Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

Model Predictive Control Design: New Trends and Tools

Alberto Bemporad

Abstract— **Model-based design is well recognized in industry as a systematic approach to the development, evaluation, and implementation of feedback controllers. Model Predictive Control (MPC) is a particular branch of model-based design: a dynamical model of the open-loop process is explicitly used to construct an optimization problem aimed at achieving the prescribed system's performance under specified restrictions on input and output variables. The solution of the optimization problem provides the feedback control action, and can be either computed by embedding a numerical solver in the real-time control code, or pre-computed off-line and evaluated through a lookup table of linear feedback gains. This paper reviews the basic ideas of MPC design, from the traditional linear MPC setup based on quadratic programming to more advanced explicit and hybrid MPC, and highlights available software tools for the design, evaluation, code generation, and deployment of MPC controllers in real-time hardware platforms.**

I. INTRODUCTION

Model Predictive Control (MPC) is a widely spread technology in industry for control design of highly complex multivariable processes [1]–[3]. The idea behind MPC is to start with a *model* of the open-loop process that explains the dynamical relations among system's variables (command inputs, internal states, and measured outputs). Then, constraint specifications on system variables are added, such as input limitations (typically due to actuator saturation) and desired ranges where states and outputs should remain. Desired performance specifications complete the control problem setup and are expressed through different weights on tracking errors and actuator efforts (as in classical linear quadratic regulation). The rest of the MPC design is automatic. First, an optimal control problem based on the given model, constraints, and weights, is constructed and translated into an equivalent optimization problem, which depends on the initial state and reference signals. Then, at each sampling time, the optimization problem is solved by taking the current (measured or estimated) state as the initial state of the optimal control problem. For this reason the approach is said *predictive*, as in fact the optimal control problem is formulated over a time-interval that starts at the current time up to a certain time in the future. The result of the optimization is an optimal sequence of future control moves. Only the first sample of such a sequence is actually applied to the process; the remaining moves are discarded. At the next time step, a new optimal control problem based on new measurements is solved over a shifted prediction horizon. For this reason the approach is also called "receding-horizon" or "rolling-horizon" control. Such a receding-horizon mechanism represents a way of transforming an open-loop design methodology (i.e., optimal control) into a feedback one, as at every time step the input applied to the process depends on the most recent measurements.

The author is with the Department of Information Engineering, University of Siena, Italy. `http://www.dii.unisi.it/ bemporad`

Provided that the model is accurate enough and the performance index and constraints express true performance objectives, MPC provides near-optimal performance. One must be warned, however, that there exists a tradeoff between model accuracy and complexity of the optimization: the simpler the model (and performance index/constraints), the easier is solving the optimization. Henceforth, while in building *simulation models* one looks for the most accurate model to numerically reproduce the behavior of the process as faithfully as possible, *prediction models* used in MPC (as well as in any other model-based control design techniques, from pole-placement to Bode diagrams, etc.) are usually very simple, yet representative enough to capture the main dynamical relations. System identification is an excellent tool to obtain the most representative prediction model within a prescribed bound of model complexity [4].

Summing up, MPC directly embodies technical specifications (abstract model, performance, limits) into the control algorithm, and in particular no a-posteriori patches are required to take into account limitations on system's variables. In this respect, MPC is a systematic design flow, being independent of the chosen model and performance/constraint specifications. Engineering insight focuses on the evaluation of the closed-loop performance, where the simulation model in feedback with the designed MPC controller is tested for different set-point scenarios (e.g., in Simulink), and is fundamental for tuning weights, variable limits, and prediction models until satisfactory simulation results are obtained.

The design flow terminates with the deployment of the controller. MPC design tools, like the ones reviewed in this paper, generate C-code that can immediately (and automatically) embedded in control hardware platforms.

The paper is organized as follows: the basics of MPC based on linear models and quadratic programming are reviewed first in Section II. Section III and IV summarize the main philosophy behind *explicit* MPC and MPC based on *hybrid* models, respectively. The paper also presents MATLAB tools supporting all the reviewed MPC techniques, namely the Model Predictive Control Toolbox for MATLAB [5], and the Hybrid Toolbox for MATLAB [6].

