Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Nonlinear parametric optimization using cylindrical algebraic decomposition

Ioannis A. Fotiou*, Pablo A. Parrilo† and Manfred Morari*

Abstract— **In this paper, a new method is presented for optimization of parametric families of polynomial functions subject to polynomial constraints. The method is based on cylindrical algebraic decomposition (CAD). Given the polynomial objective and constraints, the method constructs the corresponding CAD offline, extracting in advance all the relevant structural information. Then, given the parameter value, an online procedure uses the precomputed information to efficiently evaluate the optimal solution of the original optimization problem. The method is very general and can be applied to a broad range of problems.**

I. INTRODUCTION

Model predictive control is a very active area of research with broad industrial applications [1]. It is among the few control methodologies that provides a systematic way to perform nonlinear control synthesis under state and input constraints. This ability of dealing with constraints is one of the main reasons for the practical success of model predictive control (MPC) [2].

MPC uses on-line optimization to obtain the solution of an optimal control problem in real time. This method has been proven most effective for applications. Typically, the optimal control problem can be formulated as a discrete-time mathematical program, whose solution yields a sequence of control moves. Out of these control moves only the first is applied, according to the receding horizon control (RHC) scheme. The specific form of the corresponding mathematical program can be a linear program (LP), a quadratic program (QP) or a general nonlinear program (NLP).

Technology and cost factors, however, make the direct implementation of receding horizon control difficult, or in some cases impossible. In the standard linear MPC case, the corresponding optimization problem is a convex QP, with linear constraints. In this case, an alternative approach to the solution of the optimal control problem is to compute the control law off-line, by solving the corresponding program parametrically [3]. That is, we compute the explicit formula giving the solution of the mathematical program (control inputs) as a function of the problem parameters (measured state). The solution is then efficiently implemented on-line as a lookup table.

In the general case of nonlinear systems and constraints, the MPC formulation gives rise to a *parametric optimization problem*, where as before the solution can be expressed as an explicit function of the parameters. However, in contrast to the linear MPC case, no "simple" expression of the optimal solution is possible, as it necessarily involves implicit algebraic functions. Nevertheless, while a closed form expression is not possible, a parametrization of the optimal solution is still possible by combining a precomputation stage using algebraic techniques and the on-line solution of univariate polynomial equations.

In this paper, we use cylindrical algebraic decomposition (CAD) to perform nonlinear parametric optimization of polynomial functions subject to polynomial constraints. The proposed method encompasses linear and quadratic parametric optimization as special cases.

II. PARAMETRIC OPTIMIZATION AND CAD

A nonlinear parametric optimization problem generally assumes the form

$$\min_u J(u,x) \qquad \text{s.t.} \quad g(u,x) \leq 0 \tag{1}$$

where $J(u,x)$ is a polynomial function in u and x, $u \in \mathbb{R}^n$ is the decision variable vector, $x \in \mathbb{R}^m$ is the parameter vector and $g(u,x)$ is a vector polynomial function. The inequality is meant in the usual componentwise fashion. By parametric optimization, we mean minimizing the function $J(u,x)$ with respect to u for any given parameter x in the region of interest.

Therefore, the nonlinear parametric optimization problem this paper addresses is finding a computational procedure for evaluating the maps

$$\begin{aligned} u^*(x): \quad & \mathbb{R}^m \quad \longrightarrow \quad \mathbb{R}^n \\ J^*(x): \quad & \mathbb{R}^m \quad \longrightarrow \quad \mathbb{R} \end{aligned} \tag{2}$$

where

$$u^* = \arg\min_u J(u,x)$$
$$J^* = \min_u J(u,x).$$

To keep our discussion simple, we restrict our attention to those parameters x for which problem (1) has a unique minimizer. That way $x \longrightarrow u^*$ is a function and not a point-to-set map. We also assume that the feasible set is compact. Nevertheless, the algorithm presented in this paper is easily extended to the cases where the minimizer is not unique or the feasible set is not compact.

Before presenting our approach we first have to describe some basic concepts.

