ELSEVIER

Journal of Process Control 10 (2000) 1–8

JOURNAL OF PROCESS CONTROL

www.elsevier.com/locate/jprocont

Model predictive control of large scale processes

Jeremy G. Van Antwerp, Richard D. Braatz*

Large Scale Systems Research Laboratory, Department of Chemical Engineering, University of Illinois at Urbana-Champaign, 600 South Mathews Avenue, Box C-3, Urbana, IL 61801-3792, USA

Abstract

Model predictive control is a receding horizon control policy in which a linear or quadratic program with linear constraints is solved on-line at each sampling instance. An algorithm is developed that allows quick computation of suboptimal control moves. The linear constraint set is approximated by an ellipsoid and a change of variables is performed so that a solution may be computed efficiently via bisection. The ellipsoid is rescaled on-line to reduce conservatism. This allows the implementation of model predictive control algorithms to large scale processes using simple control hardware. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Model-based control; Large-scale systems; Quadratic programming

1. Introduction

In the late 1970s, *Model Predictive Control* began to be applied in the chemical process industries. In Model Predictive Control [1–3] (and its many variants, MPHC, GPC, DMC, IDCOM, MMC), the control objective is optimized on-line subject to the constraints. A linear program (LP) or quadratic program (QP) is solved at each sampling instance. These optimization problems can be large (1000 variables and 2000 or more constraints for a process with 200 manipulated variables and a control horizon of five).

Even today, there are applications for which the computational expense associated with solving the LP or QP on-line can be inconvenient. Control hardware in industrial environments is often not state-of-the-art, and in some cases is over a decade old. There are many industrial processes for which it is desirable to have the advantages of advanced control, but for which the time and expense of installing modern control hardware is not justified. Even if modern control hardware is available, reducing the computations associated with the control algorithm can free up resources to provide more flexibility in the control algorithm (e.g. to take nonlinearities or model uncertainties into account), and to implement other algorithms (e.g. on-line identification, fault detection and isolation, data reconciliation).

As an example, a control algorithm implemented on an industrial-scale adhesive coater was much less expensive and provided performance similar to a model predictive control algorithm that solved the QP on-line [4]. The company engineers preferred the more reliable and easier-to-implement control algorithm as it could be quickly migrated to other coaters within the company.

In the past few years, several researchers have developed algorithms that quickly compute an optimal or suboptimal solution to the linear or quadratic programs associated with model predictive control [5–8]. Much of this work was focused on application to sheet and film processes (which can have 100s–1000s of inputs and outputs), although the algorithms that were developed apply to more general processes. In this manuscript, it is shown how to compute a fast suboptimal solution to generic QP-based model predictive control problems.

2. Algorithm

The proposed algorithm, called the iterated ellipsoid (IE) algorithm, consists of two parts. The first set of computations is performed off-line. The linear constraint set is approximated by an ellipsoid, and a change of variables is performed to give the problem a simpler structure. The second set of computations is performed on-line. It efficiently computes a suboptimal solution via bisection and rescales the ellipsoid to reduce conservatism.

* Corresponding author. Tel.: +1-217-333-5073; fax: +1-217-333-5052.

E-mail address: braatz@uiuc.edu (R.D. Braatz)

PII: S0959-1524(99)00050-5

2.1. Off-line computation

Model Predictive Control (MPC) typically consist of multiple computational steps, for example, model prediction, state estimation, and control computation [1–3]. This paper will focus on the computation of the control moves, as this is the most computationally expensive step in an MPC algorithm. The control move computation for the most popular formulations of MPC can be written as a linearly constrained quadratic program (QP):

$$\min_{\mathbf{Fz}\leqslant\mathbf{q}} \ \|\mathbf{Hz}-\mathbf{c}\|_2^2 \tag{1}$$

where it is assumed that $\mathbf{H}$ has full column rank. The linear constraints $\mathbf{Fz}\leqslant\mathbf{q}$ form a bounded convex polytope. It is assumed that any pure equality constraints have been removed by redefining the manipulated variables, so that the polytope has an interior. To efficiently compute a suboptimal solution to (1), the linear constraints will be approximated by an ellipsoid. This approximation is performed *off-line* as follows. We compute the ellipsoid of maximal volume which is completely contained within the polytope defined by the linear constraints. An ellipsoid E is defined by

$$E=\left\{\mathbf{z}\middle|\mathbf{z}=\mathbf{By}+\mathbf{d},\,\mathbf{y}^T\mathbf{y}\leqslant 1\right\} \tag{2}$$

where $\mathbf{d}$ is a vector that defines the center of the ellipsoid and $\mathbf{B}$ is a symmetric positive definite matrix that defines the semi-axes of the ellipsoid. Finding the largest (maximal volume) ellipsoid which is completely contained within the polytope $\mathbf{Fz}\leqslant\mathbf{q}$ is a convex optimization problem [9]:

$$\max_{\mathbf{B},\mathbf{d}} \ \log\det(\mathbf{B}) \tag{3}$$

subject to the constraints

$$\mathbf{B}=\mathbf{B}^{\mathrm{T}}>0 \tag{4}$$

$$\|\mathbf{BF_i}\|+\mathbf{F_i}^T\mathbf{d}\leqslant q_i,\ \forall i\in[1,L] \tag{5}$$

where $\mathbf{F_i}^T$ is a row vector corresponding to the ith row of $\mathbf{F}$, q_i is the ith element of $\mathbf{q}$, and L is the number of elements in $\mathbf{q}$. This optimization problem can be solved in polynomial time by interior point methods [10].