II. LINEAR MPC ALGORITHM

Several formulations and variations of MPC algorithms have been proposed in the literature and have been implemented in various tools. The simplest MPC algorithm is based on the linear discrete-time prediction model

$$x(t+1) = Ax(t) + Bu(t) \qquad (1)$$

of the open-loop process, where $x(t) \in \mathbb{R}^n$ is the state vector at time t, and $u(t) \in \mathbb{R}^m$ is the vector of manipulated variables to be determined by the controller, and on the solution of the finite-time optimal control problem

$$\min_U \quad x_N' P x_N + \sum_{k=0}^{N-1} [x_k' Q x_k + u_k' R u_k] \tag{2a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1 \tag{2b}$$

$$x_0 = x(t) \tag{2c}$$

$$u_{\min} \le u_k \le u_{\max},\ k = 0, \ldots, N-1 \tag{2d}$$

$$y_{\min} \le Cx_k \le y_{\max},\ k = 1, \ldots, N \tag{2e}$$

where N is the prediction horizon, $U \triangleq [\, u_0' \; \ldots \; u_{N-1}' \,]' \in \mathbb{R}^{Nm}$ is the sequence of manipulated variables to be optimized, $Q = Q' \ge 0$, $R = R' > 0$, and $P = P' \ge 0$ are weight matrices of appropriate dimensions defining the performance index, $u_{\min}, u_{\max} \in \mathbb{R}^m$, $y_{\min}, y_{\max} \in \mathbb{R}^p$, $C \in \mathbb{R}^{p \times n}$ define constraints on input and state variables, respectively, and "$\le$" denotes component-wise inequalities. By substituting $x_k = A^k x(t) + \sum_{j=0}^{k-1} A^i B u_{k-1-i}$, Eq. (2) can be recast as the Quadratic Programming (QP) problem

$$U^*(x(t)) \triangleq \arg\min_U \quad \frac{1}{2} U' H U + x'(t) C' U + \frac{1}{2} x'(t) Y x(t) \tag{3a}$$

$$\text{s.t.} \quad GU \le W + Sx(t) \tag{3b}$$

where $U^*(x(t)) = [\, u_0'^*(x(t)) \; \ldots \; u_{N-1}'^*(x(t)) \,]'$ is the optimal solution, $H = H' > 0$ and C, Y, G, W, S are matrices of appropriate dimensions [5]–[7]. Note that Y is not needed to compute $U^*(x(t))$, it only affects the optimal value of (3a).

The MPC control algorithm is based on the following iterations: at time t, measure or estimate the current state $x(t)$, solve the QP problem (3) to get the optimal sequence of future input moves $U^*(x(t))$, apply

$$u(t) = u_0^*(x(t)) \tag{4}$$

to the process, discard the remaining optimal moves, repeat the procedure again at time $t+1$.

In the absence of constraints (2d)–(2e), for $N \to \infty$ (or, equivalently, for $N < \infty$ and by choosing P as the solution of the algebraic Riccati equation associated with matrices (A, B) and weights (Q, R)), the MPC control law (3)–(4) coincides with the Linear Quadratic Regulator (LQR) [7]. From a design viewpoint, the MPC setup (2) can be therefore thought as a way of bringing the LQR methodology to systems with constraints.

The basic MPC setup (2) can be extended in many ways:

Tracking. Usually one has to make a certain output vector $y(t) = Cx(t) \in \mathbb{R}^p$ track a reference signal $r(t) \in \mathbb{R}^p$ under constraints (2d)–(2e). In order to do so, the cost function (2a) is replaced by

$$\sum_{k=0}^{N-1} (y_k - r(t)) Q_y (y_k - r(t)) + \Delta u_k' R \Delta u_k \tag{5}$$

where $Q_y = Q_y' \ge 0 \in \mathbb{R}^{p \times p}$ is a matrix of output weights, and the increments of command variables $\Delta u(t) \triangleq u(t) - u(t-1)$ are the new optimization variables, possibly further constrained by $\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}$. In the above tracking setup vector $[x'(t) \; r'(t) \; u'(t-1)]'$ replaces $x(t)$ in (3b) and the control law (4) becomes $u(t) = u(t-1) + \Delta u_0^*(x(t), r(t), u(t-1))$.

