Optimization Algorithms for Bilinear Model–Based Predictive Control Problems

H. H. J. Bloemen
TNO TPD, Instrumentation and Information Systems, 2600 AD Delft, The Netherlands

T. J. J. van den Boom and H. B. Verbruggen
Delft University of Technology, Delft Center for Systems and Control, 2628 CD Delft, The Netherlands

DOI 10.1002/aic.10122
Published online in Wiley InterScience (www.interscience.wiley.com).

Model-based predictive control (MPC) for discrete-time bilinear state–space models is considered. The optimization problem of the bilinear MPC algorithm is nonlinear in general. It is demonstrated that the structural properties of the bilinear state–space model provide a way to formulate the nonlinear optimization problem as a sequence of quadratic programming problems that exactly represent the original objective function. The proposed optimization algorithm is compared to one that is based on a linearization about an input trajectory. To benefit from the advantages of both algorithms, a hybrid algorithm is proposed, which outperforms the other two in most cases. The applicability of the proposed bilinear MPC algorithm is demonstrated on a polymerization process. © 2004 American Institute of Chemical Engineers *AIChE J,* 50: 1453–1461, 2004

Keywords: predictive control, bilinear, discrete time, nonlinear optimization, polymerization

Introduction

Linear model-based control techniques are useful when controlling processes in a limited operating region because the behavior of the process can usually be approximated accurately by a linear model. However, if there is a need to operate the process over a large operating region, often the nonlinear behavior of the process cannot be neglected, and in such cases one has to design a controller based on a nonlinear model that is able to represent the plant behavior.

The class of discrete-time bilinear state–space models forms a subclass of the nonlinear models, and extends the class of linear state–space models with models that contain a product term between the current state and the current input. These models can be represented by

$$x(k+1) = Ax(k) + \left[B + \sum_{p=1}^{n} F_p x_p(k)\right] u(k) \tag{1}$$

$$y(k) = Cx(k) \tag{2}$$

where $x \in \mathbb{R}^n$ is the state, x_p denotes the pth element of the state vector, $u \in \mathbb{R}^m$ is the input, and $y \in \mathbb{R}^l$ is the output. The matrices A, B, F_p, and C are state–space matrices of conformal dimensions. The focus on discrete-time bilinear state–space models is motivated by the recent developments in the area of black-box subspace-identification algorithms for these systems (Chen and Maciejowski, 2000; Favoreel et al., 1999; Verdult et al., 1999).

Model-based predictive control (MPC) is a control methodology in which the input is calculated through a minimization of a performance index, subject to constraints. For an overview on MPC the reader is referred to Allgöwer et al. (1999). Usually a quadratic performance index is chosen that minimizes a trade-off between the deviation of the states x from

Correspondence concerning this article should be addressed to H. H. J. Bloemen at Bloemen@tpd.tno.nl.

© 2004 American Institute of Chemical Engineers

their set points x_s over a prediction horizon H_p in the future, and the deviation of the inputs u from their set points u_s over a control horizon H_c in the future. For steady-state offset free control the pair (u_s, x_s) should match a steady state. Given a set point y_s for the output y, the corresponding steady-state values for (u_s, x_s) can be found by solving the following set of equations

$$x_s = Ax_s + \left(B + \sum_{p=1}^{n} F_p x_{s,p}\right) u_s \tag{3}$$

$$y_s = Cx_s \tag{4}$$

by use of a nonlinear programming algorithm, for example. Thus, the performance index is given by (notation: $\|b\|^2_\Psi = b^T \Psi b$)

$$J(k) = \sum_{j=1}^{H_p} \|x(k+j) - x_s(k+j)\|^2_{Q_x} + \sum_{j=1}^{H_c} \|u(k+j-1) - u_s(k+j-1)\|^2_{R_u} \tag{5}$$

where Q_x and R_u are semipositive and positive definite weighting matrices, respectively. The time dependency of the set points can be caused by knowledge of future set point changes for example. Assume that the input constraints are given by

$$\Gamma_u u(k+j-1) \leq \theta_u \qquad \forall\, j = 1 \cdots H_c \tag{6}$$

Then the constrained bilinear MPC problem is defined as

$$\min_{u(k),\ldots,u(k+H_c-1)} J(k) \tag{7}$$

$$\text{subject to Eqs. 1 and 6} \tag{8}$$

For bilinear models, these optimization problems in general are nonconvex because Eq. 1 serves as a nonlinear equality constraint. A powerful algorithm for solving general nonlinear (constrained) optimization problems is sequential quadratic programming (SQP) [see, for example, Boggs and Tolle (1995) for a review on SQP]. In SQP the original problem is approximated by a quadratic program. Because of the approximating nature, the solution to the quadratic program does not necessarily improve the original performance index. Therefore this solution is used only as a search direction for a subsequent line search. Optimization algorithms that are more specifically directed toward MPC problems use linearized model predictions and constraints to obtain a quadratic approximation of the original nonlinear problem (see, for example, Brooms and Kouvaritakis, 2000; Di Marco et al., 1997; Kouvaritakis et al., 1999). The difference between these and the general SQP algorithm is a different quadratic approximation of the original problem. The use of linearized model predictions ensures that the Hessian (second-order derivative) is positive definite, a property that is not automatically guaranteed by the general SQP algorithm. The latter requires special approximations to enforce this property (see Boggs and Tolle, 1995).

This article focuses on solving the MPC optimization problem for bilinear discrete-time models. A common way of solving such an optimization problem is to use model predictions that are linearized about an input trajectory, as explained above (another alternative could be to use linearizations about the operating point). An outline of this algorithm is given in the next section.

Alternatively, it is shown that the nonlinear MPC optimization problem for discrete-time bilinear models can be solved by exploiting the structure of the bilinear model. This does not involve an approximation of the performance index, but rather involves the search directions of the minimization algorithm. In these specific search directions the nonlinear optimization problem reduces to an exact QP subproblem (that is, it is not approximated). Therefore, the solution of each QP subproblem provides the best update in the corresponding direction, which implies that no line search needs to be performed. Next, a hybrid algorithm that benefits from the advantages of both separate algorithms is presented. The performance of these algorithms is compared by means of a numerical example. The applicability of the proposed bilinear MPC algorithm is demonstrated on a polymerization process. The conclusions are given in the final section.

