CHALMERS
UNIVERSITY OF TECHNOLOGY

Hierarchical Model Predictive Control for System With Noise and Inequality Constraints

Master's Thesis in Model Predictive control

Mo Xu

Department of Electrical Engineering
Division of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018

Hierarchical Model Predictive Control for System With Noise and Inequality Constraints

Mo Xu

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2018

Hierarchical Model Predictive Control for System With Noise and Inequality Constraints

Mo Xu

Supervisor: Paolo Falcone, CHALMERS
Examiner: Sebastien Gros, CHALMERS

Master's Thesis 2018
Department of Electrical Engineering
Division of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Chalmers Reproservice/Department of Electrical Engineering
Gothenburg, Sweden 2018

Hierarchical Model Predictive Control for System With Noise and Inequality Constraints

Mo Xu
Division of Signals and Systems
Chalmers University of Technology

Abstract

With the development of technology and the increasing need of advanced control theory, MPC (Model Predictive Control) is receiving more and more attention from industry and academic field nowadays. Due to the constraints arising from physical limit or safety, MPC used in industry usually faces the problem of infeasibility. However, the commonly used solution which is the soft-constrained MPC is costly and time consuming in tuning for the weights of constraints to reach an ideal performance.

To solve the problems mentioned above, a hierarchical MPC algorithm is proposed in this thesis. By grouping the constraints according to their importance and solving the QP (Quadratic Programming) problems hierarchically, this new algorithm let us have more control over the performance of the controller without spending time on tuning the weights of slack variables. In addition, due to the introduction of slack variables, this algorithm is recursive feasible all the time.

An example autonomous driving vehicle problem is introduced in this thesis for illustrating the proposed design methodology. By comparing the performance of the proposed algorithm against a soft-constrained MPC, it can be concluded that the hierarchical MPC behaves better w.r.t constraints with higher priority and guarantees the smallest violation in constraints with higher priority . However, these advantages are achieved at the cost of increased computational complexity. Further research on reducing the calculation cost is needed to make the hierarchical MPC more practical and have a wider application field.

Keywords: Model Predictive Control, recursive feasibility, constraints, hierarchical, autonomous driving, priority

Acknowledgements

This thesis would not be possible without the help of many people.

First I want to thank my supervisor Paolo Falcone for offering me this thesis at Chalmers University of Technology and always helped me out when I got stuck or confused.

Also many thanks to my examiner Sebastien Gros who offered me many guidance and useful literature during this thesis.

Last but not least, I want to thank Ankit Gupta, PhD student of Electrical Engineering, for the vehicle model and all the necessary knowledge of it.

Finally, thanks to my parents and friends who accompanied me in this long period, always offering support and love.

Thank you!

Mo Xu
Gothenburg, Sweden, Sept 2018

Contents

1
Introduction

Model predictive control (MPC) originated from the process industry in the 1970s to deal with optimal control problems with constraints, and today is a well-established control field. The success of predictive control in solving complex industrial processes over the past 30 years has fully demonstrated its great potential in handling complex constrained optimization control problems. A large body of literature on predictive control has pointed out that the greatest attraction of predictive control lies in its ability to explicitly handle constraints. This capability stems from the reformulation of the constrained control problem into an optimization problem, where constraints are naturally enforced. Successful applications indicate that predictive control, as a practically available constraint control algorithm, has been widely accepted by process control industry [12].

In recent years, in many fields such as advanced manufacturing, energy, environment, aerospace, and medical care, there have been many reports that use predictive control to solve constrained control problems. For example supply chain management in semiconductor production, materials manufacturing with high pressure, building energy-saving control, urban sewage treatment, flight control, satellite attitude control, etc [13].

1.1 Model predictive control

Since the beginning of 21st century, with the advancement of science and technology and the development of human society, control engineers have placed higher and higher demands on control that cannot be satisfied with the classic design tools, but hope that the control system can achieve better performance by optimization. However, at the same time, optimization has been constrained by more factors. In addition to the constraints of physical conditions, safety, economy (quality, energy consumption, etc.) and societal aspects (such as environment friendly) must also be considered. These two requirements placed new challenges on the constrained optimization control of complex systems.

Although predictive control has achieved success in industry worldwide, it still has the following limitation. From the perspective of existing algorithms, it is mainly applicable to systems with slow dynamics and high-performance computers, which greatly limits its promotion in a wider field [3].

In addition, while solving MPC in practical work, infeasibility problems sometimes pop out because of the overly conservative constraints or the effect of disturbances. The most common way to solve this problem is to soften some or all of the constraints by introducing slack variables. Engineers could adjust the performance of the controller by tuning the weights of these slack variables, but it usually requires profound experience and takes a lot of time to obtain an desired behavior when there are a lot of constraints in the system.

In summary, although the application of predictive control technology has achieved great success in industry and is considered as the only advanced technology that can effectively solve the optimal control of multivariable constraint systems in a systematic and intuitive way in the process control field, its application fields and objects are still limited by the drawbacks of existing algorithms. For a wider range of application areas and more complex objects, MPC is far from being a systematic methods and techniques.

1.2 Literature review

The increasing need of constrained optimization control in various application field which arises from the development of science and technology drives the research on predictive control theory. The number of papers related to model predictive control which are published in the conference and academic journals increases greatly in the past 10 years. The research field carried out these years can be summarized as following.

1.2.1 Computational aspects

The online computation cost for solving constrained optimization problems greatly limits the application of MPC in industrial area and people have carried out a wide range of research on structures, strategies and algorithms to solve this problem. For example: hierarchical and distributed control structures, offline design/online synthesis and input parameterization strategy and approximate optimization algorithms [5].

In recent years, more attention has been paid to the use of distributed structures to reduce computational complexity in large-scale systems. By decomposing the large-scale optimal problems into multiple small-scale problems, distributed model predictive control not only reduces the computational complexity greatly, but also improves the robustness of the overall system. The research focus of distributed model predictive control includes the handling of coupled dynamics and constraints, the guarantee of global stability and the evaluation of optimality [15].

The improvement or appropriate approximation of the standard optimization algorithm is also a kind of attempt to reduce the online calculation complexity of model predictive control. Proposed in the literature [8], by replacing the commonly used

quadratic programming and semi-definite programming with the extended Newton-Raphson algorithm, the calculation complexity can be reduced greatly in linear MPC. In addition, there is new development in using neural network for quadratic programming. Compared with previous work, the simplified dual neural network has achieved good results in ensuring convergence to the global optimal solution and reducing the computational complexity [10].

1.2.2 Robust predictive control

Since mid-1990s, robust predictive control theory has become the focus of predictive control theory research since the beginning of this century. Most of the early robust predictive control studies are based on the assumption that the states are measurable, but it is not true in a large number of practical systems. Therefore, in recent years, many studies have been conducted on robust predictive control using output feedback instead of state feedback. However, the error of the state reconstruction brings new challenge to ensure the feasibility and stability of the system [11].

In robust predictive control, the requirement of feasible areas, online calculation complexity and controller performances often conflicts with each other. Considering that the constraints in the practical system usually have linear and asymmetrical forms, the literature [9] replaces the traditional ellipsoidal invariant set with polyhedral invariant set since it is more practical and less conservative. The performance can be improved since a larger stabilizable set and extra degrees of freedom is provided by the algorithm.

1.2.3 Nonlinear system

It is common in process industry that the system is nonlinear or stochastic. In recent years, in order to improve the practical application of MPC, new developments in predictive control theory for nonlinear systems and stochastic systems have been made such as nonlinear predictive controller design and stochastic predictive control theory [2].