Prediction horizons. In order to limit the complexity of the QP problem (3), rather then a single horizon N several different horizons can be used in (2): an output horizon N_y in (2a), an input horizon $N_u \le N_y$ in (2d) with $u_k = 0$ for $k \ge N_u$, and a constraint horizon $N_{cy} \le N_y$ in (2e). In particular, the shorter the horizon N_u, the smaller is the number of optimization variables mN_u to be optimized in (3); moreover, the shorter the horizon N_{cy}, the smaller is the number q of constraints in (3b).

Soft constraints. To prevent infeasibility of the QP problem (and hence to avoid that the MPC controller halts for some values of $x(t)$) output constraints are usually treated as "soft constraints"

$$y_{\min} - \epsilon V_{\min} \le Cx_k \le y_{\max} + \epsilon V_{\max} \tag{6}$$

where the "panic" variable $\epsilon \ge 0$ is introduced to allow constraint violations, and $V_{\min}$, $V_{\max}$ are vectors of $\mathbb{R}^p$ with nonnegative entries (the larger the entry, the relatively softer is the corresponding constraint). The term $\rho\epsilon^2$ is added to penalize ϵ in the cost function, where ρ is usually some order of magnitude larger than input and output weights. Problem (2) with (6) replacing (2e) still maps to a QP problem of the form (3), where the optimization vector U includes now ϵ as the $(mN_u + 1)$th entry.

Rejection of measured disturbances. In order to take into account measured disturbances $v(t) \in \mathbb{R}^{n_v}$ entering the system, $x(t+1) = Ax(t) + Bu(t) + B_v v(t)$, one can simply change (2b), (2e) to

$$\begin{array}{l} x_{k+1} = Ax_k + Bu_k + B_v v(t) \\ y_{\min} \le Cx_k + D_v v(t) \le y_{\max} \end{array} \tag{7}$$

where $v(t)$ is the most recent available measurement of the disturbance entering the process, and is supposed constant over the prediction horizon. A more general alternative way to change any control design approach based on linear models to include information about measured disturbances is to change model (2b) to

$$\begin{bmatrix} x_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_v \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ v_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k, \tag{8}$$

and (2e) to $y_{\min} \le Cx_k + D_v v_k \le y_{\max}$, and feed the extended signal $[x'(t) \; v'(t)]$ back in (3)–(4).

Unmeasured disturbances. Unmeasured disturbances $d(t) \in \mathbb{R}^{n_d}$ entering the system, $x(t+1) = Ax(t) + Bu(t) + B_d d(t)$, $y(t) = Cx(t) + D_d d(t)$ can be taken into account by treating the signal $d(t)$ as the output of a linear system with (unmeasured) state $x_d(t)$ driven by white noise $n(t)$ having zero mean and unit covariance [5]. Then, the extended state $[x'(t) \; x_d'(t)]$ is estimated from available plant measurements through a linear observer (e.g., a Kalman filter). In particular, when $x_d(t+1) = x_d(t) + n(t)$ and $d(t) = x_d(t)$, $B_d = 0$, $D_d = I$, we talk about "output integrators", that are often introduced to ensure zero tracking error in steady state when $r(t)$ and $d(t)$ are constant [8].

Preview. One of the capabilities of MPC is to explicitly take into account future samples of the reference vector $r(t)$ and/or measured disturbance $v(t)$ in (2), which automatically results in a feedforward (non-causal) control action. One way of doing this is to replace $r(t)$ with $r(t+k)$ in (5), and similarly replace $v(t)$ with $v(t+k)$ in (7). An alternative and more general way to change a model-based control design method to anticipate future reference samples $r(t+1)$, …, $r(t+M-1)$, $M \leq N$ is to augment the prediction model (2b) with the following linear model

$$\left\{\begin{array}{rcl} x_{k+1}^{M-1} & = & x_k^{M-2} \\ x_{k+1}^{M-2} & = & x_k^{M-3} \\ & \vdots & \\ x_{k+1}^{1} & = & x_k^{0} \\ x_{k+1}^{0} & = & x_k^{0} \\ r(t+k) & = & x_k^{M-1},\ k=0,\dots,N-1 \end{array}\right.$$

with initial condition

$$\left\{\begin{array}{rcl} x_0^{M-1} & = & r(t) \\ x_0^{M-2} & = & r(t+1) \\ \vdots & = & \vdots \\ x_0^{1} & = & r(t+M-2) \\ x_0^{0} & = & r(t+M-1) \end{array}\right.$$

and to treat $x^{M-1}(t) = r(t)$, $x^{M-2}(t) = r(t+1)$, …, $x^0(t) = r(t+M-1)$ as additional state variables. The same technique applies to the preview of measured disturbance signals $v(t)$.

