A Two Phase MPC and its Application to a Grinding Process

Published in:
Conference of Control and Fault-Tolerant Systems (SysTol’10), Nice, France, 6-8 Oct 2010

DOI:
10.1109/SYSTOL.2010.5675949

Published: 01/01/2010

Please cite the original version:
A two phase MPC and its application to a grinding process

Alexey Zakharov, Alexandre Boriouchkine, Sirkka-Liisa Jämsä-Jounela

Abstract—The growing complexity of the control systems and the increased use of nonlinear models cause a dramatic increase in the computational requirements of MPCs. Therefore, more computationally efficient MPC are needed. This paper presents a two-phase MPC approach for decreasing computational demand without sacrificing its efficiency. The first phase of the MPC treats the input variables as independent decision variables of the objective optimization, since the largest part of the objective value arises from a few earliest sampling intervals. In contrast, the second phase combines input variables, defining the rest of the MPC objective value, in an open-loop control which is specified by a few independent decision variables. The method is compared against the traditional Quadratic Programming implementation of an MPC for the Grinding Plant control problem. The two-phase MPC demonstrates a better performance compared with the traditional controller with the same control horizon.

I. INTRODUCTION

CURRENT economic conditions has set new challenges for productivity and keeping the operational conditions of processes within required boundaries. These challenges have resulted in the growing complexity of industrial control systems. Consequently, the increasing complexity of control systems requires more efficient advanced level (higher-level) control strategy which almost universally utilizes model predictive controllers (MPCs). However, the use of MPCs can be computationally heavy because of several factors, such as complex process models, as well as a greater number of manipulated and controlled variables, and constraint management. Nevertheless, a limited computational power forces to make a compromise between efficiency and computational time of a control strategy. However, in several applications the control strategy efficiency is critical. To shorten the computational time without losing efficiency, a modification must be done to existing MPC techniques. Several options exist for improving the efficiency of an MPC. One of such techniques is the dynamic programming, since it provides many useful insights into the MPC performance problems. However, dynamic programming reduces the problem to Hamilton-Jacobi-Bellman (HJB) equation that in most cases can not be solved analytically, because it is a partial differential equation, and solving HJB equation is computationally unattractive due to the high dimensionality. For this reason, Dynamic Programming is not applied directly to practical control problems, but instead, different simplified methods are used.

In the 60s, the development of Dynamic Programming—employing control techniques led to the invention of the linear quadratic regulator (LQR). The LQR is a rare case when the HJB equation can be solved analytically. The LQR was found to be useful for many practical applications and it might be considered as the direct predecessor of modern MPCs (so called ‘zero generation’ of MPC).

Unfortunately, the linear-quadratic regulator is limited by linear dynamics, quadratic objective function and absence of constraints, thus leaving many industrial problems out of its scope. Moreover, it is well acknowledged that economic operating points in typical process units often lie at the intersection of constraints. As a result, a successful industrial controller must keep the system as close to the constraints as possible without actually violating them. Thus, the next generation of MPC appeared having the following main features: linear process constraints, a linear process model, a quadratic objective and a finite time horizon (see [1], [2]). The finite horizon was used to approximate the infinite horizon problem, which hardly can be solved. Since in the presence of constraints even the solution of the finite horizon optimization problem cannot be derived analytically, the quadratic programming was employed to perform the optimization.

In the early 90s, it was discovered that the constrained optimization can cause feasibility problems, especially, when large disturbances appear. Therefore, the most of the modern MPC software products have been enforced to use soft output constraints [3].

On the other hand, because of the finite horizon formulation, MPC faced stability problems. Attempts to achieve stability included different prediction and control horizon approaches and the introduction of a terminal cost to the MPC objective. These methods were criticized in the study [4] as 'playing games', because there were no clear conditions to guarantee stability. Thereby, the stability of MPC was studied actively during the early 90’s ([5] and [6] are among the first papers exploring this question) and a comprehensive review of these studies is provided in [7]. Briefly, the stability is almost universally established through
the use of the value function of MPC as a natural Lyapunov function. On the other hand, the Dynamic Programming provided some useful insights concerning the MPC stability. One example of that is the ‘inverse optimality principle’, which is used to ensure the stability of MPC by utilizing the fake HJB equation (for details, see for example [4], [8]).

It is well known that the performance of MPC depends on the quality of underlying model: an MPC is as good as its model. For that reason during the last decade the focus was moved to the nonlinear MPC utilizing of a more accurate nonlinear process model. Basically, the implementation of such MPC cannot be based on QP anymore. Therefore, the convex optimization techniques are employed instead of QP. On the other hand, some researchers approximate the nonlinear model by a number of linear ones: a multi-model MPC presented for example in [9], and fuzzy MPC developed in [10] is also based on a number of linear process models.

