A Study on Optimization Algorithms in MPC

Yuki Nakagaki, Guisheng Zhai
Dept. Mathematical Sciences, Shibaura Institute of Technology, Saitama, Japan
E-mail: zhai@shibaura-it.ac.jp

Abstract. We consider the implementation problem of Model Predictive Control (MPC) especially for linear control systems. In the real time computation of MPC, we need to choose an optimization method to solve at each sampling periods. For this purpose, we suggest a new combination of the existing numerical calculation methods to speed up the computer calculation. More precisely, we first introduce the well known KKT theorem in optimization problems, which is used in the MPC calculation procedure. Secondly, we review the MPC calculation and present a linear Lagrangian algorithm for the KKT theorem, which is combined with Wolfe conditions and Dual method. Finally, in order to demonstrate the proposed approach, we provide one numerical example and another example for stabilization of inverted pendulums.

1. Introduction

It is commonly recognized that optimal control is one important tool for real control systems. Optimal control deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. A control problem includes a cost functional that is a function of state and control variables. An optimal control is a set of differential equations describing the paths of the control variables that minimize the cost function. The theory of optimal control is concerned with operating a dynamic system at minimum cost. Concerning the feature of the systems under consideration, when the system dynamics is described by a set of linear differential equations and the cost function is specified as a quadratic one, we come up with the LQ (Linear Quadratic) problem. In this case, one of the main results in the literature is that the optimal solution can be computed by solving a Riccati equation in LQR (Linear Quadratic Regulator) theory. But LQR theory can not provide the optimal solution explicitly to even linear systems with constraints. For this reason and also for the purpose of dealing with external disturbance, the approach of Model Predictive Control (MPC) has been established.

In order to deal with various constraints on state/output or input of the system, Model Predictive Control as an on-line optimal algorithm has been proposed and studied widely. Nowadays, MPC is an indispensable part of industrial control engineering and is also a essential component in advanced control applications [1]. The book by J. M. Maciejowski provides a systematic and comprehensive course on predictive control suitable for researchers and graduate students, as well as practicing engineers.

MPC is a form of control in which the current control action is obtained by solving at each sampling instant a finite horizon open-loop optimal control problem [2], by using the
current state of plant as the initial state. The optimization at each instant yields an optimal control sequence, and the first control in this sequence is applied to the plant. When the cost function in the above optimal control problem is defined as a quadratic function with respect to the state/output, Model Predictive Control can be rewritten as a quadratic programming (QP) problem. Therefore, we need to solve the QP problem at each instant, and desire a fast numerical algorithm for QP. For nonlinear systems, several fast numerical algorithms have been developed [3], but the formulation and notations are quite complicated in general cases. Since we mainly deal with linear control systems in this paper, we aim to propose a new fast computation algorithm.

Moreover, we consider linear control systems which have output or input constraints. It is known and also pointed out in [1] that iteration methods are necessary for the implementation, and in general the procedure is time consuming when we require a specified accuracy of the solutions at the iteration. In order to reduce the number of iterations, the Wolfe conditions has been established in [4, 5], where the optimal step size was adjusted during the iteration. However, the state/output or input constraints may be violated when applying the Wolfe conditions. So, we must correct the input to satisfy the constraints. For this purpose, we propose to use the Dual method [6] together with the Wolfe conditions. Since the Dual method requires an initial value of the equality constraints, it is determined by a solution with Wolfe conditions. In this paper, we assume inequality constraints on upper and lower limits, which may include equality constraints.

This paper is organized as follows: in Section 2 the nonlinear programming and algorithms are summarized. The optimization problem in MPC is presented in Section 3. A combined algorithm for QP in MPC and simulation are shown in Section 4, and the application to a real system is shown in Section 5. Some conclusions are given in Section 6.

Notation
Throughout this paper, \( f \) is a scalar function, \( g_i \) denotes a component of the constraint vector \( g = [g_1, g_2, \ldots]^T \), \( U \) is a control input vector, \( U^* \) denotes the optimal solution, and \( U^{(k)} \) represents the iteration value of \( U \) at \( k \).

2. Nonlinear Programming and Algorithms
In this section, for the benefit of our discussion later, we summarize some important theorems in nonlinear programming problems and associated algorithms.

