A LOGARITHMIC-TIME SOLUTION TO THE POINT LOCATION PROBLEM FOR CLOSED-FORM LINEAR MPC

C.N. Jones [*] **P. Grieder** [**] **S.V. Raković** [***]

[*] *Control Group, Department of Engineering University of Cambridge, Trumpington Street Cambridge CB2 1PZ, UK*
[**] *Automatic Control Laboratory, Swiss Federal Institute of Technology, Physikstrasse 3, ETL K13.2, CH-8092 Zurich*
[***] *Imperial College London, Exhibition Road, London SW7 2BT, United Kingdom*

Abstract: Closed-form Model Predictive Control (MPC) results in a polytopic subdivision of the set of feasible states, where each region is associated with an affine control law. Solving the MPC problem on–line then requires determining which region contains the current state measurement. This is the so-called *point location problem.* For MPC based on linear control objectives (e.g., 1- or ∞-norm), we show that this problem can be written as an additively weighted nearest neighbour search that can be solved on–line in time linear in the dimension of the state space and logarithmic in the number of regions. We demonstrate several orders of magnitude sampling speed improvement over traditional MPC and closed-form MPC schemes.

Keywords: Predictive Control, Parametric Programming, Controller Complexity

1 INTRODUCTION

It is standard practice to implement an MPC controller by solving on–line an optimal control problem that, when the system is linear and the constraints are polyhedral, amounts to computing a single linear or quadratic program at each sampling instant depending on the type of control objective. In recent years, it has become well-known that the optimal input is a piecewise affine function (PWA) defined over a polyhedral partition of the feasible states (Borrelli, 2003). Several methods of computing this affine function can be found in the literature (e.g., (Tøndel *et al.*, 2003*a*; Bemporad *et al.*, 2002; Borrelli, 2003)). The on–line calculation of the control input then becomes one of determining the region that contains the current state and is known as the *point location problem.*

The complexity of calculating this function is clearly dependent on the number of affine regions in the solution. This number of regions is known to grow very quickly and possibly exponentially, with horizon length and state/input dimension (Bemporad *et al.*, 2002). The complexity of the solution therefore implies that for large problems an efficient method for solving the point location problem is needed.

The key contributions to this end have been made by (Tøndel *et al.*, 2003*b*) and (Borelli *et al.*, 2001). In (Tøndel *et al.*, 2003*b*), the authors propose to construct a binary search tree over the polyhedral state-space partition. Therein, auxiliary hyperplanes are used to subdivide the partition at each tree level. Note that these auxiliary hyperplanes may subdivide existing regions. The necessary on–line identification time is logarithmic in the number of subdivided regions, which may be significantly larger than the original number of regions. Although the scheme works very well for smaller partitions, it is not applicable to large controller structures due to the prohibitive pre-processing time. If R is the number of regions and $\bar{F}$ the average number of facets defining a region, then the approach requires the solution to $R^2 \cdot \bar{F}$ LPs[1]. However, the scheme in (Tøndel *et al.*, 2003*b*) is applicable to *any* type of *closed–form* MPC controller, whereas the algorithm proposed in this paper considers only the case in which controllers are obtained via a *linear* cost. The approach proposed here is not directly applicable to non-convex controller partitions and can only be applied to controllers obtained for a *quadratic* cost if the solution exhibits a specific structure.

In (Borelli *et al.*, 2001) the authors exploit the convexity properties of the piecewise affine (PWA) value function of linear MPC problems to solve the point location problem efficiently. Instead of checking whether the point is contained in a polyhedral region, each affine piece of the value function is evaluated for the current state. Since the value function is PWA and convex, the region containing the point is associated to the affine function that yields the largest value. Although this scheme is efficient, it is still linear in the number of regions.

In this paper, we combine the concept of region identification via the value-function (Borelli *et al.*, 2001) with the construction of search trees (Tøndel *et al.*, 2003*b*), by using the link between *parametric* linear programming, Voronoi Diagrams and Delaunay triangulations, recently established in (Raković *et al.*, 2004). We demonstrate that the PWA cost function can be interpreted as a weighted power diagram, which is a type of Voronoi diagram, and exploit recent results in (Arya *et al.*, 1998) to solve the point location problem for Voronoi diagrams in logarithmic time at the cost of very simple pre-processing operations on the controller partition.

