LONG HORIZON MODEL PREDICTIVE CONTROL FOR NONLINEAR INDUSTRIAL PROCESSES

A.A. Tiagounov[*], **J. Buijs**[†], **S. Weiland**[*], **B. De Moor**[†]

[*] Department of Electrical Engineering,
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven,
The Netherlands
a.tiagounov@tue.nl, fax: +31 40 243 45 82
[†] Department of Electrical Engineering, ESAT-SISTA,
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10,
3001 Heverlee (Leuven), Belgium

Keywords: Model predictive control, quadratic programming.

Abstract

This paper considers an MPC algorithm for nonlinear plants. The MPC problem amounts to solving a quadratic programming problem. A structured interior-point method (IPM) is proposed to solve the MPC optimization problem so as to obtain a feasible computational effort for longer prediction horizons. We compare the proposed method with standard QP solvers. The choice of the QP optimizer is investigated for two nonlinear industrial cases, namely an evaporation process and a high-purity distillation column. The effectiveness of the structured IPM is demonstrated for long horizon MPC controller design of the first process.

1 Introduction

Model Predictive Control (MPC), also known as moving or receding horizon control, has originated in industry as a real-time computer control algorithm to solve linear multi-variable control problems subject to constraints and time delays. MPC has received a great deal of attention and receives an ever growing interest for applications in industrial process control. Various MPC algorithms differ mainly in the type of model used to represent the process and its disturbances, as well as the cost functions to be minimized subject to constraints. In this paper we consider a nonlinear MPC scheme where the predicted future process behaviour is represented as a cumulative effect of a nonlinear prediction component and a component based on linear models defined along the predicted trajectory [2, 3, 11]. The first component constitutes a future output prediction using nonlinear simulation models, given past process inputs and measured disturbance signals. The second component uses linearized models for prediction of future process outputs as required for calculation of optimum future process inputs that bring the process behavior closest to the desired behavior. The online optimization can be typically reduced to either a linear program or a quadratic program (QP). The MPC controller solves online a constrained optimization problem and determines optimal control inputs over a fixed future time-horizon, based on the predicted future behaviour of the process, and based on the desired reference trajectory. Certain problems may require longer prediction horizons to cover a broader range of dynamics, due to performance specifications or for stability reasons [4, 9]. Sometimes it is rather difficult to implement long horizon MPC controllers online, since the optimization programs (typically constrained QP's) tend to become too large to be solved in real-time when standard QP solvers are used. To reduce computational complexity we propose to solve the QP problem using a structured interior-point method [1, 8]. The computational cost of this approach is linear in the horizon length, whereas standard approaches have a computational cost which is cubic in the horizon length. It will be shown that in some cases the structured method can solve MPC problem much faster for longer prediction horizons than standard QP solvers. The effectiveness of the proposed method will be demonstrated for an industrial evaporation process. At the same time the structured IPM will not prove to be a good choice for a distillation column.

2 Nonlinear MPC algorithm

Consider the system equations of a nonlinear process model

$$\dot{x} = f(x,u), u \in \mathbb{R}^{n_u} \tag{1}$$
$$y = g(x), y \in \mathbb{R}^{n_y}. \tag{2}$$

For digital MPC controller design the control signal can be assumed to be constant between the sampling intervals. We define the one-step ahead prediction of the states as

$$x(k+1|k) = F_T(x(k|k), u(k)), \tag{3}$$

where $F_T(x(k|k), u(k))$ denotes the terminal state vector obtained by integrating (1) for one sample interval T with the initial condition $x(k|k)$ and constant input $u(k)$.

We consider an MPC problem which amounts to finding an optimal control sequence $\{u(k+j)\}_{j=0}^{N-1}$ at the current moment

k minimizing the performance index

$$J_k(u) = \sum_{j=1}^{N} \Big[\big\| y(k+j) - y_{\text{ref}}(k+j) \big\|_Q^2 + \tag{4}$$