We will now write the ellipsoid in (2) in an equivalent form. Solving for $\mathbf{y}$ in (2) and inserting this into the constraint $\mathbf{y}^T\mathbf{y}\leqslant 1$ gives

$$[\mathbf{B}^{-1}(\mathbf{z}-\mathbf{d})]^T[\mathbf{B}^{-1}(\mathbf{z}-\mathbf{d})]\leqslant 1. \tag{6}$$

Rearranging shows that the ellipsoid (2) is equivalent to

$$E=\left\{\mathbf{z}\middle|(\mathbf{z}-\mathbf{d}^T\mathbf{B}^{-2}(\mathbf{z}-\mathbf{d})\leqslant 1\right\}. \tag{7}$$

The QP (1) with the linear constraints replaced by the ellipsoid constraint is now in the form

$$\min_{(\mathbf{z}-\mathbf{d})^T\mathbf{B}^{-2}(\mathbf{z}-\mathbf{d})\leqslant\alpha} \ \|\mathbf{Hz}-\mathbf{c}\|_2^2 \tag{8}$$

where $\alpha>0$ is a scaling parameter which is optimized online to minimize conservatism. The QP (8) is equivalent to

$$\min_{\bar{\mathbf{x}}^T\mathbf{B}^{-2}\bar{\mathbf{x}}\leqslant\alpha} \ \left\|\mathbf{H}\bar{\mathbf{x}}-\mathbf{b}\right\|_2^2 \tag{9}$$

where $\bar{\mathbf{x}}=\mathbf{z}-\mathbf{d}$ and $\mathbf{b}=\mathbf{c}-\mathbf{Hd}$. Now define $\mathbf{x}=\mathbf{B}^{-1}\bar{\mathbf{x}}$. This gives

$$\min_{\mathbf{x}^T\mathbf{x}-\alpha\leqslant 0} \ \mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-2\mathbf{b}^T\mathbf{Ax}+\mathbf{b}^T\mathbf{b}=\min_{g(\mathbf{x})\leqslant 0} f(\mathbf{x}) \tag{10}$$

where $\mathbf{A}=\mathbf{HB}$ has full column rank.

To summarize, the general linear constraints in (1) were approximated with an ellipsoid and the problem was converted to an equivalent form (10) with simpler structure via a change of variables.

2.2. On-line computation

The functions $f(\mathbf{x})$ and $g(\mathbf{x})$ in (10) are convex and continuously differentiable with respect to $\mathbf{x}$ for all $\mathbf{x}$, and $\mathbf{x}=\underline{0}$ is a feasible point. Thus a necessary and sufficient condition for $\hat{\mathbf{x}}$ to be the global solution of (10) is that there exists a Lagrange multiplier $\lambda\geqslant 0$ which satisfies

1. $\lambda g(\hat{\mathbf{x}})=0$; and
2. $\nabla_x(\hat{\mathbf{x}}^T\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}}-2\mathbf{b}^T\mathbf{A}\hat{\mathbf{x}}+\mathbf{b}^T\mathbf{b})+\lambda\nabla_x(\hat{\mathbf{x}}^T\hat{\mathbf{x}}-\alpha)=0$.

First consider the case where $\lambda=0$. Then the solution to the constrained optimization problem is equal to the solution of the unconstrained optimization problem and is calculated from condition 2 to give

$$\hat{\mathbf{x}}_u=(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, \tag{11}$$

where the matrix $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is calculated once off-line and stored for use here. The first step of the IE algorithm is to compute this unconstrained solution. Next, it is checked against a linear transformation of the original constraints in (1):

$$\mathbf{FB}\hat{\mathbf{x}}_u\leqslant\mathbf{q}-\mathbf{Fd} \tag{12}$$

where $\mathbf{FB}$ and $\mathbf{q} - \mathbf{Fd}$ are calculated off-line and stored for use here. This transformation is done to avoid having to perform a change of variables repetitively. If the unconstrained solution satisfies these constraints, then it is implemented.

If the unconstrained solution violates the constraints, it follows that one or more constraints are active, and a global solution $\hat{\mathbf{x}}$ to (10) must satisfy the following three conditions:

1. $\lambda > 0$;
2. $\hat{\mathbf{x}}^T\hat{\mathbf{x}} = \alpha$; and,
3. $2\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} - 2\mathbf{A}^T\mathbf{b} + 2\lambda\hat{\mathbf{x}} = 0$

Solve for $\hat{\mathbf{x}}$ in the third equation to give

$$\hat{\mathbf{x}}(\lambda) = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b}, \tag{13}$$

substitute into the second, and define the function $h(\lambda)$ which satisfies the equation

$$h(\lambda) \equiv \hat{\mathbf{x}}(\lambda)^T\hat{\mathbf{x}}(\lambda) = \alpha. \tag{14}$$

The function $h(\lambda)$ is strictly monotonically decreasing in λ, and there exists a unique finite $\lambda > 0$ which satisfies $h(\lambda) = \alpha$. Further, this λ may be determined by bisection.