Optimization Algorithms

In the following three subsections optimization algorithms are presented for the optimization problem as stated in Eqs. 7 and 8. All these optimization algorithms are based on locally minimizing the objective function in certain search directions. Consequently, these algorithms will find a local optimum, which is not guaranteed to be the global optimum. In general, finding the global optimum is a much more difficult task for nonlinear optimization problems, and will be left for future research.

An optimization algorithm based on linearizations about a trajectory

Optimization algorithms that are specifically directed toward solving general MPC problems are those that use a linearization of the model about an input trajectory (Brooms and Kouvaritakis, 2000; Di Marco et al., 1997; Kouvaritakis et al., 1999). The linearized state prediction is then given by

$$\tilde{x} \approx \tilde{x}_{lin} = \tilde{x}_0 + V_{lin} \cdot (\tilde{u} - \tilde{u}_0) \tag{9}$$

where $\tilde{x}$ is a vector of stacked states $x(k+j)$ for $j = 1 \cdots H_p$; $\tilde{u}$ is a vector of stacked inputs $u(k+j-1)$ for $j = 1 \cdots H_c$; $\tilde{x}_{lin}$ is the Jacobian linearization of $\tilde{x}$; and $\tilde{x}_0$ is the vector of state predictions evaluated at $\tilde{u}_0$, which is the input trajectory used for linearization. Replacing the state prediction of the bilinear model by the linearized prediction gives an approximation of the original performance index

$$J(k) \approx J_{lin}(k) = (\tilde{x}_{lin} - \tilde{x}_s)^T \tilde{Q}_x (\tilde{x}_{lin} - \tilde{x}_s) + (\tilde{u} - \tilde{u}_s)^T \tilde{R}_u (\tilde{u} - \tilde{u}_s) \tag{10}$$

where $\tilde{Q}_x$ is the block diagonal matrix $diag(Q_x, \ldots, Q_x)$; $\tilde{R}_u$ is the block diagonal matrix $diag(R_u, \ldots, R_u)$; and $\tilde{x}_s$ and $\tilde{u}_s$ are vectors of stacked values of the set points for the states and inputs, respectively. $J_{lin}(k)$ is quadratic in $\tilde{u}$.

For a constrained bilinear MPC problem, a QP subproblem has to be solved. For an unconstrained bilinear MPC problem, the solution of the QP subproblem can be found analytically. Setting the derivative of $J_{lin}(k)$ (Eq. 10), with respect to $\tilde{u}$, to zero results in the following expression for the optimal value $\tilde{u}^*_{qp}$ (the asterisk denotes the optimal value)

$$\tilde{u}^*_{qp} = -(V^T_{lin}\tilde{Q}_x V_{lin} + \tilde{R}_u)^{-1}[V^T_{lin}\tilde{Q}_x(\tilde{x}_0 - V_{lin}\tilde{u}_0 - \tilde{x}_s) - \tilde{R}_u\tilde{u}_s] \tag{11}$$

We note that $\tilde{u}^*_{qp}$ is not necessarily a better solution than $\tilde{u}_0$ for the original nonlinear optimization problem, given the approximating nature of the QP subproblem (the original nonlinear state prediction is approximated by a linear one). After solving a QP subproblem, a line search in the direction $\tilde{u}^*_{qp}$ is performed to find the best solution $\tilde{u}^*_{ls}$ in the direction of$\tilde{u}^*_{qp}$

$$\tilde{u}^*_{ls} = (1 - \alpha)\tilde{u}_0 + \alpha\tilde{u}^*_{qp} \tag{12}$$

This line search, which is essentially a one-dimensional nonlinear optimization problem in α, can easily be performed by the Golden Section algorithm (Rao, 1978). The Golden Section algorithm iteratively reduces the section for α in which the (local) optimum is located. Infeasible points can be discarded from the section. When the relative difference between the maximal and minimal value of $J(k)$ for the points evaluated in the section is smaller than λ_{ls}, the line search is terminated. The optimization algorithm is terminated when the relative decrease of the performance index is smaller than some specified small positive value λ.

The optimization algorithm is then given by the following steps.

Algorithm 1 (Optimization Algorithm Based on Linearization)

1. Give an initial estimate $\tilde{u}_0$ for the input sequence.
2. Calculate $J_{old}(k)$ (Eq. 5), given the input sequence.
3. (a) Calculate $\tilde{x}_0$ and V_{lin} (Eq. 9).
   (b) Calculate $\tilde{u}^*_{qp}$ by solving the QP subproblem.
   (c) Calculate $\tilde{u}^*_{ls}$ by performing a line search in the direction of $\tilde{u}^*_{qp}$ (Eq. 12). The line search is terminated when the relative difference between the maximal and minimal value of $J(k)$ for the points evaluated in the section of the golden section algorithm is smaller than λ_{ls}.
4. Calculate $J_{new}(k)$ (Eq. 5), given $\tilde{u}^*_{ls}$.
5. If $[J_{old}(k) - J_{new}(k)]/J_{old}(k) > \lambda$, set $J_{old}(k) = J_{new}(k)$, $\tilde{u}_0 = \tilde{u}^*_{ls}$ and go to step 3; otherwise, stop, implement $u_{ls}(k)^*$, shift the horizons by one step, and start from step 1. This is the so-called receding horizon mode of the MPC.

Compared to a general SQP algorithm, algorithm 1 has the advantage that the Hessian is guaranteed to be positive definite. A drawback of both a general SQP algorithm and algorithm 1 is the fact that the QP subproblem is based on an approximation. When this approximation is poor, the search direction generated in the QP step is likely to be poor. In the next section an algorithm is presented that does not suffer from this drawback. Instead of approximating the optimization problem, the algorithm in the next subsection uses specific search directions in which the bilinear MPC optimization problem is convex by itself.

