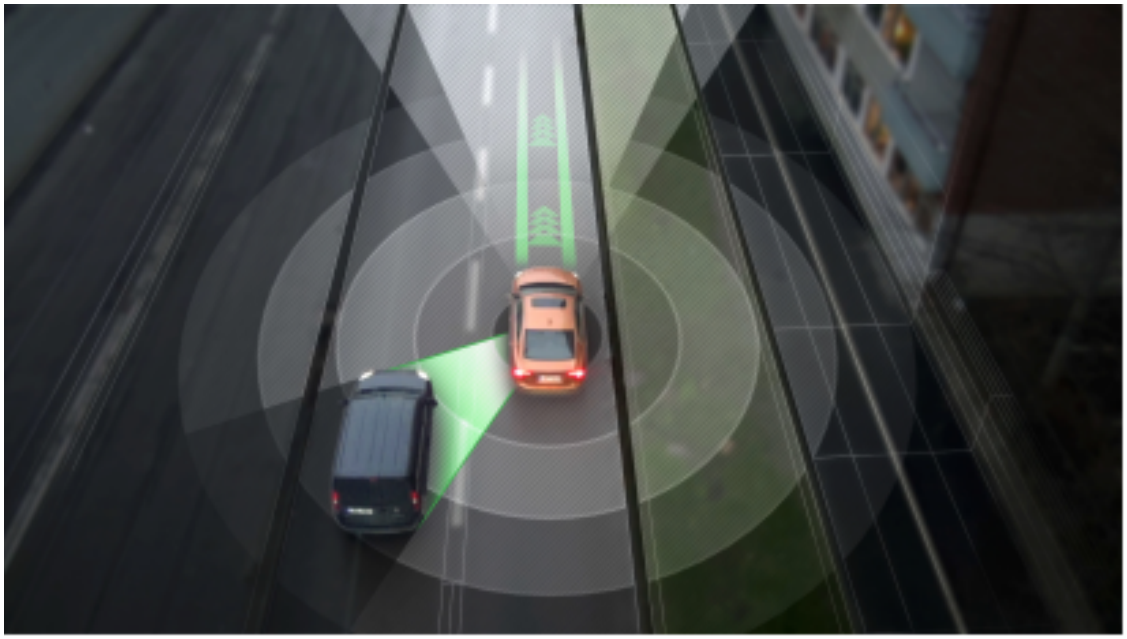
CHALMERS
UNIVERSITY OF TECHNOLOGY



Implementation of An Optimization Algorithm for Autonomous Driving Applications

A fast QP solver with few memory footprint

XU SHENG, WANG TAO

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Implementation of An Optimization Algorithm for Autonomous Driving Applications

A fast QP solver with few memory footprint

XU SHENG, WANG TAO

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Division of Automatic control, Automation and Mechatronics
Automatic control research group
Chalmers University of Technology
Gothenburg, Sweden 2015

Implementation of An Optimization Algorithm for Autonomous Driving Applications
A fast QP solver with few memory footprint
XU SHENG, WANG TAO

Supervisor: Sébastien Gros, Signals and Systems
Examiner: Sébastien Gros, Signals and Systems

Department of Signals and Systems
Division of Automatic control, Automation and Mechatronics
Automatic control research group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Chalmers Bibliotek, Reproservice
Gothenburg, Sweden 2015

Implementation of An Optimization Algorithm for Autonomous Driving Applications
A fast QP solver with few memory footprint
XU SHENG, WANG TAO
Department of Signals and Systems
Chalmers University of Technology

Abstract

Model Predictive Control (MPC) has attracted increasing interest in recent years and become popular in the time-critical applications with the help of the improvement in computational power and efficient numerical methods. However, the scope for applying either explicit or implicit MPC controller is confined to small-scale problems when it comes to real-time implementation. Technically, for a typical linear quadratic constrained MPC, the structure of the Quadratic Programmings (QP) that arises in MPC can be exploited to simplify the implementation of the numerical methods. In this thesis, an appropriate re-ordering of the variables is used and the methods that dealing with gradient information is investigated and applied for solving the QP problem.

The primary purpose of this thesis project is to develop a QP solver for the longitudinal motion control problems faced in autonomous driving applications. The convergence rates of several variants of gradient optimization methods are introduced based on the previous work and Nesterov's fast gradient method is selected due to its optimal convergence rate, simple algorithmic scheme and applicability to a wider area of applications even in limited resources systems such as embedded systems. Meanwhile, several techniques used for removing the constraints in optimization problems are also presented e.g. Euclidean projection is used for dealing with inequality constraints, Lagrangian relaxation, which relaxes the equality constraints and decouples the original optimization problem into a two-level one, and hot-start method. At last, a pre-conditioner (linear transformer) is used to find the optimal Lipschitz constant. The solver which includes all the properties mentioned above is finally implemented in C for better performance in real applications.

Keywords: Model Predictive Control; Quadratic Programming; Fast Gradient Methods; Lagrangian Relaxation; Euclidean Projection; Hot Start; Preconditioning.

Acknowledgements

First of all, we would like to express our sincere gratitude to our supervisor Assistant Professor Sébastien Gros for his systematic guidance, patience and friendly attitude. He is always willing to answer our questions and offer us some good ideas for solving problems.

We are also very thankful to Dr Emil Klintberg for the help and support during our master thesis work. He was continuously providing us with the excellent explanations of the mathematical concepts required for understanding the convex optimization problems.

We would like to express our gratitude to Dr Julia Nilsson not only for giving us some good advice for building the MPC model, but also for useful simulation data for testing our solver.

A thank also goes to Mohammad Ali for offering us such an interesting and challenging thesis topic.

We are also grateful to the Systems and Signals department for a friendly studying environment, which makes us feel happy during our whole master study and thesis.

Finally, we would like to thank our parents and friends who always support and accompany with us.

XU SHENG, WANG TAO, Gothenburg, Sweden, June 2015

Contents

Notation

Specific Sets

$\mathbb{R}$	Real numbers.
$\mathbb{R}^n$	Real n-vectors ($n \times 1$ matrices).
$\mathbb{R}^{m \times n}$	Real $m \times n$ matrices
$\mathbb{R}_+$	Nonnegative real numbers.
$\mathbb{R}_{++}$	Positive real numbers.
$\mathbb{Z}$	Integers.
$\mathbb{Z}_+$	Nonnegative integers.
$\mathbb{S}^n$	Symmetric $n \times n$ matrices.
$\mathbb{S}^n_+$	Symmetric positive semidefinite $n \times n$ matrices.
$\mathbb{S}^n_{++}$	Symmetric positive definite $n \times n$ matrices.

Vectors and Matrices

$\lambda_{max}(\mathbf{X})$	Maximum eigenvalue of matrix $\mathbf{X}$.
$\lambda_{min}(\mathbf{X})$	Minimum eigenvalue of matrix $\mathbf{X}$.
$\sigma_{max}(\mathbf{X})$	Maximum singular value of matrix $\mathbf{X}$.
$\sigma_{min}(\mathbf{X})$	Minimum singular value of matrix $\mathbf{X}$.
$\boldsymbol{I}_n$	$n \times n$ Identity matrix.
$\mathbf{X}^T$	Transpose of matrix $\mathbf{X}$.

Norms and Inequalities

$\|\mathbf{x}\|$	2-norm of vector $\mathbf{x}$.
$\mathbf{x} \preceq \mathbf{y}$	Componentwise inequality between vectors $\mathbf{x}$ and $\mathbf{y}$.
$\mathbf{x} \prec \mathbf{y}$	Strict componentwise inequality between vectors $\mathbf{x}$ and $\mathbf{y}$.

Functions and Derivatives

$f : A \mapsto B$	f is a function on the set $\mathbf{dom} f \subseteq A$ into the set B.
$\mathbf{dom} f$	Domain of function f.
∇f	Gradient of function f.
$\nabla^2 f$	Hessian of function f.

Chapter 1

Introduction

1.1 Setting

Due to the increasing amount of personal car ownership, the problem of traffic safety has become more and more serious [1]. Within this context, the concept autonomous driving has been proposed and become a fast-developing and promising area since the 1980s [2]. An autonomous car can be roughly interpreted as the car with self-driving assistance functions [3]. In order to realize the autonomous and safe driving, autonomous driving technology utilizes the on-board sensors, e.g, radars, ultrasonic sensors and GPS to sense its environment; steer and control the speed; plan the path dynamically based on the current traffic situation [4]. Undoubtedly, autonomous driving technology will bring enormous benefits to the automotive industry.

As one of the leading manufactures of vehicles in the world, Volvo Cars has always been considered as one of the leading corporations in the intelligent vehicle field. According to the unique pilot proposed by Volvo, there will be 100 self-driving vehicles to real customers in 2017. The vehicles will be capable of driving autonomously on a highly trafficked ring route around the city of Gothenburg without requiring driver supervision.

Many autonomous driving applications involve longitudinal and lateral motion control of the vehicles [5]. Longitudinal control aims at controlling the distance between the ego vehicle and the surrounding vehicles and obstacles in order to avoid collision. Lateral control means the steering control in the direction vertical to the longitudinal direction and keep the vehicle in the desired trajectory. Most of the longitudinal and lateral control problems are formulated under the framework of Model Predictive Control (MPC) [6]. In this thesis, only the longitudinal control problem is considered and the optimal value of key variables such as position, velocity and acceleration in the longitudinal direction when performing self-driving behaviours (i.e. lane changing) are obtained as the solutions of a finite time horizon optimal control problem with constraints. Specifically, a cost function is minimized subject to the application based constraints which may include vehicle dynamics, limitations of the system and collision avoidance with the surrounding vehicles and obstacles [6].

When it comes to real-time implementation, the large scale MPC problem requires a complicated linear algebra handling which makes it difficult to code in the real-time system because of the computational complexity. To reduce the computational burden, the structure of QPs arises in the typical linear-quadratic MPC problems can be exploited and solved efficiently by some numerical methods. The QP problems considered in this master thesis are restricted to convex QP problems, which means the cost function and the inequality constraints are convex; the equality constraints are affine. However, the hard constraints (the constraints that the system must adhere to) introduced in the optimization problem will complicate the problem and meanwhile, in some cases it is not necessary to get the exact optimal solution which may take a large number of iterations. Therefore, dual decomposition, and more generally Lagrangian relaxation is used to introduce the hard equality constraints into the objective function and soften the hard constraints by using a vector called Lagrangian multipliers as the penalty for violating the constraints. On the other hand, the inequality constraints can be removed by Euclidean projection elegantly if the inequality constraints are easy to handle, which will release the computation burden properly.

1.2 Project Purpose and Goal

The purpose of this project is to evaluate the performance of the different numerical methods. The methods are applied to the convex QP problems which are transformed from the MPC formulations in order to avoid the complicated linear algebra handling.

The aim of this project is to develop a QP solver for the general autonomous driving applications by using an efficient optimization algorithm. The solver should then be validated in a specific application, e.g. lane changing, to make sure that it fulfills the control requirements.

1.3 Structure of the Thesis

The rest of this thesis is configured as following.

An introduction to the methods for dealing with the convex optimization problems are given in chapter 2, which includes gradient method, fast gradient method and some of its variants. Meanwhile, the comparison between the performance of different optimization methods is also given in that chapter.

Chapter 3 deals with the principle of the MPC and QP problem formulation. A re-ordering of variables is introduced to transform MPC into QP. It was followed by

the Lagrangian relaxation which used for relaxing the equality constraints.

Chapter 4 introduces the implementation background of the QP solver. First, the MPC formulation is given, which includes a longitudinal motion based state-space model; a reference tracking objective function; constraints on the inputs and state variables. Then a QP formulation is presented which is followed by the Lagrangian relaxation and parameter analysis of the algorithm. Finally, some techniques for speeding up the solver are given and the results of the solver derived from real applications are presented.