$$\big\| u(k+j-1) - u_{\text{ref}}(k+j-1) \big\|_R^2 \Big]$$

subject to the constraints

$$y_{\min} \leq \quad y(k+j) \quad \leq y_{\max} \tag{5}$$

$$u_{\min} \leq \quad u(k+j) \quad \leq u_{\max} \tag{6}$$

$$\Delta u_{\min} \leq \quad \Delta u(k+j) \quad \leq \Delta u_{\max} \tag{7}$$

for all k and for $j = 0, \ldots, N-1$, where $\Delta u(k+j) = u(k+j) - u(k+j-1)$. Here, $\|x\|_Q^2$ will mean $x^\top Q x$. The MPC is trying to make inputs and outputs follow its reference trajectories $u_{\text{ref}}(k+j)$ and $y_{\text{ref}}(k+j)$. The design parameters include the prediction horizon N and positive semi-definite weighting matrices $Q \in \mathbb{R}^{n_y}$, $R \in \mathbb{R}^{n_u}$.

More generally we define

$$F_{jT}(x(k|k), \{u(k+i)\}_{i=0}^{j-1}) = $$
$$F_T(\ldots(F_T(x(k|k), u(k)), \ldots), u(k+j-1))$$

as j compositions of F_T to represent the terminal states obtained by integrating (1) for j sampling intervals with initial condition $x(k|k)$ and piecewise constant input. Note from (3) that the state $x(k+1|k)$ is related to the undecided manipulated variable $u(k)$ through nonlinear integration. This makes the optimization required for the input move computation a nonlinear problem. To prevent this we further approximate the equation by linearizing $F_T(x(k|k), u(k))$ at some nominal input value $u_{\text{nom}}(k)$ (its choice will be explained later):

$$x(k+1|k) \approx$$
$$F_T(x(k|k), u_{\text{nom}}(k)) + B_k(u(k) - u_{\text{nom}}(k)), \tag{8}$$

where B_k represents a discretized version of one of the following Jacobians:

$$A_k^{\text{c}} = \left.\frac{\partial f}{\partial x}\right|_{x(k|k), u_{\text{nom}}(k)} \qquad B_k^{\text{c}} = \left.\frac{\partial f}{\partial u}\right|_{x(k|k), u_{\text{nom}}(k)}.$$

We can generalize this idea to develop multi-step predictions. Note from (3) that

$$x(k+2|k) = F_T(x(k+1|k), u(k+1)), \tag{9}$$

where $x(k+2|k)$ is related in a nonlinear fashion not only to $u(k+1)$, but also to $u(k)$ appearing in expression (3) for $x(k+1|k)$. By appropriate linearization, we would like to derive an approximation that is linear with respect to the undecided inputs $u(k)$ and $u(k+1)$. The linear relationship that approximates the local behavior can be obtained by linearizing the expression $F_T(x(k+1|k), u(k+1))$ with respect to $x(k+1|k) = F_T(x(k|k), u_{\text{nom}}(k))$ and $u(k+1) = u_{\text{nom}}(k+1)$ as follows

$$F_T(x(k+1|k), u(k+1)) \approx F_{2T}(x(k|k), u_{\text{nom}})$$
$$+A_{k+1}(x(k+1|k) - F_T(x(k|k), u_{\text{nom}}(k)))$$
$$+B_{k+1}(u(k+1) - u_{\text{nom}}(k+1)), \tag{10}$$

where A_{k+1} and B_{k+1} represent discretized versions of the following Jacobians:

$$A_{k+1}^{\text{c}} = \left.\frac{\partial f}{\partial x}\right|_{x=F_T(x(k|k), u_{\text{nom}}(k)), u_{\text{nom}}(k+1)}$$

$$B_{k+1}^{\text{c}} = \left.\frac{\partial f}{\partial u}\right|_{x=F_T(x(k|k), u_{\text{nom}}(k)), u_{\text{nom}}(k+1)}.$$

Note that u_{nom} is a piecewise constant input taking, for instance, values of $\{u_{\text{nom}}(k), u_{\text{nom}}(k+1)\}$ at the time interval $[k, k+2]$.