* Automatic Control Laboratory, Swiss Federal Institute of Technology, CH-8092 Zurich, Switzerland.
† Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139-4307, USA.
Corresponding Author: Email: fotiou@control.ee.ethz.ch

0-7803-9568-9/05/$20.00 ©2005 IEEE

A. Cylindrical algebraic decomposition

Given a set $P \subset \mathbb{R}[x_1,\dots,x_n]$ of multivariate polynomials in n variables, a CAD is a special partition of $\mathbb{R}^n$ into components, called *cells*, over which all the polynomials have constant signs. The algorithm for computing a CAD also provides a point in each cell, called *sample point*, which can be used to determine the sign of the polynomials in the cell [4].

To perform optimization, a CAD is associated with a Boolean formula. This Boolean formula can either be quantified or quantifier free. By a quantifier-free Boolean formula we mean a formula consisting of polynomial equations $(f_i(x) = 0)$ and inequalities $(f_j(x) \leq 0)$ combined using the Boolean operators $\wedge$ (and), $\vee$ (or), and $\rightarrow$ (implies). In general, a *formula* is an expression in the variables $x = (x_1,...,x_q)$ of the following type:

$$Q_1x_1...Q_sx_s \quad \mathscr{F}(f_1(x),...,f_r(x)) \tag{3}$$

where Q_i is one of the quantifiers $\forall$ (for all) and $\exists$ (there exists). Furthermore, $\mathscr{F}(f_1(x),...,f_r(x))$ is assumed to be a quantifier-free Boolean formula.

B. Construction of CAD

Obtaining the CAD involves computing with discriminants and resultants. This procedure is called *projection phase* and has as many steps as the number of variables x_i in $\mathscr{F}(f_1(x),...,f_r(x))$ minus one. The main idea is, given the set of polynomials $\{f_i(x)\}$, to obtain in each step $k = 1,\dots,q-1$ a new set of polynomials $\mathscr{P}_k(f_i(x))$ eliminating one variable at a time. That is, the new set of polynomials will depend only on $q-1$ variables $\{x_1,x_2,\dots,x_{q-1}\}$.

Along with the new set of polynomials, the CAD construction algorithm provides us with a special set of polynomials attached to each projection level, called the *projection level factors* denoted by $\{L_i^d\}_{i=1..t_d}$. The set of the real roots of these polynomials contains critical information about the CAD, defining the boundaries of its cells. These roots can be isolated points in $\mathbb{R}^n$, curves, surfaces or hypersurfaces, depending on the dimension of the projection space.

C. Posing the problem

Suppose we have to solve problem (1). We associate with problem (1) the following boolean expression

$$(g(u,x) \leq 0) \wedge (\gamma - J(u,x) \geq 0) \tag{4}$$

We then compute the CAD defined by the polynomial expressions in (4). For this, we use QEPCAD [5]. The signs of the polynomials appearing in (4) as well as of $\mathscr{P}_k(f_i(x))$ resulting from the projection steps are determined in each cell. These signs, in turn, determine wether (4) is true or false in each particular cell. For our purposes, it is enough that QEPCAD determines the truth or falsehood of full-dimensional cells only. The sample points in these cells are rational numbers and the computations associated with them are much easier than in the general case. We instruct QEPCAD to do so with the `measure-zero-error` command.

All the information we need to solve problem (1) is the level factor polynomials associated with the CAD of system (4), the sample points, and the knowledge of which cells are "true" and which "false."

D. A simple CAD

Let us look at the CAD of the following set of polynomial inequalities

$$\left\{ \begin{array}{rcl} u^4 - 10u^2 + u + 1 & \leq & \gamma \\ -7u + 17 & \leq & -\gamma \end{array} \right\} \tag{5}$$

The level factor polynomials for system (5) are

$$\begin{array}{lcl} L_1^2(\gamma,u) & = & u^4 - 10u^2 + u - \gamma + 1 \\ L_2^2(\gamma,u) & = & 7u - \gamma - 17 \\ & & \\ L_1^1(\gamma) & = & 256\gamma^3 + 12032\gamma^2 + 133728\gamma - 149989 \\ L_2^1(\gamma) & = & \gamma^4 + 68\gamma^3 + 1244\gamma^2 + 934\gamma - 49857 \end{array} \tag{6}$$

Note that the level-two factors are the polynomials as they appear in (5). This is always the case with the last projection level, since no variable has yet been eliminated (projected).