Delays. Transport delays on input and output channels can also be naturally embedded in the MPC setup, for instance by mapping delays into additional states corresponding to poles in $z = 0$.

Finally, we remark that the above features of linear MPC can be combined together arbitrarily.

A. Tools for Design, Evaluation and Deployment

The standard way of computing the linear MPC control action, which is implemented in most commercial MPC packages, is to solve the QP problem (3) on line at each time t.

Commercial software tools can be separated into two categories: (1) tools with a proprietary real-time industrial control system (e.g., DMCplus by Aspen Technology, Inc. and RMPCT by Honeywell, Inc.); (2) tools intended primarily for analysis and prototyping. An example of the latter is the MPC Toolbox for MATLAB (The Mathworks, Inc.) [5], which appeared in 1995, and currently has over thousands of licensees worldwide. While the original (v1.0) MPC Toolbox was command-driven, difficult to use in conjunction with nonlinear process simulations, and could not be applied to an experimental system, since the release of version 2.0 the MPC Toolbox is a very convenient environment to design, simulate and rapidly prototype MPC controllers.

The MPC Toolbox allows one to program and manipulate MPC controllers as MATLAB objects through a variety of methods and functions for simulation, analysis, and tuning. Linear MPC controllers can be therefore embedded in arbitrarily complex MATLAB programs, with maximum versatility.

The MPC Toolbox includes all the main features of linear MPC described above and others. A Graphical User Interface provides an environment for easily designing, simulating, and calibrating MPC controllers, which can be later exported to MATLAB's workspace as MPC objects. A Simulink library allows the use of MPC objects in simulation models, therefore providing a large versatility in simulating the effects of MPC in complex scenarios. The MPC Toolbox also supports the creation of MPC objects based on prediction models obtained through the Systems Identification Toolbox [4], as well as on models obtained via the new automatic linearization tools of Simulink.

The MPC Simulink block is based on an S-function written in C, which can be dealt with by Real-Time Workshop to deploy the controller from Simulink to a target system for implementation, or to xPC-Target for prototyping. Alternatively, the MPC controller can be deployed using the OPC Toolbox, by connecting a controller operating MATLAB directly to an OPC-compliant system, see [9] for a detailed example.

III. EXPLICIT MPC

One of the drawbacks of the MPC law (4) is the need to solve the QP problem (3) on line, which has traditionally labeled MPC as a technology for slow processes. Advances in microcontroller and computer technology are progressively changing the concept of "slow", but still solving a QP prevents the application of MPC in many contexts. Computation speed is not the only limitation: the QP code might cause concerns due to software certification issues, a problem which is particularly acute in safety critical applications.

To get rid of on-line QP, an alternative approach to evaluate the MPC law (4) was proposed in [7]. Rather then solving the QP problem (3) on line for the current vector $x(t)$, the idea is to solve (3) *off line* for all vectors x within a given range and make the dependence of u on x *explicit* (rather than implicitly defined by the optimization procedure (3)). The key idea is to treat (3) as a *multi-parametric* quadratic programming problem, where $x(t)$ is the vector of parameters. It turns out that the optimizer $U^* : \mathbb{R}^n \mapsto \mathbb{R}^{mN_u}$ is a piecewise affine and continuous function, and consequently the MPC controller defined by (4) can be represented explicitly as

$$u(x) = \left\{\begin{array}{ccc} F_1 x + g_1 & \text{if} & H_1 x \leq k_1 \\ \vdots & & \vdots \\ F_M x + g_M & \text{if} & H_M x \leq k_M. \end{array}\right. \quad (9)$$

It turns also out that the set of states X^* for which problem (3) admits a solution is a polyhedron, and that the optimum value in (3) is a piecewise quadratic, convex, and continuous function of $x(t)$. The controller structure (9) is simply a look-up table of linear gains (F_i, g_i), where the ith gain is selected according to the set of linear inequalities $H_i x \leq K_i$ that the state vector satisfies. Hence, the evaluation of the MPC controller (4), once put in the form (9), can be carried out by a very simple piece of control code. In the most naive implementation, the number of operations depends linearly in the worst case on the number M of partitions, or even logarithmically if the partitions are properly stored [10]. Note also that (9) can be parallelized quite easily in case multiple arithmetic logic units are available on chip. Experiments on the implementation of explicit MPC on field programmable gate arrays (FPGA) and application specific integrated circuits (ASIC) with sampling times around 1 μs have been recently reported in [11].

