Katholieke Universiteit Leuven

Departement Elektrotechniek

ESAT-SISTA/TR 00-122

Quadratic Programming In Model Predictive Control For Large Scale Systems[1]

Jeroen Buijs and Bart De Moor[2]

September 2001
Accepted for the 15th IFAC World Congress on Automatic Control, Barcelona, July 2002

[1]This report is available by anonymous ftp from *ftp.esat.kuleuven.ac.be* in the directory *pub/sista/buijs/reports/qplarge.ps.gz*

[2]K.U.Leuven, Dept. of Electrical Engineering (ESAT), Research group SISTA, Kasteelpark Arenberg 10, 3001 Leuven(Heverlee), Belgium, Tel. 32/16/32 17 09, Fax 32/16/32 19 70, WWW: *http://www.esat.kuleuven.ac.be/sista*. E-mail: *bart.demoor@esat.kuleuven.ac.be*.

Abstract

Model Predictive Control(MPC) is widely used, especially in the chemical process industry. Model Predictive Controllers calculate the optimal control inputs at each time step, based on past information, a plant model, a quadratic objective and given constraints. Typically this involves solving a large linearly constrained quadratic program(LCQP) at each time step. Standard methods turn out to be too slow to calculate the desired inputs in time, i.e. each sampling instant. By exploiting the structure of the LCQP, specific methods for solving the QP's in MPC were developed recently. We will compare these methods with classical QP solvers and show their necessity when controlling large systems from the example of a high density polyethylene production plant. *Copyright ©2002 IFAC*

QUADRATIC PROGRAMMING IN MODEL PREDICTIVE CONTROL FOR LARGE SCALE SYSTEMS

J. Buijs and B. De Moor*

* *Katholieke Universiteit Leuven*
Department of Electrical Engineering, ESAT-SISTA
Kasteelpark Arenberg 10, 3001 Heverlee (Leuven), Belgium
Tel.: 32/16/32 11 60, Fax: 32/16/32 19 70
E-mail: {jeroen.buijs, bart.demoor}@esat.kuleuven.ac.be

Abstract: Model Predictive Control(MPC) is widely used, especially in the chemical process industry. Model Predictive Controllers calculate the optimal control inputs at each time step, based on past information, a plant model, a quadratic objective and given constraints. Typically this involves solving a large linearly constrained quadratic program(LCQP) at each time step. Standard methods turn out to be too slow to calculate the desired inputs in time, i.e. each sampling instant. By exploiting the structure of the LCQP, specific methods for solving the QP's in MPC were developed recently. We will compare these methods with classical QP solvers and show their necessity when controlling large systems from the example of a high density polyethylene production plant. *Copyright ©2002 IFAC*

Keywords: Model Predictive Control, Convex optimization, Quadratic Programming

1. INTRODUCTION

Model Predictive Control (MPC) is a control technique that has been developed for the chemical process industry and is widely used in that area (Qin and Badgwell, 1997). This control strategy uses a model and the state at the current time instant to predict the effect of control inputs on the future states and/or outputs of the system. A quadratic cost function is defined, involving the outputs, states and/or inputs of the system over a certain time horizon, with the model equations acting as constraints. The finite horizon allows to incorporate constraints on the inputs and outputs, which is one of the main differences with standard controller design methods and also one of the main reasons why MPC has become very popular in the chemical industry where many (hard) input and output constraints have to be dealt with.

One of the main drawbacks of nowadays model predictive controllers is the need to solve an optimization problem at each time step in order to be able to guarantee stability and a good performance. MPC controllers are rather difficult to implement on-line, because these optimization problems, typically linearly constrained quadratic problems (LCQP), tend to become too large to be solved in real-time when standard LCQP solvers are used. State-of-the-art implementations of model predictive controllers circumvent this problem by using a priorisation scheme for the constraints, or other tricks to solve the dynamic optimization problem. Recently, several methods were proposed to reformulate the opti-

mization problem or the entire MPC problem, in order to generate solutions in lesser time. Many of these methods try to approximate the given control problem by making assumptions, for instance about the control signal structure (e.g. Command Governing (Bemporad, 1997)). Nevertheless, the most promising methods do not make such assumptions, but exploit the structure of the given optimization problem. It will be shown that these structured methods can tackle the problem at hand for large scale systems in a much better way than standard LCQP solvers.