The vector $\hat{\mathbf{x}}(\lambda)$ [and the function $h(\lambda)$] may be computed very efficiently by taking advantage of the fact that the matrix $\mathbf{A}^T\mathbf{A}$ has the following SVD decomposition

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{R}\mathbf{V}^T, \tag{15}$$

where $\mathbf{V}$ is a unitary matrix and $\mathbf{R}$is a diagonal matrix containing the singular values of $\mathbf{A}^T\mathbf{A}$. Thus

$$\hat{\mathbf{x}}(\lambda) = \mathbf{V}(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{V}^T\mathbf{A}^T\mathbf{b}, \tag{16}$$

where the matrix $\mathbf{V}^T\mathbf{A}^T$ was computed once off-line for use here. The vector $(\mathbf{V}^T\mathbf{A}^T)\mathbf{b}$ is computed once for each control move to be implemented and each element of this vector is divided by the corresponding diagonal element of $\mathbf{R} + \lambda\mathbf{I}$. The vector $\hat{\mathbf{x}}(\lambda)$ is then computed by a single matrix–vector multiplication.

When $\hat{\mathbf{x}}$ in (11) does not satisfy the constraint set, then an optimal solution to (1) must have one or more of its constraints active. Thus, if the solution obtained from (13) does not have an active constraint then the scaling parameter α is adjusted and the process of solving (14) is repeated until a solution is obtained which has an active constraint.

Using a result from [9], it can be shown that $\alpha \in [1, \bar{\alpha}]$, where

$$\bar{\alpha} = \min\left\{n^2, \hat{\mathbf{x}}_u^T\hat{\mathbf{x}}_u\right\} \tag{17}$$

where $\hat{\mathbf{x}}_u$ is the unconstrained solution given in (11). Using a result given in [11], this lower bound on $\bar{\alpha}$ can be tightened to

$$\bar{\alpha} = \min\left\{n, \hat{\mathbf{x}}_u^T\hat{\mathbf{x}}_u\right\} \tag{18}$$

when the polytope is symmetric. The case $\alpha = 1$ defines an ellipsoid completely enclosed by the polytope $\mathbf{Fz} \leqslant \mathbf{q}$. The case $\alpha = n^2$ ($\alpha = n$ if the ellipsoid is symmetric) defines an ellipsoid completely enclosing the polytope $\mathbf{Fz} \leqslant \mathbf{q}$. These bounds imply that the α which makes the constraints active also can be computed by bisection.

In summary, the on-line computation consists of an inner loop and an outer loop for each constrained control move to be computed. The inner loop finds the λ which makes $h(\lambda) = \alpha$ for a given α via bisection and requires nothing more expensive than a matrix-vector multiplication at each step. The outer loop finds, via bisection, the α which makes the constraints active. The constraints are checked in the outer loop, which requires a single matrix–vector multiplication.

The IE algorithm is not a standard ellipsoidal algorithm [12], since B in (2) is computed only once. Standard ellipsoidal algorithms recompute a new ellipsoid that encloses the optimal solution at each step — this is at a higher computational cost relative to the IE algorithm which only *rescales* the ellipsoid at each step. The IE algorithm may be interpreted as a penalty function approach with a time varying weight on the control action [13]. The penalty function approach computes the solution to an unconstrained optimization problem, but ensures feasibility to the original optimization problem by penalizing the control action in the objective function until the constraints are satisfied. In our case, this is accomplished by a variable penalty term and a constant weighting matrix defined by the orientation of a well-chosen ellipsoid.

Standard ellipsoidal algorithms [12] are much slower than the fastest interior point algorithms [10], which have cubic growth in the number of flops as a function of problem size. For the standard MPC formulation, this is $O((nm)^3)$, where m is the control horizon and n is the number of manipulated variables [7]. In contrast, the IE algorithm's most expensive step is a matrix–vector multiplication, which has quadratic growth $O((nm)^2)$ as a function of problem size. The structured interior point method of [7] is $O(m(n + 2n_s)^3)$, where n_s is the number of states of the system. Hence the IE algorithm will be faster than the structured interior point algorithm for problems with large numbers of manipulated variables, but will perform more slowly for problems with very long control horizons.

The performance of the IE algorithm can be monitored on-line. A lower bound to the optimal QP objective (1) is

obtained by solving (13) and (14) for $\bar{\alpha}$ defined in (17) and (18). For the same past history, the solution to the IE algorithm gives an upper bound for the QP objective (1). The difference between the lower and upper bounds provides an upper bound on the conservatism of the IE algorithm.

