A global piecewise smooth Newton method for fast large-scale model predictive control

Panagiotis Patrinos[a], Pantelis Sopasakis[a], Haralambos Sarimveis[*a]

E-mail addresses: patrinos@mail.ntua.gr (P. Patrinos), chvng@mail.ntua.gr (P. Sopasakis), hsarimv@central.ntua.gr (H. Sarimveis).

Abstract

In this paper, the strictly convex quadratic program (QP) arising in model predictive control for constrained linear systems is reformulated as a system of piecewise affine equations. A regularized piecewise smooth Newton method with exact line-search on a convex, differentiable, piecewise-quadratic merit function is proposed for the solution of the reformulated problem. The algorithm has considerable merits when applied to MPC over standard active set or interior point algorithms. Its performance is tested and compared against state-of-the art QP solvers on a series of benchmark problems. The proposed algorithm is orders of magnitudes faster, especially for large-scale problems and long horizons. For example, for the challenging crude distillation unit model of Pannocchia *et al.* (2006) with 252 states, 32 inputs and 90 outputs the average running time of the proposed approach is 1.57 ms.

1. Introduction

Model predictive control (MPC) owes its popularity to the fact that it is the only control methodology that can stabilize linear or nonlinear systems subject to hard input and state constraints. The rationale of the method is to solve at each time instant a finite horizon optimal control problem, obtain an optimal input sequence and apply to the system the first element of that sequence. The popularity of the method is due to the strong theoretical background that has been developed over the past few years (Mayne, Rawlings, Rao, Scockaert, 2000, Rawlings & Mayne, 2009) the development of efficient optimization algorithms and codes, and the substantial increase in computational power. For linear systems the resulting optimal control problem can be formulated as a quadratic program (QP) and off-the-shelf QP solvers allow the application of MPC for small to medium scale processes with a moderate prediction horizon and slow dynamics.

However, the repeated solution of the finite horizon optimal control on-line remains the main bottleneck of the methodology. Specifically, for high sampling rates and large scale systems the computational time needed to solve the QP sub problem becomes a limiting factor of the method. Furthermore, standard MPC theory suggests that a terminal cost and a terminal set need to be incorporated in the finite horizon optimal control problem to guarantee asymptotic stability and constraint satisfaction for the closed-loop system. However, for large-scale linear systems it is prohibitive to compute a polyhedral terminal invariant set, such as the maximal positive invariant set corresponding to the system in closed loop with the unconstrained LQR control law. One could also use a sublevel set of the terminal cost as a terminal constraint but then the favorable structure of the problem would be lost since this is a convex quadratic inequality. Thus, the only viable route to achieve stability for large scale systems is to use a long prediction horizon and possibly to place a larger weight in the terminal cost so as to force the terminal state to lie on the terminal set (Limon, Alamo, Salas, Camacho, 2006, Mayne, Rawlings, 2009). In that case, a large prediction horizon is mandatory and the increase in the size of the QP problem is unavoidable.

There are two major categories of QP algorithms, active set and interior point methods. Active set methods try to identify the set of inequality constraints that are satisfied as equalities at the optimal solution, and can be classified into primal and dual feasible algorithms. Primal active set algorithms require an initial feasible solution and at each iteration, an equality constrained QP corresponding to the working set (the active set of the current solution) is solved in order to compute a direction along which the objective function is decreased. If this direction is nonzero then the maximum step size that maintains primal feasibility is selected and a new working set is computed,

otherwise the constraint that corresponds to the smallest Lagrange multiplier is deleted from the working set. Thus, primal active set methods add or drop constraints from the initial active set as they move towards the optimal solution while maintaining primal feasibility. Due to the complicated structure of the primal solution set, primal active set methods spend on average about one third to one half of the total effort to compute an initial feasible solution (Goldfarb & Idnani, 1983).

On the other hand, dual active set algorithms (Goldfarb & Idnani, 1983) start with a feasible dual vector and constraints are added or deleted from the working set, increasing the objective function of the dual while maintaining dual feasibility. It is easy to compute an initial feasible dual vector since the feasible dual set is usually described by non-negativity or box constraints. Specifically in MPC, one can shift by one stage the primal-dual pair obtained at the previous time step and use it as a warm-start solution for the dual active set solver at the current step. A disadvantage of the dual active set method is that a primal feasible solution corresponding to a dual solution can only be guaranteed upon termination of the algorithm. Hence, a meaningful control sequence, in the sense that it satisfies input, state and terminal constraints can be obtained only upon convergence of the algorithm.

Although theoretically the worst-case running time of active set methods can be exponential in the dimension of the problem, they usually perform very well in practice. On the downside, a notable drawback of active set methods is that the working set changes slowly as the algorithm moves towards the solution. Furthermore, active set methods require the solution of an equality constrained QP at each iteration, which means that a linear system of dimension equal to the number of primal variables plus the number of active constraints needs to be solved. This is usually performed by updating the QR factorization of the indefinite KKT (Karush-Kuhn-Tucker) matrix and then solving the corresponding linear system by backward substitution. Alternatively, Bartlett & Biegler (2006) use a Schur complement technique for updating the working set in the dual active set algorithm, exploiting inherent problem structure present in many problems like model predictive control.

Interior point algorithms, Wright (1997), Nesterov & Nemirovskii (1994), move from an interior point of the feasible set towards the optimal solution by following the so called central path. The main advantage of interior point algorithms against active set methods is their fast convergence rate and their ability to exploit structure in the problem. The latter is due to the fact that the structure of the large system of linear equations that needs to be solved at each iteration, remains unchanged. Specifically, Rao, Rawlings & Wright (1998) applied Mehrotra's predictor-corrector algorithm (Mehrotra, 1992) using a discrete-time Riccati recursion to solve the system of linear equations in MPC problems, thus reducing the computational effort to $O(N(m+n)^3)$, where N is the prediction horizon, m is the number of input variables and n is the number of state variables. Recently, Wang & Boyd (2010) proposed an infeasible start primal barrier method, where the structured system of linear equations at each Newton iteration is solved efficiently using block elimination. They also propose variants of the method where the barrier parameter or the iteration limit is kept fixed. However in that case, the resulting solution may not be primal feasible. In the large, it can be argued that for MPC problems with long prediction horizons interior point methods are more efficient than active set algorithms. Furthermore, interior point methods have polynomial worst-case complexity. However, interior point algorithms possess two considerable disadvantages when applied to MPC, compared to active set solvers. Firstly, although the system of linear equations that needs to be solved at each iterate is structured, its dimension can be considerably larger than that of active set methods. Secondly, since interior point methods need a strictly feasible primal-dual pair as a starting point, warm-starting is an open issue. To sum up, active set methods usually require a large number of computationally cheap iterations while interior point methods need only a few but more expensive steps.

To overcome the limitations of the aforementioned techniques, some other alternatives have been proposed in the literature for solving efficiently MPC problems. Specifically, taking advantage of the simplicity of the constraints appearing in the dual of a strictly convex QP, Axehill & Hansson (2008) proposed a gradient projection method (Nocedal & Wright, 2006) applied to the dual QP of the MPC problem. Unlike, the dual active set algorithm, gradient projection permits large changes of the working set. In Richter, Jones & Morari (2009) the optimal gradient

method of Nesterov (1983) is employed to solve MPC problems with lower and upper bound constraints on the manipulated variables only. The method requires only a matrix-vector product per iteration for computing the gradient. The simple structure of the constraint set allows to compute the projection efficiently. Using the theory of Nesterov (1983), they provide an upper bound for the number of iterations needed to achieve certain accuracy for the optimal solution of the MPC problem. However, the inability of the approach to handle state constraints is certainly a limiting factor for the method. Furthermore, although the method of Nesterov is optimal in terms of rate of convergence among methods that use first order information, it may require many iterations to achieve a desired level of accuracy when the condition number of the hessian is large. Another drawback is that the upper bound in the number of iterations for the warm-start approach requires the off-line solution of a non-convex multi-level optimization problem, which is extremely hard to solve, especially for large-scale systems.

On the other hand, the observation that MPC for constrained linear systems can be formulated as a parametric QP (Bemporad, Morari, Dua & Pistikopoulos, 2002), has enabled the off-line solution of the optimal control problem and the explicit calculation of the MPC controller as a piecewise affine mapping of the measured state. Therefore, only the evaluation of a piecewise affine mapping is needed to compute the MPC control law on-line. Despite the fact that efficient algorithms for parametric QPs have been recently developed, Patrinos & Sarimveis (2010), Fukuda, Jones & Columbano (2009), the complexity of the MPC controller increases exponentially with the state and input dimensions, and the prediction horizon. Hence, the applicability of parametric programming is limited to small to medium scale systems and prediction horizons. To overcome the limitation of parametric MPC, Ferreau, Bock & Diehl (2008) propose an online active set strategy that exploits the solution information of the previous QP by moving along a homotopy path (Best, 1996) from the previous to the current solution while adding and dropping constraints like a usual active set algorithm. Provided that the active set does not change much from one QP to the next, this approach performs faster than a standard active set solver. Explicit MPC enumerates all critical regions for an MPC problem, i.e. all full dimensional polyhedral sets for which a given combination of active constraints remains optimal. However, in real-time the system traverses only a small fraction of those critical regions. Based on this observation, Pannocchia, Rawlings & Wright (2007) proposed the partial enumeration method. As the name suggests, only a small number of active sets and expressions of the PWA control law are computed off-line via simulations and stored in a look-up table. In real-time, the table of active sets is scanned in terms of decreasing order of frequency of appearance. If none of the already stored active sets satisfies the KKT conditions they provide some alternative options, like solving an MPC problem with a smaller horizon, look for the least suboptimal active set, or use the previously computed control sequence. Independently of the controller's calculation, they solve the MPC problem to obtain the true optimal active set and update the look-up table.

In the present work, we show how MPC for linear systems with arbitrary polyhedral state and control constraints reduces to finding a zero of a system of piecewise affine equations. One could then apply a piecewise smooth Newton method (Kojima & Shindo, 1986, Facchinei & Pang, 2003b) in order to find the MPC control law. The algorithm converges in quadratic rate locally. In order to globalize the Newton method and obtain convergence from any starting point, whether it is feasible or not for the QP, one can follow the route used in most of the globalized versions of Newton methods and perform line search based on a merit function. For the MPC problem one can easily obtain a convex, continuously differentiable, piecewise quadratic merit function, i.e. a convex quadratic spline, whose gradient is equivalent to the optimality conditions of the MPC problem. Therefore, MPC essentially reduces to the unconstrained minimization of a convex quadratic spline. The technique is inspired by the work of Li & Swetits (1997, 1999). It is worth noting that such kind of reformulations have been proposed for support vector machine classification with very encouraging results (Mangasarian, 2002). Furthermore, similar piecewise smooth Newton methods have been employed for the solution of Huber's M-estimation problems in linear regression (Madsen & Nielsen, 1990, Chen & Pinar, 1998) with very encouraging results. Exploiting the structure of the problem, it will be shown that the system of linear equations that needs to be solved at each iteration is positive semidefinite and of significantly smaller dimension than that of the dimensions of the problem. The linear system of equations can be solved effectively by updating the Cholesky factor at each iteration. Furthermore, exact line-search can be performed very fast, leading to very fast convergence rates and small computational times.

The proposed algorithm is compared against a dual active set solver (QPC, Wills, 2009), an interior point solver (BPMPD, Mészáros, 1999) and the online active set strategy (qpOASES) of Ferreau, Bock and Diehl (2008) in random instances of MPC problems and various benchmark problems from the literature. The results are very encouraging, especially for large-scale systems and long prediction horizons.

The paper is organized as follows: Section 2 describes the notation used throughout the paper. Section 3 contains the standard MPC formulation for constrained linear systems. Section 4 presents the reformulation of the strictly convex quadratic program resulting from MPC as a system of piecewise affine (PWA) equations and a local piecewise smooth Newton method is presented. In section 5 a continuously differentiable piecewise quadratic merit function for the system of PWA equations. Section 6 describes the regularized piecewise smooth Newton method with exact line search. Section 7 contains implementation details regarding Cholesky factor updates and the line search procedure. Section 8 gives a detailed implementation of the proposed algorithm for MPC problems. Section 9 presents a qualitative comparison between the regularized Newton method and known algorithms for QP when applied to MPC. In section 10, the proposed algorithm is compared against state-of-the art QP solvers in benchmark MPC problems. Finally, the paper ends with some concluding remarks and possible future extensions at section 11.

