International Journal of Control
Vol. 00, No. 00, DD Month 200x, 1–25

Parametric Optimization and Optimal Control using Algebraic Geometry Methods

Ioannis A. Fotiou*,[1] , Philipp Rostalski*, Pablo A. Parrilo† and Manfred Morari*

*Automatic Control Laboratory
Swiss Federal Institute of Technology (ETH), 8092 Zurich, Switzerland.

†Laboratory for Information and Decision Systems
Massachusetts Institute of Technology, Cambridge, MA 02139-4307, USA.

(*v2.0 released February 2006*)

We present two algebraic methods to solve the parametric optimization problem that arises in nonlinear model predictive control. We consider constrained discrete-time polynomial systems and the corresponding constrained finite-time optimal control problem. The first method is based on cylindrical algebraic decomposition. The second uses Gröbner bases and the eigenvalue method for solving systems of polynomial equations. Both methods aim at moving most of the computational burden associated with the optimization problem off-line, by pre-computing certain algebraic objects. Then, an on-line algorithm uses this pre-computed information to obtain the solution of the original optimization problem in real time fast and efficiently. Introductory material is provided as appropriate and the algorithms are accompanied by illustrative examples.

1 Introduction

Model predictive control is a very active area of research with broad industrial applications (Qin and Badgwell, 2003). 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) (Garcia et al., 1989).

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 mathematical program, whose solution yields a sequence of control moves. Based on the system model, the system evolution over a finite number of future time steps (the prediction horizon) is predicted and a performance criterion over the same time period is minimized. Out of these control moves only the first is applied, then the prediction horizon is shifted forwards and the procedure is repeated. This is called receding horizon control (RHC). 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, where the system model is linear, 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 mathematical program parametrically (Bemporad et al., 2002). That is, the explicit formula giving the solution of the mathematical program (control inputs) as a function of the problem parameters (measured state) is computed. The optimal control law is then implemented on-line as a look-up table, which greatly increases the computational efficiency of the controller.

[1] Corresponding author. Email: fotiou@control.ee.ethz.ch

International Journal of Control
ISSN 0020-7179 print/ ISSN 1366-5820 online
http://www.tandf.co.uk/journals

In the case of polynomial systems and constraints, the formulation of the MPC problem gives rise to a polynomial optimization problem (polynomial program). However, as opposed to the linear MPC case, a simple closed form expression for its solution may not exist (because it may involve implicit algebraic functions). While this might be the case, we stress the fact that an off-line partial precomputation of the optimal control law is still feasible using algebraic techniques (Fotiou et al., 2006a, 2005, 2006b). The punchline of this approach is that most of the computational burden associated with solving the optimization problem is moved off-line, leaving thus an easier task for the on-line controller implementation.

In this paper, we present the algebraic tools that help us perform polynomial parametric optimization for optimal control problems arising in model predictive control. These include Gröbner bases, cylindrical algebraic decomposition and the eigenvalue method for solving systems of polynomial equations. We focus on the optimization of a polynomial cost function subject to polynomial (semi-algebraic) constraints.

The rest of the paper is structured as follows. In Section 2 we introduce the class of systems and the problem under study. In Section 3 we present the CAD-based algorithm and provide illustrative examples. Subsequently, in Section 4, we introduce Gröbner bases and show how the generalized companion matrices can be employed in parametric optimization. Finally, the paper is concluded and future research directions are outlined.

2 Constrained finite-time optimal control and parametric optimization

In this section we define the class of systems under consideration and describe the constrained optimal control problem arising in model predictive control. Finally, we pose the parametric optimization problem to be subsequently studied.