While the computer code for evaluating MPC in the explicit form (9) is certainly simpler than the code embedding the QP solver, from the point of view of memory requirements the explicit form may be more demanding, as M may get large. As shown in Table I[1], M depends mainly (and, unfortunately, exponentially in the worst case) on the number q of constraints in problem (3), and only mildly on the number n of states (but clearly the number of coefficients in F_i, H_i depends linearly on n). It also depends on the number mN_u of optimization variables, mainly because q depends on mN_u in case of input constraints.

states\horizon	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 5$
n=2	3	6.7	13.5	21.4	19.3
n=3	3	6.9	17	37.3	77
n=4	3	7	21.65	56	114.2
n=5	3	7	22	61.5	132.7
n=6	3	7	23.1	71.2	196.3
n=7	3	6.95	23.2	71.4	182.3
n=8	3	7	23	70.2	207.9

TABLE I
NUMBER OF FEEDBACK GAINS IN EXPLICIT MPC

Whether the explicit form is preferable to the QP-based one depends on available CPU time, data memory, and program memory. As a pure exemplification, Table II compares the CPU time required by on-line QP (using the compiled C solver of the MPC Toolbox) and by the evaluation of the piecewise affine explicit form (using the compiled C code from the Hybrid Toolbox) on the two-input/two-output example reported in [7, Example 7.2]. The table reports the average time on 100 random states, references, and previous inputs, and, in brackets, the worst-case time[2].

A. Tools for Design, Evaluation and Deployment

The Hybrid Toolbox for MATLAB [6] allows one to design explicit MPC control laws. The toolbox can be freely downloaded from `http://www.dii.unisi.it/hybrid/toolbox`. It contains among other things various functions for the design, simulation and code generation of MPC controllers in explicit form. In particular, MPC objects developed through the MPC Toolbox can be converted to

$2N_u$	QP (ms)	explicit (ms)	regions	[storage kb]
4	1.1 [1.5]	0.005 [0.1]	25	16
8	1.3 [1.9]	0.023 [1.1]	175	78
20	2.5 [2.6]	0.038 [3.3]	1767	811
30	5.3 [7.2]	0.069 [4.4]	5162	2465
40	10.9 [13.0]	0.239 [15.6]	11519	5598

TABLE II
CPU TIME: ON-LINE VS. OFF-LINE OPTIMIZATION

explicit form through a multiparametric quadratic programming solver based on the algorithm described in [12]. The Hybrid Toolbox also provides functions for manipulation and visualization of polyhedral objects and polyhedral partitions, and contains Simulink blocks to simulate (and possibly rapid prototype) explicit MPC controllers. The toolbox provides the source C code for evaluating the piecewise affine map (9), whose coefficients are automatically embedded in a header file by appropriate methods. Several demos are available in the Hybrid Toolbox distribution. Similar functionalities have been introduced more recently also in the Multi Parametric Toolbox [13].

IV. HYBRID MPC

So far we have considered MPC schemes based on linear prediction models. While several formulations of MPC based on general smooth nonlinear prediction models (as well as on uncertain linear models) exist, most of them rely on nonlinear optimization methods for generic nonlinear functions/constraints to compute the control actions, and are therefore very seldom deployed in practical applications.

Recently, MPC based on *hybrid* dynamical models has emerged as a very promising approach to handle switching linear dynamics, on/off inputs, logic states, as well as logic constraints on input and state variables [14]. Hybrid dynamics are often so complex that a satisfactory feedback controller cannot be synthesized by using analytical tools, and heuristic design procedures usually require trial and error sessions, extensive testing, are time consuming, costly and often inadequate to deal with the complexity of the hybrid control problem properly.

As for the linear MPC case, hybrid MPC design is a systematic approach to meet performance and constraint specifications in spite of the aforementioned switching among different linear dynamics, logical state transitions, and more complex logical constraints on system's variables.

