COMPLEXITY REDUCTION IN EXPLICIT LINEAR MODEL PREDICTIVE CONTROL

Petter Tøndel * **and Tor A. Johansen** *

* *Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway.*

Abstract: Explicit piecewise linear (PWL) state feedback solutions to constrained linear model predictive control (MPC) problems can be obtained by solving multi-parametric quadratic programs (mp-QP) where the parameters are the components of the state vector. This allows MPC to be implemented via a PWL function evaluation without real-time optimization. The main drawback of this approach is dramatic increase in off-line computational complexity and number of regions in the state space partition as the number of states, inputs and constraints increases. Here we study two approaches to complexity reduction. First, we consider input trajectory parameterization. Second, we develop a search tree that allows PWL function evaluation to be implemented in real time with low computational complexity.

1. INTRODUCTION

Recently, several algorithms for computing explicit solutions to constrained linear model predictive control (MPC) problems have been reported (Bemporad *et al.* 1999, Bemporad *et al.* 2000*b*, Seron *et al.* 2000, Bemporad *et al.* 2000*a*, Tøndel *et al.* 2001, Johansen *et al.* 2000*b*, Johansen *et al.* 2000*a*). Their main motivation is that an explicit solution avoids the need for real-time optimization, and may therefore open new application areas where MPC has not traditionally been used due to the need for high sampling rates or software reliability issues.

In (Bemporad *et al.* 1999, Bemporad *et al.* 2000*b*) it was recognized that the MPC problem is a multi-parametric quadratic program (mp-QP), when the state is viewed as a parameter to the problem. They show that the solution (the control input) is a piecewise linear (PWL) function on a polyhedral partition of the state space and develop an mp-QP algorithm to compute this function. In (Tøndel *et al.* 2001) a significantly more efficient mp-QP solver is developed by inferring additional information about neighboring regions during the iterative solution. Alternative formulations and solutions based on mp-LP as well as extensions to hybrid systems using multi-parametric mixed-integer LP can be found in (Bemporad *et al.* 2000*a*).

In (Johansen *et al.* 2000*b*, Johansen *et al.* 2000*a*) a different solution approach is taken, starting with the Hamilton-Jacobi-Bellman equation for the optimal control problem. The solution strategy allows sub-optimality and complexity reduction to be introduced by pre-determining a small number of sampling instants when the active set is allowed to change on the horizon. An alternative sub-optimal approach was introduced in (Bemporad and Filippi 2001) where small slacks are introduced on the optimality conditions and the mp-QP algorithm (Bemporad *et al.* 1999, Bemporad *et al.* 2000*b*) is modified for the relaxed problem. This leads to reduced computational complexity and reduced complexity of the solution (in terms of less regions in the partition). Since in MPC we only need the first sample of the control for implementation, one may in many cases recognize several neighboring regions where the solution leads to the same locally linear control law. Whenever the union of such polyhedra remains polyhedral one may use this to reduce the number of regions required for implementing the control law (Bemporad *et al.* 2000*b*). However, the recognition of such regions is hard (Bemporad *et al.* 2001).

The present paper contains two main contributions; First we study how one of the standard complexity reduction methods from conventional MPC can be applied also in the explicit MPC case, namely the idea of input trajectory parameterization. Typically this is implemented by input blocking, i.e. pre-determining a small number of sampling instants when the control input is allowed to change. Second, it is studied how to efficiently evaluate the PWL function that defines the explicit solution. This is non-trivial since the number of regions in the partition may be large, see (Borrelli *et al.* 2001) for an alternative approach that exploits the convexity of the cost function. We develop a binary search tree to be used in the real-time implementation to determine with low worst-case computational complexity in which polyhedral region an arbitrary state belongs.