This paper is organized as follows. In Section 2, we give a description of the MPC problem and the optimization problem it leads to. Section 3 discusses standard methods to solve LCQP's and their disadvantages with respect to large problems. In section 4, structure exploiting methods for solving the LCQP in MPC applications are discussed and in section 5 some implementation details are discussed and the example of a high density polyethylene plant is given to show that these methods can play an important role in next generation MPC controllers.

2. MODEL PREDICTIVE CONTROL: PROBLEM DEFINITION

Let us consider a given discrete time linear model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{1}$$

where $u_k \in \mathbb{R}^{n_u}$ is the input vector, $y_k \in \mathbb{R}^{n_y}$ the output vector and $x_k \in \mathbb{R}^{n_x}$ the state vector at time instant k. This model is typically the result of a linearization of a non-linear system around a given working point. The control task is to find, at each time instant k, an optimal input sequence $u_k, \ldots, u_{k+N}$ over the horizon N, that generates a good tracking behaviour of the outputs while certain given constraints are fulfilled. This is usually done by defining a quadratic cost function in the outputs (or the states) and the inputs. Typically, the horizon is finite. If we define a tracking reference signal y^r for the outputs, this will lead to an optimization problem of the form

$$\begin{aligned} \min_{u_k, y_k} \; & \frac{1}{2} \sum_{k=0}^{N-1} [(y_k - y_k^r)^T Q_k (y_k - y_k^r) + u_k^T R_k u_k] \\ & + \frac{1}{2} y_N^T \bar{Q}_N y_N - y_N^r \tilde{Q}_N y_N \\ \text{s.t.} \; & \begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \\ D_k u_k \le d_k \\ E_{k+1} x_{k+1} \le e_{k+1} \\ \qquad\qquad k = 0, \ldots, N-1 \\ y_N = Cx_N \\ x_0 \text{ fixed and given} \end{cases} \end{aligned} \tag{2}$$

The matrices $\bar{Q}_N$ and $\tilde{Q}_N$ can be calculated in various ways to guarantee stability, depending on the assumption that one makes on the control law for $N > k$ and depending on whether A is stable or not. For unstable systems, an endpoint constraint of the form $Fx_N = 0$ is introduced to ensure stability of the unstable modes. The matrix F is constructed from the part of A that contains these modes. For more details we refer to (Rawlings and Muske, 1993). Notice that in almost all MPC applications, only the optimal input of the cuurent time instant is applied and that the calculations are repeated each time instant, giving rise to a moving window approach. In (Mayne *et al.*, 2000) it is shown that under certain assumptions, the solution can be proven to give asymptotically stable control.

In most MPC applications the state space equations are used to eliminate the states from the optimization problem in order to reduce the size (Muske and Rawlings, 1993). If we introduce

$$\begin{aligned} \mathbf{u} &= [u_k^T, u_{k+1}^T, \ldots, u_{k+N}^T]^T \\ \mathbf{d} &= [d_0^T, \ldots, d_{N-1}^T]^T \\ \mathbf{e} &= [e_1^T, \ldots, e_N^T]^T \\ Q_D &= \text{diag}\{Q_1, \ldots, Q_{N-1}\} \\ R_D &= \text{diag}\{R_0, \ldots, R_{N-1}\} \\ D_D &= \text{diag}\{D_0, \ldots, D_{N-1}\} \\ E_D &= \text{diag}\{E_1, \ldots, E_N\} \end{aligned} \tag{3}$$

