Suboptimal Solutions for Time Varying Time Delayed MPC Controllers

T. Veli Mumcu[1], K. Gulez[1]
[1]*Department of Control and Automation Engineering, Yildiz Technical University, Istanbul, Turkey*
tmumcu@yildiz.edu.tr

***Abstract*—Model Predictive Control (MPC) with constraints is still an interesting subject and offers many problems to work on. This study basically aims to understand the optimization process and the decrease of convex quadratic costs in a single model predictive controller. For those processes where the system dynamics change so slowly it is essential to obtain the control law as soon as possible to minimize the time delay on the controller side. This study proposes an early termination of the optimization process and the suboptimal solution to the quadratic programming. To define the early termination in the following chapters it is discussed and explained when, where. The implementation of the strategy is also illustrated with a case study and it is compared to the LQR controller for the regulator problem.**

***Index Terms*—Model predictive control, MPC, time varying time delay, quadratic programming.**

I. INTRODUCTION

The main difference between the conventional control and model predictive control is that conventional control uses a pre-computed control law (offline computation). Model predictive control or receding horizon control uses on-line optimization to solve a finite horizon open loop control problem based on iterative algorithms, applying the plant current state as the initial state. The advantages of model predictive control are that it can handle multivariable or time-varying plants; constraints can be employed directly in the problem [1], [2].

To make the implementation of MPC easy one of the methods that can be used is real-time implementation. Today execution time characteristics of MPC tasks are one of most interesting problems. The time-varying execution times introduce delays to the control systems which are not easy to compensate. The more time needed for the optimization the more the latency has the systems. The latency has an effect in control systems the same as input time delay which affects the control performance negatively unless it is well compensated. From a real time perspective it is held till now only by Cervin *et al* who considers the MPC tasks as periodic tasks for EDF scheduling based on early termination of iterative algorithms. In these papers, it is claimed that the early termination of the iterative algorithms can increase the performance index of MPC and still have efficient results for the prediction of the control signals and fulfil the stability conditions [1], [3], [4].

In the last years, MPC controllers are often studied for time delay in sense of network control [5] where feasible networked MPC scheme is used for discrete time interconnected systems on which time varying transmission delay affects the network [6] via minimizing the upper bound of the cost function it is used as a robust one in discrete-time uncertain systems with time-varying delay, input constraints and bounded nonlinear perturbations. Reference [7] discusses delay compensation of the moving horizon estimator and aims to compensate the loss by updating the covariance matrix for those states if there is incoming data from the sensors for them. Another compensation study [8] takes the time varying time delays introduced by the communication network into consideration and offers a solution by modelling delays as disturbances. Reference [9] proposes MPC algorithm for linear parameter varying systems with unbounded delay and derives the sufficient conditions from linear matrix inequalities using relaxation matrices. Rather than other time varying time delays this study focuses on the delays on controller side.

II. MPC WITH TIME-VARYING TIME DELAY

The basic question in a single MPC with constraints is how long optimization process should go on. It is known that each iteration optimization process goes on further; a better value is achieved for the control signal. However, while the objective function decreases in every each iteration, the total cost depends on delay increase.

Up to now, the previous work which has been done in this field suggests interrupting the optimization when the costs depending on delay begin to increase. At this point, it is suggested to use a simple stop criterion. With this criterion it is aimed to hold the costs depending on the delay at a certain level. One basic illustration to this idea can be seen on Fig. 1.

Here on the figure, line one illustrates the optimization process which leads to a decrease of the objective function in every iteration step. Line two illustrates the costs depending on time-delay which is based on delay time before the control signal is given to the process. The idea in the previous work is to interrupt the optimization at a point on which costs start to increase when the optimization

Manuscript received April 9, 2013; accepted October 6, 2013.
This research has been supported by Yildiz Technical University Scientific Research Projects Coordination Department. Project Number: 2010-04-02-KAP05.

process reaches a sub-optimal value which is not the best but still good for obtaining the control signal.

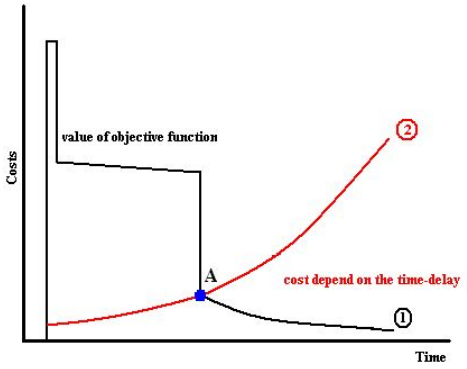


Fig. 1. Simple stop criterion proposed in the previous work [1], [2].

For this figure then point A is the point where the optimization process is interrupted at a sub-optimal value and save time, before the delay dependent costs start increase vigorously. However, it is not declared in the work how this point theoretically can be obtained. The previous work suggests us to use a simple stop criterion based on experienced maximum allowed delay and needs to be searched for every single system. Moreover, it is not known if this point or maximum allowed delay is the best value where the optimization process can be interrupted or since it is based on the experience can there be a better value or a point which is not experienced, yet [3].