[1] Email: Tor.Arne.Johansen@itk.ntnu.no

2. LINEAR MPC WITH CONSTRAINTS

The main aspects of formulating a linear MPC problem as a multi-parametric QP will, for convenience, be repeated here. See (Bemporad *et al.* 2000*b*) for further details. Consider the linear system

$$x(t+1) = Ax(t) + Bu(t) \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state variable, $u(t) \in \mathbb{R}^m$ is the input variable, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and (A, B) is a controllable pair. For the current $x(t)$, MPC solves the optimization problem

$$\min_{U \triangleq \{u_t,...,u_{t+M-1}\}} J(U, x(t)) \tag{2}$$

such that

$$\begin{aligned} &y_{\min} \le y_{t+k|t} \le y_{\max},\ k = 1, ..., N \\ &u_{\min} \le u_{t+k} \le u_{\max},\ k = 0, 1, ..., M-1, \\ &u_{t+k} = u_{t+k-1},\ M \le k \le N-1 \\ &x_{t|t} = x(t) \\ &x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k},\ k \ge 0 \\ &y_{t+k|t} = Cx_{t+k|t},\ k \ge 0 \end{aligned} \tag{3}$$

The cost function is given by

$$\begin{aligned} J(U, x(t)) = &\sum_{k=0}^{N-1} x_{t+k|t}^T Q x_{t+k|t} + u_{t+k}^T R u_{t+k} \\ &+ x_{t+N|t}^T P x_{t+N|t} \end{aligned} \tag{4}$$

with symmetric $R > 0$, $Q \ge 0$, $P > 0$. The final cost matrix P is usually calculated from the algebraic Riccati equation with the assumption that no constraints are active for $k \ge N$. This and related problems can by some algebraic manipulation be reformulated as

$$V_z(x(t)) = \min_z \frac{1}{2} z^T H z \tag{5}$$

$$\text{subject to} \quad Gz \le W + Sx(t) \tag{6}$$

where $z \triangleq U + H^{-1}F^T x(t), U = \left[u_t^T, ..., u_{t+N-1}^T\right]^T$. The vector $x(t)$ is the current state, which can be treated as a vector of parameters. Note that $H > 0$ since $R > 0$. The number of inequalities is denoted q and the number of free variables is $n_z = m \cdot N$. Then $z \in \mathbb{R}^{n_z}$, $H \in \mathbb{R}^{n_z \times n_z}$, $G \in \mathbb{R}^{q\times n_z}$, $W \in \mathbb{R}^{q\times 1}$, $S \in \mathbb{R}^{q\times n}$. The solution of the optimization problem (5)-(6) can be found in an explicit form $z^* = z^*(x(t))$. Bemporad *et al.* (1999) showed that the solution $z^*(x(t))$ (and $U^*(x(t))$) is a continuous PWL function of $x(t)$ defined over a polyhedral partition of the parameter space, and $V_z(x(t))$ is a convex (and therefore continuous) piecewise quadratic function.

3. MULTI-PARAMETRIC QUADRATIC PROGRAMMING

As shown in (Bemporad *et al.* 1999, Bemporad *et al.* 2000*b*), the mp-QP problem (5)-(6) can be solved by applying the Karush-Kuhn-Tucker (KKT) conditions

$$Hz + G^T\lambda = 0, \lambda \in \mathbb{R}^q \tag{7}$$

$$\lambda_i \left(G^i z - W^i - S^i x\right) = 0, i = 1, ..., q \tag{8}$$

$$\lambda \ge 0 \tag{9}$$

$$Gz - W - Sx \le 0 \tag{10}$$

For ease of notation we write x instead of $x(t)$. Superscript i on some matrix denotes the i^{th} row. Since H has full rank, (7) gives

$$z = -H^{-1}G^T\lambda \tag{11}$$

Let $z^*(x)$ be the optimal solution to (5)-(6) for a given x. Let $\breve{\lambda}$ be the Lagrange multipliers of the inactive constraints, $\breve{\lambda} = 0$, and $\tilde{\lambda}$ the Lagrange multipliers of the active constraints, $\tilde{\lambda} \ge 0$. Assume for the moment that we know which constraints are active at the optimum for a given x. We can now form matrices $\tilde{G}, \tilde{W}$ and $\tilde{S}$ which contains the rows G^i, W^i and S^i corresponding to the active constraints.

