPC

control vector

$$u^* = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} U^* = \underbrace{- \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} G^{-1} F}_{\text{K}} x_0 \tag{22}$$

where at every time step k, apply control law

$$x_0 = x_k, \qquad u_k = K x_k \tag{23}$$

The above control law also verifies that unconstrained MPC is similar to linear state feedback controller as shown in the Figure 5.

**Matlab example 1**. Consider the following discrete-time model of a mechanical system

$$x_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x$$

where the input to the process model is acceleration $u[m/s^2]$, states are $x = [x1\ x2]^T$ the position$[m]$ and velocity$[m/s]$ with a sampling time of $dt = 0.1s$. Write a Matlab script that regulate position and velocity $x_1$, $x_2 \to 0$ using unconstrained MPC with the following parameters:

$$N = 2, \qquad R = 1/10, \qquad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

**Solution**. Run the following code and save it as $unconstrained_MPC.m$.

```
%This is an example code of Unconstrained MPC
%Author: Subhan Khan

%Problem: Consider a discrete-time model of Mechanical System
%Input: Acceleration
%Output: Position [x1] and [x2] velocity


clear all;
clc; close all
x0 = [10;0]; % Initial Condition
y0 = [0;0];
```

```
dt = 0.1; %
u = 0;
T_end = 10;
t = 0:dt:T_end;
Nsim = length(t) - 1;
A = [1 1;0 1]; B = [0;1]; C = [1 0;0 1]; %States
n = size(A,2);
N = 2; R = 1/10; Q = [1 0;0 0]; P = [1 0;0 1]; %Parameters
S_x = eye(n+n,n);
S_x(1:n,1:n) = A;
S_x(n+1:n+n,1:n) = A^2;
S_u = zeros(n+n,n);
S_u(1:n,1) = B;
S_u(n+1:n+n,:) = A.*B;
S_u(n+1:n+n,1) = 1;
Qb = diag([1,0,1,1]); Rb = diag([1/10,1/10]);
 x1 = x0;
 u1 = u;
F = S_u'*Qb*S_x;
G = Rb + S_u'*Qb*S_u;
K = -[1 0]*inv(G)*F;
for k = 1 : Nsim + 1
    u(:, k) = K*x0(:,k);
    x0(:, k + 1) = A*x0(:,k) + B*u(:, k);
    y0(:,k) = C*x0(:,k);
end
%----------------------When R = 6----------------------
Rb = diag([6,6]);
F = S_u'*Qb*S_x;
G = Rb + S_u'*Qb*S_u;
K = -[1 0]*inv(G)*F;
for k = 1 : Nsim + 1
    u1(:, k) = K*x1(:,k);
    x1(:, k + 1) = A*x1(:,k) + B*u1(:, k);
    y1(:,k) = C*x1(:,k);
end

figure
subplot(221)
plot(t,y0(1,:),t,y1(2,:))
xlabel('Time[s]'); ylabel('Position (m)');
```