III. Mathematical Methods Which Can Be Used for Defining the Premature Optimization in a Single MPC

In the light of the previous works done in this field it is known that the premature optimization process is studied with the active set methods. For the optimization process when active set methods are used, it is known that the optimum point is obtained by adding and removing the constraints to the active set. With this method it is not clear how the objective function will decrease as the number of the iterations increases in time. Because of working with different constraints inside the active set at different iterations it is also not clear how far the process should go on to achieve a better error and how many iterations it takes for the algorithm to reach the optimum point. All these factors make it not clear not only to decide whether it is logical to optimize further or not, but also theoretically not feasible to reach to the optimum point at a certain time [3], [4].

In the optimization process inside an MPC algorithm, the active set, simplex and interior point methods are the common solvers that can be used. Above it is clarified why it wouldn't be so suitable to use active set methods [10], [11].

For this work basically three reasons make interior point algorithms so attractive to implement to the optimization process for finding the premature optimal point. First the complexity of the total algorithm is always polynomial, has mostly O (n^3l) complexity. Second, it offers an estimation of how far from the optimal value a solution at a given iteration is. Third, the numbers of the iteration which is needed to find the optimal point is bounded by an upper value bounded, and it is also one of fastest algorithm for the solution of the convex quadratic problem [12], [13].

IV. Defining the Premature Optimization in a Single MPC

A. Defining the Cost Function for Convex Quadratic Optimization Problem

In order to define the premature optimization in a single MPC, system equations with a control signal with time delay (Fig. 2) which is caused by solving the optimization problem should be transferred from continuous time to the discrete time.

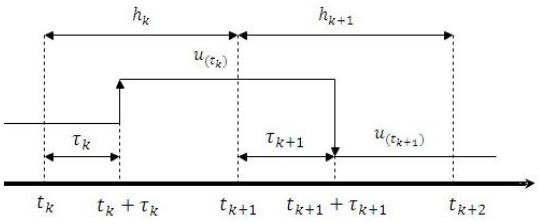


Fig. 2. Time model of the control signal with time delay [14].

The plant (1) has a continuous-time state space model:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t-^2), \\ y = Cx(t). \end{cases} \tag{1}$$

Our aim is to discretise (1) with time delay so that the time delay can be transferred from continuous time to discrete time. Thus, the discreted time system equations [15] are seen in (2):

$$\begin{cases} x(t_{k+1}) = \ddot{}(h_k)x(t_k) + \Gamma_1(h_k,^2_k)u(t_{k-1}) + \\ +\Gamma_0(h_k,^2_k)u(t_k), \\ y(t_k) = Cx(t_k). \end{cases} \tag{2}$$

The matrix form of (2) is (3)

$$\begin{bmatrix} x(t_{k+1}) \\ u(t_k) \end{bmatrix} = \underbrace{\begin{bmatrix} \ddot{}(h_k) & \Gamma_1(h_k,^2_k) \\ 0 & 0 \end{bmatrix}}_{H} \begin{bmatrix} x(t_k) \\ u(t_{k-1}) \end{bmatrix} + \underbrace{\begin{bmatrix} \Gamma_0(h_k,^2_k) \\ I \end{bmatrix}}_{J} u(t_k). \tag{3}$$

The cost function which is needed for optimization problem is shown below (4). To make the discretization process easier here a classic LQR cost function assumptions (5) will be used. Here after (6), it is assumed that the delay is embedded within the control signal u_k to make the following equations easier to follow. The cost function in continuous time is seen in (4)

$$J = \int_{kh}^{(k+1)h} \begin{pmatrix} x(t) - x_{ref} \\ u(t-^2) - u_{ref} \end{pmatrix}^T \begin{pmatrix} Q_C & N_C \\ N_C^T & R_C \end{pmatrix} \begin{pmatrix} x(t) - x_{ref} \\ u(t-^2) - u_{ref} \end{pmatrix}. \tag{4}$$

Here, cost function is chosen as LQR that's why x_{ref} and u_{ref} will be zero. Then, the final cost function in continuous time is as in (6):

$$\begin{cases} N_C \to 0, \\ Q_C = {Q_C}^T > 0, \\ R_C = {R_C}^T > 0. \end{cases} \tag{5}$$

$$J = \int_{kh}^{(k+1)h} \begin{pmatrix} x(t) \\ u(t-^2) \end{pmatrix}^T \begin{pmatrix} Q_C & 0 \\ 0 & R_C \end{pmatrix} \begin{pmatrix} x(t) \\ u(t-^2) \end{pmatrix}. \tag{6}$$

Once the discretized cost function is obtained, the following operations will be executed in order to transform the cost function so that it fits to the convex quadratic problem formulation.