2. Notation

The finite set of integers $\{1,\ldots,m\}$ is denoted by $[1,m]$. For a set $\mathcal{I} \subseteq [1,m]$ $\mathcal{I}^c$ denotes its complement in $[1,m]$, i.e. $\mathcal{I}^c = [1,m] \setminus \mathcal{I}$. If $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a matrix, $\boldsymbol{c} \in \mathbb{R}^m$ is a vector and $\mathcal{I} \subseteq [1,m]$, $\mathcal{J} \subseteq [1,n]$ then $\mathbf{A}_{\mathcal{I}\cdot}$ is the matrix formed by the rows of $\mathbf{A}$ whose indices are in $\mathcal{I}$, $\mathbf{A}_{\mathcal{I}\mathcal{J}}$ denotes the matrix formed by the rows and columns of $\mathbf{A}$ whose indices are in $\mathcal{I}$ and $\mathcal{J}$ respectively and $\boldsymbol{c}_{\mathcal{I}}$ denotes the vector formed by the elements of $\boldsymbol{c}$ whose indices are in $\mathcal{I}$. When we find it convenient, we will use the following notation $\mathbf{A}_{1:k,1:j}$ to denote the submatrix of $\mathbf{A}$ formed by the first k rows and j columns of $\mathbf{A}$. The symbol $\otimes$ denotes the kronecker product. For a vector $\boldsymbol{y} \in \mathbb{R}^m$, $[\boldsymbol{y}]_+$ denotes a vector whose i-th component is $\max\{\boldsymbol{y}_i, 0\}$. For a given pair of vectors $\boldsymbol{\ell} \in \mathbb{R}^m$, $\boldsymbol{u} \in \mathbb{R}^m$, with $\boldsymbol{\ell}_i \leqslant \boldsymbol{u}_i$ for $i \in [1,m]$, $\text{mid}(\boldsymbol{\ell}, \boldsymbol{u}; \boldsymbol{y})$ denotes the vector whose i-th component is $\max\{\min\{\boldsymbol{y}_i, \boldsymbol{u}_i\}, \boldsymbol{\ell}_i\}$ for any $\boldsymbol{y} \in \mathbb{R}^m$.

3. Model predictive control for constrained linear systems

We consider the following discrete-time linear, time-invariant system:

$$\boldsymbol{x}(t+1) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t) \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ is the state and $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ is the control input. We assume perfect state measurement and that $(\mathbf{A}, \mathbf{B})$ is stabilizable. The system's input and state should belong to the following polyhedral set:

$$\mathcal{Z} = \{(\boldsymbol{u}, \boldsymbol{x}) \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \mid \boldsymbol{f}_{\min} \leqslant \mathbf{F}_{\boldsymbol{u}}\boldsymbol{u} + \mathbf{F}_{\boldsymbol{x}}\boldsymbol{x} \leqslant \boldsymbol{f}_{\max}\} \tag{2}$$

where $\mathbf{F}_{\boldsymbol{u}}$, $\mathbf{F}_{\boldsymbol{x}}$, are $n_c \times n_u$ and $n_c \times n_x$ matrices respectively, $\boldsymbol{f}_{\min} \in \mathbb{R}^{n_c}$, $\boldsymbol{f}_{\max} \in \mathbb{R}^{n_c}$ with $-\infty \leqslant \boldsymbol{f}^i_{\min} < \boldsymbol{f}^i_{\max} \leqslant \infty$.

Consider the following regulator problem:

$$V_N(\boldsymbol{x}(t)) = \min \left\{ \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) + V_f(\boldsymbol{x}_N) \;\middle|\; \begin{array}{l} \boldsymbol{x}_0 = \boldsymbol{x}(t) \\ \boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k \\ (\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{Z}, k \in [0, N-1] \\ \boldsymbol{x}_N \in \mathcal{X}_f \end{array} \right\} \tag{3}$$

where $\ell(\boldsymbol{x},\boldsymbol{u}) = \frac{1}{2}(\boldsymbol{x}'\mathbf{Q}\boldsymbol{x} + \boldsymbol{u}'\mathbf{R}\boldsymbol{u})$ and $V_f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}'\mathbf{P}_f\boldsymbol{x}$ and $\mathcal{X}_f = \{\boldsymbol{x} \in \mathbb{R}^{n_x} \mid \boldsymbol{k}_{\min} \leqslant \mathbf{H}_f\boldsymbol{x} \leqslant \boldsymbol{k}_{\max}\}$ is a polyhedral terminal set, where $\mathbf{H}_f$ is an $n_f \times n_x$ matrix and $\boldsymbol{k}_{\min} \in \mathbb{R}^{n_f}$, $\boldsymbol{k}_{\max} \in \mathbb{R}^{n_f}$ with $-\infty \leqslant \boldsymbol{k}^i_{\min} < \boldsymbol{k}^i_{\max} \leqslant \infty$. We assume that $\mathbf{R}$ is symmetric positive definite and $\mathbf{Q}$, $\mathbf{P}_f$ are symmetric positive semidefinite matrices of compatible dimensions. . Notice that we have used lower and upper bounds in the definition of $\mathcal{Z}$ and $\mathcal{X}_f$. This assumption is not restrictive, since we allow upper or lower bounds to be equal to infinity. In fact, it is a common practice in MPC to have lower and upper bounds on the input and state variables. Furthermore, it will be shown next that this formulation significantly reduces the complexity of the proposed technique. Let $\boldsymbol{x} = (\boldsymbol{x}_0,\ldots,\boldsymbol{x}_N)$ and $\boldsymbol{u} = (\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{N-1})$.

Removing equality constraints, performing trivial algebraic manipulations and omitting terms from the cost function that are independent of $\boldsymbol{u}$, the finite horizon optimal control problem can be expressed as:

$$\min\{\tfrac{1}{2}\boldsymbol{u}'\mathbf{M}\boldsymbol{u} + \boldsymbol{c}(\boldsymbol{x}(t))'\boldsymbol{u} \mid \boldsymbol{b}_{\min}(\boldsymbol{x}(t)) \leqslant \mathbf{G}\boldsymbol{u} \leqslant \boldsymbol{b}_{\max}(\boldsymbol{x}(t))\} \tag{4}$$

where $\mathbf{M} = \tilde{\mathbf{R}} + \tilde{\mathbf{B}}'\tilde{\mathbf{Q}}\tilde{\mathbf{B}}$, $\boldsymbol{c}(\boldsymbol{x}) = \mathbf{C}\boldsymbol{x}$, $\mathbf{C} = \tilde{\mathbf{B}}'\tilde{\mathbf{Q}}\tilde{\mathbf{A}}$, $\boldsymbol{b}_{\min}(\boldsymbol{x}) = \tilde{\boldsymbol{f}}_{\min} - \tilde{\mathbf{F}}_{\boldsymbol{x}}\tilde{\mathbf{A}}\boldsymbol{x}$, $\boldsymbol{b}_{\max}(\boldsymbol{x}) = \tilde{\boldsymbol{f}}_{\max} - \tilde{\mathbf{F}}_{\boldsymbol{x}}\tilde{\mathbf{A}}\boldsymbol{x}$,

$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{I}_N \otimes \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_f \end{bmatrix}$, $\tilde{\mathbf{R}} = \mathbf{I}_N \otimes \mathbf{R}$, $\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{I} \\ \mathbf{A} \\ \vdots \\ \mathbf{A}^N \end{bmatrix}$, $\tilde{\mathbf{B}} = \mathbf{E}\left(\mathbf{I}_N \otimes \mathbf{B}\right)$, $\mathbf{E} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \ldots \mathbf{0} \\ \mathbf{I}_{n_x} & \mathbf{0} & \ldots \mathbf{0} \\ \mathbf{A} & \mathbf{I}_{n_x} & \ldots \mathbf{0} \\ \vdots & \vdots & \ddots \vdots \\ \mathbf{A}^{N-1} & \mathbf{A}^{N-2} & \ldots \mathbf{I}_{n_x} \end{bmatrix}$ and where

$\tilde{\mathbf{F}}_{\boldsymbol{u}} = \begin{bmatrix} \mathbf{I}_N \otimes \mathbf{F}_u \\ \mathbf{0} \end{bmatrix}$, $\tilde{\mathbf{F}}_{\boldsymbol{x}} = \begin{bmatrix} \mathbf{I}_N \otimes \mathbf{F}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_f \end{bmatrix}$, $\tilde{\boldsymbol{f}}_{\min} = \begin{bmatrix} \mathbf{1}_N \otimes \boldsymbol{f}_{\min} \\ \boldsymbol{k}_{\min} \end{bmatrix}$, $\tilde{\boldsymbol{f}}_{\max} = \begin{bmatrix} \mathbf{1}_N \otimes \boldsymbol{f}_{\max} \\ \boldsymbol{k}_{\max} \end{bmatrix}$.

A similar formulation can be obtained for set-point tracking, offset-free MPC and disturbance rejection problems. We omit the details for brevity.

4. Reformulation of MPC problem as system of piecewise affine equations

Consider the strictly convex quadratic program:

$$\min\{\tfrac{1}{2}\boldsymbol{u}'\mathbf{M}\boldsymbol{u} + \boldsymbol{c}'\boldsymbol{u} \mid \boldsymbol{b}_{\min} \leqslant \mathbf{G}\boldsymbol{u} \leqslant \boldsymbol{b}_{\max}\} \tag{5}$$

The structure of (5) is the same with that of the MPC problem (4), but the dependence on the current state has been omitted for shake of clarity of exposition. According to the Karush-Kuhn-Tucker conditions, $\boldsymbol{u}$ is optimal for (5) if and only if there exists a $\boldsymbol{y} \in \mathbb{R}^m$ such that:

$$\mathbf{M}\boldsymbol{u} + \boldsymbol{c} + \mathbf{G}'\boldsymbol{y} = 0 \tag{6}$$

$$\begin{aligned} \boldsymbol{b}_i^{\min} &= \mathbf{G}_{i\cdot}\boldsymbol{u} && \Rightarrow \boldsymbol{y}_i \geqslant 0 \\ \boldsymbol{b}_i^{\min} < \mathbf{G}_{i\cdot}\boldsymbol{u} &< \boldsymbol{b}_j^{\max} && \Rightarrow \boldsymbol{y}_i = 0 \\ \mathbf{G}_{i\cdot}\boldsymbol{u} &= \boldsymbol{b}_i^{\max} && \Rightarrow \boldsymbol{y}_i \leqslant 0 \end{aligned} \tag{7}$$

The complementarity conditions (7) are obviously equivalent to:

$$\begin{aligned} \tau \boldsymbol{b}_i^{\min} = \tau \mathbf{G}_{i\cdot} \boldsymbol{u} \quad &\Rightarrow \boldsymbol{y}_i \geqslant 0 \\ \tau \boldsymbol{b}_i^{\min} < \tau \mathbf{G}_{i\cdot} \boldsymbol{u} < \tau \boldsymbol{b}_j^{\max} \quad &\Rightarrow \boldsymbol{y}_i = 0 \\ \tau \mathbf{G}_{i\cdot} \boldsymbol{u} = \tau \boldsymbol{b}_i^{\max} \quad &\Rightarrow \boldsymbol{y}_i \leqslant 0 \end{aligned} \tag{8}$$

for any positive scalar τ. The meaning of τ will be clear later on. Thus, (6) and (8) are both necessary and sufficient conditions for optimality for any $\tau > 0$.

Since $\mathbf{M}$ is positive definite, we can solve (6) to express $\boldsymbol{u}$ as an affine function of $\boldsymbol{y}$:

$$\boldsymbol{u} = -\mathbf{M}^{-1}(\mathbf{G}'\boldsymbol{y} + \boldsymbol{c}) \tag{9}$$

In order to express the complementarity conditions (8) as a system of nonlinear equations we will use the following definition of a B-function (Facchinei & Pang, 2003b).

