TWMCC ⋆ Texas-Wisconsin Modeling and Control Consortium

Technical report number 2006-01

The Partial Enumeration Method for Model Predictive Control: Algorithm and Examples

Gabriele Pannocchia*
Department of Chemical Engineering, Industrial Chemistry and Science of Materials
University of Pisa
Via Diotisalvi, 2
Pisa 56126 (Italy)

James B. Rawlings†
Department of Chemical and Biological Engineering
University of Wisconsin-Madison
1415 Engineering Drive
Madison, WI 53706 (USA)

Stephen J. Wright‡
Computer Science Department
Univerisity of Wisconsin-Madison
1210 West Dayton Street
Madison, WI 53706 (USA)

March 22, 2006

*email: g.pannocchia@ing.unipi.it
†email: jbraw@che.wisc.edu
Author to whom all correspondence should be addressed.
‡email: swright@cs.wisc.edu.

Abstract

Partial enumeration is presented as a method for treating large, linear model predictive control applications that are out of reach with available MPC methods. PE uses both a table storage method and online optimization to achieve this goal. Versions of PE are shown to be closed-loop stable. PE is applied to an industrial example with more than 250 states, 30 inputs, and a 25-sample control horizon. The performance is less than 0.01% suboptimal, with average speedup factors in the range of 80–220, and worst case speedups in the range of 4.9–39.2, compared to an existing MPC method. Small tables with only 25–200 entries were used to obtain this performance, while full enumeration is intractable for this example.

Keywords

Model predictive control, On-line optimization, Large scale systems

1 Introduction

It is well known that a disadvantage of optimal control is its inherent complexity [10]. Unless a problem has special structure (such as the linear, unconstrained models that produce the classic LQ regulator), the evaluation and online implementation of the optimal feedback control presents a daunting challenge. Due to its roots in optimal control, model predictive control (MPC) certainly inherits this disadvantage.

As reviewed by Mayne et al. [15], most of the early industrial implementations of MPC traded the offline complexity of solving and storing the entire optimal feedback control law $u(x)$ for the online complexity of solving the open-loop optimal control problem given the particular x initial condition of interest at any given sample time. For linear models, the online problem is a quadratic program (QP), and efficient QP solvers allowed practitioners to tackle processes with small to moderate model dimension and control horizon. See the following papers for further discussion of methods for efficient online solution of the MPC QPs: [22], [6], [4], [1], [14].

Recently, researchers have developed interesting methods for solving and storing the closed-loop feedback law for linear, *constrained* models [2, 27, 28, 29]. These techniques work well for problems of low dimension. Some authors are exploring ideas for storing feedback solutions also for nonlinear models [16, 5, 3, 9]. The issue of complexity remains, however. The information to be stored for looking up the optimal solution grows exponentially in the dimension of the process model and the control horizon. Practitioners face the following obstacle to further progress. Small problems are well-addressed by both online methods (solving small dimensional QPs online is tractable) and offline methods (finding and storing the full solution offline and looking up the stored solution online are tractable). But, as the problem size increases, eventually neither online nor offline MPC computational approaches can deliver the control decision in the available sample time.

It is this class of large problems that we address with the partial enumeration (PE) technique of this paper. We eschew optimal control for something that may be slightly suboptimal, but which

remains tractable for real-time implementation on large problems. We combine online optimization and storage methods to achieve this end. The following features of the control problem must be present for our approach to be effective: (i) The large magnitude deterministic disturbances and setpoints signals change reasonably slowly with time. (ii) The stochastic disturbances, which change quickly with time, have reasonably small magnitude. As the name "partial enumeration" suggests, we determine which active constraint sets appear with highest frequency, given a reasonable collection of disturbances and setpoint changes. The optimal solution for these frequently occurring active constraint sets are computed offline and stored in a table. This table is searched online for the best control. During online operation we expect the optimal solution to be missing from the table at many sample times because the number of entries in the table is small compared to the number of possible optimal active sets. However, the table is adapted online to incorporate new active sets as the need for them arises. When the optimal solution is not found at a sample time, a simpler, suboptimal strategy is used for the current control, but the optimal solution is also found and inserted in the table. It does not matter if finding this optimal solution takes several samples. When this optimal solution is available, it is inserted in the table and the least recently used optimal solution is discarded so the table size remains small, to allow for efficient online searching. In this way, the table adapts itself as different disturbance scenarios are encountered. At most sample points, the solution is found in the table (giving an optimal control), but by not enforcing strict optimality at every sample, the control can be computed quickly even for large problems.

Under the category of methods that approximate the solution of the MPC QP, Kouvaritakis et al. [13] proposed a method for fast computation of a suboptimal input sequence by means of an inner ellipsoidal approximation to the polyhedral constraint set. They later developed heuristic methods to search outside the ellipsoid for a less conservative solution [12]. A current limitation of this method is that it does not allow the origin of the system to be shifted without solving a large semidefinite program (SDP). This method is therefore not applicable to the problem considered here, because we augment the system with an integrating disturbance model to achieve offset-free control (see section 2.1). The estimate of the integrating disturbance shifts the origin of the system at *every* sample time.

Rojas et al. [24] described a technique for approximately solving the MPC QP by performing an offline SVD of the Hessian of the objective function, and using it for an online change of coordinates. Basis vectors of the SVD are added until the next addition violates an input constraint. This approach assumes that the origin (steady state) is strictly inside the feasible region. This method is not well suited for the applications considered here because the steady state is often constrained, so the origin is usually on the boundary of the feasible region (see section 4.2).

In the remainder of the paper, we define a precise PE algorithm, present some of its theoretical closed-loop properties, and demonstrate the effectiveness of PE on some reasonably large-scale, industrially relevant examples. In the examples we explore the following issues: What is a good choice for the table size, how often is the optimal solution in the table, how suboptimal is the closed-loop performance, and how quickly can PE solve large control problems compared to current methods.

2 Model Predictive Control

2.1 Controller formulation

We consider a linear time-invariant system model, augmented with integrating states to remove offset [20]:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k \\ d_{k+1} &= d_k \\ y_k &= Cx_k + C_d d_k \,, \end{aligned} \tag{2.1}$$

in which $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the input, $y_k \in \mathbb{R}^p$ is the measured output, $d_k \in \mathbb{R}^p$ is the additional integrating state, and the matrices A, B, C, B_d, C_d are fixed matrices with appropriate dimensions. Given the measured output, and the previous estimate of the augmented state (defined later on), the "updated" estimate is computed as:

$$\begin{bmatrix} \hat{x}_{k|k} \\ \hat{d}_{k|k} \end{bmatrix} = \begin{bmatrix} \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(y_k - \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} \right) , \tag{2.2}$$

in which the matrices $L_x \in \mathbb{R}^{n\times p}$ and $L_d \in \mathbb{R}^{p\times p}$ can be computed by an appropriate Riccati equation (see e.g. [20]). We also denote with $\{\hat{x}_{k+j|k}\}$ and $\{\hat{d}_{k+j|k}\}$ (with $j > 0$) the corresponding predicted sequences obtained using the model (2.1) for a given input sequence $\{u_{k+j|k}\}$. We assume that a linear transformation $z = H_z y$ (usually a subvector) of the measured output variable vector y have known setpoints $\bar{z}$. Without loss of generality, we consider the following "hard" input constraints:

$$Du_k \leq d \,, \tag{2.3}$$

in which $D \in \mathbb{R}^{q\times m}$ and $d \in \mathbb{R}^q$ are specified by the user.

Remark 1. *"Hard" constraints as in* (2.3) *can be used to represent input bound constraints. Input rate-of-change constraints can be written as "hard" mixed constraints in the form $Du_k - Gx_k \leq d$ using a state augmentation with the past input (see e.g. [23]). Output constraints, instead, should be considered as "soft" state constraints, since in practice it is not possible to guarantee feasibility of such constraints. Hard mixed constraints and soft state constraints can be readily included in the controller formulation but are omitted for simplicity of presentation.*

A target calculation module is used at each sampling time to compute the state and input steady-state that drive the controlled variables to their setpoints while respecting the constraints (2.3). Several different formulations can be chosen for target calculation, for example, linear or quadratic objective function, hard constraints with precedence rank ordering, or soft constraints. In this work we choose the following "two-stage" target calculation scheme [19]. First, we try to solve the following quadratic program:

$$\min_{\bar{u}_k, \bar{x}_k} \frac{1}{2} (\bar{u}_k - \bar{u})' R_s (\bar{u}_k - \bar{u}) \tag{2.4a}$$

subject to:

$$\bar{x}_k = A\bar{x}_k + B\bar{u}_k + B_d\hat{d}_{k|k} \tag{2.4b}$$
$$\bar{z} = H_z(C\bar{x}_k + C_d\hat{d}_{k|k}) \tag{2.4c}$$
$$D\bar{u}_k \leq d, \tag{2.4d}$$

in which R_s is a positive definite matrix, $\bar{u}$ represents the desired setpoint for the inputs, and $\bar{z}$ represents the desired output setpoints. If (2.4) is infeasible, we conclude that, for the current integrating state estimate, we cannot track the controlled variables z to the given setpoint vector $\bar{z}$ without offset. In the latter case, we solve a second quadratic program aimed at minimizing the offset:

$$\min_{\bar{u}_k,\bar{x}_k,\bar{z}_k} \frac{1}{2}\left\{(\bar{z}_k - \bar{z})'Q_s(\bar{z}_k - \bar{z}) + (\bar{u}_k - \bar{u})'R_s(\bar{u}_k - \bar{u})\right\} \tag{2.5a}$$

subject to: (2.4b), (2.4d) and

$$\bar{z}_k = H_z(C\bar{x}_k + C_d\hat{d}_{k|k}), \tag{2.5b}$$

where Q_s is a positive definite matrix.