We focus on MPC problems with 1- or ∞-norm objectives and show that evaluating the optimal PWA function for a given state can be posed as a nearest neighbour search over a finite set of points. In (Arya *et al.*, 1998) an algorithm is introduced that solves the nearest neighbour problem in n dimensions with R regions in time $\mathcal{O}(c_{n,\epsilon} n \log R)$ and space $\mathcal{O}(nR)$ after a pre-processing step taking $\mathcal{O}(nR\log R)$, where $c_{n,\epsilon}$ is a factor depending on the state dimension and an error tolerance ϵ. Hence, the optimal control input can be found on–line in time logarithmic in the number of regions R.

[1] It is possible to improve the pre-processing time at the cost of less efficient (non-logarithmic) on-line computation times.

The remainder of this paper is organised as follows. In Section 2 the basic MPC problem is formulated, the structure of the closed-form solution is discussed and the problem addressed in this paper is formally defined. Section 3 demonstrates that the point location problem can be posed as a nearest neighbour search over R points. Section 4 provides a brief overview of the logarithmic nearest neighbour algorithm from (Arya *et al.*, 1998). Section 5 provides numerical examples and compares the approach to the current state of the art. Finally, conclusions are given in Section 6.

DEFINITIONS AND NOTATION

Definition 1. (Grünbaum, 2000) A *polyhedron* is the intersection of a finite number of halfspaces: $\mathcal{P} \triangleq \{x \in \mathbb{R}^n \mid Ax \le b\}$. A *polytope* is a bounded polyhedron.

Definition 2. (Face) $\mathcal{F}$ is a *face* of the polytope $\mathcal{P} \subset \mathbb{R}^n$ if there exists a hyperplane $\left\{x \in \mathbb{R}^n \mid a^T x = b\right\}$, where $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, such that $\mathcal{F} = \mathcal{P} \cap \left\{x \in \mathbb{R}^n \mid a^T x = b\right\}$ and $a^T x \le b$ for all $x \in \mathcal{P}$.

Given any integer q let $\mathbb{N}_q \triangleq \{1, 2, \ldots, q\}$.

2 PROBLEM FORMULATION

We consider discrete-time, linear, time-invariant models:

$$x^+ = Ax + Bu,$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, (A, B) is controllable and x^+ is the state at the next point in time given the current measured state $x \in \mathbb{R}^n$ and the input $u \in \mathbb{R}^m$. The state x and the input u are constrained to lie in the polytopic sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ respectively at each point in time, where we assume that the origin is in the interior of $\mathcal{X}$ and $\mathcal{U}$.

The MPC problem is defined, as usual, by specifying a finite-horizon optimal control problem:

$$\begin{aligned} V^\star(x) = \min_{\mathbf{u}} \sum_{k=0}^{N-1} \left(\left\|Qx_k\right\|_p + \left\|Ru_k\right\|_p \right) + \left\|Q_f x_N\right\|_p \\ \text{subject to} \qquad\qquad \\ x_{k-1} \in \mathcal{X}, u_{k-1} \in \mathcal{U}, \forall k \in \mathbb{N}_N \\ x_N \in \mathcal{X}_f \subseteq \mathcal{X}, \quad x_0 = x, \qquad (1) \\ x_k = Ax_{k-1} + Bu_{k-1},\ \forall k \in \mathbb{N}_N \end{aligned}$$

where $\mathbf{u} \triangleq \{u_0, u_1, \ldots, u_{N-1}\}$. If the p–norm used is the 1– or the ∞–norm, then (1) can be re-written as a linear program (LP):

$$\begin{aligned} V^\star(x) = \underset{y}{\text{minimize}} \quad & c^T y \\ \text{subject to} \quad & (x, y) \in \mathcal{P}, \end{aligned} \qquad (2)$$

by introducing an appropriate set of l slack variables which are concatenated with $\mathbf{u}$ to form y.

The polyhedron $\mathcal{P}$ is closed and incorporates all constraints from (1). The interested reader is referred to (Borrelli, 2003; Bemporad *et al.*, 2000) for details on how to compute an appropriate polyhedron $\mathcal{P}$ and cost c such that (2) is equivalent to (1).