An optimization algorithm based on fixed search directions

Because of the nonlinearity in the state equation of the bilinear model, the predictions of the future states no longer are affine functions of the future inputs. To demonstrate the way in which the nonlinearity of the bilinear model is manifested in the future state predictions, consider for example a single-input bilinear model for notational convenience. Then, Eq. 1 can be written as

$$x(k+1) = Ax(k) + [B + F_t x(k)]u(k) \tag{13}$$

where $F_t = [F_1\ F_2 \cdots F_n]$. Using Eq. 13, the predictions of the future states as a function of the inputs and the current state $x(k)$ are given by

$$x(k+1) = Ax(k) + [B + F_t x(k)]u(k) \tag{14}$$

$$x(k+2) = A^2x(k) + [AB + AF_t x(k) \quad B + F_t Ax(k)] \times \begin{bmatrix} u(k) \\ u(k+1) \end{bmatrix} + (F_t B + F_t^2 x(k))u(k)u(k+1) \tag{15}$$

$$\begin{aligned} x(k+3) = {} & A^3x(k) + [A^2B + A^2F_t x(k) \quad AB + AF_t Ax(k) \\ & B + F_t A^2 x(k)]\begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \end{bmatrix} + [AF_t B + AF_t^2 x(k) \\ & F_t AB + F_t AF_t x(k) \quad F_t B + F_t^2 Ax(k)]\begin{bmatrix} u(k)u(k+1) \\ u(k)u(k+2) \\ u(k+1)u(k+2) \end{bmatrix} \\ & + [F_t^2 B + F_t^3 x(k)]u(k)u(k+1)u(k+2) \end{aligned} \tag{16}$$

and so forth.

The above equations are substituted into the performance index (Eq. 5). The nonconvexity of the performance index results from all the products between the inputs $u(k)\cdots u(k + H_c - 1)$ appearing in the predictions of the state. However, note that the order of every $u(k + j - 1)$ is never larger than one. This property is exploited in the optimization algorithm that is presented in this section. This property also holds for multiple-input systems, given that $u(k + j - 1)$ is multiplied by each of the states $x_p(k + j - 1)$, which depend on the inputs up to $u(k + j - 2)$.

The above observation implies that, if $u(k + i - 1)$ is used as a fixed search direction for the nonlinear optimization algorithm [that is, $u(k + j - 1)\ \forall\ j \neq i$ are kept constant], the performance index (Eq. 5) is quadratic in this specific search direction. Therefore, in this fixed direction the bilinear MPC problem reduces to a QP subproblem. The solution to this QP problem exactly provides the minimum of the nonlinear performance index in the corresponding direction. Iteratively changing the search direction $u(k + i - 1)$ and solving the optimization problem in this direction thus decrease the nonlinear performance index. In this way the structure of the bilinear model is exploited for solving the bilinear MPC problem.

If $u(k + j - 1)\ \forall\ j \neq i$ are kept constant, then the state predictions can be written as

$$\tilde{x} = K_i + V_i u(k + i - 1) \tag{17}$$

where K_i and V_i are matrices depending on $u(k + j - 1)\ \forall\ j \neq i$ and $x(k)$. A numerically simple way for calculating K_i is to calculate $\tilde{x}$ by simulating the bilinear model, setting $u(k + i - 1) = 0$

$$K_i = \tilde{x}[u(k + i - 1) = 0] \tag{18}$$

Calculating $\tilde{x}$ by setting $u_r(k + i - 1) = 1$, then provides a way to compute V_i

$$(V_i)_r = \tilde{x}[u_r(k + i - 1) = 1] - K_i \tag{19}$$

where the subscript r refers to the rth input and the rth column of V_i $(r = 1, \ldots, m)$. Note that Eq. 19 can also be used to compute V_{lin} in Eq. 9: $V_{lin} = [V_1, \ldots, V_{H_c}]$.

Thus, by using Eq. 17, $J(k)$ can be minimized in the direction $u(k + i - 1)$ by solving an (exact) QP subproblem. For an unconstrained bilinear MPC problem the solution of the QP subproblem can be found analytically. Setting the derivative of $J(k)$ (Eq. 5), with respect to $u(k + i - 1)$, to zero results in the following expression for the optimal value $u(k + i - 1)^*$

$$u(k + i - 1)^* = -(V_i^T \tilde{Q} V_i + R)^{-1} \times (V_i^T \tilde{Q}[K_i - \tilde{x}_s] - Ru_s(k + i - 1)) \tag{20}$$

The iterations of successively minimizing $J(k)$ in the direction $u(k + i - 1)$ are terminated when the relative decrease of the performance index is smaller than some specified small positive value λ. Because not all the inputs have the same impact on the value of the performance index, the stop criterion in the algorithm below is evaluated only after successively optimizing once in every direction $u(k + i - 1)\ \forall\ i = 1 \cdots H_c$. The optimization algorithm is given by the following steps.

Algorithm 2 (Optimization Algorithm Based on Fixed Search Directions)

1. Give an initial estimate for the input sequence.
2. Calculate $J_{old}(k)$ (Eq. 5), given the input sequence.
3. For $i = 1 \cdots H_c$:
    (a) Set $u(k + i - 1) = 0$, calculate K_i (Eq. 18).
    (b) For $r = 1, \ldots, m$ set $u_r(k + i - 1) = 1$, $u_t(k + i - 1) = 0\ \forall\ t \neq r$, calculate the rth column of V_i (Eq. 19).
    (c) Calculate $u(k + i - 1)^*$ by solving the QP subproblem.
4. Calculate $J_{new}(k)$ (Eq. 5), given the input sequence.
5. If $[J_{old}(k) - J_{new}(k)]/J_{old}(k)\ \lambda$, set $J_{old}(k) = J_{new}(k)$ and go to step 3; otherwise, stop, implement $u(k)^*$, shift the horizons by one step, and start from step 1.

