European Journal of Control 25 (2015) 51–59

Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.elsevier.com/locate/ejcon

Fast Model Predictive Control with soft constraints

Arthur Richards

Department of Aerospace Engineering, University of Bristol, Queens Building, University Walk, Bristol BS8 1TR, UK

ARTICLE INFO

Article history:
Received 22 November 2013
Received in revised form
7 August 2014
Accepted 19 May 2015
Recommended by B. Jayawardhana
Available online 14 June 2015

Keywords:
Model Predictive Control
Soft constraints
Interior point methods

ABSTRACT

This paper describes a fast optimization algorithm for Model Predictive Control (MPC) with soft constraints. The method relies on the Kreisselmeier–Steinhauser function to provide a smooth approximation of the penalty function for a soft constraint. This is analogous to the approximation of a hard constraint by a smooth logarithmic barrier function. By introducing this approximation directly into the objective of an interior point optimization, there is no need for additional slack variables to capture constraint violation. Simulation results show significant speed-up compared to using slack variables.

1. Introduction

Model Predictive Control (MPC) naturally handles constraints on system behaviour [13]. It is widely adopted in the process control industry [17] and becoming popular for faster systems such as aircraft [7,19]. Although methods for off-line pre-computation of the MPC control law have been developed [1,8], solving the MPC optimization in real-time remains an attractive prospect, motivating the development of highly tailored algorithms for MPC solvers [11,21].

The contribution of this paper is an extension of the Fast MPC method of Wang and Boyd [21] for the efficient inclusion of *soft constraints* [13, Section 3.4]. Since MPC involves the solution of a constrained optimization, it is important to consider if that optimization will be feasible. Many robust MPC methods exist to tackle this problem – see, for examples, Refs [20,6,3] – but these are typically quite complex and demand some model of the uncertainty to be available. A simpler strategy is to allow the optimizer to "soften" some constraints, *i.e.* to violate some of them at a penalty [4].

Soft constraint penalties work best when they are non-smooth [4,9], taking effect only but immediately when a constraint is violated. One way to avoid the complexity of a non-smooth optimizer is to capture the constraint violation using additional "slack" decision variables. The alternative approach of this paper is to avoid slack variables altogether and introduce the constraint violation penalty directly into the cost function of the optimizer. Since a non-smooth objective would cause problems for a fast gradient-based optimizer, the Kreisselmeier–Steinhauser (KS) function [10] is adopted to provide a smooth approximation to the penalty. A similar approach was taken by Wills and Heath [23] using a quadratic loss penalty for constraint violation. Note that the smooth approximation of the 1-norm penalty comes at the cost of losing the "exact penalty function" guarantee [5], *i.e.* the property that constraint violation will not occur unless unavoidable.

An important consideration is maintaining the special sparsity pattern of the Hessian matrix of the quadratic program (QP) that has to be solved on-line. Wang and Boyd's "Fast MPC" algorithm [21] exploits this structure by using a highly tailored factorization of the Hessian to calculate the Newton steps extremely efficiently. The new development of this paper adds soft constraints without increasing the number of decision variables or changing the structure of the Hessian.

The KS function provides a smooth approximation to the maximum over a set of functions, avoiding the gradient discontinuities where the maximum switches from one function to another. It is popular for constraint aggregation in design optimization [16,24], used to combine a large number of constraints $g_i(x) \leq 0 \forall i$ into a single constraint $\max_i g_i(x) \leq 0$ within a gradient-based optimizer. For the work in this paper on soft constraints, the KS function will be employed to represent the soft constraint penalty. In essence, the KS function will approximate the soft constraints in the same way as a logarithmic barrier function [2] approximates hard constraints. As in the case of the log barrier, the approximate solution tends to the exact solution as the barrier parameter is varied.

The paper begins with a review of the Fast MPC method from Ref. [21] in Section 2, including the use of slack variables for soft constraint handling. Section 3.1 provides a brief review of the KS function and its properties. The use of the KS function for soft constraints is developed in detail in Section 3.2. Simulation results

E-mail address: arthur.richards@bristol.ac.uk

http://dx.doi.org/10.1016/j.ejcon.2015.05.003
0947-3580/

and performance comparisons are presented in Section 5 before finishing with conclusions.