Consequently, the nonlinearity of the models along with other factors, such as the complexity of control systems, increases the computational requirements for MPCs. However, the computational requirements of MPCs are critical for many applications, especially, for large and fast processes. Therefore, many researchers have concentrated their efforts on reducing the number of on-line computations: an explicit MPC have been proposed in [11]. Similarly, the method proposed in [12] explicitly computes the solution according to a rule, which depends on the current state of the system. However, the implementation of both methods is feasible only for low-dimensional systems. Some speed up methods for quadratic programming tasks arising in MPCs are developed in [13] and [14]. Suboptimal MPC implementation, based on modifications of the unconstrained MPC strategy, is considered in [15].

In contrast to computational requirements and stability, another important property of an MPC, namely, optimality did not attract so much attention in the literature, even though the finite-horizon MPCs do not provide the optimal solution of problems. In general, the researchers do not focus on the exploration of optimality because of the idea that a close-to-optimal solution may be found through increasing the control horizon. However, an example is given in [16], where MPC performance is not a non-decreasing function of the optimization horizon. In addition, a longer control horizon also requires more computations and a compromise must be made between the close-to-optimal properties of the controller and its computational demands.

Simultaneously with traditional MPC development, some attempts were made to estimate the solution of HJB equation indirectly. For example, an iterative approach was proposed in [17]. Unfortunately, until today, highly efficient methods based on HJB equation have not been developed and the conclusion was made in [18] that ‘compared with conventional NMPC the computational burden of currently available methods for the HJB successive approximation approach remains remains prohibitive’.

Even HJB equations are unattractive for numerical implementation, the Dynamic Programming appears still to provide a useful insight into the MPC optimality. One example is presented in the work [19], where HJB equation was employed to estimate the ‘degree of suboptimality’ of MPC solutions, which was further used for adaptive determining of the MPC horizon.

Another idea risen from Dynamic Programming, (which present research is focused on) is the desire to have the terminal cost of MPC as close to the value function of the infinite horizon problem as possible [7]. If the value function is employed as a terminal cost, MPC provides the optimal solution even with the time horizon being unity. In particular, the solution of Riccati equation, which is the value function of the unconstrained infinite horizon problem, is proposed as the terminal cost for MPC objective in many papers (see [20], [21]). Although the stability is attained within the approach, in general, it is not possible to expect that a ‘good’ approximation of the value function can be found. In particular, if the MPC setpoint lies on the border of constraints, quadratic functions cannot capture the essential asymmetry of the value function.

In the present paper, the emphasis is moved on the estimation of the value function of the infinite horizon problem, which provides close-to-optimal behaviour of the controller even with a short control horizon. Thus, the method achieves a decrease in computational demand without sacrificing its efficiency. An industrial application (a model of a Grinding process) is used to test the developed method.

The paper is organized as follows. Section 2 contains a description of the proposed MPC controller, and Section 3 a description of the grinding process. In Section 4, the results are presented and compared against a QP implementation of MPC, and Section 5 contains the conclusion.

I. DESCRIPTION OF THE TWO-PHASE MPC

A. The idea of the two-phase MPC

In this section, the two-phase method will be presented for a simple linear-discrete-state-space dynamics:

\[ x(k+1) = Ax(k) + Bu(k) \]
\[ y(k+1) = Cx(k+1) \]

where \( x = (x_1, x_2, ..., x_n) \) is the vector of the current state of the system, \( y = (y_1, y_2, ..., y_m) \) is the vector of the system outputs, and \( u = (u_1, u_2, ..., u_l) \) is the vector of the input variables. For the sake of simplicity, it is assumed that there is no noise in neither, the dynamics nor in the measurements, and the state of the system is exactly known. In addition, the process is assumed to have \( M \) linear constraints:

\[ P_i y(k) \leq q_i, i = 1, 2, ..., M \]  

Under dynamics (1) and constraints (2), the optimal setpoint \( y^* \) is usually defined by the higher level of the
control hierarchy (for example at the real time optimization layer). At this setpoint, the steady state of the system \( x^* \) and the optimal steady state control \( u^* \) are defined using dynamics (1).