If $H_c < H_p$, then assumptions concerning the inputs beyond the control horizon are needed. Two popular choices are either $u(k + j - 1) = u_s(k + H_c - 1)\ \forall\ j > H_c$ (that is, equal to the set point for u), or $u(k + j - 1) = u(k + H_c - 1)\ \forall\ j > H_c$. For algorithm 2 only $u(k + j - 1) = u_s(k + H_c - 1)\ \forall\ j > H_c$ applies because, otherwise, the order of $u(k + H_c - 1)$ is larger than one in certain terms appearing in the predictions of the state, in which case the optimization problem in the direction $u(k + H_c - 1)$ no longer is a QP subproblem.

The main advantages of algorithm 2 compared to a general SQP algorithm are as follows:

(1) In every QP step of algorithm 2 the performance index and constraints are exact and not an approximation as in the SQP algorithm.

(2) Because of this, no line search is needed after the QP steps. The solution of the QP step in algorithm 2 provides the optimal value in the search direction.

(3) The Hessian is always positive definite, provided that Q and R are positive semidefinite and positive definite, respectively.

(4) Every QP step is of a low dimension (m) in algorithm 2, whereas in the SQP algorithm it is of dimension $m \cdot H_c$.

The main drawback of algorithm 2 is a slow convergence when the performance index is shaped like a steep valley that is not aligned with one of the directions $u(k + i - 1)$.

A hybrid optimization algorithm

The discussions at the end of the previous subsections motivate the formulation of a hybrid optimization algorithm, given that the advantages and disadvantages of the algorithms 1 and 2 are attributed to different events.

If the initial estimate for the input sequence is far from the optimal solution, then the approximation used in algorithm 1 may be poor, which may result in a slow convergence or in a large computational demand. Because algorithm 2 does not use an approximation, this algorithm does not suffer from an initialization that is far from the optimal solution. An indicator for such an event is the step length α resulting from the line search, step 3c in algorithm 1, which will be small when the search direction $\tilde{u}^*_{qp}$ in algorithm 1 is poor.

If algorithm 2 ends up in a steep valley of $J(k)$, which is not aligned with a specific direction $u(k + i - 1)$, then the convergence of this algorithm is slow. Such an event may happen close to the optimum if the sensitivity of the performance index to a certain combination of degrees of freedom is much larger than to others. Because algorithm 1 performs better in such cases, this motivates a switch to algorithm 1. An indicator for such an event is the improvement in the performance index achieved in one iteration of algorithm 2, which will then be small.

The above discussion motivates the following hybrid optimization algorithm.

Algorithm 3 (A Hybrid Optimization Algorithm)

1. Give an initial estimate for the input sequence and set $s = 0$ to select the optimization algorithm.
2. Calculate $J_{old}(k)$ (Eq. 5), given the input sequence.
3. If $s = 0$, perform step 3 of algorithm 2; otherwise, perform step 3 of algorithm 1.
4. Calculate $J_{new}(k)$ (Eq. 5), given $\tilde{u}^*$.

Table 1. Tuning Parameters in the Optimization Algorithms and Their Values

Parameter	Value	Purpose	Used by Algorithm
λ	0.001	Termination of the optimization algorithm	1, 2, 3
λ_{ls}	0.001	Termination of the line search	1, 3
λ_{fd}	0.1	Switch to algorithm 1	3
α_{min}	0.5	Switch to algorithm 2	3

5. (a) If $s = 0$ and $[J_{old}(k) - J_{new}(k)]/J_{old}(k) \leq \lambda_{fd}$, set $s = 1$, $J_{old}(k) = J_{new}(k)$, and go to step 3.
   (b) If $s = 1$ and $\alpha \leq \alpha_{min}$, set $s = 0$.
6. If $[J_{old}(k) - J_{new}(k)]/J_{old}(k)\ \lambda$, set $J_{old}(k) = J_{new}(k)$ and go to step 3; otherwise, stop, implement $u(k)^*$, shift the horizons by one step, and start from step 1.

Step 5a ensures that the last iteration will always be algorithm 1.

Comparison of the Optimization Algorithms

In this section the performance of the optimization algorithms, presented in the previous section, are compared for a bilinear model specified by the following matrices

$$A = \begin{bmatrix} 0.78 & -0.08 & 0.26 & -0.13 \\ -0.14 & 1.07 & 0.03 & 0.06 \\ 0.27 & 0.08 & 0.60 & 0.01 \\ -0.04 & 0.10 & 0.08 & 1.00 \end{bmatrix} \qquad B = \begin{bmatrix} 0.18 \\ 0.40 \\ 0.85 \\ 0.58 \end{bmatrix} \tag{21}$$

$$F_t = \begin{bmatrix} 0.27 & 0.07 & -0.17 & -0.36 \\ 0.01 & -0.33 & 0.21 & 0.14 \\ -0.02 & 0.18 & 0.31 & -0.10 \\ -0.21 & 0.41 & -0.40 & 0.17 \end{bmatrix} \tag{22}$$

The tuning parameters of the MPC are

$$H_c = 4 \qquad H_p = 10 \qquad Q = 10I \qquad R = 1 \tag{23}$$

$$x_s(k+j) = 0 \qquad \forall j = 1, \ldots, H_p$$
$$u_s(k+j-1) = 0 \qquad \forall j = 1, \ldots, H_c \tag{24}$$

and the inputs are subject to the constraint

$$-2 \leq u(k+j-1) \leq 2 \qquad \forall j = 1, \ldots, H_c \tag{25}$$

The tuning parameters of the optimization algorithms are listed in Table 1. The values of the initial states are chosen randomly from a uniform distribution between −4 and 4. The algorithms are tested for 100 different initial states. At the first sampling instant the algorithms are initialized with an input sequence equal to the zero vector. In subsequent sampling instants, the shifted optimal input vector from the previous sampling instant plus zero for the last input is taken. The MPC algorithm, using the different optimization algorithms, is run for 100 sampling instants, for each initial state. For all the simulations the 2-norm of the end state was smaller than 2.5% of the 2-norm of the initial state, indicating the convergence to zero.