The first Nm dimensions of the optimiser $y^\star(x)$ of LP (2) defines the optimal control sequence $\mathbf{u}^\star(x) \triangleq \{u_0^\star(x), \dots, u_{N-1}^\star(x)\}$ for the optimal control problem (1). In MPC, the problem (1) is solved at each sampling instant, and the control law $\kappa(\cdot)$ is defined as the first element in the optimal input sequence:

$$\kappa(x) \triangleq u_0^\star(x).$$

2.1 Solution Structure

Since the problem (2) is an LP, it can be solved off-line as a parametric linear program (pLP). See, for instance, (Borrelli, 2003) for an algorithm for computing the solution to a pLP. First, we need to introduce the notion of a *complex* of polytopes:

Definition 3. (Grünbaum, 2000) A finite family $\mathscr{C}$ of polytopes in $\mathbb{R}^n$ is a *complex* if

- Every face of a member of $\mathscr{C}$ is itself a member of $\mathscr{C}$
- The intersection of any two members of $\mathscr{C}$ is a face of each of them

If a polytope $\mathcal{Q}$ is a member of a complex $\mathscr{C}$ we call $\mathcal{Q}$ a face of $\mathscr{C}$ and write $\mathcal{Q} \in \mathscr{C}$. Faces of dimension n are called *cells* of the complex.

A basic result on the nature of the solution to a parametric linear program is given next:

Theorem 1. (Solution to a pLP).
Let $\mathcal{P} \subset \mathbb{R}^{n+n_y}$ be a polyhedron and

$$\pi(\mathcal{P}) \triangleq \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^{n_y} \text{ such that } (x, y) \in \mathcal{P}\}.$$

For each x in $\pi(\mathcal{P})$, let

$$V^\star(x) \triangleq \inf_y \{c^T y \mid (x, y) \in \mathcal{P}\} \tag{3}$$

where $c \in \mathbb{R}^{n_y}$.

Then $V^\star : \mathbb{R}^n \to \mathbb{R}$ is a convex, piecewise affine function defined over a complex $\mathscr{C}$ whose cells partition $\pi(\mathcal{P})$. Furthermore, there exists a continuous, piecewise affine function[2] $\upsilon : \mathbb{R}^n \to \mathbb{R}^{n_y}$ such that $c^T \upsilon(x) = V^\star(x)$ for every $x \in \pi(\mathcal{P})$.

Thus by Theorem 1, the optimal cost of (2) is a convex, piecewise affine function of the state x, taking $\mathbb{R}^n$ to $\mathbb{R}$ and is defined over a complex $\mathscr{C} = \{\mathcal{R}_1, \dots, \mathcal{R}_R\}$:

$$V^\star(x) = F_r^T x + f_r, \quad \text{if } x \in \mathcal{R}_r, \quad r \in \mathbb{N}_R, \tag{4}$$

where each cell $\mathcal{R}_r$ is a polytope. Furthermore, the optimiser of LP (2) is a piecewise affine function of x taking $\mathbb{R}^n$ to $\mathbb{R}^{N(m+l)}$ as is the control law $\kappa(\cdot)$, which takes $\mathbb{R}^n$ to $\mathbb{R}^m$ and is defined over the same complex:

$$\kappa(x) = u_0^\star(x) = T_r x + t_r, \text{ if } x \in \mathcal{R}_r, \ r \in \mathbb{N}_R.$$

2.2 Point Location Problem

Problem 1. Given a measured state x and complex $\mathscr{C} = \{\mathcal{R}_1, \dots, \mathcal{R}_R\}$, determine any integer[3] $\mathsf{i}(x) \in \mathbb{N}_R$ such that polytope $\mathcal{R}_{\mathsf{i}(x)}$ contains x.

The function $\mathsf{i}(x)$ defines the control law $\kappa(x)$ as

$$\kappa(x) = u_0^\star(x) = T_{\mathsf{i}(x)} x + t_{\mathsf{i}(x)}.$$

As $V^\star(x)$ is convex, the calculation of $\mathsf{i}(x)$ can be written as (Borelli *et al.*, 2001):

$$\mathsf{i}(x) = \arg\max_{r \in \mathbb{N}_R} \left\{F_r^T x + f_r\right\}. \tag{5}$$

As was proposed in (Borelli *et al.*, 2001), $\mathsf{i}(x)$ can be computed from (5) by simply evaluating the cost $F_r^T x + f_r$ for each $r \in \mathbb{N}_R$ and then taking the largest. This procedure requires $2nR$ flops and has a storage requirement of $(n+1)R$.