Definition 1

Given a pair of extended valued scalars σ and σ' with $\sigma < \sigma'$, we call a function $\phi(\sigma,\sigma';\cdot,\cdot) : \mathbb{R}^2 \to \mathbb{R}$ a B-function (B for box) if $\phi(\sigma,\sigma';r,s) = 0$ if and only if $\sigma \leqslant r \leqslant \sigma'$ and (r,s) satisfies:

$$\begin{aligned} \sigma = r \quad &\Rightarrow s \geqslant 0 \\ \sigma < r < \sigma' &\Rightarrow s = 0 \\ r = \sigma' &\Rightarrow s \leqslant 0 \end{aligned} \tag{10}$$

Hence, a B-function allows the reformulation of complementarity constraints like (8) as a system of nonlinear equations. For our purposes we will consider the B-function derived from the mid function. It is easy to verify that:

$$\phi(\sigma,\sigma';r,s) = r - \mathrm{mid}(\sigma,\sigma';r+s) \tag{11}$$

is a piecewise affine B-function. Therefore, the complementarity conditions (8) can be expressed using (11) as follows:

$$\tau\mathbf{G}\boldsymbol{u} - \mathrm{mid}(\tau\boldsymbol{b}^{\min}, \tau\boldsymbol{b}^{\max}; \tau\mathbf{G}\boldsymbol{u} + \boldsymbol{y}) = 0 \tag{12}$$

Next, equation (9) is employed to eliminate $\boldsymbol{u}$ from (12), arriving at the following system of piecewise affine equations:

$$\Phi_{\tau,\mathrm{mid}}(\boldsymbol{y}) = 0 \tag{13}$$

where:

$$\Phi_{\tau,\mathrm{mid}}(\boldsymbol{y}) = \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d}) + \mathrm{mid}(\tau\boldsymbol{b}^{\min}, \tau\boldsymbol{b}^{\max}; \boldsymbol{y} - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})) \tag{14}$$

with $\mathbf{D} \equiv \mathbf{G}\mathbf{M}^{-1}\mathbf{G}'$ and $\boldsymbol{d} = \mathbf{G}\mathbf{M}^{-1}\boldsymbol{c}$. Notice that $\mathbf{D}$ is positive semidefinite (Horn & Johnson, 1990). We denote the set of solutions of (13) as $\mathcal{Y}^\star$. Since (13) is equivalent to the optimality conditions for the dual of the QP in (5), it follows that $\mathcal{Y}^\star$ is polyhedral. Summing up, one can compute the solution of the QP (4) by solving the system of PWA equations (13) to compute a dual optimal solution and then use (9) to compute $\boldsymbol{u}$.

The piecewise affine mapping has an amenable structure that deserves further investigation. Let $S_{-1,i} \triangleq (-\infty, \tau\boldsymbol{b}_i^{\min}]$, $S_{0,i} \triangleq [\tau\boldsymbol{b}_i^{\min}, \tau\boldsymbol{b}_i^{\max}]$ and $S_{1,i} \equiv [\tau\boldsymbol{b}_i^{\max}, \infty)$. For $\kappa = (\kappa_1, \ldots, \kappa_m)$ with $\kappa_i \in \{-1, 0, 1\}$, define:

$$C_\kappa = \{\boldsymbol{y} \in \mathbb{R}^m \mid \boldsymbol{y}_i - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})_i \in S_{\kappa_i, i},\ i \in [1,m]\} \tag{15}$$

Obviously C_κ is a polyhedral set and $\bigcup_{\kappa \in \{-1,0,1\}^m} C_\kappa = \mathbb{R}^m$. For each $\kappa \in \{-1,0,1\}^m$ let $\alpha_\kappa = \{j \in [1,m] \mid \kappa_j = 0\}$, $\beta_\kappa = \{j \in [1,m] \mid \kappa_j = -1\}$, $\gamma_\kappa = \{j \in [1,m] \mid \kappa_j = 1\}$ and $\delta_\kappa = \beta_\kappa \bigcup \gamma_\kappa$. For each subset δ of $[1,m]$ let $\mathbf{E}_\delta$ denote the $m \times m$ diagonal matrix with its j-th diagonal element being equal to 1 if $j \in \delta$ and 0 otherwise. Then the piecewise affine mapping (14) can be described as follows:

$$\Phi_{\tau,\text{mid}}(\boldsymbol{y}) = (\mathbf{E}_{\delta_\kappa} \tau\mathbf{D} + \mathbf{E}_{\alpha_\kappa})\boldsymbol{y} + \tau(\mathbf{E}_{\delta_\kappa}\boldsymbol{d} + \mathbf{E}_{\beta_\kappa}\boldsymbol{b}^{\min} + \mathbf{E}_{\gamma_\kappa}\boldsymbol{b}^{\max}), \text{ if } \boldsymbol{y} \in C_\kappa \tag{16}$$

Hence, the mapping $\Phi_{\tau,\text{mid}}$ coincides with an affine mapping of the form (16) in each polyhedral set C_κ.

For an arbitrary $\boldsymbol{y} \in \mathbb{R}^m$ consider the following index sets:

$$\begin{aligned}
\alpha(\boldsymbol{y}) &= \{i \in [1,m] \mid \tau\boldsymbol{b}_i^{\min} < \boldsymbol{y}_i - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})_i < \tau\boldsymbol{b}_i^{\max}\} \\
\beta_=(\boldsymbol{y}) &= \{i \in [1,m] \mid \tau\boldsymbol{b}_i^{\min} = \boldsymbol{y}_i - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})_i\} \\
\beta_<(\boldsymbol{y}) &= \{i \in [1,m] \mid \tau\boldsymbol{b}_i^{\min} > \boldsymbol{y}_i - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})_i\} \\
\gamma_=(\boldsymbol{y}) &= \{i \in [1,m] \mid \boldsymbol{y}_i - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})_i = \tau\boldsymbol{b}_i^{\max}\} \\
\gamma_>(\boldsymbol{y}) &= \{i \in [1,m] \mid \boldsymbol{y}_i - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})_i > \tau\boldsymbol{b}_i^{\max}\}
\end{aligned} \tag{17}$$

Let $\delta(\boldsymbol{y}) = \beta_<(\boldsymbol{y}) \bigcup \gamma_>(\boldsymbol{y})$ and $\delta_=(\boldsymbol{y}) = \beta_=(\boldsymbol{y}) \bigcup \gamma_=(\boldsymbol{y})$. Consider the following family of index sets:

$$\mathscr{B}(\boldsymbol{y}) = \{\mathcal{I} \subseteq [1,m] \mid \mathcal{I} = \delta(\boldsymbol{y}) \bigcup \delta',\ \delta' \subseteq \delta_=(\boldsymbol{y})\} \tag{18}$$

For each $\mathcal{I} \in \mathscr{B}(\boldsymbol{y})$, let $\mathbf{H}(\mathcal{I}) = \mathbf{E}_\mathcal{I}\tau\mathbf{D} + \mathbf{E}_\mathcal{N}$, where $\mathcal{N} = [1,m] \setminus \mathcal{I}$. Consider the set valued map $\mathcal{A}_{\text{mid}} : \mathbb{R}^m \rightrightarrows \mathbb{R}^{m \times m}$:

$$\mathcal{A}_{\text{mid}}(\boldsymbol{y}) = \left\{\mathbf{H}(\mathcal{I}) \middle| \mathcal{I} \in \mathscr{B}(\boldsymbol{y})\right\} \tag{19}$$

Notice that $\mathcal{A}_{\text{mid}}(\boldsymbol{y})$ contains $2^{|\delta_=(\boldsymbol{y})|}$ matrices. It is easy to notice that each of the $2^{|\delta_=(\boldsymbol{y})|}$ matrices corresponds to the Jacobian of the affine pieces (16)to which $\boldsymbol{y}$ belongs. The set valued mapping $\mathcal{A}_{\text{mid}} : \mathbb{R}^m \rightrightarrows \mathbb{R}^{m \times m}$ plays the same role as the Jacobian of a smooth system of equations in the classical Newton method. In fact, it can be easily shown that $\mathcal{A}_{\text{mid}}$ is a strong Newton approximation in the sense of Definition 7.2.2 in Facchinei & Pang (2003b). The Newton direction is calculated by solving the following linear system:

$$\mathbf{H}\boldsymbol{r} = -\Phi_{\tau,\text{mid}}(\boldsymbol{y}) \tag{20}$$

for any $\mathbf{H} \in \mathcal{A}_{\text{mid}}(\boldsymbol{y})$. In the spirit of Algorithm 7.2.14 in Facchinei & Pang (2003b), we consider the following local Newton method:

Algorithm 1: Piecewise Smooth Newton Algorithm

Input : $\boldsymbol{y}^0 \in \mathbb{R}^m$

Step 1: Set $k = 0$

Step 2: If $|| \Phi_{\tau,\text{mid}}(\boldsymbol{y}^k) || \leqslant \varepsilon$ stop

Step 3: Select an element $\mathbf{H}^k \in \mathcal{A}_{\text{mid}}(\boldsymbol{y}^k)$. Find a solution $\boldsymbol{r}^k \in \mathbb{R}^m$ of the linear system $\mathbf{H}^k \boldsymbol{r} = -\Phi_{\tau,\text{mid}}(\boldsymbol{y}^k)$

Step 4: Set $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \boldsymbol{r}^k$ and $k \leftarrow k+1$. Go to step 2.

The solution of the linear system (20) in step 3 is the critical step of the piecewise smooth Newton method. However, due to the simplicity of the elements of $\mathcal{A}_{\text{mid}}$, (20) has considerably favorable structure that can be exploited to reduce computations. Specifically, consider a $\boldsymbol{y} \in \mathbb{R}^m$ and choose $\mathcal{I} \in \mathscr{B}(\boldsymbol{y})$. Let $\boldsymbol{q} = \mathbf{E}_{\delta(\boldsymbol{y}) \cup \delta_=(\boldsymbol{y})} \boldsymbol{d} + \mathbf{E}_{\beta_<(\boldsymbol{y}) \cup \beta_=(\boldsymbol{y})} \boldsymbol{b}^{\min} + \mathbf{E}_{\gamma_>(\boldsymbol{y}) \cup \gamma_=(\boldsymbol{y})} \boldsymbol{b}^{\max}$. Then, using (16), (20) becomes:

$$(\mathbf{E}_{\mathcal{I}} \tau \mathbf{D} + \mathbf{E}_{\mathcal{N}}) \boldsymbol{r} = -[(\mathbf{E}_{\mathcal{I}} \tau \mathbf{D} + \mathbf{E}_{\mathcal{N}}) \boldsymbol{y} + \boldsymbol{q}] \tag{21}$$

System (21) can be decomposed as follows:

$$\begin{bmatrix} \mathbf{I}_{\mathcal{N}\mathcal{N}} & \mathbf{0} \\ \tau \mathbf{D}_{\mathcal{I}\mathcal{N}} & \tau \mathbf{D}_{\mathcal{I}\mathcal{I}} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{\mathcal{N}} \\ \boldsymbol{r}_{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{y}_{\mathcal{N}} \\ \tau(\mathbf{D}_{\mathcal{I}\cdot} \boldsymbol{y} + \boldsymbol{q}_{\mathcal{I}}) \end{bmatrix} \tag{22}$$

Consequently, system (22) is equivalent to:

$$\boldsymbol{r}_{\mathcal{N}} = -\boldsymbol{y}_{\mathcal{N}} \tag{23}$$

$$\mathbf{D}_{\mathcal{I}\mathcal{I}} \boldsymbol{r}_{\mathcal{I}} = -(\mathbf{D}_{\mathcal{I}\mathcal{I}} \boldsymbol{y}_{\mathcal{I}} + \boldsymbol{q}_{\mathcal{I}}) \tag{24}$$

Since $\mathbf{D}$ is positive semidefinite, $\mathbf{D}_{\mathcal{I}\mathcal{I}}$ is also positive semidefinite, as a principal submatrix of $\mathbf{D}$. Therefore, in order to compute the Newton direction one has to solve a positive semidefinite system of order $|\mathcal{I}|$. This is particularly favorable for MPC, since the set of active constraints $\mathcal{I}$, is usually very small compared to $\mathcal{N}$.

The convergence properties of algorithm 1 are given in Theorem 1.

Theorem 1. Assume that $\boldsymbol{y}^\star \in \mathbb{R}^m$ is a solution of $\Phi_{\tau,\text{mid}}(\boldsymbol{y}) = 0$ and that any $\mathbf{H} \in \mathcal{A}_{\text{mid}}(\boldsymbol{y}^\star)$ is nonsingular. Then for any $\boldsymbol{y}^0$ sufficiently close to $\boldsymbol{y}^\star$, the Algorithm 1 is well defined and generates a sequence $\{\boldsymbol{y}^k\}$ that converges to $\boldsymbol{y}^\star$ Q - quadratically.

Its proof follows easily from Theorem 7.2.15 in Facchinei & Pang (2003b) and the fact that the Jacobians of the pieces of $\Phi_{\tau,\text{mid}}$, being affine, are globally Lipschitz.