2.1. KKT Theorem
Consider the nonlinear programming problem

\[
\begin{align*}
\text{minimize} & \quad f(U) \\
\text{subject to} & \quad g_i(U) \leq 0 \quad (i = 1, \ldots, n_c)
\end{align*}
\]  

where the functions \( f(U) \) and \( g_i(U) \)'s are all continuously differentiable with respect to the variable (vector) \( U \). For this optimization problem, we introduce a multiplier for each constraint and construct the following Lagrangian function

\[
L(U, \mu) = f(U) + \sum_{i=1}^{n_c} \mu_i g_i(U) = f(U) + \mu^T g(U)
\]
where $\mu_i$’s are the multipliers, and

$$
\mu = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_{nc} \end{bmatrix}^T \quad g(U) = \begin{bmatrix} g_1(U) & g_2(U) & \cdots & g_{nc}(U) \end{bmatrix}^T.
$$

(4)

In addition, for any given $U$, we define the active set

$$
I(U) = \{1 \leq i \leq nc : g_i(U) = 0\},
$$

(5)

and say the multiplier $\mu_i$ is active (nonactive) if $i \notin I(U)$ ($i \in I(U)$).

Then, the well known KKT theorem [7, 8] gives a necessary conditions for local optimality.

**Theorem 2.1 (Karush-Kuhn-Tucker Theorem)**

Assume that $U^*$ is a local minimum solution of the above optimization problem, and $\nabla g_i(U^*)$, $i \in I(U^*)$, are linearly independent. Then, there exists a unique set of multipliers $\mu_i$, such that

$$
\nabla_U L(U^*, \mu^*) = \nabla f(U^*) + \sum_{i=1}^{nc} \mu_i \nabla g_i(U^*) = 0
$$

(6)

$$
\nabla_{\mu} L(U^*, \mu^*) = g(U^*) \leq 0
$$

(7)

$$
\mu_i^* \geq 0 \quad (i = 1, \ldots, nc)
$$

(8)

$$
\mu_i^* = 0 \quad (i \notin I(U^*)).
$$

(9)

Furthermore, the following theorem provides a sufficient conditions for global optimality.

**Theorem 2.2 (KKT Theorem for Convex $f(U)$ and Linear Independent $g_i(U)$’s)**

Assume that $f(U)$ is convex and $g_i(U)$'s are linear functions (convex). Then, $U^*$ is a global minimum solution of the optimization problem, if and only if there exist a unique set of multipliers $\mu_i$, such that (6)-(9) are satisfied.

### 2.2. Lagrangian algorithm

Recall that for the nonlinear programming problem (1)-(2), the Lagrangian function is defined by (3). With an initial vector $\mu^{(0)} = [\mu_1 \mu_2 \ldots \mu_{nc}]^T$, the Lagrangian algorithm for solving it is given by

$$
U^{(k+1)} = U^{(k)} - \alpha^{(k)} \left( \nabla f(U^{(k)}) + \nabla g(U^{(k)})^T \mu^{(k)} \right)
$$

(10)

$$
\mu^{(k+1)} = \max(\mu^{(k)} + \beta^{(k)} g(U^{(k)}), 0), \quad k = 0, 1, \ldots
$$

(11)

where $\alpha^{(k)}, \beta^{(k)}$ are step sizes, and $\mu^{(k)}$ is the $k$-th approximation of $\mu$.

**Theorem 2.3**

In the above Lagrangian algorithm for updating $U^{(k)}$ and $\mu^{(k)}$, provided that $\alpha$ and $\beta$ are sufficiently small, there is a neighborhood of $(U^*, \mu^*)$ such that if the initial pair $(U^{(0)}, \mu^{(0)})$ is in this neighborhood, then the nonactive multipliers reduce to zero in finite time and remain at zero thereafter, and the pair $(U^{(k)}, \mu^{(k)})$ in the algorithm converges to $(U^*, \mu^*)$ with at least a linear order of convergence.