Having calculated the targets, we formulate the MPC subproblem to be solved at each decision timepoint by defining deviation variables w_j and v_j as follows:

$$w_j = \hat{x}_{k+j|k} - \bar{x}_k, \quad v_j = u_{k+j|k} - \bar{u}_k. \tag{2.6}$$

The MPC optimization problem is then defined as follows:

$$\min_{\{w\}_{j=1}^{N},\{v\}_{j=0}^{N-1}} \sum_{j=0}^{N-1} \frac{1}{2}\left\{w_j'Qw_j + v_j'Rv_j + 2w_j'Mv_j\right\} + \frac{1}{2}w_N'Pw_N \tag{2.7a}$$

subject to:

$$w_{j+1} = Aw_j + Bv_j, \qquad j = 0,\ldots,N-1, \tag{2.7b}$$
$$Dv_j \leq d - D_s b_s, \qquad j = 0,\ldots,N-1, \tag{2.7c}$$

where D_s is defined as $D_s = \begin{bmatrix} D & 0 \end{bmatrix}$, b_s is defined as $b_s = \begin{bmatrix} \bar{u}_k' & \bar{x}_k' \end{bmatrix}'$, R is positive definite, the matrices $\hat{Q} = Q - MR^{-1}M'$ and P are positive semidefinite, and $(\hat{A}, \hat{Q}^{1/2})$ is detectable where $\hat{A} = A - BR^{-1}M'$. Notice that (2.7c) corresponds to enforcing the hard constraints (2.3) over the control horizon N. Moreover, it is important to remark that from (2.4d) it follows that $d - D_s b_s \geq 0$.

Remark 2. *The quantities b_s and w_0 are* parameters *in this formulation, and in general both will change at each decision timepoint. The other quantities Q, R, M, P, A, B, D, D_s, and d remain constant for purposes of our calculations.*

Remark 3. *In the nominal case, i.e. when the initial state is known and the actual plant satisfies* (2.1) *with $d_k = 0$ for all k, it follows that $\hat{d}_{k|k} = 0$ for all k. Hence, the target b_s, computed either from* (2.4) *or* (2.5)*, remains constant at each decision timepoint (unless the setpoint $\bar{z}$ is changed). When the actual system differs from* (2.1) *and/or the initial state is not known, $\hat{d}_{k|k}$ may change at each decision timepoint, and so does b_s.*

We can use (2.7b) to eliminate the state variables $w = \begin{bmatrix} w_1' & \cdots & w_N' \end{bmatrix}'$ from the formulation. By doing so we obtain the following strictly convex quadratic program:

$$\min_{v} \frac{1}{2} v' \mathbf{H} v + \mathbf{g}' v \tag{2.8a}$$

subject to

$$\mathbf{A} v \geq \mathbf{b}, \tag{2.8b}$$

where $v = \begin{bmatrix} v_0' & \cdots & v_{N-1}' \end{bmatrix}'$; $\mathbf{H}$ (positive definite) and $\mathbf{A}$ are constant matrices whose definitions are reported in Appendix A. The vectors $\mathbf{g}$ and $\mathbf{b}$ depend on the parameters b_s and w_0 as follows:

$$\mathbf{g} = \mathbf{G}_w w_0 \tag{2.9a}$$

$$\mathbf{b} = \bar{\mathbf{b}} + \mathbf{B}_s b_s, \tag{2.9b}$$

where $\bar{\mathbf{b}}$ is a constant vector while $\mathbf{G}_w$ and $\mathbf{B}_s$ are constant matrices (definitions reported in Appendix A). Given the optimal solution v^* obtained from either (2.7) or (2.8), we use (2.6) to recover the first input u_k, that is,

$$u_k = \bar{u}_k + v_0^*, \tag{2.10}$$

and we inject u_k into the plant.

Remark 4. *If the system matrix A is not strictly Hurwitz, the numerical solution of* (2.8) *may be difficult due to ill-conditioning. It is therefore recommended to reparameterize the inputs in* (2.6) *as follows (see [25]):*

$$v_j = (u_{k+j|k} - \bar{u}_k) - L w_j, \tag{2.11}$$

where L is chosen such that $A + BL$ is strictly Hurwitz and (2.7a)–(2.7c) *are modified accordingly.*

2.2 Solving the MPC Subproblem

The problem (2.7) or its alternative formulation (2.8) must be solved once at each decision timepoint. Each problem differs from the ones that precede and follow it only in the parameters b_s and w_0. The unique solution to (2.8) must satisfy the following optimality (KKT) conditions:

$$\mathbf{H} v^* + \mathbf{g} - \mathbf{A}_a' \lambda_a^* = 0 \tag{2.12a}$$

$$\mathbf{A}_a v^* = \mathbf{b}_a \tag{2.12b}$$

$$\mathbf{A}_i v^* \geq \mathbf{b}_i \tag{2.12c}$$

$$\lambda_a^* \geq 0\,, \tag{2.12d}$$

for some choice of active constraint indices a, whose complement i denotes the inactive constraint indices. Assuming, for the moment, that the matrix $\mathbf{A}_a$ has full row rank (that is, the active constraints are linearly independent), we note that the first two equations yield a linear system with a square nonsingular coefficient matrix, namely,

$$\begin{bmatrix} \mathbf{H} & -\mathbf{A}_a' \\ \mathbf{A}_a & 0 \end{bmatrix} \begin{bmatrix} v^* \\ \lambda_a^* \end{bmatrix} = \begin{bmatrix} -\mathbf{g} \\ \mathbf{b}_a \end{bmatrix}. \tag{2.13}$$

The variables v^* and λ_a^* are uniquely determined by this equation.

The options for solving this problem are of three basic types.

1. Solution of the formulation (2.8) using active-set techniques for quadratic programming such as `qpsol`.
2. Solution of the (larger but more structured) problem (2.7) using an interior point quadratic programming solver.
3. The multi-parametric quadratic programming (mp-QP) approach.

The first approach has the advantage of compactness of the formulation; the elimination of the states can reduce the dimensions of the matrices considerably. However, $\mathbf{H}$ and $\mathbf{A}$ are in general not sparse, and since their dimension is proportional to N, the time to solve (2.8) is usually $O(N^3)$, since at least one factorization of a coefficient matrix like the one in (2.13) is required. Active-set methods such as `qpsol` essentially search for the correct active set a by making a sequence of guesses, adding and/or removing an element from a at each iteration. If the inequalities (2.12c) and (2.12d) are satisfied by the solution v^* and λ_a^* corresponding to the current a, the algorithm terminates with a solution. Otherwise, the violations of these conditions provide guidance as to which indices should be added to or dropped from a. Milman and Davison [18, 17] describe an active-set approach based on (2.8) in which changes to a are made in "blocks," feasibility is not enforced at every iterate, and active set changes are made preferentially in the early part of the time interval. Computational results are presented to show convergence in fewer iterations than conventional active-set methods, but no supporting theory is given. Since we wish to solve a sequence of problems of the form (2.8) which differ only in the parameters b_s and w_0 in (2.9), it is possible to "warm start" an active-set method using information from the instance of (2.8) at the previous decision point. This approach has the potential of avoiding a fresh factorization of a matrix of size $O(N)$, and when the changes in b_s and w_0 are small, the solution of the current problem might be obtainable in a few active-set steps and as little as $O(N^2)$ computation time. However, current implementations seem not to exploit information from the previous decision point to this extent.

Details of the second approach are described in [22]. In this technique, the highly structured nature of problem (2.7) is exploited by using a stagewise ordering of the primal and dual variables. A banded Gaussian elimination algorithm is applied to the systems of linear equations that are solved to obtain the primal-dual step at each iteration.