5. Reformulation of MPC problem as unconstrained minimization of a convex quadratic spline

Algorithm 1 is a local Newton method in the sense that it converges only when started from an initial vector close to the solution. In order to globalize the Newton method, so as to converge from any starting point, the usual technique is to use a line-search on a merit function in order to appropriately scale the Newton direction (Dennis & Schnabel, 1996, Facchinei & Pang, 2003b). We define a merit function for the system of piecewise affine equations (13) as a function $\psi_\tau : \mathbb{R}^m \to \mathbb{R}$ such that $\arg\min \psi_\tau = \mathcal{Y}^\star$, i.e. the set of minimizers of ψ_τ equals the set of solutions to

$\Phi_{\tau,\text{mid}}(\boldsymbol{y}) = 0$. An obvious selection of a merit function would be $\psi_\tau(\boldsymbol{y}) = \frac{1}{2} || \Phi_{\tau,\text{mid}}(\boldsymbol{y}) ||^2$. However this function is neither smooth nor convex, and is difficult to rely on to perform a line-search.

It turns out that for (13) we can chose a merit function which is convex, continuously differentiable and piecewise quadratic, i.e. a convex quadratic spline (Li, Swetits, 1997). Hence, solving (13) is equivalent to the unconstrained minimization of a convex quadratic spline. By picking any τ such that $0 < \tau \leqslant 1/ || \mathbf{D} ||$, the matrix $\mathbf{I} - \tau\mathbf{D}$ is positive definite. Therefore (13) is equivalent to :

$$(\mathbf{I} - \tau\mathbf{D})\Phi_{\tau,\text{mid}}(\boldsymbol{y}) = 0 \tag{25}$$

Using the identity $\text{mid}(\sigma, \sigma'; z) = z + [\sigma - z]_+ - [z - \sigma']_+$, $\Phi_{\tau,\text{mid}}(\boldsymbol{y})$ can be expressed as:

$$\Phi_{\tau,\text{mid}}(\boldsymbol{y}) = \boldsymbol{y} + [\tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d} + \boldsymbol{b}^{\min}) - \boldsymbol{y}]_+ - [\boldsymbol{y} - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d} + \boldsymbol{b}^{\max})]_+ \tag{26}$$

Using (26)and the calculus rule $(\frac{1}{2} || [\boldsymbol{z}]_+ ||^2)' = [\boldsymbol{z}]_+$ one can easily infer that $(\mathbf{I} - \tau\mathbf{D})\Phi_{\tau,\text{mid}}(\boldsymbol{y})$ is the gradient of the following piecewise quadratic function:

$$\psi_\tau(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}'(\mathbf{I} - \tau\mathbf{D})\boldsymbol{y} - \frac{1}{2} || [\tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d} + \boldsymbol{b}^{\min}) - \boldsymbol{y}]_+ ||^2 - \frac{1}{2} || [\boldsymbol{y} - \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d} + \boldsymbol{b}^{\max})]_+ ||^2 \tag{27}$$

Obviously ψ_τ is continuously differentiable and piecewise quadratic. Furthermore, it is convex (Li, Swetits, 1997). Obviously, $\arg\min \psi_\tau = \mathcal{Y}^\star$ and since $\mathcal{Y}^\star$ is equal to the solution set of the dual of (5), one can compute a dual optimal solution of (5) by computing an unconstrained minimizer of ψ_τ .

6. Global Piecewise Smooth Newton Method with Exact Line Search

Another difficulty one may encounter when starting from a vector not sufficiently close to the solution is that the Newton system (20) may not be consistent. Even if it is consistent, it may have multiple solutions. However, one would want to use the Newton direction when (20) is consistent. For that purpose, Li & Swetits (1997, 1999) proposed the following notion of regularized solutions of linear systems:

Definition 2: Consider the linear system $\mathbf{H}\boldsymbol{r} = \boldsymbol{s}$, where $\mathbf{H} \in \mathbb{R}^{k\times m}$ and $\boldsymbol{s} \in \mathbb{R}^k$. Consider a subset $\mathcal{J}$ of $[1, k]$ such that $\mathbf{H}_{\mathcal{J}\mathcal{J}}$ is nonsingular and has the same rank with $\mathbf{H}$. Then the unique vector $\boldsymbol{r} \in \mathbb{R}^n$ such that $\boldsymbol{r}_{\mathcal{J}^c} = \boldsymbol{s}_{\mathcal{J}^c}$ and $\mathbf{H}_{\mathcal{J}\mathcal{J}}\boldsymbol{r}_{\mathcal{J}} = \boldsymbol{s}_{\mathcal{J}} - \mathbf{H}_{\mathcal{J}\mathcal{J}^c}\boldsymbol{r}_{\mathcal{J}^c}$ is called a regularized solution of $\mathbf{H}\boldsymbol{r} = \boldsymbol{s}$ corresponding to the index set $\mathcal{J}$.

Notice that when $\mathbf{H}\boldsymbol{r} = \boldsymbol{s}$ is consistent, then a regularized solution is actually a solution of the original system (Theorem 4.1, Li & Swetits, 1998). Having all tools necessary for a global version of the Piecewise Smooth Newton method (algorithm 1) in our disposal, we propose the following regularized Newton method with exact line search.

Algorithm 2: Regularized Piecewise Smooth Newton Method with Exact Line Search

Input : $\boldsymbol{y}^0 \in \mathbb{R}^m$

Step 1: Set $k = 0$

Step 2: If $|| \Phi_{\tau,\text{mid}}(\boldsymbol{y}^k) || \leqslant \varepsilon$ stop

Step 3: Select an element $\mathbf{H}^k \in \mathcal{A}_{\text{mid}}(\boldsymbol{y}^k)$. Find a regularized solution $\boldsymbol{r}^k \in \mathbb{R}^m$ of the linear system $\mathbf{H}^k\boldsymbol{r} = -\Phi_{\tau,\text{mid}}(\boldsymbol{y}^k)$

Step 4: Compute step-size $t_k = \min(\arg\min_t \psi_\tau(\boldsymbol{y}^k + t\boldsymbol{r}^k))$

Step 5: Set $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + t_k \boldsymbol{r}^k$ and $k \leftarrow k+1$. Go to step 2.

There are two differences between Algorithm 2 and Algorithm 4.1 in Li & Swetits (1997). The first is that in Algorithm 2 any element of the family of matrices $\mathcal{A}_{\text{mid}}$, which serves as the Newton approximation (see Facchinei & Pang, 2003b), is chosen to compute the Newton direction. Li & Swetits (1997) use $\mathbf{H}^k = \mathbf{H}(\mathcal{I}_{\text{Li}})$ for $\mathcal{I}_{\text{Li}} = \delta(\boldsymbol{y}) \bigcup \delta_{=}(\boldsymbol{y})$ in step 3. Notice that $\mathcal{I}_{\text{Li}} \in \mathscr{B}(\boldsymbol{y})$ and in fact $\mathcal{I}_{\text{Li}} \supseteq \mathcal{I}$ for any $\mathcal{I} \in \mathscr{B}(\boldsymbol{y})$. Therefore, our scheme allows more flexibility and perhaps less computations since the system in (24) is of order $|\mathcal{I}|$. The second difference is that we do not require the step-size to be smaller than one. However, since $\phi_\tau(t) \triangleq \psi_\tau(\boldsymbol{y}^k + t\boldsymbol{r}^k)$ may have many minimizers (in fact the interval of minimizers may not be upper-bounded) we select the smallest one. This is a technique similar to that proposed in Chen & Pinar (1998) for Huber's robust M-estimation problems. Allowing values larger than one for the step size can significantly accelerate the convergence of the algorithm.

As in the local Newton method, at iteration k, let $\mathcal{I}_k \in \mathscr{B}(\boldsymbol{y}^k)$ and $\mathcal{N}_k = [1,m] \setminus \mathcal{I}_k$. Then, one can compute a regularized solution of (20) solving the following simplified system:

$$\boldsymbol{r}_{\mathcal{N}_k} = -\boldsymbol{y}^k_{\mathcal{N}_k} \tag{27a)(28}$$

$$\boldsymbol{r}_{\mathcal{J}_k^c} = -(\mathbf{D}_{\mathcal{J}_k^c \mathcal{I}_k} \boldsymbol{y}^k_{\mathcal{I}_k} + \boldsymbol{q}_{\mathcal{J}_k^c}) \tag{27b)(29}$$

$$\mathbf{D}_{\mathcal{J}_k \mathcal{J}_k} \boldsymbol{r}_{\mathcal{J}_k} = -(\mathbf{D}_{\mathcal{J}_k \mathcal{I}_k} \boldsymbol{y}^k_{\mathcal{I}_k} + \boldsymbol{q}_{\mathcal{J}_k}) - \mathbf{D}_{\mathcal{J}_k \mathcal{J}_k^c} \boldsymbol{r}_{\mathcal{J}_k^c} \tag{27c)(30}$$

where $\mathcal{J}_k \subseteq \mathcal{I}_k$ is such that $\mathbf{D}_{\mathcal{J}_k \mathcal{J}_k}$ is nonsingular and has the same rank with $\mathbf{D}_{\mathcal{I}_k \mathcal{I}_k}$. It follows that $\boldsymbol{r}_{\mathcal{N}_k}$ and $\boldsymbol{r}_{\mathcal{J}_k^c}$ are trivially determined and one has to solve the linear system in (30) to obtain the Newton direction. However, since $\mathbf{D}_{\mathcal{I}_k \mathcal{I}_k}$ is always positive semidefinite and $\mathbf{D}_{\mathcal{J}_k \mathcal{J}_k}$ is a nonsingular principal submatrix of $\mathbf{D}_{\mathcal{I}_k \mathcal{I}_k}$, it follows that $\mathbf{D}_{\mathcal{J}_k \mathcal{J}_k}$ is positive definite. Therefore, $\boldsymbol{r}_{\mathcal{J}_k}$ can be obtained by computing the Cholesky factorization of the positive definite matrix $\mathbf{D}_{\mathcal{J}_k \mathcal{J}_k}$ and then solving the corresponding lower and upper triangular linear systems. In fact, the process of selecting the index set $\mathcal{J}_k \subseteq \mathcal{I}_k$ and computing the Cholesky factor of $\mathbf{D}_{\mathcal{J}_k \mathcal{J}_k}$ can be performed in one step, using the regularized Cholesky method (Li & Swetits, 1999).

Theorem 2 gives the favorable convergence properties of Algorithm 2. Its proof follows from previous works (Li & Swetits, 1997, Li & Swetits, 1999, Chen & Pinar, 1998, Facchinei & Pang, 2003).

Theorem 2.

(a) For every limit point $\boldsymbol{y}^\star$ of a sequence $\{\boldsymbol{y}^k\}$ generated by Algorithm 2 we have $\boldsymbol{y}^\star \in \mathcal{Y}^\star$.

(b) Algorithm 2 terminates in a finite number of iterations

(c) Suppose that any $\mathbf{H} \in \mathcal{A}_{\text{mid}}(\boldsymbol{y}^\star)$ is nonsingular and that $\{\boldsymbol{y}^k\}$ converges to $\boldsymbol{y}^\star$. Then $\{\boldsymbol{y}^k\}$ converges Q-quadratically to $\boldsymbol{y}^\star$.

Proof

(a) This follows from the fact the $\boldsymbol{r}^k$ is a descent direction (remark (iv) in Li & Swetits, 1999), Lemma 3.4 of Li & Swetits (1997) and the fact that any minimize of ψ_τ belongs to $\mathcal{Y}^\star$.

(b) This follows from theorem 4.4 of Chen & Pinar (1998) with some obvious modifications.

(c) This follows from the fact that unit step-size is eventually accepted in step 4 (see theorem 4.4 in Chen & Pinar, 1998) and theorem 1.

7. Implementation of Regularized Piecewise Smooth Newton Method with Exact Line Search

In this section, details on the implementation of the regularized piecewise smooth Newton method are presented. The two critical steps of the method are the computation the regularized Newton direction (step 3) and the line-search (step 4).

7a. Updating the Cholesky factor

Regarding step 3, instead of computing the Cholesky factor at iteration from scratch we can compute it by modifying the Cholesky factor obtained at the previous iteration, performing much less work. Let $\mathcal{J}_{k-1}$ denote the index set corresponding to the regularized solution of the Newton system (20) at iteration $k-1$ and let $\mathbf{L}$ denote the corresponding Cholesky factor, i.e. $\mathbf{D}_{\mathcal{J}_{k-1}\mathcal{J}_{k-1}} = \mathbf{L}\mathbf{L}'$ with $\mathbf{L}$ lower triangular. Our goal is to obtain the Cholesky factor of $\mathbf{D}_{\mathcal{J}_k\mathcal{J}_k}$, where $\mathcal{J}_k \subseteq \mathcal{I}_k$ is such that $\mathbf{D}_{\mathcal{J}_k\mathcal{J}_k}$ is positive definite and has the same rank with $\mathbf{D}_{\mathcal{I}_k\mathcal{I}_k}$. One can easily see that $\mathbf{D}_{\mathcal{I}_k\mathcal{I}_k}$ can be obtained from $\mathbf{D}_{\mathcal{J}_{k-1}\mathcal{J}_{k-1}}$ by adding and deleting rows and columns corresponding to indices belonging to $\mathcal{I}_k \setminus \mathcal{J}_{k-1}$ and $\mathcal{J}_{k-1} \setminus \mathcal{I}_k$, respectively. A similar procedure is proposed in Li, Nijs (2003) using LDL factorizations. Next, the procedures of modifying the Cholesky factor of a positive definite matrix when a row and column is added or deleted, are described.