B. Cancelling Cross Term (Cross Term Elimination) [16]

Cross term N_d is cancelled (7)–(10) since the aim in the objective function to build the costs based on only state x_k and control signal u_k:

$$\underline{M} = \underline{N}_d \underline{R}_d^{-1}, \tag{7}$$

$$\tilde{\underline{u}}_k = \underline{u}_k + \underline{M}^T \underline{x}_k, \tag{8}$$

$$\tilde{\underline{Q}} = \underline{Q} - \underline{M}\underline{N}_d, \tag{9}$$

$$\tilde{\underline{H}} = \underline{H} - \underline{J}\underline{M}^T. \tag{10}$$

The system equations after cross term elimination are (11)–(13):

$$\tilde{\underline{H}} = \underline{H} - \underline{J}\underline{M}^T, \tag{11}$$

$$\tilde{\underline{u}}_k = \underline{u}_k + \underline{M}^T \underline{x}_k, \tag{12}$$

$$\underline{x}_{k+1} = \tilde{\underline{H}}\underline{x}_k + \underline{J}\,\tilde{\underline{u}}_k. \tag{13}$$

Matrix decomposition (Cholesky factorization) [16] of the weighting terms of the cost function is (14), (15):

$$\tilde{\underline{Q}} = \tilde{\underline{C}}'\tilde{\underline{C}}, \tag{14}$$

$$\underline{R}_d = \tilde{\underline{L}}'\tilde{\underline{L}}. \tag{15}$$

The system equations (11)–(13) after Cholesky factorization are (16)–(18):

$$\tilde{\underline{H}} = \underline{H} - \underline{J}\underline{M}^T, \tag{16}$$

$$\hat{\underline{u}}_k = \tilde{\underline{L}}\tilde{\underline{u}}_k, \tag{17}$$

$$\underline{X}_{k+1} = \tilde{\underline{H}}\underline{X}_k + \tilde{\underline{J}}\,\hat{\underline{u}}_k. \tag{18}$$

After these operations the discreted cost function for the delay depended MPC is (19)

$$J = \sum_{h=1}^{Np} \left(\tilde{\underline{C}}\underline{X}_k\right)^T \left(\tilde{\underline{C}}\underline{X}_k\right) + \hat{\underline{u}}_k^T \hat{\underline{u}}_k. \tag{19}$$

Now the delay-depended MPC (20) can be built for the regulator case using the transformed system equations

$$\underbrace{\begin{bmatrix} \underline{x}_{k+1} \\ \vdots \\ \underline{x}_{k+Nu} \\ \underline{x}_{k+Nu+1} \\ \vdots \\ \underline{x}_{k+Np} \end{bmatrix}}_{\underline{X}p} = \underbrace{\begin{bmatrix} \tilde{\underline{H}} \\ \vdots \\ \tilde{\underline{H}}^{Nu} \\ \underline{H}^{Nu+1} \\ \vdots \\ \tilde{\underline{H}}^{Np} \end{bmatrix}}_{\underline{A}p} \underbrace{\underline{X}_k}_{initialstate} + \underbrace{\begin{bmatrix} \tilde{\underline{J}} & & & \\ \vdots & \ddots & & \\ \underline{H}^{Nu-1}\tilde{\underline{J}} & \cdots & \tilde{\underline{J}} & \\ \underline{H}^{Nu}\tilde{\underline{J}} & \cdots & \tilde{\underline{J}} & \\ \vdots & & \ddots & \\ \underline{H}^{Np-1}\tilde{\underline{J}} & \cdots & & \tilde{\underline{J}} \end{bmatrix}}_{Nu(B1),Np-Nu(B2)} \begin{bmatrix} \hat{\underline{u}}_k \\ \vdots \\ \hat{\underline{u}}_{k+Nu-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{20}$$

where $\underline{x}_p = \underline{A}_p \underline{X}_k + \underline{B}_1 \hat{\underline{u}}_k^*$ and x_p is substituted in cost function J, and Q is chosen as C^TC Then, our objective function for the quadratic programming will become (21)

$$J = \underline{x}_p^T \tilde{\underline{Q}} \underline{x}_p + \begin{pmatrix} \hat{\underline{u}}_k^* \end{pmatrix}^T \hat{\underline{u}}_k^* =$$

$$= \left[\tilde{\underline{C}}\left(\underline{A}_p \underline{x}_k + \underline{B}_1 \hat{\underline{u}}_k^*\right)\right]^T \left[\tilde{\underline{C}}\left(\underline{A}_p \underline{x}_k + \underline{B}_1 \hat{\underline{u}}_k^*\right)\right] +$$

$$+ \begin{pmatrix} \hat{\underline{u}}_k^* \end{pmatrix}^T \hat{\underline{u}}_k^* = \begin{pmatrix} \hat{\underline{u}}_k^* \end{pmatrix}^T \underline{T} \begin{pmatrix} \hat{\underline{u}}_k^* \end{pmatrix} + 2\begin{pmatrix} \hat{\underline{u}}_k^* \end{pmatrix}^T \underline{c}. \tag{21}$$

C. Defining the Constraints for Convex Quadratic Optimization Problem

It is known that MPC is solved via quadratic programming in case of constraints. Now the constraints (22) should be defined for the control horizon and introduced to the convex quadratic optimization problem.

$$\underline{u} \le \begin{bmatrix} \underline{u}_k \\ \vdots \\ \underline{u}_{k+hc-1} \end{bmatrix} \le \bar{\underline{u}}. \tag{22}$$

All operations applied to the system equations and cost function should also be applied to the constraints.