Chapter 2

Methods and Algorithms

According to the receding horizon idea of MPC, the high-intensive computation task comes from that an optimization problem needs to be solved at every sampling instant. In the past, when the first appearance of MPC in the process industries, this is not an issue because the sampling periods are usually very long (in the range of minutes or hours) which leaves enough time to compute the solution. The last two decades, however, witnessed an increasing interest in applying MPC to the fast-sampled systems (e.g. embedded systems) and efficient algorithms has become the core element of the MPC problems.

The MPC problem considered in this thesis is constrained linear-quadratic MPC and its introduction is postponed to chapter 3. There are vast number of optimization methods to solve this kind of MPC and in general they can be categorized into online method and offline method [7].

A commonly used offline method is multi-parametric programming which specializes in obtaining the explicit solution beforehand for each initial state. The state space is partitioned into several polyhedral subsections in order to get the explicit solution which is in the form of a piecewise affine map. Then the control action is implemented online in the form of a look-up table [8]. The main drawback of this method is that the partitioned polyhedral subsections may grow exponentially with the input and state dimension. Possible solutions to that can be found in [9, 10], but still suffers from the restriction of the problem dimensions.

Online methods, on the other hand, are typically towards to the large-scale MPC problems. Two important iterative methods are active set methods and interior point methods. However, for active set methods the convergence rate analysis is not a priori and the computational complexity certificates for interior point methods is too conservative compared to the practical observed one. Meanwhile, the iterative scheme for both methods involves the solution of a system of linear equations which is not simple and may cause numerical issues [11].

The gradient methods which avoid all the problems mentioned above are introduced in this chapter. Also, the Nesterov fast gradient methods which have a faster con-

vergence rate than the classic gradient method and a simple algorithmic scheme are extracted from [12].

In this chapter the simple unconstrained optimization problem is considered:

$$f^* = \min_{x \in \mathbb{R}^n} f(x) \tag{2.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is assumed to be strongly convex (Definition A.2.3, Proposition A.2.2).

2.1 Gradient Method

First of all, an optimization problem without any constraints is considered here. For this kind of problem, typical algorithms generate a relaxing sequence $\{x_k\}_{k=0}^{\infty}$ by using the line search scheme:

$$x_{k+1} = x_k + h_k d_k \tag{2.2}$$

where h_k is the step size and d_k is the searching direction. In order to make sure f is decreased at each iteration $f(x_{k+1}) \leq f(x_k)$, the searching direction is obvious by checking the first order Taylor series expansion around x_k:

$$f(x_{k+1}) = f(x_k) + h_k \nabla f(x_k)^T d_k + o(h_k ||d_k||) \tag{2.3}$$

Since the term $h_k \nabla f(x_k)^T d_k$ dominates the term $o(h_k||d_k||)$ with small positive h_k, a reasonable choice of d_k is to make sure that $\nabla f(x_k)^T d_k < 0$. The gradient method, with the simple choice $d_k = -\nabla f(x_k)$ is introduced here. A general gradient method scheme is given as following:

Step 1 Set a initial $x_0 \in \mathbb{R}^n$

Step 2 Compute $x_{k+1} = x_k - h_k \nabla f(x_k) \quad k = 1, 2...$

Step 3 Go Back to Step 2 until find an optimal solution($x_{k+1} = x_k$)

There are also some other choices of the direction d_k and different ways of choosing the step size h_k [13]. Here the constant step size strategy is used.

Figure 2.1 shows that how the gradient method works. $\|\nabla f(x)\|$ will tend to be 0 if the problem is feasible. From the above scheme, it is obvious that the convergence rate of the gradient method is determined by $\nabla f(x)$ and h_k. It has been proved by Nesterov that for a smooth convex function with Lipschitz continuous gradient constant L (Definition A.2.4), the optimal choice of the constant step size is $h_k = \frac{1}{L}$ and the upper bound of the convergence rate of this step size strategy is [12]:

$$f(x_k) - f^* \leq \frac{2L||x_0 - x^*||^2}{k + 4} \tag{2.4}$$

where f^* is $f(x^*)$ and x^* is the minimum of $f(x)$.

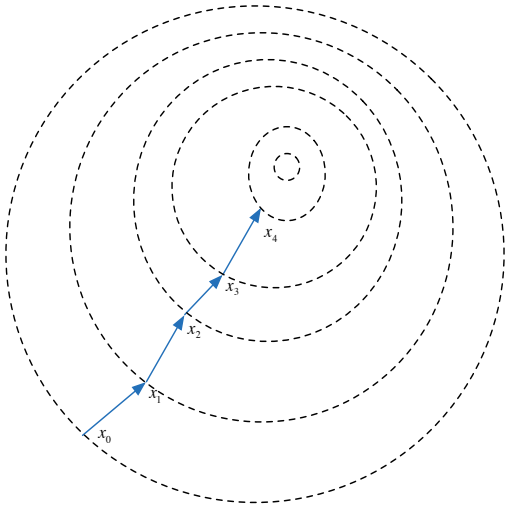


Figure 2.1: Simple illustration of gradient method

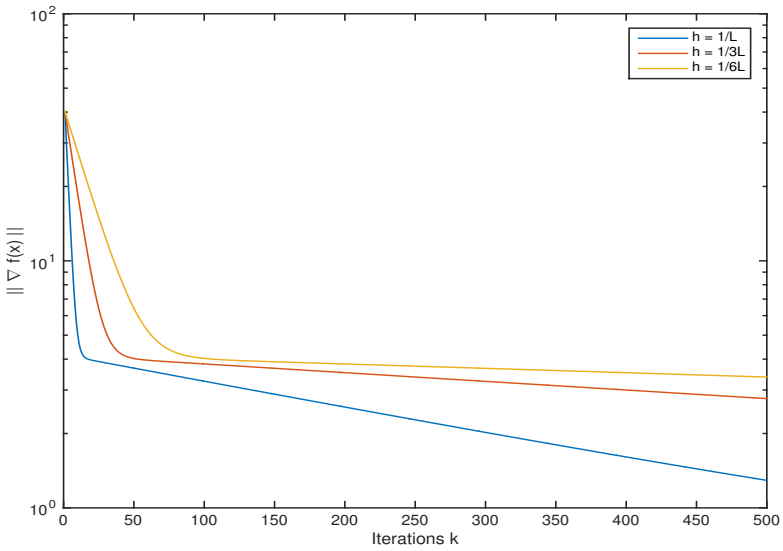


Figure 2.2: Convergence rate of gradient method for convex case

Figure 2.2 shows the convergence rate of the gradient method with different constant step sizes.

If the function is also strongly convex with convexity parameter μ (Definition A.2.3), the optimal step size which minimizes the upper bound of the convergence rate is $h_k = \frac{2}{L+\mu}$, and the following convergence rates are obtained:

$$||x_k - x^*|| \leq \left(\frac{Q_f - 1}{Q_f + 1}\right)^k ||x_0 - x^*|| \tag{2.5}$$

$$f(x_k) - f^* \leq \frac{L}{2}\left(\frac{Q_f - 1}{Q_f + 1}\right)^{2k} ||x_0 - x^*||^2 \tag{2.6}$$

where $Q_f = \frac{L}{\mu} \geq 1$ is called the condition number of the function f.

Even if μ is not guaranteed to obtain, by choosing the step size $\frac{1}{L}$ still results in the following convergence rates:

$$||x_k - x^*||^2 \leq \Big(\frac{Q_f - 1}{Q_f + 1}\Big)^k ||x_0 - x^*||^2 \tag{2.7}$$

$$f(x_k) - f^* \leq \frac{L}{2}\Big(\frac{Q_f - 1}{Q_f + 1}\Big)^k ||x_0 - x^*||^2 \tag{2.8}$$

If the knowledge of convexity parameter is included in the step size $h_k = \frac{2}{L+\mu}$, it can be seen that convergence rate is square of that for $\frac{1}{L}$ case. The convergence rate in (2.6), (2.8) is linear while the convergence rate in (2.4) is sublinear, and the definitions of different convergence rate are given in the Appendix (Appendix A.4). The convergence rate for gradient method can be expressed as following by considering the convex and strongly convex cases for $\frac{1}{L}$ step size strategy:

$$f(x_k) - f^* \leq min\left\{\frac{L}{2}\Big(\frac{Q_f - 1}{Q_f + 1}\Big)^k, \frac{2L}{k+4}\right\} ||x_0 - x^*||^2 \tag{2.9}$$

The lower complexity bound is also an important property of the objective function when the gradient method is implemented. According to Nesterov, an assumption needs to be made before analyzing complexity of the method:

Assumption 2.1.1 *An iterative method $\mathcal{M}$ generates a sequence of test points $\{x_k\}$ such that*

$$x_k \in x_0 + \mathbf{Lin}\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\}, \quad k \geq 1. \tag{2.10}$$

where $\mathbf{Lin} : \mathbb{R}^{p\times k} \mapsto \mathbb{R}^p$ means the linear combination.

Clearly, gradient method fulfills this assumption. Then the following lower complexity bound for the smooth convex function is derived from the following theorem:

Theorem 2.1.1 *For any $k, 1 \leq k \leq \frac{1}{2}(p-1)$ and any $x_0 \in \mathbb{R}^n$, there exist a smooth convex function f such that for gradient method the following bounds exists:*

$$f(x_k) - f^* \geq \frac{3L||x_0 - x^*||^2}{32(k+1)^2} \tag{2.11}$$

$$||x_k - x^*||^2 \geq \frac{1}{8}||x_0 - x^*||^2 \tag{2.12}$$

For the strongly convex function, the bounds are described in the following theorem.

Theorem 2.1.2 *For any $x_0 \in \mathbb{R}^\infty$, there exist a smooth strongly convex function f such that for gradient method the following bounds exists:*

$$f(x_k) - f^* \geq \frac{\mu}{2}\left(\frac{\sqrt{Q_f} - 1}{\sqrt{Q_f} + 1}\right)^{2k} ||x_0 - x^*||^2 \tag{2.13}$$

$$||x_k - x^*||^2 \geq \left(\frac{\sqrt{Q_f} - 1}{\sqrt{Q_f} + 1}\right)^{2k} ||x_0 - x^*||^2 \tag{2.14}$$

For ill-conditioned function (large condition number), there may be an increasing of 'zigzag' behaviours when the gradient iteration points approaching to the optimal point [13]. It is not the optimal method according to Nesterov, since the upper bound of the convergence rate is not in the same order as the lower bound. Hence, a more effective method is required to improve the performance.

2.2 Fast Gradient Method

The fast gradient method was developed by Yurii Nesterov in 1983 [14]. The fast gradient methods often referred to as momentum methods, which means the updating of the current iteration sequence depends on the previous iterations, and the momentum grows as the iteration continues [15]. The main idea of this method is to create a new scheme by using an estimated sequence.

Definition 2.2.1 *A pair of sequences* $\{\phi_k(x)\}_{k=0}^{\infty}$ *and* $\{\lambda_k\}_{k=0}^{\infty}$, $\lambda_k \geq 0$ *is called an estimated sequence of* $f(x)$ *if* $\lambda_k \to \infty$ *and*

$$\phi_k(x) \leq (1-\lambda_k)f(x) + \lambda_k\phi_0(x), \quad \forall x \geq 0,\ \forall x \in \mathbb{R}^n. \tag{2.15}$$

The next lemma shows that why these sequence is needed.

