2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Fast Model Predictive Control with Soft Constraints

Arthur Richards*†
*Department of Aerospace Engineering, University of Bristol
Queens Building, University Walk, Bristol, BS8 1TR, UK
†Lecturer, Email: arthur.richards@bristol.ac.uk

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. 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 slacks.**

I. Introduction

Model Predictive Control (MPC) naturally handles constraints on system behaviour [1]. It is widely adopted in the process control industry [2] and becoming popular for faster systems such as aircraft [3], [4]. Although methods for off-line pre-computation of the MPC control law have been developed [5], [6], solving the MPC optimization in real-time remains an attractive prospect, motivating the development of highly tailored algorithms for MPC solvers [7], [8].

The contribution of this paper is an extension of the Fast MPC method of Wang and Boyd [8] for the efficient inclusion of *soft constraints* [1, 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 [9]–[11] - 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", *i.e.* to violate some of the constraints at a penalty [12], ideally weighted such that constraints will only be violated when no alternative exists [13]. Methods typically involve introducing slack variables to represent the level of constraint violation.

The approach of this paper is to avoid slack variables altogether and introduce the constraint violation penalty directly into the cost function of the optimizer. 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 [8] 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.

Soft constraint penalties work best when they are non-smooth [12], [13], taking effect only but immediately when a constraint is violated. Since a non-smooth objective would cause problems for a fast gradient-based optimizer, the Kreisselmeier-Steinhauser (KS) function [14] is adopted to provide a smooth approximation to the penalty. A similar approach was taken by Wills and Heath [15] 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 [16], *i.e.* the property that constraint violation will not ocur unless unavoidable.

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 [17], [18], 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 [19] approximates hard constraints.

The paper begins with a review of the Fast MPC method from Ref. 8 in Section II, including the use of slack variables for soft constraint handling. Section III 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 IV. Simulation results and performance comparisons are presented in Section V before finishing with conclusions.

II. 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 [8] to provide the context for the new representation of soft constraints. This section also reviews the "traditional" way to include soft constraints in MPC using slack variables, used later for comparison.

A. Forming and Solving the QP

Consider the following standard MPC optimization problem, with quadratic cost and linear dynamics and constraints [1]. This has to be solved at each time step in order

to generate the closed-loop control $u(k)$:

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

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

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

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

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

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,

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

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

$$\min_{\mathbf{z}} \mathbf{z}^T \mathbf{H}\mathbf{z} + \mathbf{g}^T \mathbf{z} \tag{4a}$$

subject to

$$\mathbf{P}\mathbf{z} \leq \mathbf{h} \tag{4b}$$

$$\mathbf{C}\mathbf{z} = \mathbf{b} \tag{4c}$$

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:

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

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

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

subject to

$$\mathbf{C}\mathbf{z} = \mathbf{b} \tag{6b}$$

Crucially, this problem is still convex, so it can be solved to optimality by a Newton method [19]. Augmenting the problem with Lagrange multipliers ν, the residuals are

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

$$\mathbf{r}_p = \mathbf{C}\mathbf{z} - \mathbf{b} \tag{8}$$

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

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

where

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

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

The key enabler of Fast MPC is that the matrices $\mathbf{\Phi}$ and $\mathbf{C}$ have an sparse structure that permits efficient solution of the linear system in (9). The reader is directed to Reference 8 to see how this is performed.

B. Including Soft Constraints

Introduce into the problem a set of soft constraints:

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

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

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

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). de Oliveira and Biegler [12] 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.