For nonlinear systems, the modeling method has developed a lot in using the Takagi-Sugeno (T-S) model to characterize its dynamic characteristics. Based on the nonlinear system described by the T-S model, the literature [4] proposed a control law depending on the membership function that guarantees the recursive feasibility of the convex optimization problem and the the closed-loop stability.

The robust predictive control is no longer applicable for a constrained system that are subject to stochastic uncertainty. The literature [1] proposed the concept of probability invariant set to provide a method of handling probabilistic constraints and ensuring closed loop stability.

2
Preliminaries

Vehicle dynamics and MPC used in the thesis are introduced in this section.

2.1 A car model of lateral vehicle dynamics

A car model with 2 df (degrees of freedom) is showed in Figure 2.1. The 2 df are the vehicle lateral position y and the vehicle yaw angle ψ respectively. The vehicle lateral position y is the distance from the vehicle to the center of rotation of the vehicle which is denoted by O along the lateral axis of the vehicle. The vehicle yaw angle ψ is the angle from the global x axis to the direction of the vehicle. And V_x is the longitudinal velocity of the vehicle [14].

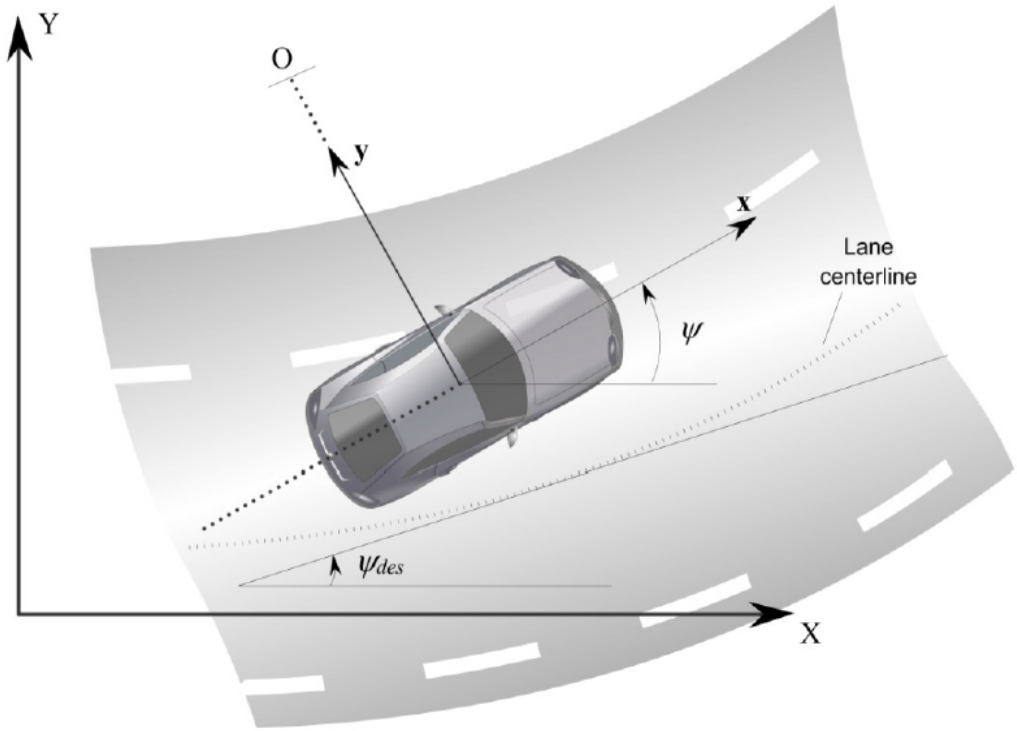

Figure 2.1: Autonomous driving vehicle [14]

By applying Newton's second law of motion along the vehicle's lateral axis, we have

$$ma_y = F_{yf} + F_{yr}, \tag{2.1}$$

Where $a_y = \dot{V}$ is the vehicle's inertial acceleration along y axis and F_{yf} and F_{yr} are the lateral tire forces provided by the front and rear wheels respectively. a_y consists of two parts: the acceleration $\ddot{y}$ which arises from the vehicle's movement in y axis and the centripetal acceleration $V_x\dot{\psi}$. Hence

$$a_y = \ddot{y} + V_x dot\psi, \tag{2.2}$$

By using the expression of a_y from Eq.(2.2) in Eq.(2.1), we get the following equation

$$m(\ddot{y} + V_x dot\psi) = F_{yf} + F_{yr}, \tag{2.3}$$

The torque balance along the z axis yields the following equation

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr}. \tag{2.4}$$

where l_f and l_r are the distances from the axis of the front tire and the rear tire to the gravity center of the vehicle respectively.

It can be shown that the lateral tire force provided by a tire is proportional to the "slip-angle" when it is small. The slip angle of a tire is defined as the angle between the steering direction of the tire and the direction of the vehicle's velocity.

Figure 2.2: Vehicle model [14]

$$\alpha_f = \delta - \theta_{Vf}, \tag{2.5}$$

The slip angle of the front wheel in figure 2.2 can be expressed as Eq.(2.5), where θ_{Vf} is the angle between the vehicle's velocity and its longitudinal direction and δ is the front wheel steering angle.

Similarly, the slip angle of rear tire can be approximated as

$$\alpha_r = -\theta_{Vr}, \tag{2.6}$$

Therefore, the lateral tire force provided by the front wheels can be written as

$$F_{yf} = 2C_{af}(\delta - \theta_{Vf}), \tag{2.7}$$

where the coefficient C_{af} is called the cornering stiffness of the front tire. Since there are two front wheels in the vehicle, the expression is timed by 2.

Similarly the lateral tire force provided by the rear wheels can be written as

$$F_{yr} = 2C_{ar}(-\theta_{Vr}). \tag{2.8}$$

where C_{ar} is the cornering stiffness of the rear tire and the same reason for the factor 2.

The following equations can be used to calculate θ_{Vf} and θ_{Vr}:

$$tan(\theta_{Vf}) = \frac{V_y + l_f\dot{\psi}}{V_x}, \quad tan(\theta_{Vr}) = \frac{V_y - l_r\dot{\psi}}{V_x}. \tag{2.9}$$

Applying small angle approximations and substituting $\dot{y}$ for V_y,

$$\theta_{Vf} = \frac{\dot{y} + l_f\dot{\psi}}{V_x}, \quad \theta_{Vr} = \frac{\dot{y} - l_r\dot{\psi}}{V_x}. \tag{2.10}$$

Substituting Eqs.(2.5), (2.6), (2.9) and (2.10) into Eqs.(2.3) and (2.4), the state space model can be written as

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & -V_x - \frac{2C_{\alpha f}l_f - 2C_{\alpha r}l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{\alpha f}l_f - 2C_{\alpha r}l_r}{I_z V_x} & 0 & -\frac{2C_{\alpha f}l_f^2 + 2C_{\alpha r}l_r^2}{I_z V_x} \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}l_f}{I_z} \end{bmatrix} \delta. \end{aligned} \tag{2.11}$$

2.2 State-space model of lateral deviation and orientation error

It would be much more convenient if the states are lateral deviation and orientation error with respect to the road when the goal is to develop a control system of the vehicle for path-following. Therefore the vehicle model we get in section 2.1 will be rewritten with following states. [14]

e_1: the distance from the vehicle to the center of path
e_2: vehicle orientation with respect to the road.