Lemma 2.2.1 *For a given estimated sequence of* $f(x)$ *and a sequence* $\{x_k\}$*, it holds*

$$f(x_k) \leq \phi_k^* \equiv \min_{x\in\mathbb{R}^n} \phi_k(x) \tag{2.16}$$

then $f(x_k) - f^* \leq \lambda_k[\phi_0(x^*) - f^*] \to 0$.

Thus, computing the convergence rate of the λ_k provides the same result for the convergence rate of the optimization process x_k. In order to ensure (2.16), a quadratic function is formed:

$$\phi_0(x) = \phi_0^* + \frac{\gamma}{2}\|x - v_0\|^2 \tag{2.17}$$

The details about building an estimate sequence and update ϕ_k^*, γ_k and v_k are given in [12].

2.2.1 Fast Gradient Method with Constant Step Size

According to the previous discussion, the fast gradient method based algorithm is obtained as follows:

Algorithm 1 Fast Gradient Method with Constant Step Size

Require: Initial inputs: $x_0 = y_0 \in \mathbb{R},\ 0 \leq \alpha_0 \leq 1,\ L,\ q \in (0,1)$

1: **for** $k = 0 : K_{iteration}$ **do**
2: $\quad x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$
3: $\quad$ Compute $\alpha_{k+1} \in (0,1)$ from $\alpha_{k+1}^2 + (\alpha_k^2 - q)\alpha_{k+1} - \alpha_k^2 = 0$
4: $\quad \theta_k = \frac{\alpha_k(1-\alpha_k)}{\alpha_k^2+\alpha_{k+1}}$
5: $\quad y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k)$
6: **end for**

In the scheme above, the sequence y_k is the momentum term and in some cases, $q = 0$ is a common setting. Since the updating of α_{k+1} and θ_k is not easy to compute, a suboptimal updating scheme $\theta = \frac{a-1}{a+2}$, $a = 1, 2, ..., K$ is often used in the fast gradient method, which grantees the same upper bound of convergence rate.

Algorithm 2 Simplified Fast Gradient Method with Constant Step Size

Require: Initial inputs: $x_0 = y_0 \in \mathbb{R}$, L

1: **for** $k = 0 : K_{iteration}$ **do**
2: $\quad x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$
3: $\quad \theta_k = \frac{a-1}{a+2}$
4: $\quad y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k)$
5: $\quad a = a + 1$
6: **end for**

The Figure 2.3 presents the convergence rates of two gradient methods. It is obvious that a great improvement has been achieved by using the fast gradient method compared with the standard one.

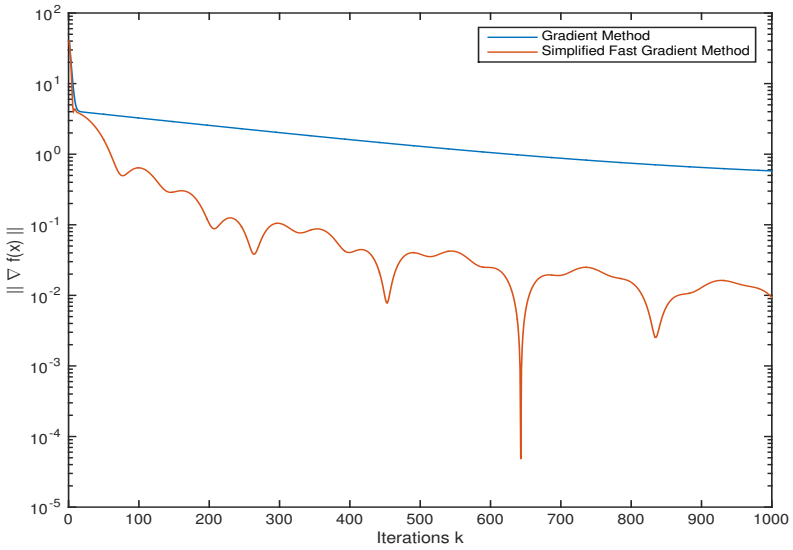


Figure 2.3: Comparison between the standard and fast gradient method

The convergence rates of the above schemes for strongly convex and convex cases are:

$$f(x_k) - f^* \leq min\left\{L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k, \frac{4L}{(k+2)^2}\right\}||x_0 - x^*||^2 \tag{2.18}$$

2.2.2 Fast Gradient Method with Adaptive Restart

As the result shows in Figure 2.3, the value of the objective function is not monotone in the fast gradient method and there are some ripples exist in the trace. Some efficient methods to remove the ripples are presented in [15]. There are two adaptive schemes introduced here:

- **Function scheme:** restart whenever $f(x_{k+1}) > f(x_k)$.
- **Gradient scheme:** restart whenever $\nabla f(y_k)^T(x_{k+1} - x_k) > 0$.

Since the value of objective function in fast gradient methods is not guaranteed to be monotone, the objective value needs to be checked like the above two schemes in order to restart the algorithm. Here restart means setting the current iteration as the initial inputs, clean the previous iterations and start the algorithm again. For the fast gradient method with second scheme, all quantities have already been calculated which avoids the unnecessary computations. Hence, the fast gradient method with second scheme is implemented in this thesis.

Algorithm 3 Fast Gradient Method with Adaptive Restart

Require: Initial inputs: $x_0 = y_0 \in \mathbb{R},\ L,\ a = 1$

for $k = 0 : K_{iteration}$ **do**
  $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$
  $\theta_k = \frac{a-1}{a+2}$
  $y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k)$
  if $\nabla f(y_k)^T(x_{k+1} - x_k) > 0$ **then**
    $y_{k+1} = x_{k+1}$
    $a = 1$
  end if
  $a = a + 1$
end for

A great improvement has been achieved by restarting the update scheme θ_k as Figure 2.4 shows.

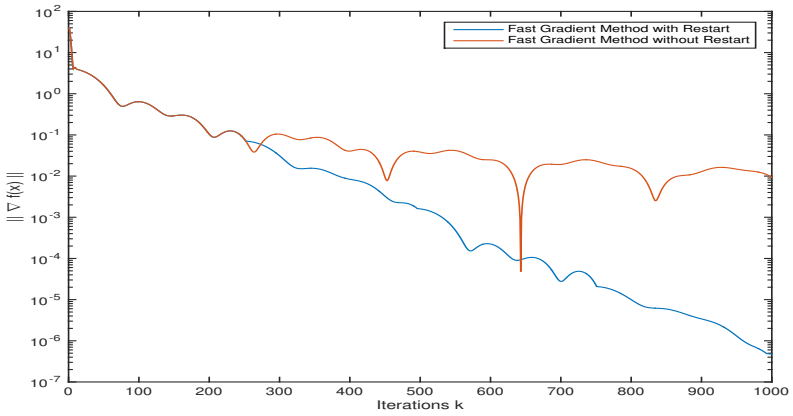


Figure 2.4: Convergence of fast gradient method with and without restart

From Figure 2.4, adaptive restart helps to avoid the unnecessary ripples so that it can reach a higher accurate result with fewer iterations. Nevertheless, it brings some additional cost when it comes to compute the condition of the restart. Thus, the number of iterations is not the only criterion to judge the efficiency of the algorithm when the adaptive restart is applied. The convergence rate of fast gradient method with adaptive start is linear for strongly convex case and the restart scheme does not affect the convergence rate [15].

2.2.3 Fast Gradient Method with Adaptive Step Size

According to the above algorithms, a large Lipschitz constant may lead to conservative steps, while a small value may result in the divergence of the algorithm. However, computing an optimal Lipschitz constant sometimes is intractable, thus an adaptive method is used to find a proper approximation of the Lipschitz constant [16].

Algorithm 4 Fast Gradient Method with Adaptive Lipschitz Approximation

Require: Initial inputs: $x_0 = y_0 \in \mathbb{R},\ L_0 > 0,\ \rho > 1$

for $k = 0 : K_{iteration}$ **do**
  while True **do**
    $x = y_k - \frac{1}{L_k}\nabla f(y_k)$
    if $|(y_k - x)^T(\nabla f(y_k) - \nabla f(x))| \leq \frac{1}{2}L_k\|x - y_k\|_2^2$ **then**
      break
    else
      $L_k = \rho L_k$
    end if
  end while
  $x_{k+1} = x$
  $\theta_k = \frac{k-1}{k+2}$
  $y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k)$
end for

In Algorithm 4, the Lipschitz constant is searched iteratively by an exponential method, the information about the Lipschitz constant is not required [17]. There are a lot of ways to search for an optimal Lipschitz constant of the function. A small initial guess and a conservative step can help to find an accurate Lipschitz constant while it requires a large amount of computation. Therefore, an appropriate method is needed in real applications according to different problems.

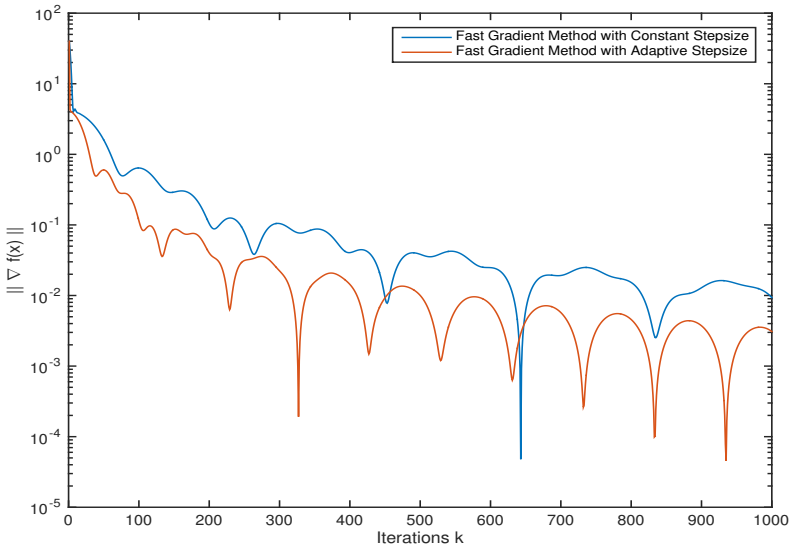


Figure 2.5: Convergence of fast gradient method with constant and adaptive step size

As shown in Figure 2.5, fewer iterations are required for adaptive Lipschitz constant scheme. An obvious improvement can be seen in Figure 2.6 if the adaptive restart is introduced in Algorithm 5.