As briefly mentioned before, the set of real roots of the level-one factors $L_i^1(\gamma)$ in (6) will partition the γ space into zero- and one-dimensional cells. These roots can clearly be seen in Figure 1: lines parallel to the u axis mark their position. These positions correspond to points where the two polynomials intersect or the tangent to them becomes parallel to the u axis (critical points).

Accordingly, the root set of the level-two factors, *together with the one of the first level*, will define the boundaries of the cells in the joint (γ,u) space. The true cells of the (γ,u) space are specially marked. In optimization, they would correspond to the problem feasible region in the same space, the parameter x having been specified.

III. The algorithm

In the general case of nonlinear parametric optimization there exists no explicit, closed-form formula giving the optimizer u^* or the optimal value J^* as a function of the parameter x. For example, there exists no expression that gives the roots of a polynomial of degree greater than four in terms of elementary functions of its coefficients. We will however present an algorithm which constitutes an efficient computational procedure to evaluate the map from $x \in \mathbb{R}^m$ to $u^* \in \mathbb{R}^n$ and $J^* \in \mathbb{R}$. Our objective is namely to evaluate the maps in (2). The CAD for system system (4) associated to the optimization problem at hand has been constructed in advance and the information contained therein is available to the algorithm presented below.

Note that the variables we now deal with are $(x_1,\dots,x_m,\gamma,u_1,\dots,u_n)$ appearing in (4). The projection steps of the CAD construction phase will first eliminate u_n moving from the end of the list to the beginning.

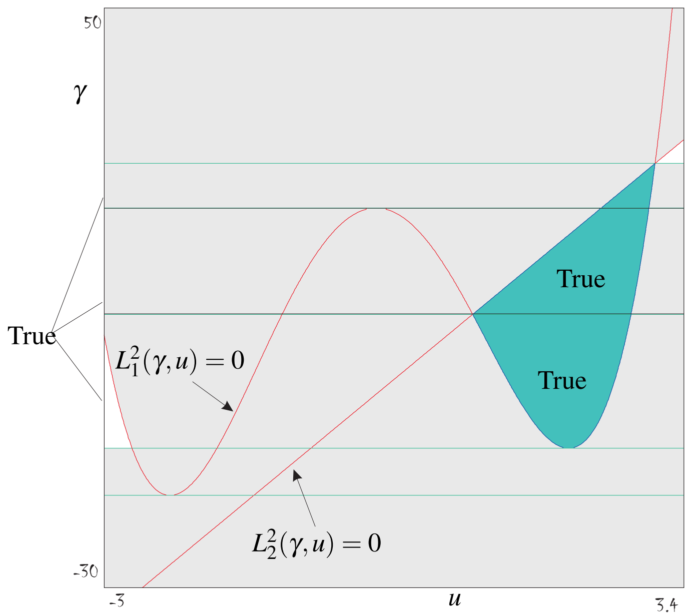

Fig. 1. CAD of the polynomials in (5).