The most convenient way to introduce soft constraints into the formulation in Section II-A 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

$$\begin{bmatrix} \mathbf{F}_x \\ \tilde{\mathbf{F}}_x \\ 0 \end{bmatrix} \mathbf{x}(k+j) + \begin{bmatrix} \mathbf{F}_u & 0 \\ \tilde{\mathbf{F}}_u & -\mathbf{1} \\ \mathbf{0} & -1 \end{bmatrix} \begin{pmatrix} \mathbf{u}(k+j) \\ s(k+j) \end{pmatrix} \leq \begin{bmatrix} \mathbf{f} \\ \tilde{\mathbf{f}} \\ 0 \end{bmatrix}. \tag{13}$$

Note that this enforces $s(k+j) \geq 0$ and $s(k+j) \geq \tilde{\mathbf{f}}_{x_i}^T \mathbf{x}(k+j) + \tilde{\mathbf{f}}_{u_i}^T \mathbf{u}(k+j)) - \tilde{f}_i$ for all i where $\tilde{\mathbf{f}}_{x_i}^T$ is 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 $\sum_j \|(v(k+j)\|_\infty$. Since (13) is identical in form to (2c), the augmented problem can be directly solved using the Fast MPC algorithm presented in Section II-A.

This approach introduces an additional N decision variables, one for every time step in the horizon. Using different norms for the soft constraint penalty, 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. Instead, it would leave Φ with an "arrow" structure [19, p670] which can also be exploited for efficient solutions, but this is beyond the scope of this paper. In the remainder of this work, we introduce a method for soft constraints without any additional slack variables.

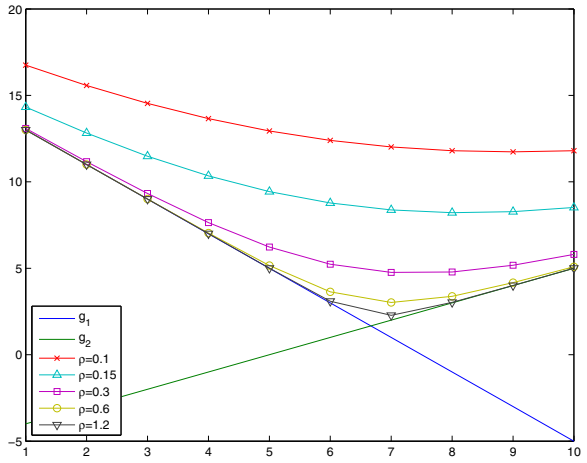


Fig. 1. Example of Kreisselmeier-Steinhauser Function

III. The Kreisselmeier-Steinhauser Function

The Kreisselmeier-Steinhauser (KS) function [14] is a smooth approximation to the maximum of a set of functions,

$$KS(\mathbf{g}(x)) = \frac{1}{\rho}\log\left[\sum_j e^{\rho g_j(x)}\right]$$

with the property that $KS(\mathbf{g}(x)) \to \max_j g_j(x)$ as $\rho \to \infty$. Large values of ρ can lead to numerical problems if one or more of the functions are positive, due to growth of the exponential. Therefore, an alternative but equivalent form is used, which avoids these problems:

$$KS(\mathbf{g}(x)) = g_{\max}(x) + \frac{1}{\rho}\log\left[\sum_j e^{\rho(g_j(x)-g_{\max}(x))}\right] \quad (14)$$

where $g_{\max}(x) = \max_j g_j(x)$. Figure 1 shows an example of the KS function approximating the maximum of two linear functions in one dimension. The effect of the KS function in smoothing the gradient discontinuity is analagous to the effect of a logarithmic barrier function in smoothing the step change from feasible to infeasible. In both cases, the smooth approximation tends to the original function as the parameter ρ is changed.

IV. Soft Constraints using the KS Function

Our approach is to apply the KS function to the "max" operations in the constraint violation expression (12) to achieve a smooth approximation of the penalty $\sum_j \|\mathbf{v}(k+j)\|_1$. Note that alternative approaches to get the ∞-norm are also possible.