Algorithm 5 Adaptive Restart Version of Algorithm 4

Require: Initial inputs: $x_0 = y_0 \in \mathbb{R},\ L_0 > 0,\ \rho > 1,\ a = 1$

for $k = 0 : K_{iteration}$ **do**
  while True **do**
    $x = y_k - \frac{1}{L_k}\nabla f(y_k)$
    if $|(y_k - x)^T(\nabla f(y_k) - \nabla f(x))| \leq \frac{1}{2}L_k\|x - y_k\|_2^2$ **then**
      break
    else
      $L_k = \rho L_k$
    end if
  end while
  $x_{k+1} = x$
  $\theta_k = \frac{a-1}{a+2}$
  $y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k)$
  if $\nabla f(y_k)^T(x_{k+1} - x_k) > 0$ **then**
    $y_{k+1} = x_{k+1}$
    $a = 1$
  end if
  $a = a + 1$
end for

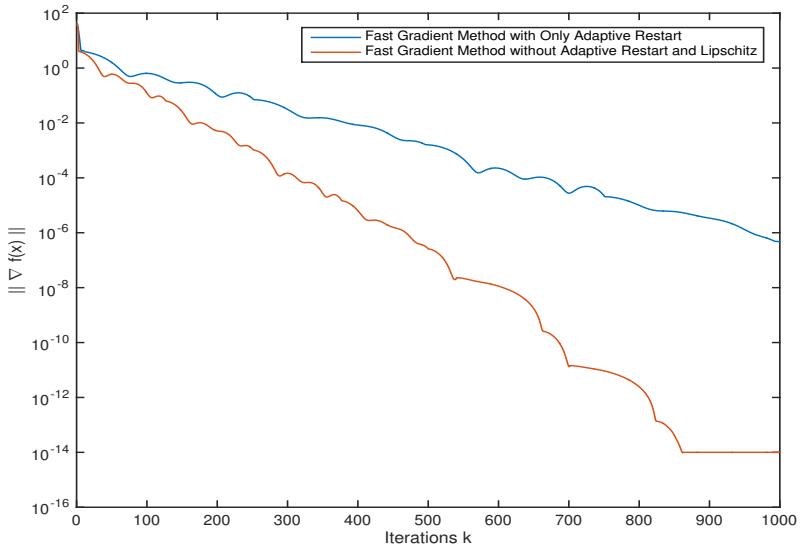


Figure 2.6: Convergence of adaptive restart with and without adaptive Lipschitz

After introducing the adaptive searching for the Lipschitz constant, the execution time for each iteration becomes longer. In addition to that, when a large Lipschitz constant is found, it is kept for the following computations, which may slow down the convergence rate of the algorithm. Hence, L_k can be set to be the initial value after an optimal Lipschitz constant is found during one iterative computation. Then the Lipschitz constant can always be guaranteed to be optimal for each iteration.

Algorithm 6 Lipschitz Resetting Version of Algorithm 4

Require: Initial inputs: $x_0 = y_0 \in \mathbb{R},\ L_0 > 0,\ \rho > 1,\ a = 1$

1: **for** $k = 0 : K_{iteration}$ **do**
2: **while** True **do**
3: $x = y_k - \frac{1}{L_k}\nabla f(y_k)$
4: **if** $|(y_k - x)^T(\nabla f(y_k) - \nabla f(x))| \leq \frac{1}{2}L_k\|x - y_k\|_2^2$ **then**
5: break
6: **else**
7: $L_k = \rho L_k$
8: **end if**
9: **end while**
10: $L_k = L_0$
11: $x_{k+1} = x$
12: $\theta_k = \frac{a-1}{a+2}$
13: $y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k)$
14: **end for**

In Algorithm 6, Line 10 is added to reset the Lipschitz constant.

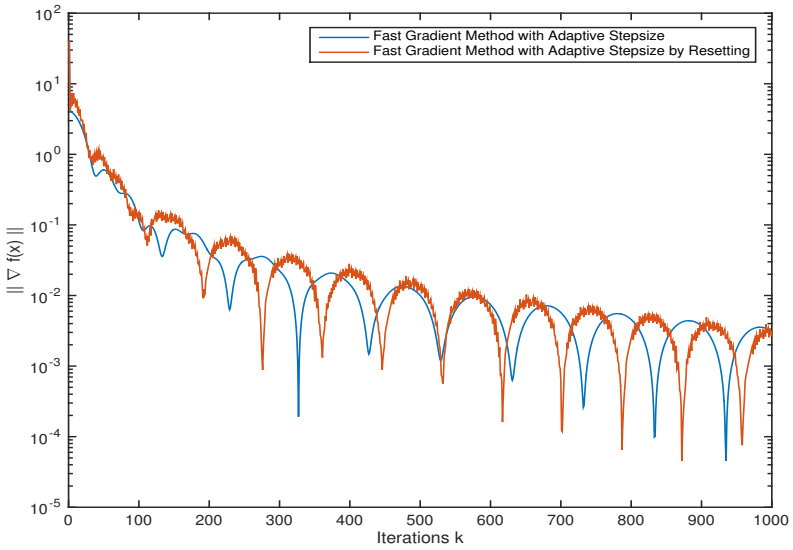


Figure 2.7: Convergence of adaptive step size method with and without resetting

Figure 2.7 illustrates the difference between the method with and without resetting. The line 7 in Algorithm 6 represents the way to find the optimal step size can be replaced by other methods:

- linear function: $L_k = L_k + \rho,\ \rho > 0$
- exponential function: $L_k = (L_k)^\rho,\ L_0 > 1,\ \rho > 1$
- logarithmic function: $L_k = log_a(L_k), a > 0$ and $a \neq 1,\ L_0 > 0$

It is not clear that how these methods affect the performance of the algorithm. For different problems, some efforts should be made to choose an appropriate method and optimize the corresponding parameters for a better performance. In this thesis, it is not difficult to compute the optimal Lipschitz constant and the method with an adaptive step size will not be discussed more in chapter 4.

Chapter 3

Constrained Linear-Quadratic MPC Framework

The main idea of Model Predictive Control (MPC) is to obtain the optimal input based on the current available information of the state and the prediction of the state trajectories over a finite time horizon in terms of the system model. Specifically, the optimal input sequence is derived by solving an optimization problem with respect to an objective function, and after the first element of the input sequence is applied to the plant, the whole process is repeated at next sampling instant when the new state information is available. The main reason for the success application of MPC is its systematic handling of state and/or input constraints. A brief illustration of MPC is shown in Figure 3.1.

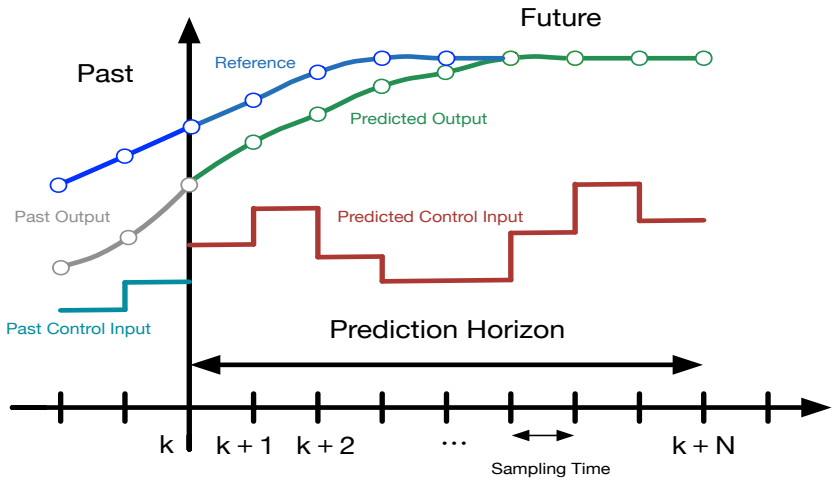


Figure 3.1: Geometric illustration of MPC

The convex optimization problems, which means the objective function to be minimized and the feasible region of the solutions are convex (Definition A.1.1, Definition A.2.1), is backed up by a mature theory of numerical analysis methods for solving them. The sub-class Quadratic Programming (QP) problem is closely related to the

MPC formulation and efficient numerical methods can be applied to QP problems to get the optimal solution based on the constraints without much complicated linear algebra handling (fast gradient method). An illustration of QP is shown in Figure 3.2. The level sets of QP problem are ellipsoids in space and the constraints are usually formed as a polyhedron (Definition A.1.6, Definition A.1.7).

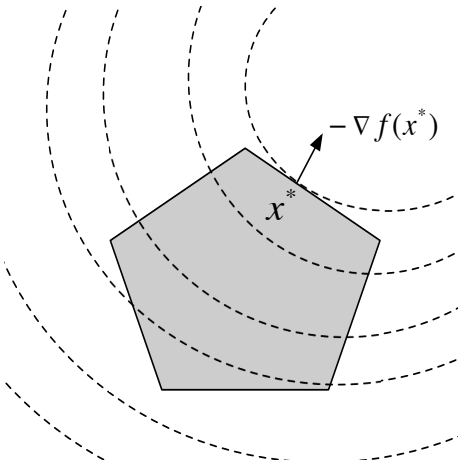


Figure 3.2: Geometric illustration of QP

In this chapter, Section 3.1 provides a brief introduction to the general constrained linear-quadratic MPC formulation which is the basis for the application of the optimization method in Chapter 4. Section 3.2 introduces the standard form of QP problems and the reformulation from MPC, a specific ordering of variables is given which results in a banded matrix for easier coding. The concept of Lagrangian relaxation is presented in section 3.2 which used for decoupling the original problem into a two-level optimization problem [18].

3.1 Constrained Linear-Quadratic MPC

A constrained linear-quadratic MPC framework consists of a discrete time state space model, inequality constraints (bounded sets for states and inputs) and a quadratic objective function with finite time horizon, which is shown in the following format:

$$\begin{aligned} \min_{x,u} \quad & x_N^T P_f x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0 \dots N-1 \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad k = 0 \dots N-1 \\ & x_0 \text{ is given}, \quad x_N \in \mathcal{X}_f \end{aligned} \tag{3.1}$$

In this thesis, the state $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and the input $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ are restricted to the closed convex sets $\mathcal{U}$ and $\mathcal{X}$ with the origins in their interiors, which results in the input and state constrained MPC. The prediction horizon is N and the terminal

state is confined to the closed convex terminal set $\mathcal{X}_f \in \mathbb{R}^{n_x}$ which contains origin. The performance of this constrained MPC regulation problem can be tuned by changing the weighting matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$, where Q, R and P_f are positive definite. It has been proved that if the terminal weighting matrix P_f and the terminal constraint set $\mathcal{X}_f$ are chosen appropriately, the nominal stability of the closed loop is guaranteed, and under the condition of the stability system, an infinity horizon constrained MPC can be transformed into a finite horizon MPC by solving an equation $A^T P_f A = P_f - Q$ which called matrix Lyapunov equation [19]. A terminal weighting matrix can be obtained from that equation to guarantee the closed-loop stability. More details about the control and stability related topics are referred to the literature [19, 20].

3.2 Quadratic Programming

For the input and state constrained linear-quadratic MPC problem, the framework based on which the optimization method is applied to in the following sections is a convex quadratic program in the following format:

$$\begin{aligned} \min \quad & f(x) = \frac{1}{2} x^T P x + q^T x + r \\ \text{s.t.} \quad & Gx = h \\ & Ex \leq s \end{aligned} \tag{3.2}$$

where $P \in \mathbb{S}_+^n$, $E \in \mathbb{R}^{m \times n}$ and $G \in \mathbb{R}^{l \times n}$. The objective function is convex and the constraint functions are affine (sum of a linear function and a constant).

In fact, in this thesis, the inequality constraints are simplified to box bounds (Definition A.3.1) which can be handled easily. Generally, the constant part r is neglected so that a compact formulation is obtained:

$$\begin{aligned} \min \quad & f(x) = \frac{1}{2} x^T P x + q^T x \\ \text{s.t.} \quad & Gx = h \\ & x \in \mathcal{X} \end{aligned} \tag{3.3}$$

3.2.1 Reformulation

By integrating the state and the input sequence from (3.1) together, a new optimization variable z is available and the QP problem can be obtained:

$$\begin{aligned} \min \quad & f(z) = \frac{1}{2} z^T H z + g^T z \\ \text{s.t.} \quad & Cz = b \\ & z \in \mathcal{Z} \end{aligned} \tag{3.4}$$

It is obvious that the way of constructing z affects the forms of the parameters H, g, C and d. In this thesis, z is chosen in a specific ordering as following:

$$z = \begin{bmatrix} x_0 & u_0 & \cdots & x_{N-1} & u_{N-1} & x_N \end{bmatrix}^T \tag{3.5}$$

As long as z is fixed, the parametric matrices are determined correspondingly. And in this formulation, C is symmetric and b is always equal to 0.