3. Simulation examples

This section will illustrate the properties of the iterated ellipsoid (IE) algorithm via simulation examples. The performance of the IE algorithm is compared to quadratic programming (QP) and to two recently proposed algorithms [8] — the Modified QP (MQP) and the Linear Approximation (LA) algorithms. These algorithms were selected for simulation studies because the MQP algorithm is the most accurate approximate MPC algorithm to appear in the literature to date, while the LA algorithm is the fastest approximate MPC algorithm to appear in the literature to date. Comparisons are also made to a "fast" exact LP [6], an approximate LP [6], and a fast QP algorithm [7]. Simple simulation models (simple dynamics, no measurement noise, no model uncertainty) were selected both for brevity and to facilitate a direct comparison of the algorithms. The QPs generated for the QP and MQP methods in these examples were solved using IMSL's QP solver, which is an active set method implemented in FORTRAN [14–16]. This QP solver was selected because it is widely distributed, and because a detailed simulation study indicated that the code was of comparative efficiency to other well-known QP solvers such as QPSOL [16]. The IE and LA algorithms were also implemented in FORTRAN. To provide a fair comparison between the IE, QP, MQP, and IE algorithms, all algorithms were all compiled with the same level of optimization. Also, to provide a fair comparison, none of the implemented algorithms were given a "warm start" [17] (all of the IE, QP, and MQP algorithms' computational times could be reduced for future sampling instances by using warm starts).

3.1. Simplified paper machine

The purpose of this example is to compare the speed of the IE algorithm to a QP active set method [14–16] and the MQP algorithm (the LA algorithm is the fastest algorithm but can perform more poorly as will be demonstrated in the next section).

Consider the Toeplitz model for the simplified paper machine model studied in [8]:

$$\mathbf{y}(k+1) = 0.6\mathbf{y}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{d}(k) \tag{19}$$

where

$$\mathbf{B} = 2.5\begin{bmatrix} 1 & 0.7 & 0.4 & 0.1 & 0 & 0 & \dots & 0 \\ 0.7 & 1 & 0.7 & 0.4 & 0.1 & 0 & \dots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & \dots & 0 & 0 & 0.1 & 0.4 & 0.7 & 1 \end{bmatrix} \in \mathcal{R}^{n\times n} \tag{20}$$

and $\mathbf{y}(k) \in \mathcal{R}^n$, $\mathbf{u}(k) \in \mathcal{R}^n$. Although this model is far too simple to describe a modern paper machine [18], it is used here to directly compare with previous results. The disturbance vector $\mathbf{d}(k)$ is equal to 2.5, $\forall k \geqslant 1$. The initial profile is $\mathbf{y}(0) = 0$. The inputs, $\mathbf{u}(k)$, are constrained such that $|u_i(k)| \leqslant 1$, $\forall i$, which constrains the second-order bending moment to be less than four [4]. We consider an output horizon of $p = 10$ and a control horizon of $m = 5$. The objective function to be minimized is:

$$\sum_{i=1}^{10} (\mathbf{r}(k+i) - \mathbf{y}(k+i|k))^T (\mathbf{r}(k+i) - \mathbf{y}(k+i|k)) \tag{21}$$

Fig. 1 shows the solution time for this problem as a function of problem size (n) for the IE, QP, and MQP algorithms. The times are shown as the total computational time for control calculations performed at 10 consecutive sampling instances. The above disturbance was scaled to have much larger magnitude so that the constraints would be active at each sampling instance, in which case the IE algorithm will perform at its slowest (since it is much faster when the unconstrained solution is optimal), taking approximately the same computation time at each sampling instance.

The problem is a quadratic program (QP) of size mn. The number of flops for the fastest generic algorithms

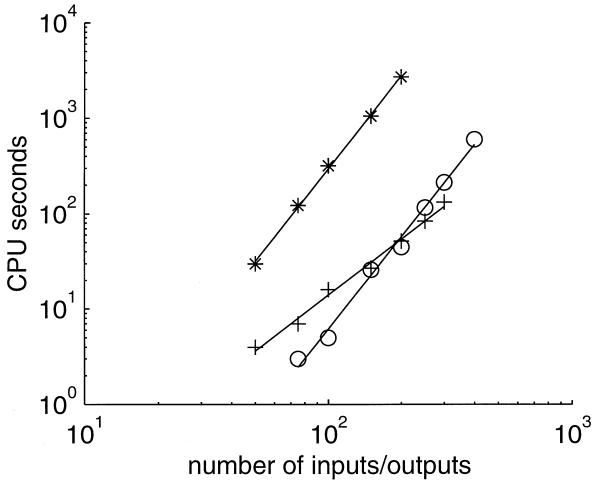

Fig. 1. On-line computation time on an UltraSparc 143 MHz as a function of problem size for 10 sampling times for a disturbance in which the constraints were active at each sampling instance. QP (*), IE (+), and MQP (o).

for solving the QP is $O((mn)^3)$ [10]. The MQP algorithm solves a quadratic program of size n instead of mn by assuming that inputs $\mathbf{u}(k+1), \ldots, \mathbf{u}(k+m-1)$ are unconstrained. Thus the computational cost for the fastest MQP algorithm is $O(n^3)$ flops. The IE algorithm solves a problem of size mn by approximating the polytopic constraint set as an ellipsoid (see the Algorithm section). The IE algorithm's most expensive step is a matrix–vector multiplication, whose cost is $O((mn)^2)$ flops.