and the matrices F_c, H_c and H_N,

$$\begin{aligned} F_c &= \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \\ H_c &= \begin{bmatrix} CB & 0 & 0 & \ldots & 0 \\ CAB & CB & 0 & \ldots & 0 \\ \vdots & & \ddots & & \\ CA^{N-1}B & CA^{N-2}B & & CB & 0 \end{bmatrix} \\ H_N &= \begin{bmatrix} A^N B & A^{N-1}B & \ldots & B \end{bmatrix} \end{aligned} \tag{4}$$

we can rewrite (2) as

$$\begin{aligned} \min_{\mathbf{u}} \; & \mathbf{u}^T (H_c^T Q H_c + R + {H_N}^T \bar{q}_{N_c} H_N)\mathbf{u} \\ & + x_0^T (F_c^T Q H_c + (CA^{N_c+1})^T \bar{q}_{N_c} H_N)\mathbf{u} \\ \text{s.t.} \; & \begin{cases} D_D \mathbf{u} \le \mathbf{d} \\ E_D H_c \mathbf{u} \le \mathbf{e} - E_D F_c x_0 \\ x_0 \text{ fixed and given} \end{cases} \end{aligned} \tag{5}$$

The number of variables is now reduced from $(n_y + n_x + n_u) * N$ to $n_u * N$. Since the cost is quadratic and all constraints are linear in $\mathbf{u}$, standard QP solvers might be used to solve this problem. The two best known methods to solve LCQP's are active set methods and interior point methods. They are briefly discussed in the next section,

where it will be made clear that these methods will give problems solving large scale problems.

3. STANDARD QP SOLVERS

Two well known classes of iterative methods exist for solving LCQP's like (5). The first class contains the **active set methods(ASM)**, the second one **interior point methods(IPM)**, see e.g. (Nash and Sofer, 1996), (Nocedal and Wright, 1999).

ASM try iteratively to find the set of constraints that are active at the optimum. To obtain that goal they solve an equality constrained problem at each time step to determine a new search direction. Each iteration thus requires solving a dense set of equations in Nn_u variables, leading to $O(N^3 n_u^3)$ operations per iteration. The total number of iterations will increase with the number of active constraints since approximately in each iteration one constraint will become active.

IPM try to solve the non-linear set of Karush-Kuhn-Tucker equations iteratively by making linear approximations to this set. These sets of equations are of the same size as for active set methods. Nevertheless, when the problems grow larger, interior point methods can make use more efficiently of sparse solvers and will be faster. Also the number of iterations to reach a point close to the optimum is typically independent of the number of constraints and will be smaller as for ASM.

Several publicly available codes of these algorithms can be found (Table 1). The fact that

Table 1. Some available Active Set and Interior Point Methods for LCQP's.

Name	Class	Where to be found
Mosek	IPM	`www.mosek.com`
Loqo	IPM	`www.princeton.edu/~rvdb`
Quadprog	ASM	Matlab's QP solver
NAG (e04n-series)	ASM	`www.nag.com`

all these methods become very slow when the number of variables and/or constraints increases, is shown in Figure 1, where the computational time is plotted versus the number of variables respectively for two of these algorithms. Another problem with these standard methods is that they generally require a lot of memory usage to store large dense matrices. For large problems memory problems may occur.

It is clear from these aspects that those standard methods are not suited for solving the given MPC problem for large scale systems with large horizons.

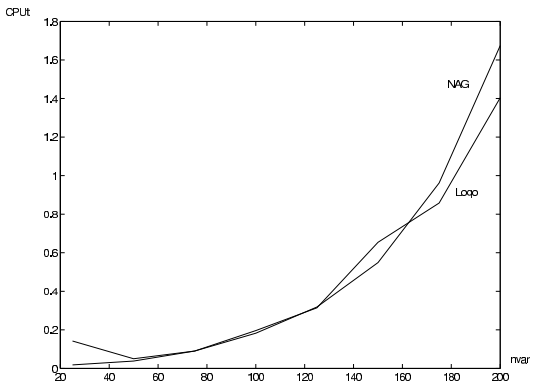

Fig. 1. Relation between number of variables in the given dynamic optimization and CPU time for some standard methods. Simulations were don on an Intel PIII

4. EXPLOITING STRUCTURE

When the output equation $y_k = Cx_k$ is used to eliminate the outputs in (5), but the states are not eliminated and kept as optimization variables, the number of variables becomes $N(n_u + n_x)$, but the remaining sets of equations that have to be solved will have a sparse structure that can be exploited.