Note from (8) that $x(k+1|k) - F_T(x(k|k), u_{\text{nom}}(k)) \approx B_k(u(k) - u_{\text{nom}}(k))$. Substitute the affine approximation (10) into (9) to obtain that

$$x(k+2|k) \approx F_{2T}(x(k|k), u_{\text{nom}})$$
$$+ \quad \begin{bmatrix} A_{k+1}B_k & B_{k+1} \end{bmatrix} \begin{bmatrix} u(k) - u_{\text{nom}}(k) \\ u(k+1) - u_{\text{nom}}(k+1) \end{bmatrix}.$$

Carrying out the same derivations for $x(k+j|k)$, $j = 1, \ldots, N$, we obtain

$$x(k+j|k) \approx F_{jT}(x(k|k), u_{\text{nom}}))$$
$$+ \left[\prod_{i=1}^{j-1} A_{k+i}B_k \quad \prod_{i=2}^{j-1} A_{k+i}B_{k+1} \quad \ldots \quad B_{k+j-1}\right]$$
$$\times \begin{bmatrix} u(k) - u_{\text{nom}}(k) \\ u(k+1) - u_{\text{nom}}(k+1) \\ \vdots \\ u(k+j-1) - u_{\text{nom}}(k+j+1) \end{bmatrix}, \tag{11}$$

where A_{k+i} and B_{k+i} represent discretized versions of the following Jacobians:

$$A_{k+i}^{\text{c}} = \left.\frac{\partial f}{\partial x}\right|_{x=F_{iT}(x(k|k), u_{\text{nom}}), u_{\text{nom}}(k+i)}$$

$$B_{k+i}^{\text{c}} = \left.\frac{\partial f}{\partial u}\right|_{x=F_{iT}(x(k|k), u_{\text{nom}}), u_{\text{nom}}(k+i)}. \tag{12}$$

Note that the expression (11) requires integration of the ODE (1) for j sample time steps into the future and computation of the matrices for each of the j sample times (i.e., A_{k+i} and B_{k+i} for $i = 0, \ldots, j-1$). To reduce the computational complexity, the matrices A_{k+i} and B_{k+i} are often kept constant at the initial values of A_k and B_k throughout the prediction horizon. To avoid the computational complexity, we will adapt this simplification. Hence, (11) simplifies to

$$x(k+j|k) \approx F_{jT}(x(k|k), u_{\text{nom}}) \tag{13}$$
$$+ \left[A_k^{j-1}B_k \quad A_k^{j-2}B_k \quad \ldots \quad B_k\right]$$
$$\times \begin{bmatrix} u(k) - u_{\text{nom}}(k) \\ u(k+1) - u_{\text{nom}}(k+1) \\ \vdots \\ u(k+j-1) - u_{\text{nom}}(k+j+1) \end{bmatrix}.$$

Similar techniques as described above can be found in, for example [3, 10].

In order to develop a prediction for the output that is linear with respect to the undecided input moves, we linearize equation (2) with respect to $x(k|k)$:

$$y(k) \approx g(x(k|k)) + C_k(x(k) - x(k|k)),$$

where C_k is a Jacobian matrix defined as

$$C_k = \left.\frac{\partial g}{\partial x}\right|_{x(k|k)}.$$

Carrying out with the same idea,

$$\begin{aligned} y(k+j|k) &= g(x(k+j|k)) \approx g(x(k|k)) \\ &+ C_k(x(k+j|k) - x(k|k)). \end{aligned} \tag{14}$$

Define the so-called optimizing control input as

$$\delta u(k+j) = u(k+j) - u_{\text{nom}}(k+j)$$

and combine (14) with the optimal multi-step prediction equation (13) for $x(k+j|k)$ to obtain

$$\begin{aligned} \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+N|k) \end{bmatrix} &= \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} (g(x(k|k)) - C_k x(k|k)) \\ &+ \begin{bmatrix} C_k F_T(x(k|k), u_{\text{nom}}) \\ C_k F_{2T}(x(k|k), u_{\text{nom}}) \\ \vdots \\ C_k F_{NT}(x(k|k), u_{\text{nom}}) \end{bmatrix} \\ &+ \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g_N & g_{N-1} & \cdots & g_1 \end{bmatrix} \begin{bmatrix} \delta u(k) \\ \delta u(k+1) \\ \vdots \\ \delta u(k+N-1) \end{bmatrix}, \end{aligned} \tag{15}$$

where $g_j = C_k A_k^{j-1} B_k, j = 1, \ldots, N$ represent the Markov parameters. Note that $F_{NT}(x(k|k), u_{\text{nom}})$ can be computed recursively since

$$\begin{aligned} & F_{NT}(x(k|k), \{u_{\text{nom}}(k+i)\}_{i=0}^{N-1}) = \\ & \quad F_T(F_{(N-1)T}(x(k|k), \{u_{\text{nom}}(k+i)\}_{i=0}^{N-2}), \\ & \quad u_{\text{nom}}(k+N-1)). \end{aligned}$$