In the following sections we will show that with a negligible pre-processing step, (5) can be computed in *logarithmic* time, which is a significant improvement over the *linear* time result of (Borelli *et al.*, 2001).

3 POINT LOCATION AND NEAREST NEIGHBOURS

In this section we show that for pLPs, the point location problem can be written as an *additively weighted nearest neighbour search*, or a search over R points in $\mathbb{R}^n$ to determine which is closest to the state x.

Consider the finite set of points called *sites* $\mathcal{S} \triangleq \{s_1, \dots, s_R\}$ and the weights $\mathcal{W} \triangleq \{w_1, \dots, w_R\}$, where $(s_i, w_i) \in \mathbb{R}^n \times \mathbb{R}$, $\forall i \in \mathbb{N}_R$. Given a point x in $\mathbb{R}^n$, the weighted nearest neighbour problem is the determination of the point $s_r \in \mathcal{S}$ that is closest to x, for all $(s_j, w_j) \in \mathcal{S} \times \mathcal{W}$, $j \in \mathbb{N}_R$. Associated with each site is a set of points $\mathcal{L}_r \subset \mathbb{R}^n$ such that for each $x \in \mathcal{L}_r$, x is closer to s_r than to any other site:

$$\mathcal{L}_r \triangleq \{x \mid \left\|s_r - x\right\|_2^2 + w_r \le \left\|s_j - x\right\|_2^2 + w_j, \ \forall j \in \mathbb{N}_R\}. \tag{6}$$

Note that the sets $\mathcal{L}_r$ form a complex $\mathscr{C}_V \triangleq \{\mathcal{L}_1, \dots, \mathcal{L}_R\}$(Aurenhammer, 1991). If the weights

[2] Note that in general, the optimiser of (3) is set-valued.

[3] The state may be on the boundary of several regions.

w_r are all zero, then the sets $\mathcal{L}_r$ form a *Voronoi diagram*, otherwise they are called a *power diagram* (Aurenhammer, 1991).

We now state the following result:

Theorem 2. If $\mathscr{C}$ is a solution complex, then $\mathscr{C}$ is a power diagram.

PROOF. It suffices to show that for any solution complex of pLP (2), $\mathscr{C} \triangleq \{\mathcal{R}_1, \dots, \mathcal{R}_R\}$, it is possible to define a set of sites and weights such that their power diagram is equivalent to $\mathscr{C}$.

It follows from Theorem 1 and (4)–(5) that x is contained in cell $\mathcal{R}_r$ if and only if

$$F_r^T x + f_r \geq F_j^T x + f_j, \qquad \forall j \in \mathbb{N}_R,$$

or equivalently, if and only if:

$$-F_r^T x - f_r \leq -F_j^T x - f_j, \qquad \forall j \in \mathbb{N}_R.$$

Define the R sites and weights as:

$$\begin{aligned} s_r &\triangleq \frac{F_r}{2} \\ w_r &\triangleq -f_r - \left\| \frac{F_r}{2} \right\|_2^2 = -f_r - \|s_r\|_2^2 \end{aligned} \tag{7}$$

For all $r \in \mathbb{N}_R$ and a given x it follows that:

$$\|s_r - x\|_2^2 + w_r = -F_r^T x - f_r + \|x\|_2^2$$

Recalling the definition of $\mathcal{L}_r$ in (6) we obtain the following $\forall j \in \mathbb{N}_R$:

$$\begin{aligned} \mathcal{L}_r &\triangleq \left\{ x \;\middle|\; \begin{array}{l} \|s_r - x\|_2^2 + w_r \\ \quad \leq \|s_j - x\|_2^2 + w_j, \end{array} \right\} \\ &= \left\{ x \;\middle|\; \begin{array}{l} -F_r^T x - f_r + \|x\|_2^2 \\ \quad \leq -F_j^T x - f_j + \|x\|_2^2, \end{array} \right\} \\ &= \{x \mid -F_r^T x - f_r \leq -F_j^T x - f_j, \} \\ &= \{x \mid F_r^T x + f_r \geq F_j^T x + f_j, \} \\ &= \mathcal{R}_r \end{aligned}$$