For the interior point case this is thoroughly explained in (Rao and Rawlings, 1997). There a primal-dual interior point method is developed to solve the Karush-Kuhn-Tucker(KKT) equations for optimality, which are given by (assuming a zero reference without loss of generality),

$$\begin{cases} Ru_k + B^T p_k + D_k^T \lambda_k = 0 \\ E_{l+1}^T \xi_{l+1} - p_l + C^T Q C x_{l+1} + A^T p_{l+1} = 0 \\ E_N^T \xi_{N-1} - p_{N-1} + \bar{Q}_N x_N = 0 \\ x_{k+1} - Ax_k - Bu_k = 0 \\ d_k - D_k u_k - t_k^\lambda = 0 \\ e_{k+1} - E_{k+1} x_{k+1} - t_{k+1}^\xi = 0 \\ t_k^{\lambda T} \lambda_k = 0 \\ (t_{k+1}^\xi)^T \xi_{k+1} = 0 \\ \lambda_k, \xi_{k+1} \geq 0 \\ k = 0, \ldots, N-1 \\ l = 0, \ldots, N-2 \end{cases} \quad (6)$$

where the Lagrange multipliers p_k for the state space equations and λ_k, ξ_{k+1} for the input and output constraints respectively, and the corresponding slacks t_k^λ, t_k^ξ were introduced. The linear approximation of these KKT equations which has to be solved at each iteration step, can be rewritten as

$$\begin{bmatrix} \bar{R}_0 & B^T & & & & & & \\ B & & -I & & & & & \\ & -I & \bar{Q}_1 & & A^T & & & \\ & & & \bar{R}_1 & B^T & & & \\ & & A & B & & -I & & \\ & & & & -I & \bar{Q}_2 & & A^T \\ & & & & & & \bar{R}_2 & B^T \\ & & & & & A & B & \ddots \\ & & & & & & & \bar{Q}_N \end{bmatrix} \Delta s = r \quad (7)$$

with

$$\begin{aligned} \Delta s &= [\Delta u_0^T, \Delta p_0^T, \Delta x_1^T, \ldots, \Delta x_N^T]^T \\ r &= [r_{u_0}^T, r_{p_0}^T, r_{x_1}^T, \ldots, r_{x_N}^T]^T \end{aligned} \quad (8)$$

where Δs is the step direction to be determined in the current iteration step and r the residual vector resulting from the previous iteration respectively. The matrix on the left hand side of (7) has a banded structure with some newly introduced variables, defined in Appendix A. Applying a block elimination scheme the given set of equations can now be solved recursively in a Ricatti-like manner, by calculating, in a first recursion loop, for $k = N, \ldots, 2$,

$$\begin{aligned} &\Pi_N = \bar{Q}_N + E_N^T T_k^{\xi^{-1}} \Xi_k^{-1} E_N \\ &\pi_N = r_{x_N} \\ &\Pi_{k-1} = \tilde{Q}_{k-1} - A^T \Pi_k B \tilde{R}_{k-1}^{-1} B^T \Pi_k A \\ &\pi_{k-1} = \tilde{r}_{x_{k-1}} - A^T \Pi_k B \tilde{R}_{k-1}^{-1} \tilde{r}_{u_{k-1}} \end{aligned} \quad (9)$$

and in a second loop, for $k = 1, \ldots, N-1$, the step directions

$$\begin{aligned} \Delta u_k &= \tilde{R}_k^{-1}(\tilde{r}_{u_k} - B^T \Pi_k A \Delta x_k) \\ \Delta x_{k+1} &= A \Delta x_k + B \Delta u_k - r_{p_k} \\ \Delta p_k &= \Pi_k \Delta x_k - \pi_k \end{aligned} \quad (10)$$