Note also that the first term of the right-hand side of (15) drops out if the output vector consists of linear combinations of the state (i.e., $y(k) = C_k x(k)$). In order to keep the notation simple, we will denote (15) in the matrix notation as

$$Y = Y_{\text{nom}} + G_{\text{u}} \delta U.$$

Note that Y_{nom} can be computed from the state estimate $x(k|k)$ by performing an integration of the nonlinear ODE. Matrix G_{u} must be recomputed at each time step based on the updated Jacobian matrices.

Remark 1 The local linearization in (11) makes sense only when the computed inputs $\{u(k+i)\}_{i=0}^{j-1}$ do not deviate much from u_{nom}. For nonlinear models this can be achieved by finding a nominal trajectory $u_{\text{nom}}(k+j|k)$ which is as close as possible to the optimal strategy $u_{\text{opt}}(k+j|k)$. A simple but effective choice is to start with $u_{\text{nom}}(k+j|k) = u_{\text{opt}}(k+j|k-1)$, i.e. the optimal control policy derived at the previous sample. Naturally, the input trajectories computed in the subsequent optimization is likely to be a better approximation of the actual future input sequence [2, 11].

Remark 2 Note that a simple relationship exists between the control actions Δu and δu:

$$\begin{aligned} \Delta u(k) &= u(k) - u(k-1) \\ &= u_{\text{nom}}(k) + \delta u(k) - u(k-1) \\ \Delta u(k+1) &= u(k+1) - u(k) \\ &= u_{\text{nom}}(k+1) + \delta u(k+1) \\ &- u_{\text{nom}}(k) - \delta u(k) \\ &\text{etc.} \end{aligned}$$

Then

$$\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N-1) \end{bmatrix} = E \begin{bmatrix} \delta u(k) \\ \delta u(k+1) \\ \vdots \\ \delta u(k+N-1) \end{bmatrix} + F,$$

with

$$E = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ -I & I & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \cdots & -I & I \end{bmatrix}, F = EU_{\text{nom}} - \begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $U_{\text{nom}} = \begin{bmatrix} u_{\text{nom}}(k) & u_{\text{nom}}(k+1) & \ldots & u_{\text{nom}}(k+N-1) \end{bmatrix}^\top$. The latter relationship will be used to formalize constraints on input increments.

Once U_{nom} has been specified, the objective function (4) becomes a quadratic form in δU:

$$\begin{aligned} & J(\delta U) = \\ & \quad (Y_{\text{nom}} + G_{\text{u}}\delta U - Y_{\text{ref}})^\top \bar{Q} (Y_{\text{nom}} + G_{\text{u}}\delta U - Y_{\text{ref}}) \\ & \quad + (U_{\text{nom}} + \delta U - U_{\text{ref}})^\top \bar{R} (U_{\text{nom}} + U - U_{\text{ref}}), \end{aligned}$$

where $\bar{Q} = \text{diag}(Q, \ldots, Q) \in \mathbb{R}^{Nn_{\text{y}} \times Nn_{\text{y}}}$ and $\bar{R} = \text{diag}(R, \ldots, R) \in \mathbb{R}^{Nn_{\text{u}} \times Nn_{\text{u}}}$. The objective can easily be transformed into the standard quadratic cost index

$$\frac{1}{2} U^\top H U + f^\top U + c, \tag{16}$$

where

$$\begin{aligned} H &= 2(G_{\text{u}}^\top \bar{Q} G_{\text{u}} + \bar{R}) && (17) \\ f &= G_{\text{u}}^\top \bar{Q}(Y_{\text{nom}} - Y_{\text{ref}}) + \bar{R}(U_{\text{nom}} - U_{\text{ref}}) && (18) \\ c &= (Y_{\text{u}} - Y_{\text{ref}})^\top \bar{Q} (Y_{\text{u}} - Y_{\text{ref}}) \\ &+ (U_{\text{nom}} - U_{\text{ref}})^\top \bar{R} (U_{\text{nom}} - U_{\text{ref}}). && (19) \end{aligned}$$