Cancelling cross term (23)–(26) (cross term elimination):

$$\tilde{\underline{u}}_k = \underline{u}_k + \underline{M}^T \underline{x}_k, \tag{23}$$

$$\underline{u}_k = \tilde{\underline{u}}_k - \underline{M}^T \underline{x}_k, \tag{24}$$

$$\tilde{\underline{H}} = \underline{H} - \underline{J}\underline{M}^T, \tag{25}$$

$$\underline{x}_{k+1} = \tilde{\underline{H}}\underline{x}_k + \underline{J}\,\tilde{\underline{u}}_k. \tag{26}$$

The new constraints after cross term elimination are (27)

$$\underline{\underline{u}} + \underline{\underline{M}}^T \underline{x}_k \le \tilde{\underline{u}}_k \le \bar{\underline{u}} + \underline{\underline{M}}^T \underline{x}_k. \quad (27)$$

Matrix decomposition of the constraints (Cholesky factorization) is (28)–(31):

$$\tilde{\underline{J}} = \underline{J}\underline{L}^{-1}, \quad (28)$$

$$\tilde{\underline{u}}_k = \frac{\hat{\underline{u}}_k}{\tilde{\underline{L}}}, \quad (29)$$

$$\tilde{\underline{H}} = \underline{H} - \underline{J}\underline{M}^T, \quad (30)$$

$$\underline{x}_{k+1} = \tilde{\underline{H}}\underline{x}_k + \tilde{\underline{J}}\,\hat{\underline{u}}_k. \quad (31)$$

The new constraints after Cholesky factorization are (32)

$$\tilde{\underline{L}}\left(\underline{\underline{u}} + \underline{\underline{M}}^T \underline{X}_k\right) \le \hat{\underline{u}}_k \le \tilde{\underline{L}}\left(\bar{\underline{u}} + \underline{\underline{M}}^T \underline{X}_k\right). \quad (32)$$

Thus, it can easily be seen that the constraints for the control signal in this case depended on the state vector again nevertheless this dependence should be removed by introducing another MPC (33) only for control horizon to the system

$$\underbrace{\begin{bmatrix} \underline{x}_{k+1} \\ \vdots \\ \underline{x}_{k+Nu} \end{bmatrix}}_{\underline{X}p} = \underbrace{\begin{bmatrix} \tilde{\underline{H}} \\ \vdots \\ \tilde{\underline{H}}^{Nu} \end{bmatrix}}_{\underline{A}p} \underbrace{\underline{x}_k}_{initialstate} + \underbrace{\begin{bmatrix} \tilde{\underline{J}} & & & \\ \vdots & & \tilde{\underline{J}} & \\ \tilde{\underline{H}}^{Nu-1} & \tilde{\underline{J}}^{Nu-1} & \cdots & \tilde{\underline{J}} \end{bmatrix}}_{Nu(B1)} \underbrace{\begin{bmatrix} \hat{\underline{u}}_k \\ \vdots \\ \hat{\underline{u}}_{k+Nu-1} \end{bmatrix}}_{\left(\hat{\underline{u}}_k\right)^*}. \quad (33)$$

where $\underline{x}_p = \underline{A}_{Nu}\underline{x}_k + \underline{B}_1 \hat{\underline{u}}_k^*$

The altered constraints after MPC formulation are (34)

$$\frac{\tilde{\underline{L}}\left(\underline{\underline{u}} + \underline{\underline{M}}^T \left(\underline{A}_{Nu}\underline{x}_k\right)\right)}{1 - \tilde{\underline{L}}\underline{\underline{M}}^T \underline{B}_1} \le \left(\hat{\underline{u}}_k\right)^* \le \frac{\tilde{\underline{L}}\left(\bar{\underline{u}} + \underline{\underline{M}}^T \left(\underline{A}_{Nu}\underline{x}_k\right)\right)}{1 - \tilde{\underline{L}}\underline{\underline{M}}^T \underline{B}_1}. \quad (34)$$

V. Convex Quadratic Programming for MPC via A Path Following Method

Once the MPC is built under the constraints, it leads to a quadratic programming problem. Literally this problem can mainly be solved either using an active set algorithm (simplex for quadratic case), or an interior point algorithm. For this MPC structure with constraint(s) it is chosen to have an interior point algorithm since these methods have the following advantages: Based on the search direction criterion it is often possible to prove that the cost function decreases in every iteration, however, this is not possible in active set or simplex methods since the value of the cost function can remain same [10]. Interior point methods have an iteration bound which gives information in the current iteration about the distance to the optimum point [11].

Since the iteration calculation is constituted of fixed numbers of equations, iteration has always the same processing time on a microprocessor.

All these advantages help us to define the pre-optimization point where the total cost begins to increase after a certain number of iterations because of the time-varying time delay. The more time it takes to calculate the control signal the more the system will become distant to the point where the state variables are measured.

A path following method is chosen here to solve the MPC problem. Today there are several packages which can be used to solve this problem via an interior point algorithm (Newton-KKT [13], Quadprog-middle scale algorithms [17], Yalmip (interior point option) [18], QPC [19], etc).