Row and column addition: Consider a $k \times k$ symmetric positive definite matrix $\mathbf{D}$ and its Cholesky factor $\mathbf{L}$. Let $\bar{\mathbf{D}} = \begin{bmatrix} \mathbf{D} & \boldsymbol{c}_1 \\ \boldsymbol{c}_1' & c_2 \end{bmatrix}$, i.e. the $(k+1)\times(k+1)$ matrix obtained by appending the vector $\boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_1 \\ c_2 \end{bmatrix}$ and its transpose, where $\boldsymbol{c}_1 \in \mathbb{R}^k$ and $c_2 \in \mathbb{R}$, at the end of the columns and rows of $\mathbf{D}$. Notice that $\bar{\mathbf{D}}$ may not be positive definite. We will next show how one can obtain the Cholesky factor $\bar{\mathbf{L}}$ of $\bar{\mathbf{D}}$, if $\bar{\mathbf{D}}$ is positive definite, or conclude that $\bar{\mathbf{D}}$ is merely positive semidefinite.

We have:

$$\begin{bmatrix} \mathbf{L} & 0 \\ \boldsymbol{\ell}_1' & \ell_2 \end{bmatrix}\begin{bmatrix} \mathbf{L} & 0 \\ \boldsymbol{\ell}_1' & \ell_2 \end{bmatrix}' = \begin{bmatrix} \mathbf{D} & \boldsymbol{c}_1 \\ \boldsymbol{c}_1' & c_2 \end{bmatrix} \tag{28}$$

from which we obtain:

$$\mathbf{L}\boldsymbol{\ell}_1 = \boldsymbol{c}_1 \tag{29a}$$

$$\ell_2^2 = c_2 - \boldsymbol{\ell}_1'\boldsymbol{\ell}_1 \tag{29b}$$

Therefore, $\bar{\mathbf{D}}$ is positive definite if and only if $c_2 - \boldsymbol{\ell}_1'\boldsymbol{\ell}_1$ is positive. Summing up, we obtain the following algorithm for concluding if $\bar{\mathbf{D}}$ is positive definite and if so, for obtaining its Cholesky factor from that of $\mathbf{D}$.

Algorithm 3: $[\bar{\mathbf{L}}, p] = \text{cholAdd}(\mathbf{L}, \boldsymbol{c})$ **Row and column addition**

Input : Cholesky factor of $\mathbf{D} \in \mathbb{R}^{k\times k}$, $\mathbf{L}$, vector $\boldsymbol{c} \in \mathbb{R}^{(k+1)\times 1}$

Output : Cholesky factor $\bar{\mathbf{L}}$ of $\bar{\mathbf{D}} = \begin{bmatrix} \mathbf{D} & \boldsymbol{c}_{1:k} \\ \boldsymbol{c}'_{1:k} & c_{k+1} \end{bmatrix}$ if $\bar{\mathbf{D}}$ is positive definite.

Step 1 : Solve lower triangular system $\mathbf{L}\boldsymbol{\ell}_1 = \boldsymbol{c}_{1:k}$ for $\boldsymbol{\ell}_1$

Step 2 : $d = \boldsymbol{c}_k - \boldsymbol{\ell}'_1\boldsymbol{\ell}_1$

Step 3 : **if** $d > 0$ **then** $\ell_2 = \sqrt{d}$ **,** $\bar{\mathbf{L}} = \begin{bmatrix} \mathbf{L} & 0 \\ \boldsymbol{\ell}'_1 & \ell_2 \end{bmatrix}$**, otherwise** $\bar{\mathbf{D}}$ is not positive definite.

Step 1 is the solution of a lower triangular system by forward substitution and requires k^2 flops. Step 2 requires additional $3k-1$ flops, while step 3 requires 1 flop in the case of $\bar{\mathbf{D}}$ being positive definite. Consequently, the total number of flops required by algorithm 1 is $k^2 + 3k$.

Row and column deletion: Suppose that we have the Cholesky factor of the $k \times k$ symmetric positive definite matrix $\mathbf{D}$ and we want to compute the Cholesky factor of the matrix $\bar{\mathbf{D}}$ obtained from $\mathbf{D}$ by deleting its j-th column and j-th row. Denote by $\boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}'_{12} & c_{22} & \boldsymbol{c}'_{32} \end{bmatrix}'$ the j-th column of $\mathbf{D}$. Since $\mathbf{D}$ is symmetric, its j-th row is $\boldsymbol{c}'$. Also let $\mathbf{D}_{11} = \mathbf{D}_{1:j-1,1:j-1}$, $\mathbf{D}_{31} = \mathbf{D}_{j+1:k,1:j-1}$ and $\mathbf{D}_{33} = \mathbf{D}_{j+1:k,j+1:k}$.

Before deleting row and column j we have the original factorization:

$$\mathbf{L}\mathbf{L}' = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\ell}'_{12} & \ell_{22} & \mathbf{0} \\ \mathbf{L}_{31} & \boldsymbol{\ell}_{32} & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{L}'_{11} & \boldsymbol{\ell}_{12} & \mathbf{L}'_{31} \\ \mathbf{0} & \ell_{22} & \boldsymbol{\ell}'_{32} \\ \mathbf{0} & \mathbf{0} & \mathbf{L}'_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{D}_{11} & \boldsymbol{c}_{12} & \mathbf{D}'_{31} \\ \boldsymbol{c}'_{12} & c_{22} & \boldsymbol{c}'_{32} \\ \mathbf{D}_{31} & \boldsymbol{c}_{32} & \mathbf{D}_{33} \end{bmatrix} \tag{30)(31}$$

After deleting row and column j, we have:

$$\bar{\mathbf{L}}\bar{\mathbf{L}}' = \begin{bmatrix} \bar{\mathbf{L}}_{11} & \mathbf{0} \\ \bar{\mathbf{L}}_{31} & \bar{\mathbf{L}}_{33} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{L}}'_{11} & \bar{\mathbf{L}}'_{31} \\ \mathbf{0} & \bar{\bar{\mathbf{L}}}'_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}'_{31} \\ \mathbf{D}_{31} & \mathbf{D}_{33} \end{bmatrix} \tag{31)(32}$$

Combining (30) and (31) we obtain:

$$\bar{\mathbf{L}}_{11} = \mathbf{L}_{11} \tag{32a)(33}$$

$$\bar{\mathbf{L}}_{31} = \mathbf{L}_{31} \tag{32b)(34}$$

$$\bar{\mathbf{L}}_{33}\bar{\mathbf{L}}'_{33} = \mathbf{L}_{33}\mathbf{L}'_{33} + \boldsymbol{\ell}_{32}\boldsymbol{\ell}'_{32} \tag{32c)(35}$$

From (35) we observe that $\bar{\mathbf{L}}_{33}$ is the Cholesky factor of $\mathbf{S} + \boldsymbol{\ell}_{32}\boldsymbol{\ell}'_{32}$, where $\mathbf{S}$ is the matrix whose Cholesky factor is $\mathbf{L}_{33}$. Therefore, $\bar{\mathbf{L}}_{33}$ can be computed by a standard rank-1 update (Dongarra, Bunch, Moler & Stewart, 1979) of

$\mathbf{L}_{33}$. Consequently we obtain the following algorithm for modifying the Cholesky factor of a matrix whose j-th column and j-th row are deleted.

Algorithm 4: $\bar{\mathbf{L}} = \text{cholDelete}(\mathbf{L}, j)$ **Row and column deletion**

Input : Cholesky factor of $\mathbf{D} \in \mathbb{R}^{k\times k}$, $\mathbf{L}$, index j to delete

Output : Cholesky factor $\bar{\mathbf{L}}$ of $\bar{\mathbf{D}}$ obtained from $\mathbf{D}$ by deleting its j-th row and column

Step 1 : $\bar{\mathbf{L}}_{1:j-1,1:j-1} = \mathbf{L}_{1:j-1,1:j-1}$, $\bar{\mathbf{L}}_{j:k-1,1:j-1} = \mathbf{L}_{j+1:k,1:j-1}$

Step 2 : Perform rank-1 update: $\bar{\mathbf{L}}_{j:k-1,j:k-1}\bar{\mathbf{L}}'_{j:k-1,j:k-1} = \mathbf{L}_{j+1:k,j+1:k}\mathbf{L}'_{j+1:k,j+1:k} + \mathbf{L}_{j+1:k,j}\mathbf{L}'_{j+1:k,j}$

The total number of flops required by algorithm 2 is equal to the number of flops required to perform the rank-1 update, which is $2(k-j)^2 + 4(k-j)$ using method C1 of Gill, Golub, Murray & Saunders (1974), see also Davis & Hager (1999).

Returning to step 3 of the regularized piecewise smooth Newton method, an index set $\mathcal{J}_k \subseteq \mathcal{I}_k$ such that $\mathbf{D}_{\mathcal{J}_k\mathcal{J}_k}$ is positive definite and has the same rank with $\mathbf{D}_{\mathcal{I}_k\mathcal{I}_k}$ and the Cholesky factor of $\mathbf{D}_{\mathcal{J}_k\mathcal{J}_k}$, can be obtained by the following procedure.

Algorithm 5: $[\mathbf{L}, \mathcal{J}_k] = \text{updateCholesky}(\mathbf{L}, \mathbf{D}, \mathcal{I}_k, \mathcal{J}_{k-1})$ **Cholesky Factor updating**

Step 1: $\tau_1 = \mathcal{J}_{k-1} \setminus \mathcal{I}_k, \tau_2 = \mathcal{I}_k \setminus \mathcal{J}_{k-1}$, $\mathcal{J}_k = \mathcal{I}_k \bigcap \mathcal{J}_{k-1}$, $n_\tau = |\tau_1|$

Step 2: while $n_\tau > 0$, do $\mathbf{L} \leftarrow \text{cholDelete}(\mathbf{L}, \tau(n_\tau))$, $\tau_1 \leftarrow \tau_1 \setminus \tau_1(n_\tau)$, $n_\tau \leftarrow n_\tau - 1$ end while

Step 3: for j in τ_2, $\mathcal{J} = \mathcal{J}_k \bigcup \{j\}$, do $[\mathbf{L}, p] \leftarrow \text{cholAdd}(\mathbf{L}, \mathbf{D}_{\mathcal{J}_\tau j})$. If $p = 1$ then $\mathcal{J}_k \leftarrow \mathcal{J}$, end for

---

7b. Line Search

The line search procedure in step 4 of algorithm 2 looks for the smallest step-size that minimizes the univariate function:

$$\theta(t) = \psi_\tau(\boldsymbol{y} + t\boldsymbol{r}) \tag{33)(36}$$

Function θ is continuously differentiable as a composition of continuously differentiable functions, and convex piecewise quadratic as the composition of a convex piecewise quadratic function with an affine function (Exercise 10.22 (b), Rockafellar & Wets, 2009). Also, θ is bounded below since ψ_τ is bounded below. Therefore, there exists a finite $\bar{t}$ such that $\theta(\bar{t}) = 0$. Furthermore, since $\boldsymbol{r}$ is a descent direction (see theorem 2) it follows that $\theta'(0) < 0$ and since θ' is nondecreasing it follows that $\bar{t} > 0$.