2.1 *Polynomial dynamical systems*

Consider the class of discrete-time nonlinear systems that can be described as constrained polynomial systems of the following form:

$$x(k+1) = f(x(k), u(k)) \ , \tag{1}$$

subject to the inequality constraints

$$g(u(k), x(k)) \leq 0, \quad k = 1, \ldots, N \ , \tag{2}$$

where $x = [x_1(k), \cdots, x_n(k)]^T \in \mathbb{R}^n$ is the continuous state vector, $u = [u_1(k), \cdots, u_m(k)]^T \in \mathbb{R}^m$ is the continuous input vector, N is the prediction horizon and $f \in \mathbb{R}[x_1, \ldots, x_n, u_1, \ldots, u_m]$ is a real polynomial function in x and u. The real-valued vector polynomial function $g \in \mathbb{R}[x_1, \ldots, x_n, u_1, \ldots, u_m]^q$ defines the constraints of the system in the joint state-input space $\mathbb{R}^{n+m}$. The subset of $\mathbb{R}^{n+m}$ defined by $g(x(k), u(k))$ has the property of being semi-algebraic, the definition of which follows:

Definition 2.1 (Semi-algebraic set) A subset $\mathcal{S}$ of $\mathbb{R}^n$ is semi-algebraic if it can be constructed by finitely many applications of the union, intersection and complementation operations, starting from sets of the form

$$\{x \in \mathbb{R}^n \mid F(x) \leq 0\} \ ,$$

where F is an element of $\mathbb{R}[x_1, \ldots, x_n]$, the ring of real polynomials in n variables.

In other words, a semi-algebraic set can be roughly thought of as one defined by a finite number of polynomial inequalities. Examples of semi-algebraic sets include the interior of an ellipsoid and the set of points of a curve embedded in the n-dimensional Euclidean space. Both can be described by means of polynomial inequalities.

LEMMA 2.2 *A semi-algebraic set $\mathcal{S} \subseteq \mathbb{R}^n$ can be written as a finite union of intersections of basic (semi-algebraic) sets s of the form $\{g(x) \leq 0\}$ and $\{g(x) < 0\}$, where $g \in \mathbb{R}[x_1, \ldots, x_n]$, i.e. there always exist polynomials $\{g_{ij}(x)\}$ such that*

$$\mathcal{S} = \bigcup_{i=1}^{\tilde{n}} \bigcap_{j=1}^{n_i} s_{ij} \ ,$$

where the sets s_{ij} are of the form $\{g_{ij}(x) \leq 0\}$ or $\{g_{ij}(x) < 0\}$.

Proof : It follows immediately from definition 2.1 and the fact that every statement in logic consisting of a finite combination of union, intersection and complementation operations can be written in disjunctive normal form (Chandru and Hooker, 1999). □

2.2 *Constrained finite-time optimal control*

We consider the problem of regulating system (1) to the origin. Assuming that the origin is a stable equilibrium point, we introduce a cost function that penalizes the deviation of the state and control action from zero:

$$J(u, x_0) = L_N(x(N), u(N)) + \sum_{k=0}^{N-1} L_k(x(k), u(k)) \ , \tag{3}$$

where $L_k \in \mathbb{R}[x_1, \ldots, x_n, u_1, \ldots, u_m]$ are polynomial functions in x and u, called *stage costs*, whereas L_N is called *terminal state* cost. The vector $u := [u(0)^T, \ldots, u(N-1)^T]^T \in \mathbb{R}^{mN}$ is the optimization vector consisting of all the decision variables (control inputs) for $k = 0, \ldots, N-1$. The parameter $x_0 = x(0) = [x_1(0), \ldots, x_n(0)]^T \in \mathbb{R}^n$ is the initial state of the system. Obtaining the optimal control moves over the prediction horizon is then equivalent to solving the following constrained finite-time optimal control (CFTOC) problem:

$$\begin{aligned} &\min_u \ J(u, x_0) \\ &\text{s.t.} \ \begin{cases} x(k+1) = f(x(k), u(k)) \\ g(u(k), x(k)) \leq 0, \quad k = 0, \ldots, N \ . \end{cases} \end{aligned} \tag{4}$$