In the third approach [2], the dependence of the problem on the parameters b_s and w_0 is used to express the solution explicitly in terms of these parameters, and to partition the space occupied by

(b_s, w_0) into polyhedral regions within which these expressions are valid. Since our partial enumeration scheme is related to this approach, we sketch it here. From (2.13), we have by substituting from (2.9) that

$$\begin{bmatrix} \mathbf{H} & -\mathbf{A}_a' \\ \mathbf{A}_a & 0 \end{bmatrix} \begin{bmatrix} v^* \\ \lambda_a^* \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{\mathbf{b}}_a \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B}_{as} \end{bmatrix} b_s + \begin{bmatrix} -\mathbf{G}_w \\ 0 \end{bmatrix} w_0, \tag{2.14}$$

where $\bar{\mathbf{b}}_a$ and $\mathbf{B}_{as}$ represent the row subvector/submatrix corresponding to the active set a. Assuming again for the present that $\mathbf{A}_a$ has full row rank, it is clear that the solution of (2.14) depends linearly on w_0 and b_s, so we can write

$$v^* = \mathbf{K}_{as} b_s + \mathbf{K}_{aw} w_0 + \mathbf{c}_v, \tag{2.15a}$$

$$\lambda_a^* = \mathbf{L}_{as} b_s + \mathbf{L}_{aw} w_0 + \mathbf{c}_\lambda, \tag{2.15b}$$

where $\mathbf{K}_{as}$, $\mathbf{K}_{aw}$, $\mathbf{L}_{as}$, $\mathbf{L}_{aw}$, $\mathbf{c}_v$, and $\mathbf{c}_\lambda$ are quantities that depend on the active set a and the problem data but not on the parameters b_s and w_0. The region of validity of the solutions in (2.15) is determined by the remaining optimality conditions (2.12c) and (2.12d). By substituting from (2.15), we can transform these conditions into explicit tests involving the parameters, as follows:

$$\mathbf{L}_{as} b_s + \mathbf{L}_{aw} w_0 + \mathbf{c}_\lambda \geq 0, \tag{2.16a}$$

$$\mathbf{A}_i \mathbf{K}_{as} b_s + \mathbf{A}_i \mathbf{K}_{aw} w_0 + (\mathbf{A}_i \mathbf{c}_v - \mathbf{b}_i) \geq 0. \tag{2.16b}$$

These formulae motivate a scheme for solving the parameterized quadratic programs that moves most of the computation "offline." In the offline part of the computation, the coefficient matrices and vectors in formulae (2.15) and (2.16) are calculated and stored for all sets a that are valid active sets for some choice of parameters (b_s, w_0). In the online computation, the active set a for which (2.16) holds is identified, for a given particular value of (b_s, w_0). Having found the appropriate a, the optimal v^* and λ_a^* are then computed from (2.15).

For the offline computation, Bemporad et al. [2] describe a scheme for traversing the space occupied by (b_s, w_0) to determine all possible active sets a. In general, there will be values of (b_s, w_0) for which more than one active set a allows satisfaction of the KKT conditions (2.12). In such cases, we may have either that $\mathbf{A}_a$ does not have full row rank and/or some components of λ_a^* are zero and/or some components of $\mathbf{A}_i v^* - \mathbf{b}_i$ are zero. It is not clear that the scheme proposed in [2, Section 4.1.1] is able to handle such situations appropriately. Tondel et al. [28] describe a more efficient scheme for finding all active sets a for which the set of (b_s, w_0) satisfying (2.16) is nonempty and has full dimension. They also show how degeneracies and redundancies can be removed from the description (2.16). Johansen and Grancharova [11] construct a partition of the space occupied by (b_s, w_0) into hypercubes (with faces orthogonal to the principal axes) and devise a feedback law for each hypercube that is optimal to within a specified tolerance. (Refinement of the hypercube partition is performed to maintain solution quality.) The nature of the tree allows it to be traversed efficiently during the online computation. Given the large number of possible active sets (potentially exponential in the number of inequality constraints), the online part of the computation may be slow if many evaluations of the form (2.16) must be performed before the correct a is identified. Tondel et al. [29] show how to construct a binary search tree in which the leaf nodes correspond to valid active sets, while branching at the internal nodes is determined by satisfaction or nonsatisfaction of an affine inequality.

We note that the technique described above can be applied also to formulation (2.7); the modifications required for handling equality constraints in addition to the inequalities are straightforward.

3 Partial Enumeration

3.1 Introduction

From a purely theoretical viewpoint, the complete enumeration strategy described above is unappealing, as it replaces a polynomial-time method for solving each one of the MPC problems (i.e. an interior-point method) by a method that is obviously not polynomial. Practically speaking, the offline computation—identification of all regions in the (b_s, w_0) space—may be tractable for SISO problems with relatively short control horizon N, but it quickly becomes intractable as the dimensions (m, n) and control horizon N grow. For the online computation, we have even for the carefully constructed search tree described in [29] that the number of evaluations of the inequalities (2.16) performed during a traversal of the tree is linear in N.[1]

The goal of an enumeration strategy is to make the online computations rapid, producing an input that is (close to) optimal, within the decision time allowed by the system. It is less important to minimize the amount of offline computation, although there should be, of course, reasonable limits on the amount of such computation. Our goal is to expand the size and complexity of systems for which enumeration strategies may be viable, by restricting the possible active sets a that are evaluated in the online computation to those that have arisen most often at recent decision points. Essentially, we propose to use the history of the system to improve the practical efficiency of the control computation. We have observed that on numerous large-size practical problems, the parameters (b_s, w_0) encountered at the decision points during a "time window" fall within a relatively small number of regions defined by the inequalities (2.16) over all possible active sets a. Hence, it would appear that the offline computations performed by complete enumeration, which involve a comprehensive partitioning of the (b_s, w_0) space, are largely wasted.

Our *partial enumeration* approach has the following key features:

1. We store the matrices and vectors in (2.15a) and (2.16) for only a small subset of possible active sets a, making no attempt to partition the entire space. These data are stored in a table of fixed length, with one table entry for each a.

2. We keep a count of how frequently each active set a in the table was optimal during the last T decision points (for some integer T). Given (b_s, w_0) at the current decision point, we search the entries in the table in order of decreasing frequency of correctness during the last T decision points.

3. Suppose now that none of the active sets a in the table passes the tests (2.16) for the given (b_s, w_0). In this case, we have some practical options to define the current control input.

[1] According to [29], the depth of the tree is logarithmic in the number of regions of distinct control laws in (b_s, w_0) space, which is generally exponential in the number of inequalities (2.8b), which is generally linear in N.

(a) We could solve a simplified subproblem (the same problem with a shorter control horizon) to obtain a suboptimal input in the time available.

(b) We could look for the "least suboptimal" a from the table and use this a to define an input. We choose this a such that the condition (2.16b) is satisfied and the control sequence given in (2.15a) yields the lowest objective function value.

(c) We could fall back on a control decision computed at an earlier decision point. We discuss this approach further in the next subsection.

4. Independently of the controller's calculation, we solve the MPC problem (2.7) (or (2.8)), to obtain the appropriate input and correct a. We compute the data for the (2.15a) and (2.16) corresponding to the optimal active set a and add it to the table. If the table thereby exceeds its maximum size, we delete the entry that was correct least recently.

5. The table is initialized by simulating the control procedure for a given number of decision stages (a "training period"), adding random disturbances in such a way as to force the system into regions of (b_s, w_0) space that are likely to be encountered during actual operation of the system.

The table should be large enough to fit the range of optimal active sets encountered during the current range of operation of the system, that is, to keep the number of "misses" at a reasonable level. When the system transitions to a new operating region (corresponding to a different part of the (b_s, w_0) space), some of the current entries in the table may become irrelevant. There will be a temporary spike in the proportion of misses. Once there has been sufficient turnover in the table, however, we can expect the fraction of misses to stabilize at a lower value. Naturally, we use the techniques described in [28] to eliminate redundant inequalities in (2.16).

3.2 Implementation

In order to give a proper description of how the partial enumeration MPC solver is implemented in this work, we need to describe two alternatives that are used to compute the control input when the optimal active set is not in the current table. The first one is the "restricted" or "reserve" control sequence, defined as:

$$v^{\text{res}} = (v_0^{\text{res}}, v_1^{\text{res}}, \ldots, v_{N-2}^{\text{res}}, v_{N-1}^{\text{res}}) = (v_1^*, v_2^*, \ldots, v_{N-1}^*, Kw_N^*), \tag{3.1}$$

where, in this definition, $\{v_j^*\}$ and w_N^* are the optimal inputs and terminal state computed at the previous decision timepoint. The second alternative is the "short control horizon" optimal control sequence defined as the optimal solution of (2.7) (or (2.8)) for a short horizon $\bar{N} < N$, chosen so that the time required to solve this problem is much shorter than that required to solve the problem with control horizon N. We denote this control sequence as:

$$v^{\text{sh}} = (v_0^{\text{sh}}, \ldots, v_{\bar{N}-1}^{\text{sh}}). \tag{3.2}$$

We can now give a formal definition of the proposed partial enumeration MPC algorithm.