In order to derive an LTI system, the vehicle speed and the curvature of road are set constant. It is assumed that the radius of the road R is large enough so that the small angle assumptions can be used. The rate of change of the desired orientation of the vehicle is defined as

$$\dot{\psi_{des}} = \frac{V_x}{R}, \tag{2.12}$$

The desired acceleration of the vehicle is defined as

$$\frac{V_x^2}{R} = V_x \dot{\psi_{des}}. \tag{2.13}$$

$\ddot{e_1}$ and e_2 are defined as following

$$\begin{aligned} \ddot{e_1} = (\ddot{y} + V_x\dot{\psi}) - \frac{V_x^2}{R} = \ddot{y} + V_x(\dot{\psi} - \dot{\psi_{des}}) \\ e_2 = \psi - \psi_{des}, \end{aligned} \tag{2.14}$$

and

$$\dot{e_1} = \dot{y} + V_x(\psi - \psi_{des}), \tag{2.15}$$

Eq.(2.15) is consistent with Eq.(2.14) if the velocity V_x is constant. Otherwise, by integrating Eq.(2.14) we obtain

$$\dot{e_1} = \dot{y} + \int V_x e_2 dt. \tag{2.16}$$

Eq.(2.16) gives us a model that is nonlinear and time-variant which makes it very difficult to design a controller for it. So the longitudinal velocity is assumed to be a constant and thus gives us an LTI model.

Substituting Eq.(2.14) and (2.15) into (2.3) and (2.4), we have

$$\begin{aligned} m\ddot{e_1} =& \dot{e_1}(-\frac{2}{V_x}C_{\alpha f} - \frac{2}{V_x}C_{\alpha r}) + e_2(2C_{\alpha f}l + 2C_{\alpha r}) \\ &+ (\dot{e_2} + \dot{\psi_{des}})(-\frac{2}{V_x}C_{\alpha f}l_f + \frac{2}{V_x}C_{\alpha r}l_r) + 2C_{\alpha f}\delta, \end{aligned} \tag{2.17}$$

and

$$\begin{aligned} I_z\ddot{e_2} = & 2C_{\alpha f}\delta l_1 + (e_2 - \frac{\dot{e_1}}{V_x})(2C_{\alpha f}l_f - 2C_{\alpha r}l_r) \\ & + (\dot{e_2} + \dot{\psi_{des}})(-\frac{2C_{\alpha f}l_f^2}{V_x} - \frac{2C_{\alpha r}l_r^2}{V_x}) - I_z\ddot{\psi_{des}}. \end{aligned} \tag{2.18}$$

The state space model can be written as

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & \frac{2C_{\alpha f}+2C_{\alpha r}}{m} & \frac{-2C_{\alpha f}l_f+2C_{\alpha r}l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_zV_x} & -\frac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_z} & -\frac{2C_{\alpha f}l_f^2+2C_{\alpha r}l_r^2}{I_zV_x} \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}l_f}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ -\frac{-2C_{\alpha f}l_f+2C_{\alpha r}l_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_{\alpha f}l_f^2+2C_{\alpha r}l_r^2}{I_zV_x} \end{bmatrix} \dot{\psi}_{des}. \tag{2.19}$$

The path-following problem of the autonomous driving vehicle can therefore be expressed as a state-space equation with dynamics given in Eq.(2.19). Note that the lateral velocity V_x and the road curvature $\dot{\psi}_{des}$ are both assumed to be constant to obtain an LTI system.

2.3 Model predictive control

MPC uses the current plant measurements, the current state, the model, and the constraints to calculate future input that minimize the cost function while satisfying the constraints on both states and input. The MPC typically sends out only the first element of the control sequence, and repeats the calculation when the next input is required [17].

The core of the MPC approach, the receding horizon idea can be expressed as following:

1. At time k, predict the process response over a finite prediction horizon N; this response depends on the sequence of future control inputs over the control horizon M.

2. Calculate the optimum control sequence that gives the minimum cost function while fulfilling the constraints.

3. Apply the first element in the control sequence as input to the system in time k, k→k+1, and return to step 1 [6].
Below is an example of MPC for an LTI system. Consider the following discrete Linear Time-Invariant(LTI) system

$$x(k+1) = Ax(k) + Bu(k) + Ld(k), \tag{2.20}$$

Where $x \in \mathbb{R}^{dimx}, u \in \mathbb{R}^{dimu}$ are the state and input vectors respectively. d is the unknown input disturbance.

The system is subject to a set of inequality constraints that usually come from the physical limitations.

$$C_x x \leq D_x, \quad C_u u \leq D_u, \quad C_f x(N) \leq D_f. \tag{2.21}$$

A regular MPC scheme for system (1) can then be formulated as following

1. At sampling time k, solve the following optimization problem for the optimal control sequence $\mathbf{u} = [u^0(0;x), u^0(1;x), \ldots, u^0(N-1;x)]^T$ with a control horizon of N. Here we assume that the prediction and control horizon are equal.

$$\begin{aligned} V^*(x) = &\min_{\mathbf{u}} \sum_{i=0}^{N-1} l(x(i), u(i)) + V^f(x(N)) \\ s.t \quad & x(i+1) = Ax(i) + Bu(i) + Ld(i) \\ & C_x x \leq D_x, \quad C_u u \leq D_u \\ & C_f x(N) \leq D_f. \end{aligned} \tag{2.22}$$

V is the objective we want to minimize which usually consists of two parts: the stage cost and the terminal penalty. They are usually defined as

$$\begin{aligned} l(x(k), u(k)) &= \frac{1}{2}(\|x\|_Q^2 + \|u\|_R^2), \\ V^f(x(N)) &= \frac{1}{2}\|x(N)\|_p^2. \end{aligned} \tag{2.23}$$

2. Apply the first control input to the system

$$u(k) = u^0(0;x). \tag{2.24}$$

3. Let $k := k+1$ and go to (1).

However, it is usually impossible to guarantee that the optimization problem is feasible at every sampling instant. Due to the presence of state constraints or noise, it is very usual that the optimization problem is infeasible for some initial states, i.e. there is no solution that fulfills all constraints.

Assume $x \in X_N$ is a feasible state which means that there is a solution to the optimization problem for that x. The following question is then natural to ask: after having applied the computed input, will the next state x^+ still within the feasible set X_N? This is clearly a property we want to see in the algorithm which is known as recursive feasibility. The definition of it is as following [6]

Definition: The model predictive controller is feasible for the closed-loop system $x^+ = f(x, k_n(x))$ if

$$x(0) \in X_N \Rightarrow x(k+1) = f(x(k), k_N(x(k))) \in X_N, \forall k \in \mathbb{N}. \tag{2.25}$$

where X_N is the feasible set.

The somewhat disappointing answer is, however, that there is no general guarantee for recursive feasibility in MPC, even in the nominal case with a perfect model and no disturbances. One of the most commonly used method that ensures this characteristic is introduced in the next section.

2.4 Soft-constrained MPC

A method that guarantees the recursive feasibility of MPC is to soften its constraints so that they can be violated if necessary. A straightforward way for softening constraints which is called soft-constrained MPC is to introduce slack variables which are defined such that they are non-zero only if the corresponding constraints are violated. The new soft-constrained MPC can be formulated as

$$\begin{aligned} V^*(x) = & \min_{\mathbf{u}} \sum_{i=0}^{n-1} l(x(i), u(i)) + V^f(x(N)) + f(\varepsilon) \\ s.t \quad & x(i+1) = Ax(i) + Bu(i) + Ld(i) \\ & C[x,u]^T - \varepsilon \leq D \\ & \varepsilon \geq 0. \end{aligned} \tag{2.26}$$

where the C, D matrices are the corresponding inequality constraint matrices in Eq.(2.22). ε is the slack vector. $f(\varepsilon)$ is the penalty function for constraint violations. Generally, it has two types [18]

Quadratic form:

$$f(\varepsilon) = \frac{1}{2}\varepsilon^T S \varepsilon, \tag{2.27}$$

Where S is the weight matrix.