The newly introduced variables in 9 and 10 are again defined in Appendix A. For the first step of this loop, i.e. for $k = 0$ (to initialize the recursion), a small set of equations remains to be solved,

$$\begin{bmatrix} R_0 & B^T & 0 \\ B & 0 & -I \\ 0 & -I & \Pi_1 \end{bmatrix} \begin{bmatrix} \Delta u_0 \\ \Delta p_0 \\ \Delta x_1 \end{bmatrix} = \begin{bmatrix} r_{u_0} \\ r_{p_0} \\ \pi_1 \end{bmatrix} \quad (11)$$

Using well known matrix inversion lemma's and assuming invertibility where required, the inverse of the left hand side matrix of (11) can easily be shown to be equal to

$$\begin{aligned} &\begin{bmatrix} R_0 & B^T & 0 \\ B & 0 & -I \\ 0 & -I & \Pi_1 \end{bmatrix}^{-1} = \begin{bmatrix} J & K \\ L & M \end{bmatrix}, \\ &J = (R_0 + B^T \Pi_1 B)^{-1} \\ &K = (R_0 + B^T \Pi_1 B)^{-1} B^T \begin{bmatrix} \Pi_1 & I \end{bmatrix} \\ &L = -K^T \\ &M = \begin{bmatrix} S & T \\ V & W \end{bmatrix} \\ &S = -\Pi_1 + \Pi_1 B (R_0 + B^T \Pi_1 B)^{-1} B^T \Pi_1 \\ &T = -I + \Pi_1 B (R_0 + B^T \Pi_1 B)^{-1} B^T \\ &V = -I + B (R_0 + B^T \Pi_1 B)^{-1} B \Pi_1 \\ &W = B (R_0 + B^T \Pi_1 B)^{-1} B^T \end{aligned} \quad (12)$$

and thus only some small inverses will have to be calculated in order to solve the entire set of equations.

The conclusion of these calculations is that because of the recursive scheme, only $O(N(n_u + n_x)^3)$ operations are needed per iteration, since the set of equations is divided in N separate blocks. Moreover, this scheme is very easy to implement and requires almost no memory.

Notice that other methods can be chosen to decompose the given banded matrix and solve the linear set of equations, see e.g. (Gopal and Biegler, 1998).

5. IMPLEMENTATION ISSUES AND EXAMPLE

In (Rao and Rawlings, 1997) the interior point method is implemented as a primal-dual Mehrotra's predictor-corrector method. Of course other variants of interior point methods can be chosen. It is worth noticing that in these primal-dual methods, infeasible starting points are allowed. Nevertheless that way the iteration trajectory will run through infeasible regions with the disadvantage that when the algorithm has to be stopped before reaching the optimal solution, it is not sure the algorithm will return a feasible iterate. Moreover, the convergence of the algorithm is highly dependent on the starting point. The duality gap, defined as

$$DG = \frac{\lambda^T t^\lambda + \xi^T t^\xi}{m} \quad (13)$$

where m is the number of inequality constraints and $\lambda, \xi, t^\lambda, t^\xi$ the vector of all input and output constraint Lagrange multipliers and slacks respectively, can be used as a stopping criterion ($DG \leq \epsilon$) as is often the case in IPM. The number of iterations, i.e. the number of times the recursion has to be done, can then be approximated after one iteration as

$$n_{it} = \frac{\log \frac{DG(0)}{\epsilon}}{\log \frac{DG(0)}{DG(1)}} \qquad (14)$$

where $DG(i)$ denotes the duality gap before the (i+1)-th iteration. This is an extension of the exact calculation of the number of outer iterations for standard IPM (Boyd and Vandenberghe, 2000). If the time for one iteration is approximately known, one has thus also an approximation for the time needed to obtain a given accuracy.

