Ben Gurion University of the Negev

Model Predictive and Interpolation Control
362.2.5229

Mobile Robot Tracking

Authors:
Shuxin Cui (850060641)

January 15, 2023

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Contents

Introduction

This report includes the implementation of three different interpolation control(IC) algorithm and two slightly different model predictive control(MPC) algorithm dealing with the tracking problem of a two wheel mobile robot. To get better results in tracking, I built different system models for IC and MPC respectively. See the attachment for the code and its documentation.

The report will divided into three sections. The first section will present the formulation of first model, some basics for all three IC control algorithm and the results of the IC algorithm implementation. Then, at Section 2, the formulation of model 2, MPC basics and the results of the MPC algorithm implementation will be given. The last section is some comment about all these algorithms.

1 Interpolation Control

1.1 Problem formulation

The formulation of the first model will be given here. Suppose that the control objective is to make the mobile robot move along a straight line(x axis in figure1). It is assumed that the robot system conforms to nonholonomic constraints, and the robot body has no lateral sliding.

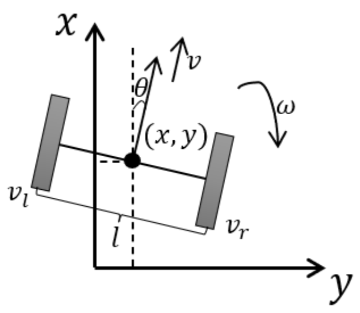


Figure 1: Mobile Robot

$$\begin{cases} v = \frac{1}{2}(v_l + v_r) \\ \omega = \frac{1}{2}(v_l - v_r) \end{cases} \tag{1.1}$$

Besides, we also have:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{1.2}$$

Considering that in tracking problem, the error will be reduced to a small magnitude. According to small-angle approximations, equation (1.2) can be reformed as following.

$$\begin{cases} \dot{x} = v \\ \dot{y} = v\theta \\ \dot{\theta} = \omega \end{cases} \tag{1.3}$$

Taking v as a system constant and w as the control signal. Remind that the control objective is to ensure the mobile robot moving along x axis, which means y and θ in equation (1.3) needs to converge to 0. Set $e_1 = y$, $e_2 = \theta$ and reform equation (1.3).

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \tag{1.4}$$

The implementation of IC algorithm requires a time discrete model. Taking the sampling time as Δt. For simplicity, setting a new $x = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$(notice that this is not the same x in (1.2) or (1.3)), $u = w$, $A = I + \Delta t \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix}$ and $B = \Delta t \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, the following equation can be obtained,

$$x(t+1) = Ax(t) + Bu(t) \tag{1.5}$$

In this way, the tracking problem is translated into a regulation problem. The system constraint is given as,

$$\begin{cases} x(t) \in X, X = \{x \in \mathbb{R}^n : F_x x \leq g_x\} \\ u(t) \in U, U = \{u \in \mathbb{R}^m : F_u u \leq g_u\} \end{cases} \tag{1.6}$$

The cost function for (1.5) is,

$$J = \int_0^\infty \|x(t)\|_Q + R \cdot u(t)^2 \tag{1.7}$$

For Nominal case, v is a constant value, which mean that all the parameter of the system (1.5) is known. For Robust case, consider a v with disturbance but with know constraints.

1.2 Nominal case IC

System Equation

$$x(t+1) = Ax(t) + Bu(t) \tag{1.8}$$

Where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, respectively the state and input vectors at time step k. The state variables and the control variables are subject to the following polytopic constraints

$$\begin{cases} x(t) \in X, X = \{x \in \mathbb{R}^n : F_x x \leq g_x\} \\ u(t) \in U, U = \{u \in \mathbb{R}^m : F_u u \leq g_u\} \end{cases} \tag{1.9}$$

Where the matrix F_x, F_u and the vectors g_x, g_u are assumed to be constant with $g_x > 0$, $g_u > 0$ such that the origin is contained in the interior of X, U.

1.2.1 Algorithm

Algorithm 1 Interpolation based control

```
Input: Maximum invariant set: $\Omega_{max}$
    Maximum controlled invariant set: $C_N$
    feedback control gain over $\Omega_{max}$: $K$
    control values at vertices of $C_N$(vertex control): $U_v$
Output: State space partition of $C_N$: $P$
    the feedback control laws over the partitions of $C_N$
1: while true do
2:   Measure the current state $x_k$
3:   if $x \in \Omega_{max}$ then
4:     $u = Kx$
5:   else
6:     Solve the following LP problem
         $c^* = \min_{r_v, c} c$
         $\begin{cases} F_N r_v \leq c g_N \\ F_o(x - r_v) \leq (1-c) g_o \\ 0 \leq c \leq 1 \end{cases}$
7:     $u = c u_v + (1-c) u_o$
8:   end if
9: end while
```

The ideal of IC is very simple. This algorithm includes two different controller. Let's say controller u_v and u_o. u_o is a stronger controller with our desired performance(e.g converge fast/low cost, etc.), while u_v is a controller that only ensures the convergence of the system. We learnt to use Vertex control[1] and Minkowski[2] to obtain the u_v on course.

In order to explain the algorithm more clearly, see algorithm 1:

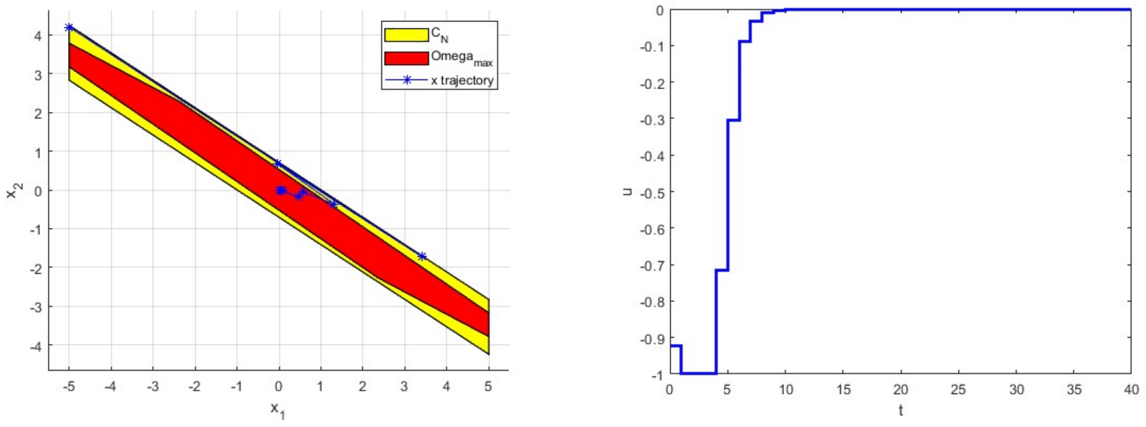
As showed above, the algorithm requires a preprocessing of computing the Maximum invarient set: Ω_{max} and Maximum controlled invariant set: C_N. The computation for these sets is given in [3]. The algorithm applies only u_o inside Ω_{max} and will apply a interpolated controller u between u_o and u_v outside Ω_{max} (inside C_N).

1.2.2 Implementation

Set $v = 1$ as a constant value. Set the constraint for x and u in (1.5) as: $\begin{bmatrix}-5\\-5\end{bmatrix} \leq x \leq \begin{bmatrix}5\\5\end{bmatrix}$ and $-1 \leq u \leq 1$. The LQR is used here to compute the controller gain for Ω_{max} computation. By using the $dlqr()$ command in matlab, the obtained controller gain is

$$K^* = [-1.4114 - 1.9075]' \tag{1.10}$$

With initial condition $x = \begin{bmatrix}-5\\4.2\end{bmatrix}$, the control input and the performance of the controller is showed as below,



(a) x trajectory (b) Control signal u

Figure 2: Performance of Nominal Case IC

1.3 Robust Case IC

System Equation

$$x_{k+1} = A_k x_k + B_k u_k + D w_k \tag{1.11}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and $w_k \in \mathbb{R}^d$ are respectively, the measurable state, the input and the disturbance vectors. The matrices $A_k \in \mathbb{R}^{n\times n}$, $B_k \in \mathbb{R}^{n\times m}$ and $D \in \mathbb{R}^{n\times d}$. A_k and B_k satisfy,

$$\begin{cases} A_k = \sum_{i=1}^{q} \alpha_{i,k} A_i \\ B_k = \sum_{i=1}^{q} \alpha_{i,k} B_i \\ \sum_{i=1}^{q} \alpha_{i,k} = 1, \alpha_{i,k} \geq 0 \end{cases} \tag{1.12}$$

[1]Vertex control Algorithm B
[2]Minkowski Norm Minimization Control Algorithm C
[3]Set Theoretic Methods in Control A.1

where the matrices A_i, B_i are given.

The state, control input and the disturbance are subject to the following bounded polytopic constraints,

$$\begin{cases} x_k \in \ X, \ X = \{x \in \mathbb{R}^n : F_x x \leq g_x\} \\ u_k \in \ U, \ U = \{u \in \mathbb{R}^m : F_u u \leq g_u\} \\ w_k \in W, W = \{w \in \mathbb{R}^d : F_w w \leq g_w\} \end{cases} \tag{1.13}$$

where the matrices F_x, F_u and F_w and the vectors g_x, g_u and g_w are assumed to be constant with $g_x > 0$, $g_u > 0$ and $g_w > 0$. The inequalities are component-wise.

1.3.1 Algorithm

The algorithm of Robust case IC is the same as showed in Nominal case. The difference is located in the computation for sets Ω_{max} and C_N. For the computation of Ω_{max} and C_N in robust case, see[4].

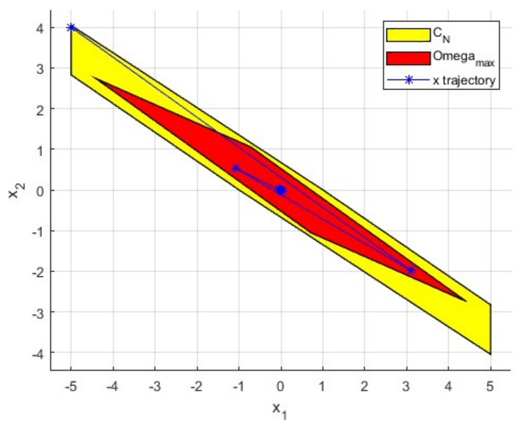
1.3.2 Implementation

Take v as a system parameter with disturbance and $0.8 \leq v \leq 1$. Therefore,

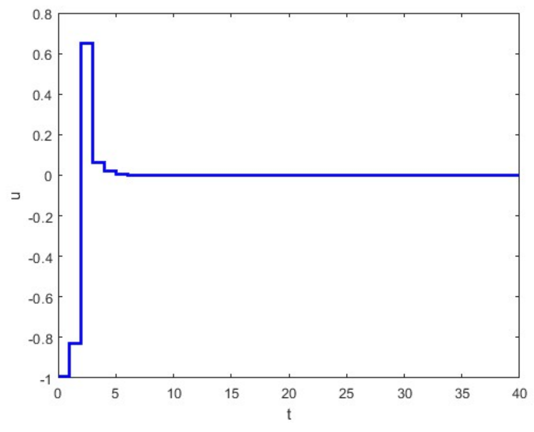
$$A_1 = \begin{bmatrix} 1 & 1.8 \\ 1 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 2.2 \\ 1 & 1 \end{bmatrix}$$

Set the constraint for x and u in (1.5) as: $\begin{bmatrix} -5 \\ -5 \end{bmatrix} \leq x \leq \begin{bmatrix} 5 \\ 5 \end{bmatrix}$ and $-1 \leq u \leq 1$. The LQR is used here to compute the controller gain for Ω_{max} computation. Using A_1 to compute the K.

With initial condition $x = \begin{bmatrix} -5 \\ 4 \end{bmatrix}$, the control input and the performance of the controller is showed as below,



(a) x trajectory



(b) Control signal u

Figure 3: Performance of Robust Case IC

1.4 Improved IC with Minkowski

The system equation is the same as[5].