Algorithm 1 (Partial enumeration MPC).
Data at time k: a table with $L_k \leq L_{\max}$ entries (each entry contains the matrices/vectors in the systems (2.15a)-(2.16)*, a time stamp of the last time the entry was optimal, a counter of how many times the entry was optimal), "restricted" sequence v^{res} (computed at time $k-1$), previous target $b_s^{\text{prev}} = \left[\vec{u}_{k-1}' \quad \vec{x}_{k-1}'\right]'$, current target b_s and deviation state w_0.*
Repeat the following steps:

1. *Search the table entries in the decreasing order of optimality rate. If an entry satisfies* (2.16) *for the given w_0 and b_s, compute the optimal control sequence from* (2.15a)*, define the current control input as in* (2.10)*, and go to Step 5. Otherwise,*

2. *If the following condition holds:*

$$\delta = \frac{\|b_s - b_s^{\text{prev}}\|_2}{1 + \|b_s\|_2} \leq \delta_{\max}, \tag{3.3}$$

   for a user-specified (small) positive scalar $\delta_{\max}$, define the current control input as:

$$u_k = v_0^{\text{res}} + \bar{u}_{k-1}, \tag{3.4}$$

   and go to Step 4. Otherwise,

3. *Solve the "short control horizon" MPC problem and define the current control input as:*

$$u_k = v_0^{\text{sh}} + \bar{u}_k. \tag{3.5}$$

4. *Solve* (2.7) *(or* (2.8)*) and compute the matrices/vectors in* (2.15a)-(2.16) *for the optimal active set. Add the new entry (possibly deleting the entry that was optimal least recently if the $L_k = L_{\max}$).*

5. *Update time stamp and frequency for the optimal entry. Define v^{res} for the next decision timepoint as in* (3.1)*.*

6. *Increase $k \leftarrow k+1$ and go to Step 1.*

Remark 5. *In Step 4, for systems with a large number of states the solution of* (2.8) *is to be preferred, while for systems with moderate state dimension and large control horizon the solution of* (2.7) *is expected to be more convenient [22].*

Remark 6. *Notice that for the calculation of the optimal control input in* (2.10)*, it is not necessary to store and use the full matrices/vectors in* (2.15a)*. The first m rows will suffice.*

Remark 7. *Notice that* (3.4) *and* (3.1) *imply that the current (fall-back) control input is defined as the corresponding optimal one computed at the previous decision timepoint, i.e. $u_k = u_{k|k-1}$.*

If, for the current deviation state/target (b_s, w_0), the optimal active set is not found in the table, then Step 4 is executed: The optimal input sequence is calculated for this (b_s, w_0) and the corresponding information is added to the table, in time (we assume) for the next stage $k+1$. When

these computations are not completed prior to the next sampling time, as may happen in practice, several modifications to the algorithm are needed. Since we need to adjust the reserve sequence v^{res} in Step 5 without knowing the results of Step 4, we leave this sequence unchanged, except to shift it forward one interval, as in (3.1). When, at some future time point, the computations for the given pair (b_s, w_0) are completed, we add the corresponding information to the table. We also update v^{res} with the newly available optimal sequence (with the obsolete entries for the earlier stages removed) provided that the reserve sequence was not updated in the interim as a result of finding an optimal sequence in the table. Moreover, it is important to notice that, in (3.3), b_s^{prev} denotes the target around which the reserve sequence was computed, and $\bar{u}_{k-1}$ in (3.4) is the corresponding input target. We report in Section 4 on an example in which there are delays in performing the computations in Step 4.

3.3 Properties

We now present the main theoretical properties of the proposed partial enumeration MPC algorithm. In particular, Theorem 4 states that the proposed partial enumeration MPC algorithm retains the nominal constrained closed-loop stability of the corresponding "optimal" MPC algorithm.

We assume that P in (2.7) is chosen as the optimal cost-to-go matrix of the following "unconstrained" linear quadratic infinite horizon problem:

$$\min_{w,v} \sum_{j=0}^{\infty} \frac{1}{2} \left\{ w_j' Q w_j + v_j' R v_j + 2 w_j' M v_j \right\} \tag{3.6a}$$

subject to:

$$w_{j+1} = A w_j + B v_j, \quad j = 0, 1, 2, \ldots. \tag{3.6b}$$

With such a choice of P and given the corresponding stabilizing gain matrix

$$K = -(R + B'PB)^{-1}(B'PA + M'), \tag{3.7}$$

the unconstrained "deviation" input and state evolves as:

$$v_j = K w_j, \quad w_{j+1} = A w_j + B v_j = A_K w_j, \quad j = 0, 1, 2, \ldots, \tag{3.8}$$

in which $A_K = A + BK$ is a strictly Hurwitz matrix, and the optimal objective value is $\frac{1}{2} w_0^T P w_0$. We also define O_∞ to be the set of target/initial state pairs such that the unconstrained solution (3.8) satisfies the constraints (2.7c) at all stages. Formally,

$$O_\infty = \{ (b_s, w) \,|\, D v_j \leq d - D_s b_s \text{ for all } v_j, w_j \text{ satisfying (3.8), with } w_0 = w \}. \tag{3.9}$$

We say that O_∞ has a *finite representation* if this infinite set of inequalities can be reduced to a finite set.

Remark 8. *The formulae* (3.8) *can be used to eliminate the v_j and w_j from this definition and write O_∞ as an infinite system of inequalities involving only w and b_s, as follows:*

$$O_\infty = \{ (b_s, w) \,|\, D_w A_K^j w \leq d - D_s b_s \text{ for all } j = 1, 2, \ldots \}, \tag{3.10}$$

where $D_w = DK$. Notice that O_∞ is non-empty since $(b_s, 0) \in O_\infty$ for any "feasible" target b_s, i.e. such that (2.4d) *holds.*

We now define the following "parameterized" sets:

$$W(b_s) = \{w \,|\, (b_s, w) \in O_\infty\}, \tag{3.11}$$

$$W_N(b_s) = \{w \,|\, w_j \text{ and } v_j, \text{ optimal solution to (2.7) with } w_0 = w, \text{ satisfy } w_N \in W(b_s)\}. \tag{3.12}$$

Notice that $W_N(b_s)$ is the set of initial states such that the optimal state trajectory enters $W(b_s)$ at most in N timesteps. We have the following results.

Lemma 1. *Assume that b_s satisfies $D_s b_s < d$, that $\tilde{W}(b_s) = \{w | D_w w \leq d - D_s b_s\}$ is bounded, and that A_K has all eigenvalues inside the open unit circle. Then, $W(b_s)$ is non-empty, is finitely determined and contains the origin in its interior.*

Proof. In the notation of [7], we set $A \leftarrow A_K$ and $C \leftarrow I$, the output constraint set Y is equal to $\tilde{W}(b_s)$, and $W(b_s)$ is $O_\infty(A, C, Y)$. Thus, finite determination of $W(b_s)$ follows as an application of [7, Thm. 4.1]. Moreover, from [7, Thm. 2.1] we have that the origin is in the interior of $W(b_s)$ because A_K is asymptotically stable (notice that only Lyapunov stable is required for this result) and $\tilde{W}(b_s)$ contains the origin in its interior since $D_s b_s < d$. Clearly, this also implies that $W(b_s)$ is non-empty. □

Lemma 2. *Suppose the assumptions of Lemma 1 hold and that P in* (2.7) *is chosen as the solution of the Riccati equation associated with* (3.6)*. We then have*

$$W(b_s) \subseteq W_1(b_s) \subseteq W_2(b_s) \subseteq \cdots. \tag{3.13}$$

Moreover, $W_N(b_s)$ is positively invariant for the system $\bar{w}_{j+1} = A\bar{w}_j + B\bar{v}_j$ with $\bar{v}_j$ defined as the first component of the control sequence solution of (2.7)*.*

Proof. Result (3.13) is proved by induction. Notice first that $W(b_s) \subseteq W_1(b_s)$, because if $\bar{w} \in W(b_s)$ the input sequence optimal solution to (2.7) with $N = 1$ and $w_0 = \bar{w}$ is simply $v^* = v_0^* = Kw_0 = K\bar{w}$, with K given in (3.7). [This value is feasible and yields the minimum cost $\frac{1}{2} w_0' P w_0$; see also [26, Lemma 1]]. Assume now that $\bar{w} \in W_l(b_s)$ for some $l > 0$ and let $v^* = (v_0^*, v_1^*, \ldots, v_{l-1}^*)$ and $w^* = (w_0^*, w_1^*, \ldots, w_l^*)$ be the corresponding optimal input and state sequences in (2.7) with $N = l$ and $w_0 = \bar{w}$. By definition of $W_l(b_s)$ we have that $w_l^* \in W(b_s)$, and therefore $v^* = (v_0^*, v_1^*, \ldots, v_{l-1}^*, Kw_l^*)$ and $(w_0^*, w_1^*, \ldots, w_l^*, A_K w_l^*)$ are the optimal input and state sequences in (2.7) with $N = l + 1$ and $w_0 = \bar{w}$. Moreover, $w_{l+1}^* = A_K w_l^* \in W(b_s)$ and therefore $\bar{w} \in W_{l+1}(b_s)$, that is $W_l(b_s) \subseteq W_{l+1}(b_s)$.