Assume that $\tilde{G}$ has full row rank, such that the rows of $\tilde{G}$ are linearly independent. For the active constraints, (8) and (11) gives $-\tilde{G}H^{-1}\tilde{G}^T\tilde{\lambda} - \tilde{W} - \tilde{S}x = 0$, which leads to

$$\tilde{\lambda} = -(\tilde{G}H^{-1}\tilde{G}^T)^{-1}(\tilde{W} + \tilde{S}x). \tag{12}$$

Eq. (12) can now be substituted into (11) to obtain

$$z = H^{-1}\tilde{G}^T(\tilde{G}H^{-1}\tilde{G}^T)^{-1}(\tilde{W} + \tilde{S}x). \tag{13}$$

We have now characterized the solution to (5)-(6) for a given optimal active set, and a fixed x. However, as long as the active set remains optimal in a neighborhood of x, the solution (13) remains optimal, when z is viewed as a function of x. Next, we characterize the region where this active set remains optimal. First, z must remain feasible (10)

$$GH^{-1}\tilde{G}^T(\tilde{G}H^{-1}\tilde{G}^T)^{-1}(\tilde{W} + \tilde{S}x) \le W + Sx. \tag{14}$$

Second, the Lagrange multipliers λ must remain nonnegative (9)

$$-(\tilde{G}H^{-1}\tilde{G}^T)^{-1}(\tilde{W} + \tilde{S}x) \ge 0. \tag{15}$$

The equations (14) and (15) describe a polyhedron in the state space. This region is denoted as the **critical region** CR_0 corresponding to the given set of active constraints which Bemporad *et al.* (1999) showed that when you pick an arbitrary $x_0 \in X$ and let (z_0, λ_0) be the corresponding values satisfying the KKT conditions, then one can find the critical region CR_0 from (14) and (15). This region is a convex polyhedral set and represents the largest set of parameters x such that the combination of active constraints at the minimizer remains optimal (Bemporad *et al.* 1999).

Algorithms have been developed by (Bemporad *et al.* 2000*b*, Johansen *et al.* 2000*b*, Tøndel *et al.* 2001) for constructing polyhedral partitions of the state space that explicitly defines the PWL function $\hat{z}^*(x)$. Below, we give a simplified description of the algorithm, while a complete description and analysis that also covers degeneracy and infeasibility is found in (Tøndel *et al.* 2001):

Algorithm 1 (mp-QP)

1. Initialize the list of unexplored active sets $\mathcal{U}$ with an arbitrary active set. Initialize the list of explored active sets $\mathcal{E}$ to be empty.

2. Choose an arbitrary active set in $\mathcal{U}$, compute the associated linear state feedback (13), Lagrange multiplier (12) and polyhedral region defined by (14) and (15).

3. Remove the active set under consideration from $\mathcal{U}$ and add it to $\mathcal{E}$.

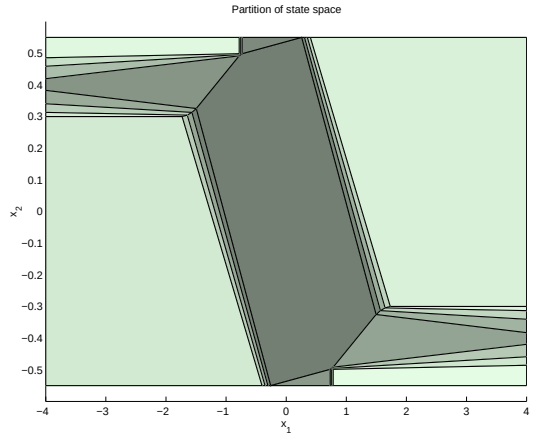

Fig. 1. Polyhedral partition of state space, $N = 4$.

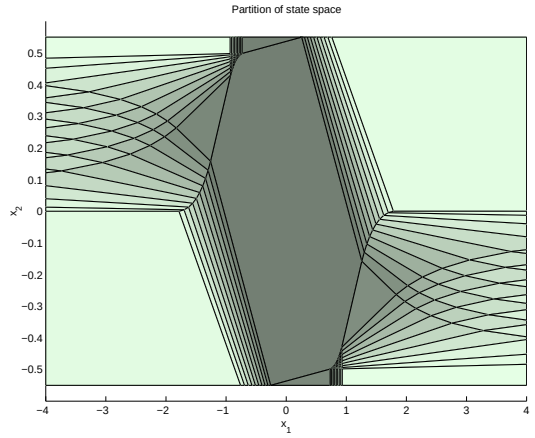

Fig. 2. Polyhedral partition of state space, $N = 10$.