First, the soft constraints (11) are rewritten in the compact form adapted from (4b):

$$\tilde{\mathbf{P}}\mathbf{z} \le \tilde{\mathbf{h}} \quad (15)$$

such that the penalty term is

$$\sum_i \max\{0, \tilde{\mathbf{p}}_i^T\mathbf{z} - \tilde{h}_i\} \quad (16)$$

Applying the KS function approximation to each of the "max" operators yields the following smooth penalty function to be added to the objective of the nonlinear minimization problem (6a)

$$\theta(\mathbf{z}) = \sum_i \frac{1}{\rho}\log\left[1 + e^{\rho(\tilde{\mathbf{p}}_i^T\mathbf{z} - \tilde{h}_i)}\right] \quad (17)$$

It remains to determine the gradient and Hessian of this function for it to be incorporated into the Fast MPC Newton method. First, noting that the numerical robustness of the KS function depends on avoiding large exponentials by offsetting from the maximum, as in (14), define two exponentials, one for the case where a constraint is satisfied and the other for its violation

$$e_i^+ = e^{\rho(\tilde{\mathbf{p}}_i^T\mathbf{z} - \tilde{h}_i)} \quad (18)$$

$$e_i^- = e^{\rho(\tilde{h}_i - \tilde{\mathbf{p}}_i^T\mathbf{z})} = 1/e_i^+. \quad (19)$$

Then the gradient of the penalty function is given by

$$\nabla\theta(\mathbf{z}) = \tilde{\mathbf{P}}^T\tilde{\mathbf{d}} \quad (20)$$

where

$$\tilde{d}_i = \frac{e_i^+}{1+e_i^+} = \frac{1}{1+e_i^-}. \quad (21)$$

It is necessary to calculate the vector $\tilde{\mathbf{d}}$ within the optimizer at each iterate of the decision variable $\mathbf{z}$. This can be done element by element using whichever form of (21) is best, numerically, in each case. Typically, it is best to use the smaller of e_i^+ and e_i^- to avoid overflow. Similarly, the Hessian of the penalty is given by

$$\nabla^2\theta(\mathbf{z}) = \rho\tilde{\mathbf{P}}^T\mathrm{diag}(\hat{\mathbf{d}})\tilde{\mathbf{P}} \quad (22)$$

where

$$\hat{d}_i = \frac{e_i^+}{(1+e_i^+)^2} = \frac{e_i^-}{(1+e_i^-)^2}. \quad (23)$$

Observe that the expression for $\hat{d}_i$ is the same regardless of which exponential is used, since the second derivative of the KS function is symmetric about the zero crossing.

Finally, the system of equations defining the Newton step (9) can be modified to accommodate the soft constraints as follows:

$$\begin{bmatrix} \tilde{\mathbf{\Phi}} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \Delta\mathbf{z} \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{r}}_d \\ \mathbf{r}_p \end{bmatrix} \quad (24)$$

where

$$\tilde{\mathbf{\Phi}} = \mathbf{\Phi} + \rho\tilde{\mathbf{P}}^T\mathrm{diag}(\hat{\mathbf{d}})\tilde{\mathbf{P}} \quad (25)$$

and

$$\tilde{\mathbf{r}}_d = \mathbf{r}_d + \tilde{\mathbf{P}}^T\tilde{\mathbf{d}}. \quad (26)$$

By comparing the additional term (22) with the original expression for $\mathbf{\Phi}$ in (10), it can be seen that the block structure of Φ that is crucial to the solution process has not been affected. In the extreme case, if the weights $\mathbf{Q}$ and $\mathbf{R}$ are diagonal and both soft and hard constraints define boxes,

then $\mathbf{\Phi}$ will (still) be diagonal[1]. Even without that extreme case, since (9) and (24) share identical block structure, the methods developed in Ref. 8 for solving (9) can be directly applied to solving (24).