This theorem has been proved in [9].
2.3. Wolfe conditions

In computing the step size $\alpha^{(k)}$, we face a trade off. We would like to choose $\alpha^{(k)}$ to give a substantial reduction of $f$, but at the same time we do not want to spend too much time making the choice. In a line search method, we first find a descent direction $d^{(k)}$ satisfying

$$\nabla f(U^{(k)})^T d^{(k)} < 0.$$ 

Then, we consider the global minimization of the univariate function $\phi(\cdot)$ defined by

$$\phi(\alpha) = f(U^{(k)} + \alpha d^{(k)}), \quad \alpha > 0$$

with respect to the scalar variable $\alpha$. However, to find even a local minimizer of $\phi$ to moderate precision generally requires too much evaluation of the objective function $f$ and possibly the gradient $\nabla f$. Some practical strategies have been proposed to perform an inexact line search in order to determine a step size that achieves adequate reductions in $f$ at minimal cost. Here, we discuss various termination conditions for such line search algorithms, where the iterated step sizes may not lie near minimizers of the univariate function $\phi(\alpha)$ defined in (12), but the function $f$ definitely decreases.

One popular inexact line search condition stipulates that $\alpha^{(k)}$ should first of all give sufficient decrease in the objective function $f$, as measured by the following inequality:

$$f(U^{(k)} + \alpha^{(k)} d^{(k)}) - f(U^{(k)}) \leq \rho_1 \alpha^{(k)} \nabla f(U^{(k)})^T d^{(k)}$$

for some constant $\rho_1 \in (0, 1)$. In other words, the reduction in $f$ should be proportional to both the step size $\alpha^{(k)}$ and the directional derivative $\nabla f(U^{(k)})^T d^{(k)}$. Inequality (13) is sometimes called the Armijo condition [6].

The sufficient decrease condition is not enough by itself to ensure that the algorithm makes reasonable progress because it is satisfied for all sufficiently small values of $\alpha$. To rule out unacceptably short steps we introduce a second requirement, called the curvature condition, which requires $\alpha^{(k)}$ to satisfy

$$\nabla f(U^{(k)} + \alpha^{(k)} d^{(k)})^T d^{(k)} \geq \rho_2 \nabla f(U^{(k)})^T d^{(k)}$$

for some constant $\rho_2 \in (\rho_1, 1)$ where $\rho_1$ is the constant from (13).

The sufficient decrease and curvature conditions are known collectively as the Wolfe conditions. We summarize them here for discussion later

$$\begin{cases} f(U^{(k)} + \alpha^{(k)} d^{(k)}) - f(U^{(k)}) \leq \rho_1 \alpha^{(k)} \nabla f(U^{(k)})^T d^{(k)} \\ \nabla f(U^{(k)} + \alpha^{(k)} d^{(k)})^T d^{(k)} \geq \rho_2 \nabla f(U^{(k)})^T d^{(k)} \end{cases}$$

where $0 < \rho_1 < \rho_2 < 1$.

The Wolfe conditions have been used in many line search methods, and are particularly important in the implementation [6]. This calculation algorithm for Wolfe conditions (15) is described in the reference [10].
2.4. Dual method

Dual method is an approach for convex quadratic programming problem with constraints. Consider the following optimization problem with equation and inequality constraints,

$$\begin{align*}
\text{minimize} & \quad \left\{ \frac{1}{2} U^T QU + q^T U \right\} \\
\text{subject to} & \quad (a^i)^T U - b_i = 0, \quad i \in E \\
& \quad (a^j)^T U - b_j \leq 0, \quad j \in G
\end{align*}$$

where $E$ is the index set of equation constraints and $G$ is the index set of inequality constraints.

First, ignoring the inequality constraints, we consider the following problem (17).

$$\begin{align*}
\text{minimize} & \quad \left\{ \frac{1}{2} U^T QU + q^T U \right\} \\
\text{subject to} & \quad (a^j)^T U - b_j = 0, \quad j \in E
\end{align*}$$

The solutions of (17), derived from the KKT theorem, are given by

$$\begin{align*}
\hat{U} &= (A_I^+)^T b_I - H_I q \\
\hat{\lambda}_I &= -(A_I^T Q^{-1} A_I)^{-1} b_I - A_I^+ q
\end{align*}$$

where $A_I^+ := (A_I^T Q^{-1} A_I)^{-1} A_I^T Q^{-1}$, $H_I := Q^{-1} (I - A_I A_I^+)$, $A_I$ and $b_I$ are the matrices collecting the vertical vectors $a^j$ and $b^j$ ($j \in I$) in horizontal direction, respectively.