Example 2.3 Consider the linear time invariant system

$$x(k+1) = \frac{1}{2}x(k) + u(k) \ , \tag{5}$$

with $x, u \in \mathbb{R}$ and prediction horizon $N = 2$, subject to the constraints

$$\|x(k+j)\|_\infty \leq 5 \ \ \forall j = 1 \ldots N \ .$$

Define the cost function

$$J(u, x_0) = x(N)^2 + \sum_{k=0}^{1} x(k)^2 + u(k)^2 \ .$$

Using state update equation (5), it can be expressed as a function of the initial state x_0 and the input control sequence $\{u(0), u(1)\}$:

$$J(u, x_0) = 2u(1)^2 + \frac{1}{2}x_0u(1) + u(0)u(1) + \frac{21}{16}x_0^2 + \frac{5}{4}u(0)x_0 + \frac{9}{4}u(0)^2 \ .$$

The resulting CFTOC problem is formulated as

$$\begin{aligned} \min_{\{u(0),u(1)\}} \ & \left\{2u(1)^2 + \tfrac{1}{2}x_0u(1) + u(0)u(1) + \tfrac{21}{16}x_0^2 + \tfrac{5}{4}u(0)x_0 + \tfrac{9}{4}u(0)^2\right\} \\ \text{s.t.} \ & \begin{bmatrix} x(1) - 5 & = & \frac{1}{2}x_0 + u(0) - 5 \\ -x(1) - 5 & = & -\frac{1}{2}x(1) + u(0) - 5 \\ x(2) - 5 = \frac{1}{2}x(1) + u(1) - 5 & = & \frac{1}{2}\left(\frac{1}{2}x_0 + u(0)\right) + u(1) - 5 \\ -x(2) - 5 = -\frac{1}{2}x(1) + u(1) - 5 & = & -\frac{1}{2}\left(\frac{1}{2}x_0 + u(0)\right) + u(1) - 5 \end{bmatrix} \leq 0 \ . \end{aligned} \tag{6}$$

The above optimization problem (6) is an instance of a parametric convex quadratic minimization problem that has been extensively studied in the literature (Bemporad et al., 2002; Grieder et al., 2004). A closed form expression for problem (6) can be computed that gives the optimal solution and corresponding optimizer as a function of the parameter x_0. It turns out that for linear systems with constraints this function is a piecewise affine (PWA) map. For the more general class of polynomial systems we study in this paper, however, such a closed form expression does not always exist. In contrast to the linear MPC case (Bemporad et al., 2002), 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 or eigenvalue computations.

In the following, with a slight abuse of notation, we will rename x_0 to x, because by using state update equation (1) every other $x(k)$ can be substituted as a function of u an x_0. Problem (4) is written in the more compact form

$$\min_u \ J(u, x) \qquad \text{s.t.} \quad g(u, x) \leq 0 \ , \tag{7}$$

where $J(u, x)$ is a polynomial function in u and x, $u \in \mathbb{R}^r$ (with $r = mN$) is the decision variable vector and the initial state $x = x(0) \in \mathbb{R}^n$ is the parameter vector.

2.3 *Parametric optimization*

Let $u \in \mathbb{R}^r$ be the decision-variable vector and $x \in \mathbb{R}^n$ be the parameter vector. The class of optimization problems that this paper deals with can generally assume the following form:

$$\min_u \ J(u, x) \qquad \text{s.t.} \quad g(u, x) \leq 0, \tag{8}$$

where $J(u, x) \in \mathbb{R}[x_1, \ldots, x_n, u_1, \ldots, u_r]$ is the objective function and $g \in \mathbb{R}[x_1, \ldots, x_n, u_1, \ldots, u_r]^q$ is a vector polynomial function representing the constraints of the problem. By parametric optimization, we mean minimizing the function $J(u, x)$ with respect to u for any given value of the parameter x in the region of interest. Therefore, the polynomial parametric optimization problem is finding a computational procedure for evaluating the maps

$$\begin{aligned} u^*(x) &: \mathbb{R}^n \longrightarrow \mathbb{R}^r \\ J^*(x) &: \mathbb{R}^n \longrightarrow \mathbb{R} \ , \end{aligned} \tag{9}$$

where

$$\begin{aligned} u^* &= \arg\min_u J(u,x) \\ J^* &= \min_u J(u,x) \ . \end{aligned} \tag{10}$$