[4]Set Theoretic Methods in Control A.2

[5]Nominal case of IC 1.2

1.4.1 Algorithm

Improved IC

C_N is the set of all states that can be brought into Ω in no more than N steps. In the previous methods, C_N is computed iteratively. For the reason that the computation of C_N involves the polyhedral set projection operation, it's computationally expensive to calculate C_N, especially for high order systems.

Here we will introduce another method to compute the controller. Recall the implicit representation of C_1,

$$C_1 = \left\{ x \in \mathbb{R}^n, \exists u_0 \in \mathbb{R}^m : \begin{cases} F_o(Ax + Bu_0) \leq g_o \\ F_x x \leq g_x, F_u u \leq g_u \end{cases} \right\} \tag{1.14}$$

Analogously, for C_N

$$C_N = \left\{ \begin{array}{l} x \in \mathbb{R}^n, \\ \exists u_0 \in \mathbb{R}^m, \\ \exists u_1 \in \mathbb{R}^m, \\ \quad \vdots \\ \exists u_{N-1} \in \mathbb{R}^m, \end{array} : \begin{cases} F_o(A^N x + A^{N-1}Bu_0 + \cdots + Bu_{N-1}) \leq g_o, \\ F_x(A^{N-1}x + A^{N-2}Bu_0 + \cdots + Bu_{N-2}) \leq g_x, \\ F_x(A^{N-2}x + A^{N-3}Bu_0 + \cdots + Bu_{N-3}) \leq g_x, \\ \quad \vdots \\ F_x x \leq g_x, \\ F_u u_0 \leq g_u, \\ F_u u_1 \leq g_u, \\ \quad \vdots \\ F_u u_{N-1} \leq g_u \end{cases} \right\} \tag{1.15}$$

For simplicity, denote

$$U_N = \begin{bmatrix} u_0^T & u_1^T & \cdots & u_{N-1}^T \end{bmatrix}^T \tag{1.16}$$

Rewrite equation (1.15) into half-space representation as

$$C_N = \left\{ x \in \mathbb{R}^n, \exists U_N \in R^{Nm} : \left\{ F_N \begin{bmatrix} x \\ U_N \end{bmatrix} \leq g_N \right. \right\} \tag{1.17}$$

where

$$F_N = \begin{bmatrix} F_oA^N & F_oA^{N-1}B & F_oA^{N-2}B & \cdots & F_oA^2B & F_oAB & B \\ F_xA^{N-1} & F_xA^{N-2}B & F_xA^{N-3}B & \cdots & F_oAB & B & 0 \\ F_xA^{N-2} & F_xA^{N-3}B & F_xA^{N-3}B & \cdots & B & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ F_xA & B & 0 & \cdots & 0 & 0 & 0 \\ F_x & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & F_u & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & F_u & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & F_u & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & F_u \end{bmatrix} \text{ and } g_N \begin{bmatrix} g_o \\ g_x \\ g_x \\ \vdots \\ g_x \\ g_x \\ g_u \\ g_u \\ \vdots \\ g_u \\ g_u \end{bmatrix} \tag{1.18}$$

The implicit representation of C_N as showed above in equation (1.17) can be used for interpolation control. Just as the other two IC algorithm we have talked above, it's still an LP problem. We need to find the best convex decomposition x_v, x_o by minimizing c, where $x_v \in C_N$ and $X_o \in \Omega$. The new LP problem is as following

$$c^* = \min_{c, x_v, x_o, U_N} \{c\} \quad s.t. \begin{cases} F_N \begin{bmatrix} x \\ U_N \end{bmatrix} \leq g_N \\ F_o x_o \leq g_o \\ cx_v + (1-c)x_o = x \\ 0 \leq c \leq 1 \end{cases} \tag{1.19}$$

Set $r_v = cx_v$, $r_o = (1-c)x_o$ and $V_N = cU_N$ and simplify the equations, we can get

$$c^* = \min_{c,r_v,V_N} \{c\} \quad s.t. \begin{cases} F_N \begin{bmatrix} r_v \\ V_N \end{bmatrix} \leq cg_N \\ F_o(x - r_v) \leq (1-c)g_o \\ 0 \leq c \leq 1 \end{cases} \tag{1.20}$$

For each step the interpolation control action is

$$u = v_0^* + (1-c^*)Kx_o^* \tag{1.21}$$

The algorithm of improved interpolation control is as below 2

Algorithm 2 Improved Interpolation control

```
Input: $X$, $U$(constraint for $u$), $\Omega$, $C_N$
Output: a sequence of control signal for system
1: while true do
2:     Measure state $x$
3:     Solving the LP problem in equation (1.20)
4:     Apply the controller described in equation (1.21)
5:     Wait to next time step
6: end while
```

With this algorithm, only one LP problem is asked to be solved for every time instance, which improved the speed of calculation. The price of this algorithm is the number of decision variables. This method has Nm more decision variables compare to the standard one.

Usually, the maximal controlled invariant set of a low-gain controller will be taken as the Ω. Then, with a big enough Ω, a small N is sufficient (i.e. $N = 1$) to get C_N.

Improved IC with Minkowski

Instead of solving (1.20) and (1.21), the improved IC with Minkowski solves,

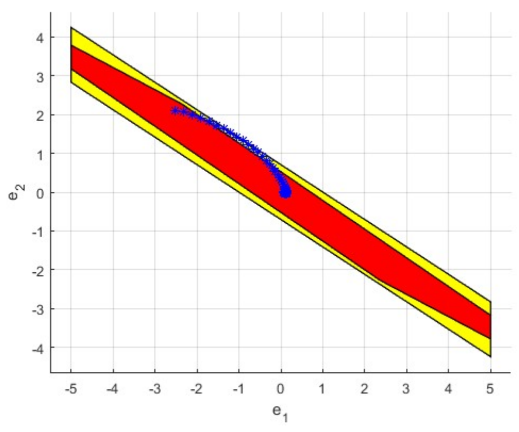
$$\min_{c,u,x_o} \{c\} \quad s.t. \begin{cases} F_N(x(t+1) - x_o(t+1)) \leq cg_N \\ F_o x_o \leq (1-c)g_o \\ F_u u \leq gu \end{cases} \tag{1.22}$$