3.2.2 Lagrangian Relaxation

Lagrangian relaxation is an efficient way to simplify the difficult constrained optimization problem by providing an approximate solution to the original problem. For a QP problem, the Lagrangian is defined by dualizing the equality constraints:

$$L(x, \lambda) = \frac{1}{2}x^T Hx + g^T x + \lambda^T (Cx - b) \tag{3.6}$$

with Lagrange multiplier $\lambda \in \mathbb{R}^{N \times n}$. Then the Lagrange dual function $d(\lambda)$ is formed as:

$$d(\lambda) = \min_{x \in \mathcal{X}} \frac{1}{2}x^T Hx + g^T x + \lambda^T (Cx - b) \tag{3.7}$$

The Lagrange dual function $d(\lambda)$ gives lower bounds for the optimal value p^* of primal problem (3.3):

$$d(\lambda) \leq p^* \tag{3.8}$$

This can be proved easily from [21]. Suppose a feasible point x_f for the problem (3.3), then:

$$L(x_f, \lambda) = f(x_f) + \lambda^T (Cx_f - b) = f(x_f) \tag{3.9}$$

therefore, the following inequality is fulfilled:

$$d(\lambda) = \min_{x \in \mathcal{X}} f(x) + \lambda^T (Cx - b) \leq L(x_f, \lambda) = f(x_f) \tag{3.10}$$

Since this holds for all the feasible point includes the optimal one x^*, which results in (3.8).

This dual function has been proved to be an unconstrained concave function [22]. The optimal λ is obtained by solving the following optimization problem which is called the Lagrange dual problem and it helps to find the best lower bound of the optimal value of the primal problem:

$$\max_{\lambda} d(\lambda) \tag{3.11}$$

The difference $p^* - d^*$ is called the optimal duality gap, and if the strongly duality holds, the optimal duality gap is 0 and the optimal solution of the primal problem can be recovered from $x^*(\lambda)$ [23].

$$x^*(\lambda) = \arg\min_{x \in \mathcal{X}} \frac{1}{2}x^T Hx + g^T x + \lambda^T (Cx - b) \tag{3.12}$$

In conclusion, the original QP problem is transformed into a two-level problem:

$$\max_{\lambda \in \mathbb{R}^{N \times n}} \min_{x \in \mathcal{X}} \frac{1}{2}x^T Hx + g^T x + \lambda^T (Cx - b) \tag{3.13}$$

The inner problem (Lagrange dual function) is a QP with inequality constraints and the outer problem (Lagrange dual problem) is a dual concave (Definition A.2.2) optimization problem without constraints.

Chapter 4

Implementation

In this chapter, a state-space model is built based on the dynamics of vehicles. The structured linear-quadratic MPC formulation is utilized to create a QP problem by implementing the techniques introduced in the previous chapter and the fast gradient method is used for solving the problem. In addition, the main body of the algorithm is programmed in C to get a better performance in real applications.

4.1 Vehicle Dynamics

The longitudinal motion control problem is the focus of this project in lane change maneuvers and a simplified model of the longitudinal motion can be expressed as:

$$\begin{aligned} x_{k+1} &= x_k + v_k t_s + a_k \frac{t_s^2}{2} + \Delta a_k \frac{t_s^3}{6} && k = 0, \ldots, N \\ v_{k+1} &= v_k + a_k t_s + \Delta a_k \frac{t_s^2}{2} && k = 0, \ldots, N \\ a_{k+1} &= a_k + \Delta a_k t_s && k = 0, \ldots, N \end{aligned} \tag{4.1}$$

where x_k, v_k and a_k represent the vehicle's longitudinal position, velocity and acceleration respectively during the prediction horizon N, t_s is the sampling period and $\Delta a_k = a_{k+1} - a_k$. Meanwhile, the variables are restricted by the maximum and minimum values:

$$\begin{aligned} x_{min} \leq x_k \leq x_{max} && k = 0, \ldots, N \\ v_{min} \leq v_k \leq x_{max} && k = 0, \ldots, N \\ a_{min} \leq a_k \leq a_{max} && k = 0, \ldots, N \\ \Delta a_{min} \leq \Delta a_k \leq \Delta a_{max} && k = 0, \ldots, N-1 \end{aligned} \tag{4.2}$$

4.2 Problem Formulation

According to the longitudinal motion model, a state vector which includes all three variables and the input is introduced here:

$$\begin{aligned} w_k &= [\ x_k,\ v_k,\ a_k\ ]^T \\ u_k &= \Delta a_k \end{aligned} \tag{4.3}$$

For a typical reference tracking problem, the references for the variables are needed to make sure that the variables following the desired trajectories. In the case of longitudinal motion control, they are the trajectories of position, velocity and acceleration, i.e. $w_{k(ref)} = [\ x_{k(ref)},\ v_{k(ref)},\ a_{k(ref)}\ ]^T$ and $u_{k(ref)} = \Delta a_{k(ref)}$, then here comes the optimization problem:

$$\begin{aligned}
\min_{w,u} \quad & \sum_{k=0}^{N-1}((w_k - w_{k(ref)})^T Q(w_k - w_{k(ref)}) + (u_k - u_{k(ref)})^T R(u_k - u_{k(ref)}))+ \\
& (w_N - w_{N(ref)})^T Q_f(w_N - w_{N(ref)}) \\
\text{s.t.} \quad & w_{k+1} = Aw_k + Bu_k \\
& w_k \in \mathcal{W}, \quad u_k \in \mathcal{U} \\
& w_0 \text{ is given}
\end{aligned} \tag{4.4}$$

where $\mathcal{W}$ and $\mathcal{U}$ denote constraint sets for the state vector w_k and input u_k, and Q, R, P_f are corresponding weighting matrices. The state matrix A and input matrix B can be derived readily from (4.2):

$$A = \begin{bmatrix} 1 & t_s & \frac{t_s^2}{2} \\ 0 & 1 & t_s \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{t_s^3}{6} \\ \frac{t_s^2}{2} \\ t_s \end{bmatrix} \tag{4.5}$$

In order to apply the optimization methods to constrained MPC problem (4.5), it is framed as a standard QP by grouping the state and input sequence together.

$$\begin{aligned}
\min \quad & f(z) = \frac{1}{2} z^T H z + g^T z \\
\text{s.t.} \quad & Cz = b \\
& z \in \mathcal{Z}
\end{aligned} \tag{4.6}$$

where $z = [\ w_0\ u_0\ \cdots\ w_{N-1}\ \ u_{N-1}\ w_N\ ]^T$.

The Hessian matrix $H \subseteq \mathbb{R}^{(4N+3)\times(4N+3)}$ and a new constraint matrix $C \subseteq \mathbb{R}^{(4N+3)\times 3N}$ are reformulated accordingly. With the specific ordering of optimization variable z, a banded matrix C is available, which is easier for coding in C programming language when it comes to the calculations related to C matrix (the gradient of the dual function), because the coupling of the variables are quite close to each other which makes it easier to deal with the elements of matrix.

$$H = \begin{bmatrix} 2Q & & & & & & \\ & 2R & & & & & \\ & & \ddots & & & & \\ & & & 2Q & & & \\ & & & & 2R & & \\ & & & & & 2Q_f \end{bmatrix},\ C = \begin{bmatrix} A & B & -I & & & & \\ & & A & B & -I & & \\ & & & & \ddots & & \\ & & & & & A & B & -I \end{bmatrix} \tag{4.7}$$

With that specific choice of z, $d \subseteq \mathbb{R}^{4N+3}$ is a column vector where all elements are zeros. At last, $g^T \subseteq \mathbb{R}^{4N+3}$ is expressed in the following form:

$$g^T = -2 * [\ w_{0(ref)}^T Q\ u_{0(ref)}^T R\ \cdots\ w_{N-1(ref)}^T Q\ u_{N-1(ref)}^T R\ w_{N(ref)}^T Q_f\ ] \tag{4.8}$$

4.3 Dual Function Framework

After the problem formulation, the Lagrange dual problem is formed as mentioned in (3.13).

$$\max_{\lambda \in \mathbb{R}^{3N}} \min_{z \in \mathcal{Z}} \frac{1}{2} z^T H z + g^T z + \lambda^T (Cz - d) \tag{4.9}$$

In order to solve the dual problem (4.9), the fast gradient method is implemented. The inner QP is considered as a function with respect to z for a given λ (maybe a reasonable initial guess). Since the Hessian matrix (second-order derivatives of function) of Lagrangian and other parameters are available, the inner QP can be solved exactly in one-step computation by projecting the exact solution on the box bounds:

$$z^*(\lambda) = P_{\mathcal{Z}}(-H^{-1}(g + C^T \lambda)) \tag{4.10}$$

where $P_{\mathcal{Z}}$ means projection on a convex set.

Here the primal problem is assumed to be strongly convex which means the Hessian matrix is positive definite, because for convex function there is a possibility that the Lagrangian is not lower bounded which means the Lagrange dual problem is not solvable. For strongly convex function, it is lower bounded by a quadratic term regarding convexity parameter. In this thesis, strongly convex means the H matrix (4.7) is positive definite which means positive definite weighting matrices. The weighting matrices are assumed to be diagonal for simplicity reasons. Substituting $z^*(\lambda)$ into the Lagrangian, the Lagrange dual problem with respect to λ needs to be solved:

$$\max_{\lambda \in \mathbb{R}^{3N}} d(\lambda) \tag{4.11}$$

where $d(\lambda)$ has been proved to be a concave, piece-wise quadratic and once continuously differentiable function without any constraints [24, 25]. However, due to the box bounds it is difficult to get the solution by deriving the explicit expression of Lagrange dual function. The gradient of Lagrange dual function $d(\lambda)$, however, is readily obtained [22]:

$$\nabla d(\lambda) = Cz^*(\lambda) - b \tag{4.12}$$

The fast gradient method is now ready to be applied to solve the Lagrange dual problem and the optimal Lipschitz constant is easy to calculate (introduced in the later section). The fast gradient method for solving the Lagrange dual problem is designed as following:

Algorithm 7 Fast Gradient Method for the Dual Problem

Require: Initial inputs: $\lambda_0 = \lambda_0^{(est)},\ L,\ a = 1$

for $k = 1 : K_{iteration}$ **do**
  $z^*(\lambda_k^{(est)}) = P_{\mathcal{Z}}(-H^{-1}(g + C^T\lambda_k^{(est)}))$
  $\nabla d(\lambda_k^{(est)}) = Cz^*(\lambda_k^{(est)}) - b$
  $\lambda_{k+1} = \lambda_k^{(est)} + \frac{1}{L}\nabla d(\lambda_k^{(est)})$
  $\theta_k = \frac{a-1}{a+2}$
  $\lambda_{k+1}^{(est)} = \lambda_{k+1} + \theta_k(\lambda_{k+1} - \lambda_k)$
  if $-\nabla d(\lambda_k^{(est)})^T(\lambda_{k+1} - \lambda_k) > 0$ **then**
    $\lambda_{k+1}^{(est)} = \lambda_{k+1}$
    $a = 1$
  end if
  $a = a + 1$
end for

It is obvious that an effective way to improve the performance of the algorithm is to set a proper initial λ_0 and λ_0^{est}. Before that, some important parameters will be given first.

4.3.1 Computation of the Lipschitz Constant

The dual function $d(\lambda)$ has a Lipschitz continuous gradient under the condition of positive definite Hessian matrix [26]:

$$||\nabla d(\lambda_1) - \nabla d(\lambda_2)|| \leq L||\lambda_1 - \lambda_2|| \tag{4.13}$$

with the Lipschitz constant

$$L = \frac{\|C\|^2}{\lambda_{min}(H)} \tag{4.14}$$

where $\lambda_{min}(H)$ is the minimal eigenvalue of H and $\|C\|$ represents the maximum singular value of C.