For the sake of simplicity, we assume that the feasible set defined by $g(u,x)$ is compact, and thus the minimum is attained. Also, in order for (9) not to be point-to-set maps, we focus our attention to one (any) optimizer.

3 Cylindrical algebraic decomposition and parametric optimization

In this Section, we first introduce the notion of cylindrical algebraic decomposition (CAD). Because of its intrinsic complexity, we choose an example-driven approach towards illustrating it, rather than giving the full mathematical details - for that, refer to Caviness and Johnson (1998). In the sequel, we show how it can be used to perform parametric optimization and conclude with an example.

3.1 *General description*

Given a finite set $P \subset \mathbb{R}[y_1, \ldots, y_n]$ 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.

To get some insight into the CAD, consider a nonempty subset of $\mathbb{R}^n$. We call such sets *regions*.

Definition 3.1 For any subset X of $\mathbb{R}^n$, a *decomposition* of X is a finite collection of disjoint regions whose union is X.

For a region R, the *cylinder over* R, written $Z(R)$, is the cartesian product $R \times \mathbb{R}^n$. A *section* of $Z(R)$ is a set s of points $(a, f(a))$, where a ranges over R and f is a continuous, real-valued function on R. In other words, s is the graph of f. We call such a graph the *f-section* of $Z(R)$. A *sector* of $Z(R)$ is a set $\hat{s}$ of all points (a,b), where a ranges over R and $f_1(a) < b < f_2(a)$ for continuous, real-valued functions f_1, f_2, with $f_1 < f_2$ for all points in R. The constant functions $f = +\infty$ and $f = -\infty$ are also allowed. Such an $\hat{s}$ is the (f_1, f_2)-sector of $Z(R)$. Continuous real-valued functions $f_1 < f_2 < \cdots < f_k$, $k \geq 0$, defined on R, naturally determine a decomposition of $Z(R)$ consisting of the following regions: (1) the (f_i, f_{i+1})-sectors of $Z(R)$ for $0 \leq i \leq k$, where $f_0 = -\infty$ and $f_{k+1} = +\infty$ and (2) the f_i-sections of $Z(R)$ for $1 \leq i \leq k$. We call such a decomposition a *stack over* R (determined by $f_1, \ldots, f_k$) (Caviness and Johnson, 1998).

Definition 3.2 A decomposition D of $\mathbb{R}^n$ is *cylindrical* if either

(i) $n = 1$ (region R is one dimensional) and D is a stack over $\mathbb{R}^0$, or
(ii) $n > 1$ and there is a cylindrical decomposition D' of $\mathbb{R}^{n-1}$ such that for each region R of D', some subset of D is a stack over R.

Definition 3.3 A decomposition is *algebraic* if each of its regions is a semi-algebraic set.

Clearly, a cylindrical algebraic decomposition is one which is both cylindrical and algebraic. 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 $\{f_i(y) = 0\}$ and inequalities $\{f_j(y) \leq 0\}$ combined using the Boolean operators $\wedge$ (and), $\vee$ (or), and $\rightarrow$ (implies). In general, a *formula* is an expression in the variables $y = (y_1, ..., y_n)$ of the following type:

$$Q_1 y_1 ... Q_n y_n \quad \mathcal{F}(f_1(y), ..., f_\phi(y)) \tag{11}$$

where $\mathcal{F}(f_1(y), ..., f_\phi(y))$ is assumed to be a quantifier-free Boolean formula and Q_i is one of the quantifiers $\forall$ (for all) and $\exists$ (there exists) (Munro, 1999).