Linear form:

$$f(\varepsilon) = s^T \varepsilon. \tag{2.28}$$

Where s is the vector with positive units.

If the original, hard-constrained MPC is feasible, we would like the soft-constrained problem to produce the same control action. In order to guarantee this, the weights in the cost function have to be chosen large enough such that the QP solver will try to keep the slack variables at zero if possible.

2.5 KKT Conditions

In optimization problems, the KKT (Karush–Kuhn–Tucker) conditions are the first-order necessary conditions for a solution to be optimal. The KKT conditions are usually not solved directly, except in some special cases a closed-form solution can be derived analytically [16].

In the beginning, the KKT conditions were named after Albert W. Tucker and Harold W. Kuhn, who first published the conditions in 1951. However, it was discovered later that this necessary conditions for the optimization problem had been proposed by William Karush in his master's thesis in 1939.

Consider the following optimization problem

$$\begin{aligned} minimize \quad & f(x) \\ s.t \quad & h(x) = 0 \\ & g(x) \leq 0. \end{aligned} \tag{2.29}$$

where x is the optimization variable, f is the cost function, $h_j(1, \ldots, l)$ and $g_i(1, \ldots, m)$ are the equality constraints and inequality constraints respectively. The optimization problem has a total of l equality constraints and m inequality constraints.

Assume x^* is a local minimum. Then there exists unique vectors μ^* and λ^* that fulfill

$$\begin{aligned} \nabla f(x^*) + \nabla g(x^*)\mu^* + \nabla h(x^*)\lambda^* = 0 \\ \mu^* \geq 0 \\ g(x^*) \leq 0, \quad h(x^*) = 0 \\ \mu_i^* g_i(x^*) = 0, i = 1, \ldots, m. \end{aligned} \tag{2.30}$$

Eq.(2.30) are referred to as the KKT conditions. In this equation, λ and μ are the Lagrange multipliers for the equality constraints and inequality constraints respectively. The difference between them is that μ has to be non-negative. The KKT condition Eq.(2.30) can be expressed conveniently as a condition on the Lagrangian L.

$$\nabla_x L(x^*, \mu^*, \lambda^*) = 0, \ where \ \ L(x, \mu, \lambda) = f(x) + \mu^T g(x) + \lambda^T h(x). \tag{2.31}$$

The last equation of Eq.(2.30) is called the complementary slackness condition. The purpose of this equation is that if $g_i(x^*) < 0$ (inactive constraint), then $\mu_i = 0$. Conversely, if $g_i(x^*) = 0$ (active constraints), then $\mu_i > 0$ (the constraint is strictly active) or $\mu_i = 0$ (the constraint is not strictly active) [6].

The KKT conditions are very useful for searching local optimum, and they are also the basis of many optimization algorithms. However, since KKT conditions only provide necessary conditions for a solution to be optimal, the following sufficient conditions for a local optimum is introduced.

Consider x^*, μ^*, λ^* that is found by using KKT conditions with all active constraints being strictly active, then for all $d \neq 0$ such that

$$[\frac{\partial g(x^*)}{\partial x^*}, \frac{\partial h(x^*)}{\partial x^*}]^T * d = 0, \tag{2.32}$$

The following equation must hold

$$d^T \nabla_x^2 L(x^*, \mu^*, \lambda^*) d > 0. \tag{2.33}$$

Then x^* is a local minimum.

3
Algorithm

The hierarchical MPC scheme which is the core of this thesis is proposed followed by discussions about its limitations and drawbacks.

3.1 Main result

Consider the following minimization problem

$$\begin{aligned} S_i = & arg\min_{x\in\Omega} \|w\|^2 \\ s.t \quad & Cx - w \leq D \\ & w \geq 0. \end{aligned} \tag{3.1}$$

The solution set S_i is nonempty since the minimization problem is coercive.

For the new set S_i we have the following proposition:

Proposition 3.1: Given a solution $x^* \in S_i$ and consider each linear inequality $c^j x \leq d^j$ in Eq.(3.1), we have

$$S_i = \Omega \cap \begin{cases} c^j x \leq d^j & if\ c^j x_1^* \leq d^j \\ c^j x = c^j x^* & if\ c^j x_1^* > d^j. \end{cases} \tag{3.2}$$

Which means that all optimal solutions deactivate the same set of inequality constraints and violate other active inequality constraints by a same amount [7].

Proof: Let us consider an optimal solution x^*, w^* to the minimization problem 3.1 within the set Ω. The KKT optimality conditions give that for every vector v not exceeding the boundary of Ω from x^*, we have

$$w^{*T} C v \geq 0. \tag{3.3}$$

This inequality obviously holds when $w^* = 0$. If $w^* > 0$, due to the definition of optimum, all the other points within the set Ω have a w that fulfills $w > 0$, thus we have

$$\begin{aligned} w^{*T} C v &= w^{*T} C(x_1 - x^*) = w^{*T}(Cx_1 - d) - w^{*T}(Cx^* - d) \\ &= w^{*T} w_1 - w^{*T} w^* = w^{*T}(w_1 - w^*). \end{aligned} \tag{3.4}$$

Suppose Eq.(3.3) is invalid, which means that there exists x_1, w_1 fulfills $w^{*T}(w_1 - w^*) < 0$. Consider a point $x_2 = \theta x^* + (1-\theta)x_1,\ \theta \in (0,1)$ between $x^*,\ x_1$, since the set Ω is convex, $x_2 \in \Omega$. Thus, we have

$$\begin{aligned} w_2 &= Cx_2 - D = C(\theta x^* + (1-\theta)x_1) - D \\ &= \theta(Cx^* - D) + (1-\theta)(Cx_1 - D) = \theta w^* + (1-\theta)w_1 \\ \|w_2\|^2 - \|w^*\|^2 &= (\theta w^* + (1-\theta)w_1)^T(\theta w^* + (1-\theta)w_1) - w^{*T}w^* \\ &= (\theta^2 - 1)w^{*T}w^* + (1-\theta)^2 w_1^T w_1 + 2\theta(1-\theta)w^{*T}w_1 \\ &= (1-\theta)[2\theta w^{*T}w_1 + (1-\theta)w_1^T w_1 - (1+\theta)w^{*T}w^*] \\ &= (1-\theta)[(1-\theta)(w_1^T w_1 - w^{*T}w_1) + (1+\theta)(w^{*T}w_1 - w^{*T}w^*)] \\ &= (1-\theta)^2[(w_1^T w_1 - w^{*T}w_1) + \frac{1+\theta}{1-\theta}(w^{*T}w_1 - w^{*T}w^*)]. \end{aligned} \tag{3.5}$$

Since $w^{*T}w_1 - w^{*T}w^* < 0$ and $\frac{1+\theta}{1-\theta} \in (1, +\infty)$, we can always find a θ that makes $\|w_2\|^2 - \|w^*\|^2 < 0$ which contradicts the assumption that x^* is optimum, so inequality 3.3 is proved.