2. Review of fast MPC

This section presents the nomenclature to be adopted and reviews the key points of Fast MPC. This review is an abbreviated presentation of the work by Wang and Boyd [21] to provide the context for the new representation of soft constraints. This section also reviews the usual way to include soft constraints in MPC using slack variables, used later for comparison.

2.1. Forming and solving the QP

Consider the following standard MPC optimization problem, with quadratic cost and linear dynamics and constraints [13]. This has to be solved at each time step in order to generate the closed-loop control $u(k)$:

$$\min \sum_{j=0}^{N-1} \ell(\mathbf{x}(k+j), \mathbf{u}(k+j)) + \mathbf{x}(k+N)^T \mathbf{Q}_f \mathbf{x}(k+N) + \mathbf{q}_f^T \mathbf{x}(k+N) \quad (1)$$

subject to $\forall j \in \{0, \ldots, N-1\}$

$$\mathbf{x}(k+j+1) = \mathbf{A}\mathbf{x}(k+j) + \mathbf{B}\mathbf{u}(k+j) \quad (2a)$$

$$\mathbf{F}_f \mathbf{x}(k+N) \le \mathbf{f}_f \quad (2b)$$

$$\mathbf{F}_x \mathbf{x}(k+j) + \mathbf{F}_u \mathbf{u}(k+j)) \le \mathbf{f} \quad (2c)$$

whose decision variables are $(\mathbf{u}(k) \ldots \mathbf{u}(k+N-1))$ and $(\mathbf{x}(k+1) \ldots \mathbf{x}(k+N))$. Define the stage cost to be a combination of linear and quadratic terms:

$$\ell(\mathbf{x}, \mathbf{u}) = \mathbf{x}\mathbf{Q}\mathbf{x} + \mathbf{u}^T \mathbf{R}\mathbf{u} + \mathbf{q}^T \mathbf{x} + \mathbf{r}^T \mathbf{u} \quad (3)$$

Re-arranging the decision variable in the form

$$\mathbf{z} = (\mathbf{u}(k), \mathbf{x}(k+1), \ldots, \mathbf{u}(k+N-1), \mathbf{x}(k+N))$$

means the optimization can be written as a quadratic program (QP) of the form

$$\min_{\mathbf{z}} \mathbf{z}^T \mathbf{H}\mathbf{z} + \mathbf{g}^T \mathbf{z} \quad (4a)$$

subject to

$$\mathbf{P}\mathbf{z} \le \mathbf{h} \quad (4b)$$

$$\mathbf{C}\mathbf{z} = \mathbf{b} \quad (4c)$$

The purpose of ordering in this way is to achieve banded structures in the matrices $\mathbf{P}$, $\mathbf{C}$ and $\mathbf{H}$ that can be exploited for fast solution. Wang and Boyd introduce a logarithmic barrier function to represent the inequality constraints:

$$\phi(\mathbf{z}) = \sum_i -\log(h_i - \mathbf{p}_i^T \mathbf{z}) \quad (5)$$

where $\mathbf{p}_i^T$ is the row i of matrix $\mathbf{P}$. Thus the final form of the optimization is an equality-constrained nonlinear program:

$$\min_{\mathbf{z}} \mathbf{z}^T \mathbf{H}\mathbf{z} + \mathbf{g}^T \mathbf{z} + \kappa\phi(\mathbf{z}) \quad (6a)$$

subject to

$$\mathbf{C}\mathbf{z} = \mathbf{b} \quad (6b)$$

Crucially, this problem is still convex, so it can be solved to optimality by a Newton method [2]. Augmenting the problem with Lagrange multipliers ν associated with constraints (6b), the residuals are

$$\mathbf{r}_d = 2\mathbf{H}\mathbf{z} + \mathbf{g} + \kappa\mathbf{P}^T \mathbf{d} + \mathbf{C}^T \nu \quad (7)$$

$$\mathbf{r}_p = \mathbf{C}\mathbf{z} - \mathbf{b} \quad (8)$$

noting that $\mathbf{P}^T \mathbf{d} = \nabla\phi(\mathbf{z})$ where $d_i = 1/(h_i - \mathbf{p}_i^T \mathbf{z})$ and $\mathbf{p}_i^T$ is row i of matrix $\mathbf{P}$. Then the necessary conditions for optimality are $\mathbf{r}_d = \mathbf{0}$ and $\mathbf{r}_p = \mathbf{0}$. Finding the Newton step requires solving the equation

$$\begin{bmatrix} \Phi & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z} \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} \mathbf{r}_d \\ \mathbf{r}_p \end{bmatrix} \quad (9)$$

where

$$\Phi = 2\mathbf{H} + \kappa\mathbf{P}^T \operatorname{diag}(\mathbf{d})^2 \mathbf{P}, \quad (10)$$

with the second term being $\kappa\nabla^2\phi(\mathbf{z})$.