1.4.2 Implementation

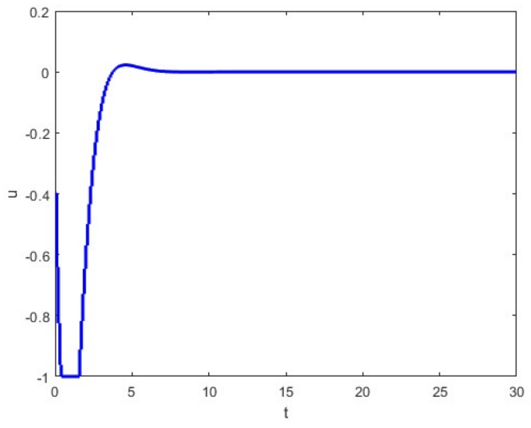
Set $v = 1$, and all constrains as former sections: $\begin{bmatrix} -5 \\ -5 \end{bmatrix} \leq x \leq \begin{bmatrix} 5 \\ 5 \end{bmatrix}$ and $-1 \leq u \leq 1$. Remind that the x here is actually the state of the error system.

In the former cases, the problem can be taken as stabilize at some specific point. Here, a tracking problem is proposed for the IC controller to solve. To simulate this process, set the reference trajectory as x_r and the system state as x_{act}. At every time step, using $x = x_{act} - x_r$ to get the tracking error, then use the Improved IC with Minkowski as presented above to generate a control signal u. After this, instead of using the error dynamics (1.5) to update the system, using (1.2) (needs to discretize) to update the system.

Below 4 is the trajectory of tracking error and control u generated.



(a) Tracking error Trajectory



(b) Control signal u

Figure 4: Performance of Improved IC

Below 5 is the tracking perfomance of this controller. As can be seen, the robot can track quiet well.

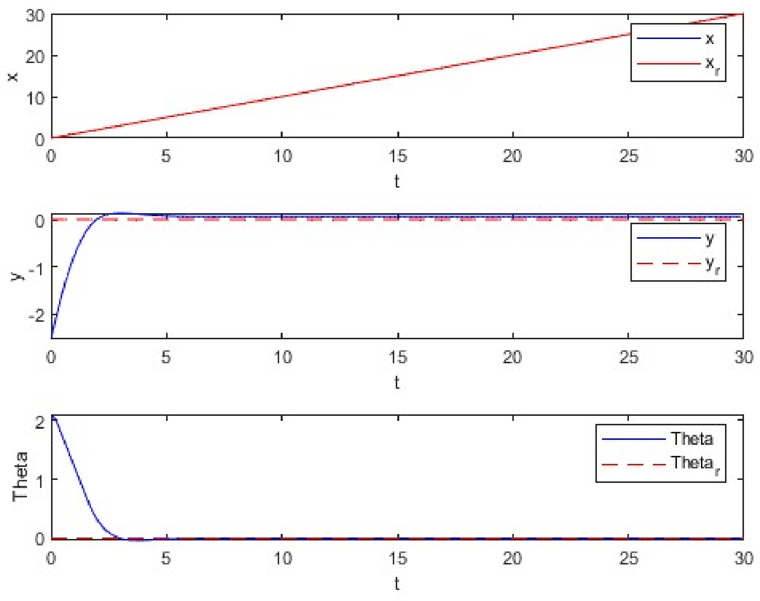


Figure 5: Compare between System State and Reference

Comment:

Because this is a tracking problem, and the system error is not updated by the error dynamic as we used in former cases, additional attention should be payed to the starting point. A feasible starting point is related also to the property of tracking trajectory and the way of updating to the target point. In order to achieve a better performance in tracking, I also tried to froze the target point until the tracking error converges to some extent.

Besides, due to too many simplify to the system model, this model is not suitable for complicated trajectory.

Though the results is not presented here, this model performs very bad in my experiment for a complicated trajectory.

2 Model Predictive Control

2.1 Problem Formulation

The kinematic model is given as (1.2). This model can be expressed in a more general state space form as

$$\begin{aligned} \dot{\xi} &= f(\xi, u) \\ \xi &= \begin{bmatrix} x & y & \theta \end{bmatrix}^T \\ u &= \begin{bmatrix} v & w \end{bmatrix}^T \end{aligned} \tag{2.1}$$

where ξ is the state and u is the input; $f(\cdot)$ represents the corresponding mapping relationship. Since the trajectory status of the reference system is known, the relationship of the state vector and control vector of the reference system is expressed as

$$\begin{aligned} \dot{\xi}_r &= f(\xi_r, u_r) \\ \xi_r &= \begin{bmatrix} x_r & y_r & \theta_r \end{bmatrix}^T \\ u_r &= \begin{bmatrix} v_r & w_r \end{bmatrix}^T \end{aligned} \tag{2.2}$$

Expand equation (2.1) by Taylor series at any reference point (ξ_r, u_r) and retain only the first-order terms and ignore the higher-order terms. The following equation can be obtained:

$$\dot{\bar{\xi}} = \begin{bmatrix} \dot{x} - \dot{x}_r \\ \dot{y} - \dot{y}_r \\ \dot{\theta} - \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v_r \sin\theta_r \\ 0 & 0 & -v_r \cos\theta_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \theta - \theta_r \end{bmatrix} + \begin{bmatrix} \sin\theta_r & 0 \\ \cos\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v - v_r \\ w - w_r \end{bmatrix} \tag{2.3}$$

The linearized error model of the mobile robot is further discretized by Euler method as

$$\bar{\xi}(k+1|t) = A_k \bar{\xi}(k) + B_k \bar{u}(k) \tag{2.4}$$

Where $A_k = \begin{bmatrix} 1 & 0 & -\Delta t v_r \sin\theta_r \\ 0 & 1 & -\Delta t v_r \cos\theta_r \\ 0 & 0 & 1 \end{bmatrix}$, $B_k = \begin{bmatrix} \Delta t \sin\theta_r & 0 \\ \Delta t \cos\theta_r & 0 \\ 0 & \Delta t \end{bmatrix}$ and Δt is the sampling time. With the error system expressed by (2.4), the tracking problem is again transformed into the problem of regulating to the origin. With less assumptions, this model should be able to tracking more complicated trajectories.