Then the optimization problem amounts to solving a quadratic programing problem with the objective function (16) with (17)–(19) subject to the constraints (5)–(7). The constraints can be transformed into the following form

$$\begin{bmatrix} E \\ -E \\ I \\ -I \\ G \\ -G \end{bmatrix} \delta U \leq \begin{bmatrix} b_1 - F \\ -b_2 + F \\ d_1 - U_{\text{nom}} \\ -d_2 + U_{\text{nom}} \\ y_1 - Y_{\text{nom}} \\ -y_2 + Y_{\text{nom}} \end{bmatrix}, \tag{20}$$

where b_1, b_2, d_1 and d_2 are of dimension Nn_{u} and consist of N copies of Δu_{max}, Δu_{min}, u_{max} and u_{min} respectively. In the same way, vectors y_1 and y_2 are of dimension Nn_{y} and consist of N copies of y_{max}, y_{min}.

3 Quadratic Optimization: Structured IPM

The constrained quadratic optimization problem formulated in section 2 can be solved in various ways. First of all, one can choose to eliminate the states and solve (4). If the states are eliminated, one can either use an active set method or an interior-point method [5, 7]. In this section we shortly discuss the properties of the various algorithms. In particular, active set methods (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. The set of equality constraints consists of the guess at that iteration of which constraints will be active in the optimum. This means that in each iteration a dense set of equations in Nn_{u} variables has to be solved. Moreover, the total number of iterations will increase with the number of active constraints since typically in each iteration only one or some constraints will become active. This leads to the fact that ASMs (such as MATLAB's "quadprog" routine), though still widely used as the standard methods for solving QPs in the MPC algorithms, may lead to very expensive combinatorial procedures if applied to large MPC optimization problems. At the same time we may be able to provide ASMs initial guesses of the active set or the optimal point to the algorithm (hot starting) based on the previous MPC solution.

Interior-point methods (IPM) try to solve the non-linear set of Karush-Kuhn-Tucker equations iteratively by making linear approximations to this set. An advantage over ASMs is that interior-point methods (such as Mosek) can efficiently make use of sparsity, e.g. in the constraints, and will therefore often be faster for solving MPC related problems, since bound or rate of change constraints on the control inputs give rise to sparse constraint matrices. 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 than ASMs. A main disadvantage is that they are less suited to use a priori information and difficulties may arise finding a good starting point.

The advantage of the methods with state variables eliminated is the smaller number of decision variables. The major problem is that solving QPs with these methods typically requires a computational time that increases with the *third power* of the number of variables. An other way to solve the constrained quadratic optimization problem is to retain the states as optimization variables. The advantage of this approach is that for MPC related problems the structure in the given QP, originating from the system dynamics, can be exploited to develop methods that have a computational time increasing *linearly* with the horizon length. Sometimes it may be difficult to solve the QP optimization problem with long prediction horizon in the given time using standard optimization techniques, so we propose an other method that will give a faster solution for some specific cases. This algorithm is trying to exploit the relation between the optimization variables as given by the model equations, in order to reduce the number of operations in the optimization. We again consider the nonlinear MPC optimization problem with the quadratic cost function at time k:

$$\min_{\{\delta u(j)\}_{j=0}^{N-1}} \sum_{j=1}^{N} \Big[\big\| (y_{\mathrm{nom}}(j) + y(j) - y_{\mathrm{ref}}(j)) \big\|_Q^2 + \big\| (u_{\mathrm{nom}}(j-1) + \delta u(j-1) - u_{\mathrm{ref}}(j-1)) \big\|_R^2 \Big] \tag{21}$$

subject to

$$\begin{aligned} C_c(k+j) &\geq C_{u(k+j)}\delta u(k+j) + C_{x(k+j)}x(k+j), \\ x(k+j+1) &= A_{k+j}x(k+j) + B_{k+j}\delta u(k+j), \\ y(k+j) &= C_{k+j}x(k+j) + D_{k+j}\delta u(k+j) \end{aligned}$$