Since we did not deal with the inequality constraints, $(a^s)^T \hat{U} - b_s \leq 0$ may be violated for some $s \in G$ (i.e. $(a^s)^T \hat{U} - b_s > 0$). From now, we deal with such inequality constraint (indexed by $s$) one by one, by considering the following problem

$$\begin{align*}
\text{minimize} & \quad \left\{ \frac{1}{2} (\hat{U} + d)^T Q (\hat{U} + d) + q^T (\hat{U} + d) \right\} \\
\text{subject to} & \quad (a^j)^T (\hat{U} + d) - b_j = 0, \quad j \in I \\
& \quad (a^s)^T (\hat{U} + d) - b_s = 0
\end{align*}$$

with a correction variable $d$.

According to the KKT Theorem, the optimal solution of (20) takes the following form

$$\begin{align*}
\tilde{d} &= -\hat{U} + (A_I^+)^T b_I - H_I (q + \lambda_s a^s) \\
&= -\lambda_s H_I a^s.
\end{align*}$$

Substituting the above equation into the final equation in (20), we obtain

$$\lambda_s = \frac{b_s - (a^s)^T \hat{U}}{-(a^s)^T H_I a^s}.$$  \hspace{1cm} (22)

Then, the optimal $\tilde{d}$ is given by

$$\tilde{d} = \frac{b_s - (a^s)^T \hat{U}}{(a^s)^T H_I a^s} H_I a^s,$$  \hspace{1cm} (23)
which leads to the corrected solution \( \hat{U} + \hat{d} \). In addition, the corresponding multiplier vector is modified by

\[
\lambda_I = -(A_I^T Q^{-1} A_I)^{-1} b_I - A_I^T (q + \lambda_s a^s) \\
= \hat{\lambda}_I + \frac{b_s - (a^s)^T \hat{U}}{(a^s)^T H_I a^s} A_I^T a^s. \tag{24}
\]

Furthermore, in dual method [11], we consider both the primal problem (P) and the dual problem (D). The primal problem is to update the correction vectors, while the dual problem is to choose the multipliers satisfying nonnegative \( \lambda_j \geq 0, j \in I \cap G \). We construct the following algorithm based on the above ideas.

\textbf{Algorithm 1}

(i) (initialization): Assume that \( I^{(0)} = E \). Then, the solution \( U^{(0)} \) which is obtained by solving problem (17) is the initial solution. The multipliers is \( \lambda_I^{(0)} \) and \( \lambda_j = 0 \ (j \in G) \), and set \( k = 0 \).

(ii) (termination condition): If \( U^{(k)} \) is the feasible solution in (16), then it is the optimal solution, and thus the programming is finished. If \( U^{(k)} \) is not the feasible solution in (16), then we choose a constraint index \( s \) where \( U^{(k)} \) is not satisfied, and go to step (iii).

(iii) (direction search): We calculate search direction \( \varepsilon^{(k)} \) and \( r^{(k)} \) where \( \varepsilon^{(k)} := -H_I a^s \), \( r^{(k)} := -A_I^T a^s \).

(iv) (choosing step size): If \( \varepsilon^{(k)} = 0 \), set \( t^{(k)}(P) = \infty \). Otherwise we calculate \( \lambda_s \) by (22). If \( r^{(k)} \geq 0 \), set \( t^{(k)}(D) = \infty \). Otherwise we calculate \( t^{(k)}(D) \), and set \( t^{(k)} = \min \{ \lambda_s, t^{(k)}(D) \} \).

(v) (update): If \( t^{(k)} = \infty \), since feasible solution does not exist in convex quadratic programming, it is finished. If \( t^{(k)}(D) = \infty \), (a) is executed or else (b) is executed.

(a) We remove suffix \( l \) that has the step size \( t^{(k)}(D) \) from suffix set \( I^{(k)} \), and we calculate \( \lambda_j^{(k)} \) and go to step (iii).

\[
\lambda_j^{(k)} := \begin{cases} 
\lambda_j^{(k)} - r_j \lambda_I^{(k)} / r_I, & j \in I^{(k)} \\
\lambda_j^{(k)} r_I, & j = s \\
0, & \text{otherwise}
\end{cases}, \tag{25}
\]

(b) We calculate the following iteration,

\[
\begin{pmatrix} 
\tilde{U} \\
\tilde{\lambda}_I \\
\tilde{\lambda}_s
\end{pmatrix} = 
\begin{pmatrix} 
U^{(k)} \\
\lambda_I^{(k)} \\
\lambda_s^{(k)}
\end{pmatrix} + t^{(k)} \begin{pmatrix} 
\varepsilon^{(k)} \\
r^{(k)}
\end{pmatrix}. \tag{26}
\]

If \( \lambda_s \leq t^{(k)}(D) \), the following \text{b-1} or else \text{b-2} is executed.