A typical objective function of an MPC with the control horizon \( N \) has the following form:

\[
J_N(x(0), u) = \sum_{k=0}^{N-1} l(y(k), \Delta u(k)) + F(x(N), u(N-1)) \tag{3}
\]

where different forms of \( l(y, \Delta u) \) may be used in different controllers while the terminal cost \( F(x, u) \) is required to stabilize the controller. Furthermore, in a widely used approach the terminal cost is substituted with the last control action of a control horizon which is applied until the end of a prediction horizon:

\[
F(x(N), u(N-1)) = \sum_{k=N}^{N-1} l(y(k), u(N-1)) \tag{4}
\]

In addition, the solution of Riccati equation can be utilized as the terminal cost.

Alternatively, dynamic programming can guarantee the stability and optimality of the MPC, even with the control horizon of unity, if the value function of infinite horizon problem is used as the terminal cost (5):

\[
V(x(0), u) = \min_J J_\infty(x(0), u) = \min \sum_{k=0}^{\infty} l(y(k), \Delta u(k)) \tag{5}
\]

This can be proven by considering the fact that in dynamic programming theory, the optimal control action may be found as the control minimizing the objective (6).

\[
u^*(x(0), u(-1)) = \arg\min_{u(0)} [l(y(0), \Delta u(0)) + V(x(1), u(0))] \tag{6}
\]

Similarly to dynamic programming, the MPC derives current control action through minimization of an objective. In case of an MPC the terminal cost \( F(x, u) \) in the objective, Equation (3), is similar to the value function \( V(x, u) \) of Equation (6). As a result, MPC approach may be considered as an implementation of dynamic programming method, but with the ‘inaccurate’ approximation of the value function. Thus, a control horizon longer than one must be employed to obtain the satisfactory performance of the MPC. However, if estimation of the value function is accurate, a short control horizon may be selected without sacrificing the MPC performance. Thus, a great decrease in the number of decision variables of MPC optimization can be achieved providing highly efficient MPC.

The paper presents a method for an accurate estimation of the value function \( V(x, u) \). Ideally, if the optimal open-loop control \( u^*(x) \) was known, it could be substituted in to (5) to estimate the function \( V(x(N), u(N-1)) \) accurately. However, the optimal control is unknown and a set \( U \) of second phase open-loop controls must be used to get an acceptable estimation of the value function, defined as follows:

\[
V^U_k(x(N), u(N-1)) = \min_u \sum_{k=N}^{N-1} l(x(k), \Delta u(k)) \tag{7}
\]

where \( K \) is the second phase horizon. Therefore, this paper defines a set of open-loop controls which are determined by a small number of parameters. Thus, the decision variables of MPC objective optimization include the manipulated variables up to the control horizon (the first phase) and the parameters defining the open-loop controls applied after the control horizon (the second phase). Summarizing, the novelty of the approach results from the optimization of the open-loop controls during the second phase.

However, the efficient implementation of the two-phase MPC relies on two aspects. First, in order to ensure accuracy of estimation (7), the set of control strategies \( U \) must be ‘dense’ in a sense that at any point \( x(N), u(N-1) \) a close-to-optimal open-loop control can be found in \( U \). Second, the computational complexity of the MPC controller increases with the increasing number of parameters defining the second phase open-loop controls. Furthermore, increased number of parameters can reduce the benefit of having a shorter control horizon. Therefore, the second phase controls must be described by a small number of parameters while, simultaneously, being ‘dense’ in the sense defined above.

### B. Two sets of second phase open-loop controls

Let us consider the following one-parametric set of functions presented in Fig. 1:

\[
g_\alpha(k) = \alpha \exp(-c_1 \exp(-c_2 k)), \tag{8}
\]

where coefficients \( c_1 \) and \( c_2 \) are fixed. In order to define a open-loop control for the whole MPC, it is needed to define an individual control for each input variable. This can be done in the following way:

\[
u_i(N-k-1) = \left[ u_i(N-1) - u^*_i \right] g_{\alpha(i)}(k) + u^*_i, \quad i = 1, \ldots, \ell \tag{9}
\]

where \( u^*_i \) is the steady state optimal control. Thus the system open-loop control is defined by vector \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_\ell) \), where every element defines the control for the respective manipulated variable.

---

**Fig. 1** One-parametric set of functions \( g_\alpha(k) \) with \( c_1 = 0.2 \) and \( c_2 = 0.1 \).
The proper choice of constants $c_1$ and $c_2$ may be a problem if the described above set of open-loop controls is used. In order to avoid it, another set of second phase open-loop controls is introduced by adding a time scale parameter $\beta$ as follows:

$$s_{a,b}(t) = \alpha \exp(-c_1\beta t) + (1-\alpha) \exp(-c_2\beta t).$$

The open-loop controls of the MPC are constructed similarly to Equation (9):

$$u_i(N+k-1) = \left(u_i(N-1) - u_i^*(k)\right) e_i + u_i^*, i=1,\ldots,l$$

Here different parameters $a_i$ are used for different input variables of the MPC, but a single parameter $\beta$ is used to define the time scale for all input variables. In the present paper, the set of second phase open-loop controls defined by Equation (11) is used for two-phase MPC implementation and testing.