Here is (35) objective function with constraints again. The quadratic programming solver QPC can handle this problem directly, however for other solvers these constraints should be altered to the form $\underline{A}\underline{x} \le \underline{b}$ or $\underline{A}\underline{x} \ge \underline{b}$ just as in (36):

$$J = \frac{1}{2}\left[\frac{1}{2}\left(\hat{\underline{u}}_k^*\right)^T \underline{T}\left(\hat{\underline{u}}_k^*\right) + \underline{c}^T\left(\hat{\underline{u}}_k^*\right)\underline{c}\right], \quad (35)$$

$$\frac{\tilde{\underline{L}}\left(\underline{\underline{u}} + \underline{\underline{M}}^T \left(\underline{A}_{Nu}\underline{X}_k\right)\right)}{1 - \tilde{\underline{L}}\underline{\underline{M}}^T \underline{B}_1} \le \left(\hat{\underline{u}}_k\right)^* \le \frac{\tilde{\underline{L}}\left(\bar{\underline{u}} + \underline{\underline{M}}^T \left(\underline{A}_{Nu}\underline{X}_k\right)\right)}{1 - \tilde{\underline{L}}\underline{\underline{M}}^T \underline{B}_1}. \quad (36)$$

The constraints (37) have the form of $\underline{A}\underline{x} \ge \underline{b}$

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \left(\hat{\underline{u}}_k\right)^* \\ \left(\hat{\underline{u}}_k\right)^* \end{bmatrix} \ge \begin{bmatrix} \dfrac{\tilde{\underline{L}}\left(\underline{\underline{u}} + \underline{\underline{M}}^T \left(\underline{A}_{Nu}\underline{X}_k\right)\right)}{1 - \tilde{\underline{L}}\underline{\underline{M}}^T \underline{B}_1} \\ \dfrac{\tilde{\underline{L}}\left(\bar{\underline{u}} + \underline{\underline{M}}^T \left(\underline{A}_{Nu}\underline{X}_k\right)\right)}{1 - \tilde{\underline{L}}\underline{\underline{M}}^T \underline{B}_1} \end{bmatrix}^*. \quad (37)$$

In the constraint formula it shouldn't be forgotten that the state variable $\underline{x}_k$ is the initial value of the variable. In order to have the representation seen easier, this representation is expressed in the form of classic quadratic programming [9] in vector representation as in (38)

$$J = \frac{1}{2}x^T Q x + c^T x. \quad (38)$$

VI. Convergence of Quadratic Programming

This part deals with solving the quadratic programming problem in a general case. In each iteration, a pre-optimization criterion for a delay depended MPC makes only sense in case of decreasing costs. The decrease of the costs in interior point methods is always guaranteed with a safe search direction criterion. This criterion can be found for the solvers in their convergence analysis. Together decreasing costs and iteration bound help us to see how far a given iteration is from the optimum point.

In this work the solver [12] is used. The following proof

belongs to the reference [12] and is added here to explain the idea how cost decrease in optimization process. In this solver the regular convergence ratio (39) is defined. The proof of (39) is explained in the following (39)–(45). Here, some symbol definitions are listed below:

n: Number of variables in quadratic programming,
$\lambda>0$ is the barrier parameter,
and chosen as $\mathrm{a} = \left(1-\frac{\sim}{\sqrt{n}}\right)z^k$.
s^k: Slackness vector,
z^k: Mean value,
α: $\alpha<1$,
S^k: diag (s^k),
ΔX: diag (Δx),
$\mu = z^k$.

$$z^{k+1} \le \left(1-\frac{1}{4\sqrt{n}}\right)z^k. \tag{39}$$

Lemma I:

$$z^{k+1} \le \left(1-\frac{\sim}{\sqrt{n}}+\frac{\sim^2}{4n}\right)z^k. \tag{40}$$

Proof:
Note that:

$$X^k\Delta s+\mu\left(X^k\right)^{-1}\Delta x = \mathrm{a}e - X^k s^k, \tag{41}$$

$$\begin{aligned} X^{k+1}s^{k+1} &= \left(X^k+\Delta X\right)\left(s^k+\Delta s\right) = \\ &= X^k s^k+\Delta X s^k+X^k\Delta s+\Delta X\Delta s = \\ &= X^k s^k+X^k\Delta s+\mu\left(X^k\right)^{-1}\Delta x+\Delta X s^k+\Delta X\Delta s = \\ &= \mathrm{a}e+\Delta X\left(s^k+\Delta s-\mu\left(X^k\right)^{-1}e\right). \end{aligned} \tag{42}$$

From (41), (42) the following (43) is obtained

$$X^{k+1}s^{k+1} = \mathrm{a}e+\Delta X\left(X^k\right)^{-1}\left(\mathrm{a}e-\mu\left(X^k\right)^{-1}\Delta x-\mu e\right). \tag{43}$$

From (43)

$$\begin{aligned} nz^{k+1} &= e^T X^{k+1}s^{k+1} = \\ &= \mathrm{n}\mathrm{a}+\left(\mathrm{a}-\alpha\right)e^T\left(X^k\right)^{-1}\Delta x-\alpha\left\|\left(X^k\right)^{-1}\Delta x\right\|^2 = \\ &= \mathrm{n}\mathrm{a}-\frac{\sim z^k}{\sqrt{n}}e^T\left(X^k\right)^{-1}\Delta x-\alpha\left\|\left(X^k\right)^{-1}\Delta x\right\|^2 \le \\ &\le \mathrm{n}\mathrm{a}+\frac{\sim z^k}{\sqrt{n}}\left|e^T\left(X^k\right)^{-1}\Delta x\right|-\alpha\left\|\left(X^k\right)^{-1}\Delta x\right\|^2 \le \\ &\le \mathrm{n}\mathrm{a}+\frac{\sim z^k}{\sqrt{n}}\left\|e^T\right\|\left\|\left(X^k\right)^{-1}\Delta x\right\|-\alpha\left\|\left(X^k\right)^{-1}\Delta x\right\|^2 = \\ &= \mathrm{n}\mathrm{a}+\sim z^k\left\|\left(X^k\right)^{-1}\Delta x\right\|-\alpha\left\|\left(X^k\right)^{-1}\Delta x\right\|^2 = \\ &= \mathrm{n}\mathrm{a}+z^k\left(\sim\left\|\left(X^k\right)^{-1}\Delta x\right\|-\left\|\left(X^k\right)^{-1}\Delta x\right\|^2\right) \le \\ &\le \mathrm{n}\mathrm{a}+\frac{\sim^2 z^k}{4}. \end{aligned} \tag{44}$$