\text{b-1} \quad \langle U^{(k+1)}, \lambda_I^{(k+1)}, \lambda_s^{(k+1)} \rangle := \langle \tilde{U}, \tilde{\lambda}_I, \tilde{\lambda}_s \rangle, \quad I^{(k+1)} := I^{(k)} \cup \{ s \}, \quad \text{set } k := k + 1 \text{ and go to step (ii)}.

\text{b-2} \quad \langle U^{(k)}, \lambda_I^{(k)}, \lambda_s^{(k)} \rangle := \langle \tilde{U}, \tilde{\lambda}_I, \tilde{\lambda}_s \rangle, \text{we remove suffix } l \text{ that has the step size } t^{(k)}(D) = -\frac{\lambda_s^{(k)}}{r_I} \text{ from suffix set } I^{(k)}, \text{and calculate } \lambda_j^{(k)} \text{ and go to step (iii)}.

This algorithm is described based on [10].
3. Optimization Problem in MPC

Throughout this paper, we will work with a linear time-invariant (LTI) system of the discrete-time form

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k)
\end{align*}
\]

(27)  
(28)

where \( x(k) \in \mathbb{R}^{n_x} \), \( u(k) \in \mathbb{R}^{n_u} \), and \( y(k) \in \mathbb{R}^{n_y} \) denote the state, control input, and measured output at sampling instant \( k \), respectively; \( A \in \mathbb{R}^{n_x \times n_x} \), \( B \in \mathbb{R}^{n_x \times n_u} \), \( C \in \mathbb{R}^{n_y \times n_x} \) are the system matrices.

When perfect knowledge of the system state \( x(k) \) is available, we aim to construct the control input sequence

\[ u(k), u(k+1), \ldots, u(k+N_p-1), \]

(29)
at each sampling instant \( k \), where \( N_p \) is the prediction horizon, so that a specified cost function is minimized subject to some constraints. Denote the predicted state vectors with the above control input by \( x(k+1|k), x(k+2|k), \ldots, x(k+N_p|k) \). Define the entire predicted output vector and the control input vector as

\[
Y = \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+N_p|k) \end{bmatrix}, \quad U = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_p-1) \end{bmatrix},
\]

(30)

and suppose the reference input vector for \( Y \) is given by the vector \( r_p \). Then, the optimal control problem (OCP) is described by

\[
\begin{align*}
\text{minimize} \quad & f(U) = \frac{1}{2} (r_p - Y)^T Q (r_p - Y) + \frac{1}{2} U^T R U \\
\text{subject to} \quad & GU \leq b,
\end{align*}
\]

(31)  
(32)

where \( Q = Q^T \geq 0 \) and \( R = R^T > 0 \) are weight matrices, the constant matrix \( G \) and the vector \( b \) in (32) describe the linear inequality constraint.

After simple calculation, we obtain the relation between \( Y \) and \( U \) as

\[
Y = W x(k) + Z U
\]

(33)

where

\[
W = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix}, \quad Z = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ CAB^{N_p-1} B & \cdots & CAB & CB \end{bmatrix}.
\]

(34)

Then, the cost function in (31) is rewritten into

\[
f(U) = \frac{1}{2} (r_p - Wx(k) - ZU)^T Q (r_p - Wx(k) - ZU) + \frac{1}{2} U^T R U.
\]

(35)
Since the constraint (32) is linear, the cost function (35) is quadratic with nonnegative definite matrix $Q$ and positive definite matrix $R$, the OCP (35) (and thus (31)) with (32) is a convex quadratic programming (QP) problem. It is known that this type of QP problem has a unique minimum, and can be solved theoretically. Moreover, in typical MPC for real systems, we usually apply the first element $u(k)$ in the control input vector $U$ to the system, and repeat the above optimization procedure at all sampling instant later.