The example of a high density polyethylene plant is taken here to make some comparisons and show some properties. A non-linear HDPE model was linearized around a certain grade to obtain a linear state space model for the given grade. This model contains 45 states, 4 control inputs and 19 relevant outputs. The structured IPM was implemented in Matlab and all simulations were done in Matlab 5 on an Intel PIII 850MHz with 1Gb RAM. Notice that for instance Matlab's Quadprog did need that much memory (tests on a PC with 256Mb RAM failed for that reason) whereas the structured IPM could do with lesser memory (tests on a PC with 256Mb RAM gave no problems at all). The linear model is such that the reference is zero.

In Figure 2 the trajectories of some inputs and outputs are shown, first when the constraint bounds were taken such that the constraints were inactive, then with some active constraints. The number of iterations to obtain the solution at the first sample was approximated to be 9 using (14), while it turned out to be 11 in practice. In Figure 3 the CPU time needed to calculate the trajectories with active constraints for different horizons is shown. As expected, the calculation time increases linearly with N. For a naive active set implementation (here Matlab's `Quadprog`, modified to solve LCQP's only), calculation times become huge if the horizon increases. It is clear that standard methods require too much time and too much memory when large horizons are involved.

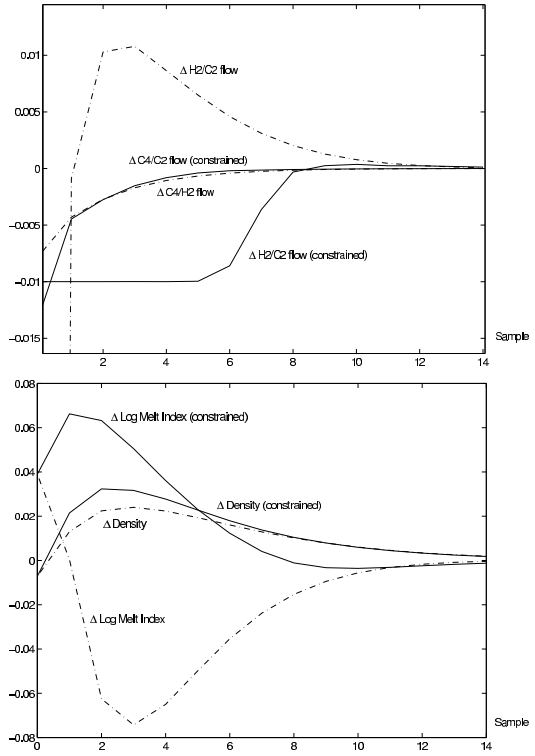

Fig. 2. Input (Above) and Output (below) trajectories for two inputs and two outputs, the dotted line when no constraints were active, the full line when the Δ H2/C2 flow was bounded above and below by -0.01 and 0.01 and the density was bounded by -0.04 below. The Δ C4/C2 flow is in promille, the Δ H2/C2 flow in ppm and the density in kg/m^3

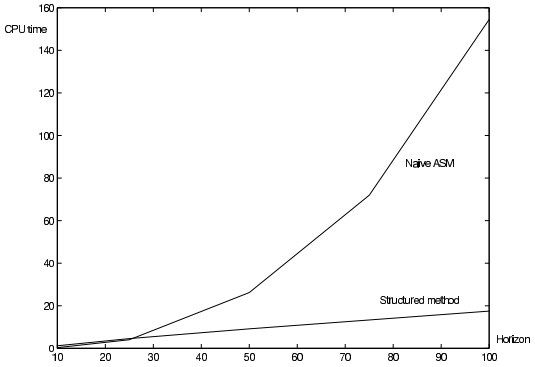

Fig. 3. Increase in CPU time when the horizon increases.

6. CONCLUSIONS

This paper shows that standard QP solvers are not suited to tackle the dynamic optimization in model predictive controllers for on-line use in real time. That is the reason why state-of-the-art model predictive controllers try to approximate the given optimization problems by priorisation, defining move times, etc. Nowadays methods are being developed which explicitly take the structure of the given problem into account such that the next generation of model predictive controllers will be able to tackle the dynamic optimization problem. From the example given it is clear that for problems with large horizons and many (active) constraints naive implementations of QP solvers become useless since it will be impossible to give an optimal solution within the available time when running in real-time. Moreover, dense methods require more memory to store the full dense matrices.