2.2 Algorithm

The system equation is the same as[6].

With a given system state $x(k)$, the MPC optimization problem is defined as,

$$V(x(k)) = \min_{\mathbf{u}=\begin{bmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix}} \left\{ \sum_{t=1}^{N} x_t^T Q x_t + \sum_{t=1}^{N} u_t^T R u_t \right\} \tag{2.5}$$

subject to

$$\begin{cases} x_{t+1} = Ax_t + Bu_t, & t = 0, 1, \cdots, N-1 \\ x_t \in X, & t = 1, \cdots, N \\ u_t \in U, & t = 0, 1, \cdots, N-1 \\ x_0 = x(k) \end{cases}$$

[6]Nominal case of IC 1.2

Using the state space model (1.8), the future state variables are expressed sequentially using the set of future control variable values,

$$\begin{cases} x_1 = Ax_0 + Bu_0 \\ x_2 = Ax_1 + Bu_1 = A^2x_0 + ABu_0 + Bu_1 \\ \quad \cdots \\ x_N = A^N x_0 + A^{N-1}Bu_0 + A^{N-2}Bu_1 + \cdots + Bu_{N-1} \end{cases} \tag{2.6}$$

The set of (2.6) can be written in a compact matrix form as,

$$\mathbf{x} = A_a x_0 + B_a \mathbf{u} = A_a x(k) + B_a \mathbf{u} \tag{2.7}$$

with $\mathbf{x} = \begin{bmatrix} x_1^T & x_2^T & \cdots & x_N^T \end{bmatrix}^T$, $\mathbf{u} = \begin{bmatrix} u_0^T & u_1^T & \cdots & u_{N-1}^T \end{bmatrix}^T$ and

$$A_a = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \qquad B_a = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1} & A^{N-2}B & \cdots & B \end{bmatrix}$$

The MPC optimization problem (2.5) can be rewritten as,

$$V(x(k) = \min_{\mathbf{u}} \mathbf{x}^T Q_a \mathbf{x} + \mathbf{u}^T R_a \mathbf{u}) \tag{2.8}$$

where

$$Q_a = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & Q \end{bmatrix}, \qquad R_a = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & R \end{bmatrix}$$

By substituting (2.7) in (2.8), one gets

$$V(x(k)) = \min_{\mathbf{u}} \{\mathbf{u}^T H \mathbf{u} + 2x^T(k) F \mathbf{u} + x^T(k) Y x(k)\} \tag{2.9}$$

where

$$H = B_a^T Q_a B_a + R_a, \qquad F = A_a^T Q_a B_a, \qquad Y = A + a^T Q_a A_a$$

Consider now the constraints (1.9) along the horizon and transform the constraint into a standard form for $\mathbf{u}$(So that the problem is transformed into a Quadratic programming problem). Combining (2.7), the constraint can be expressed as,

$$G\mathbf{u} \leq Ex(k) + S \tag{2.10}$$

Where

$$G = \begin{bmatrix} F_u^a \\ F_x^a B_a \end{bmatrix}, \qquad E = \begin{bmatrix} 0 \\ -F_x^a A_a \end{bmatrix}, \qquad S = \begin{bmatrix} g_u^a \\ g_x^a \end{bmatrix}$$

In which,

$$F_x^a = \begin{bmatrix} F_x & 0 & \cdots & 0 \\ 0 & F_x & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & F_x \end{bmatrix}, \qquad g_x^a = \begin{bmatrix} g_x \\ g_x \\ \vdots \\ g_x \end{bmatrix},$$

$$F_u^a = \begin{bmatrix} F_u & 0 & \cdots & 0 \\ 0 & F_u & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & F_u \end{bmatrix}, \qquad g_u^a = \begin{bmatrix} g_u \\ g_u \\ \vdots \\ g_u \end{bmatrix}$$

From (2.9) and (2.10), the MPC problem can be formulated as,

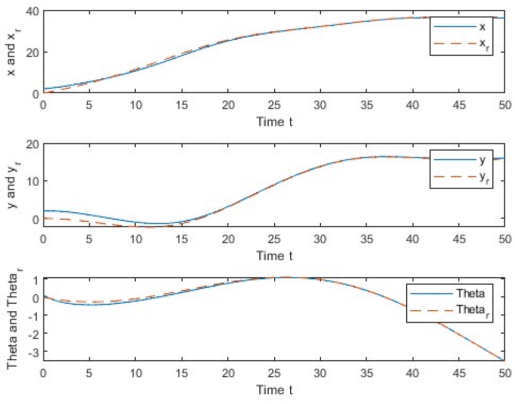
$$V_1(x(k)) = \min_{\mathbf{u}} \{\mathbf{u}^T H \mathbf{u} + 2x^T(k) F \mathbf{u}\} \quad s.t \quad G\mathbf{u} \leq Ex(k) + S \tag{2.11}$$

The term $x^T(k)Yx(k)$ is removed here because it doesn't influence the optimal argument. The value of the cost function at optimum is simply obtained from (2.11) by

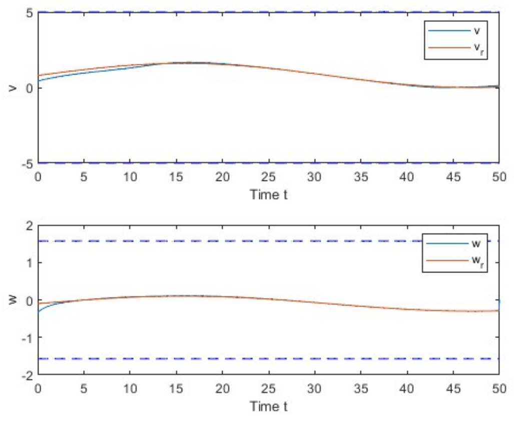
$$V(x(k)) = V_1(x(k)) + x^T(k) Y x(k)$$

2.3 Implementation

Remind that the A and B matrix in our model built in section 2.1 is time varying. To implement the algorithm, A and B is frozen at every time step. The figure 6 below is the performance of MPC controller using the A_a and B_a following the form in section 2.2.