Example 3.4

$$\exists\, y_1 \quad \left[\; y_1^2 + a y_1 + b \leq c \;\right]$$

is a quantified formula, whereas the following *equivalent* one is quantifier-free:

$$\left[\; 4c + a^2 \geq 4b \;\right] \;.$$

Tarski showed in (Tarski, 1948) that for every formula including quantifiers there is always an equivalent quantifier-free formula. Obtaining the latter from the former is called *quantifier elimination*. The CAD associated to (11) depends only on its quantifier free part $\mathcal{F}(f_1(y), ..., f_\phi(y))$ and can be used to perform quantifier elimination on (11) (Brown, 2003).

3.2 *Constructing the CAD*

Generally speaking, the construction of the CAD involves three phases. The first, the *projection phase* computes successive sets of polynomials in $n-1, n-2, \ldots, 1$ variables. The main idea is, given the set of polynomials $\{f_i(y)\}$ in (11), in each step $k = 1, \ldots, n-1$ a new set of polynomials $\mathcal{P}_k(f_i(y))$ is obtained by eliminating one variable at a time. That is, the new set of polynomials at every step depends only on $n-k$ variables $\{y_1, y_2, \ldots, y_{n-k}\}$. Of course, the elimination order does matter and a good choice leads to lower computational complexity. The implementation of the projection phase uses sophisticated routines (Collins, McCallum, Hong or Lazard projection operators, for instance) and their description goes beyond the scope of an introductory exposition.

The second phase is the *base phase* and it constructs a decomposition of $\mathbb{R}$, at the lowest level of projection, after all but one variable have been eliminated. The last phase is the *extension phase* where the $\mathbb{R}$-decomposition is successively extended to a decomposition of $\mathbb{R}^2$, $\mathbb{R}^2$ to $\mathbb{R}^3$,...,$\mathbb{R}^{n-1}$ to $\mathbb{R}^n$. In this way, a decomposition of the full $\mathbb{R}^n$-space is obtained.

Additionally, along with every set of polynomials $\mathcal{P}_k(f_i(y))$, the CAD construction algorithm returns a special set of polynomials attached to each projection level d, 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. The level factor polynomials, also called *level factors*, play a central role in parametric optimization.

Remark 1 **Truth evaluation.** Every cell c in every projection level of a CAD has an associated *truth value* $v(c)$. The truth value of a cell is "true" if $\mathcal{F}(f_1(y), ..., f_\phi(y))$ in (11) is true in that cell and "false" otherwise. To determine the truth values of the cells, the CAD algorithm proceeds as follows. If first evaluates the truth value $v(c)$ of the cells in the space of all the variables $\mathbb{R}^n$ by evaluating $\mathcal{F}(f_1(y), ..., f_\phi(y))$ for their corresponding sample points $(s_1, \ldots, s_n) \in \mathbb{R}^n$. Then, it propagates this truth downwards by taking into account the type of quantifiers Q_i's in (11) (Caviness and Johnson (1998), p. 176). This way, the CAD construction algorithm attaches a truth value $v(c)$ to each cell of every projection level. Schematically, the procedure followed is

- Projection phase from R^n to $\mathbb{R}^{n-1}$ ... $\mathbb{R}$,
- Base phase, where sample points for the R-decomposition are computed,
- Extension phase, where sample points for $\mathbb{R}$, $\mathbb{R}^2$, ... $\mathbb{R}^n$ are computed,
- Truth evaluation at uppermost level $\mathbb{R}^n$,
- Truth propagation down to $\mathbb{R}$.

The truth value of the cells play an important role in determining the solution J^* of the optimization problem.