If the QP(Quadratic Programming) problem 3.1 has only 1 optimum solution, proposition 3.1 holds. Next, suppose that it has two optimal solutions, x_1^*, w_1^* and x_2^*, w_2^*. If $w_1^* = w_2^* = 0$, it is obvious that the proposition 3.1 holds. If not, since the set Ω is convex, the vector $x_2^* - x_1^*$ is still within the range of Ω from x_1^*, therefore, we have

$$w_1^{*T}C(x_2^* - x_1^*) \geq 0, \tag{3.6}$$

which is equivalent to

$$w_1^{*T}(Cx_2^* - d) - w_1^{*T}(Cx_1^* - d) \geq 0, \tag{3.7}$$

which is actually

$$w_1^{*T}w_2^* - \|w_1^*\|^2 \geq 0, \tag{3.8}$$

The same be be written from x_2^*

$$w_2^{*T}w_1^* - \|w_2^*\|^2 \geq 0, \tag{3.9}$$

By combining 3.7 and 3.8 we have

$$\|w_2^* - w_1^*\|^2 = \|w_2^*\|^2 + \|w_1^*\|^2 - 2w_2^{*T}w_1^* \leq 0. \tag{3.10}$$

but the squared norm cannot be negative which means it is zero, i.e $w^{*1} = w^{*2}$. The proof is the same for multiple optimum solutions bigger than 2.

Therefore, if $w_1^{*j} > 0 \Rightarrow w_2^{*j} = w_1^{*j} > 0$ which means $c^j x_1^* = c^j x_2^*,\ if\ c^j x_1^* > d^j$. And $w_1^{*j} = 0 \Rightarrow w_2^{*j} = 0 \Rightarrow c^j x_2^* \leq d^j$ which means $c^j x_2^* \leq d^j,\ if\ c^j x_1^* \leq d^j$. The proposition 3.1 is proved.

3.2 Hierarchical MPC for an LTI system

The core of hierarchical MPC is to solve QP problems by grouping its inequality constraints and do the calculation recursively. We have following scheme with constraints' priority decreasing with k,

$$\begin{aligned} & S_0 = \mathbb{R}^n \\ & S_{k+1} = arg\min_{x \in S_k} \|\varepsilon\|^2 \\ s.t \quad & C_k x - \varepsilon \leq d_k, \quad \varepsilon \geq 0. \end{aligned} \tag{3.11}$$

ε is the slack variable introduced in this algorithm to guarantee its recursive feasibility and $C_k x \leq d_k$ is the inequality constraints with priority level k.

Consider an LTI problem in 2.22 and follow the steps in 3.10, a hierarchical MPC scheme can be developed as following by applying the proposition 3.1 directly.

Algorithm 1 Hierarchical MPC scheme

1: Input state space matrices A,B,C,D and cost function V
2: Sort constraints according to their importance, priority decreasing with k
3: Set $\bar{c}$, $\bar{d}$ to empty matrices
4: **for** $k = 0$ to $k = p - 1$ **do**
5:

$$\begin{aligned} & \text{solve} \quad \min_{x^*} \|\varepsilon\|^2 \\ s.t \quad & Ax = B, \quad \begin{bmatrix} \bar{c} \\ C_k \end{bmatrix} x - \varepsilon \leq \begin{bmatrix} \bar{d} \\ D_k \end{bmatrix}, \quad \varepsilon \geq 0 \end{aligned}$$

6: **for** all c^j in C_k **do**
7: **if** $c^j x^* \leq d^j$ **then**
8:

$$\bar{c} = \begin{bmatrix} \bar{c} \\ c^j \end{bmatrix}, \bar{d} = \begin{bmatrix} \bar{d} \\ d^j \end{bmatrix}$$

9: **else**

$$A = \begin{bmatrix} A \\ c^j \end{bmatrix}, B = \begin{bmatrix} B \\ c^j x^* \end{bmatrix}$$

10: **end if**
11: **end for**
12: **end for**
13:

$$\begin{aligned} & \text{solve} \quad \min_{x^*} V \\ s.t \quad & Ax = B, \quad \bar{c}x \leq \bar{d} \end{aligned}$$

14: **return** x^*

3.2.1 Some remark of the algorithm

Remark 1: In the algorithm above it is assumed that the equality constraints have the highest priority while the objective V has the lowest one. However, the equality constraints can be put somewhere else other than the top of the priority pyramid. By rewriting the equality constraints into inequalities, it can treated in the same way as other inequality constraints.

Remark 2: It can be observed from the algorithm above that the number of QP problems needs to solved in each for loop is equal to the number of priority levels. The trade-off between the performance and calculation speed should be considered carefully before grouping the constraints.

Remark 3: Due to increase of calculation time, this hierarchical MPC should only be used when the original MPC problem is infeasible and the number of inequality constraints exceeds a certain value.

4

Simulations

Simulation results of the hierarchical MPC are showed in this chapter and a comparison is made between the new algorithm and the common soft-constrained MPC to show their difference in performances and calculation speed.

4.1 Vehicle model

The model in figure 4.1 we applied the algorithm to is an autonomous vehicle driving on a road with cliffs on both sides. As mentioned in chapter 2, the vehicle speed and the curvature of the road are set to a constant to make the whole problem an LTI system. In addition, a white noise is added to the input. Table 4.1 shows all the parameters we need to formulate the problem.

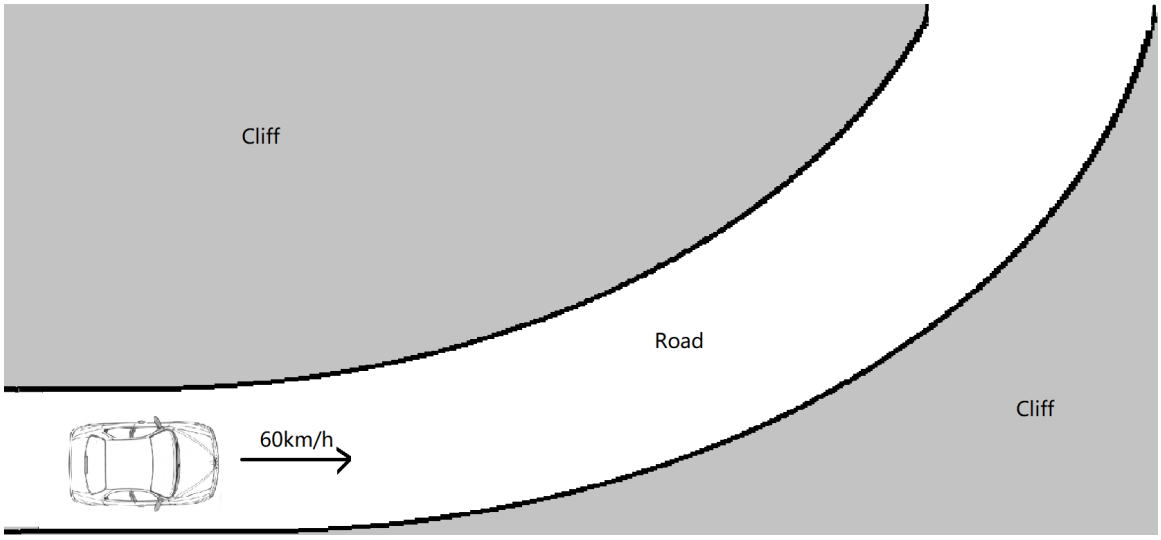

Figure 4.1: Vehicle model

As the magnitude of the speed of vehicle is fixed to 60km/h, the only input to the vehicle is the steering angle of wheels.