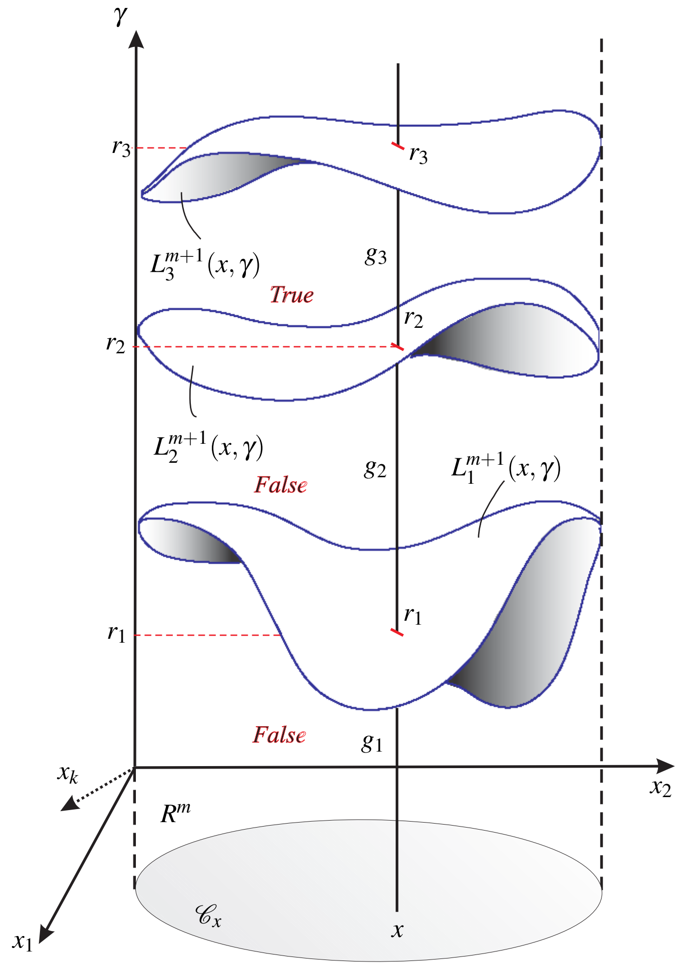

Fig. 2. Lifting to the γ space. The Figure is based on Christopher Brown's ISSAC 2004 CAD tutorial slides.

A. The algorithm

First step *(Initialization)* We determine in what cell in the x-space the given x parameter lies. This can be done by checking the cells of the CAD and the corresponding level factor polynomials defining their boundaries. The roots of the level factors of the first level (i.e. the level resulting after the last projection step) partition the x_1 space into level-one cells that are either (zero dimensional) points in $\mathbb{R}$ or (one dimensional) line segments that may also extend to infinity. Accordingly, level-two factors (together with the root set information of the first level factors) partition the (x_1,x_2) space into level-two cells that can be points embedded in $\mathbb{R}^2$, one-dimensional curves or two-dimensional subsets of $\mathbb{R}^2$. Same holds for higher dimensions up to $\mathbb{R}^m$. For fixed parameters, the point $x \in \mathbb{R}^m$ belongs to a unique cell $\mathscr{C}_x \subseteq \mathbb{R}^m$.

Second step *(Finding J^*)* We now lift cell $\mathscr{C}_x$ up to the space of the cost-associated variable γ (see Figure 2). Some cells in the joint (x,γ) space will be true, some will be false. In Figure 2, the surfaces represent the zero sets of the $(m+1)$-level factors L_i^{m+1}. Fixing the value of x we obtain zero- and one-dimensional subcells along the γ axis. These are depicted in Figure 2 with the thick black line rising from point x. We then look for that true cell in $\mathbb{R}^{m+1}$ which for the given value of x contains the smallest γ value among all other cells in the cylinder above $\mathscr{C}_x$. We may think of it as the first true cell counting form the bottom up. We denote it by $\mathscr{G} \subseteq \mathbb{R}^{m+1}$. If no such cell exists, then problem (1) is infeasible for the given value of x. If the cell exists but happens to be unbounded from below, then for the given value of x optimization problem (1) is unbounded (from below). In case it is not, the minimum is attained and its value is the minimum value of γ in $\mathscr{G}$ obtained for the fixed value of x.

The reasoning above is depicted in Figure 2. By substituting the fixed value of x into the $(m+1)$-level factors, we obtain three roots r_1, r_2 and r_3, each one corresponding to a different level factor, which partition the resulting γ axis rising from $x \in \mathscr{C}_x$ into four one-dimensional subcells $g_i \subseteq \mathbb{R}$. It happens that $\mathscr{G}$ is the cell over $\mathscr{C}_x$, between surfaces L_2 and L_3. Consequently, the optimal cost value will be r_2 which is found by solving the univariate polynomial equation $L_2^{m+1}(\gamma)=0$. This equation may have more than one real root. Which one corresponds to γ^* is given by the CAD sample point information and can be unambiguously determined. This will become clear in the example to follow.

Third step *(Finding u^*)* To determine the optimizer $u^* \in \mathbb{R}^n$ we have to lift the $(x,\gamma) \in \mathbb{R}^{m+1}$ point in the space of the decision variables. Remember that we restrict our attention to those x that yield a unique optimizer $u^* \in \mathbb{R}^n$.