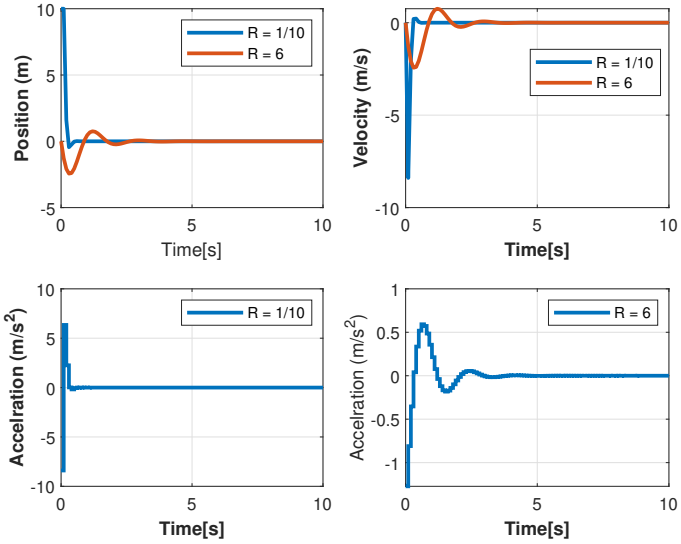


Figure 6: Actual states and control input with different penalty matrix R

```
legend('R = 1/10', 'R = 6');
grid on;
subplot(222)
plot(t,y0(2,:),t,y1(2,:))
xlabel('Time[s]'); ylabel('Velocity (m/s)');
legend('R = 1/10', 'R = 6');
grid on;
subplot(223)
stairs(t,u)
xlabel('Time[s]'); ylabel('Accelration (m/s^2)');
legend('R = 1/10')
grid on;
subplot(224)
stairs(t,u1)
xlabel('Time[s]'); ylabel('Accelration (m/s^2)');
legend('R = 6')
grid on;
```

In Figure 6,it is obvious that with an unconstrained MPC, the limits of states and input are not bounded, which means that if we change the penalty matrix $R$ there will be a change in the amplitude of both states and input.

## 2.1 Constrained MPC

For the constrained MPC, we will reformulate the cost function of unconstrained MPC and impose input or state constraints. For this lecture, our focus would be constraints imposed on input. For the optimization problem, our decision variable U will impact on cost function be as follows:

$$\min_{J(U)} = x_o^T H x_0 + 2U^T F x_0 + U^T G U \tag{24}$$

with $U_{min} \leq U \leq U_{max}$, , $F$, $G$ are the same as in the unconstrained MPC. To solve, $U^*$ we will use QP-based solver in Matlab quadprog, which will solve the quadratic cost and linear constraints. In addition, $U_{min} = [u_{min} \ \ldots \ u_{min}]$ and $U_{max} = [u_{max} \ \ldots \ u_{max}]$ The constrained MPC problem in (24) can be converted to QP as follows

$$\begin{aligned} \min_z V(z) = \frac{1}{2} z^T H_n z + f^T z \\ \textbf{s.t.} \qquad Ez \leq b \end{aligned} \tag{25}$$

with decision variable $U = z$, cost function $H_n = 2G$, $f = 2Fx_0$, constraints $E = [I \ -I]^T$, and $b = [Umax \ -Umin]^T$. The identity matrix $I = N \times N$.

**Matlab Example 2**. We will now impose input constraints on the Matlab Example 1. Run the following code and save it as $Constrained_MPC.m$

```
%This is an example code of Constrained MPC
%Author: Subhan Khan

%Problem: Consider a discrete-time model of Mechanical System
%Input: Acceleration
%Output: Position [x1] and [x2] velocity


clear all;
clc;
x0 = [10;0]; % Initial Condition
y0 = [0;0];
dt = 0.1; % sampling Time
u = 0;
T_end = 10;
t = 0:dt:T_end;
Nsim = length(t) - 1; %Total Simulation
A = [1 1;0 1]; B = [0;1]; C = [1 0;0 1]; %States
n = size(A,2);
N = 2; R = 1/10; Q = [1 0;0 0]; P = [1 0;0 1]; %Parameters
S_x = eye(n+n,n);
```

```
S_x(1:n,1:n) = A;
S_x(n+1:n+n,1:n) = A^2;
S_u = zeros(n+n,n);
S_u(1:n,1) = B;
S_u(n+1:n+n,:) = A.*B;
S_u(n+1:n+n,1) = 1;
%Penalty Matrices
Qb = diag([1,0,1,1]); Rb = diag([1/10,1/10]);
%When R = 6;
%Rb = diag([6,6]);
F = S_u'*Qb*S_x;
G = Rb + S_u'*Qb*S_u;
%K = -[1 0]*inv(G)*F;
H = 2*G; f = 2*F*x0;
E1 = eye(N,N); E2 = eye(N,N);
%Setting up Constraints
E = [E1;-E2];
u_max = 1;
u_min = -1;
b = [u_max;-u_min;u_max;u_min];
for k = 1 : Nsim + 1
    xp = x0(:,k); % Measure the state x0 at time instant k
    f = 2 * (F * xp); %update cost vector
    uopt = quadprog(H, f, E, b); %Compute optimal control U^*
    u(:,k) = uopt(1); %Apply the first element u^*[0] of U to the platform
    x0(:, k + 1) = A*x0(:,k) + B*u(:, k);
    y0(:,k) = C*x0(:,k);
end
figure
subplot(211)
plot(t,y0(1,:))
xlabel('Time[s]'); ylabel('Position (m)');
subplot(212)
plot(t,y0(2,:))
xlabel('Time[s]'); ylabel('Velocity (m/s)');
figure
stairs(t,u)
xlabel('Time[s]'); ylabel('Accelration (m/s^2)');
```