The order of the computational requirements for each algorithm can be estimated from the slope of its line in Fig. 1. The slope for both QP algorithms is approximately 3.2 while the slope for the IE algorithm is 2.0. Thus, for large scale problems, the lower order of the IE algorithm makes it faster than the MQP and QP algorithms.

Fig. 2 and 3 compare the closed loop performance of the algorithms. Measurement noise and model uncertainty were not included in the simulation to facilitate this comparison. The plot of the outputs shows that the QP performs better than the IE algorithm at the first time step (a standard deviation of 0.0565 rather than 0.0512) but after that, both algorithms achieve perfect disturbance rejection. The MQP algorithm achieves performance very similar to the QP solution in simulations in [8].

On the other hand, the IE algorithm provides a noticeably smoother series of input vectors (see Fig. 3). This can be an advantage in many practical applications for which a 'sawtooth' input vector is undesirable (e.g. it may result in excessive wear to a slice lip).

Although a direct comparison of the IE algorithm with the fast LP-based approach of [6] is like comparing apples and oranges since different objective functions are used, here we compare the solution times for the two algorithms. Based on the FFT, MM, and TFFTdp benchmarks (see www.netlib.org/performance), the HP 9000 J210 2-processor workstation used in [6] is approximately 1.5 times as fast as the single processor Sun Ultra 1 workstation used to perform these simulations. Since the control horizon is 5, the number of decision variables for our 100 actuator case is 500, in which case the IE algorithm took $16/10 = 1.6$ CPU s per sampling instance. The fastest time to compute the exact LP solution to a simplified paper machine control problem with 439 manipulated variables and a control horizon of one in [6] was 4.95 CPU s, which was for a V-shaped disturbance (second to last column in Table 6 in [6]). After scaling for computer speed and the number of decision variables, we estimate that the IE algorithm would require 20% of the computational time of the fast exact LP algorithm if run on the same computer

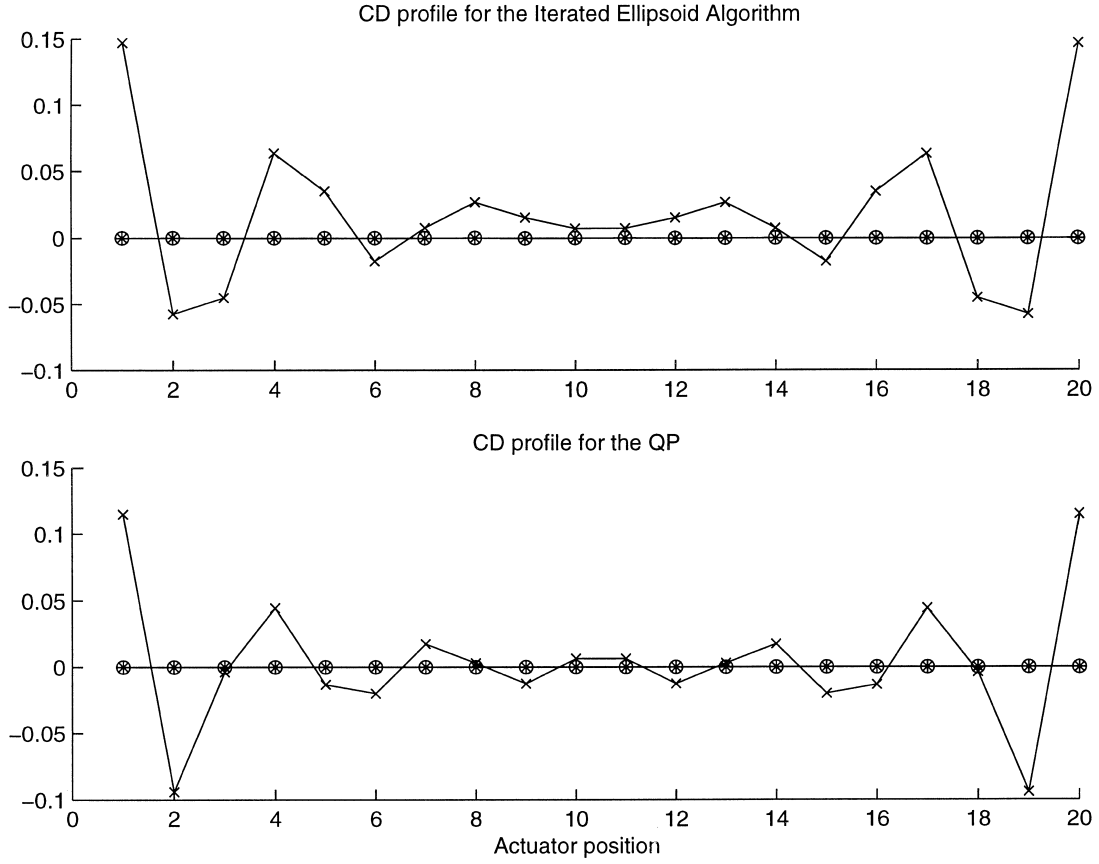

Fig. 2. Output profiles for the QP and IE algorithms: $t = 1(\times)$, $t = 2(+)$, $t = 3(\mathrm{o})$, $t = 4(*)$.