To prove the second claim, suppose that $\bar{w}_j \in W_N(b_s)$ and let $v^* = (v_0^*, v_1^*, \ldots, v_{N-1}^*)$ and $w^* = (w_0^*, w_1^*, \ldots, w_N^*)$ be the solution of (2.7) with $w_0 = \bar{w}_j$. We have that $\bar{w}_{j+1} = A\bar{w}_j + B\bar{v}_j = Aw_0^* + Bv_0^* = w_1^*$. It follows from $w_N^* \in W(b_s)$ that the solution of (2.7) with $w_0 = \bar{w}_{j+1}$ is simply $(v_1^*, \ldots, v_{N-1}^*, Kw_N^*)$ and $(w_1^*, w_2^*, \ldots, w_N^*, A_K w_N^*)$, and the terminal state $A_K w_N^*$ belongs to $W(b_s)$. Hence, $\bar{w}_{j+1} \in W_N(b_s)$, as claimed. □

We now state our two main results.

Theorem 3. *Suppose the assumptions of Lemma 1 hold. Assume that for the current deviation state w_0 there are infinite state and input sequences $\{w_j\}$ and $\{v_j\}$ that satisfy* (2.7b) *and* (2.7c)

(where these constraints hold for all *j) and such that the corresponding objective function in* (3.6) *is finite. Then, there exists a finite integer N' (which depends on w_0) such that*

$$w_0 \in W_N(b_s) \text{ for all } N \geq N'. \tag{3.14}$$

Proof. Since $\{w_j\}$ and $\{v_j\}$ are feasible for the infinite-horizon problem (3.6) subject to (2.7b) and (2.7c) (where these constraints hold for *all* j, not just for a finite horizon) and since the corresponding objective function is finite, it follows that there are optimal sequences $\{w_j^*\}$ and $\{v_j^*\}$ that solve this infinite-horizon problem [see e.g. [21, Th. A.3]]. Moreover since the optimal objective is finite and $(\hat{A}, \hat{Q}^{1/2})$ is detectable, we have that $(w_j^*, v_j^*) \to 0$ for this optimal sequence [see proof of Theorem 1 in [26] and also [8, p.251]]. Since under the assumptions of Lemma 1, $W(b_s)$ contains an open neighborhood of the origin, there exists a finite integer N' such that $w_j^* \in W(b_s)$ and $v_j^* = Kw_j^*$ for any $j \geq N'$. It follows that $\{w_j^*\}$ and $\{v_j^*\}$ also solves the problem (3.6), (2.7b), and (2.7c) in which the constraints (2.7c) are applied only at stages $j \leq N'$. Because of our choice of P, the objective (3.6a) is identical to (2.7a), with $N = N'$ and for the values $(w_0^*, w_1^*, \ldots, w_{N'}^*)$ and $(v_0^*, v_1^*, \ldots, v_{N'-1}^*)$. It follows that these truncated sequences are in fact optimal for the problem (2.7) with $N = N'$. Since $w_{N'}^* \in W(b_s)$, it follows that $w_0 \in W_{N'}(b_s)$. The proof is completed by using (3.13). □

Theorem 4. *The control input u_k computed from Algorithm 1 is feasible with respect to* (2.3) *for any current deviation state w_0 and target b_s. Moreover, under the assumptions of Lemma 2, the outlined procedure is closed-loop nominally stabilizing, that is, $(w_j, v_j) \to 0$ for any $w_0 \in W_N(b_s)$.*

Proof. For the first statement, we check that the constraint is satisfied whether the input is calculated in steps 1., 2., or 3. If in step 1., then u_k is obtained from (2.10) and (2.7), so the bound (2.3) is enforced explicitly. If in step 2., then from the problem (2.7) at the previous time step $k-1$, we have by setting $j = 1$ in (2.7c) that $Du_k = D\bar{u}_{k-1} + Dv_0^{\text{res}} = D\bar{u}_{k-1} + Dv_1^* \leq d$, as required. If step 3. is used, then again the bound $Du_k \leq d$ is applied explicitly in solving the short control horizon version of (2.7).

For the second statement, we have for the nominal case that $(\bar{x}_k, \bar{u}_k)$ is independent of k and that $b_s = b_s^{\text{prev}}$ at all steps. Hence, if the step is not computed in step 1., it is certainly assigned in step 3. In either case, the overall sequence of inputs will be identical to the sequence obtained by solving (2.7) at the first stage, with inputs beyond stage N obtained from (3.8) for $j \geq N$. (Feasibility of the "tail" of the sequence follows from $w_0 \in W_N(b_s)$.) It follows from our assumptions on A_K that $(w_j, v_j) \to 0$, as claimed. □

4 Application examples

In this section we present a number of examples to evaluate the proposed partial enumeration algorithm and compare it with a commercial active-set solver, `qpsol`. All simulations are performed using Octave[2] on a 1.2 GHz PC running Debian Linux, and time is measured with the function

[2] Octave (`www.octave.org`) is freely distributed under the terms of the GNU General Public License.

`cputime`. In all examples the terminal penalty P is chosen as the solution of the corresponding Riccati equation. Partial enumeration solvers are implemented, for backup calculation of a suboptimal input, with $\delta_{\max} = 0.0001$ and short control horizon of $\bar{N} = 3$.

For each partial enumeration solver the following performance indices are considered.

- Optimality rate:
$$O_R := \frac{N_{\text{opt}}}{N_{\text{tot}}} \tag{4.1}$$
where N_{opt} is the number of decision timepoints in which the optimal solution was found in the table and N_{tot} is the overall number of decision timepoints.

- Suboptimality index:
$$S_I := \frac{|\Phi - \Phi^*|}{\Phi^*} \tag{4.2}$$
where Φ is the achieved closed-loop objective function and Φ^* is the optimal one (obtained by injecting the optimal input at each decision timepoint).

- Average speed factor:
$$A_{SF} := \frac{T^*_{\text{aver}}}{T_{\text{aver}}} \tag{4.3}$$
where T^*_{aver} and T_{aver} are the average times required to compute the optimal input with `qpsol` and the (sub)optimal one with the partial enumeration solver, respectively.

- Worst-case speed factor:
$$W_{SF} := \frac{T^*_{\max}}{T_{\max}} \tag{4.4}$$
in which $T^*_{\max}$ and $T_{\max}$ are the maximum times required to compute the optimal input with `qpsol` and the (sub)optimal one with the partial enumeration solver, respectively.

4.1 Example #1

The first example is a stable system with $m = 3$ inputs, $p = 2$ outputs and $n = 12$ states, in which the sampling time is 1 sec and the normalized inputs must satisfy the constraints: $-\mathbf{1} \leq u_k \leq \mathbf{1}$. The gain matrix and system's eigenvalues are:

$$G = \begin{bmatrix} 1.000 & 0.0610 & 0.419 \\ -0.109 & 0.255 & -0.748 \end{bmatrix}$$
$$\lambda(A) = \{0.433 \pm 0.863i, \quad 0.670 \pm 0.675i, \quad 0.812 \pm 0.514i,$$
$$0.836 \pm 0.454i, \quad 0.737, \quad 0.819, \quad 0.935, \quad 0.908\}$$

The MPC regulator is designed with the following parameters: $N = 100$, $Q = C'C$, $R = 0.01 I_m$, $M = 0$, and thus the possible different active sets are $3^{200} = 2.656 \times 10^{95}$. Several partial enumeration solvers are compared: PE1, PE25, PE50, PE100 and PE200 use an active-set table of 1, 25, 50,

Table 1: Example #1 performance indices.

	O_R	S_I	A_{SF}	W_{SF}
SH	–	6.083	523	240
PE1	0.888	0.0580	302	230
PE25	0.890	0.0537	225	46.0
PE50	0.891	0.0537	181	20.9
PE100	0.892	0.0537	135	17.8
PE200	0.894	0.0537	89.6	14.4

100 and 200 entries, respectively. As a comparison another solver is also considered: SH always computes and injects the optimal solution of (2.8) with a short control horizon of $\bar{N} = 3$.