The last inequality (43) is true since the quadratic term reaches its maximum value at this value $\left\|\left(X^k\right)^{-1}\Delta x\right\| = \frac{\sim}{2}$.

Note that, $\alpha = z^k$ and $\mathrm{a} = \left(1-\sim/\sqrt{\mathrm{n}}\right)\mathrm{z}^{\mathrm{k}}$.

$$z^{k+1} \le \left(1-\frac{\sim}{\sqrt{n}}+\frac{\sim^2}{4n}\right)z^k. \tag{45}$$

With this proof lasted in (45), the aim is to show that in a path following solver the costs are decreasing through optimization process rather than they decrease with a regular factor. Thus, the general iteration bound belongs to [11]. This iteration bound can be updated to the solvers using the right decrease factors.

Lemma II: If the barrier parameter μ starts with the initial value μ0 and updated by 1 - θ, with $0 < \theta < 1$

$$\left\lceil \frac{1}{=}\log\frac{n\mu^0}{£} \right\rceil. \tag{46}$$

After at most iterations $n\mu \le £$ is obtained.

Proof: The duality gap is determined $n\mu^0$ and after the iteration, it is decreased with *1-θ*. When *k* iterations passes, the duality gap is lesser than ε if

$$\left(1-{}^{-}\right)^k n\mu^0 \le £. \tag{47}$$

Introducing the logarithm of both sides of (46), (47) is obtained

$$k\log\left(1-{}^{-}\right)+\log\left(n\mu^0\right) \le \log £, \tag{48}$$

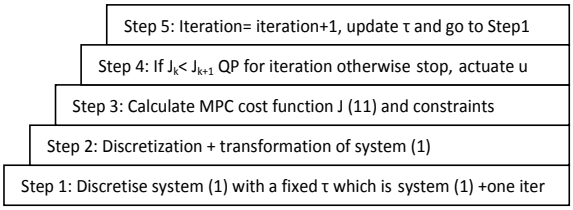
where $-\log(1-\theta) \ge \theta$, this is valid if (49) is true

$$k^{-} \ge \log\left(n\mu^0\right)-\log £ = \log\frac{n\mu^0}{£}. \tag{49}$$

This explains the lemma II in (46).

VII. IMPLEMENTATION OF THE DELAY DEPENDED MPC ON A PROCESSOR

The implementation of the time varying time delay dependent MPC will be as follows:



In the first step the time delay is defined as a fixed time delay which is the total time of discritizing system matrices and calculating the cost function and constraints for quadratic programming problem plus time needed to solve the quadratic problem. In each iteration, it is checked either the calculated delay depended cost is higher than the cost one iteration before. If the total cost still decrease, the QP for given iteration steps is solved. Once the total cost starts to increase instead of decreasing then the solver is stopped and actuate the control signal.

This code implementation can be done both offline and online. In offline version the cost curves of a given system can be calculated and the optimum cost is found for optimum iteration steps for the system. After this point there is no need to iterate further since the total costs start to increase instead of decreasing. In the latter one, during the quadratic programming, the time-varying time delay costs can be checked and compared with the help of sub-codes. Once it is realized that the delay depended costs start to increase the quadratic solver can be interrupted and control signal can be actuated.

A. *Case Study*

H_p – prediction horizon chosen as twelve sampling instant.

H_u – control horizon chosen as four sampling instant.

Plant

$\underline{A}_C = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}$ – system matrix.

$\underline{B}_C = \begin{bmatrix} 1 & 1 \end{bmatrix}$ – system input matrix.

h = 0.20 ms – sampling time.

$\underline{Q}_C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ – weighting sub matrix.

$\underline{N}_C = \begin{bmatrix} 0 & 0 \end{bmatrix}$ – weighting sub matrix.

$\underline{R}_C = \begin{bmatrix} 1 \end{bmatrix}$ – weighting sub matrix

$\underline{J}_C = \begin{bmatrix} [Q] & [N] \\ [N^T] & [R] \end{bmatrix}$ – continuous cost function.

τ = 5 %–50 % of sampling time, time-varying time delay.

$\underline{u}; \bar{u} \rightarrow [-2;4]$ – constraint for the control signal.

$\underline{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$

$\underline{x}_0 = \begin{bmatrix} x_1 & x_2 & u \end{bmatrix}$ – state vector and initial conditions.

$\underline{x}_0 = \begin{bmatrix} 3 & 1 & 0 \end{bmatrix}$ – initial conditions.