(a) Trajectory compare with Reference



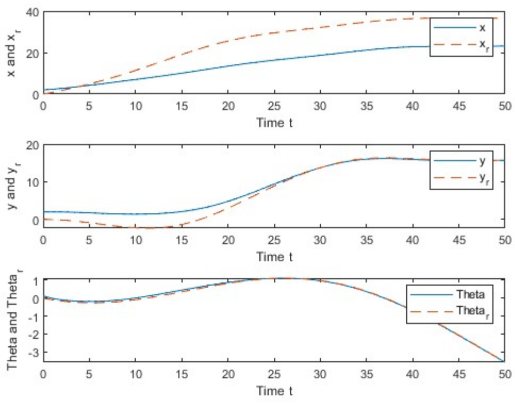
(b) Control signal u

Figure 6: Performance of MPC1

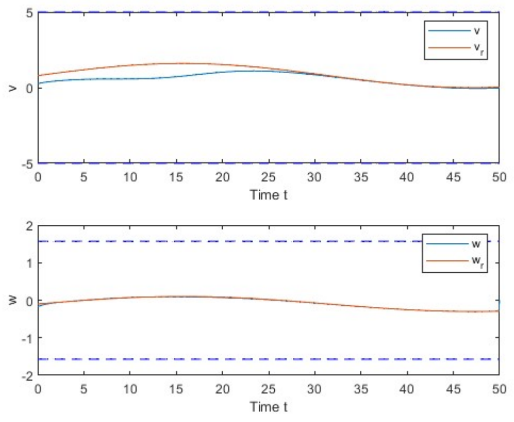
The following formulation of A_a and B_a can also be used to do the MPC. This expression considered the time invariance of A and B

$$A_a = \begin{bmatrix} A_k \\ A_k A_{k+1} \\ \vdots \\ \alpha(k,0) \end{bmatrix}, \qquad B_a = \begin{bmatrix} B & 0 & \cdots & 0 \\ A_k B & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \alpha(k,1) & \alpha(k,2)B & \cdots & B \end{bmatrix}$$

The performance is as below 7



(a) Trajectory compare with Reference



(b) Control signal u

Figure 7: Performance of MPC2

Comment

MPC1 and MPC2 used the same system parameter(e.g reference, Prediction Horizon), but the performance of MPC1 is better for some reason. Because this is also a tracking problem, we have a similar issue with the starting point. This controller will not be able to obtain a solution if the tracking error is too big.

Besides the controller written by myself, I also tried the MPCController in MPT3. MPT3 is a little bit slower than mine, I suppose the reason is because this way needs to create a new system each time step (because the system is time varying).

3 Summary

This project tackled with a tracking problem of a two wheel mobile robot. Two different model are built and four different algorithms are implemented for dealing the problem. Each control algorithm has its own unique charm.

The foundation concept of IC control is quiet easy $--$ doing the interpolation between different controller. Interpolation is something we learned in junior high school. I have to admit that if it's not this course, I will never expect the simple interpolation can perform so good and how powerful this tool can be after combined with some other advanced math tools. The limitation for this controller is that it only works for linear time invariant models. However, with some linearization math tools, it should also be able to work well with the nonlinear cases. On course, several other advanced IC algorithm is presented. The concept of Interpolation control based on Quadratic programming is also very interesting. It's a pity that I was not able to read the details and implement that one before the deadline of this project. I think IC is some algorithm with great potential and flexibility.

The idea of MPC is a little bit abstract compare to IC. This algorithm is very popular in recent years. From all the materials about MPC, they all say that this is some algorithm that can predict the future behavior of a plant. After I understood the algorithm, it's actually also not complicated. The algorithm expanded the system to N horizon, and then calculate the optimal controller at this horizon. In this way, we can save computation resources from the time infinity case and also achieve local optimality. For this algorithm, choosing a right N is very important. If it's too big, it will take too many computation resources. It it's too small, then the performance may be not good. By the way, while I was play with the example 3.1 on book "Constrained Control of Uncertain, Time-Varying, Discrete-Time Systems", I found that the starting point of MPC should be inside C_N set.

Comparing between IC and MPC, IC can save more computation resources and more safe from my point of view. In this sense, IC has better practicality then MPC. After all, the real-time performance and cost-effectiveness of the control algorithm is very important in applications.

There's still lots of inadequacies of my work presented here. For example, there is some problem with the starting point of tracking problem. I suppose it's because the target is changing, but I didn't dive deep in how they effect the result. In my future research work, to discover more about these algorithms, I need to learn more about set and optimization theories.

Thank you very much for giving this course. It's short, but truly improved a lot of my understanding for control theories. Now I'm having a more complete overview about control and can better understand other people's work.

Appendix A Set theoretic methods

A.1 Nominal case

Set invariance:

Set invariance is the fundamental concept for the design of controllers for constrained systems. x is a variable with dynamics and will change along time. If there exist a invariant set Ω for x, it means that, for all $x_0 \in \Omega$, we have $x_k \in \Omega$ for all k.

Pre-set Computation:

$$\mathbf{Pre}(\Omega) = \{x_k | x_{k+1} \in \Omega\}$$

The computation of pre-set needs the system dynamic.