The quality of a simulation is quantified by

$$J_s = \sum_{i=1}^{100} \|x(i)\|_Q^2 + \|u(i-1)\|_R^2 \tag{26}$$

Denote the number of floating point operations for each sampling instant k by $FL(k)$, then the computational load is quantified by

$$J_c = \max\{FL(0), \ldots, FL(99)\} \tag{27}$$

Because algorithm 1 can be regarded as an existing algorithm, and algorithms 2 and 3 as new, the above performance measures are normalized with respect to the performance of algorithm 1 for the corresponding initial state. The logarithm is taken to obtain the same absolute values but of opposite sign for a specific increase or decrease in performance [for example, log(2) = −log(1/2)], leading to the following measures:

$$J_{s,n} = \log\left(\frac{J_s}{J_s \text{ of algorithm 1}}\right)$$
$$J_{c,n} = \log\left(\frac{J_c}{J_c \text{ of algorithm 1}}\right) \tag{28}$$

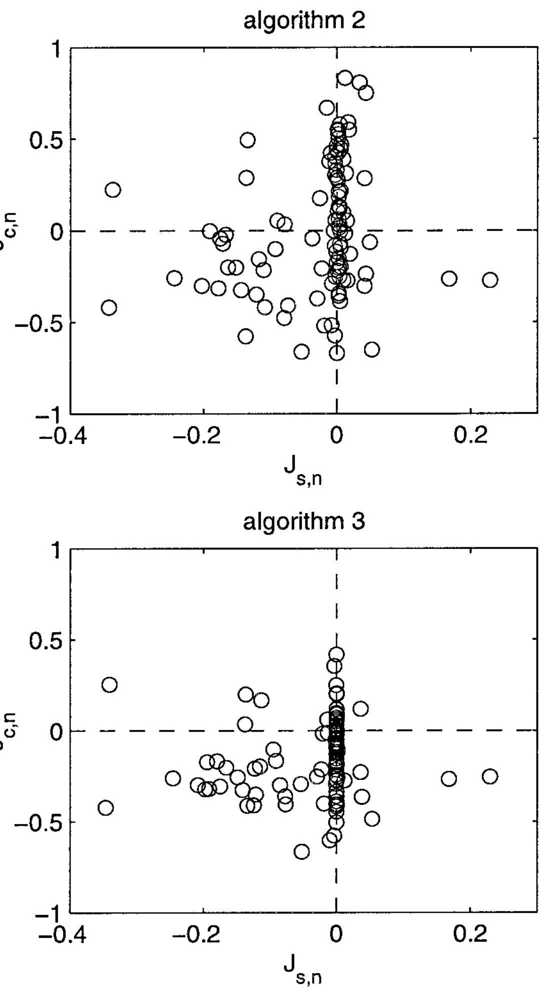

Figure 1. Performance measures (Eq. 28) for algorithms 2 and 3. Every circle corresponds to the result of one simulation.

Negative values of $J_{s,n}$ and $J_{c,n}$ thus correspond to a better performance with respect to the simulation results and computational load, respectively, compared to the results obtained through the use of algorithm 1. For algorithms 2 and 3 the above performance measures are plotted in Figure 1 for all initial states. Every initial state represents a circle in the plots. Comparing the performance of algorithm 2 with algorithm 1 (upper plot of Figure 1) it is obvious that for many trials a better $J_{s,n}$ is achieved (values smaller than zero) in the case of algorithm 2, and for only a few trials a significantly worse $J_{s,n}$ is achieved. There are also many points just right of the line $J_{s,n} = 0$, indicating convergence to the same optimum. This can be explained as follows. Close to the optimum the approximation of the performance index by algorithm 1 is likely to be good, and as a result, the performance of algorithm 1 is good. Given that algorithm 2 relies on fixed search directions, it may terminate close to the optimum while "complete" convergence has not yet been achieved (that is, at a slightly worse value for $J_{s,n}$) when $J(k)$ is shaped like a valley that is not aligned with one of these directions. This explains the large number of points just right of the line $J_{s,n} = 0$ and just above the line $J_{c,n} = 0$, that is, at a larger computational load (attributed to a slow convergence). On the other hand the computational load of algorithm 2 may be smaller than that of algorithm 1 because of a faster convergence in the first iterations. Overall, neither algorithm 2 or 1 has a clear advantage concerning the computational load.

Comparing the upper and lower plot of figure 1 demonstrates that algorithm 3 gives the best overall performance. Many points give a lower value both for $J_{s,n}$ and for $J_{c,n}$. Following the same discussion concerning the differences between algorithms 2 and 1, and comparing the plots of Figure 1, it follows that the benefits of algorithm 3 arise from a faster convergence in the last stage of the optimization, close to the optimum, where it switches to algorithm 1, therefore reducing the computational load in this stage. Thus the hybrid algorithm enables the points just right of the line $J_{s,n} = 0$ in the upper plot to migrate onto the line $J_{s,n} = 0$ in the lower plot, and moreover, most points migrate toward lower values of $J_{c,n}$.

Bilinear MPC for Control of a Polymerization Process

In this section the potential of a bilinear-MPC approach is demonstrated. The "real" process considered in this article is a continuous-time white-box simulation model of the free-radical polymerization of methylmethacrylate (Schmidt and Ray, 1981), and is presented in detail in the next subsection. Note that this is a "general" nonlinear model (neither linear nor bilinear). From data obtained from an identification experiment both a linear and a bilinear model are computed. These models are used within an MPC framework based on the quasi-infinite horizon paradigm (Chen and Allgöwer, 1998). The details of the MPC algorithms and the control results are presented both for the linear model and the bilinear model.