A time-varying time delay effect always makes control signal calculations harder since the system moves to another point once the control signal is calculated and introduced to the system. Here with this process which is introduced above our aim is to minimize the time-varying time delay effect for a MPC with constraints so that the calculated control signal will fit to the system. Thus, it is aimed to interrupt the optimization process in MPC with constraints in order to minimize the time-varying time delay which is needed to calculate the control signal.

For the given system above, the discretized cost and the iteration for the optimization are introduced as 5 % time delay to the system. This is increased up to 50 % percent time delay over the optimization process which needs ten iterations to find the minimum. It is seen that the discrete cost function which has the time varying time delay effect starts to increase after the third optimization iteration. After the third optimization iteration, the discretized cost function starts to increase instead of decreasing because of the increasing time delay effect. Here the optimization process interrupted at the third iteration in quadratic programming since the calculated costs are minima at this point. The discritized cost values over the iterations can be seen in Fig. 3. Fig. 4 is given to compare the MPC with a traditional one. In the trajectory following overall performance of a MPC with a pre-interrupted optimization algorithm is faster than traditional LQR. However, this can be the opposite based on single state variables.

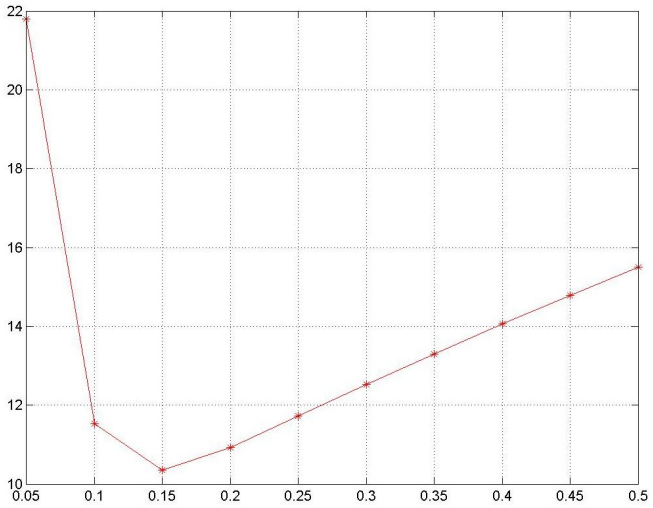


Fig. 3. Discrete cost function J_D vs time varying time delay in MPC.

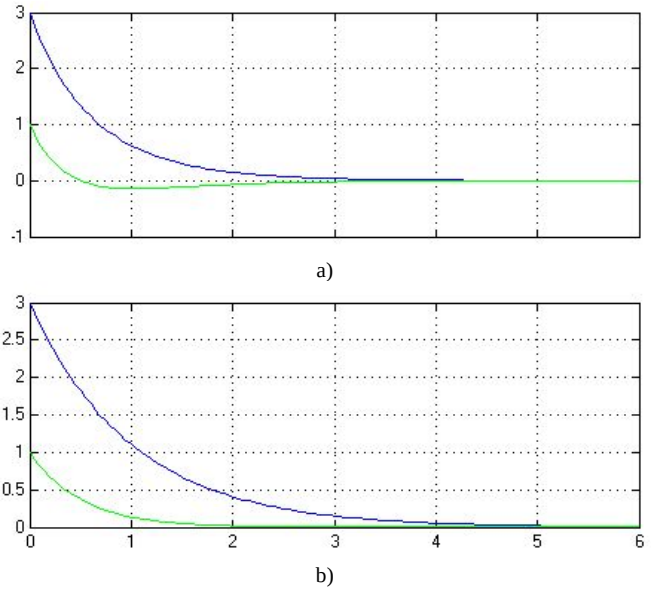


Fig. 4. States (x_1, x_2): traditional MPC (a) versus LQR (b).

The cost decrease is seen as on Table I. On the table, it is seen that during the optimization process cost starts to increase again because of the time varying time delay effect. Here for this system the optimization algorithm can be interrupted at the third iteration.

For a MPC with initial values the output changes of the classical MPC and the proposed one can be seen on Fig. 6.

TABLE I. COST DECREASE IN OPTIMIZATION PROCESS.

Iter num.	J_D MPC	Iter num.	J_D MPC
1	21.9880	6	12.2703
2	11.9092	7	13.3071
3	10.5534	8	14.3183
4	11.2672	9	15.1031
5	11.9631	10	15.8042

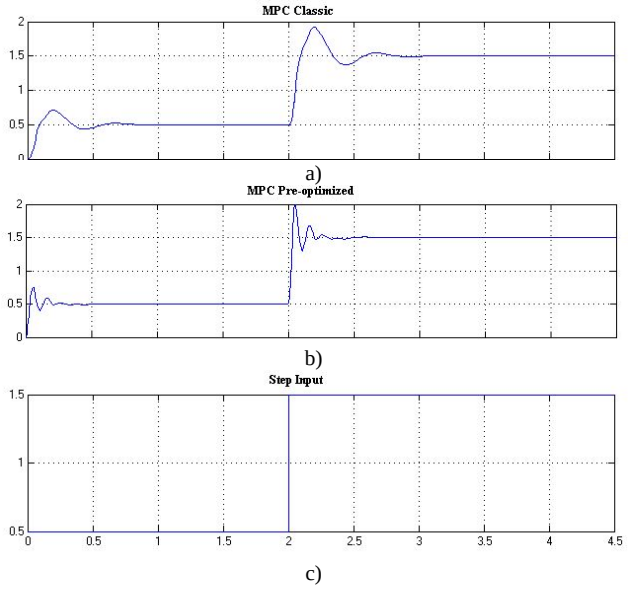


Fig. 5. The output changes of MPC Classic vs. MPC Pre-optimized for step input.