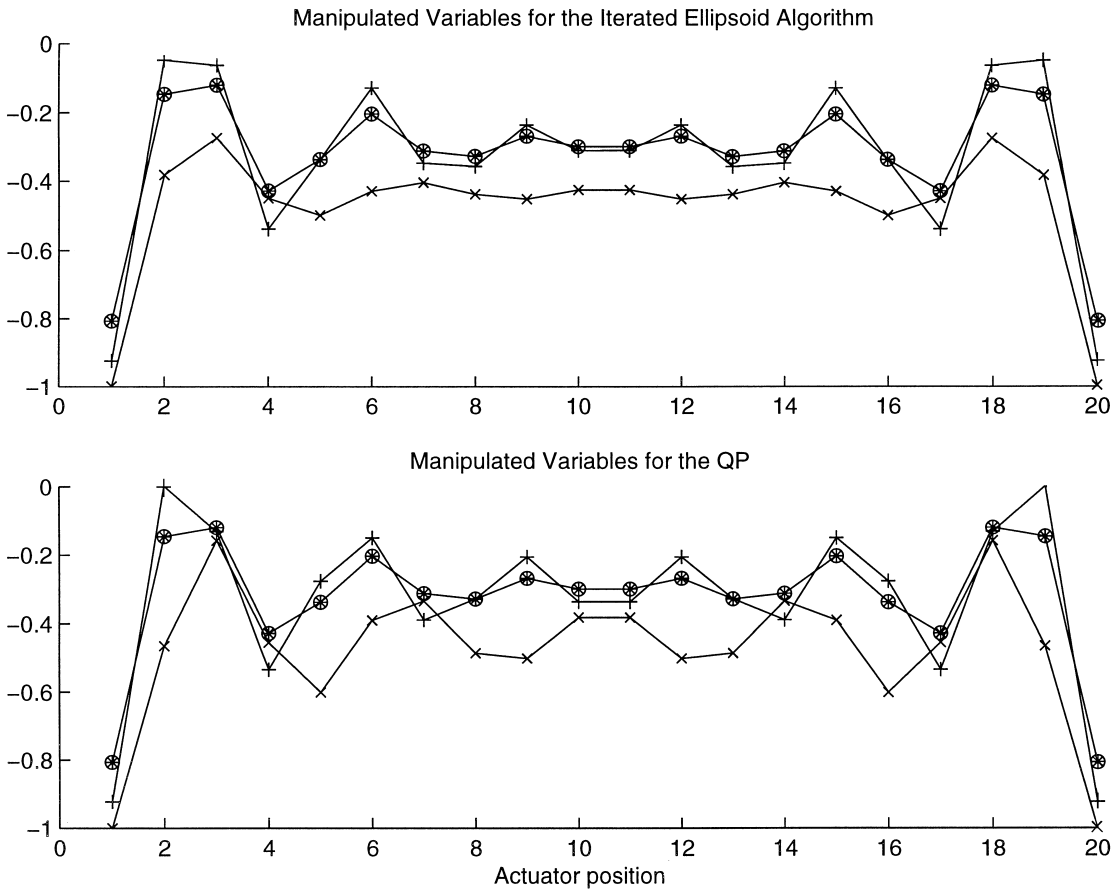

Fig. 3. Manipulated variables given by the QP and IE algorithms: $t=1(\times)$, $t=(+)$, $t=3$ (o), $t=4$ (*).

and same problem. Ref. [6] also describes an approximate LP algorithm (called the Vertex LP in Tables 5–8 of [6]) whose solution times are less than 4.91 CPU s for all disturbances considered. After scaling for computer speed and the number of decision variables, we estimate that the IE algorithm would require 20% of the computation time of the approximate LP algorithm if run on the same computer and same problem. Note that the exact and approximate LP algorithms of [6] explicitly exploit structure in the plant interactions matrix in solving the optimization problem, while the IE algorithm does not. Another consideration that should be stressed in the comparison is that the exact fast LP algorithm provides a global solution, while the IE algorithm computes a suboptimal solution.

Based on the FFT and TFFTdp benchmarks, the Alphastation 250 used by [7] is roughly 1.8 times as fast as the Sun Ultra 1. Rao et al studied a 100×100 process with bound constraints on the input with a control horizon of 10. This gives the same number of decision variables as the paper machine example with 200 actuators (since we consider a control horizon of 5). To solve the on-line MPC optimization problem using Rao et al.'s sparse structured interior point (SSIP) method took 43.8 CPU s [7]. For the IE algorithm to solve a problem with the same number of decision variables took $52/10=5.2$ CPU s for one sampling instance. Factoring in the difference in computers, it is estimated that the IE algorithm would take 7% of the computation time of the SSIP method if run on the same computer for the same paper machine example. As discussed in the Algorithm section, the IE algorithm will be faster for problems with large numbers of manipulated variables, while the Rao et al algorithm will be faster for algorithms with long control horizons.