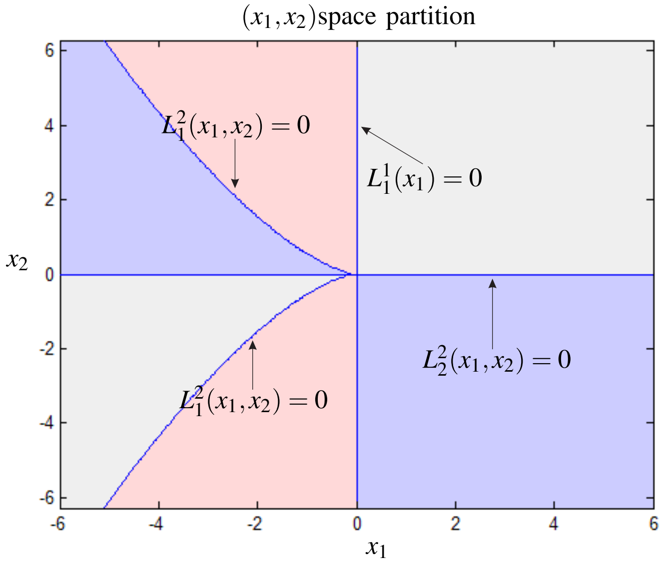

Fig. 3. Level factors partitioning the x space of problem (7).

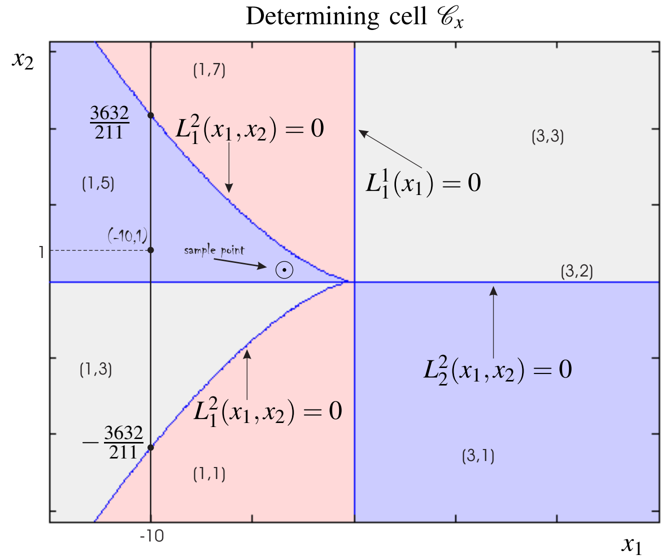

Fig. 4. Point $(-10,1)$ lies in cell $(1,5)$. The sample point of this cell is labelled with a $\odot$ sign. The level factor polynomials define the boundaries of the cells.

First, we substitute the values of x and γ^* in the level factors $\{L_i^{m+2}(x,\gamma,u_1)\}_{i=1\dots t_{m+2}}$. What we obtain is t_{m+2} univariate polynomials in u_1 which we denote by $\{s_i^1(u_1)\}_{i=1\dots t_{m+2}}$. We solve them to calculate their real roots set $\mathscr{A}_1$. We then use the precomputed CAD cell information together with the level factor signs of each cell to determine which root in $\mathscr{A}_1$ corresponds to the optimizer u_1^*. This root will be part of the optimizer vector $u^*=(u_1^*,\dots,u_n^*)$. This can be formalized as follows

$$\text{Level m+2:}\quad \begin{array}{l}\{L_i^{m+2}(x,\gamma^*,u_1)\}\equiv\{s_i^1(u_1)\}\\ \{s_i^1(u_1)\}=0 \overset{CAD}{\Longrightarrow} u_1=u_1^*.\end{array}$$

Similarly, we now substitute optimizer u_1^* in the level factors $\{L_i^{m+3}(x,\gamma,u_1,u_2)\}_{i=1\dots t_{m+3}}$ to obtain the polynomials $\{s_i^2(u_2)\}_{i=1\dots t_{m+3}}$. By solving them and using the associated CAD information and level factor signs (which information is already available) we obtain the next optimizer u_2^*. Same procedure is followed until we have calculated the optimizer components up to u_n.

It has to be emphasized that the proposed algorithm, when implemented online, only needs to perform the traversal of a tree (see Figure 6) and solve univariate polynomial equations. All other information needed is precomputed offline when the CAD is constructed.