3.3 *A simple CAD*

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

$$\left\{ \begin{array}{r} u^4 - 10u^2 + u + 1 \leq \gamma \\ -7u + 17 \leq -\gamma \end{array} \right\} . \tag{12}$$

The direction of the inequalities in (12) is relevant to the construction of the CAD only as far as the truth value of the cells is concerned. The projection phase has as input a set of polynomials – inequalities are irrelevant. The starting set of polynomials, for this example, is

$$\left\{ \begin{array}{r} u^4 - 10u^2 + u - \gamma + 1 \\ 7u - \gamma - 17 \end{array} \right\} , \tag{13}$$

i.e. the polynomials obtained if we move all the terms in (12) to the left-hand side and ignore the inequality signs. By projecting variable u in (13) first, the level factors obtained for system (12) are

$$\begin{array}{ll} \text{Level 2:} & \begin{array}{l} L_1^2(\gamma, u) = u^4 - 10u^2 + u - \gamma + 1 \\ L_2^2(\gamma, u) = 7u - \gamma - 17 \end{array} \\ \text{Level 1:} & \begin{array}{l} L_1^1(\gamma) = 256\gamma^3 + 12032\gamma^2 + 133728\gamma - 149989 \\ L_2^1(\gamma) = \gamma^4 + 68\gamma^3 + 1244\gamma^2 + 934\gamma - 49857 \, . \end{array} \end{array} \tag{14}$$

Note that the level-two factors are the polynomials as they appear in (13). This is always the case with the last (uppermost) projection level, since no variable has yet been eliminated (projected). We observe that $L_1^1(\gamma)$ is the discriminant $\mathrm{Dis}_u(L_1^2)$ of L_1^2 with respect to u and L_2^1 is the resultant $\mathcal{R}_u(L_1^2, L_2^2)$ of L_1^2 and L_2^2 with respect to u.

The set of real roots of the level-one factors $L_i^1(\gamma)$ in (14) 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).

For this simple low-dimensional example, it is transparent how the intersection and critical points of system (13) are computed – i.e. by means of simple resultant and discriminant operations. As already mentioned, for more complicated settings the projection phase is implemented using special *projection operators* that eliminate one variable at a time. They can be though of as a very systematic way of computing with resultants for doing real elimination (elimination over the field of real numbers $\mathbb{R}$).

Accordingly, the root set of the level-two factors, *together with the one of the first level*, define the boundaries of the cells in the joint (γ, u)-space. The regions in the (γ, u)-space where system (12) is true are the *true cells* of $\mathbb{R}^2$ and are specially marked on Figure 1. In optimization, for a specific choice of parameter x, the union of these regions correspond to the feasible region of the problem.

Remark 2 The CAD construction algorithm has undergone important improvements since its initial conception in 1975 by Collins. One such improvement is the construction of a *partial* CAD instead (Caviness and Johnson (1998), p. 174). The descriptions in this paper are generic and hold for both cases. In practice, however, one would be better off – from a computational point of view – using the partial CAD.

3.4 *Posing the problem*

Suppose we have to solve problem (8). We associate with problem (8) the following boolean expression:

$$(g(u, x) \leq 0) \wedge (\gamma - J(u, x) \geq 0) \, . \tag{15}$$

We then compute the CAD defined by the polynomial expressions in (15). The signs of the polynomials appearing in (15) as well as of $\mathcal{P}_k(f_i(x))$ resulting from the projection steps are determined in each cell. These signs, in turn, determine wether (15) is true or false in each particular cell – see Remark *1.* To perform optimization, it is enough that we calculate the truth value of only the full-dimensional cells in every projection level. In example (12), the full-dimensional cells of $\mathbb{R}^2$ are depicted in Figure 1 as two-dimensional regions. The curves form the one-dimensional cells and the points are the zero-dimensional cells. A point embedded in $\mathbb{R}$, for example, is not full dimensional, since its dimension is zero, less than one, the dimension of the ambient space. On the other hand, a line segment embedded in $\mathbb{R}$ qualifies as full-dimensional. The sample points in the full-dimensional cells can always be chosen as rational numbers (as opposed to algebraic) and the computations associated with them are much easier than in the general case.