Thus the equivalence of the power diagram of the set of sites and weights (7) and the solution complex $\mathscr{C}$ of a corresponding pLP is established. □

A very important consequence of Theorem 2 is that the point location problem (5) can be solved by determining which site s_r is closest to the current state x:

$$\begin{aligned} \mathsf{i}(x) &= \left\{ r \in \mathbb{N}_R \;\middle|\; \begin{array}{l} \|s_r - x\|_2^2 + w_r \leq \\ \|s_j - x\|_2^2 + w_j, \end{array} \forall j \in \mathbb{N}_R \right\} \\ &= \min_{r \in \mathbb{N}_R} \left\| \begin{pmatrix} s_r \\ \sqrt{w_r} \end{pmatrix} - \begin{pmatrix} x \\ 0 \end{pmatrix} \right\| \end{aligned}$$

Since this problem has been well studied in the computational geometry literature we propose to adapt an efficient algorithm introduced in (Arya *et al.*, 1998) that solves the nearest neighbour problem in *logarithmic time* and thereby solves the point location problem in *logarithmic time.* The next section will give a brief introduction to the algorithm introduced in (Arya *et al.*, 1998).

Remark 1. In (Aurenhammer, 1987) it was shown that a complex is a power diagram if and only if there exists a piecewise affine, continuous and convex function in $\mathbb{R}^{n+1}$ such that the projection of each affine piece of the function from $\mathbb{R}^{n+1}$ to $\mathbb{R}^n$ is a cell in the complex. This piecewise affine function is called a *lifting* of the complex. From the proof of Theorem 2, it is clear that the solution complex of every pLP has a lifting.

Remark 2. If a 2–norm is used in the formulation of the MPC problem (1) then the resulting solution complex may or may not have a lifting. Although it is not difficult to find problems for which a lifting does not exist, general conditions for the existence of a lifting for quadratic costs are not known. See (Aurenhammer, 1991; Rybnikov, 1999) for details on testing when a complex has an appropriate lifting.

4 APPROXIMATE NEAREST NEIGHBOUR: LOGARITHMIC SOLUTION

In this section, the key aspects of the approximate nearest neighbour search algorithm presented in (Arya *et al.*, 1998) will be restated. Given a point $q \in \mathbb{R}^n$, a positive real ϵ and a set of R points in $\mathbb{R}^n$, the point p is a $(1+\epsilon)$-approximate nearest neighbour of q, if its distance from q is within a factor of $(1+\epsilon)$ of the distance from the true nearest neighbour.

Remark 3. The ϵ error is required in order to prove the logarithmic search time (Arya *et al.*, 1998). As the optimal feedback $\kappa(x)$ can be chosen to be continuous (see Theorem 1) this error in determining the region translates into a maximum error in the input that is proportional to ϵ. Therefore, the error in the control input can be made arbitrarily small with an appropriate selection of ϵ.

As shown in (Arya *et al.*, 1998), it is possible to pre-process the R data points in $O(nR \log R)$ time and $O(nR)$ space, such that the approximate nearest neighbour can be identified in $O(c_{n,\epsilon} \log R)$ time, where $c_{n,\epsilon}$ is a factor depending only on state-space dimension n and accuracy ϵ.

The authors in (Arya *et al.*, 1998) propose to construct a so called *balanced box-decomposition tree* or BBD-tree. The BBD-tree is a hierarchical decomposition of the state-space into hyperrectangles (cells) whose sides are orthogonal to the coordinate axes. The BBD tree has two key properties which are vital in obtaining the logarithmic runtime bounds. Namely, as one descends the BBD-tree, the number of points associated to each cell decreases exponentially *and* the aspect ratio (ratio of longest to shortest side of each cell) is bounded by a constant.

The BBD-tree is constructed through the repeated application of two operations, *splits* and *shrinks*. A *split* subdivides a cell into two equally sized *children* by adding an axis-orthogonal hyperplane. This operation guarantees the exponential decrease in the number of points associated to each cell but it cannot give bounds on the aspect ratio. The *shrink*, partitions a cell into two subcells by using a hyper-rectangle which is located in the interior of the parent cell. The shrink operation corresponds to 'zooming in' to regions where points are highly clustered. A simple strategy to construct the BBD-tree is to apply splits and shrinks alternately. This procedure is repeated until the number of points associated to each cell is at most one.