B. Illustrative example

To illustrate the proposed method let us look at the following parametric minimization problem

$$\min_u\ u^4+x_1u^2+x_2u+1, \qquad (7)$$

with $x=[x_1,x_2]$ being the parameter. After the construction of the CAD, we obtain the following level factors

$$\begin{aligned}
L_1^1(x_1) &= x_1\\
L_1^2(x_1,x_2) &= 27x_2^2+8x_1^3\\
L_2^2(x_1,x_2) &= x_2\\
L_1^3(x_1,x_2,\gamma) &= 256\gamma^3+128x_1^2\gamma^2-768\gamma^2+144x_1x_2^2\gamma\\
&\quad +16x_1^4\gamma-256x_1^2\gamma+768\gamma+27x_2^4+4x_1^3x_2^2\\
&\quad -144x_1x_2^2-16x_1^4+128x_1^2-256\\
L_1^4(x_1,x_2,\gamma,u) &= u^4+x_1u^2+x_2u-\gamma+1.
\end{aligned}$$

We note that the last level factor is the input formula of the Boolean expression (4) – the optimization problem is unconstrained. The factor polynomials of the first two levels partition the (x_1,x_2) space as seen in Figure 3.

Let us choose $x_1=-10$ and $x_2=1$. The level factor L_1^1 partitions the x_1 space into two (full-dimensional) cells, namely, $(-\infty,0)$ and $(0,\infty)$. The given value of x_1 belongs in the first cell, which is indexed by 1. The root of $L_2^2(-10,x_2)=0$ is readily $x_2=0$ and that of $L_1^2(-10,x_2)=0$ is $x_2=\pm\frac{3632}{211}$. That means, for the specific value of x_1, the x_2 space is partitioned in four (full-dimensional) cells. As shown in Figure 4, where all the full dimensional cells have been labelled, the given x point lies in cell $(1,5)$, since $0<1<\frac{3632}{211}$.

Once we determined $\mathscr{C}_x$, we obtain its sample point from the CAD. For cell $(1,5)$ it is $(-1,\frac{1}{4})$ and it is marked on Figure 4 with a $\odot$ sign. We can then lift this point up to the (γ,u) space. The level factors partition the (γ,u) space as depicted in the left part of Figure 5. Lifting point (x_1,x_2)

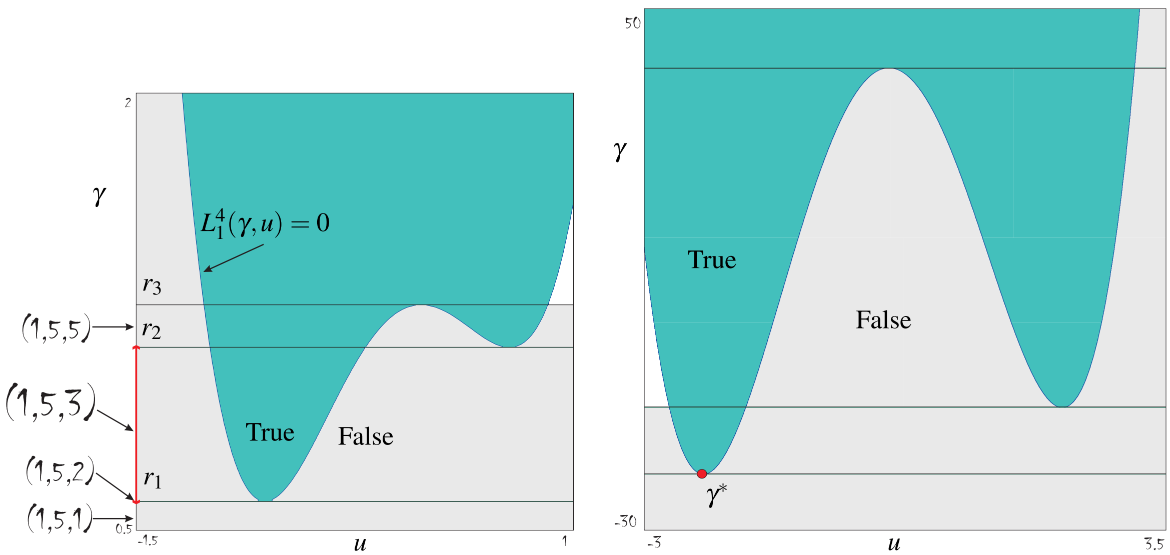

Fig. 5. On the left side, sample point $(-1,\frac{1}{4})$ is lifted to the (γ,u) space. On the right side, point $(-10,1)$ is lifted to the same space. It is easily observed that the two figures are topologically the same. This is because both points $(-1,\frac{1}{4})$ and $(-10,1)$ lie in the same cell $\mathscr{C}_x$.

produces similar results as shown in the right part of Figure 5. We observe that both parts of Figure 5 are topologically the same. This is always the case. The topology of the stack of cells built upon a point x depends only on cell $\mathscr{C}_x$, not on its actual coordinates.