The approach consists of modeling the switching open-loop process and constraints using the language HYSDEL [15] and then automatically transform the model into the set of linear equalities and inequalities

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \quad (10a)$$
$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \quad (10b)$$
$$E_2\delta(k) + E_3z(k) \leq E_1u(k) + E_4x(k) + E_5, \quad (10c)$$

involving both real and binary variables, denoted as the Mixed Logical Dynamical (MLD) model, where $x(k) =$

[1]Data are averaged over 20 randomly generated single-input single-output systems subject to input saturation ($q = 2N$ constraints in (3b))

[2]Experiment performed on an laptop with Intel Centrino 1.4 GHz processor.

$\left[\begin{smallmatrix} x_c(k) \\ x_\ell(k) \end{smallmatrix}\right]$ is the state vector, $x_c(k) \in \mathbb{R}^{n_c}$ and $x_\ell(k) \in \{0,1\}^{n_\ell}$, $y(k) = \left[\begin{smallmatrix} y_c(k) \\ y_\ell(k) \end{smallmatrix}\right] \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$ is the output vector, $u(k) = \left[\begin{smallmatrix} u_c(k) \\ u_\ell(k) \end{smallmatrix}\right] \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ is the input vector, $z(k) \in \mathbb{R}^{r_c}$ and $\delta(k) \in \{0,1\}^{r_\ell}$ are auxiliary variables, A, B_i, C, D_i and E_i denote real constant matrices, E_5 is a real vector, $n_c > 0$, and p_c, m_c, r_c, n_ℓ, p_ℓ, m_ℓ, $r_\ell \geq 0$. Inequalities (10c) must be interpreted componentwise.

The associated finite-horizon optimal control problem based on quadratic costs takes the form (3) with $U = \left[u_0' \cdots u_{N-1}' \; \delta_0' \cdots \delta_{N-1}' \; z_0' \cdots z_{N-1}' \right]'$ and subject to the further restriction that some of the components of U must be either 0 or 1. The problem is therefore a Mixed-Integer Quadratic Programming (MIQP) problem, for which both commercial [16] and public domain [17] solvers are available. When infinity norms $\|Qx_k\|_\infty$, $\|Ru_k\|_\infty$, $\|Px_k\|_\infty$ are used in (2a) in place of quadratic costs, the optimization problem becomes a Mixed-Integer Linear Programming (MILP) problem [6], which can be also handled by more efficient public domain solvers such as [18], as well as by commercial solvers [16].

The technique has been tested in several application problems of industrial interest, for instance in the ABB application [19].

Regarding complexity, unfortunately MIP's are $\mathcal{NP}$-complete problems. However, the state of the art in solving MIP problems is growing constantly, and problems of relatively large size can be solved quite efficiently, see e.g. the benchmarks reported at `http://plato.asu.edu/ftp/milpf.html` and `http://plato.asu.edu/ftp/miqp.html`. While MIP problems can always be solved to the global optimum, closed-loop stability properties can be guaranteed as long as the optimum value in (3) decreases at each time step. Usually MIP solvers provide good feasible solutions within a relatively short time compared to the total time required to find and certify the global optimum.

Extensions of the hybrid MPC formulation introduced above have been recently proposed for stochastic hybrid systems [20], and for event-based continuous-time hybrid systems [21].

A. Explicit Hybrid MPC

An alternative way to solving MIP problems on line is to extend explicit MPC ideas to the hybrid case.

For hybrid MPC problems based on infinity norms, [22] showed that an equivalent piecewise affine explicit reformulation – possibly discontinuous, due to binary variables – can be obtained through off-line multiparametric mixed-integer linear programming techniques.

The use of linear norms has some practical disadvantages, due to the fact that typically good performance can only be achieved with long time horizons, compared to the horizons requested by quadratic costs. Thanks to the possibility of converting hybrid models (such as those designed through HYSDEL) to an equivalent piecewise affine (PWA) form [23], an explicit hybrid MPC approach dealing with quadratic costs was proposed in [24], based on dynamic programming (DP) iterations. Multiparametric quadratic programs (mpQP) are solved at each iteration, and quadratic value functions are compared to possibly eliminate regions that are proved to never be optimal. A different approach still exploiting the PWA structure of the hybrid model was proposed in [25], where all possible switching sequences are enumerated, an mpQP is solved for each sequence, and quadratic costs are compared on-line to determine the optimal input (in this respect, one could define the approach semi-explicit). To overcome the problem of enumerating all switching sequences and storing all the corresponding mpQP solutions, backwards reachability analysis is exploited in [26] (and implemented in the Hybrid Toolbox). A procedure to post-process the mpQP solutions and eliminate all polyhedra (and their associated control gains) that never provide the lowest cost was suggested in [26]. Typically the DP approach provides simpler explicit solutions when long horizons N are chosen, but on the contrary tends to subdivide the state-space in a larger number of polyhedra than the enumeration approach for short horizons.