4. For each facet of the corresponding polyhedral representation determine the active set in the neighboring region and determine the active set in the neighboring region. For each active set that is unexplored (i.e. not already in $\mathcal{E}$), add it to $\mathcal{U}$.

5. If $\mathcal{U}$ is non-empty, go to 2, otherwise terminate.

□

Example. Consider the double integrator (Johansen *et al.* 2000*b*)

$$A = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} T_s^2 \\ T_s \end{bmatrix}$$

where the sampling interval $T_s = 0.05$, and consider the MPC problem with cost matrices $Q = \text{diag}(1, 0)$, $R = 1$, and the matrix P given as the solution of the algebraic Riccati equation. The constraints in the system are $-0.5 \leq x_2 \leq 0.5$, $-1 \leq u \leq 1$. Figures 1-2 show the partitions for horizons $N = 4$ and $N = 10$, and Table 1 summarizes the complexity and computation times of the exact solutions for $N = 1$ to $N = 15$.

As figures 1-2 show, most of the regions in the partition are very small. This is unfortunate when considering the on-line processing time required to determine in which polyhedral critical region an arbitrary state x belongs.

□

Table 1. Number of regions and computation times (800 MHz CPU) for exact solutions.

N	Regions	CPU time(s)
1	5	0.1
2	13	0.2
3	23	0.3
4	35	0.6
5	51	1.0
6	71	1.6
7	95	2.5
8	123	3.7
9	155	5.3
10	191	7.3
11	231	9.7
12	277	13.0
13	325	17.3
14	379	21.7
15	437	27.0

4. INPUT TRAJECTORY PARAMETERIZATION

The input trajectory is defined by the n_z elements of the vector U. The input is allowed to change its value at every sampling instant. The main idea of input trajectory parameterization is to introduce a class of input trajectories with less degrees of freedom in order to reduce the dimensions of the optimization problem and thereby reducing the computational complexity. This is implemented in some form in most practical MPC algorithms. With a discrete-time formulation the most common approach is to pre-determine a number of sampling instants when the control input is not allowed to change, i.e.

$$U = T\hat{U} \tag{16}$$

where $\dim \hat{U} < \dim U$. For example, if $N = 5, m = 1$ and we require that the input is kept constant for the first two samples and also for the three last samples, we have $\hat{U} = (\hat{u}_1,\ \hat{u}_2)^T$ and

$$T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}^T$$

Hence, the five-sample trajectory is parameterized by 2 parameters. Due to the receding horizon implementation of MPC, the implemented control input can change every sample and the degree of sub-optimality can usually be kept fairly small, especially for open-loop stable plants. It is also experienced that such an input trajectory parameterization may be beneficial from a robustness point of view, i.e. the closed loop performance is less sensitive to modelling error.

In the explicit MPC formulation, the parameterization (16) leads to the free variable $\hat{z} = \hat{U} + T^T H^{-1} F^T x$, also of reduced dimension. The approach can be implemented with only trivial modification of the data input and output to the mp-QP solver. Hence, the explicit solution remain PWL and continuous as a function of the state.

Example, continued. The partition of the double integrator is now computed using parameterization of the input, with 1, 2, 3 and 4 parameters and horizon 15. The partitions are shown in Figure 3. Table 2 shows the errors in the control input, where $\mathbf{e}_{\max}$ is the maximal error in the control input compared to the exact solution, and $\mathbf{e}_{av}$ is the average error in the control input. We can see that it is important to choose the number of parameters large enough to get an acceptable result. Table 3 shows the corresponding errors for exact solutions with horizons 1-4. Parameterization of the input with N parameters has reduced the complex-

ity while introduced a small degree of sub-optimality, as expected.

Table 2. Errors using input blocking with 1-4 parameters, compared to exact solution.

Parameter	CPU time(s)	Regions	$e_{\max}$	e_{av}
1	0.1	7	0.87	0.2481
2	0.3	17	0.30	0.0108
3	0.4	25	0.18	0.0056
4	0.7	41	0.09	0.0024

Table 3. Errors in exact solutions with horizons 1-4, compared to exact solution with horizon 15.