Next, notice that θ' can be expressed as:

$$\theta'(t) = \eta t + \beta - \boldsymbol{\delta}'[\boldsymbol{\delta}t - \boldsymbol{\gamma}]_+ \tag{34)(37}$$

where $\eta = (\boldsymbol{r} - \hat{\boldsymbol{r}})'\boldsymbol{r}$, $\beta = (\boldsymbol{r} - \hat{\boldsymbol{r}})'\boldsymbol{y}$, $\boldsymbol{\delta} = \begin{bmatrix} (\boldsymbol{r} - \hat{\boldsymbol{r}}) \\ -(\boldsymbol{r} - \hat{\boldsymbol{r}}) \end{bmatrix}, \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{y} - \hat{\boldsymbol{y}} - \hat{\boldsymbol{b}}^{\min} \\ \hat{\boldsymbol{b}}^{\max} - (\boldsymbol{y} - \hat{\boldsymbol{y}}) \end{bmatrix}$, $\hat{\boldsymbol{y}} = \tau(\mathbf{D}\boldsymbol{y} + \boldsymbol{d})$ and $\hat{\boldsymbol{r}} = \tau\mathbf{D}\boldsymbol{r}$

Notice that θ' is a piecewise affine function with breakpoints $s_j = \gamma_j / \delta_j$, $j \in [1, 2m]$. The following algorithm finds the smallest $\bar{t}$ such that $\theta'(\bar{t}) = 0$ in a finite number of operations, where we have we assumed that the positive breakpoints of θ' have been sorted in ascending order (see also Madsen & Nielsen, 1990).

Algorithm 6: $\bar{t} \leftarrow \mathrm{LineSearch}(\eta, \beta, \delta, \gamma)$ **Exact PWA line search**

Step 1: $\mathcal{P} = \{j \in [1, 2m] \mid \delta_j > 0\}$, $\mathcal{L} = \{j \in [1, 2m] \mid s_j > 0\}$, $n_{\mathcal{L}} = \mid \mathcal{L} \mid$, $k = 0$

Step 2: $a = \eta + \sum_{j \in \mathcal{J}} \delta_j^2$, $b = \beta - \sum_{j \in \mathcal{J}} \delta_j \gamma_j$ where $\mathcal{J} = (\mathcal{P} \setminus \mathcal{L}) \bigcup (\mathcal{L} \setminus \mathcal{P})$

Step 3: for $i \in \mathcal{L}$

Step 4: $k \leftarrow k + 1$

Step 5: if $as_i + b \geqslant 0$ then $\bar{t} = -b / a$ stop

Step 6: if $i \in \mathcal{P}$ then $a \leftarrow a - \delta_i^2$, $b \leftarrow b + \delta_i \gamma_i$, else if $k < n_{\mathcal{L}}$ then $a \leftarrow a + \delta_i^2$, $b \leftarrow b - \delta_i \gamma_i$

Step 7: end for

Step 8: $\bar{t} = -b / a$

Algorithm 6 starts from the smallest positive breakpoint and successively updates the slope and intercept of the affine expression of θ' valid on the interval $(s_{i-1}, s_i]$. If $\theta'(s_i) \geqslant 0$ (step 5) then there exists zero of θ' that coincides with the zero of the affine expression on that interval. Otherwise, the algorithm proceeds by computing the affine expression in the next interval (step 6). Furthermore, since the search is started from the smallest positive breakpoint the algorithm is guaranteed to find the left-most $\bar{t}$ such that $\theta'(\bar{t}) = 0$. The slope and intercept are updated in step 6 using the following reasoning: For any $t \in \mathbb{R}$ let $\mathcal{J}_1(t) = \{j \in [1, 2m] \mid t > s_j, \delta_j > 0\}$ and $\mathcal{J}_2(t) = \{j \in [1, 2m] \mid t \leqslant s_j, \delta_j < 0\}$. Then:

$$\theta'(t) = \eta t + \beta - \sum_{j=1}^{2m} \delta_j (\delta_j t - \gamma_j)_+ = \eta t + \beta - \sum_{\{j \in [1,2m] : \delta_j t - \gamma_j \geqslant 0\}} \delta_j (\delta_j t - \gamma_j) = (\eta - \sum_{j \in \mathcal{J}_1(t) \cup \mathcal{J}_2(t)} \delta_j^2) t + \beta + \sum_{j \in \mathcal{J}_1(t) \cup \mathcal{J}_2(t)} \delta_j \gamma_j$$

Suppose that the breakpoints are sorted in ascending order, i.e. $s_1 \leqslant s_2 \leqslant ... \leqslant s_{2m}$. Then:

$$\mathcal{J}_1(s_i) = \{j \in [1, i-1] \mid \delta_j > 0\} \text{ and } \mathcal{J}_2(s_i) = \{j \in [i, 2m] \mid \delta_j < 0\}$$

For any $t \in (s_{i-1}, s_i]$ we have $\mathcal{J}_1(t) = \mathcal{J}_1(s_i)$ and $\mathcal{J}_2(t) = \mathcal{J}_2(s_i)$. Hence $\theta'(t) = a_i t + b_i$ for $t \in (s_{i-1}, s_i]$, with $a_i = \eta - \sum_{j \in \mathcal{J}_1(s_i) \cup \mathcal{J}_2(s_i)} \delta_j^2$, $b_i = \beta + \sum_{j \in \mathcal{J}_1(s_i) \cup \mathcal{J}_2(s_i)} \delta_j \gamma_j$

Next, consider s_{i+1}. Then $\mathcal{J}_1(s_{i+1}) = \{j \in [1, i] \mid \delta_j > 0\}$ and $\mathcal{J}_2(s_{i+1}) = \{j \in [i+1, 2m] \mid \delta_j < 0\}$

Therefore $\mathcal{J}_1(s_{i+1}) = \begin{cases} \mathcal{J}_1(s_i) \bigcup \{i\}, & \delta_i > 0 \\ \mathcal{J}_1(s_i) & , \text{ otherwise} \end{cases}$ and $\mathcal{J}_2(s_{i+1}) = \begin{cases} \mathcal{J}_2(s_i) \setminus \{i\}, & \delta_i < 0 \\ \mathcal{J}_2(s_i), & \text{otherwise} \end{cases}$

Thus, we obtain the following updating rule for the slope and the intercept of the piecewise affine function θ':

$$a_{i+1} = \begin{cases} a_i - \delta_i^2 \ , & \text{if } \delta_i > 0 \\ a_i + \delta_i^2 \ , & \text{if } \delta_i < 0 \end{cases} \text{ and } b_{i+1} = \begin{cases} b_i + \delta_i \gamma_i \ , & \text{if } \delta_i > 0 \\ b_i - \delta_i \gamma_i \ , & \text{if } \delta_i < 0 \end{cases}$$

It remains to examine what happens for $t > s_{2m}$. In that case, only $\mathcal{J}_1(t)$ can change:

$$a_{2m+1} = \begin{cases} a_{2m} - \delta_{2m}^2, & \text{if } \delta_i > 0 \\ a_{2m} & , \text{ if } \delta_i < 0 \end{cases} \text{ and } b_{2m+1} = \begin{cases} b_{2m} + \delta_{2m} \gamma_{2m}, & \text{if } \delta_i > 0 \\ b_{2m} & , \text{ if } \delta_i < 0 \end{cases}$$

8. Detailed implementation for MPC

In this section the detailed implementation of the regularized piecewise smooth Newton method for constrained linear MPC is presented. Offline the following matrices are calculated: $\mathbf{D} \equiv \mathbf{G}\mathbf{M}^{-1}\mathbf{G}'$, $\hat{\mathbf{D}} = \tau\mathbf{D}$, $\hat{\mathbf{S}} = \tau\mathbf{G}\mathbf{M}^{-1}\mathbf{C}$, $\hat{\mathbf{B}} = \tau\tilde{\mathbf{F}}_{\boldsymbol{x}} \tilde{\mathbf{A}}$, $\hat{\boldsymbol{f}}^{\min} = \tau\tilde{\boldsymbol{f}}_{\min}$, $\hat{\boldsymbol{f}}^{\max} = \tau\tilde{\boldsymbol{f}}_{\max}$, $\mathbf{K}_{\boldsymbol{x}} = -\mathbf{M}^{-1}_{1:nu,:}\mathbf{G}'$, $\mathbf{K}_{\boldsymbol{y}} = -\mathbf{M}^{-1}_{1:nu,:}\mathbf{C}$, where $0 < \tau \leqslant 1/\|\mathbf{D}\|$. At each time instant, after the system state becomes available the following algorithm (MPCNewton) is applied. MPCNewton takes as input arguments the current state $\boldsymbol{x}(t)$, the vector $\hat{\boldsymbol{d}} = \hat{\mathbf{S}}\boldsymbol{x}(t-1)$, the dual solution vector of the previous time step denoted by $\boldsymbol{y}$, the vector $\hat{\boldsymbol{y}} = \hat{\mathbf{D}}\boldsymbol{y} + \hat{\boldsymbol{d}}$ and the Cholesky factor and index set $\mathcal{J}$ corresponding to the regularized solution of the previous time step. The output arguments of MPCNewton are the MPC control law $\kappa_{\text{MPC}}(\boldsymbol{x}(t))$, and the updated versions of $\hat{\boldsymbol{d}}, \boldsymbol{y}, \hat{\boldsymbol{y}}, \mathbf{L}, \mathcal{J}$.

Algorithm 7: $[\kappa_{\text{MPC}}(\boldsymbol{x}(t)), \hat{\boldsymbol{d}}, \boldsymbol{y}, \hat{\boldsymbol{y}}, \mathbf{L}, \mathcal{J}]$=MPCNewton$(\boldsymbol{x}(t), \hat{\boldsymbol{d}}, \boldsymbol{y}, \hat{\boldsymbol{y}}, \mathbf{L}, \mathcal{J})$

Step 1 : $\hat{\boldsymbol{g}} \leftarrow \hat{\mathbf{S}}\boldsymbol{x}(t)$, $\hat{\boldsymbol{b}}^{\min} \leftarrow \hat{\boldsymbol{f}}^{\min} - \hat{\mathbf{B}}\boldsymbol{x}(t)$, $\hat{\boldsymbol{b}}^{\max} \leftarrow \hat{\boldsymbol{f}}^{\max} - \hat{\mathbf{B}}\boldsymbol{x}(t)$

Step 2 : $\hat{\boldsymbol{y}} \leftarrow \hat{\boldsymbol{y}} + \hat{\boldsymbol{g}} - \hat{\boldsymbol{d}}$

Step 3: $\boldsymbol{e} \leftarrow \boldsymbol{y} - \hat{\boldsymbol{y}}$, $\Phi_{\text{mid}} \leftarrow \hat{\boldsymbol{y}} + \text{mid}(\hat{\boldsymbol{b}}_{\min}, \hat{\boldsymbol{b}}_{\max}; \boldsymbol{e})$

Step 4 : **if** $||\ \Phi_{\text{mid}}\ || \leqslant \varepsilon$ then calculate control law $\kappa_{\text{MPC}}(\boldsymbol{x}(t)) \leftarrow \mathbf{K}_{\boldsymbol{x}}\boldsymbol{x}(t) + \mathbf{K}_{\boldsymbol{y}}\boldsymbol{y}$, $\hat{\boldsymbol{d}} \leftarrow \hat{\boldsymbol{g}}$ and exit.