B. Tools for Design, Evaluation and Deployment

The Hybrid Toolbox for MATLAB provides a nice development environment for hybrid MPC. Hybrid dynamical systems described in HYSDEL are automatically converted to MATLAB MLD and PWA objects. MLD and PWA objects can be validated in open-loop simulation, either from the command line or through their corresponding Simulink blocks. Hybrid MPC controllers based on MILP/MIQP optimization can be designed and simulated, either from the command line or in Simulink, and can be converted to their explicit form for deployment. Several demos are available in the Hybrid Toolbox distribution.

V. APPLICATIONS

As reported in [27], MPC based on QP was applied in thousands of industrial applications, especially in the process industries. Several applications of explicit (hybrid) MPC have been proposed recently, especially in the automotive domain (see e.g. [28]–[32])

VI. CONCLUSIONS

MPC is a model-based design approach to the development, evaluation, and deployment of embedded feedback controllers. It heavily exploits the mathematical model of the regulated process to optimize system's performance in the presence of limitations on input and output variables.

This paper has reviewed current and new trends in MPC setup (linear vs. hybrid) and deployment (on-line vs. off-line computations), with the vision of enlarging the application domain of MPC from traditional slow (yet highly multivariable) industrial process control, to new areas such as automobiles, avionics, mechatronics, etc., as well as to domains where on-line algorithms to coordinate interconnected process units at the higher scheduling/management level are requested.

Both implementations based on on-line optimization and on evaluation of explicit forms have pros and cons. Explicit MPC is limited to relatively small problems (typically one/two inputs, up to eight states, up to five/six free control moves) but allows one to reach very high sampling frequencies (up to 1 MHz) and requires a very simple control code to be embedded in the system. On-line QP can tackle much larger control problems (say 400 outputs, 200 inputs), but requires a more complex control code and in general more CPU time per sampling step.

Hybrid MPC control can deal with very complex specifications in terms of models and constraints, but require a higher CPU time and a much more complex control code for on-line optimization. Explicit versions of hybrid MPC are possible, but still limited to small systems with few binary variables. Linear MPC handles simpler models and constraints, but requires a lighter CPU load than hybrid MPC. When a single linear model is not satisfactory enough, multiple linear models (e.g., obtained by linearizing a given nonlinear dynamics at different operating points) often provide an excellent alternative, at least when process dynamics changes smoothly.

The MATLAB toolboxes presented in this paper provide an easy environment for the control engineer to develop MPC controllers, from their setup, tuning and validation up to their rapid prototyping, covering both linear, hybrid and explicit. The Model Predictive Control Toolbox is a Mathworks' product, very well supported, user friendly and versatile at the same time, and allows the design and simulation of linear MPC controllers based on QP, as well as code generation. The Hybrid Toolbox, currently more research oriented and freely available, extends the capabilities to hybrid and explicit MPC.

VII. ACKNOWLEDGEMENTS

The author thanks Dr. Davor Hrovat for providing useful comments on this paper.

REFERENCES

[1] J. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems Magazine*, pp. 38–52, June 2000.

[2] E. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed., ser. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag, 2004.

[3] J. Maciejowski, *Predictive Control with Constraints*. Harlow, UK: Prentice Hall, 2002.

[4] L. Ljung, *System Identification Toolbox for Matlab – User's Guide*. The Mathworks, Inc., http://www.mathworks.com/access/helpdesk/help/toolbox/ident/.

[5] A. Bemporad, M. Morari, and N. L. Ricker, *Model Predictive Control Toolbox for Matlab – User's Guide*. The Mathworks, Inc., 2004, http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[6] A. Bemporad, *Hybrid Toolbox – User's Guide*, Jan. 2004, http://www.dii.unisi.it/hybrid/toolbox.