V. Results

In this section, the new method for soft constraint handling will be compared in simulation with a more traditional approach based on slack variables. The examples will consider the case of a double integrator discretized at 4Hz. The states are the position (x_1), the velocity (x_2), and the applied control force (x_3). The input is the change in control force at each step [1]. The cost weights are

$$\begin{aligned} \mathbf{Q} &= \mathbf{Q}_f = \mathrm{diag}(5, 5, 2) \\ \mathbf{R} &= 10 \end{aligned}$$

with hard constraints

$$u(k) \in [-0.5, 0.5]$$

and soft constraints

$$\begin{aligned} x_2(k) &\in [-0.2, 0.2] \\ x_3(k) &\in [-0.1, 0.1] \end{aligned}$$

The simulation includes a constant disturbance, unmodeled in the MPC, equivalent to a control force of 0.01. This ensures that some constraint violation will occur. Each simulation involves a series of increasingly large position reference pulses over a period of 200s. Both optimizers adopt fixed values for the barrier weight $\kappa = 0.002$ and, in the KS case, the parameter $\rho = 10$, in the same spirit as Fast MPC, although this is not a requirement and an increasing sequence of ρ could be be employed, analagous to decreasing κ. Both optimizers were implemented in a Simulink simulation coded in an embedded Matlab block. With a horizon of 10 steps, the optimizer with slack variables has 50 decision variables and 30 equality constraints. The optimizer using the KS function has the same number of constraints and 40 decision variables.

Figures 3 and 4 compare the simulation results for the two optimizers with varying numbers of Newton iterations. The slack variable method (Fig. 3) appears to perform well with 10 iterations, tracking the reference and obeying the velocity constraints when possible. Some violation during positive velocity is expected, due to the unmodeled disturbance. A comparison case using Matlab's `quadprog` optimizer produced very similar results. However, when the number of iterations was reduced to 8, performance degraded severely, with complete loss of convergence for 7 iterations or fewer.

The optimizer using the KS function (Fig. 4) also performs well with 10 iterations. Furthermore it maintains performance much better as the number of iterations is reduced and still has good constraint satisfaction and tracking at 5 iterations. Taking it further down to 3 iterations, distortion is starting to appear in the response.

[1]Note that Φ can never be diagonal if slack variables are used, as the coupling between the slacks and the other variables induce off-diagonal terms.

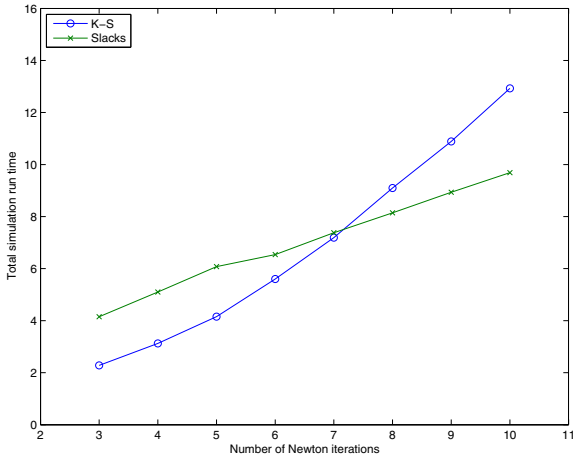


Fig. 2. Variation of Simulation Run Times with Number of Newton Iterations

Figure 2 shows the variation in simulation run time (taken as a crude comparator of solve time) with number of Newton steps performed. It can be seen that the KS method is slightly faster than the slack variable method for very low iteration counts but becomes slower when six or more Newton iterations per step are performed. This is likely due to the need for the exponential function in the KS method. Loosely, looking also at Figures 3 and 4, the performance of the KS method in a simulation taking just under 4 seconds is similar to the slack variable method running for 12, showing a significant potential speed-up.

VI. Conclusion

A new method for including soft constraints in Model Predictive Control has been presented. The method uses the Kreisselmeier-Steinhauser function to represent the constraint violation penalties. The Fast MPC method of Wang and Boyd has been adapted to include these soft constraints. The modified problem retains the favorable matrix structure of the original Fast MPC and can be solved very efficiently. Examples have shown significant potential speed-up of computation in comparison to the use of slack variables to capture constraint softening.