In order to optimize this Lipschitz constant, a linear transformation of variables with an invertible matrix P is introduced, i.e. $z = Py$. In addition, P is chosen as a diagonal matrix so that the inequality constraints are not destroyed. The gradient of the dual function is not changed $\nabla d(\lambda) = CPy^*(\lambda) - b = Cz^*(\lambda) - b$, then a new Lipschitz follows [26]:

$$L = \frac{\|CP\|^2}{\lambda_{min}(P^THP)} \neq \frac{\|C\|^2}{\lambda_{min}(H)} \tag{4.15}$$

By minimizing the left hand side of (4.15) with respect to P, an optimal Lipschitz constant L^* can be obtained.

$$L^* = \min_P \frac{\|CP\|^2}{\lambda_{min}(P^THP)} \tag{4.16}$$

This optimization problem can be considered as a convex semi-definite program [27] or solved based on the following result:

$$\|C\|^2 = \min_{P \ invertible} \frac{\|CP\|^2}{\lambda_{min}(P^T H P)} \tag{4.17}$$

The proof can be found in [28]. Hence in this case, the easiest way is to set $P = H^{-\frac{1}{2}}$, then an optimal Lipschitz constant is given by

$$L^* = \|CH^{-\frac{1}{2}}\|^2 \tag{4.18}$$

4.3.2 The Smallest Lower Iteration Bound

The lower iteration bound based on the optimal Lipschitz constant is investigated in this subsection. Instead of getting the exact optimal solution of the Lagrange dual problem which is always intractable, a suboptimal solution is enough

$$d(\lambda_k) - d^*(\lambda^*) \leq \epsilon_d \tag{4.19}$$

In order to get the smallest lower iteration bound for a specific accuracy ϵ_d, the upper bound of the convergence rate of the fast gradient method is needed.

$$d(\lambda_k) - d^* \leq min\left\{L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k, \frac{4L}{(k+2)^2}\right\}||\lambda_0 - \lambda^*||^2 \leq \epsilon_d \tag{4.20}$$

After L^* is applied in Algorithm 7, a sufficient condition for the fast gradient method to reach the accuracy ϵ_d is to terminate after

$$\left\lceil 2\sqrt{\frac{L^*\Delta_d^2}{\epsilon_d}} - 2 \right\rceil \tag{4.21}$$

iterations, and it can be seen that the lower iteration bounds for fast gradient method is almost the square root of that for gradient method with the upper bound of convergence rate (2.9) by following the same procedure. In (4.21), $\Delta_d^2 = \sup h^*$ and

$$h^* = \min_{\lambda \in \wedge^*} \|\lambda - \lambda_0\|^2 \tag{4.22}$$

where $\wedge^*$ is the set of optimal λ.

According to the equation (4.21), the smallest lower iteration bound is determined by Δ_d^2, ϵ_d and L^*. ϵ_d and L^* are constant values as long as the optimization problem framework is settled. Furthermore, Δ_d^2 is the worst case minimal squared distance between an initial value and a Lagrange multiplier in the multi-parametric case [11]. The value of Δ_d^2 is determined by λ_0 and $\wedge^*$ while getting $\wedge^*$ is difficult. Therefore, $\wedge^*$ will not be investigated too much here. However, (4.21) can be optimized if a proper λ_0 is chosen.

4.3.3 Stopping Criteria

Since it is difficult to get an exact solution which fulfills every constraints (especially for the equality constraints), a stopping criterion is always required to evaluate the result when the fast gradient method is applied. In other words, a criterion is needed to check whether the result for Algorithm 7 after K iterations is good enough or converges to the optimal solution. Moreover, as long as the criterion is set, the for loop in Algorithm 7 can be replaced by a while loop algorithm within which the condition is the stopping criterion. In this thesis, the gradient of the dual function $\|\nabla d(\lambda)\|$ is selected as the stopping criterion. In the view of the QPs, this criterion can be seen as the equality constraints. If the condition $\|\nabla d(\lambda)\| = 0$ is fully fulfilled, then final solution $z(\lambda)$ recovered from λ can be considered as the optimal solution. On the other hand, from the perspective of MPC, the criterion includes the vehicle dynamics model as its elements. The gradient $\nabla d(\lambda)$ is expanded as

$$\nabla d(\lambda) = \begin{bmatrix} w_1 - \bar{w}_1 \\ w_2 - \bar{w}_2 \\ \dots \\ w_{N-1} - \bar{w}_{N-1} \\ w_N - \bar{w}_N \end{bmatrix} \tag{4.23}$$

where these two sequences $\{w_k\}_{k=1\dots N}$ and $\{\bar{w}_k\}_{k=1\dots N}$ denote the predictive states based on MPC model and the optimal solution from the QP solver respectively.

As long as $\|\nabla d(\lambda)\|$ converges to 0, the predictive results approaching the optimal calculated values as the iteration continues, which means the dynamics of the vehicle model are matched very well. Since w_k includes three different states with different units and scales which implies that three separate accuracies can be selected respectively. In real applications, the algorithm is running in C for a high-speed performance, thus $\|\nabla d(\lambda)\|^2$ is used because of the simplicity of computation.

4.4 Analysis of Results

Before implementing the QP solver in the real time test, the design parameters in Table 4.1 are provided first. Some of these parameters are used for constructing the simple box bounds for QPs.

Table 4.1: Parameters for longitudinal motion control

$v_k \in \{10, 22\}\ [m/s]$	$t_s = 1\ [s]$
$a_k \in \{-3, 3\}\ [m/s^2]$	$N = 10$
$\Delta a_k \in \{-2, 2\}\ [m/s^2]$	

In order to get a good performance of the longitudinal motion control during the lane change maneuvers, the weighting matrices are obtained after elaborate tuning

tests:

$$Q = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}, Q_f = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}, R = 4$$

How to generate the bounds and references can be found in [6].

4.4.1 Different Stopping Criteria

One data set from the real car tests is selected to illustrate the relation between iterations and the norm of the gradients.

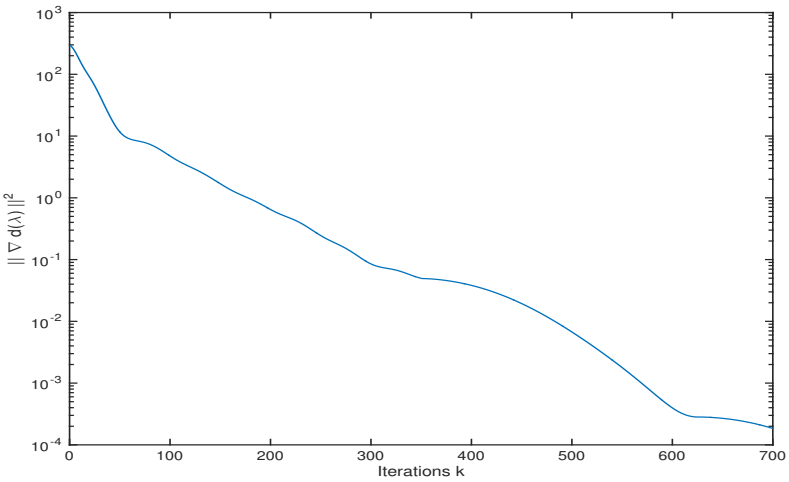


Figure 4.1: Stopping Criteria

From Figure 4.1, $\|\nabla d(\lambda)\|^2$ converges to 10^{-2} by around 500 iterations, while it is not necessary to reach such a small value for some states, especially for the distance and velocity. Hence, more information can be get when $\|\nabla d(\lambda)\|^2$ is expanded:

$$\|\nabla d(\lambda)\|^2 = \sum_{k=1}^{N} (w_k - \bar{w_k})^2 \tag{4.24}$$

$$= \sum_{k=1}^{N} (x_k - \bar{x_k})^2 + (v_k - \bar{v_k})^2 + (a_k - \bar{a_k})^2 \tag{4.25}$$

$$= \|\nabla x(\lambda)\|^2 + \|\nabla v(\lambda)\|^2 + \|\nabla a(\lambda)\|^2 \tag{4.26}$$

where $\|\nabla x(\lambda)\|^2$, $\|\nabla v(\lambda)\|^2$ and $\|\nabla a(\lambda)\|^2$ represent the stopping criteria related to the three variables: position, velocity and acceleration. The stopping criteria for them can be set respectively with the different accuracies under the framework of their own scales and units.

The following figure shows the relation between the number of iterations and these separate stopping criteria.

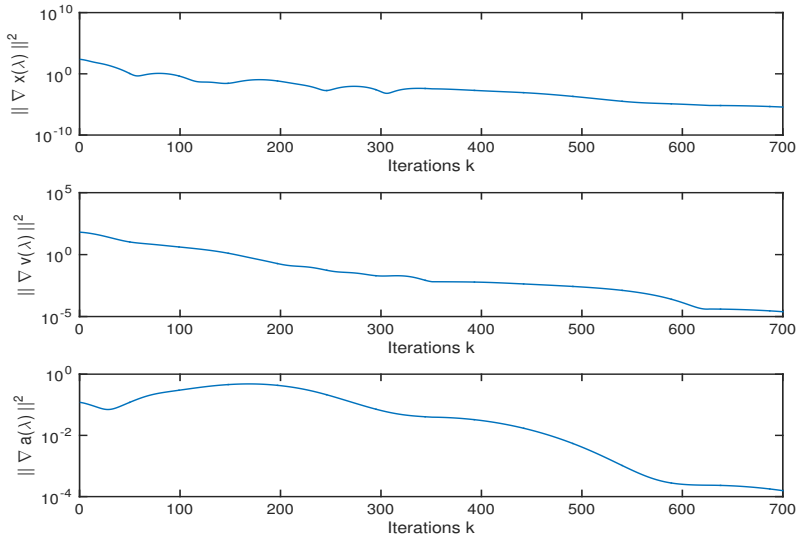


Figure 4.2: Separate Stopping Criteria

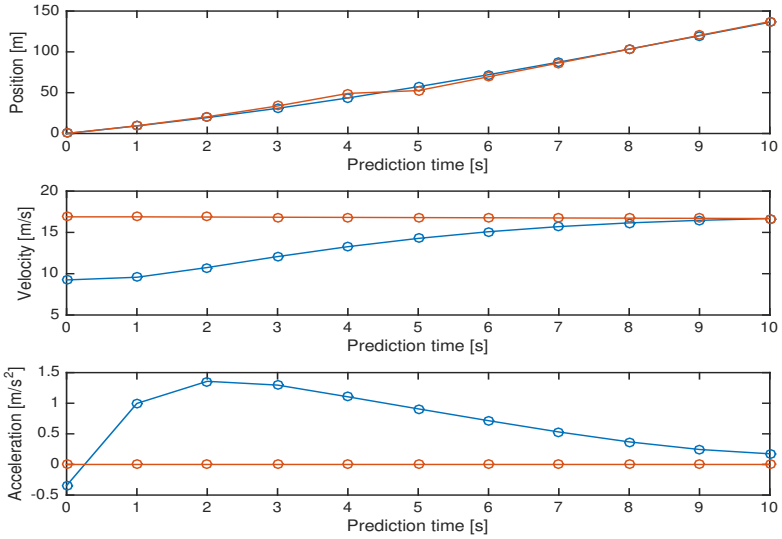


Figure 4.3: Position, velocity and acceleration of the longitudinal motion control

The tracking performance of the position, velocity and acceleration in longitudinal direction is shown in Figure 4.3. It is flexible to set three different accuracies although extracting the stopping criteria from $\|\nabla d(\lambda)\|^2$ takes some extra processes. In conclusion, separating criteria might just be an option for improving the solver.