II. DESCRIPTION OF THE GRINDING PLANT

Communition is a huge consumer of electrical power because crushing rocks into powder requires a lot of energy. According to [22], grinding typically accounts for almost 50% of the costs of a concentrator and, as a result, the optimization of grinding mills is an extremely important research topic. The aim of economic optimization is to maximize the feed rate or to achieve the desired particle size distribution, thus making production more profitable. An overview of the control methods of grinding plants is provided in [23].

![Grinding circuit](image)

Fig. 2 Grinding circuit, [20].

The ore is fed to the rod mill and then discharged into the pump sump. The slurry is then fed to a hydrocyclone, where it is separated into the overflow product and a recycled part, which is fed back to the ball-mill (for more details, see [24]). The whole circuit is presented in Fig. 2.

There are two manipulated variables available in the model:

- $u_1$: the rod-mill feed (t/h)
- $u_2$: the pump sump water addition (m$^3$/h)

and four output variables:

- $y_1$: the hydrocyclone overflow density (% solids)
- $y_2$: the fraction of particles smaller than 325 mesh (47Am) in the product (%)
- $y_3$: the tonnage through the ball mill (t/h)
- $y_4$: the pump sump level (%)

Typically the grinding process can be described by means of a transfer function of second order. The transfer function presented in [24] is used to test the MPCs:

$$u_1 \quad u_2$$

$$y_1, G_1(s) = \frac{0.0255(s-560)(s+4500)}{(s+1500)(s+5000)}, \quad G_{12}(s) = \frac{-0.14(s+1050)}{(s+1200)(s+5000)}$$

$$y_2, G_{21}(s) = \frac{-0.2(s-900)(s+5200)}{(s+1750)(s+2000)}, \quad G_{22}(s) = \frac{0.012(s+3950)(s+5000)}{(s+14400)(s+1500)}$$

$$y_3, G_{31}(s) = \frac{138(s+4050)(s+3000)}{(s+2000)(s+4000)}, \quad G_{32}(s) = \frac{-4.2(s-700)(s+5000)}{(s+5000)(s+150s)}$$

$$y_4, G_{41}(s) = \frac{5.749(s+5500)(s+2100)}{(s+1500)(s+5000)}, \quad G_{42}(s) = \frac{-1.962(s+4700)}{(s+4700)}$$

The model is converted into the discrete time state space form (1) with a sample time equals 300s. The system is described with 15 state variables and four controlled variables.

The constraints are defined as product specifications: the hydrocyclone overflow density ($y_1$) must be above 48% to meet the flotation requirements, and must be below 52% to avoid sedimentation problems. Product specification fineness ($y_2$) is defined as 47% of the particles smaller than 47 µm. In order not to overload the ball mill, the throughput ($y_3$) must not exceed 820t/h. The pump sump level ($y_4$) must remain between 15% and 85%. The constraints are defined in the following order: lower and upper constraints on $y_1$, lower and upper constraints on $y_2$, upper constraint on $y_3$, and lower and upper constraints on $y_4$. Thus, matrices $P$ and $q$ take the following form:

$$y_1 \quad y_2 \quad y_3 \quad y_4$$

$$P = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1
\end{bmatrix},$$

$$q = [52%, -48%, 47%, -47%, 820t/h, 85%, -15%]^T.$$

III. COMPARISON OF THE TWO-PHASE MPC AGAINST THE QP IMPLEMENTATION

Next, the two-phase MPC method is compared against the soft-constrained QP implementation of MPC.

In order to achieve smooth trajectories, the following objective function is used for both controllers:
\[
F(x) = \sum_{k=0}^{K} (u(k) - u(k-1))'R(u(k) - u(k-1)) + \\
\sum_{k=1}^{K} \sum_{i=1}^{M} Q_i (\max(P_i,y(k) - q_i,0))^2 + \\
\sum_{k=1}^{K} (y(k) - y^*)'S(y(k) - y^*)
\]

Where diagonal matrices \( R, Q \) and \( S \) are defined as follows:

\[
R = \begin{pmatrix}
25 & 0 \\
0 & 10
\end{pmatrix}
\]

\[
Q_i = Q_2 = 100, Q_3 = Q_4 = 25, Q_5 = Q_6 = Q_7 = 1.
\]

\[
S_{11} = 5, S_{22} = 0, S_{13} = 0.01, S_{44} = 0.05
\]

To test the controller’s ability to follow changing operating conditions, the constraints are varied as shown in Table I. In every case, the optimal steady states, presented in Table II, are obtained by maximizing the throughput of the plant.