The polymerization process

The process under consideration is a nonlinear white-box simulation model of the free-radical polymerization of methylmethacrylate in a constant volume continuous stirred tank reactor. The model equations of this simulation model are given by Schmidt and Ray (1981)

$$\frac{\partial I}{\partial t} = \frac{q_f}{V} I_f - \frac{q}{V} I - k_d \cdot I \tag{29}$$

$$\frac{\partial M}{\partial t} = \frac{q_f}{V} M_f - \frac{q}{V} M - k_p \cdot M \cdot P \tag{30}$$

$$\frac{\partial S}{\partial t} = \frac{q_f}{V} S_f - \frac{q}{V} S \tag{31}$$

$$\frac{\partial T}{\partial t} = \frac{q_f}{V} T_f - \frac{q}{V} T + \frac{-\Delta H}{\rho \cdot C_p} k_p \cdot M \cdot P - \frac{h \cdot A_c}{V \cdot \rho \cdot C_p} (T - T_c) \tag{32}$$

where q_f and q are the inlet and outlet flow rate, respectively, V is the reactor volume; I_f, M_f, and S_f denote the initiator, monomer, and solvent concentrations in the feed, respectively; I, M, and S denote the initiator, monomer, and solvent concentrations in the reactor, respectively; T_f, T, and T_c denote the temperature of the feed, the reactor, and the cooling jacket, respectively; ΔH is the reaction enthalpy; ρ and C_p are the density and heat capacity of the reactor contents, respectively; h is the heat transfer coefficient to the reactor; A_c is the heat transfer area of the reactor; and P is the total concentration of growing free radicals in the reactor, given by

$$P = \left(\frac{2f \cdot k_d \cdot I}{k_t} \right)^{1/2} \tag{33}$$

where f is the initiator efficiency. Because of the increase in density, resulting from the conversion of monomer to polymer, the outlet flow rate differs from the inlet flow rate and is given by

$$q = q_f(1 + \epsilon_p \chi_p) \tag{34}$$

where

$$\epsilon_p = M \cdot W_m \left(\frac{1}{\rho_p} - \frac{1}{\rho_m} \right) \tag{35}$$

$$\chi_p = \frac{\phi_p \rho_p}{\phi_p \rho_p + W_m \cdot M} \tag{36}$$

where W_m is the molecular weight of monomer, ρ_p and ρ_m are the densities of polymer and monomer, respectively; and ϕ_p is the volume fraction of polymer. For the reaction rates the following holds

$$k_d = 1.69 \times 10^{14} \exp\left(\frac{-30{,}000}{R \cdot T} \right) \tag{37}$$

$$k_p = k_{p0} \cdot g_p \tag{38}$$

$$k_t = k_{t0} \cdot g_t \tag{39}$$

where R is the gas constant

$$k_{p0} = 4.92 \times 10^5 \exp\left(\frac{-4353}{R \cdot T}\right) \tag{40}$$

$$k_{t0} = 9.8 \times 10^7 \exp\left(\frac{-701}{R \cdot T}\right) \tag{41}$$

and the functions g_p and g_t, which are attributed to the "gel effect," are given by

$$g_p = \begin{cases} 1, & V_f > 0.05 \\ 7.1 \times 10^{-5}\exp(171.53 \cdot V_f) & V_f \leq 0.05 \end{cases} \tag{42}$$

$$g_t = \begin{cases} 0.10575 \cdot \exp[17.15 \cdot V_f - 0.01715(T - 273.2)] \\ \quad V_f > (0.1856 - 2.965 \times 10^{-4}(T - 273.2)) \\ 2.3 \times 10^{-6}\exp(75V_f) \\ \quad V_f \leq (0.1856 - 2.965 \times 10^{-4}(T - 273.2)) \end{cases} \tag{43}$$

where the total free volume V_f is given by (Kurtz et al., 2000)

$$V_f = \max[V_{fp}\phi_p + V_{fm}\phi_m + V_{fs}\phi_s, 0] \tag{44}$$

where ϕ_p, ϕ_m, and ϕ_s are the volume fractions of polymer, monomer, and solvent in the mixture, calculated from

$$\phi_m = \frac{W_m \cdot M}{\rho_m} \tag{45}$$

$$\phi_s = \frac{W_s \cdot S}{\rho_s} \tag{46}$$

$$\phi_p = \frac{\rho - \phi_m\rho_m - \phi_s\rho_s}{\rho_p} \tag{47}$$

under the assumption that the volume fraction of initiator is approximately zero. W_s and ρ_s are the molecular weight and the density of the solvent, respectively. V_{fp}, V_{fm}, and V_{fs} are given by

$$V_{fm} = 0.025 + 0.001(T - 167) \tag{48}$$

$$V_{fp} = 0.025 + 0.00048(T - 387) \tag{49}$$

$$V_{fs} = 0.025 + 0.001(T - 181) \tag{50}$$

In this article the manipulable input is chosen to be the monomer concentration in the feed (which affects the solvent concentration in the feed) and the output is chosen to be the monomer concentration in the reactor. The values of the remaining model parameters are listed in Table 2.

Table 2. Parameter Values of the White-Box Polymerization Model

Parameter	Value	Unit	Source
R	1.987	cal/(K · mol)	
f	0.5		1, 2, 3
C_p	0.4	cal/(g · K)	1, 2
ρ	1038	g/L	1, 2
ρ_s	901	g/L	2
ρ_m	939	g/L	2
ρ_p	1200	g/L	2
h	135.6	cal/(m^2 · s · K)	1
A_c	2.8	m^2	1
$-\Delta H_p$	13.8×10^3	cal/mol	1
W_s	88.10	g/mol	2
W_m	100.11	g/mol	2, 3
q_f	0.2813	L/s	2
V	900	L	2
I_f	0.02	mol/L	
T_f	298	K	3
T_c	350	K	

Sources: (1) Adebuken and Schork, 1989; (2) Kurtz et al., 2000; (3) Schmidt and Ray, 1981.

Predictive control

Model of the Predictive Controller. The model of the previous subsection was simulated with an input trajectory to yield input–output data of the polymerization process. These input–output data were then used to compute black-box models that approximate the behavior of the white-box simulation model. These black-box models are the internal models of the predictive controller, and the white-box simulation model is not known by the predictive controller. Thus this approach mimics the "real-life" situation where a model of the true process is not exactly known.

Two black-box models were identified:

(1) A linear state–space model. This corresponds to a model of the form in Eqs. 1 and 2 with $F_p = 0$. The PO-MOESP algorithm described in Verhaegen (1994) was used to obtain this model from the input–output data.