Step 5 : $\boldsymbol{q}_{\beta_<(\boldsymbol{y}) \cup \beta_=(\boldsymbol{y})} \leftarrow \hat{\boldsymbol{g}}_{\beta_<(\boldsymbol{y}) \cup \beta_=(\boldsymbol{y})} + \hat{\boldsymbol{b}}^{\min}_{\beta_<(\boldsymbol{y}) \cup \beta_=(\boldsymbol{y})}$, $\boldsymbol{q}_{\gamma_>(\boldsymbol{y}) \cup \gamma_=(\boldsymbol{y})} \leftarrow \hat{\boldsymbol{g}}_{\gamma_>(\boldsymbol{y}) \cup \gamma_=(\boldsymbol{y})} + \hat{\boldsymbol{b}}^{\min}_{\gamma_>(\boldsymbol{y}) \cup \gamma_=(\boldsymbol{y})}$. Choose $\mathcal{I} \in \mathscr{B}(\boldsymbol{y})$,
$\mathcal{N} \leftarrow [1, m] \setminus \mathcal{I}$, $\boldsymbol{r}_{\mathcal{N}} \leftarrow -\boldsymbol{y}_{\mathcal{N}}$

Step 6 : $(\mathbf{L}, \mathcal{J}) \leftarrow \text{updateCholesky}(\hat{\mathbf{D}}, \mathbf{L}, \mathcal{I}, \mathcal{J})$

Step 7 : **if** $\mathcal{J} = \mathcal{I}$ **then**

Solve $\mathbf{L}\boldsymbol{z} = -\boldsymbol{q}_{\mathcal{I}}$ for $\boldsymbol{z}$. Solve $\mathbf{L}'\boldsymbol{s} = \boldsymbol{z}$ for $\boldsymbol{s}$. $\boldsymbol{r}_{\mathcal{I}} \leftarrow \boldsymbol{s} - \boldsymbol{y}_{\mathcal{I}}$

else,

$$\boldsymbol{r}_{\mathcal{J}^c} \leftarrow -(\mathbf{D}_{\mathcal{J}^c\mathcal{I}}\boldsymbol{y}_{\mathcal{I}} + \boldsymbol{q}_{\mathcal{J}^c})\,.\ \text{Solve}\ \mathbf{L}\boldsymbol{z} = -(\mathbf{D}_{\mathcal{J}\mathcal{I}}\boldsymbol{y}_{\mathcal{I}} + \boldsymbol{q}_{\mathcal{J}}) - \mathbf{D}_{\mathcal{J}\mathcal{J}^c}\boldsymbol{r}_{\mathcal{J}^c}\ \text{for}\ \boldsymbol{z}\,.\ \text{Solve}\ \mathbf{L}'\boldsymbol{r}_{\mathcal{I}} = \boldsymbol{z}\ \text{for}\ \boldsymbol{r}_{\mathcal{I}}$$

Step 8 : $\hat{\boldsymbol{r}} \leftarrow \hat{\mathbf{D}}\boldsymbol{r}\,,\ \boldsymbol{z} \leftarrow \boldsymbol{r} - \hat{\boldsymbol{r}}\,,\ \eta \leftarrow \boldsymbol{z}'\boldsymbol{r}\,, \beta \leftarrow \boldsymbol{z}'\boldsymbol{y}\,, \gamma \leftarrow \begin{bmatrix} \boldsymbol{e} - \hat{\boldsymbol{b}}_{\min} \\ \hat{\boldsymbol{b}}_{\max} - \boldsymbol{e} \end{bmatrix}, \delta \leftarrow \begin{bmatrix} \boldsymbol{z} \\ -\boldsymbol{z} \end{bmatrix}\,,\ \bar{t} \leftarrow \text{LineSearch}(\eta, \beta, \delta, \gamma)$

Step 9 : $\boldsymbol{y} \leftarrow \boldsymbol{y} + \bar{t}\boldsymbol{r}\,,\ \hat{\boldsymbol{y}} \leftarrow \hat{\boldsymbol{y}} + \bar{t}\hat{\boldsymbol{r}}$ Go to step 3

Regarding the complexity of the algorithm, step 6 requires approximately $O(|\mathcal{I}|^2)$ flops, step 7 requires $O(|\mathcal{J}|^2)$ flops while step 8 requires $O(m^2)$ operations at worst case. However, we have observed that the line search usually requires very few operations in practice. Step 1 builds the appropriate problem matrices in $O(mn_x)$ operations and is executed only once. In step 3 the MPC control algorithm is calculated upon convergence of the algorithm and can be performed in $O(n_u(n_x + |\mathcal{J}|))$ operations. All other steps require only vector operations of order at most $O(m)$. Notice that the Cholesky factor $\mathbf{L}$ and index set $\mathcal{J}$ from the previous run of MPCNewton are passed as input arguments, so there is no need to compute the Cholesky factor from scratch, not even at the first iteration. Specifically, we have observed that this provides considerable computational savings provided that the active sets between two consecutive time steps do not differ too much.

9. Comparison with existing approaches

Compared to active set and interior point algorithms, the regularized piecewise smooth Newton method with exact line search has some considerable advantages when applied to MPC. First, at each iteration, the simplified positive semidefinite linear system (24) is solved. The dimension of the system is equal to the cardinality of the set of active constraints which is usually considerably smaller than the total number of constraints, especially for MPC problems. On the other hand, active set algorithms solve an indefinite system of linear equations with dimension equal to the number of variables plus the number of active constraints. As far as interior point methods are concerned, the system of linear equations solved at each iteration involves both primal and dual variables but has favorable sparsity pattern for MPC problems (Rao *et al.*, 1998, Wang & Boyd, 2010).

The regularized piecewise smooth Newton method can take larger steps to optimality by performing exact line search on the piecewise quadratic merit function. On the other hand, the step-size in active set methods is dictated by the need to maintain primal or dual feasibility, while in interior point methods, the step-size is selected so as to maintain both primal and dual feasibility. Therefore, step-sizes are usually larger in the piecewise smooth Newton method, leading to larger moves towards optimality.

The other advantage of great importance is that the regularized Newton method, unlike interior point and active set methods, does not require a feasible starting point. Specifically, our experience from numerous simulations on MPC problems suggests that a very good starting point is the dual solution obtained at the previous step of the MPC.

10. Examples

A MATLAB implementation of the proposed algorithm (about 100 lines of code) was tested against state-of-the art QP solvers in MPC problems for systems of various dimensions and prediction horizons: The primal-dual interior point solver, BPMPD (Mészáros, 1999, http://www.sztaki.hu/~meszaros/bpmpd/), the dual active set solver QPC (Wills, 2009, http://sigpromu.org/quadprog/), and the online active-set solver qpOASES (Ferreau *et al.*, 2008, http://www.kuleuven.be/optec/software/qpOASES). The first two are programmed in C, while the third is programmed is C++. BPMPD is based on Mehrotra's predictor-corrector algorithm and it was found to be the fastest QP solver in the recent benchmark (http://plato.asu.edu/ftp/qpbench.html). For the interior-point solver the MPC problem is formulated as in Rao *et al.* (1998), Wang & Boyd, 2010). That is, the equality constraints corresponding to the state-update equations are not eliminated. BPMPD is an advanced solver that can identify the favorable

sparsity pattern of the problem, therefore we expect the computational effort to increase linearly in terms of the prediction horizon. For the two active set solvers, the MPC problem was formulated as in (4). The online active set strategy, qpOASES, is able to handle inequality constraints with upper and lower bounds. For the dual active set solver, QPC, the constraints had to be converted to upper bounded inequalities. Since this increases the dimension of the dual vector, we expect some extra overhead for QPC. In all the examples, we choose $\tau = 1/(1.01\|\mathbf{D}\|)$ for the MPCNewton method. All simulations were run on an Intel Core 2 Quad CPU Q9400 at 2.66GHz with 4 GB RAM, running Debian Linux.

For the active set solvers we expect the running time to increase quadratically with respect to the prediction horizon. Surprisingly, numerous simulations have provided the evidence that for MPCNewton, the increase in computational complexity is at most linear with respect to the prediction horizon. This result is not a consequence of the special block tridiagonal structure of MPC problems, since MPCNewton uses formulation (4) as an implicit active set method. However, since in MPC problems only a fraction of state and control variables are saturated throughout the prediction horizon and since MPCNewton always solves systems of linear equations with dimension equal to the active set, we obtain this favorable property. In fact, as it will be observed in the examples, the dimension of the system and the prediction horizon affect only slightly the running time of MPCNewton. This is a big advantage of MPCNewton, compared to any other QP algorithm because it allows the application of MPC for large-scale systems with very high sampling frequency.

A. Oscillating masses

This first example is taken from Wang & Boyd (2010). It consists of a sequence of six masses connected by springs to each other, and to walls on either side. There are three actuators, which exert tensions between different masses. For this example $\boldsymbol{x} \in \mathbb{R}^{12}$, $\boldsymbol{u} \in \mathbb{R}^3$ and we also assume that a random force drawn from a uniform distribution on $[-0.5, 0.5]$ acts on each mass as a disturbance. For a detailed description of the problem, we refer to Wang & Boyd (2010). The purpose of this example is not only to compare the various solvers in terms of CPU times but also to observe how their runtimes scale with the increase on the prediction horizon. Therefore, we let the prediction horizon to range between 10 to 100 with increment 10. For each prediction horizon, 10 simulations of 100 steps were run, starting from different random initial states and using different random sequences of disturbances every time. In the comparison, we also present results for the approximate interior-point barrier solver (fastMPC) of Wang & Boyd (2010, http://www.stanford.edu/~boyd/fast_mpc/). For fastMPC, we used a fixed barrier parameter of value 10^{-2}. Figure 1, depicts the dependence of the runtime for each algorithm in terms of the prediction horizon. As expected, the runtime of the interior point solvers (BPMPD and fastMPC) grows linearly with the prediction horizon, while that of the active-set solvers (QPC and qpOASES) grows quadratically. However, we can observe that the runtime of MPCNewton is of order $O(\sqrt{N})$.

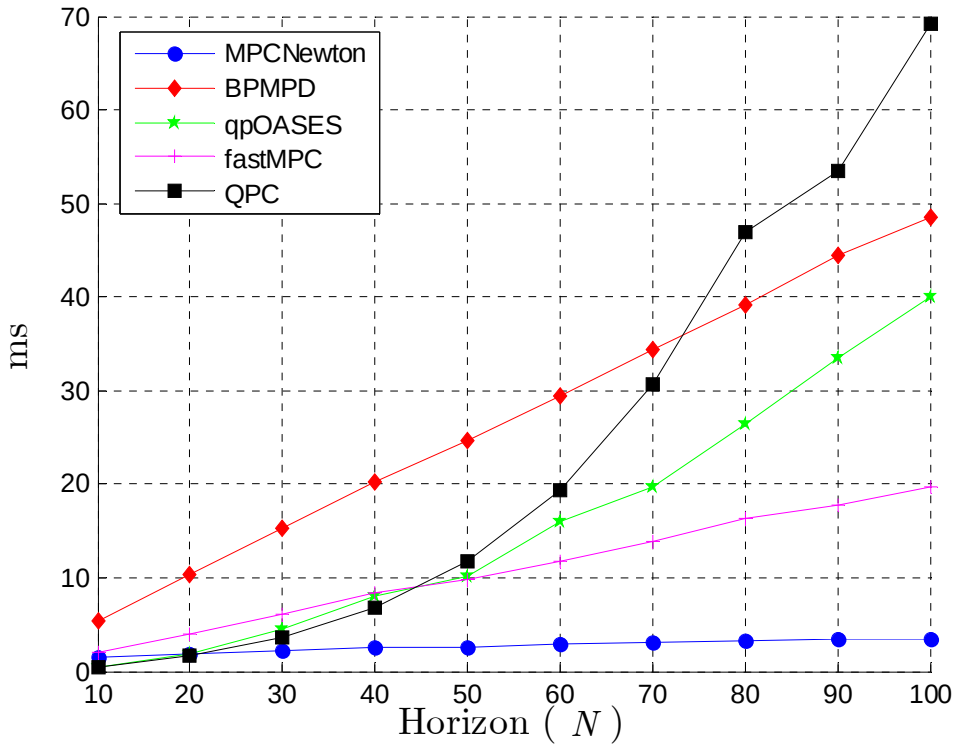

Figure 1. Runtimes with respect to prediction horizon for oscillating masses example

B. Crude Distillation Unit

The next example is the crude distillation unit model of Pannocchia *et al.*, (2006) with $\boldsymbol{x} \in \mathbb{R}^{252}$, $\boldsymbol{u} \in \mathbb{R}^{32}$, 90 outputs and bound constraints on the inputs only. The data for this example are taken from the online QP benchmark collection (http://www.kuleuven.be/optec/software/onlineQP/). Since the data is in the form of the matrices of problem (4) we were not able to compare with the interior point solvers. Average and maximum runtimes and number of iterations for MCNewton, QPC and qpOASES are presented in table 1.

	Runtime (ms)		Number of iterations	
	Average	Maximum	Average	Maximum
MPCNewton	1.75	80	1.17	5
qpOASES	125	266	3.92	313
QPC	853	2599	75.39	266

Table 1. Runtime results for the Crude Distillation Unit example

For this example, there are 7201 simulation steps, and 5 set-point changes are performed causing large changes for the active sets between consecutive steps. Although on average, the runtime of qpOASES is comparably small, at those changes it needs to perform a large number of iterations in order to move from one active set to another. For example, for the maximum number of changes in the active set (307), qpOASES performfs 313 iterations in 8.12 seconds, while QPC performs 252 iterations in 2.48 seconds. At the sime time, MPCNewton needs only 4 iterations

to identify this large change in the active set, at only 80 milliseconds. We mention that the worst runtime observed in Pannocchia *et al.*, (2006) for their partial enumeration method is 1.3 seconds.

C. Chain of Masses

The example is taken from http://www.kuleuven.be/optec/software/onlineQP/, as well. The goal is to regulate a chain of nine masses connected by springs to a certain steady state. One end of the chain is fixed on a wall while the three velocity components of the other end are used as control input. The system has 57 states, 3 inputs while the prediction horizon of the MPC controller is 80. Furthermore, the system is subject to input bound constraints and state constraints that ensure that the chain does not hit a vertical wall close to the steady state. Table 2 presents the results for this example.