for $j = 0, \ldots, N-1$. Note that constraints (20) can easily be rewritten in the given above form. Since QP comes from the dynamic MPC optimization problem, it is clear that the state space equations give a relation between the optimization variables. Moreover, considering all the constraints and the cost function, it is clear that there are typically only constraints and cost terms connecting variables at time step j with variables at time step $j-1$ or $j+1$. Once the state space equations are used to eliminate the states, the number of variables is highly reduced, but then all variables are interconnected since the state space equations were used to write every state $x(k)$ as a function of all inputs up to k. If this elimination is not carried out, the structure given by the dynamics of the plant (i.e. states and inputs at time step k only interact with states and inputs of the nearest time steps) will reflect in the KKT equations, as can be seen in [8]. The result is a linear set of equations which is much larger than for standard methods, but having a banded structure. To solve this new set of linear equations more efficiently than the smaller non-structured set, the interior-point optimizer exploits the banded structure by following a discrete Riccati recursive scheme at each iteration. The details of the optimization algorithm can be found in [1, 8].

4 Design case studies

Example 1 The first nonlinear process is based on the forced-circulation evaporator described in [6], and shown in Figure 1. A feed stream enters the process at concentration X_1 and temperature T_1, with flow rate F_1. It is mixed with a recirculating liquor, which is pumped through the elevator at flow rate F_3. The evaporator itself is a heat exchanger, which is heated by steam flowing at a rate F_{100}, with entry temperature T_{100} and pressure P_{100}. The mixture of feed and recirculating liquor boils inside the heat exchanger, and the resulting mixture of vapor and liquid enters a separator, in which the liquid level is L_2. The operating pressure inside the evaporator is P_2. Most of liquid from the separator becomes the recirculating liquor. A small portion of it is drawn off as product, with concentration X_2, at a flow rate F_2 and temperature T_2. The vapor from the separator flows to a condenser at flow rate F_4 and temperature T_3, where it is condensed by being cooled with water flowing at a rate F_{200}, with entry temperature T_{200} and exit temperature T_{201}. State variables L_2, X_2 and P_2 are the controlled outputs for this process. The manipulated variables are chosen to be