Parameter name	Value
Velocity	60km/h
Curvature of road	0.018
Sampling time	1/40s
Vehicle mass	2164kg
Moment of inertia	$4373kg/m^2$
Distance from front wheel axis to center of gravity	1.3384m
Distance from front wheel axis to rear wheel axis	2.984m
Cornering stiffness of front tire	142590
Cornering stiffness of rear tire	228080
Lateral position error constraints	[-0.4m, 0.4m]
Orientation error constraints	$[-\pi/18, \pi/18]$
Lateral speed constraints	[-3m/s, 3m/s]
Steering angle constraints	$[-\pi/36, \pi/36]$
Predict horizon	10
Control horizon	10
mean of input noise	0
variance of input noise	0.16

Table 4.1: Vehicle parameters

By applying the equation (2.19) directly, an continuous LTI system can be formulated from the vehicle model as follows

$$\begin{aligned}\begin{bmatrix}\dot{e_y}\\ \ddot{y}\\ \dot{e_\psi}\\ \ddot{\psi}\end{bmatrix} &= \begin{bmatrix}0 & 1 & 16.6667 & 0\\ 0 & -10.2774 & 0 & -11.5515\\ 0 & 0 & 0 & 1\\ 0 & 2.5313 & 0 & -11.9789\end{bmatrix}\begin{bmatrix}e_y\\ \dot{y}\\ e_\psi\\ \dot{\psi}\end{bmatrix}\\ &+ \begin{bmatrix}0 & 0\\ 65.8919 & 0\\ 0 & -1\\ 43.6411 & 0\end{bmatrix}\begin{bmatrix}\delta+\sigma\\ \gamma\end{bmatrix}.\end{aligned} \tag{4.1}$$

Where e_y, e_ψ are the lateral and orientation error respectively, $\dot{y}$ is the lateral velocity, $\dot{\psi}$ is the yaw rate, δ is the steering angle, γ is the road curvature and σ is the white noise added to the input.

The position error and the steering angle are the most important in the system since they are both directly connected to the vehicle's safety, therefore the cost function is set to be:

$$J = \frac{1}{2}(e_y^2 + \psi^2). \tag{4.2}$$

4.2 Hierarchical MPC

Before we apply the algorithm to the model, the inequality constraints of the system must be ranked first.

The highest priority is given to the position error. The reason is that there are cliffss on both sides of the road and a small violation in position might lead the vehicle fall off the cliff which will threaten the life of passengers.

The second highest priority is given to the input constraint. The steering angle of wheels is limited by the vehicle's design and the violation of it will reduce the life of the vehicle or cause safety problems.

The lateral speed and the orientation error are on the bottom level of this priority pyramid. They are less important than the previous two states since they have no direct influence on safety but only affect the feelings of the passengers on the vehicle.

The simulation result is showed as following

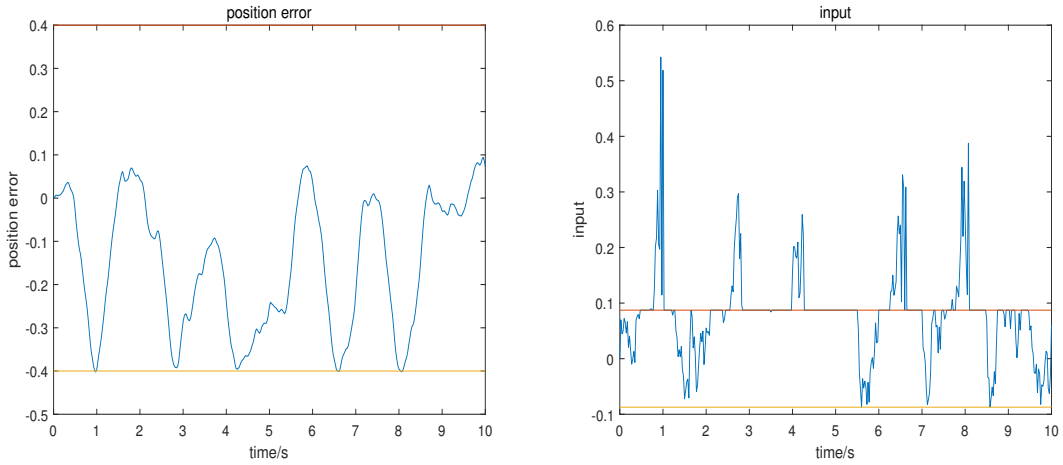

Figure 4.2: Simulation result of the position error and input over a time of 10s

It can be seen clearly from figure 4.2 that the position error stays within the range of [-0.4, 0.4] during the whole simulation period. And although the input violates its constraints from time to time, the total violation time is about 1/5 of the total simulation period. And these violations are due to the introduce of a big input noise.

As can be seen from figure 4.3, the other two states angle error and lateral speed behave well during the whole process. Although a bit noisy, they manage to stay within the range.

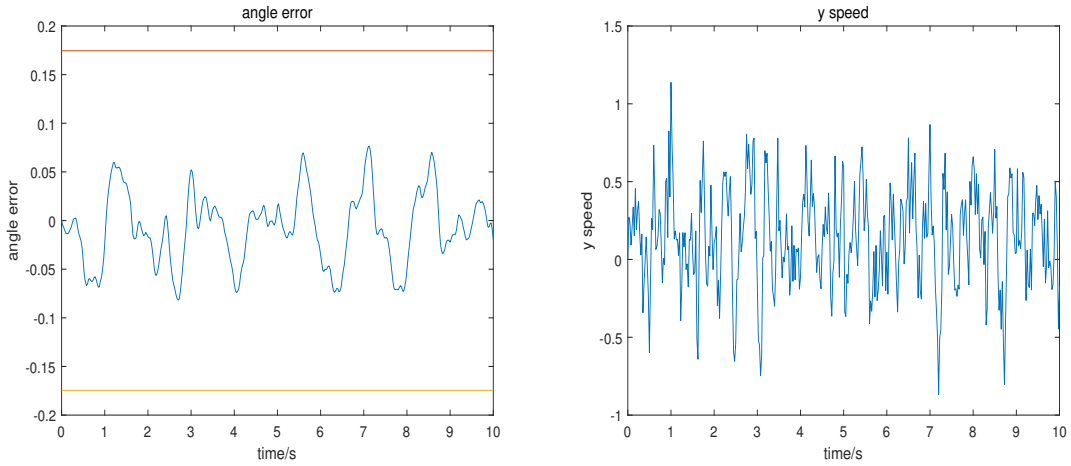

Figure 4.3: Simulation result of the angle error and lateral speed over a time of 10s

4.3 Soft-constrained MPC

To make a comparison with the regular soft constrained MPC, a Simulink model is made in matlab as follows. The two algorithm share a same input noise so that the comparison is meaningful.

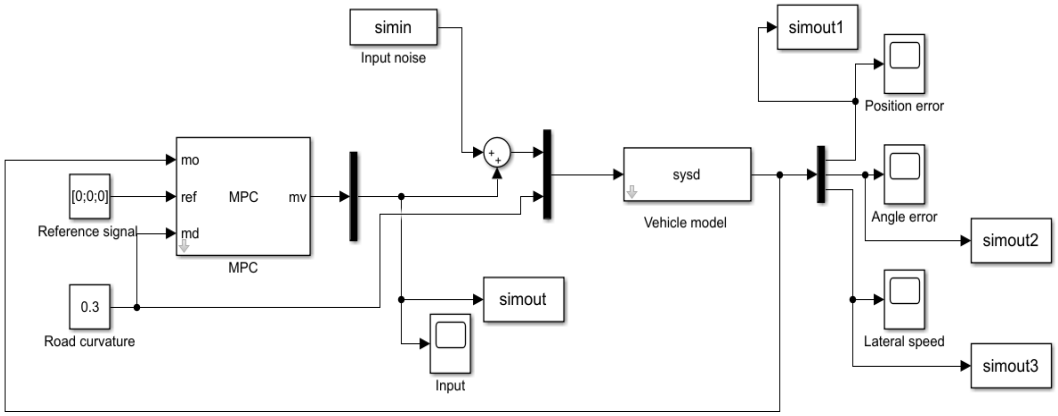

Figure 4.4: Soft-constrained MPC

The ratio between the slack variables of $e_y,\ \dot{y},\ e_\psi,\ \delta$ is set to $10000 : 1 : 1 : 1000$, the simulation results are showed in following figures.