In both of the above comparisons, the authors would like to stress that the comparisons are approximate, and that the exact LP and the SSIP algorithms give globally optimal solutions, while the IE algorithm gives a suboptimal solution.

3.2. An ill-conditioned process

This example compares the performance of the IE algorithm to the QP and fast LA algorithms. This particular example was selected because it is a system for which approximate MPC algorithms can perform poorly.

Consider the following ill-conditioned system:

$$\mathbf{y}(s) = \frac{10}{4s+1}\begin{bmatrix} 4 & -3 \\ -5 & 4 \end{bmatrix}\mathbf{u}(s) \tag{22}$$

The process condition number is 66. Both inputs are constrained $|u_i(k)| \leqslant 1$, $i = 1, 2$. The system is approximated by 20 step response coefficients and a sampling time of 1. The following tuning parameters are used (see [8] for MPC problem formulation):

$$m = 5; p = 10; \mathbf{Q}_y = \mathbf{I}; \mathbf{Q}_u = 0.1\mathbf{I}. \tag{23}$$

The setpoint is changed from (0,0) to (18,−22). The response of the system for three different control algorithms is shown in Fig. 4.

Although the purpose of this example was to compare the performance of the algorithms, for completeness, the solution times are shown in Table 1.

Zheng's LA algorithm assumes that inputs $\mathbf{u}(k+1), \ldots, \mathbf{u}(k+m-1)$ are unconstrained and then makes a linear approximation to the QP. This gives it $O(n^2)$ growth which is the same as for the IE algorithm. However, the IE algorithm does a better job of approximating the QP as seen in Fig. 4.

Fig. 5 shows the objective function values for the QP and IE algorithms and the lower bound for control horizons of 1 and 2. For a control horizon of 1, the equality of the IE objective and the lower bound gives an on-line indication that replacing the QP algorithm with the IE algorithm gives no conservatism. For a control horizon of 2, the objective function values are similar except for the first time step. The IE algorithm actually gives a lower objective function value than the QP algorithm at $k = 2$ and $k = 4$ (this is possible because model predictive control is an open loop optimization approach, and the computations after $k = 1$ have different manipulated and measured variable histories). For $k > 1$, the closeness of the IE objective and the lower bound computed on-line indicates that the conservatism of the IE algorithm is small. The gap between the IE objective and the lower bound for $k = 1$ is caused by the high condition number of the plant matrix in (22).

Table 1
Solution times for the ill-conditioned MPC problem on a Sparc 20 (75 Mhz) workstation

Algorithm	Solution time (s)
QP	0.016
IE	0.0053
LA	0.0049

Our numerical experience is that the lower bound tends to become more conservative as the process directionality and/or the process dimension increases. For example, the lower bound is too small to be useful for the simplified paper machine model, although the closed loop performance of the IE algorithm is similar to that of the QP algorithm.

The MQP algorithm of Zheng applied to a similar ill-conditioned system gave performance very similar to that obtained by the QP algorithm [8]. Existing simulation examples seems to imply that the MQP algorithm is the most accurate "approximate QP" algorithm to date. As shown in the simplified paper machine example, the IE algorithm is faster than the MQP algorithm for large scale MPC problems, but not as accurate. The LA algorithm is the fastest "approximate QP" algorithm to date. The ill-conditioned example seems to imply that the IE algorithm is more accurate than the LA algorithm, but not as fast. In comparison to the MQP and LA algorithms, the IE algorithm is intermediate in both

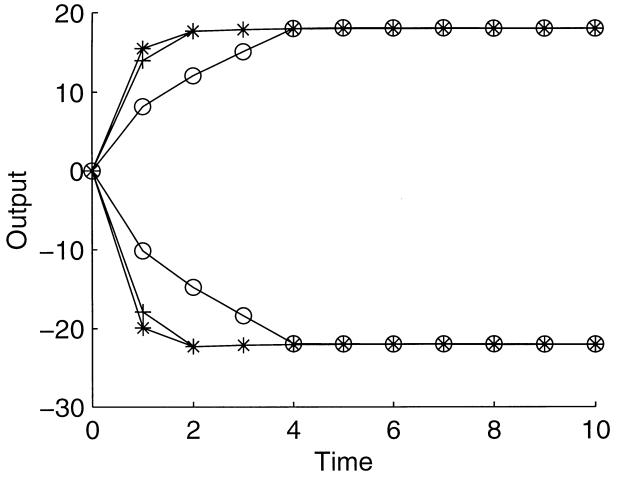

Fig. 4. System responses for the QP, IE, and LA algorithms. QP(*), IE (+), and LA (o).