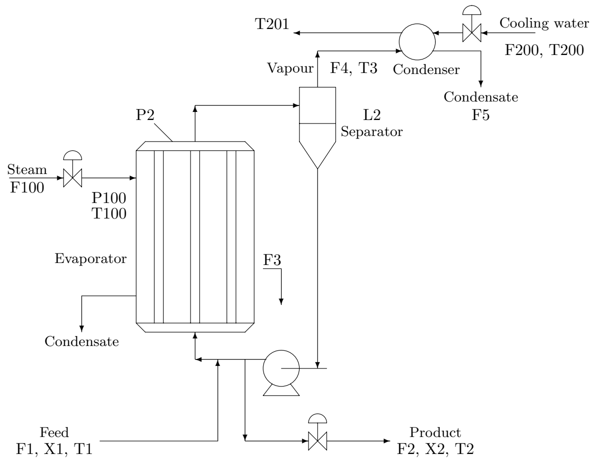

Figure 1: *Evaporation process.*

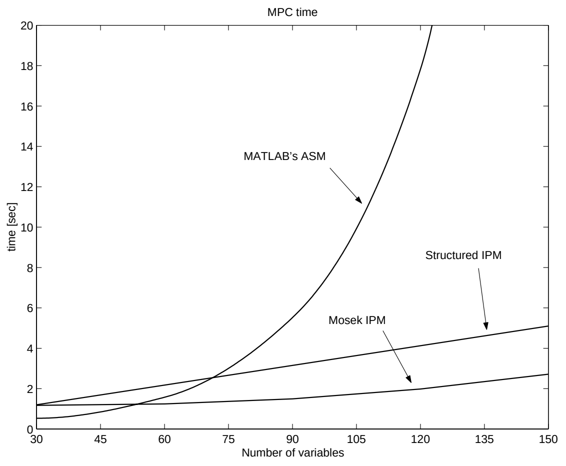

Figure 3: *MPC time comparison for the evaporation process.*

F_2, P_{100} and F_{200}. There are five disturbance signals, namely F_3, F_1, X_1, T_1 and T_{200}, which are left fixed at their equilibrium values. Input and output constraints are imposed in the optimization. Figure 2 shows the controlled outputs after using MPC described in the previous sections. The level L_2 is kept at the value of 1m. The setpoint for X_2 is ramped down linearly from 25% to 15% over a period of 20 minutes, and the operating pressure P_2 is simultaneously ramped up from 50.5 kPa to 70 kPa. Different QP solvers gave good performance results, where the reference trajectories were followed quite closely. Figure 3 gives the comparison of the maximal time required to solve one MPC problem using ASM, Mosek IPM and structured IPM with different number of variables (Nn_{u}). We see that although Mosek is much faster than ASM, these two standard QPs exhibit the behaviour related to the third power of the number of variables. At the same time we also noticed that it took the structured IPM in the MPC controller approximately 5 seconds to solve each optimization problem. The tests also showed that the time increased linearly with the horizon length. For horizons $N > 25$ the structured IPM became faster than MATLAB's ASM, and so it could be implemented online

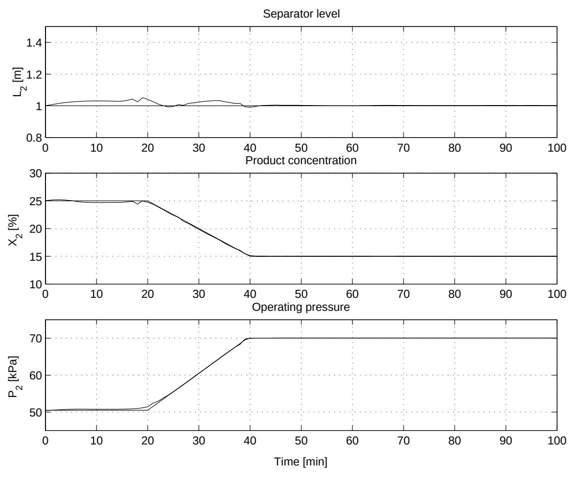

Figure 2: *MPC simulation of the evaporation process.*

with longer horizons, if required, considering the sampling time $T = 1$ min for this process. The structured IPM algorithm also became faster than Mosek for $N > 160$. Thus the evaporation process is an example which can prove the efficiency of the structured IPM for longer prediction horizons in certain cases.

Example 2 The second case study is concerned with MPC controller design for a high purity distillation column. Figure 4 presents the distillation process, its 82-state nonlinear model can be found in [12]. The column contains 41 trays that are

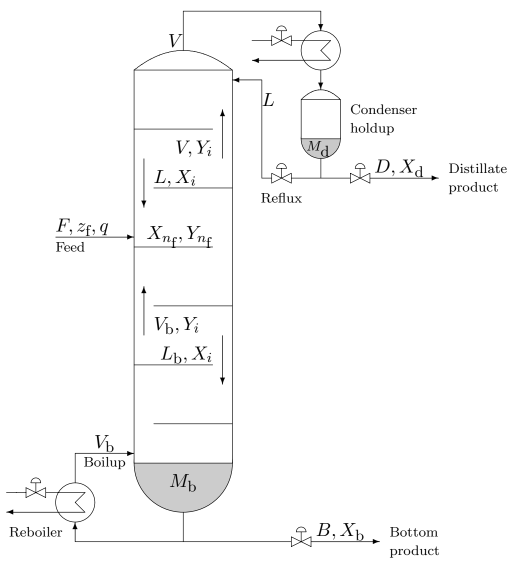

Figure 4: *Binary high-purity distillation column.*

located along its length. The raw material enters the column at a flow rate of F and with composition z_{f}. The top product, the distillate, is condensed and removed at a flow rate of D and with composition X_{d}. The bottom product is removed as a liquid at a flow rate of B and with composition X_{b}. The operation of the column requires that some of the bottom product is reboiled at a rate of V_{b} to ensure the continuity of the vapor flow. In the same way, some of the distillate is refluxed to the top tray at a rate of