References

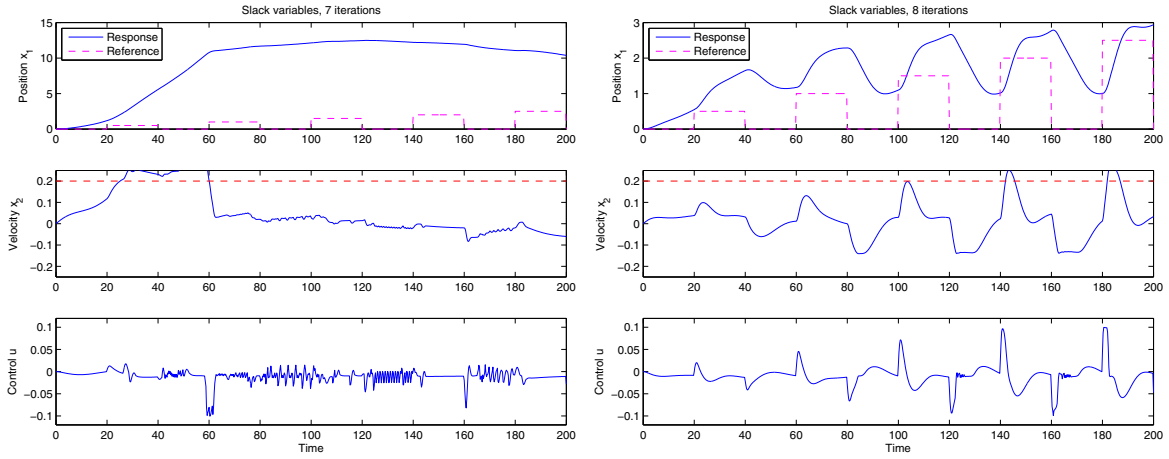
[1] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[2] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[3] A. Hennig and G. J. Balas, "MPC supervisory flight controller: A case study to flight EL AL 1862," in *AIAA Guidance, Navigation and Control Conference*, no. AIAA-2008-6789, Honolulu, Hawaii, August 2008.

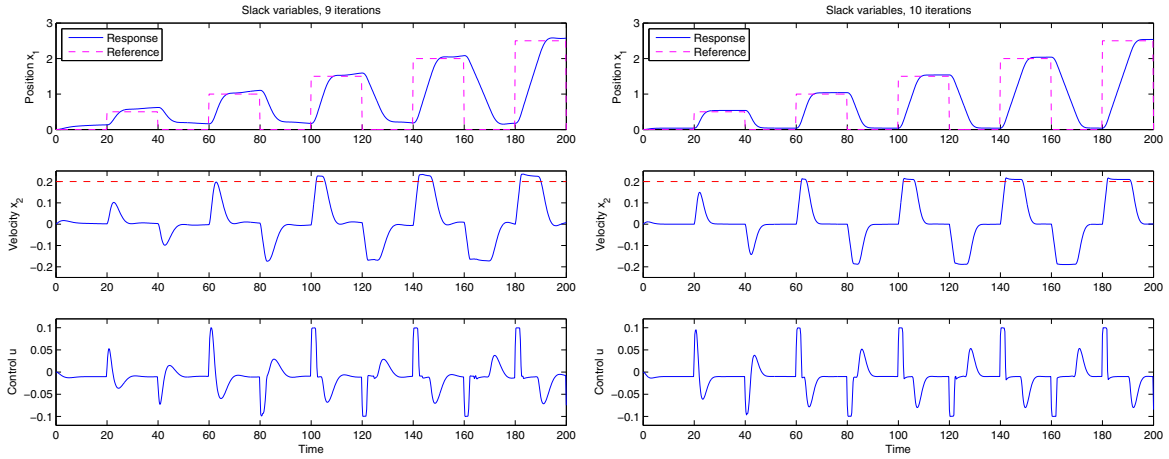
[4] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots in dynamic environment," in *Proceedings of IEEE Conference on Decision and Control*, Maui, Hawaii, 2003, pp. 3621–3626.

[5] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming - the explicit solution," *IEEE Transactions on Automatic Control*, vol. Vol 47 No 12, p. 1974, 2002.