	Runtime (ms)		Number of iterations	
	Average	Maximum	Average	Maximum
MPCNewton	2.64	24.19	2.43	23
qpOASES	7.09	36.26	2.63	16
QPC	24.30	80.82	11.53	61

Table 2. Runtime results for the Chain of masses example

This example is well suited for qpOASES, since only small changes to the active set occur. However, the average number of iterations performed by MPCNewton and qpOASES is almost the same. Furthermore, the runtime of MPCNewton is smaller due to the simplified structure of the system of linear equations solved at each iteration.

D. Randomly generated systems

The last numerical example consists of randomly generated systems of various dimensions. The entries of system matrices, $\mathbf{A}$ and $\mathbf{B}$ are taken from a normal distribution with zero mean and unit variance. Matrix $\mathbf{A}$ is then scaled, so the system is neutrally stable. Matrices $\mathbf{Q}$ and $\mathbf{R}$ are random positive definite matrices. White noise is added to the system state during the simulations. The input variables are constrained to lie in $[-0.2, 0.2]$ while state variables must lie in $[-10, 10]$. The terminal cost matrix is taken to be the solution of the algebraic Riccati equation (ARE) $\mathbf{P}_f = \mathbf{A}'\mathbf{P}_f\mathbf{A} + \mathbf{Q} - \mathbf{K}'\mathbf{R}\mathbf{K}$ and $\mathcal{X}_f = \mathbb{R}^{n_x}$. Table 3 presents the running times of the QP algorithms for the various systems and prediction horizons, while figure 2 illustrates the speed-up achieved using MPCNewton in comparison with the other solvers.

n_x	n_u	N	MPCNewton (ms)	BPMPD (ms)	QPC (ms)	qpOASES (ms)
20	8	30	19.02	62.68	123	199
		50	24.33	107	389	608
		70	29.35	149	818	1340
		90	31.84	195	1430	2022
30	12	30	24.59	167	393	493
		50	41.49	289	1545	1943
		70	32.08	408	2279	2469
		90	39.33	533	4315	4590

40	16	30	24.98	304	528	819
		50	30.98	523	1771	1977
		70	57.63	777	6211	6885
		90	56.81	980	9356	10077
50	22	30	64.47	716	1679	2706
		50	106	1260	7650	8766
		70	111	1792	15630	16310
		90	130	2330	26047	26276

Table 3. Runtime results for randomly generated systems

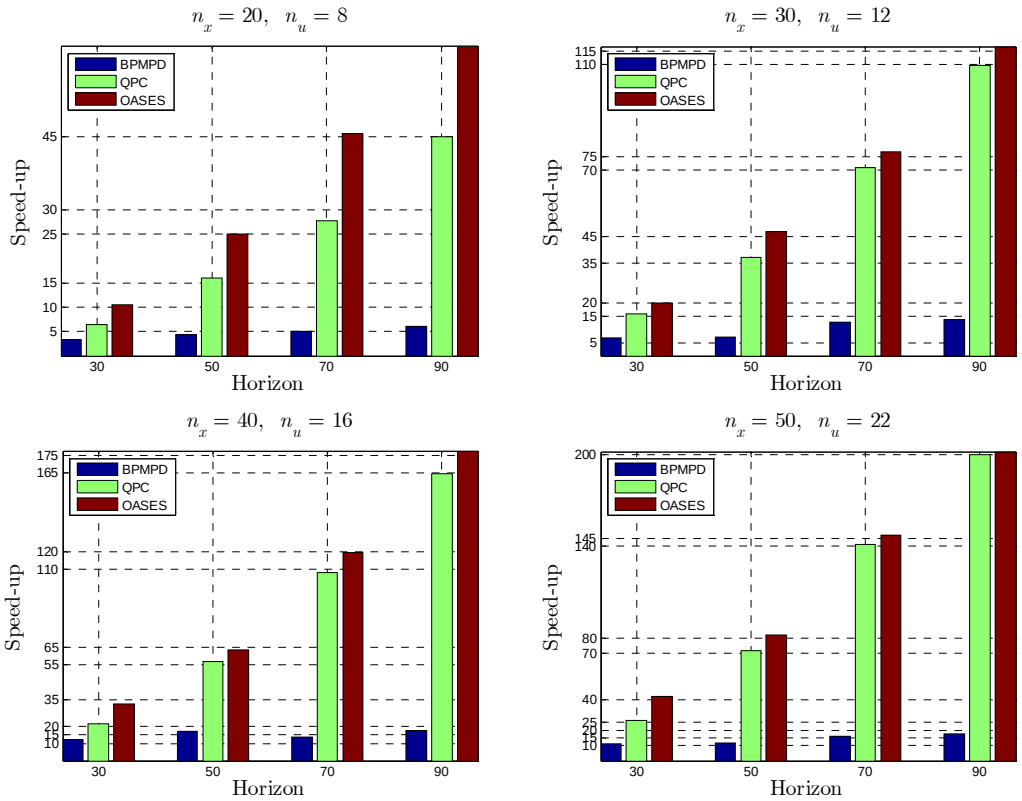

Figure 2. Speed-up achieved using MPCNewton for randomly generated systems

11. Conclusions and Future Directions

We have presented a new algorithm for fast solution of QP problems arising in MPC for constrained linear systems. The algorithm is based on a reformulation of the strictly convex QP as the unconstrained minimization of a convex quadratic spline. Therefore, the optimality conditions reduce to a system of piecewise affine equations. For the

solution of the problem, we proposed a piecewise smooth Newton method with line-search, a variant of the algorithm of Li & Swetits (1997). Furthermore, implementation details were given regarding the updating of the Cholesky factor and the line search procedure, as well as details regarding the application to MPC problems. The MATLAB implementation of the algorithm (MPCNewton) was tested on benchmark examples and compared against state-of-the art QP solvers and QP algorithms specifically tailored for MPC. MPCNewton outperforms all the QP solvers in terms of runtime, even though all other solvers are programmed in a low level language like C or C++. The speed-up becomes even larger for very large-scale problems and long prediction horizons. This makes the algorithm applicable for very demanding MPC problems where a long prediction horizon needs to be chosen in order to ensure stability of the closed-loop system. For example, for the demanding crude distillation unit model the maximum CPU time is 80 ms, allowing MPC to be carried out at 12.5Hz.

Future work can be focused on an efficient implementation of the algorithm in C++ and on deriving a bound on the number of iterations, dependent on system specific parameters only. Recently, a large amount of results has appeared in the literature regarding local and global error-bounds, i.e. inequalities that bound the distance of a point from the optimal solution (Luo & Tseng 1992a, 1992b, Li, 1995, Facchinei & Pang, 2003a). However, the constants appearing on these bounds are difficult to compute for large scale problems (e..g. Hoffman's error bound for polyhedral sets). Further work is needed in this direction.

References

Axehill, D., Hanson, A., (2008) A dual gradient projection quadratic programming algorithm tailored for model predictive control, in *Proc. of 47th IEEE Conference on Decision and Control*, pp 3057 - 3064.

Bartlett R.A., Biegler L.T. (2006) QPSchur: a dual, active set, Schur complement method for large-scale and structured convex quadratic programming algorithm. *Optimization and Engineering*, 7:5–32.

Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. (2002) The explicit linear quadratic regulator for constrained systems, *Automatica*, 38, 3–20.

Best M.J. (1996) An algorithm for the solution of the parametric quadratic programming problem. *Applied Mathematics and Parallel Computing*. Physica-Verlag: Heidelberg, 57–76.

Chen, B., Pinar, C. (1998), On Newton's method for Huber's robust M-estimation problems in linear regression, *BIT*, 38 (4), 674-684.

Davis, T.A., Hager, W.W. (1999). Modifying a sparse cholesky factorization, *Siam J. of Matrix Anal. Appl.*, **20**(3), 606-627.

Dennis, J.E., Schnabel, R. (1996) *Numerical methods for unconstrained optimization and nonlinear equations*, SIAM, Philadelphia, PA.

Dongarra, J.J., Bunch, J.R., Moler, C.B. and Stewart, G.W. (1979) *LINPACK Users' Guide*, SIAM, Philadelphia.

Facchinei, F. & Pang, J.S. (2003a) *Finite Dimensional Variational Inequalities and Complementarity Problems*, vol. I, Springer, New York.

Facchinei, F. & Pang, J.S. (2003b) *Finite Dimensional Variational Inequalities and Complementarity Problems*, vol. II, Springer, New York.

Ferreau, H. J., H. G. Bock, and M. Diehl, (2008) An Online Active Set Strategy to Overcome the Limitations of Explicit MPC, *Intl. J. Robust Nonlinear Contr.* , **18**(8), 816.

Fukuda, K., Jones, C.N. & Columbano, S., (2009) An Output-Sensitive Algorithm for Multi-Parametric LCPs with Sufficient Matrices, CRM Proceedings and Lecture Notes, vol. 48.

Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. A. (1974) Methods for modifying matrix factorizations, *Math. Comp.*, 28 , pp. 505–535.

Goldfarb D, Idnani A. (1983), A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33.

Li, W. (1995) Error bounds for piecewise convex quadratic programs and applications. *SIAM Journal on Control and Optimization*, 33, 1510-1529.

Li, W. & Nijs, J.J. (2003) An implementation of the Qspline method for solving convex quadratic programming problems with simple bound constraints. *Journal of Mathematical Sciences*, 116 (4) 3387-3410.

Li, W. & Swetits J. (1997) A new algorithm for solving strictly convex quadratic programs. *SIAM Journal on Optimization*, 7, 595-619.

Li, W. & Swetits J. (1999) Regularized Newton methods for minimization of convex quadratic splines with singular Hessians. In M. Fukishima and L. Qi (eds.), *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, Kluwer Academic Publishers (Dordrecht 1999) pp. 235-257.

Limon, D., Alamo, T., Salas, F., Camacho, E.F. (2006) On the stability of constrained MPC without terminal constraint. *IEEE Transactions on Automatic Control* 51, 832–836.

Luo, Z.Q. & Tseng P. (1992a) On the linear convergence of descent methods for convex essentially smooth minimization. *SIAM Journal on Control and* Optimization, 30, 408-425.

Luo, Z.Q. & Tseng P. (1992b) Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research* 46, 157-178.

Kojima, M. and Shindo, S. (1986) Extension of Newton and quasi-Newton methods to systems of PC^1 equations. Journal of Operations Research Society of Japan, 29, 352-374.

Madsen, K. and Nielsen, H.B. (1990) Finite algorithms for robust linear regression, *BIT*, 30, 682-699.

Mangasarian, O.L. (2002) A finite Newton method for classification problems. *Optimization Methods and Software* 17, 913-929.

Mayne, D.Q., Rawlings, J.B., Rao, C.V. and Scokaert, P.O.M. (2000) Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Mehrotra, S., (1992) On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization*, vol. 2, pp. 575-601.

Mészáros, C. (1999) The BPMPD interior-point solver for convex quadratic problems, *Optimization Methods and Software*, 11&12, pp. 431–449.

Nesterov, Y. (1983) A method for solving a convex programming problem with convergence rate $1/k^2$. *Soviet Math. Dokl.*, vol. 27, no. 2, pp. 372–376.

Nesterov, Y., Nemirovskii, A. (1994) *Interior point polynomial methods in convex programming: Theory and Applications*, SIAM, Philadelphia.

Nocedal, J. and Wright, S.J. (1999) *Numerical Optimization*, Springer, New York.

Online QP Benchmark Collection. http://www.kuleuven.be/optec/software/onlineQP/

Pannocchia, G., Rawlings, J.B. and Wright, S.J. (2006) Fast, large-scale model predictive control by partial enumeration, *Automatica*, vol. 43, no. 5, pp. 852–860,

qpOASES Homepage. http://homes.esat.kuleuven.be/~optec/software/qpOASES/

Richter, S., Jones, C.N. & Morari, M. (2009) Real-time input constrained MPC using fast gradient methods, in Proc. 48th IEEE Conference on Decision and Control, CDC, Shanghai.

Rao C.V., Wright S.J., Rawlings J.B. (1998) Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757.

Rawlings, J. B. and Mayne, D.Q. (2009) *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison.

Wang, Y., Boyd, S. (2010) Fast model predictive control using online optimization, *IEEE Transactions on Control Systems Technology*, 18(2):267-278, March 2010.

Wills, A. (2009) *QPC- Quadratic Programming in C,* Version 2.0, *http://sigpromu.org/staff/quadprog*

Wright S.J. (1997) *Primal–dual Interior-point Methods*. SIAM: Philadelphia, PA.