4.4.2 Hot Start

As mentioned in the previous section, setting an appropriate initial λ is an efficient way to improve the convergence speed of the QP solver. For longitudinal motion control during lane change maneuvers, the amount of the data sets is quite large. Meanwhile, the sampling frequency of the sensors which are used to collect data is so high that the difference between two data sets in sequence is small. Therefore, an optimal solution for the previous data set is very close to the next one since they result in the similar optimization problems. So the previous optimal solution can be chosen as a reasonable initial guess for the computation of the optimal solution for the next data set. This technique is called hot start and helps to decrease the number of iterations dramatically. Some data sets from real-car tests have been tested and the results are shown in figure 4.4:

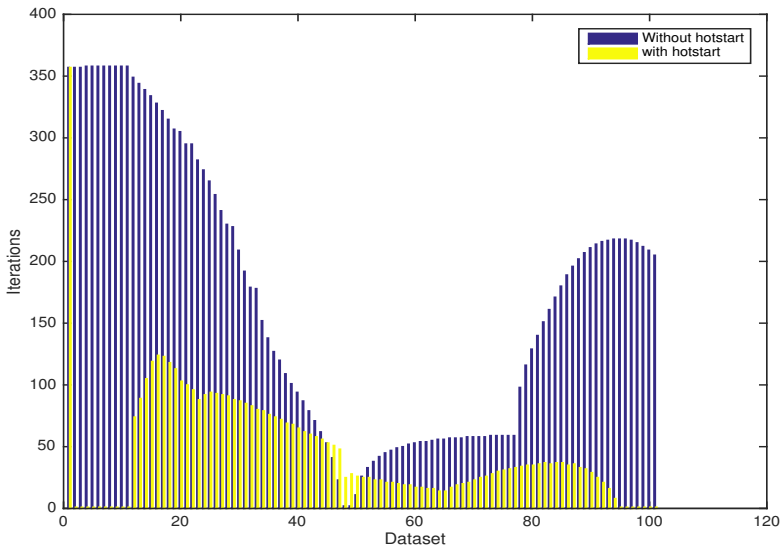


Figure 4.4: Hot start

The stopping criterion $\|\nabla d(\lambda)\|^2$ is 10^{-2}. The blue and yellow bars denote the number of iterations for method with and without hot start respectively. From Figure 4.4, a huge improvement is obtained by using the hot start technique. However, it should be noted that it can only work under the condition that the previous optimal solution is a reasonable initial start value for the next computation. Otherwise, it may lead to a worse performance, which is shown in Figure 4.5.

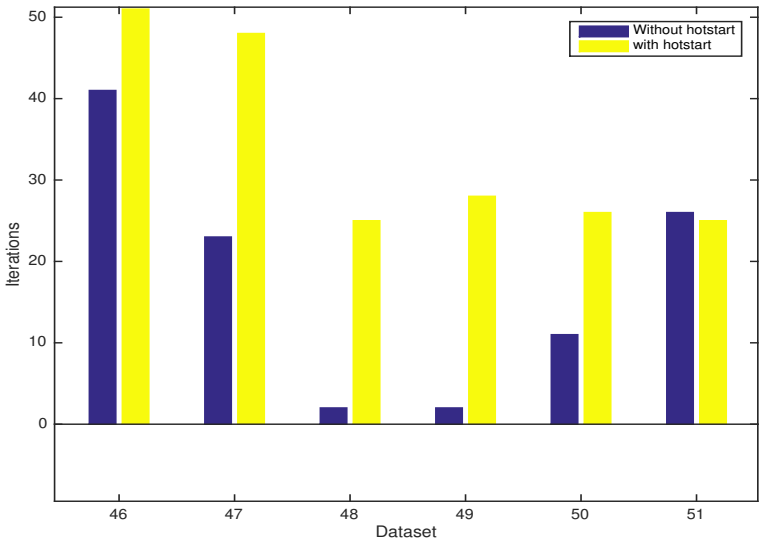


Figure 4.5: Worse performance with hot start

That is because the previous optimal solution is even further away than the original initial setting and it is impossible to figure out the timing when the hot start needs to be applied. Anyway, it can still be considered as an option for the solver.

Chapter 5

Conclusion and Discussion

5.1 Summary

In this thesis project, a fast gradient method based solver has been developed to solve the typical linear-quadratic MPC problems. Lagrangian relaxation method is introduced to dualize the original optimization problem, which results in a two-level one. The inner problem is the dual function with inequality constraints while in the outer level, it becomes an unconstrained concave optimization problem.

Since the standard gradient method only works well for a low-accuracy solution and converges slowly when a high accuracy is required, the fast gradient method is investigated to obtain a better performance. In recent years, some variants of fast gradient methods have been developed after the genius work from Nesterov, e.g. adaptive restart, adaptive Lipschitz constant and etc. It is not obvious that which one is the best and it is highly depends on the specific problem. When it comes to time-critical systems (embedded systems), in most cases, the total execution time is the most important parameter which needs to be minimized and the number of iterations is not the only thing that matters. The unit computing time for each iterative computation of the algorithm is also important.

In order to improve the performance of the QP solver, some efficient techniques have been applied. For the strongly convex objective function, a linear transformation of the optimization variables is made to minimize the Lipschitz constant (inverse of the step size). In this thesis, the Hessian matrix is assumed to be diagonal and positive definite, so it is easy to get such a transformer to fulfill the linear transformation, otherwise a convex semi-definite optimization problem has to be solved. If it is difficult or even impossible to calculate the Lipschitz constant, then an adaptive searching method can be used for finding a proper one.

On the other hand, selecting an appropriate initial value is another way to speed up the convergence. In this thesis, the hot start is applied due to the similarity between two computations in sequence. Meanwhile, it is not always able to guarantee a better performance, especially when these two computations are completely different. The stopping criterion or accuracy setting is also a key factor in the performance

of the algorithms. In this case, the gradient of the dual function is selected as the stopping criterion and can be used to evaluate the result of the QP solver. Separating criteria is also a good option in some scenarios.

Finally, developing a QP solver by C programming language helps to make the execution speed faster. In fact, the whole system is built in Simulink and the QP solver is developed in the form of S-function block. The functions in C code can be called directly in Simulink model which provides a user-friendly interface.

5.2 Future work

Some topics listed in this subsection are worth to be investigated in the future.

- Fixed point arithmetic could be an effective way to improve the performance of the solver by replacing the floating point data types. For classic gradient method, it is numerical robust with the inexact computation result from fixed point arithmetic. However, for the fast gradient method, attentions need to be paid when inexact calculations are involved [29]. On the other hand, the convergence results of the fast gradient method need to be re-investigated when it comes to the fixed point arithmetic, since they are based on the exact calculation in this thesis.

- The computation of the optimal Lipschitz constant involves the singular value decomposition and it increases the computation burden when it comes to the real-time implementation in C code. If a more compact solver, which encapsulates every details inside, is required, the adaptive searching for the optimal Lipschitz constant is a reasonable alternative.

- In this thesis, the primal objective function is assumed to be strongly convex. Some optimization methods deal with a broader class of problems can be investigated in the future if that condition is not satisfied (such as Alternating Direction Method of Multipliers (ADMM)).

Bibliography

[1] Peden, M et al. World Report on Road Traffic Injury Prevention. World Health Organization, The World Bank, Geneva, 2004.

[2] Wei, J., Dolan, J. M. and Litkouhi, B. A prediction-and cost function-based algorithm for robust autonomous freeway driving. Intelligent Vehicles Symposium (IV), 2010 IEEE. IEEE, 2010: 512-517.

[3] Wikipedia. Autonomous Car. https://en.wikipedia.org/wiki/Autonomous_car, 2015.

[4] Dokic, J., Müller, B. and Meyer, G. European Roadmap Smart Systems for Automated Driving. European Technology Platform on Smart Systems Integration (EPoSS), 2015

[5] Borrelli, F., Falcone, P., Keviczky, T., Asgari, J. and Hrovat, D. MPC-based approach to active steering for autonomous vehicle systems. International Journal of Vehicle Autonomous Systems, 2005, 3(2-4): 265-291.

[6] Nilsson, J., Brannstrom, M., Coelingh, E. and Fredriksson, J. Longitudinal and lateral control for automated lane change maneuvers. American Control Conference (ACC), 2015. IEEE, 2015: 1399-1404.

[7] Gould, N. I. and Toint, P. L. A quadratic programming bibliography. Numerical Analysis Group Internal Report, 2000, 1.

[8] Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. The explicit linear quadratic regulator for constrained systems. Automatica, 2002, 38(1): 3-20.

[9] Geyer, T., Torrisi, F. D. and Morari, M. Optimal complexity reduction of piecewise affine models based on hyperplane arrangements. American Control Conference, 2004. Proceedings of the 2004. IEEE, 2004, 2: 1190-1195.

[10] Grieder, P. and Morari, M. Complexity reduction of receding horizon control. Decision and Control, 2003. Proceedings. 42nd IEEE Conference on. IEEE, 2003, 3: 3179-3190.

[11] Richter, S. Computational complexity certification of gradient methods for real-time model predictive control. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20718, 2012.

[12] Nesterov, Y. Introductory lectures on convex optimization. Springer Science and Business Media, 2004.

[13] Bertsekas, D. P. Nonlinear programming. Athena Scientific, Belmont, MA, 1999.

[14] Nesterov, Y. A method of solving a convex programming problem with convergence rate O (1/k2). Soviet Mathematics Doklady. 1983, 27(2): 372-376.

[15] O'Donoghue, B. and Candes, E. Adaptive restart for accelerated gradient schemes. Foundations of computational mathematics, 2013, 15(3): 715-732.

[16] Jensen, T. L., Jørgensen, J. H., Hansen, P. C. and Jensen, S. H. Implementation of an optimal first-order method for strongly convex total variation regularization. BIT Numerical Mathematics, 2012, 52(2): 329-356.

[17] Becker, S. R., Candès, E. J. and Grant, M. C. Templates for convex cone problems with applications to sparse signal recovery. Mathematical Programming Computation, 2011, 3(3): 165-218.

[18] Kozma, A., Conte, C. and Diehl, M. Benchmarking large-scale distributed convex quadratic programming algorithms. Optimization Methods and Software, 2015, 30(1): 191-214.

[19] Maciejowski, J. M. Predictive control: with constraints. Pearson education, 2002.

[20] Mayne, D. Q. and Rawlings, J. B. Model predictive control: theory and design. Madison, WI: Nob Hill Publishing, LCC, 2009.

[21] Boyd, S. and Vandenberghe, L. Convex optimization. Cambridge university press, 2004.

[22] Bertsekas, D. P. and Tsitsiklis, J. N. Parallel and distributed computation: numerical methods. Prentice-Hall, Inc., 1989.

[23] Rockafellar, R. T. Convex Analysis. Princeton landmarks in mathematics. 1997.

[24] Ferreau, H. J., Kozma, A. and Diehl, M. A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC. Nonlinear Model Predictive Control. 2012, 4(1): 74-79.

[25] Frasch, J. V., Sager, S. and Diehl, M. A parallel quadratic programming method for dynamic optimization problems. Mathematical Programming Computation, 2013: 1-41.

[26] Richter, S., Jones, C. N. and Morari, M. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. Automatic Control, IEEE Transactions on, 2012, 57(6): 1391-1403.

[27] Boyd, S., El Ghaoui, L., Feron, E. and Balakrishnan, V. Linear matrix inequalities in system and control theory. Philadelphia: Society for industrial and applied mathematics, 1994.

[28] Richter, S., Jones, C. N. and Morari, M. Certification aspects of the fast gradient method for solving the dual of parametric convex programs. Mathematical Methods of Operations Research, 2013, 77(3): 305-321.