N	$e_{\max}$	e_{av}
1	0.57	0.0215
2	0.41	0.0121
3	0.31	0.0084
4	0.23	0.0071

□

5. REAL-TIME SEARCH TREE

The real-time implementation of explicit MPC corresponds to evaluating the pre-computed PWL mapping from x to u. This amounts to first determining in which critical region where the current state x belongs, and then computing the control input using the pre-computed affine state feedback. The main problem is to minimize the number of linear inequalities to evaluate in order to determine which critical region x belongs. An efficient way to exploit the convexity of polyhedral sets is to build off-line a binary search tree (for on-line use) where at each level one linear inequality is evaluated. More precisely, consider the set of polyhedral critical regions $X_1, X_2, ..., X_N$ that form a partition of the polyhedron $X \subset \mathbb{R}^n$. Let all hyper-planes defining the polyhedra in the partition be denoted $a_j^T x = b_j$ for $j = 1, 2, ..., L$. Define $d_j(x) = a_j^T x - b_j$ and represent the polyhedron X_i through its index set $\mathcal{I}_i$ such that

$$X_i = \{x \mid d_j(x) \leq 0 \text{ for all } j \in \mathcal{I}_i\} \qquad (17)$$

The idea is to construct a balanced binary search tree such that for a given $x \in X$, at each node we will evaluate one affine function $d_j(x)$ and test its sign. Based on the sign we select the left or right sub-tree. Traversing the tree from the root to a leaf node one will pass through nodes corresponding to all indices in $\mathcal{I}_i$ for some i, and one may terminate the search with the index i of the polyhedron X_i where x belongs. The main challenge is to design a tree of minimum depth such that we minimize the number of extra nodes (with inequalities not needed in the representation of X_i) we have to pass through to determine the solution. The following algorithm will construct such a binary search-tree:

Algorithm 2 (Build search tree)

1. The root node of the tree is initialized as $N_1 := (*, \mathcal{I})$, where the first element ($*$ means uninitialized) is the index of the splitting hyperplane, and the second element is the index set of all the critical regions.