It can be seen in figure 4.5 that the most important constraint which is the constraint on position error is violated 5 times during the whole simulation period of 10 seconds. There is no doubt these violations can be eliminated by tuning the weights of slack variables, but it requires time and is not guaranteed that the violation won't happen again with a new input noise due to its uncertainty.

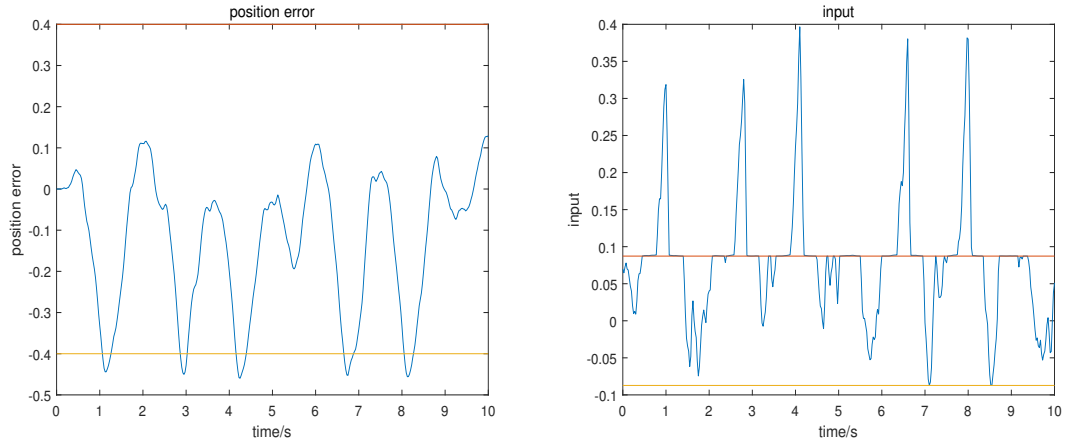

Figure 4.5: Simulation result of regular soft constrained MPC with the same input noise σ

The input behaves quiet similarly to that of hierarchical MPC. They both have 5 peaks during the whole simulation period. A further analysis will be taken in the next section.

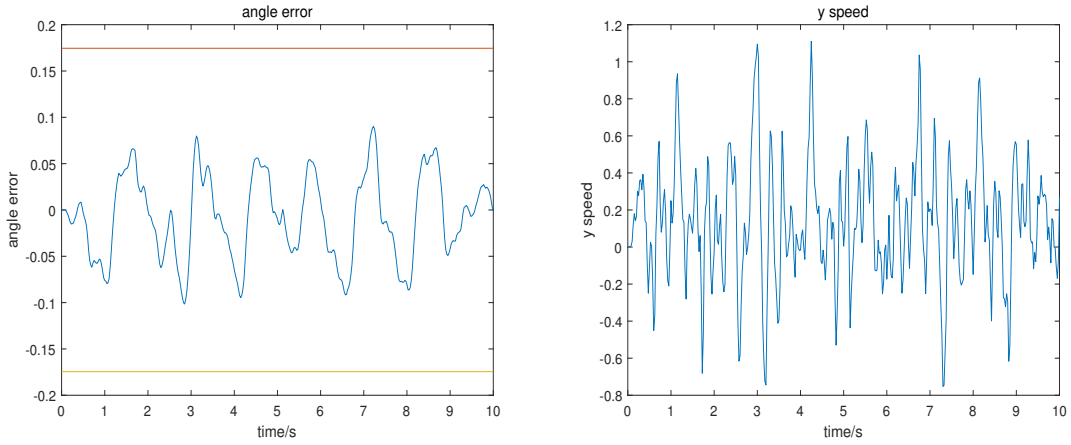

Figure 4.6: Simulation result of regular soft constrained MPC with the same input noise σ

The angle error and lateral speed also succeed in staying within their constraint during the whole process as can be seen in figure 4.6.

4.4 Comparison

A total of 50 different input noise is used to compare the performance of the two algorithms on position error and input.

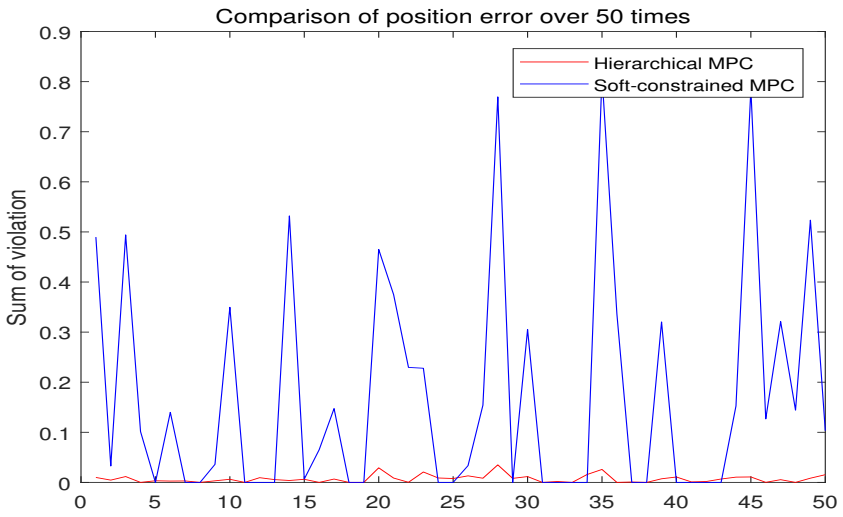

Figure 4.7: Sum of violation of the position error

As can be seen in figure 4.7, the blue line which is the sum of the violation of position error in soft-constrained MPC varies between 0 and 1 during the 50 simulation times, while the red line which is the hierarchical MPC always stays close to zero. In other words, it can be concluded that hierarchical MPC behaves much better than soft-constrained MPC in keeping states of highest priority within constraints.

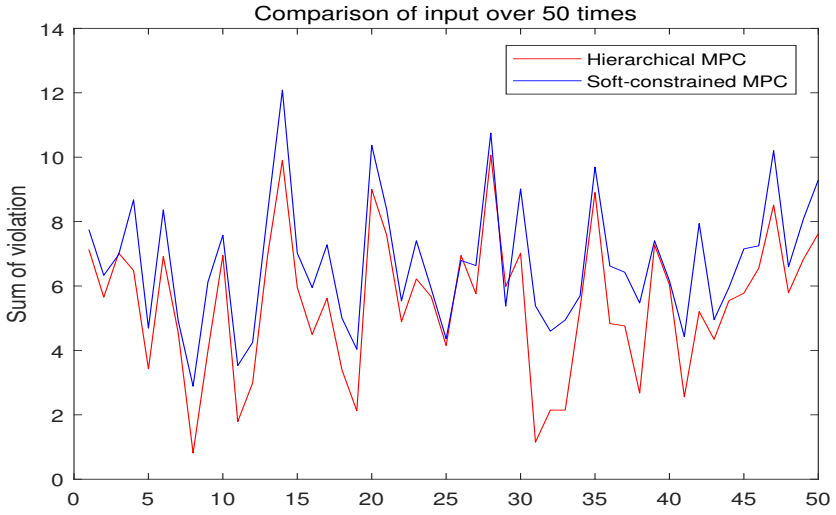

Figure 4.8: Sum of violation of the input

In figure 4.8, it's clear that although the blue line and red line share a similar trend, the red line which is the hierarchical MPC is still about 10% smaller than the blue line in total violation of the input.