(a) Slack variables: 7 iterations

(b) Slack variables: 8 iterations



(c) Slack variables: 9 iterations

(d) Slack variables: 10 iterations

Fig. 3. Simulation Results for Slack Variable Method with Varying Iteration Limits

[6] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox (MPT)," visited October 2008 2004. [Online]. Available: http://control.ee.ethz.ch/ mpt/

[7] K. Ling, B. Wu, and J. Maciejowski, "Embedded model predictive control (mpc) using a fpga," in *Proc. 17th IFAC World Congress*, 2008, pp. 15 250–15 255.

[8] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 267 –278, march 2010.

[9] P. O. M. Scokaert and D. Q. Mayne, "Min-max model predictive control for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. Vol 43 No 8, pp. 1136–1142, August 1998.

[10] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, "Optimization over state feedback policies for robust control with constraints," *Automatica*, vol. 42, pp. 523–533, 2006.

[11] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: Predictive control with restrictive constraints," *Automatica*, vol. 37(7), pp. 1019–1028, 2001.

[12] N. M. C. de Oliveira and L. T. Biegler, "Constraint handling and stability properties of model predictive control," *American Institute of Chemical Engineers' Journal*, vol. 40, pp. 1138–1155, 1994.

[13] E. Kerrigan and J. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Control 2000 Conference, Cambridge*, 2000.

[14] G. Kreisselmeier and R. Steinhauser, "Systematic control design by optimizing a vector performance index," in *Computer aided design of control systems: proceedings of the IFAC Symposium, Zürich, Switzerland, 29-31 August 1979*. Pergamon, 1980, p. 113.

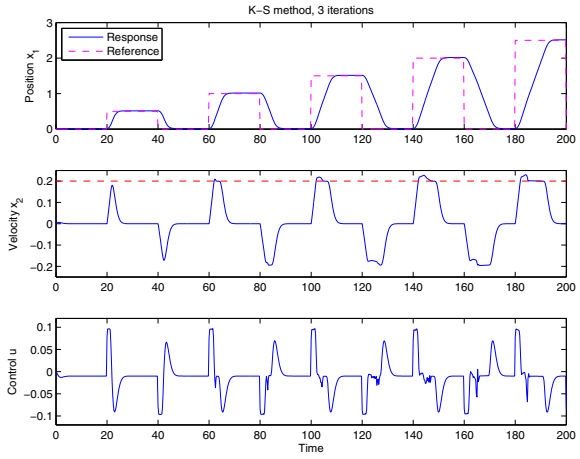
[15] A. Wills and W. Heath, "An exterior/interior-point approach to infeasibility in model predictive control," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 4, 2003, pp. 3701–3705 vol.4.

[16] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Wiley, 1987.

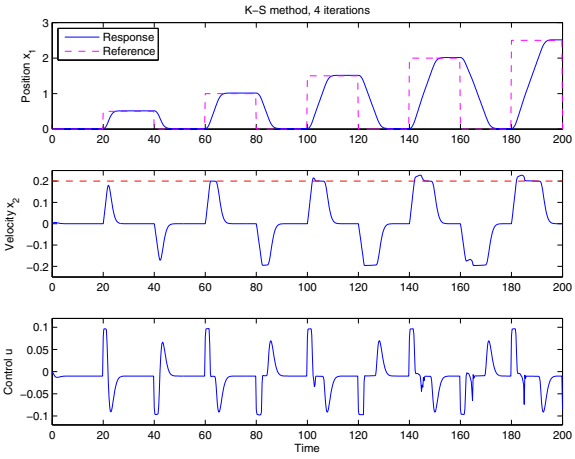
[17] J. Martins and N. Poon, "On structural optimization using constraint aggregation," in *VI World Congress on Structural and Multidisciplinary Optimization WCSMO6, Rio de Janeiro, Brasil*, 2005.

[18] G. Wrenn, "An indirect method for numerical optimization using the kreisselmeier-steinhauser function," National Aeronautics and Space Administration, Office of Management, Scientific and Technical Information Division, Tech. Rep., 1989.