2. The set of unexplored nodes is initialized as $\mathcal{U} := \{N_1\}$.

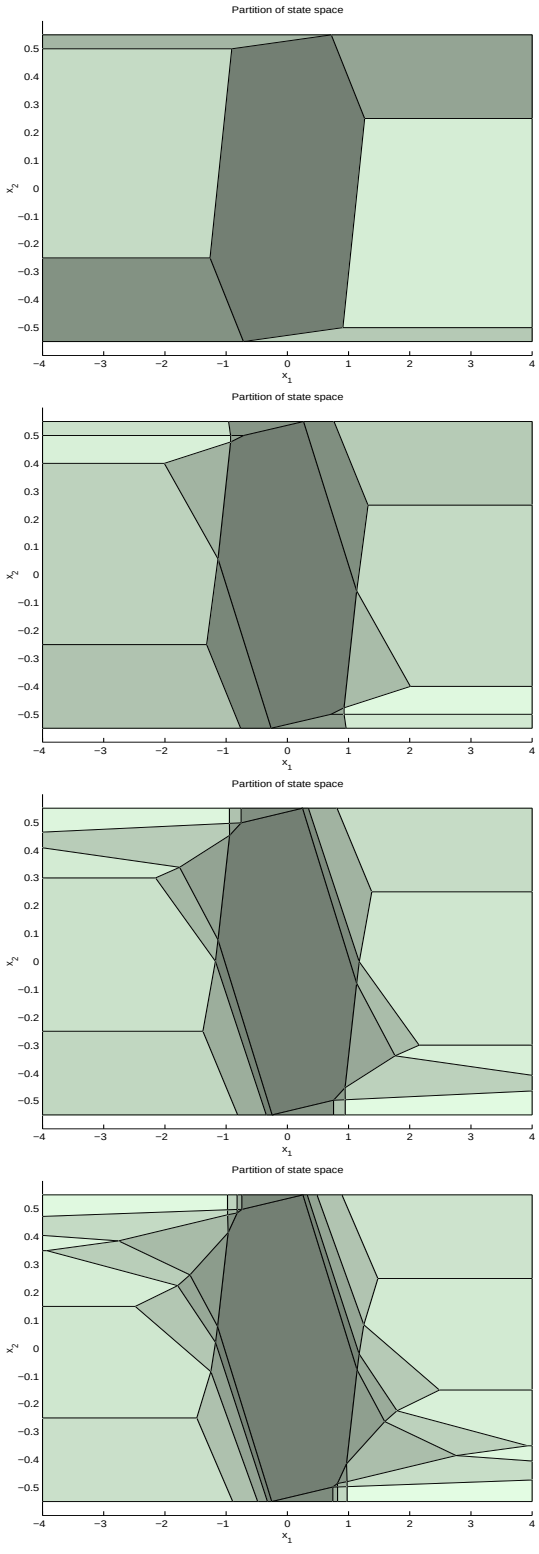

Fig. 3. Polyhedral partition of state space with horizon 15, using 1, 2, 3 and 4 parameters (from top to bottom).

3. Select any unexplored node $N_k \in \mathcal{U}$. If no such node exist, the algorithm terminates. Otherwise, remove the node from $\mathcal{U}$ and go to step 4.

4. Select a hyperplane $a_j^T x = b_j$ from the set of linear inequalities that define all the polyhedra of all critical regions $X_i,\ i \in \mathcal{I}_k$ and let $N_k := (j, \mathcal{I}_k)$.

5. Let $Y \subset X$ denote the polyhedral set defined by the inequalities of all nodes encountered when traversing the tree from the root node N_1 to the node N_k. Let $Y_k^+ := Y \cap \{x \in X | d_j(x) > 0\}$ and $Y_k^- := Y \cap \{x \in X | d_j(x) \leq 0\}$. Let $\mathcal{I}^+ = \emptyset$ and $\mathcal{I}^- = \emptyset$.

6. For all $i \in \mathcal{I}_k$, add i to $\mathcal{I}^+$ if $X_i \cap Y_k^+$ is non-empty, and add i to $\mathcal{I}^-$ if $X_i \cap Y_k^-$ is non-empty.

7. If $|\mathcal{I}^+|/|\mathcal{I}_k| \geq \alpha$ or $|\mathcal{I}^-|/|\mathcal{I}_k| \geq \alpha$, where $0.5 < \alpha < 1$ is some constant, go to step 4.

8. Create two new nodes $N^+ = (*, \mathcal{I}^+)$ and $N^- = (*, \mathcal{I}^-)$. Make these nodes the child nodes of N_k corresponding to positive and negative $d_j(x)$, respectively.

9. If $|\mathcal{I}^+| \neq 1$, add N^+ to $\mathcal{U}$.

10. If $|\mathcal{I}^-| \neq 1$, add N^- to $\mathcal{U}$.

11. Go to step 3.

□

The number of nodes and depth of the tree are strongly dependent on which hyperplanes a selected in step 4, and the value of the parameter α in step 7. In the examples below we have selected hyper-planes randomly with $\alpha = 0.75$, but there will obviously exist better heuristics. If no hyperplane exist such that the loop between steps 4 and 7 terminates, α is increased.

The notation $|\mathcal{I}|$ means the number solutions in $\mathcal{I}$ having the same first r elements. Due to steps 9 and 10 we allow the leaf nodes of the search tree to define a set of regions (rather than a unique region) where the first r elements of the rN-dimensional solution vector are the same. This is sufficient in MPC where we only need to implement the first sample of the control input trajectory. The computationally most complex operation in Algorithm 2 is in step 6 where the emptiness of some polyhedral sets are tested by solving LPs.

Example, continued.

Consider the PWL solution for the double integrator, see also Table 1. In Table 4 the results using Algorithm 2 are shown. In general, the worst-case number of arithmetic operations required to search the tree and evaluate the PWL function is $(2n+1) \cdot (D+r)$, where D is the depth of the tree, r is the number of inputs and n is the number of states. At each node there are n multiplications, n additions and 1 comparison. Moreover, $(2n+1)r$ operations are required to evaluate the affine state feedback in the region. We observe from Table 4 that the computational complexity seems to increase as $\mathcal{O}(\log N)$, where N is the number of critical regions. We note that although the computational complexity increases slowly with the number of critical regions, the memory requirement for storing the PWL function parameters and the nodes increases rapidly. Due to randomness in step 4 of the algorithm, the search tree will be different at each execution, typically causing the numbers in Table 4 to vary by less than 15 % .