However, these two advantages don't come without price. The calculation speed of hierarchical MPC is much slower than that of soft-constrained MPC as can be seen in figure 4.9. The reason behind this is that while soft-constrained MPC only have one QP problem to solve each time, the number of QP problem need to be calculated in hierarchical MPC is equal to the number of its priority level, not to mention the time it spend on grouping active constraints and inactive constraints.

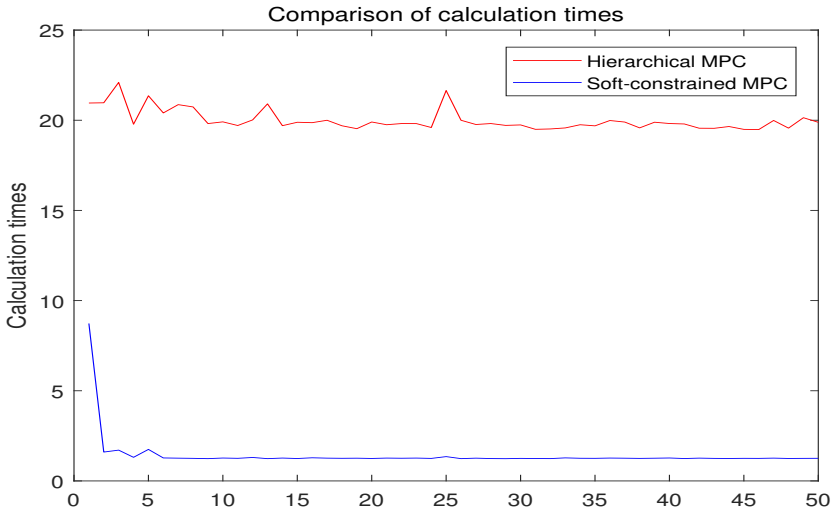

Figure 4.9: Calculation time

5
Discussion

5.1 Priority level of constraints

The first thing we need to do before applying this algorithm is to group the inequality constraints and put them in different priority levels. The inequality constraints of the vehicle model in this thesis is sequenced based on their influence on passengers' safety. However, there is no correct answer to what should be relied on while doing the sequencing. People are free to adjust the orders according to their own needs, e.g. cost, user experience, exterior.

Although in most cases, the equality constraints are the last thing we want to violate which is the reason they are given the highest priority in this thesis, they can be put somewhere other than the top of the priority pyramid if necessary. The method to do this is quite simple, we just need to rewrite the equality constraints as inequality constraints like following

$$Ax = B \Rightarrow Ax \leq B \ \& \ -Ax \leq -B. \tag{5.1}$$

However, people should be careful while doing so since it will not only increase the calculation time, but might also make the solver deliver impossible results.

5.2 Application field of the algorithm

In a system that is originally recursive feasible for a regular MPC controller, applying the hierarchical MPC to it will deliver the same result but cost much more time. In addition, consider the sacrifice of calculation speed of this algorithm, it will be more time-efficient to use the soft-constrained MPC and tune the weights if there are only a few inequality constraints. Last but not least, due to its calculation speed, this hierarchical MPC can only be used in system with slow dynamics right now.

In conclusion, the hierarchical MPC is suitable for controlling infeasible systems constrained by many inequalities with slow dynamics. With the advancement in technology and academic breakthrough, it is without doubt this algorithm will have a wider application field in the future.

5.3 Future research

Some future research based on the algorithm that is worthwhile to investigate is discussed in this section.

5.3.1 Reducing computation cost

The biggest drawback of this algorithm is the calculation speed. However, it can be seen from the algorithm chart that each QP problem is nothing but the last QP problem further constrained with some new inequality constraints. This characteristic makes it possible to reduce its calculation cost in the future.

5.3.2 Tuning the weights

Another way to save the time in tuning slack variables of soft-constrained MPC is using KKT conditions to calculate the differentiate of violation with respect to the corresponding weight. Then by applying Newton's method, the weight that is needed for the required performance can be reached in a limited time.

6
Conclusion

In this paper we have proposed a new MPC scheme for LTI systems. The scheme is based on solving hierarchical QP problems and is recursive feasible even with the presence of noise. The inequality constraints and cost function of the LTI system are grouped according to their importance and then solved hierarchically in the algorithm. After the simulation on a practical problem, it is clear that this scheme is superior to regular soft constrained MPC in the way that no tuning work for the slack variables is needed and has a better performance in the constraints with higher priority.

However, these advantages don't come without price. One drawback of the scheme is that it will sacrifice the constraints with lower priority to get a better performance on the constraints with higher one. In addition, the calculation time of the new scheme is longer than regular soft constrained MPC and it is proportional to the number of the groups of constraints. Consider the calculation cost, this new scheme is suitable when the system has a large noise and constrained to a bunch of inequality constraints.

Bibliography

[1] Model predictive control for systems with stochastic multiplicative uncertainty and probabilistic constraints. *Automatica*, 45(1):167 – 172, 2009.

[2] D. Bernardini and A. Bemporad. Stabilizing model predictive control of stochastic constrained linear systems. *IEEE Transactions on Automatic Control*, 57(6):1468–1480, June 2012.

[3] B. Ding. Dynamic output feedback predictive control for nonlinear systems represented by a takagi–sugeno model. *IEEE Transactions on Fuzzy Systems*, 19(5):831–843, Oct 2011.

[4] B. Ding. Dynamic output feedback predictive control for nonlinear systems represented by a takagi–sugeno model. *IEEE Transactions on Fuzzy Systems*, 19(5):831–843, Oct 2011.

[5] Xiaoning Du, Yugeng Xi, and Shayuan Li. Distributed model predictive control for large-scale systems. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 4, pages 3142–3143 vol.4, June 2001.

[6] Bo Egardt. *Model Predictive Control Lecture Notes*. Jan 2017.

[7] O. Kanoun, F. Lamiraux, and P. Wieber. *Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task*, volume 27, pages 785–792. Aug 2011.

[8] B. Kouvaritakis, M. Cannon, and J.A. Rossiter. Who needs qp for linear mpc anyway? *Automatica*, 38(5):879 – 884, 2002.

[9] Young Il Lee and Basil Kouvaritakis. Robust receding horizon predictive control for systems with uncertain dynamics and input saturation. *Automatica*, 36(10):1497 – 1504, 2000.

[10] Shubao Liu and Jun Wang. A simplified dual neural network for quadratic programming with its kwta application. *IEEE Transactions on Neural Networks*, 17:1500–1510, 2006.

[11] D.Q. Mayne, S.V. Raković, R. Findeisen, and F. Allgöwer. Robust output feedback model predictive control of constrained linear systems: Time varying case. *Automatica*, 45(9):2082 – 2087, 2009.

[12] Manfred Morari and Miroslav Barić. Recent developments in the control of constrained hybrid systems. *Computers Chemical Engineering*, 30(10):1619 – 1631, 2006. Papers form Chemical Process Control VII.

[13] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733 – 764, 2003.

[14] Rajesh Rajamani. *Vehicle dynamics and control*. Springer, 2006.

[15] Brett T. Stewart, Aswin N. Venkat, James B. Rawlings, Stephen J. Wright, and Gabriele Pannocchia. Cooperative distributed model predictive control. *Systems Control Letters*, 59(8):460 – 469, 2010.

[16] Wikipedia. Kkt conditions. Available at `https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions`.

[17] Wikipedia. Model predictive control. Available at `https://en.wikipedia.org/wiki/Model_predictive_control`.

[18] K. Zhao, X. Lu, W. Zheng, and C. Huang. *Direct relaxation of hard-constraint in Model Predictive Control*, pages 2366–2370. May 2012.