In the previous section, we have reviewed the existing methods for solving the above QP problem (35) with (32). However, these methods are quite time consuming even when using powerful computers. Thus, we are motivated to propose a more efficient algorithm for the above problem in this paper.

4. A Combined Algorithm for QP in MPC and Simulation

In Section 3, we have reduced the MPC implementation to the QP problem

$$\begin{align*}
\text{minimize} & \quad f(U) = \frac{1}{2}(r_p - Wx(k) - ZU)^T Q(r_p - Wx(k) - ZU) + \frac{1}{2}U^T RU \\
\text{subject to} & \quad GU \leq b,
\end{align*}$$

where $W, Z, Q, R, G$ are constant matrices, $x(k), r_p$ are known vectors, and $U$ is the vector to be determined.

4.1. A Combination of numerical algorithms

In order to speed up the numerical calculation for the QP problem, we try to combine the algorithms mentioned in Section 2 as follows.

**Algorithm 2**

(i) Set $k = 0$, $U^{(k)} = 0$ and $\mu^{(k)} = 0$. Choose a positive scalar $\varepsilon$ small enough for termination of the algorithm.

(ii) Calculate the step size $\alpha^{(k)}$ in Lagrangian algorithm satisfying the Wolfe conditions

$$\begin{align*}
\begin{cases}
    f(U^{(k)} + \alpha^{(k)}d^{(k)}) - f(U^{(k)}) & \leq \rho_1 \alpha^{(k)}\nabla f(U^{(k)})^T d^{(k)} \\
    \nabla f(U^{(k)} + \alpha^{(k)}d^{(k)})^T d^{(k)} & \geq \rho_2 \nabla f(U^{(k)})^T d^{(k)}
\end{cases}
\end{align*}$$

where $0 < \rho_1 < \rho_2 < 1$.

(iii) Set $\beta^{(k)} = \alpha^{(k)}$ and perform the following iterative calculation by Lagrangian algorithm.

$$\begin{align*}
U^{(k+1)} & = U^{(k)} - \alpha^{(k)} \left( \nabla f(U^{(k)}) + \nabla g(U^{(k)})^T \mu^{(k)} \right) \\
\mu^{(k+1)} & = \max(\mu^{(k)} + \beta^{(k)} g(U^{(k)}), 0), \quad k = 0, 1, \ldots
\end{align*}$$

Set $k \leftarrow k + 1$.

(iv) If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$, go to Step (v). Otherwise, go to Step (ii).

(v) If $U^{(k)}$ satisfies the constraint (37), stop the algorithm. Otherwise, go to Step (vi).

(vi) If $U^{(k)}$ violates the upper bound, add current upper input constraint to equation constraint set $E$. Otherwise, add current lower input constraint to equation constraint set $E$. (i.e. $U^{(k)}$ violates the lower bound.)
(vii) (initialization): Assume that \( I^{(0)} = E \). Then, the solution \( U^{(0)} \) which is obtained by solving problem (17) is the initial solution. The multipliers is \( \lambda_i^{(0)} \) and \( \lambda_j = 0 \ (j \in G) \), and set \( k = 0 \).

(viii) (direction search): We calculate search direction \( z^{(k)} \) and \( r^{(k)} \) where \( z^{(k)} := -H_f a^s \), \( r^{(k)} := -A^+_j a^s \).

(ix) (choosing step size): If \( z^{(k)} = 0 \), set \( t^{(k)}(P) = \infty \). Otherwise we calculate \( \lambda_s \) by (22). If \( r^{(k)} \geq 0 \), set \( t^{(k)}(D) = \infty \). Otherwise we calculate \( t^{(k)}(D) \), and set \( t^{(k)} = \min\{\lambda_s, t^{(k)}(D)\} \).

(x) (update): If \( t^{(k)} = \infty \), since feasible solution do not exist in convex quadratic programming, it is finished. If \( t^{(k)}(D) = \infty \), (a) is executed or else (b) is executed.