□

Table 4. Characteristics of the search trees constructed for the double integrator.

N	Regions	Nodes	Depth	Arithmetic ops.
1	5	13	4	25
2	13	29	5	30
3	23	63	6	35
4	35	97	7	40
5	51	117	8	45
6	71	173	9	50
7	95	241	9	50
8	123	397	10	55
9	155	421	10	55
10	191	493	11	60
11	231	661	11	60
12	277	775	11	60
13	325	901	12	65
14	379	1087	12	65
15	437	1195	12	65

6. SIMULATION EXAMPLE

A laboratory model helicopter (Quanser 3-DOF Helicopter) is sampled with $T = 0.01s$, and the following state-space representation is obtained

$$A = \begin{bmatrix} 1 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0.01 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0.0001 & -0.0001 \\ 0.0019 & 0.0019 \\ 0.0132 & -0.0132 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The states of the system are

- x_1 - elevation
- x_2 - pitch angle
- x_3 - elevation rate
- x_4 - pitch angle rate
- x_5 - integral of elevation error
- x_6 - integral of pitch angle error

The inputs to the system are

- u_1 - front rotor power
- u_2 - rear rotor power

The system is to be regulated to the origin with the following constraints on the inputs and pitch and elevation rates $-1 \leq u_1 \leq 3$, $-1 \leq u_2 \leq 3$, $-0.44 \leq x_3 \leq 0.44$, and $-0.6 \leq x_4 \leq 0.6$. The LQ cost function is given by

$$Q = \text{diag}(100, 100, 10, 10, 400, 200)$$
$$R = I_{2\times 2}$$

and P is given by the algebraic Riccati equation. The following four cases are considered:

(1) $N = 1$, no input parameterization.
(2) $N = 50$, input parameterization, 1 parameter.
(3) $N = 3$, no input parameterization.
(4) $N = 50$, input parameterization, 3 parameters.

The MPC controller was computed using the mp-QP Algorithm of (Tøndel *et al.* 2001) and the following table shows the number of regions and computation times in each case.

Figures 4-6 show results of simulations starting in $x(0) = (0.5, 0.5, 0, 0, 0, 0)^T$. From the accumulated

Table 5. Number of regions and computation times for helicopter example.

Case	Regions	CPU time (s)
1	33	2
2	49	3
3	2528	703
4	3464	1585

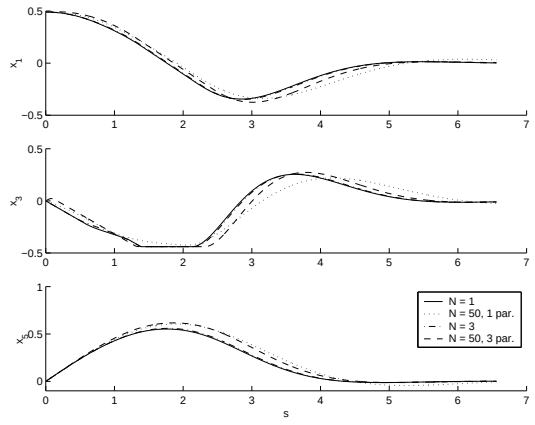

Fig. 4. States x_1, x_3 and x_5, i.e. the elevation and its derivative and integrated error.

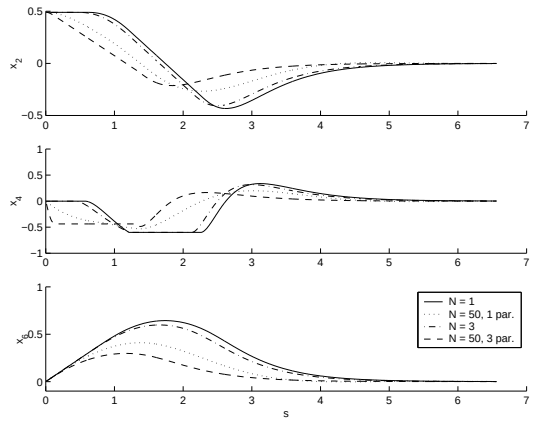

Fig. 5. States x_2, x_4 and x_6, i.e. the pitch angle and its derivative and integrated error.