In order to describe the on-line search, we will introduce the following definition: the *distance* between a point q and a cell is the closest distance between q and any part of the cell. Given a query point q, the algorithm first identifies the associated leaf cell by a simple descent through the tree in $O(\log \mathrm{R})$ time. It is possible to enumerate the c cells closest to q in increasing order in $O(cn \log \mathrm{R})$ time (Arya *et al.*, 1998). The necessary number of cells c is bounded by a constant which can be determined without constructing the BBD-tree (Arya *et al.*, 1998). Each cell is then visited (closest cell first) and the closest point seen so far is stored as p. As soon as the distance from a cell to q exceeds $dist(p,q)/(1+\epsilon)$, it follows that the search can be terminated and p can be reported as the approximate nearest neighbour (Arya *et al.*, 1998).

5 EXAMPLES

In this section we consider various systems and compare the on–line calculation times of the method proposed in this paper to the scheme in (Borelli *et al.*, 2001). Although the scheme in (Tøndel *et al.*, 2003*b*) may lead to more significant runtime improvements than (Borelli *et al.*, 2001), the necessary pre-processing time is prohibitive for large partitions and we therefore refrain from performing a comparison to that scheme.

5.1 Large Random System

Example 1. Consider the following 4-dimensional LTI system:

$$x_{k+1} = \begin{bmatrix} 0.7 & -0.1 & 0 & 0 \\ 0.2 & -0.5 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0 \\ 0.5 & 0 & 0.5 & 0.5 \end{bmatrix} x_k + \begin{bmatrix} 0 & 0.1 \\ 0.1 & 1 \\ 0.1 & 0 \\ 0 & 0 \end{bmatrix} u_k$$

Subject to constraints $||u_k||_\infty \leq 5$ and $||x_k||_\infty \leq 5$.

Example 1 was solved for the infinity norm $p = \infty$, prediction horizon $N = 5$ and for weighting matrices $Q = I$ and $R = I$. The resulting controller partition consists of $R = 12,290$ regions. The construction of the search tree required 0.03 seconds. In comparison, the approach in (Tøndel *et al.*, 2003*b*) would require the solution to approximately $151,000,000$ LPs, which is clearly prohibitive in terms of runtime. For $\epsilon = 0.01$, the average and worst-case number of floating point operations to compute the input using ANN (Mount and Arya, 1998) are $29,450$ and 36910 respectively. In comparison, the approach in (Borelli *et al.*, 2001) always takes exactly $160,000$ operations.

5.2 Randomly Generated Regions

In this section we compare the computational complexity of the approach presented in this paper with that discussed in (Borelli *et al.*, 2001) for very large systems. The currently available multi-parametric solvers (Kvasnica *et al.*, 2003) produce reliable results for partitions of up to approximately $30,000$ regions. However, methods are currently being developed that will provide solutions for much larger problems. Therefore, in order to give a speed comparison we have randomly generated vectors F_r and f_r in the form of (5). The code developed in (Arya *et al.*, 1998), which is available at (Mount and Arya, 1998), was then used to execute $1,000$ random queries and the worst-case is plotted in Figure 1. For all of the queries the error parameter ϵ was set to zero and therefore the solution returned is the exact solution. It should be noted that the preprocessing time for one million regions and 20 dimensions is merely 22.2 seconds.

Figure 1 shows the number of floating point operations (flops) as a function of the number of regions for the two approaches and the dimension of the state-space. Note that both axes are logarithmic.

A 3.0GHz Pentium 4 computer can execute approximately 800×10^6 flops/second. It follows that for a 10 dimensional system whose solution has one million regions, the control action can be computed at a rate of 20kHz using the proposed method, whereas that given in (Borelli *et al.*, 2001) could run at only 35Hz.