7. ACKNOWLEDGEMENTS

Jeroen Buijs is a Research Assistant with the K. U. Leuven. Dr.Ir. Bart De Moor is full professor at the Katholieke Universiteit Leuven (www.esat.kuleuven.ac.be/ demoor). This work was supported by grants and funding from the Research Council of the K.U.Leuven (PhD/postdoc grants, IDO , GOA Mefisto-666), the Foundation for Scientific Research Flanders (FWO)(PhD and postdoc grants and projects (G.0115.01, G.0197.02, G.0407.02), research communities (ICCoS, ANMMM)), the Flemish regional government (Bilateral Coll., IWT-project Soft4s, Eureka projects Synopsis, Impact, FLiTE, STWW-project Genprom, GBOU-project McKnow, IWT PhD grants), the Belgian Federal Government (DWTC, IUAP IV-02, Sustainable Development MD/01/024), the European Commission (TMR Alapedes, Ernsi) and industry supported direct contract research (Electrabel, ELIA, ISMC, Data4s).

The scientific responsibility is assumed by its authors.

REFERENCES

Bemporad, A. (1997). *Reference Governors: On-Line Set-Point Optimization Techniques for Constraint Fulfillment*. PhD, University of Florence.

Boyd, S. and L. Vandenberghe (2000). *Course Notes: EE364: Convex Optimization with Engineering Applications.*. Stanford University.

Gopal, V. and L.T. Biegler (1998). Large scale inequality constrained optimization and control. *IEEE Control Systems* **19(6)**, 59–68.

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

Muske, K.R. and J.B. Rawlings (1993). Model predictive control with linear models. *AIChE Journal* **39(2)**, 262–287.

Nash, S.G. and A. Sofer (1996). *Linear and Nonlinear Programming*. The McGraw-Hill Companies, Inc.

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

Qin, S.J. and T.A. Badgwell (1997). An overview of industrial model predictive control technology. *AIChE Symposium Series: Fifth Int. Conf. on Chemical Process Control* **316(93)**, 232–256.

Rao, C.V., S.J. Wright and J.B. Rawlings (1997). Application
of interior point methods to model predictive control. *Preprint ANL/MCS-P664-0597, Argonne National Laboratory*.

Rawlings, J.B. and K.R. Muske (1993). Stability of constrained receding horizon control. *IEEE Transactions on Automatic Control* **38(10)**, 1512–1516.

APPENDIX A

In equation (7), for
$k = 0, \ldots, N-1$ and $l = 1, \ldots, N-1$, the following matrices were introduced,

$$\begin{aligned} \bar{R}_k &= R_k + D_k^T T_k^{\lambda^{-1}} \Lambda_k^{-1} D_k \\ \bar{Q}_l &= C^T Q_l C + E_l^T T_l^{\xi^{-1}} \Xi_l^{-1} E_l \end{aligned}$$
with
$$\begin{aligned} T_k^{\lambda} &= \mathrm{diag}\{t_k^{\lambda}\} \\ \Lambda_k &= \mathrm{diag}\{\lambda_k\} \\ T_l^{\xi} &= \mathrm{diag}\{t_l^{\xi}\} \\ \Xi_l &= \mathrm{diag}\{\xi_l\} \end{aligned} \tag{15}$$

In equations (9) and (10), again for
$k = 0, \ldots, N-1$ and $l = 1, \ldots, N-1$, the following matrices and vectors were introduced,

$$\begin{aligned} \tilde{R}_k &= \bar{R}_k + B^T \Pi_{k+1} B \\ \tilde{Q}_l &= \bar{Q}_l + A^T \Pi_{l+1} A \\ \tilde{r}_{u_k} &= r_{u_k} + B^T \Pi_{k+1} r_{p_k} + B^T \pi_{k+1} \\ \tilde{r}_{x_k} &= r_{x_k} + A^T \Pi_{k+1} r_{p_k} + A^T \pi_{k+1} \end{aligned} \tag{16}$$