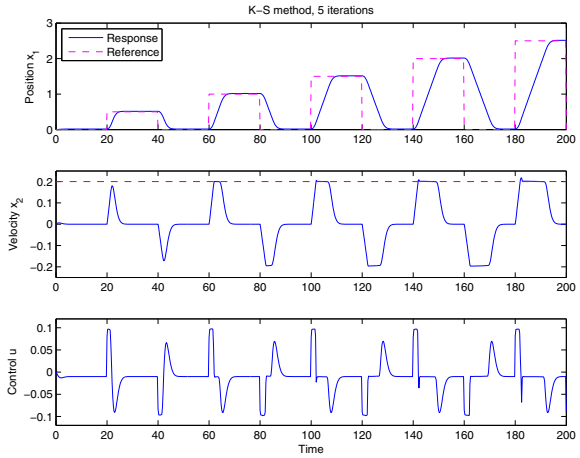
[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.



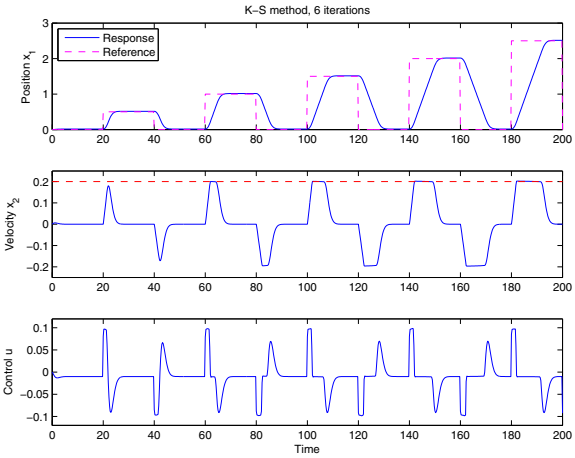
(a) KS function: 3 iterations



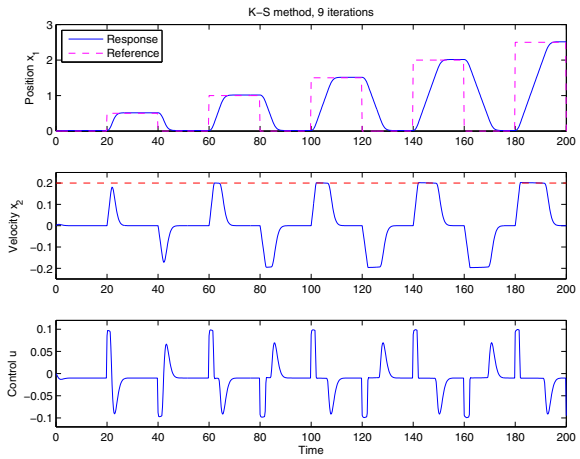
(b) KS function: 4 iterations



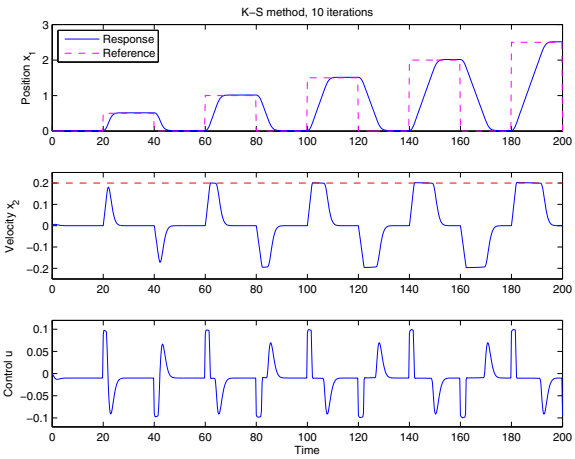
(c) KS function: 5 iterations



(d) KS function: 6 iterations



(e) KS function: 9 iterations



(f) KS function: 10 iterations

Fig. 4. Simulation Results for KS Method with Varying Iteration Limits