[7] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[8] G. Pannocchia and J. B. Rawlings, "Disturbance models for offset-free MPC control," *AIChE J.*, vol. 49, no. 2, pp. 426–437, Feb. 2003.

[9] A. Bemporad, N. Ricker, and J. Owen, "Model predictive control – New tools for design and evaluation," in *American Control Conference*, Boston, MA, 2004, pp. 5622–5627.

[10] P. Tøndel, T. A. Johansen, and A. Bemporad, "Evaluation of piecewise affine control via binary search tree," *Automatica*, vol. 39, no. 5, pp. 945–950, May 2003.

[11] T. A. Johansen, W. Jackson, R. Schrieber, and P. Tøndel, "Hardware architecture design for explicit model predictive control," in *American Control Conference*, Minneapolis, MN, 2006.

[12] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, Mar. 2003.

[13] M. Kvasnica, P. Grieder, and M. Baotić, *Multi Parametric Toolbox (MPT)*, 2006. [Online]. Available: http://control.ee.ethz.ch/~mpt/

[14] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[15] F. Torrisi and A. Bemporad, "HYSDEL — A tool for generating computational hybrid models," *IEEE Trans. Contr. Systems Technology*, vol. 12, no. 2, pp. 235–249, Mar. 2004.

[16] ILOG, Inc., *CPLEX 9.0 User Manual*, Gentilly Cedex, France, 2004.

[17] A. Bemporad and D. Mignone, *MIQP.M: A Matlab function for solving mixed integer quadratic programs*, 2000, http://www.dii.unisi.it/~hybrid/tools/miqp.

[18] A. Makhorin, *GLPK (GNU Linear Programming Kit) User's Guide*, 2004. [Online]. Available: http://www.gnu.org/software/glpk/glpk.html

[19] D. Castagnoli, E. Gallestey, and C. Frei, "Cement mills optimal (re)scheduling via MPC and MLD systems," in *IFAC Conf. on Analysis and Design of Hybrid Systems*, Saint Malo, France, June 2003.

[20] A. Bemporad and S. D. Cairano, "Optimal control of discrete hybrid stochastic automata," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, M. Morari and L. Thiele, Eds. Springer-Verlag, 2005, no. 3414, pp. 151–167.

[21] A. Bemporad, S. D. Cairano, and J. Júlvez, "Event-based model predictive control and verification of integral continuous-time hybrid automata," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, J. Hespanha and A. Tiwari, Eds. Springer-Verlag, 2006, pp. 93–107.

[22] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *American Control Conference*, Chicago, IL, June 2000, pp. 1190–1194.

[23] A. Bemporad, "Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form," *IEEE Trans. Automatic Control*, vol. 49, no. 5, pp. 832–838, 2004.

[24] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, Oct. 2005.

[25] D. Mayne and S. Rakovic, "Optimal control of constrained piecewise affine discrete time systems using reverse transformation," in *Proc. 41th IEEE Conf. on Decision and Control*, Las Vegas, Nevada, USA, Dec. 2002, pp. 1546–1551.

[26] A. Alessio and A. Bemporad, "Feasible mode enumeration and cost comparison for explicit quadratic model predictive control of hybrid systems," in *2nd IFAC Conference on Analysis and Design of Hybrid Systems*, Alghero, Italy, 2006, pp. 302–308.

[27] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[28] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat, "An MPC/hybrid system approach to traction control," *IEEE Trans. Contr. Systems Technology*, vol. 14, no. 3, pp. 541–552, May 2006.

[29] N. Giorgetti, G. Ripaccioli, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Hybrid model predictive control of direct injection stratified charge engines," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 5, Aug. 2006.

[30] R. Möbus, M. Baotić, and M. Morari, "Multi-objective adaptive cruise control," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, O. Maler and A. Pnueli, Eds., no. 2623. Springer-Verlag, 2003, pp. 359–374.

[31] N. Giorgetti, A. Bemporad, H. E. Tseng, and D. Hrovat, "Hybrid model predictive control application towards optimal semi-active suspension," *International Journal of Control*, vol. 79, no. 5, pp. 521–533, 2006.

[32] P. Ortner, P. Langthaler, J. G. Ortiz, and L. del Re, "MPC for a diesel engine air path using an explicit approach for constrained systems," in *Proc. 2006 IEEE Conf. on Control Applications*, Munich, Germany, Sept. 2006.