Therefore, the lifting for the sample points of *all* the cells the x space is decomposed into, can be done in advance. From this lifting, we can construct a function M that maps every cell $\mathscr{C}_x$ to the index i_γ that corresponds to the cell labeled $(i_{x_1},\ldots,i_{x_m},i_\gamma)$ that is the "lowest" true cell of $\mathbb{R}^{m+1}$ in the cylinder above x. Here, $i_\gamma = 3$. Cell $(1,5,3)$ is clearly marked on the left part of Figure 5. The function M can be extended to the u -space as well, giving the corresponding index information for the optimizer.

By solving the equation $L_1^3(-10,1,\gamma)=0$ we obtain $\gamma \in \{-26.25,-21.78,1.03\}$. The function M gives us the index $i_\gamma = 3$, therefore we know that γ^* is the *first*, i.e. smaller, root of the three (see Figure 5). Further substituting x and γ^* in L_1^4 and solving equation $L_1^4(-10,1,-26.25,u_1)=0$ gives $\mathscr{U}_1 = \{-2.26\}$. We readily conclude that the optimizer is $u_1^* = -2.26$. It also happens that the algebraic multiplicity of this root is two. This is in agreement with Figure 1, where line $\gamma=\gamma^*$ is *tangent* to the univariate polynomial u^4-10u^2+u+1.

The whole algorithm is in effect the traversal of the cell tree shown in Figure 6 modulo the solution of univariate polynomial equations. This tree is the instance of the more generic "roadmap" the algorithm constructs based on the CAD information. This "roadmap" is used by the algorithm to evaluate maps (2).

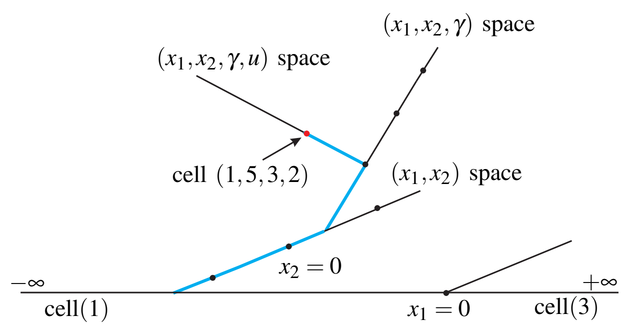

Fig. 6. Traversing the cell tree.

We repeat the above procedure for various values of x_1 and x_2. The optimizer u^* as a function of (x_1,x_2) is shown in Figure 7. We observe that the optimizer is discontinuous along the line $x_2=0$. Such discontinuities are characteristic of nonlinear parametric optimization problems.

IV. Nonlinear MPC

A. Problem formulation

Assume that we have a nonlinear discrete-time system of the form

$$x_{k+1} = f(x_k,u_k) \tag{8}$$

subject to following inequality constraints for $k=0..N$, N being the prediction horizon:

$$g(x_k,u_k) \le 0, \quad k=0..N, \tag{9}$$

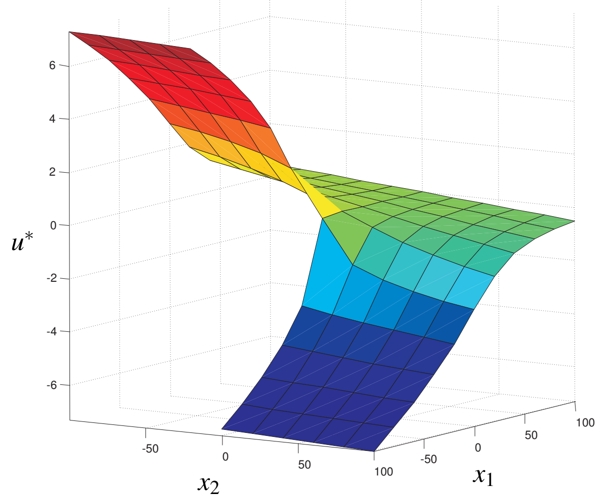

Fig. 7. Optimizer for problem (7).