The different solvers are compared in a 20,000 decision timepoints simulation featuring 20 random step disturbances acting on the states, 20 random setpoint changes and normally distributed output noise. A snap-shot of the closed-loop trajectories for inputs and outputs obtained with the optimal solver, the PE25 solver and the SH solver are reported in Figure 1 and 2. In Table 1 we report the performance indices achieved by each solver. From this table it is evident that the use of a short control horizon, which certainly reduces the computational burden, can cause severe performance degradation (SH is more than 600% suboptimal), while partial enumeration allows a fast computation of the optimal solution most of the times, with an overall much minor performance degradation (for instance PE25 is 5.4% suboptimal). From Figures 1–2, we can notice that SH is driving the inputs and outputs away from their targets, i.e. the controller is not stabilizing in that part of the simulation.

In Figure 3 we present the cumulative frequency vs the number of entries scanned in the table by each partial enumeration solver. From Figure 3 we can notice that in order to find the optimal solution it is necessary to scan only a few table entries in most of decision timepoints, i.e. to check a limited number of active sets before finding the optimal one.

In Figure 4 we describe of the evolution in time of the active-set table for PE25 during the simulation, quantified by the following indices:

$$R_d(k,0) = \frac{D(k,0)}{N_e}, \quad R_d(k,k-N_e) = \frac{D(k,k-N_e)}{N_e} \tag{4.5}$$

where $D(k,j)$ is the number of table entries at time k that are not in the table at time j, while N_e is the number of all entries. From Figure 4 we can notice that the active-set table changes over time to adjust to new disturbances and operating conditions, where different active sets become "popular" and formerly popular active sets become "obsolete" (and eventually are taken out of the table). This adaptation is a key feature of the proposed method since it allows us, despite the fact that a large number of different active sets are visited (PE25 visited 2159 different active sets), to use small tables, thereby guaranteeing low maximum and average scanning time, but still obtaining a high optimal hit rate with nearly optimal closed-loop performance.

The effect of the delay in solving the MPC problem in Step 4 (which is executed when the current deviation state/target pair is not represented in the table) was investigated by introducing a

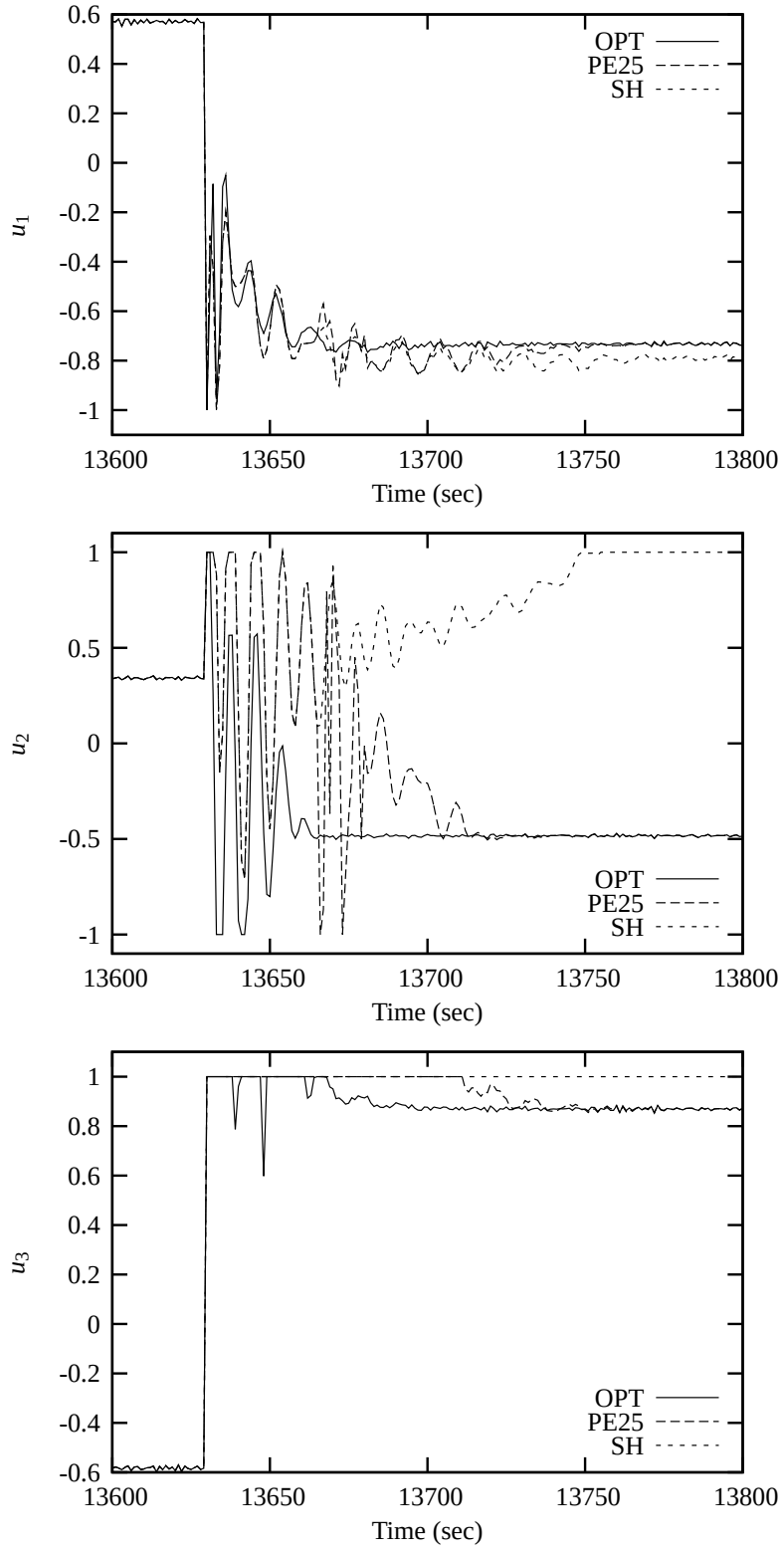

Figure 1: Closed-loop trajectories for inputs obtained with optimal, PE25 and SH solvers.

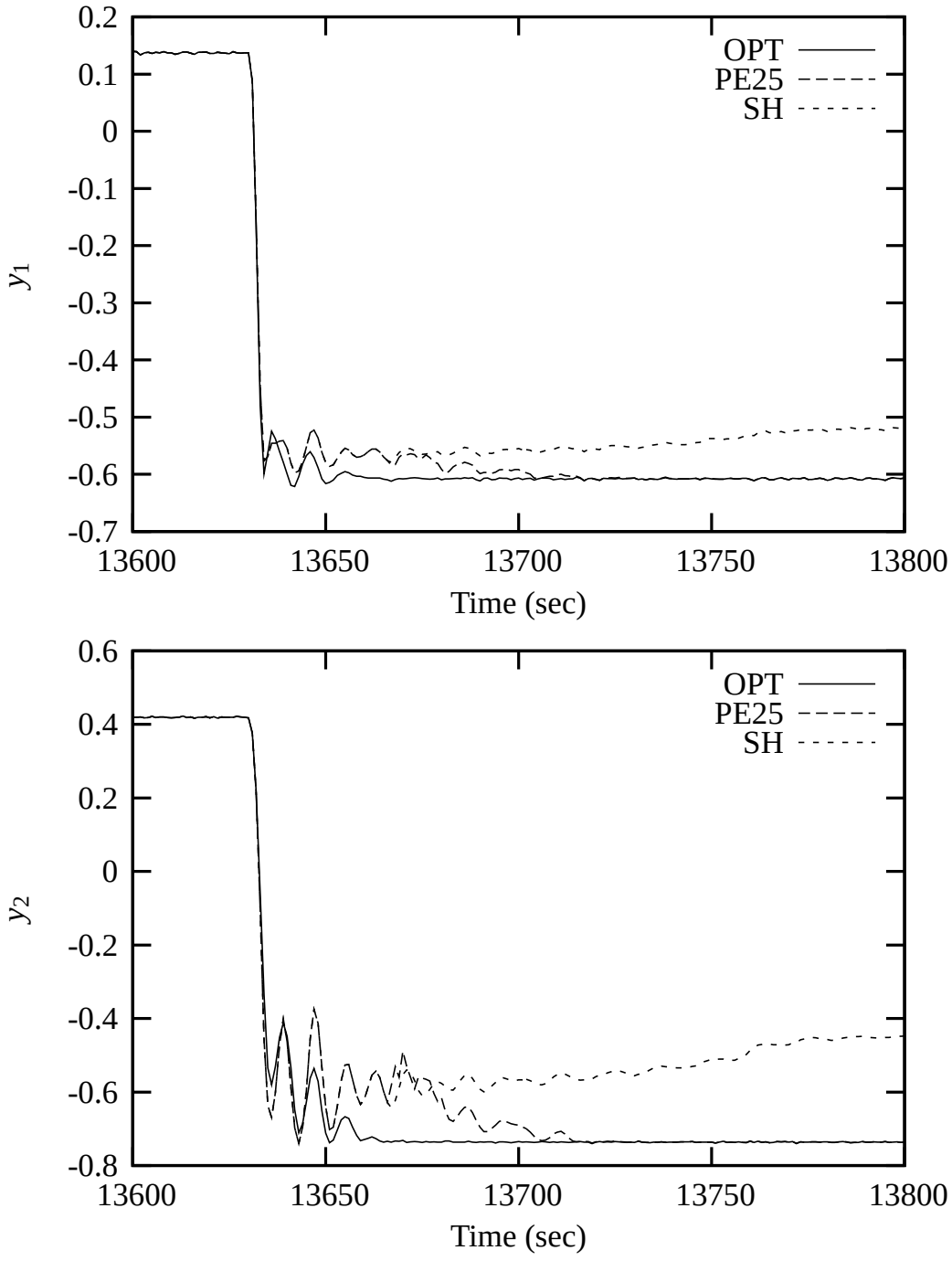

Figure 2: Closed-loop trajectories for outputs obtained with optimal, PE25 and SH solvers.