(a) We remove suffix \( l \) that has the step size \( t^{(k)}(D) = -\frac{\lambda_s^{(k)}}{r_l} \) from suffix set \( I^{(k)} \), and we calculate \( \lambda_j^{(k)} \) and go to step (iii).

\[
\lambda_j^{(k)} := \begin{cases} 
\lambda_j^{(k)} - r_j \lambda_i^{(k)}/r_l, & j \in I^{(k)} \\
\lambda_j^{(k)} r_l, & j = s \\
0, & \text{otherwise}
\end{cases}
\]  

(41)

(b) We calculate the following iteration,

\[
\begin{pmatrix}
\tilde{U} \\
\tilde{\lambda}_I \\
\tilde{\lambda}_s
\end{pmatrix} = 
\begin{pmatrix}
U^{(k)} \\
\lambda_i^{(k)} \\
\lambda_s^{(k)}
\end{pmatrix} + t^{(k)} \begin{pmatrix}
z^{(k)} \\
r^{(k)} \\
1
\end{pmatrix}.
\]  

(42)

If \( \lambda_s \leq t^{(k)}(D) \), the following b-1 or else b-2 is executed.

b-1 \((U^{(k+1)}, \lambda_i^{(k+1)}, \lambda_s^{(k+1)}) := (\tilde{U}, \tilde{\lambda}_I, \tilde{\lambda}_s), I^{(k+1)} := I^{(k)} \cup \{s\}, \text{set } k := k + 1 \) and go to step (ii).

b-2 \((U^{(k)}, \lambda_i^{(k)}, \lambda_s^{(k)}) := (\tilde{U}, \tilde{\lambda}_I, \tilde{\lambda}_s), \text{we remove suffix } l \text{ that has the step size } t^{(k)}(D) = -\frac{\lambda_s^{(k)}}{r_l} \text{ from suffix set } I^{(k)}, \text{and calculate } \lambda_j^{(k)} \text{ and go to step (iii).}

Remark 1 (Lagrangian algorithm)
When we use Lagrangian algorithm at Step (ii)-(iii), we need to decide the step size parameters \( \alpha^{(k)} \), \( \beta^{(k)} \) such that the iterative solution converges. These parameters can be constant if convergence is guaranteed. However, it is usually difficult to choose these parameters due to lack of effective criterion.

Remark 2 (Wolfe conditions)
By adding Wolfe conditions at Step (ii), where the parameters are adjusted according to (38), we can obtain the candidate step sizes, but the resultant iteration \( U^{(k)} \) may not satisfy the original constraints.

Remark 3 (Dual method)
We try to add Dual method at Step (vii)-(x) in order to deal with the constraints, which can be performed in a finite number of iterations. More precisely, we pick up the unsatisfied constraints, set them to equations, solve the optimization problem (17), and use the obtained solution to update the iteration \( U^{(k)} \). Then, we proceed to check whether the satisfied constraints remain true or not with the updated value \( U^{(k)} \).
4.2. Numerical simulation

We start from the continuous-time system

\[ \dot{x}(t) = \begin{bmatrix} 0.83 & -3.0 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t), \] (43)

with an unstable system matrix, and discretize it with the sampling period 0.1 to obtain the discrete-time system

\[ x(k+1) = \begin{bmatrix} 1.0707 & -0.3112 \\ 0.1037 & 0.9846 \end{bmatrix} x(k) + \begin{bmatrix} 0.1037 \\ 0.0051 \end{bmatrix} u(k). \] (44)

Since the numerical calculation of a real system must be performed in discrete time, the continuous-time model must be transformed into discrete-time model.

Consider the control input with the upper and lower bound constraint \([-0.3 \leq u(k) \leq 0.5]\), and set the predictive section \(N_p = 3\).

When applying control inputs to a real system, it is necessary to finish the calculation of each control sequence within the sampling periods. Therefore, we use the maximum calculation time (MCT) at all sampling periods to evaluate the algorithms.

With the above preparation, we provide the simulation results in Figures 1-3, where the control input is given by blue lines, the output by red lines, and the predictive input and output are marked green. It can be seen that the control input constraints are satisfied, the tracking performance is good, and thus our algorithm works well.

The Simulations on MATLAB are performed on a personal computer (Intel Core i7-7700 CPU @ 3.60GHz). In Figure 1, the control input is updated by Lagrangian algorithm. In Figure 2, the input is updated by Lagrangian algorithm with Wolfe conditions. In Figure 3, we show that the input is updated by Algorithm 2.