(2) A bilinear state–space model, of the form in Eqs. 1 and 2. The subspace-identification method presented by Verdult and Verhaegen (2000) was used to obtain this model from the input–output data.

See Bloemen et al. (2002) for more information regarding the identification procedure.

MPC Algorithm. The performance index of Eq. 5 is extended with a penalty on the last predicted state

$$J(k) = \sum_{j=1}^{H_p} \{\|y(k+j) - y_s(k+j)\|^2_{Q_y} + \|u(k+j-1) - u_s(k+j-1)\|^2_{R_u}\} + \|x(k+H_p) - x_s(k+H_p)\|^2_{P_x} \tag{51}$$

where Q_y is a positive-definite weighting matrix that penalizes deviations of the output y from its set point y_s, and P_x is a positive-definite weighting matrix such that $\|x(k + H_p) - x_s(k + H_p\|^2_{P_x}$ is an upper bound for the summation of the first two terms in Eq. 51 for $j = H_p + 1, \ldots, \infty$, under the restriction that $x(k + H_s)$ belongs to a target set (defined by an end-point inequality constraint), which is invariant under the model in closed loop with a feedback law. This corresponds to the so-called quasi-infinite-horizon paradigm (Chen and Allgöwer, 1998). For linear and bilinear models of which A has eigenvalues strictly within the unit disc, and in the presence of only input constraints, the above-mentioned feedback law is trivially given by $u = 0$ and the target set corresponds to the entire state space. Then P_x is given by the solution to the following Lyapunov equation

$$P_x - A^T P_x A = A^T C^T Q_y C A \tag{52}$$

Given that in this paper the black-box models, that is, the linear and bilinear state–space models, are obtained by an open-loop identification experiment, the process (white-box model) can be operated only in a stable operating region. Therefore it is plausible that the identified models have A matrices of which the eigenvalues are strictly within the unit disc.

The minimization of Eq. 51 is subject to the constraint that the input u lies within the range that is used for the identification experiment, to operate the process in the region in which it has been identified

$$u_{min} \leq u(k + j - 1) \leq u_{max} \qquad \forall j = 1, \ldots, H_p \tag{53}$$

where $u = u_s$ must be feasible for a feasible set point. For linear models the problem of minimizing Eq. 51 subject to Eq. 53 is convex and can be solved by quadratic programming. For bilinear models this minimization problem is nonconvex and can be solved using algorithm 3 presented in this article.

To obtain a good controller performance the identified black-box model, which is used within the controller, should be able to approximate accurately the behavior of the process (here, the white-box simulation model). However, in practice some degree of model mismatch is unavoidable (this could also arise because of disturbances, for example). Here the controller copes with model mismatch by adding the modeling error

$$e(k) = y_p(k) - y(k) \tag{54}$$

to the predictions of the model

$$y(k + j) = Cx(k + j) + e(k) \qquad j \geq 1 \tag{55}$$

where in Eq. 54 y_p is the output of the controlled process (Henson, 1998). Given a desired set point y_s, a certain $e(k)$ will affect the set points u_s and x_s. The feedback path provided by Eqs. 54 and 55 enables one to eliminate steady-state offset.

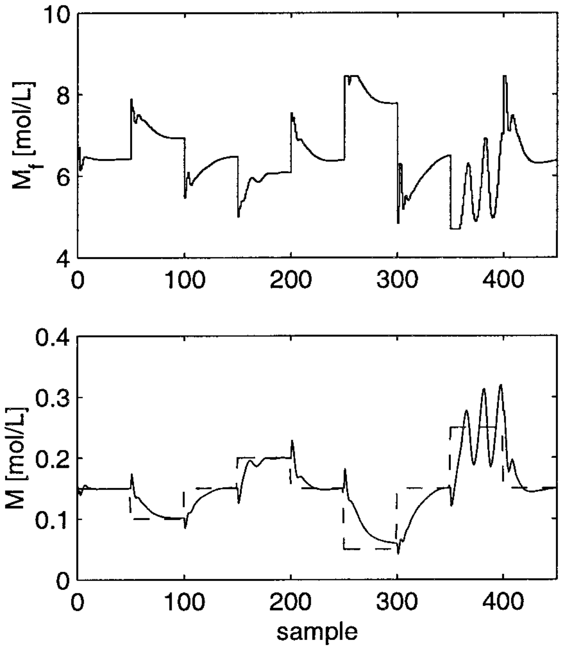

Figure 2. Control results for LMPC; upper plot: the input M_f; lower plot: the solid line represents the output M; the dashed line represents the set point for M.

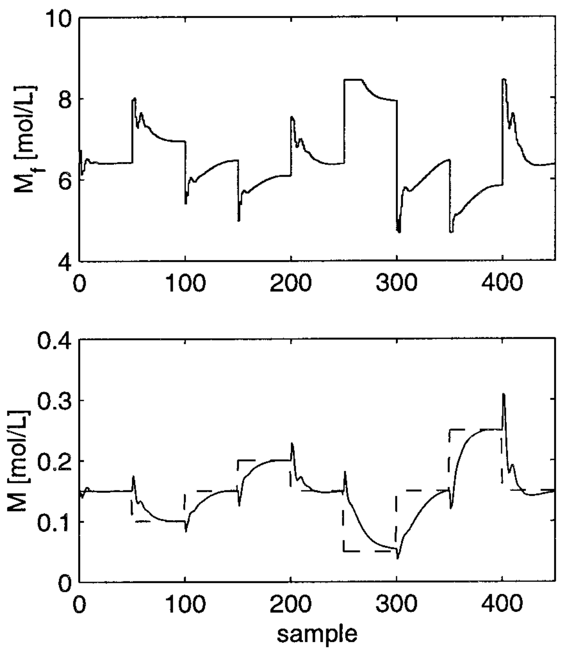

Figure 3. Control results for BMPC; upper plot: the input M_f; lower plot: the solid line represents the output M; the dashed line represents the set point for M.