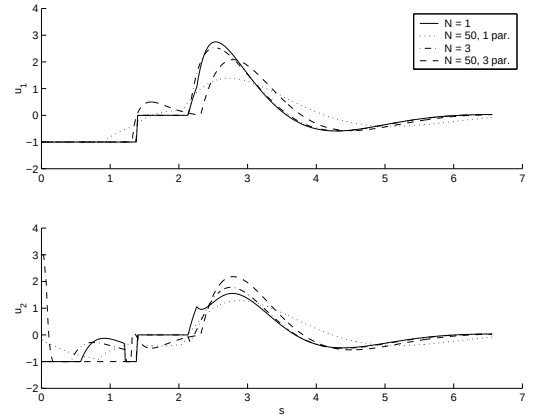

Fig. 6. Control inputs u_1 and u_2.

cost in Figure 7 one can see that the controllers using a horizon of 50 and parameterization of the input definitely outperforms the controllers with horizons of 1 and 3.

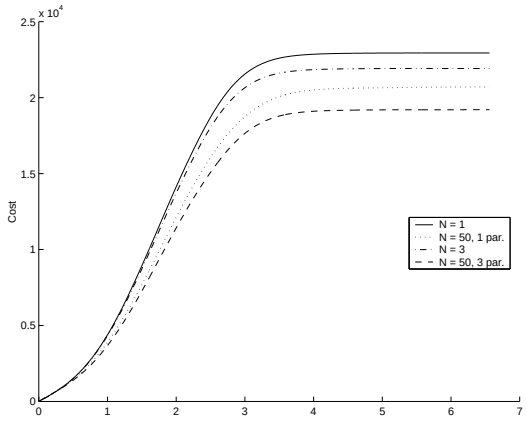

Fig. 7. Accumulated LQ cost.

7. CONCLUSION

It is shown empirically that the use of input trajectory parameterization is a useful method for reducing the computational complexity of explicit MPC based on multi-parametric quadratic programming. An algorithm for efficient real-time evaluation of the PWL explicit solution is also provided.

8. REFERENCES

Bemporad, A. and C. Filippi (2001). Suboptimal explicit MPC via approximate quadratic programming. In: *Proc. IEEE Conf. Decision and Control, Orlando*.

Bemporad, A., F. Borrelli and M. Morari (2000*a*). Optimal controllers for hybrid systems: Stability and piecewise linear explicit form. In: *Proc. Conference on Decision and Control*.

Bemporad, A., K. Fukuda and F. D. Torrisi (2001). Convexity recognition of the union of polyhedra. *Computational Gemometry* **18**, 141–154.

Bemporad, A., M. Morari, V. Dua and E. N. Pistikopoulos (1999). The explicit linear quadratic regulator for constrained systems. Technical Report AUT99-16. Automatic Control Laboratory, ETH Zurich, Switzerland.

Bemporad, A., M. Morari, V. Dua and E. N. Pistikopoulos (2000*b*). The explicit solution of model predictive control via multiparametric quadratic programming. In: *Proc. American Control Conference, Chicago*. pp. 872–876.

Borrelli, F., M. Baotic, A. Bemporad and M. Morari (2001). Efficient on-line computation of explicit model predictive control. In: *Proc. IEEE Conf. Decision and Control, Orlando*.

Johansen, T. A., I. Petersen and O. Slupphaug (2000*a*). Explicit suboptimal linear quadratic regulation with input and state constraints. Technical Report STF72-A00303. SINTEF.

Johansen, T. A., I. Petersen and O. Slupphaug (2000*b*). On explicit suboptimal LQR with state and input constraints. In: *Proc. IEEE Conf. Decision and Control, Sydney*. pp. TuM05–6.

Seron, M., J. A. De Dona and G. C. Goodwin (2000). Global analytical model predictive control with input constraints. In: *Proc. IEEE Conf. Decision and Control, Sydney*. pp. TuA05–2.

Tøndel, P., T. A. Johansen and A. Bemporad (2001). An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In: *Proc. IEEE Conf. Decision and Control, Orlando*. pp. TuP11–4.