Maximal Invarient Set Ω_{max}:

The maximal invarient set computation Ω_{max} is only for autonomous system, which has the following form of state-space function:

$$x_{k+1} = Ax_k$$

The algorithm to calculate this set is showed as below:

Algorithm 3 Ω_{max} Computation

```
Input: Matrix $A$, state constraint $\mathbf{X}$
Output: $\Omega_{max}$
 1: $\Omega_k = \mathbf{X}$
 2: while true do
 3:     $\Omega_{k+1} = \mathbf{Pre}(\Omega_k) \cap \Omega_k$
 4:     if $\Omega_{k+1} == \Omega_k$ then
 5:         break
 6:     else
 7:         $\Omega_k = \Omega_{k+1}$
 8:     end if
 9: end while
10: $\Omega_{max} = \Omega_k$
```

The emphasis of this algorithm is the computation for $\mathbf{Pre}(\cdot)$. To make it more clear, let's see the following example. Consider the autonomous system:

$$x_{k+1} = Ax_k$$

The constraint for this system is $\Omega = \{x \in \mathbb{R}^n : F_o x \leq g_o\}$. The set $\mathbf{Pre}(\Omega)$ is

$$\mathbf{Pre}(\Omega) = \{x \in \mathbb{R}^n : F_o A x \leq g_o\}$$

While dealing with the real control problem(eg. interpolation control), we can use this to calculate a target set. First, we need to design a controller K that can ensure the convergence of the system. Therefore, $u_k = B * Kx_k$. Then we set $A_c = A + B * K$, and can rewrite the system equation as following:

$$x_{k+1} = Ax_k + Bu_k = A_c x_k$$

The system is transferred into an autonomous system. We can calculate the invariant set for this system with the controller we choose.

Controlled Invariant Set:

Consider the system

$$x_{k+1} = Ax_k + Bu_k$$

X is the constraint for the system state. The set $C \subseteq X$ is controlled invariant iff $\forall x_k \in C$, there exists a control value $u_k \in U$ such that, $\forall\ k \geq 0$,

$$x_{k+1} = Ax_k + Bu_k \in C$$

N-Step Controlled Invariant Set$C_N(\Omega)$:

The concept of N-step Controlled Invariant set is based on Controlled Invariant set. It means that for system state $x \in C_N$, there exist $u_k \in U$ such that the system can converge to the target set Ω within N step($x_N \in \Omega$).

The algorithm for N-step Controlled set computation is showed as below.

Algorithm 4 $C_N(\Omega)$ Computation

```
Input: Matrix $A$, $B$
    State constraints $X$, Input constraints $U$
    Target set $\Omega$
Output: $C_N$
1: $C_k = \Omega$
2: while true do
3:     $C_{k+1} = \mathbf{Pre}(C_k) \cap X$
4:     if $C_{k+1} == C_k$ or $k > N-1$ then
5:         Break
6:     end if
7:     $C_k = C_{k+1}$
8: end while
```

It's worth noticing that if we continue this algorithm until C_N converged, this C_N is also called maximal controlled invariant set.

The computation of $\mathbf{Pre}(\cdot)$ is also very important in this algorithm. However, it's a little bit different from the one we talked for the computation of Ω_{max}. Let

$$\begin{cases} \Omega \in O, O = \{x \in \mathbb{R}^n : F_o x \leq g_o\} \\ u_k \in U, U = \{u \in \mathbb{R}^m : F_u u \leq g_u\} \end{cases}$$

Define the expanded set $P \subset \mathbb{R}^{n+m}$ as

$$P = \{x \in \mathbb{R}^n, u \in \mathbb{R}^m : \begin{bmatrix} F_o A & F_o B \\ 0 & F_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} g_o \\ g_u \end{bmatrix}\}$$

Then the $\mathbf{Pre}(\Omega)$ is

$$\mathbf{Pre}(\Omega) = Proj_x(P)$$

A.2 Robust case:

System Equation

$$x_{k+1} = A_k x_k + B_k u_k + D w_k \tag{A.1}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and $w_k \in \mathbb{R}^d$ are respectively, the measurable state, the input and the disturbance vectors. The matrices $A_k \in \mathbb{R}^{n\times n}$, $B_k \in \mathbb{R}^{n\times m}$ and $D \in \mathbb{R}^{n\times d}$. A_k and B_k satisfy,

$$\begin{cases} A_k = \sum_{i=1}^{q} \alpha_{i,k} A_i \\ B_k = \sum_{i=1}^{q} \alpha_{i,k} B_i \\ \sum_{i=1}^{q} \alpha_{i,k} = 1, \alpha_{i,k} \geq 0 \end{cases} \tag{A.2}$$

where the matrices A_i, B_i are given.

The state, control input and the disturbance are subject to the following bounded polytopic constraints,

$$\begin{cases} x_k \in \ X, \ X = \{x \in \mathbb{R}^n : F_x x \leq g_x\} \\ u_k \in \ U, \ U = \{u \in \mathbb{R}^m : F_u u \leq g_u\} \\ w_k \in W, W = \{w \in \mathbb{R}^d : F_w w \leq g_w\} \end{cases} \tag{A.3}$$

where the matrices F_x, F_u and F_w and the vectors g_x, g_u and g_w are assumed to be constant with $g_x > 0$, $g_u > 0$ and $g_w > 0$. The inequalities are component-wise.

Algorithm

Generally speaking, the robust case have the same procedure we talked in section A.1. The main difference is the computation for Ω_{max} and C_N. Suppose our controller is in the form of $u_k = Kx_k$. Set $A_{ci} = A_i + B_i$. The state constraints of the system can be reformed into the following form:

$$x \in X_c, X_c = \{x \in \mathbb{R}^n : F_c x \leq g_c\} \tag{A.4}$$

where

$$F_c = \begin{bmatrix} F_x \\ F_u K \end{bmatrix} \quad , g_c = \begin{bmatrix} g_x \\ g_u \end{bmatrix}$$

The algorithm to compute a robust invariant set Ω_{max} is showed below in **Algorithm 5**.