Control results

The tuning parameters of both the linear model–based predictive controller (LMPC) and the bilinear model–based predictive controller (BMPC) were set to $H_p = 3$, $Q_y = 10$, and $R_u = 1$. The input M_f was restricted such that the volume fraction of monomer in the feed remained between 0.5 and 0.9. The set point for the output M was changed every 50 samples, as indicated in Figures 2 and 3, where the control results for LMPC and BMPC, respectively, are plotted.

During the first 150 samples of Figures 2 and 3, where the MPC algorithms operate approximately at the center of the operating region, the performances of LMPC and BMPC were similar. In this region the behavior of the process is modeled well both by the linear model and by the bilinear model, resulting in a similar performance for both LMPC and BMPC. However, for larger values of M the approximation of the behavior of the process by the linear model deteriorates, which results in a response of LMPC that tends to be oscillatory between samples 150 and 200, and that is very oscillatory between samples 350 and 400 (see Figure 2). The approximating properties of the bilinear model are better than those of the linear model. Consequently, the performance of BMPC is better than that of LMPC for operating points that lie further from the center of the operating region used for identification. This is best illustrated between samples 350 and 400 where BMPC performs well, whereas LMPC does not (compare Figures 2 and 3). The oscillatory behavior of the plant in closed loop with LMPC can be prevented by tuning down the controller. For this example this requires Q_y to be reduced to 1 for LMPC (data not shown). However, this retuning makes the controller very conservative, which leads to a very slow response over the entire operating region, in which case the performance of LMPC using $Q_y = 1$ is (still) inferior to that of BMPC using $Q_y = 10$.

Conclusions

Herein two new optimization algorithms (algorithms 2 and 3) were presented that focus on MPC problems for discrete-time bilinear models. These algorithms were compared to an existing optimization algorithm, algorithm 1, which is suitable for more general nonlinear MPC problems as well. Algorithm 2 exploits the structure of the bilinear model by solving the bilinear MPC optimization problem by means of a sequence of exact quadratic programs in specific search directions, whereas algorithm 1 uses a quadratic approximation of the original problem, the solution of which is used for a subsequent line search. The advantages of both optimization algorithms are combined in the hybrid algorithm 3, which gives the best overall performance, as is demonstrated by means of a simulation study. More details regarding the optimization algorithms can be found in Bloemen (2002).

The potential of the bilinear MPC approach is demonstrated in the previous section. Both a linear and a bilinear model were identified from an input–output data set generated by simulation of a white-box simulation model, which is considered as the true process. The control results of BMPC outperformed the results of LMPC because the bilinear model approximates the behavior of the nonlinear true process better than the linear model.

Acknowledgments

This work was supported by the Dutch Technology Foundation (STW) under project number DEL55.3891.

Literature Cited

Adebekun, D. K., and F. J. Schork, "Continuous Solution Polymerization Reactor Control. 1. Nonlinear Reference Control of Methyl Methacrylate Polymerization." *Ind. Eng. Chem. Res.*, **28**, 1308 (1989).

Allgöwer, F., T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright, "Nonlinear Predictive Control and Moving Horizon Estimation: An Introductory Overview," *Advances in Control, Highlights of ECC'99*, Springer-Verlag, London, pp. 391–449 (1999).

Bloemen, H. H. J., "Predictive Control Based on Black-Box State–Space Models," PhD thesis, Delft University of Technology, Delft, The Netherlands (2002).

Bloemen, H. H. J., V. Verdult, T. J. J. v. d. Boom, and M. Verhaegen, "Bilinear versus Linear MPC: Application to a Polymerization Process," *Proc. 15th IFAC World Congress*, Barcelona, Spain (July 2002).

Boggs, P. T., and J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica*, 1–51 (1995).

Brooms, A. C., and B. Kouvaritakis, "Successive Constrained Optimization and Interpolation in Nonlinear Model Based Predictive Control," *Int. J. Control*, **73**(4), 312 (2000).

Chen, H., and F. Allgöwer, "A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability," *Automatica*, **34**(10), 1217 (1998).

Chen, H., and J. Maciejowski, "Subspace Identification of Deterministic Bilinear Systems, *Proc. American Control Conference*, Chicago, IL, pp. 1797–1801 (June 2000).

Di Marco, R., D. Semino, and A. Brambilla, "From Linear to Nonlinear Model Predictive Control: Comparison of Different Algorithms," *Ind. Eng. Chem. Res.*, **36**(5), 1708 (1997).

Favoreel, W., B. De Moor, and P. Van Overschee, "Subspace Identification of Bilinear Systems Subject to White Inputs," *IEEE Trans. Automatic Control*, **44**(6), 1157 (1999).

Henson, M. A., "Nonlinear Model Predictive Control: Current Status and Future Directions," *Comput. Chem. Eng.*, **23**, 187 (1998).

Kouvaritakis, B., M. Cannon, and J. A. Rossiter, "Non-linear Model Based Predictive Control," *Int. J. Control*, **72**(10), 919 (1999).

Kurtz, M. J., G.-Y. Zhu, and M. A. Henson, "Constrained Output Feedback Control of a Multivariable Polymerization Reactor," *IEEE Trans. Control Systems Technol.*, **8**(1), 87 (2000).

Rao, S. S., *Optimization: Theory and Applications*, Wiley Eastern Ltd., New Delhi, India (1978).

Schmidt, A. D., and W. H. Ray, "The Dynamic Behavior of Continuous Polymerization Reactors—I. Isothermal Solution Polymerization in a CSTR," *Chem. Eng. Sci.*, **36**, 1401 (1981).

Verdult, V., and M. Verhaegen, "Identification of Multivariable Linear Parameter Varying Systems Based on Subspace Techniques, *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, pp. 1567–1572 (Dec. 2000).

Verdult, V., M. Verhaegen, and C. T. Chou, "Identification of MIMO Bilinear State Space Models Using Separable Least Squares," *Proc. American Control Conference*, San Diego, CA, pp. 838–842 (June 1999).

Verhaegen, M., "Identification of the Deterministic Part of MIMO State Space Models Given in Innovations Form from Input–Output Data," *Automatica*, **30**(1), 61 (1994).

Manuscript received Dec. 24, 2002, and revision received Aug. 23, 2003.