[29] Devolder, O., Glineur, F., Nesterov, Y. First-order methods of smooth convex optimization with inexact oracle. Mathematical Programming, 2014, 146(1-2): 37-75.

Appendix A

Definitions From Convex Optimization

The content in the Appendix is mainly based on the concepts mentioned in [13, 21].

A.1 Convex Sets

Definition A.1.1 *If for any $x_1, x_2 \in \mathbb{X}$ and $\theta \in \mathbb{R}$, there always exists $\theta x_1 + (1-\theta)x_2 \in \mathbb{X}$, then $\mathbb{X} \subseteq \mathbb{R}^n$ is an affine set. Furthermore, the set $\mathbb{X}$ is called convex if $0 \leq \theta \leq 1$.*

Definition A.1.2 *A point x is said to be a closure point or limit point of a subset $\mathbb{X}$ of $\mathbb{R}^n$ if there exists a sequence $\{x_k\} \subset \mathbb{X}$ that converges to x. The closure of $\mathbb{X}$, denoted $\mathbf{cl}(\mathbb{X})$, is the set of all closure points of $\mathbb{X}$.*

Definition A.1.3 *(Closed and Open Sets) A subset $\mathbb{X}$ of $\mathbb{R}^n$ is called closed if it is equal to its closure. It is called open if its complement, $\{x|x \notin \mathbb{X}\}$, is closed. It is called bounded if there exists a scalar c such that $||x|| \leq c$ for all $x \in \mathbb{X}$. It is called compact if it is closed and bounded.*

Definition A.1.4 *A hyperplane is a set of the form*

$$\{x|a^T x = b\}, \tag{A.1}$$

where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$.

Definition A.1.5 *A hyperplane divides $\mathbb{R}^n$ into two halfspaces. A (closed) halfspace is a set of the form*

$$\{x|a^T x \leq b\}, \tag{A.2}$$

where $a \neq 0$.

Definition A.1.6 *A polyhedron is defined as the solution set of a finite number of linear equalities and inequalities:*

$$\mathcal{P} = \{x|a_j^T \leq b_j, j = 1, ..., m, c_j^T = d_j, j = 1, ..., p\}. \tag{A.3}$$

A polyhedron is thus the intersection of a finite number of halfspaces and hyperplanes.

Definition A.1.7 *A related family of convex sets is the ellipsoids, which have the form*

$$\varepsilon = \{x|(x - x_c)^T P(x - x_c) \leq 1\}, \tag{A.4}$$

where $P = P^T \succ 0$, i,e., P is symmetric and positive definite. The vector $x_c \in \mathbb{R}^n$ is the center of the ellipsoid. The matrix P determines how far the ellipsoid extends in every direction from x_c; the lengths of the semi-axes of ε are given by $\sqrt{\lambda_i}$, where λ_i are the eigenvalues of P.

A.2 Convex Functions

Definition A.2.1 *Let C be a convex subset of $\mathbb{R}^n$. A function $f : C \mapsto \mathbb{R}$ is called convex if*

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y), \quad \forall x, y \in C, \forall \alpha \in [0, 1]. \tag{A.5}$$

The function f is called strictly convex if the above inequality is strict for all $x, y \in C$ with $x \neq y$ and all $\alpha \in (0, 1)$. For a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, it is also said that f is convex over the convex set C if $(A.5)$ holds.

Definition A.2.2 *A continuously differentiable function $f(x)$ is called convex on $\mathbb{R}^n$ if for any $x_1, x_2 \in \mathbb{X}$ and $\mathbb{X} \subseteq \mathbb{R}^n$ is a convex set.*

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle \tag{A.6}$$

If $-f(x)$ is convex, f(x) is called concave.

Proposition A.2.1 *Let C be a convex subset of $\mathbb{R}^n$ and let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be twice continuously differentiable over $\mathbb{R}^n$.*

(a) If $\nabla^2 f(x)$ is positive semi-definite for all $x \in C$, then f is convex over C.

(b) If $\nabla^2 f(x)$ is positive definite for every $x \in C$, then f is strictly convex over C.

(c) If C is open and f is convex over C, then $\nabla^2 f(x)$ is positive semi-definite for all $x \in C$.

(d) If $f(x) = x'Qx$, where Q is a symmetric matrix, then f is convex if and only if Q is positive semi-definite. Furthermore, f is strictly convex if and only if Q is positive definite.

Definition A.2.3 *(Strongly convex) A continuously differentiable function $f(x)$ is called strongly convex on $\mathbb{R}^n$ if for any $x_1, x_2 \in \mathbb{X}$ and $\mathbb{X} \subseteq \mathbb{R}^n$ is a convex set, there exists a constant parameter called convexity $\mu > 0$ such that*

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{\mu}{2}\|x_2 - x_1\|^2 \tag{A.7}$$

Constant μ is called the convexity parameter of function f.

Proposition A.2.2 *(Strong Convexity) Let C be an open convex subset of $\mathbb{R}^n$, and let $f : C \mapsto \mathbb{R}$ be a function that is continuously differentiable over C. If f is strongly convex, then f is strictly convex. Furthermore, if f is twice continuously differentiable over C, then f satisfies (A.7) if and only if the matrix $\nabla^2 f(x) - \mu I_n$, where I_n is the identity matrix and convexity parameter $\mu > 0$, is positive semi-definite for every $x \in C$.*

Proposition A.2.3 *If C is a convex subset of $\mathbb{R}^n$ and $f : C \mapsto \mathbb{R}$ is a convex function, then a local minimum of f is also a global minimum. If in addition f is strictly convex, then there exists at most one global minimum of f.*

Definition A.2.4 *(Lipschitz Continuity of Gradient) If the function f is continuously differentiable and the gradient ∇f is Lipschitz continuous with the Lipschitz constant L on set $\mathbb{R}^n$ if*

$$f(x_2) \leq f(x_1) + \langle \nabla f(x_1), x_2 - x_1 \rangle + \frac{L}{2}\|x_2 - x_1\|^2, \quad \forall x_1, x_2 \in \mathbb{R}^n \tag{A.8}$$

The following equivalent definitions also exist for Lipschitz Continuity of gradient

$$\|\nabla f(x_2) - \nabla f(x_1)\| \leq L\|x_2 - x_1\|, \quad \forall x_1, x_2 \in \mathbb{R}^n \tag{A.9}$$

$$\nabla^2 f(x) \leq L I_n \tag{A.10}$$

A.3 Projection

Definition A.3.1 *If the infimum is always achieved, the distance of $x_0 \in \mathbb{R}^n$ to a closed set $\mathbb{X} \subseteq \mathbb{R}^n$ is defined as*

$$\textbf{\textit{Dist}}(x_0, \mathbb{X}) = inf\{\|x_0 - x\| \mid x \in \mathbb{X}\}$$

And $P_{\mathbb{X}(x)}$ is referred as projection on $\mathbb{X}$, the following expression demonstrates the meaning of $P_{\mathbb{X}}(x)$.

$$P_{\mathbb{X}}(x) = \arg\min\{\|x_0 - x\| \mid x \in \mathbb{X}\}$$

Since the inequality constraints can be considered as a box bound, the Euclidean projection of x on a box $\mathbb{X} = \{x \mid l \preceq u\}$ where $(l \preceq u)$ is given by

$$P_{\mathbb{X}}(x)_k = \begin{cases} l_k & x_k \leq l_k \\ x_k & l_k \leq x_k \leq u_k \\ u_k & x_k \geq u_k \end{cases}$$

A brief illustration of the projection on the box constraints is shown following.

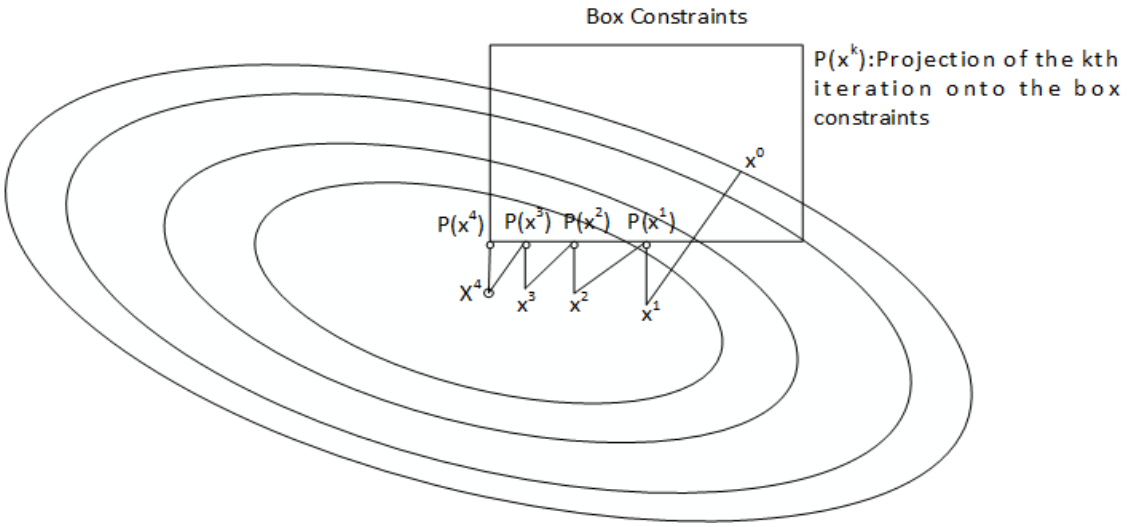


Figure A.1: Projection onto the box constraints

A.4 Rate of Convergence

Iterative optimization method generates a sequence of $\{x_k\}_{k=1}^{\infty}$ to approach the optimal solution of a function. If the function f is bounded, then the sequence converges and an optimal solution x^* exists. The rate of convergence of an optimization method is always validated in terms of the error function $e : \mathbb{R}^n \mapsto \mathbb{R}$ either in the form of Euclidean distance $e(x) = ||x - x^*||$ or the cost difference $e(x) = |f(x) - f(x^*)|$ which satisfies $e(x) \geq 0$ for all $x \in \mathbb{R}^n$ and $e(x^*) = 0$. The following list shows four typical convergence rates:

- Linear Convergence Rate: A sequence $\{e(x_k)\}_{k=1}^{\infty}$ converges linearly, if there exists $C > 0$ and $a \in (0, 1)$ such that $e(x_k) \leq Ca^k$, or with a sufficient condition $\lim_{k\to\infty} \sup \frac{e(x_{k+1})}{e(x_k)} \leq a$
- Sublinear Convergence Rate: A sequence $\{e(x_k)\}_{k=1}^{\infty}$ converges sublinearly if the convergence rate is not linear, e.g., $e(x_k) \leq \frac{C}{k+1}$ or $e(x_k) \leq \frac{C}{(k+1)^2}$ for some positive constant C, and with the sufficient condition $\lim_{k\to\infty} \sup \frac{e(x_{k+1})}{e(x_k)} = 1$
- Superlinear Convergence Rate: A sequence $\{e(x_k)\}_{k=1}^{\infty}$ converges superlinearly, if for every $a \in (0, 1)$, there exists positive constant $C > 0$ such that $e(x_k) \leq Ca^k$ holds for all k, and this is also true if $\lim_{k\to\infty} \sup \frac{e(x_{k+1})}{e(x_k)} = 0$
- Quadratic Convergence Rate: A sequence $\{e(x_k)\}_{k=1}^{\infty}$ converges at least superlinearly with order p, if for all k, there exists $C > 0, a \in (0, 1)$ and $p > 1$ such that $e(x_k) \leq Ca^{p^k}$, and if $p = 2$, the convergence rate is referred to quadratic convergence.