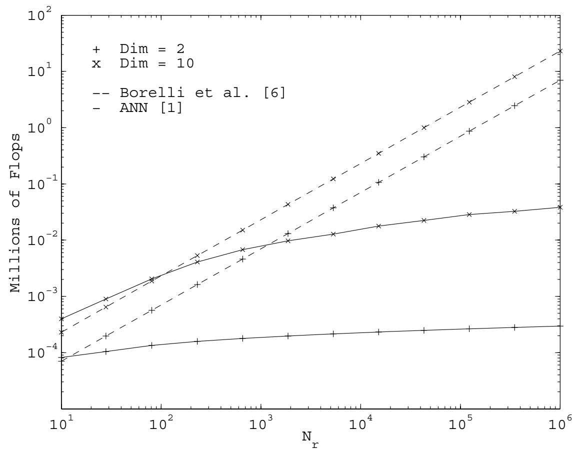

Fig. 1. Comparison of ANN (Solid lines) to (Borelli *et al.*, 2001) (Dashed lines)

It is clear from Figure 1 that the calculation speed of the proposed method is very good for systems with a large number of regions. Furthermore note that controller partitions where ANN does worse than (Borelli *et al.*, 2001) are virtually impossible to generate, i.e. a partition in dimensions $n = 10$ with less than $R = 100$ regions is very difficult to contrive. Hence, it can be expected that for all systems of interest, the proposed scheme will result in a significant increase in speed. Since explicit feedback MPC is generally being applied to systems with very fast dynamics, any speedup in the set-membership test is useful in practice, i.e. the scheme proposed here is expected to significantly increase sampling rates.

6 CONCLUSION

This paper has presented a method of solving the point location problem for linear-cost MPC problems. If the controller partition exhibits a specific structure, the proposed scheme can also be applied to quadratic-cost MPC problems. It has been shown that the method is linear in the dimension of the state-space and logarithmic in the number of regions. Numerical examples have demonstrated that this approach is superior to the current state of the art and that for realistic examples, several orders of magnitude improvement in sampling rates are possible.

The examples in this paper have been prepared with the MPT toolbox (Kvasnica *et al.*, 2003) and Figure 1 was calculated using the ANN library (Mount and Arya, 1998).

References

Arya, S., D.M. Mount, N.S. Netanyahu, R. Silverman and A.Y. Wu (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM* **45**(6), 891–923.

Aurenhammer, F. (1987). A criterion for the affine equivalence of cell complexes in $\mathbb{R}^d$ and convex polyhedra in $\mathbb{R}^{d+1}$. *Discrete and Computational Geometry* **2**, 49–64.

Aurenhammer, F. (1991). Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys.*

Bemporad, A., F. Borrelli and M. Morari (2000). Explicit solution of constrained $1/\infty$–norm model predictive control. In: *Proceedings of the 39th IEEE Conference on Decision and Control.*

Bemporad, A., M. Morari, V. Dua and E.N. Pistikopoulos (2002). The explicit linear quadratic regulator for constrained systems. *Automatica* **38**(1), 3–20.

Borelli, F., M. Baotić, A. Bemporad and M. Morari (2001). Efficient on-line computation of constrained optimal control. In: *Proceedings of the 40th IEEE Conference on Decision and Control.* Orlando, Florida. pp. 1187–1192.

Borrelli, F. (2003). *Constrained Optimal Control Of Linear And Hybrid Systems.* Vol. 290 of *Lecture Notes in Control and Information Sciences.* Springer-Verlag.

Grünbaum, B. (2000). *Convex Polytopes.* second ed.. Springer-Verlag.

Kvasnica, M., P. Grieder, M. Baotić and M. Morari (2003). Multi Parametric Toolbox (MPT). In: *Hybrid Systems: Computation and Control.* Lecture Notes in Computer Science. Springer Verlag. `http://control.ee.ethz.ch/~mpt`.

Mount, D. and S. Arya (1998). Ann: Library for approximate nearest neighbor searching. `http://www.cs.umd.edu/~mount/ANN/`.

Raković, S.V., P. Grieder and C. Jones (2004). Computation of Voronoi Diagrams and Delaunay Triangulation via Parametric Linear Programming. Technical Report AUT04-03. Automatic Control Lab. ETHZ, Switzerland. `http://control.ethz.ch/`.

Rybnikov, K. (1999). Stresses and liftings of cell complexes. *Discrete and Computational Geometry* **21**(4), 481 – 517.

Tøndel, P., T. A. Johansen and A. Bemporad (2003*a*). An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica* **39**(3), 489–497.

Tøndel, P., T. A. Johansen and A. Bemporad (2003*b*). Computation of piecewise affine control via binary search tree. *Automatica* **39**(5), 945–950.