L to ensure the continuity of the liquid flow. The vapor boilup

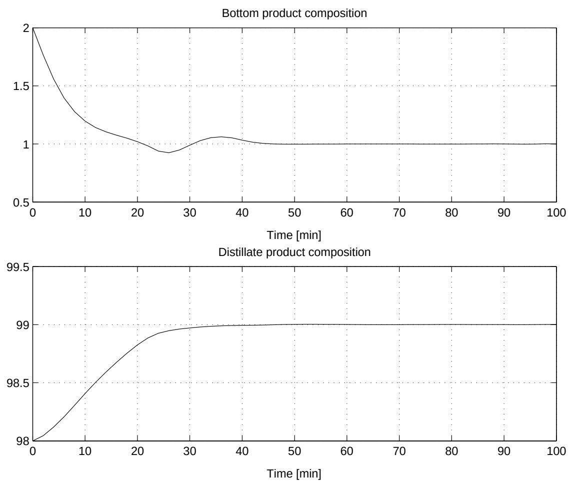

Figure 5: *MPC simulation of the distillation column.*

V and the reflux flow L are the manipulated variables. Feed flow rate F with composition z_{f} act as disturbances. The MPC objective is to bring the controlled outputs X_{b} and X_{d} to the setpoints of 1% and 99%, respectively (see Figure 5). Figure 6

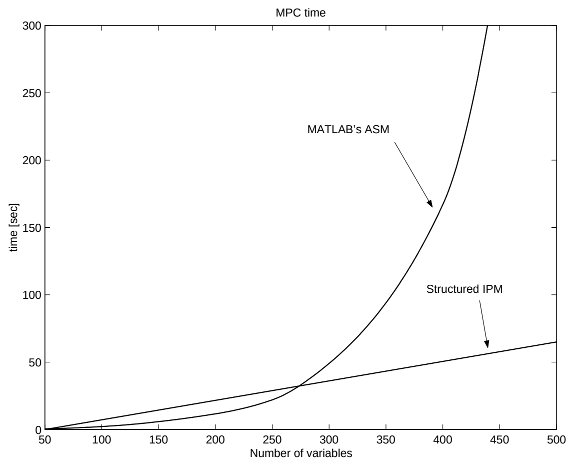

Figure 6: *MPC time comparison for the distillation process.*

gives the comparison of the maximal time required to solve one MPC problem using ASM and the structured IPM with different number of variables (Nn_{u}), where we have similar behavior to the one in the previous example. We also found that it took the structured IPM in the MPC controller approximately 10.6 seconds to solve each optimization problem for the distillation column. We noticed that the structured IPM became faster than MATLAB's ASM only for $N > 140$ and it failed to compete with Mosek even for much longer horizons. This could definitely be attributed to the fact that the algorithm retains states as optimization variables. In this process we have 2 inputs and 82 states which give this big increase in the number of optimization variables. So, the structured IPM is definitely not a good choice for the MPC optimizer of the distillation process.

5 Conclusions

In this paper a structured IPM was proposed to solve the nonlinear MPC problem. The algorithm explicitly takes the structure of the given problem into account such that the MPC controllers will be able to solve a dynamic optimization problem in shorter time. The first example proved that for MPC problems with long horizons, naive implementations of standard QP solvers could be inefficient. So, in this paper we showed that for certain systems, the structured IPM could give a real-time optimal MPC solution within the available time for much longer horizons.

References

[1] J. Buijs, J. Ludlage, W. van Brempt, and B. De Moor. Quadratic programming in model predictive control for large scale systems. In *Proc. IFAC World Congress*, Barselona, Spain, 2002.

[2] R. de Keyser. A gentle introduction to model based predictive control. In *EC-PADI2 Int. Conference on Control Engineering and Signal Processing*, Peru, 1998.

[3] J.H. Lee and N.L. Ricker. Extended kalman filter based nonlinear model predictive control. *Ind. Eng. Chem. Res.*, 33:1530–1541, 1994.

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

[5] S.G. Nash and A. Sofer. *Linear and nonlinear programming*. The McGraw-Hill Companies, Inc., 1996.

[6] R.B. Newell and P.L. Lee. *Applied process control. A case study*. Prentice Hall, 1989.

[7] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer Series in Operations Research, 1999.

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

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

[10] N.L. Ricker and J.H. Lee. Nonlinear model predictive control of the tennessee eastman challenge process. *Computers and Chemical Engineering*, 19(9):961–981, 1995.

[11] L.O. Santos, N.M.C. de Oliveira, and L.T. Biegler. Reliable and efficient optimization strategies for nonlinear model predictive control. In *Proc. of the IFAC Symposium DYCORD+95*, pages 33–38, Helsingor, Denmark, 1995.

[12] S. Skogestad and I. Postlethwaite. *Multivariable feedback control*. Wiley, 1996.