VIII. CONCLUSIONS

In this work it is seen that the cost in an MPC start to increase during the optimization process. Therefore, it is better to interrupt the optimization algorithm where the cost starts to increase instead of decreasing. A control signal found this way may be smaller than a traditional one since the interrupted cost will be smaller than the cost non-interrupted.

In the future, our aim is to research the case with several MPCs with different cost and initial values. This way we intend to find an algorithm to schedule several MPCs with different cost values running on a single processor to increase the overall system performance.

REFERENCES

[1] M. Morari, J. H. Lee, "Model predictive control: past, present and future", *Computers and Chemical Engineering*, vol. 23, pp. 667–682, 1999. [Online]. Available: http://dx.doi.org/10.1016/S0098-1354(98)00301-9

[2] J. M. Maciejowski, *Predictive control: with constraints*. Pearson Education, 2002.

[3] D. Henriksson, A. Cervin, J. Akesson, K. E. Arzen, "Feedback scheduling of model predictive controllers", in *8th IEEE RealTime and Embedded Technology and Applications Symposium*, San Jose, CA, Sept. 2002.

[4] D. Henriksson, A. Cervin, J. Akesson, K. E. Arzen, "On dynamic real-time scheduling of model predictive controllers",.in *41st IEEE Conf. on Decision and Control*, Las Vegas, Nevada USA, Dec. 2002.

[5] G. Pin, M. Filippo, T. Parisini, "Networked MPC for constrained linear systems: a recursive feasibility approach", in *Proc. of the 48th IEEE Conf. on Decision and Control, held jointly with the 28th Chinese Control Conf. (CDC/CCC 2009)*, 15–18 Dec. 2009, pp. 555–560. [Online]. Available: http://dx.doi.org/10.1109/CDC.2009. 5400031

[6] C. F. Caruntu, C. Lazar, "Predictive compensation for network-induced time-varying delays", in *16th Int. Conf. System Theory, Control and Computing (ICSTCC 2012)*, Oct. 2012, pp. 1–6.

[7] F. Valencia, J. D. Lopez, A. Marquez, J. J. Espinosa, "Moving horizon estimator for measurement delay compensation in model predictive control schemes", *50th IEEE Conf. on Decision and Control and European Control (CDC-ECC 2011)*, Dec. 2011, pp. 6678–6683. [Online]. Available: http://dx.doi.org/10.1109/CDC.2011.6161346

[8] Li Jianxiong, Fang Yiming, Shi Shengli, "Robust MPC algorithm for discrete-time systems with time-varying delay and nonlinear perturbations", in *29th Chinese Control Conf. (CCC 2010)*, July 2010, pp. 3128–3133.

[9] S. M. Lee, S. C. Jeong, D. H. Ji, S. C. Won, "Robust model predictive control for LPV systems with delayed state using relaxation matrices", in *American Control Conf. (ACC 2011)*, 2011, pp. 716–721.

[10] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, International Series in Operations Research & Management Science, Springer 2008. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-74388-2

[11] C. Roos *et al*, *Interior Point Methods for Linear Programming*, Second Edition, Springer 2005.

[12] Yinyu Ye, "Further development of the interior algorithm for convex quadratic programming", manuscript, Stanford University and Integrated Systems Inc., Stanford, CA, 1987.

[13] P. A. Absil, A. L. Tits, "Newton-KKT interior-point methods for indefinite quadratic programming", *Computational Optimization and Applications*, vol. 36, no. 1, p. 5, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10589-006-8717-1

[14] M. Sanfridson, "Quality of control and real-time scheduling allowing for time-variations in computer control systems", Ph.D. dissertation, Dept. Machine Design, Royal Inst. Technology Stockholm, Sweden, 2004.

[15] M. Izak, D. Gorges, S. Liu, "On stability and Control of Systems with time varying sampling period and time delay" in *Proc. 7th Symposium on Nonlinear Control Systems (IFAC 2007*), pp. 1056–1061.

[16] K. J. Astrom, B. Wittenmark, *Computer-Controlled Systems: Theory and Design, 2nd ed.*, Prentice-Hall: Englewood Cliffs, NJ, 1990.

[17] *MATLAB version 7.10.0*. The MathWorks Inc., 2010, Natick, Massachusetts.

[18] J. Lofberg, "YALMIP: a toolbox for modeling and optimization in MATLAB", in *IEEE Int. Symposium on Computer Aided Control Systems Design*, 2004, pp. 284–289. [Online]. Available: http://dx.doi.org/10.1109/CACSD.2004.1393890

[19] *Quadratic Programming in C Matlab Interface.* Version 2.0. [Online]. Available: http://sigpromu.org/quadprog/index.html