The prediction horizon \( K \) in QP formulation of MPC is always 10 steps longer than the control horizon \( N \). For the two-phase MPC the second phase length is also taken to be 10 steps. QP MPC is tested with control horizons equal to 2 and 10 and two-phase MPC is tested with control horizon equal to 2.

The second-phase open-loop control set described by equation (11) is used for the two-phase MPC implementation. Since there are only two input variables, a second-phase control is defined by three parameters: \( \alpha_1 \) and \( \alpha_2 \) select the shape of the open-loop controls of the first and the second input respectively, while a parameter \( \beta \) is used for time scaling. Discrete values of \( \alpha_1 \) and \( \alpha_2 \) are considered with a step equal 0.4 while \( \beta \) is taken in the following form: \( \beta = 0.9^k \) with \( k \) having only integer values.

**TABLE I**

<table>
<thead>
<tr>
<th>Time period (h)</th>
<th>Overflow solids (%)</th>
<th>Particles &lt; than 47µm (%)</th>
<th>Ball mill throughput (t/h)</th>
<th>Pump sump level (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>0...0.25</td>
<td>48</td>
<td>52</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>0.25...5</td>
<td>48</td>
<td>52</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>5...10</td>
<td>50</td>
<td>52</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>10...15</td>
<td>50</td>
<td>52</td>
<td>47</td>
<td>47</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>Time period (h)</th>
<th>Overflow solids (%)</th>
<th>Particles &lt; than 47µm (%)</th>
<th>Ball mill throughput (t/h)</th>
<th>Pump sump level (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>51.83</td>
<td>47.0</td>
<td>820.0</td>
<td>62.06</td>
</tr>
<tr>
<td>0.25...5</td>
<td>48.0</td>
<td>49.0</td>
<td>811.95</td>
<td>64.5</td>
</tr>
<tr>
<td>5...10</td>
<td>50.0</td>
<td>49.0</td>
<td>739.32</td>
<td>30.87</td>
</tr>
<tr>
<td>10...15</td>
<td>51.83</td>
<td>47.0</td>
<td>820.0</td>
<td>62.06</td>
</tr>
</tbody>
</table>

Two controlled variables are compared in Fig. 3 and the manipulated variables are presented in Fig. 4. In general it is clear that the two-phase MPC with control horizon \( N = 2 \) demonstrates behaviour, which is closer to QP MPC with
longer control horizon \( N = 10 \) rather than the same control horizon \( N = 2 \).

The efficiency of the presented methods may be evaluated using the objective function values of every method presented in Table III. Again, two-phase MPC achieves the same value of the objective function as the QP MPC with the longer prediction horizon \( N = 10 \), whereas the same control \( N = 2 \). QP MPC with the short control horizon \( N = 2 \) demonstrates 13% larger objective value.

\[
\begin{array}{|c|c|c|c|}
\hline
& \text{QP MPC, } N = 2 & \text{QP MPC, } N = 10 & \text{MPC 2 phase, } N = 2 \\
\hline
\text{Constraint 1} & 971 & 978 & 993 \\
\text{Constraint 2} & 1845 & 1358 & 1392 \\
\text{Constraint 3} & 0 & 0 & 0 \\
\text{Constraint 4} & 0 & 0 & 0 \\
\text{Setpoint 1} & 496 & 446 & 468 \\
\text{Setpoint 2} & 1326 & 1106 & 1139 \\
\text{Setpoint 3} & 1039 & 889 & 922 \\
\text{Control 1} & 827 & 886 & 781 \\
\text{Control 2} & 1037 & 1003 & 968 \\
\text{Objective} & 7543 & 6669 & 6665 \\
\hline
\end{array}
\]

IV. CONCLUSION

A two-phase MPC controller is described in the paper, where the best second phase open-loop control is chosen from a predefined set of open-loop controls. Thus, a more accurate estimation of the value function is used as a terminal cost in comparison with the QP formulation. The numerical comparison against the QP formulation of MPC has shown that similar performance might be achieved with a shorter control horizon. Thereby a progress in the trade-off between the performance and computational demands of MPC is made.

REFERENCES