where $g(\cdot)$ is a multivariate polynomial function in the state and control variables. We consider the problem of regulating system (8) to the origin. For that purpose, we define the following cost function

$$J(U_0^{N-1}, x_0) = \sum_{k=0}^{N-1} L_k(x_k, u_k) + L_N(x_N, u_N) \qquad (10)$$

where $U_0^{N-1} := [u_0, \ldots, u_{N-1}]$ is the optimization vector consisting of all the control inputs for $k = 0..N-1$ and $x_0 = x(0)$ is the initial condition of the system. Therefore, computing the control input boils down to solving the following nonlinear constrained optimization problem

$$\min_u J(U_0^{N-1}, x_0) \quad \text{s.t.} \quad g(x_k, u_k) \leq 0, \quad k = 0, \ldots, N. \qquad (11)$$

For ease of notation we will drop from now on the sub- and superscripts in (11). Problem (11) is written in the more compact form

$$\min_u J(u, x) \qquad \text{s.t.} \quad g(x, u) \leq 0, \qquad (12)$$

where $J(u,x)$ is a polynomial function in u and x, $u \in \mathbb{R}^n$ is the decision variable vector and $x \in \mathbb{R}^m$ is the parameter vector. This is exactly problem (1), a nonlinear parametric optimization problem. Our goal is to obtain the vector of control moves u.

V. Remarks

The main computational bottleneck of the proposed method is the offline construction of the CAD. On the one hand, focusing only on full dimensional cells greatly reduces the computational burden as compared to the general case since we avoid computation using algebraic numbers. On the other hand, the cylindrical algebraic decomposition is intrinsically not efficient in eliminating variables, because it does this one variable at a time. There exist more efficient methods than CAD for computation with semialgebraic sets, such as the *Critical Point Method* [6], which has much better computational properties. Unfortunately there are no currently available implementations of these methods.

In our approach however, once the CAD has been computed, the algorithm needs only perform a tree traversal and solve some univariate polynomial equations. All the information to traverse the tree has already been extracted from the CAD cell structure and level factor signs in each of the cells.

VI. Conclusions and Outlook

The main contribution of this paper is a new algorithm for performing nonlinear constrained parametric optimization of polynomial functions subject to polynomial constraints. The algorithm uses cylindrical algebraic decomposition to evaluate the map from parameter space to the corresponding optimizer and optimal value.

Secondly, the method has been linked to model predictive and optimal control problems. It has to be noted that there exists also the potential of combining dynamic programming with the proposed method to exploit the recursive structure of dynamical systems [7].

Finally, it should be stated that although the proposed algorithm is extremely general and can in principle be applied to a wide variety of problems, its application is limited by the computational cost of the CAD procedure. Unless algorithmic breakthroughs take place or more efficient methods for CAD construction are implemented, the practical relevance of the proposed scheme will be restricted to problems with a relatively small number of variables.

References

[1] S. J. Qin and T. A. Badgwell, "An overview of nonlinear mpc applications," in *Nonlinear Model Predictive Control: Assessment and Future Directions*, F. Allgöwer and A. Zheng, Eds. Birkhauser, 1999.

[2] C. Garcia, D. Prett, and M. Morari, "Model predictive control: theory and practice -a survey," *Automatica*, vol. 25, pp. 335–348, 1989.

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

[4] B. Caviness and J. Johnson, *Quantifier elimination and cylindrical algebraic decomposition*. Wien: Springer Verlag, 1998.

[5] C. W. Brown, "QEPCAD B: a program for computing with semialgebraic sets using CADs," *ACM SIGSAM Bulletin*, vol. 37, pp. 97–108, 2003.

[6] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in real algebraic geometry*. New York: Springer Verlag, 2003.

[7] I. A. Fotiou, P. A. Parrilo, and M. Morari, "Nonlinear parametric optimization using cylindrical algebraic decomposition," Automatic Control Laboratory, Tech. Rep., Mar. 2005.