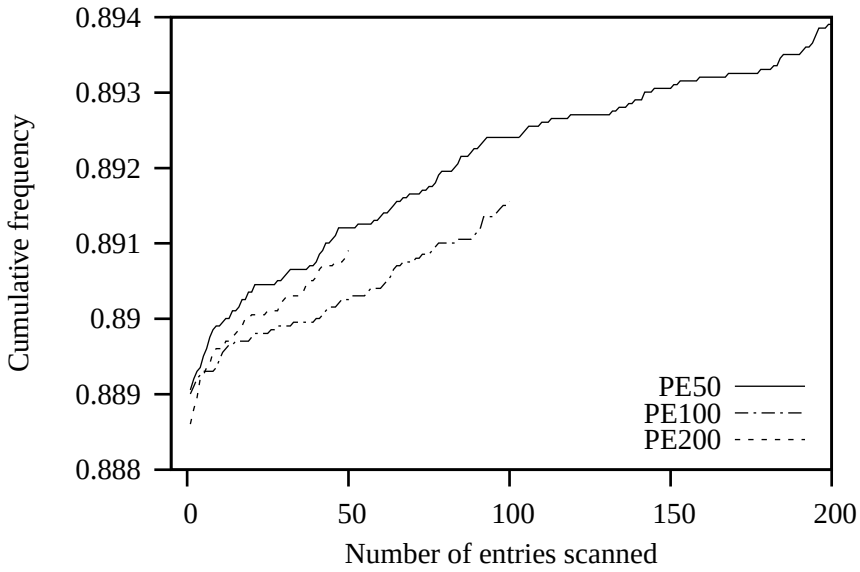

Figure 3: Example # 1: cumulative frequency of finding an optimal entry versus number of entries scanned. Most of the benefit is achieved with small tables, and then optimality increases only slowly with table size.

parameter α by which we inflate the time required for solution of the full MPC problem. Specifically, if it is neccessary to perform Step 4 at time k, we assume that the solution will become available for insertion in the table at time k_{avail} defined by

$$k_{\text{avail}} = k + \left\lceil \alpha \frac{T_{QP}(k)}{T_s} \right\rceil,$$

where $T_{QP}(k)$ is the time required to solve the full QP for this MPC and T_s is the sampling time. Figure 5 presents the results of this study for Example #1, in the case in which the table contains 25 entries. Note that, even for the small number of table entries, the suboptimality index S_I remains close to its optimal value of 0 for α up to 10^2 and within a factor of one at $\alpha = 10^3$. For $\alpha = 10^4$, the performance is as poor as the short-horizon backup strategy.

4.2 Example # 2: Crude Distillation Unit

The last example is a crude distillation unit model with $m = 32$ inputs, $p = 90$ outputs and $n = 252$ states. Inputs and outputs are normalized, and the inputs must satisfy the constraints: $u_{\min} \leq u_k \leq u_{\max}$ with $u_{\max} - u_{\min} = 2 \cdot \mathbf{1}_m$. Only four outputs, representing the quality of the crude distillation unit side products, have desired setpoints, and the MPC regulator is designed with the following parameters: $N = 25$, $Q = C'C$, $R = 20I_m$, $M = 0$. The number of possible different active sets is $3^{800} = 4.977 \times 10^{381}$. Response of the "true" plant is simulated by adding, to the nominal model response, unmeasured random step disturbances on the crude composition, on the fuel gas quality and on the steam header pressure, and normally distributed output noise.

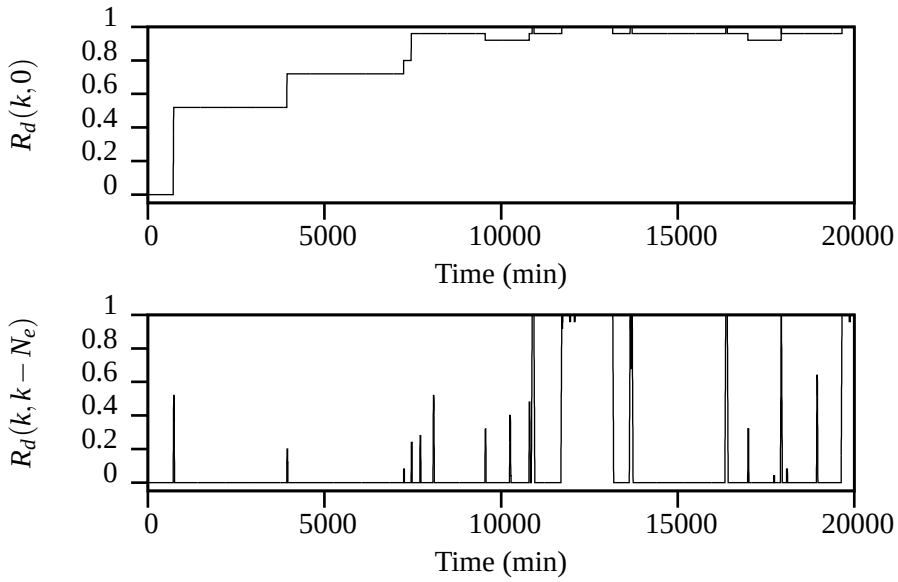

Figure 4: Example #1: table relative differences for PE25. The top figure shows that the original table is completely replaced as plant operation evolves. The bottom figure shows that the table turns over quickly during some periods and remains fairly constant during other periods.

Table 2: Example #2 performance indices.

	O_R	S_I	A_{SF}	W_{SF}
PE1	0.654	$6.9 \cdot 10^{-5}$	610	399
PE25	0.752	$6.8 \cdot 10^{-5}$	220	39.2
PE50	0.762	$6.8 \cdot 10^{-5}$	162	20.1
PE100	0.770	$6.8 \cdot 10^{-5}$	121	10.1
PE200	0.786	$6.8 \cdot 10^{-5}$	83.3	4.79

Four partial enumeration solvers are compared: PE1, PE25, PE50, PE100, PE200 use an active-set table of 1, 20, 50, 100 and 200 entries, respectively. The different solvers are compared on a 5 day (7201 decision timepoints) simulation with 10 random disturbances and 5 setpoint changes. In Table 2 we report the performance indices obtained with each solver. In Table 2 we see that partial enumeration solvers provide low suboptimality with remarkable speed factors, especially when small tables are used. This large-scale example is particularly challenging since on average `qpsol` takes about 26 seconds to compute the optimal input with a worst-case of about 53 seconds. In comparison, PE25 requires in the worst-case about 1.3 seconds, and is therefore definitely applicable for real-time implementation with sampling time of 1 minute. Active constraints are the rule in this example. On average, about 9 inputs (out of 32) are saturated at their (lower or upper) bound at each sample. The fully unconstrained solution (LQ regulator) is not feasible at *any* sample time in the simulation. The crude distillation unit operation under MPC control is always on the boundary of the feasible region, which we expect to be typical of large-scale, multivariable plants.