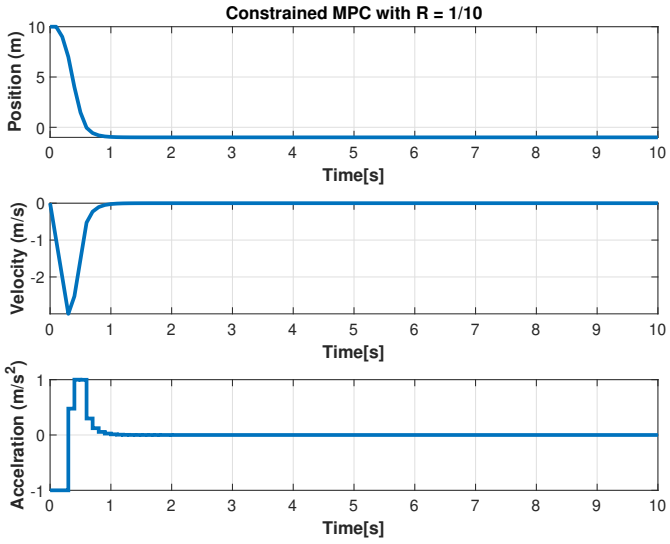


Figure 7: Actual states and control input with different penalty matrix R on constrained and unconstrained MPC

# 3 Nonlinear MPC

In the previous section we discussed about the Linear MPC, which considers linearized process model to perform optimal control. We will take a look at the nonlinear version of MPC known as nonlinear MPC (NMPC). Consider the following nonlinear process model which is subject to input and state constraints:

$$\dot{x}(t) = f(x(t), u(t)), \qquad x(0) = x_0 \tag{26}$$

subject to

$$u_{min} \leq u \leq u_{max}, \qquad x_{min} \leq x \leq x_{max} \tag{27}$$

with the following assumption that system in equation (26) has an unique continuous solution for any initial condition in the point of interest (POI) and any piece-wise continuous and right continuous input function $u(\cdot) : [0,\ N_p] \rightarrow (u_{min} \leq u \leq u_{max})$. To distinguish the nominal and actual process model within the controller, we will use a bar, for example $\bar{x}$, $\bar{u}$. Thus, to define cost function of the system in (26), we will use the following equation:

$$J(x(k), \bar{u}(\cdot)) = L_N(\bar{x}(N), \bar{u}(N)) + \sum_{k=0}^{N-1} L(\bar{x}(k), \bar{u}(k)) \tag{28}$$

subject to:

$$\dot{x}(\tau) = L(\bar{x}(k), \bar{u}(k)), \qquad \bar{x}(k) = x(k), \tag{29}$$

$$\bar{u} \in ((u_{min} \leq u \leq u_{max}), \tag{30}$$

$$\bar{u}(k) = \bar{(u)}(k + N - 1), \tag{31}$$

$$\bar{x}(k) \in (x_{min} \leq x \leq x_{max}) \tag{32}$$

where $L_N$ is the terminal cost; depends on the final state of the system, $L$ is the stage cost, which specifies the desired control performance. While terminal and stage cost consists of the following:

$$L_N(x, u) = (x_N - x_s)^T Q_N (x_N - x_s) + u^T R_N u \tag{33}$$

$$L(x, u) = (x - x_s)^T Q (x - x_s) + u^T R u \tag{34}$$

with $Q_N$ and $R_N$ are the weighted replica of $Q$ and $R$, $x_s$ is the reference or set-point. The basic structure of NMPC can be seen in the Figure 8, which requires a state-estimator to measure the estimated outputs. In our case, that state-estimator can be extended kalman filter (EKF). The basic idea behind NMPC is to