![Figure 1. Control input and output when using Lagrangian algorithm](image)

The MCT when using Lagrangian algorithm is 0.4562, the MCT when using Lagrangian algorithm together with Wolfe conditions is 0.0488, and the MCT when combing all the three
is 0.0915. In the computation, we adjusted the termination condition so as to obtain sufficient accuracy, which generally results in consumption of time in the iteration. It turns out that the proposed combination method can calculate the control input within all sampling periods.

![Figure 2](image1.png)

**Figure 2.** Control input and output when using Lagrangian algorithm with Wolfe conditions

![Figure 3](image2.png)

**Figure 3.** Control input and output when combining Lagrangian algorithm, Wolfe conditions and Dual method

5. Application to an Inverted Pendulum

Here, we apply the algorithms to an inverted pendulum model (Figure 4). The dynamic equations of the cart and the pendulum are described as follows.

- **Cart system**

  \[
  (m_c + m_p)\ddot{z}(t) + m_p l_p \cos \theta(t) \cdot \dot{\theta}(t) = -\mu_c \dot{z}(t) + m_p l_p \dot{\theta}^2(t) \sin \theta(t) + f_c(t)
  \]  

  \[ (45) \]
Pendulum system

\[ m_p l_p \cos \theta(t) \cdot \ddot{z}(t) + (J_p + m_p l_p^2) \ddot{\theta}(t) = -\mu_p \dot{\theta}(t) + m_p l_p g \sin \theta(t) \] 

(46)

Figure 4. Inverted pendulum model

In the simulation, the above differential equations are linearized at the origin, and then discretized with the sampling period 0.1. To implement the MPC in the obtained discrete-time model, we set the constraint as \(-0.3 \leq u(k) \leq 0.5\), the predictive section as \(N_p = 3\), and then carry out the numerical calculation by using the algorithms.

The MCT when using Lagrangian algorithm is 0.5609, the MCT when using Lagrangian algorithm together with Wolfe conditions is 0.1056, and the MCT when combing all the three is 0.0990. We can confirm the end of the calculation within the sampling periods. So, the proposed method can calculate control input even when applied to an inverted pendulum model.

The Simulations on MATLAB are performed on a personal computer (Intel Core i7-7700 CPU @ 3.60GHz). In Figure 5, the control input is updated by Lagrangian algorithm. In Figure 6, the input is updated by Lagrangian algorithm with Wolfe conditions. In Figure 7, we show that the input is updated by Algorithm 2.

Figure 5. Control input and output when using Lagrangian algorithm
Figures 5-7 show that the algorithm works well since the control input is within the given bound and the angle of the pendulum converges to zero.

6. Conclusions
In this paper, we presented an efficient algorithm on Model Predictive Control, which combines Lagrangian algorithm, Wolfe conditions and Dual method. It is shown that the algorithm can solve the MPC optimization problem while decreasing the calculation time compared to the existing algorithms. We will continue working on this line to improve the insufficient part, and to extend the algorithm to stochastic systems.
References

[1] Maciejowski J M 2002 Predictive Control with Constraints (London: Prentice Hall)
[2] Mayne D Q, Rawlings J B, Rao C V and Scokaert P O M 2000 Constrained model predictive control: stability and optimality Automatica 36 789-814
[3] Ohtsuka T 2004 A continuation/GMRES method for fast computation of nonlinear receding horizon control Automatica 40 563-574
[4] Wolfe P 1969 Convergence conditions for ascent methods SIAM Review 11 226-235
[5] Wolfe P 1971 Convergence conditions for ascent methods II: some corrections SIAM Review 13 185-188
[6] Nocedal J and Wright S J 2006 Numerical Optimization (Berlin: Springer)
[7] Kuhn H W and Tucker A W 1951 Proc. of 2nd Berkeley Symp. on Nonlinear Programming (Berkeley: University of California Press) 481-492
[8] Ohtsuka T 2011 An Introduction to Nonlinear Optimal Control (Tokyo: Corona)
[9] Chong E K P and Zak S H 2013 An Introduction to Optimization (New York: Wiley)
[10] Yamashita N 2015 Nonlinear Programming (Tokyo: Asakura)
[11] Goldfarb D and Idnani A 1983 A numerically stable dual method for solving strictly convex quadratic programs Mathematical Programming 27 1-33