The key enabler of Fast MPC is that the matrices Φ and $\mathbf{C}$ have a sparse structure that permits efficient solution of the linear system in (9). The reader is directed to Reference [21] to see how this is performed.

2.2. Including soft constraints

Introduce into the problem a set of soft constraints:

$$\tilde{\mathbf{F}}_x \mathbf{x}(k+j) + \tilde{\mathbf{F}}_u \mathbf{u}(k+j) \le \tilde{\mathbf{f}}. \quad (11)$$

Then the predicted soft constraint violation at each step is given by

$$\mathbf{v}(k+j) = \max\{\mathbf{0}, \tilde{\mathbf{F}}_x^T \mathbf{x}(\mathbf{k}+\mathbf{j}) + \tilde{\mathbf{F}}_u^T \mathbf{u}(\mathbf{k}+\mathbf{j}) - \tilde{\mathbf{f}}\} \quad (12)$$

where the "max" is evaluated element by element, such that no element of $\mathbf{v}(k+j)$ can be negative. Finally, the violation is penalized by adding some norm of the signal $\mathbf{v}(k+j)$ to the objective (1). For example, the following term penalizes the worst case violation at each step:

$$\sum_j \| \mathbf{v}(k+j) \|_\infty. \quad (13)$$

de Oliveira and Biegler [4] compared different weighting strategies and identified that the "exact penalty" methods, using either the 1-norm or the ∞-norm, can avoid unnecessary constraint violations. Note that different levels of weighting on violation of each constraint can be achieved by scaling the rows of $\tilde{\mathbf{F}}_x$, $\tilde{\mathbf{F}}_u$ and $\tilde{\mathbf{f}}$.

The most convenient way to introduce soft constraints into the formulation in Section 2.1 is to augment the control signal with a dummy control input $s(k+j)$ to act as a slack variable, such that the scalar input vector $u(k+j)$ is augmented to become $[\mathbf{u}^T(k+j)\, s(k+j)]^T$. Then the soft constraints are folded into additional hard constraints in the form

$$\begin{bmatrix} \mathbf{F}_x \\ \tilde{\mathbf{F}}_x \\ 0 \end{bmatrix} \mathbf{x}(k+j) + \begin{bmatrix} \mathbf{F}_u & 0 \\ \tilde{\mathbf{F}}_u & -\mathbf{1} \\ \mathbf{0} & -1 \end{bmatrix} \begin{pmatrix} \mathbf{u}(k+j) \\ s(k+j) \end{pmatrix} \le \begin{bmatrix} \mathbf{f} \\ \tilde{\mathbf{f}} \\ 0 \end{bmatrix}. \quad (14)$$

Note that this enforces $s(k+j) \ge 0$ and $s(k+j) \ge \tilde{\mathbf{f}}_{x_i}^T \mathbf{x}(k+j) + \tilde{\mathbf{f}}_{u_i}^T \mathbf{u}(k+j)) - \tilde{f}_i$ for all i where $\tilde{\mathbf{f}}_{x_i}^T$ is the row i of $\tilde{\mathbf{F}}_x$, etc. Hence augmenting the cost weight $\mathbf{r}^T = [\mathbf{0}^T 1]$ in (3) penalizes constraint violation in the form (13). Since (14) is identical in form to (2c), the augmented problem can be directly solved using the Fast MPC algorithm presented in Section 2.1.

This approach introduces an additional N decision variable $s(k+j)$, one for every time step j in the horizon. Using different norms for the soft constraint penalty [4], it is possible to introduce just one additional slack variable, capturing the worst case violation across all time steps. However, this destroys the banded structure in the matrix Φ that is central to the fast solution of the QP.

It is interesting to note that the use of a single slack variable would leave Φ with an "arrow" structure [2, p. 670] which can also be exploited for efficient solutions. Alternatively, it is possible to eliminate the slack variables along with Lagrange multipliers in interior point solvers [18, Section 3.2]. However, these ideas are