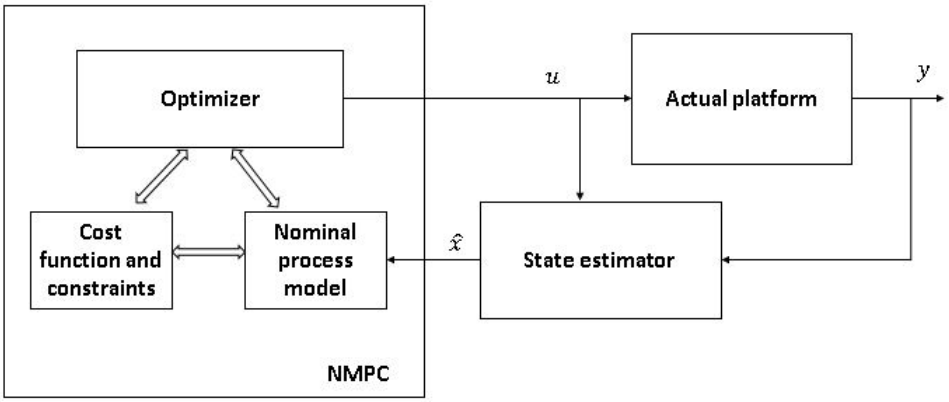


Figure 8: Basic NMPC control-loop

1. Get the estimated states from the process model.
2. Over a given prediction horizon, optimize the cost-function which results the suitable input sequence for the process model.
3. Apply the first portion of control sequence to the process model repeatedly until new measurements arrive at each time.
4. continue with first step.

To conclude from the basic steps, NMPC can do the following:

1. It allows the predictions over nonlinear process model.

2. It allows an explicit consideration of constraints imposed on the states and inputs.
3. It offers a some part of the optimization to be minimized online.
4. The optimization problem in NMPC is non-convex, which requires nonlinear linear programming to solve the optimization problem. Some available methods are proximal algorithms, sequential quadratic programming, and interior point optimization method.
5. For details consider using the online toolkits from Casadi, Acado, and optimization engine.

# 4 Project 4 MPC Questions

**1**. Consider the linearized model of an aircraft at altitude of $5000m$ and speed $128.2m/s$:

$$\dot{x} = \begin{bmatrix} -1.28 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.43 & 0 & -1.84 & 0 \\ -128.2 & -128.2 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix} u, \qquad y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x$$

where $x = [x_1\ x_2\ x_3\ x_4]$ with $x_1[rad]$ is the angle of attack, $x_2[rad]$ is the pitch angle, $x_3[rad/s]$ is the pitch rate, and $x_4[m]$ is the altitude. In addition, $u[rad]$ is the elevator angle as shown in the Figure 8. Perform the following tasks to regulate pitch angle and altitude $(x_2,\ x_4) \to 0$

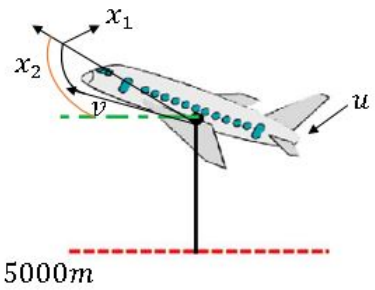


Figure 9: Aircraft control problem

**Part A**.Write a Matlab script that:

1. create a discrete-time, state-space prediction model with a sampling time of $0.25s$
2. Design an unconstrained MPC with $N = 10$, $Q = diag(0,\ 1,\ 0,\ 1)$, and $R = 10$.
3. Simulate the closed-loop response for $10s$ with the initial conditions $x(0) = [0\,10\ \ 0\,100]$

**Part B**. Write a Matlab script that:

1. create a discrete-time, state-space prediction model with a sampling time of $0.1s$
2. Design a constrained MPC with $N = 15$, $Q = diag(0,\ 1,\ 0,\ 1)$, and $R = 10$.

3. Impose input constraints $|u| \leq 0.262rad$ and $|\dot{u}| \leq 0.542rad/s$
4. Simulate the closed-loop response for 30 with the initial conditions $x(0) = [0\,0\;\;0\,5100]$. Here select your own values of $N$, $Q$ and $R$.