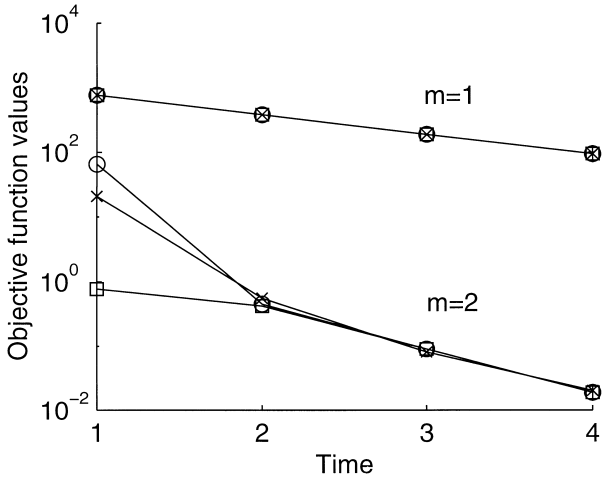

Fig. 5. Objective function values for the QP, IE, and lower bound (LB) for a control horizon of 1 and a control horizon of 2. QP (×), IE (o), and LB (square).

computational speed and accuracy for problems with a large number of manipulated variables.

4. Conclusions

The need to solve large MPC problems quickly and accurately motivated the development of the IE algorithm. The algorithm approximates the polytopic constraints of the MPC QP with an ellipsoid, and then rescales the ellipsoid to reduce conservatism. A change of variables is performed to give an equivalent problem with a simpler structure for which an optimal solution can be efficiently computed via bisection. The IE algorithm is faster than the MQP and structured QP algorithms for processes with large numbers of manipulated variables, and performs better than the LA algorithm for an ill-conditioned system.

Another application of the IE algorithm is to jump-start a more sophisticated optimization algorithm. For example, the manipulated variables computed from the IE algorithm could be used as an initial guess for an interior point algorithm [7]. The IE algorithm could also be combined with other fast MPC algorithms, such as the MQP algorithm [8], to form an even faster hybrid algorithm.

Acknowledgements

This project was supported by the UIUC Computational Science and Engineering Program and the DuPont Young Faculty Award.

References

[1] J.W. Eaton, J.B. Rawlings, Model predictive control of chemical processes, Chem. Eng. Sci. 47 (1992) 705–720.
[2] C.E. Garcia, D.M. Prett, M. Morari, Model predictive control: theory and practice — a survey, Automatica 25 (1989) 335–348.
[3] N.L. Ricker, Model predictive control with state estimation, Ind. Eng. Chem. Res. 29 (1990) 374–382.
[4] R.D. Braatz, M.L. Tyler, M. Morari, F.R. Pranckh, L. Sartor, Identification and cross-directional control of coating processes, AIChE J. 38 (1992) 1329–1339.
[5] R.D. Braatz, J.G. VanAntwerp, Advanced cross-directional control, Pulp & Paper Canada 98 (7) (1997) T237–239.
[6] P. Dave, F.J. Doyle III, J.F. Pekny. Specialized programming methods in the model predictive control of large scale systems, J. of Process Control, in press.
[7] C.V. Rao, J.C. Campbell, J.B. Rawlings, S.J. Wright. Efficient implementation of model predictive control for sheet and film forming processes. in: Proc. of the American Control Conf., Piscataway, NJ, IEEE Press, 1997, pp. 2940–2944.
[8] A. Zheng. Constrained model predictive control: is QP necessary? J. of Process Control, in press.
[9] L.G. Khachiyan, M.J. Todd, On the complexity of approximating the maximal inscribed ellipsoid for a polytope, Math. Prog. 61 (1993) 137–159.
[10] Y. Nesterov, A. Nemirovskii. Studies in Applied Mathematics, Vol. 13, Interior Point Polynomial Algorithms in Convex Programming, SIAM, Philadelphia, PA, 1994.
[11] S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan. Studies in Applied Mathematics, Vol. 15, Linear Matrix Inequalities in System and Control Theory, SIAM, Philadelphia, PA, 1994.
[12] R.G. Bland, D. Goldfarb, M.J. Todd, The ellipsoid method: a survey, Operations Research 29 (1981) 1039–1091.
[13] A.L. Peressini, F.E. Sullivan, J.J. Jr Uhl, The Mathematics of Nonlinear Programming, Springer-Verlag, New York, 1988.
[14] IMSL, Visual Numerics, Inc., 1997 (computer software).
[15] D. Goldfarb, A. Idnani, A numerically stable dual method for solving strictly convex quadratic programs, Math. Prog. 27 (1983) 1–33.
[16] M.J.D. Powell, On the quadratic programming algorithm of Goldfarb and Idnani, Mathematical Programming Study 25 (1985) 46–61.
[17] N.L. Ricker, Model-predictive control of processes with many inputs and outputs, Control and Dynamic Systems 37 (1990) 217–269.
[18] R.D. Braatz, B.A. Ogunnaike, A.P. Featherstone. Identification, estimation, and control of sheet and film processes, in: Proc. of the IFAC World Congress, Tarrytown, New York, 1996. Elsevier Science Inc., pp. 319–324.