Algorithm 5 Robust invariant set computation

```
Input: The matrices $A_{c1}$, $A_{c2}$, $\cdots$, $A_{cq}$, $D$
    The set $X_c$ in (4.4) and $W$ in (4.3)
Output: The robustly invariant set $\Omega$
1: Set $i = 0$, $F_0 = F_c$, $g_0 = g_c$ and $X_0 = \{x \in \mathbb{R}^n : F_0 x \leq g_0\}$.
2: while true do
3:     Eliminate redundant inequalities of the following polytope,
       $$X_1 = \left\{ x \in \mathbb{R}^n : \begin{bmatrix} F_0 \\ F_0 A_{c1} \\ F_0 A_{c2} \\ \cdots \\ F_0 A_{cq} \end{bmatrix} x \leq \begin{bmatrix} g_0 \\ g_0 - \max_{w \in W}\{F_0 D w\} \\ g_0 - \max_{w \in W}\{F_0 D w\} \\ \vdots \\ g_0 - \max_{w \in W}\{F_0 D w\} \end{bmatrix} \right\}$$
4:     if $X_1 == X_0$ then
5:         Break
6:     end if
7:     $X_0 = X_1$
8: end while
9: $\Omega = X_1$.
```

The calculation for C_N is a little bit more complicated compare to Ω_{max}. To calculate the **Pre**(Ω), we first define an expanded set $P \subset \mathbb{R}^{n+m}$, then the set **Pre**$(\Omega)$ is the projection of P at x. The algorithm to compute $N-$step controlled invariant set is showed in **Algorithm 6**.

Algorithm 6 Robustly $N-$step controlled invariant set computation

```
Input: The matrices $A_1$, $A_2$, $\cdots$, $A_q$, D
    Sets $X$, $U$, $W$ and $\Omega_{max}$
    Number $N$, if no $N$ then compute $C_{max}$
Output: The $N-$step robustly controlled invariant set $C_N$
    or maximum controlled invariant set $C_{max}$
 1: Set $C_0 = \Omega_{max}$ and let matrices $F_0$, $g_0$ be the half-space representation of $C_0$, i.e.$C_0 = \{x \in \mathbb{R}^n : F_0 x \leq g_0\}$
 2: iteration number $i = 0$
 3: while true do
 4:     Calculate the expanded set $P$
            $$P = \left\{ x \in \mathbb{R}^n, u \in \mathbb{R}^m : \begin{bmatrix} F_0A_1 & F_0B_1 \\ F_0A_2 & F_0B_2 \\ \vdots & \vdots \\ F_0A_q & F_0B_q \\ 0 & F_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} g_0 - \max_{w \in W}\{F_0Dw\} \\ g_0 - \max_{w \in W}\{F_0Dw\} \\ \vdots \\ g_0 - \max_{w \in W}\{F_0Dw\} \\ g_u \end{bmatrix} \right\}$$
 5:     Calculate the projection of $P$
            $$\mathbf{Pre}(C_0) = \mathbf{Proj}_x(P)$$
 6:     $C_1 = \mathbf{Pre}(C_0) \cap X$
 7:     $i = i + 1$
 8:     if i == N then
 9:         $C_N = C_1$
10:         $N$ step controlled invariant set Found, Break
11:     else if $C_0 == C_1$ then
12:         $C_{max} = C_1$
13:         maximum controlled invariant set Found, Break
14:     end if
15: end while
```

Appendix B Vertex Control

Consider the system of the form

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

Where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, respectively the state, input vectors. The state variables and the control variables are subject to the following polytopic constraints

$$\begin{cases} x_k \in X, X = \{x \in \mathbb{R}^n : F_x x \leq g_x\} \\ u_k \in U, U = \{u \in \mathbb{R}^m : F_u u \leq g_u\} \end{cases} \tag{B.2}$$

Where the matrix F_x, F_u and the vectors g_x, g_u are assumed to be constant with $g_x > 0$, $g_u > 0$ such that the origin is contained in the interior of X, U. It's assumed that the controlled invariant set C_N with some fixed integer $N > 0$ is determined in the form of a polytope, i.e.

$$C_N = \{x \in \mathbb{R}^n : F_N x \leq g_N\} \tag{B.3}$$

Any state $x_k \in C_N$ can be decomposed as follows

$$x = sx_s + (1 - s)x_0 \tag{B.4}$$

where $0 \leq s \leq 1$, $x_s \in C_N$ and x_0 is the origin.

Consider the following optimization problem

$$s^* = \min_{s, x_s}\{s\} \tag{B.5}$$

subject to

$$\begin{cases} s x_s = x \\ F_N x_s \le g_N \\ 0 \le s \le 1 \end{cases}$$

It can be proved that for all state $x \in C_N$ and x is not the origin, the optimal solution of the problem (B.5) is reached iff x is written as a convex combination of the origin and one point belonging to the boundary of the set C_N. The optimal solution to s is

$$s^* = \max\{\frac{F_N}{g_N} x\} \tag{B.6}$$

The explicit solution of the problem (B.5) is a set of n-dimensional pyramids $P_C^{(j)}$, each formed by one facet of C_N as a base and the origin as a common vertex. The controller for $x_k \in C_N^{(j)}$ is

$$u_k = K^{(j)} x_k \tag{B.7}$$

Where $K^{(j)} = U^{(j)}\{X^{(j)}\}^{-1}$. $U^{(j)}$ is the controller at vertexes $X^{(j)}$ of the decomposed component $C_N^{(j)}$ of C_N.

Appendix C Minkowski Norm Minimization Control

Vertex control is a $P-$controller. It requires a lot of off-line efforts: controlled invariant set computation, vertex enumerations, control values at vertices, simplex decomposition. When the C_N has too many vertices it's hard to give a suitable controller to each vertex. We need some algorithm give the u_v automatically. Minkowski norm minimization controller is one alternative to the vertex control.

The idea is to find $u_k \in U$ such that $x_{k+1} = Ax_k + Bu_k \in \lambda C_N$ with the smallest λ, for a given $x \in C_N$. This can be described as an LP problem

$$\min_{u,\lambda}\{\lambda\} \begin{cases} F_c(Ax + Bu) \le \lambda g_c \\ F_u u \le g_u \end{cases} \tag{C.1}$$

Considering that the Minkowski norm minimization controller is a LP problem. The solution might not be unique. One solution is to take the one with the maximum control input norm in order to avoid the idle control problem.

For the robust case, the LP problem is showed as below

$$\min_{u,\lambda}\{\lambda\} \begin{cases} F_c(A_i x + B_i u) \le \lambda g_c, i = 1, 2, \cdots, s \\ F_u u \le g_u \end{cases} \tag{C.2}$$