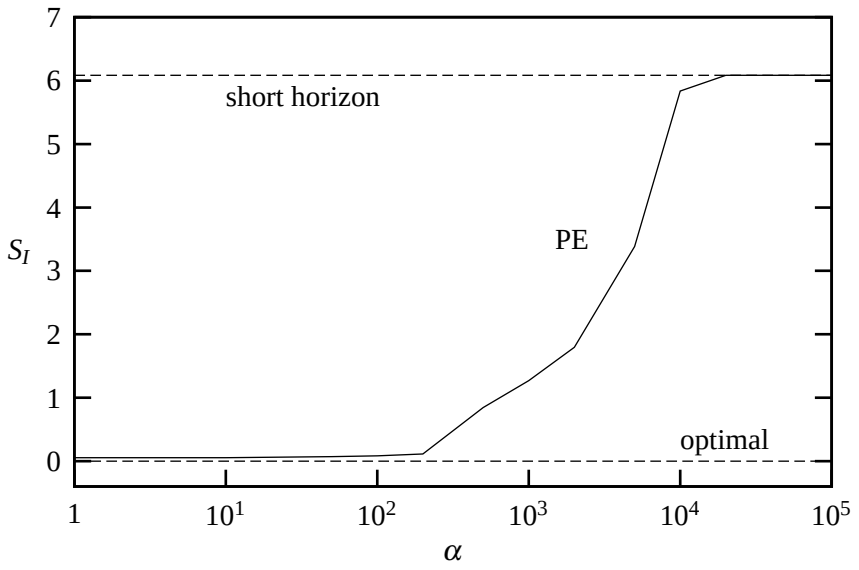

Figure 5: Example #1: suboptimality versus computational delay factor. For a wide range of the computational delay, partial enumeration provides almost optimal performance.

5 Conclusions

Partial enumeration (PE) has been presented as a method for addressing large-scale, linear MPC applications that cannot be treated with current MPC methods. Depending on the problem size and sampling time, online QP methods may be too slow, and offline feedback storage methods cannot even store the full feedback control law. The PE method was shown to have reasonable nominal theoretical properties, such as closed-loop stability. But the important feature of the method is its ability to handle large applications. The industrial distillation large example presented demonstrates suboptimality of less than 0.01%, with average speedup factors in the range of 83–220, and worst case speedups in the range of 4.8–39.2. Small tables with only 25–200 entries were used to obtain this performance. These small tables contain the optimal solution at least 75% of the time given what we think is an industrially relevant set of disturbances, noise, and setpoint changes.

Many extensions of these basic ideas are possible. If one has access to several processors, the storage table can be larger and hashed so that different CPUs search different parts of the table. Several CPUs also enable one to calculate more quickly the optimal controls missing from the current table. One can envision a host of strategies for populating the initial table depending on the user's background and experience, ranging from an empty table in which all table entries come from online operation, to more exhaustive simulation of all disturbance scenarios expected in normal plant operation. In this work we presented two backup strategies for use when the optimal control is not in the current table: a restriction of the previous, optimal trajectory, or a short control horizon solution. One can readily imagine other backup strategies that may prove useful in practice.

Of course, a complete solution for large-scale problems remains a challenge. The PE method

pushes back the current boundary between what size application is and is not tractable using MPC. But the boundary remains and may always remain, and practitioners would benefit from other alternatives that work well for the largest problems they face. A major benefit of having the ability to solve large MPC problems is to enable benchmarking and evaluating other, less computationally demanding, but perhaps less optimal, proposed control alternatives.

Acknowledgments

The authors would like to acknowledge Dr. Thomas A. Badgwell of Aspen Technology, Inc. for providing us with the crude distillation unit model.

References

[1] R. A. Bartlett, L. T. Biegler, J. Backstrom, and V. Gopal. Quadratic programming algorithms for large-scale model predictive control. *Journal of Process Control*, 12(7):775–795, 2002.

[2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[3] Mark Cannon. Efficient nonlinear model predictive control algorithms. *Annual Reviews in Control*, 28(2):229–237, 2004.

[4] Mark Cannon, B. Kouvaritakis, and J. A. Rossiter. Efficient active set optimization in triple mode MPC. *IEEE Transactions on Automatic Control*, 46(8):1307–1312, 2001.

[5] L. Chisci, P. Falugi, and G. Zappa. Gain-scheduling MPC of nonlinear systems. *International Journal of Robust and Nonlinear Control*, 13(3-4):295–308, 2003.

[6] L. Chisci and G. Zappa. Fast QP algorithms for predictive control. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 5, pages 4589–4594, 1999.

[7] E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, 36:1008–1019, 1991.

[8] G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control*. Prentice Hall, 1984.

[9] John D. Hedengren and Thomas F. Edgar. In situ adaptive tabulation for real-time control. *Industrial Engineering and Chemistry Research*, 44(8):2716–2724, 2005.

[10] Y.-C. Ho. On centralized optimal control. *IEEE Transactions on Automatic Control*, 50:537–538, April 2005.

[11] T. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control*, 48(5):810–815, May 2003.

[12] B. Kouvaritakis, M. Cannon, and J. A. Rossiter. Who needs QP for linear MPC anyway. *Automatica*, 38:879–884, 2002.

[13] B. Kouvaritakis, J. A. Rossiter, and J. Schuurmans. Efficient robust predictive control. *IEEE Transactions on Automatic Control*, 45(8):1545–1549, 2000.

[14] B. Lie, M. D. Diez, and T. A. Hauge. A comparison of implementation strategies for MPC. *Modeling Identification and Control*, 26(1):39–50, 2005.

[15] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[16] David Q. Mayne and S. Rakovic. Model predictive control of constrained piecewise affine discrete-time systems. *International Journal of Robust and Nonlinear Control*, 13(3-4):261–279, 2003.

[17] R. Milman and E. J. Davison. Evaluation of a new algorithm for model predictive control based on non-feasible search directions using premature termination. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2216–2221, December 2003.

[18] R. Milman and E. J. Davison. Fast computation of the quadratic programming subproblem in model predictive control. In *Proceedings of the American Control Conference*, pages 4723–4729, June 2003.

[19] K. R. Muske and J. B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39:262–287, 1993.

[20] G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model predictive control. *AIChE Journal*, 49:426–437, 2003.

[21] G. Pannocchia, S. J. Wright, and J. B. Rawlings. Model predictive control with active steady-state constraints: Existence and computation. Technical Report 2001–04, TWMCC, Department of Chemical Engineering, University of Wisconsin-Madison, May 2001. URL: `http://www.che.wisc.edu/jbr-group/tech-reports/twmcc-2001-04.pdf`.

[22] 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.

[23] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control System Magazine*, 20:38–52, June 2000.

[24] O.J. Rojas, G. C. Goodwin, M. M. Serón, and A. Feuer. An SVD based strategy for receding horizon control of input constrained linear systems. *International Journal of Robust and Nonlinear Control*, 14(13-14):1207–1226, 2004.

[25] J. A. Rossiter, B. Kouvaritakis, and M. J. Rice. A numerically robust state-space approach to stable-predictive control strategies. *Automatica*, 34:65–73, 1998.

[26] P. O. M. Scokaert and J. B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43:1163–1169, 1998.

[27] M. M. Seron, G. C. Goodwin, and J. A. De Doná. Characterisation of receding horizon control for constrained linear systems. *Asian Journal of Control*, 5(2):271–286, June 2003.

[28] P. Tondel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39:489–497, 2003.

[29] P. Tondel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39:945–950, 2003.

A Additional definitions

The matrices/vectors $\mathbf{H} \in \mathbb{R}^{Nm\times Nm}$, $\mathbf{G}_w \in \mathbb{R}^{Nm\times n}$, $\mathbf{A} \in \mathbb{R}^{(N+1)n\times n}$, $\mathbf{B}_s \in \mathbb{R}^{Nq\times n}$, $\bar{\mathbf{b}} \in \mathbb{R}^{Nq}$ that appear in (2.8)–(2.9) are defined as follows.

$$\mathbf{H} = \mathscr{B}'\mathscr{Q}\mathscr{B} + \mathscr{R} + \mathscr{M}'\mathscr{B} + \mathscr{B}'\mathscr{M}, \quad \mathbf{G}_w = (\mathscr{B}'\mathscr{Q} + \mathscr{M}')\mathscr{A},$$

$$\mathbf{A} = \begin{bmatrix} -D & 0 & \cdots & 0 \\ 0 & -D & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -D \end{bmatrix}, \quad \mathbf{B}_s = \begin{bmatrix} D_s \\ \vdots \\ \vdots \\ D_s \end{bmatrix}, \quad \bar{\mathbf{b}} = \begin{bmatrix} -d \\ \vdots \\ \vdots \\ -d \end{bmatrix}, \quad \text{(A.1)}$$

in which

$$\mathscr{A} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \mathscr{B} = \begin{bmatrix} 0 & 0 & \cdots & & 0 \\ B & 0 & & & 0 \\ BA & B & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & 0 \\ BA^{N-1} & BA^{N-2} & \cdots & BA & B \end{bmatrix},$$

$$\mathscr{Q} = \begin{bmatrix} Q & 0 & \cdots & & 0 \\ 0 & Q & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & 0 \\ 0 & \cdots & 0 & Q & 0 \\ 0 & \cdots & \cdots & 0 & P \end{bmatrix}, \quad \mathscr{R} = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R \end{bmatrix}, \quad \mathscr{M} = \begin{bmatrix} M & 0 & \cdots & 0 \\ 0 & M & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & M \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \quad \text{(A.2)}$$