All the information we need to solve problem (8) is the level factor polynomials associated with the CAD of system (15), the sample points, and the truth value of the cells.

We now present an algorithm which constitutes an efficient computational procedure to evaluate the map from $x \in \mathbb{R}^n$ to $u^* \in \mathbb{R}^r$ and $J^* \in \mathbb{R}$. The CAD for system (15) associated to the optimization problem is first constructed off-line. Then, an on-line algorithm uses this precomputed information to efficiently evaluate the maps (9), i.e. perform nonlinear parametric optimization.

Note that the variables we deal with are $\{x_1, \ldots, x_n, \gamma, u_1, \ldots, u_r\}$, implicitly appearing in (15). The projection phase of the (off-line) CAD construction algorithm first eliminates u_r, moving from the end of the list to the beginning. An overview of the on-line algorithm is presented in the next Section, followed by an instructive example.

3.5 *The algorithm*

The on-line algorithm consists of three steps:

First step *(Initialization)* We determine the cell in the x-space in which the given parameter x lies. We denote this unique cell with $\mathcal{C}_x \subseteq D$.

Second step *(Finding J^*)* By considering the cylinder over cell $\mathcal{C}_x$ in the joint (x, γ)-space - see also Figure 2 - the optimal cost J^* is determined.

Third step *(Finding u^*)* To determine the optimizer $u^* \in \mathbb{R}^r$ we have to lift the $(x, \gamma) \in \mathbb{R}^{n+1}$ point to the space of the decision variables. This is done by considering the sequence of all stacks above point (x, γ).

It has to be emphasized that the proposed algorithm, when implemented on-line, only needs to perform the traversal of a tree (see Figure 7) and solve univariate polynomial equations. All other information needed is precomputed off-line when the CAD is constructed.

Example 3.5 To illustrate the proposed method let us look at the following parametric minimization problem

$$\min_u \; u^4 + x_1 u^2 + x_2 u + 1 \,, \tag{16}$$

with $x = [x_1, x_2]$ being the parameter. For the sake of clarity, we chose an unconstrained example. The order of elimination is chosen as $x_1 \prec x_2 \prec \gamma \prec u$. That means the projection phase of the CAD construction algorithm first eliminates u, then γ and so on. The corresponding boolean expression associated with problem (16) is

$$\big(\gamma - (u^4 + x_1 u^2 + x_2 u + 1)\big) \geq 0 \,. \tag{17}$$

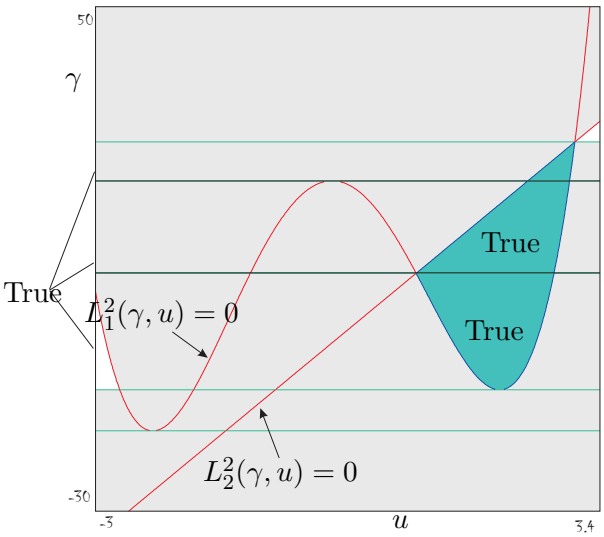

Figure 1. CAD of the polynomials in (12). The (u,γ)-space is decomposed into sections and sectors defined by the corresponding level factor polynomials.

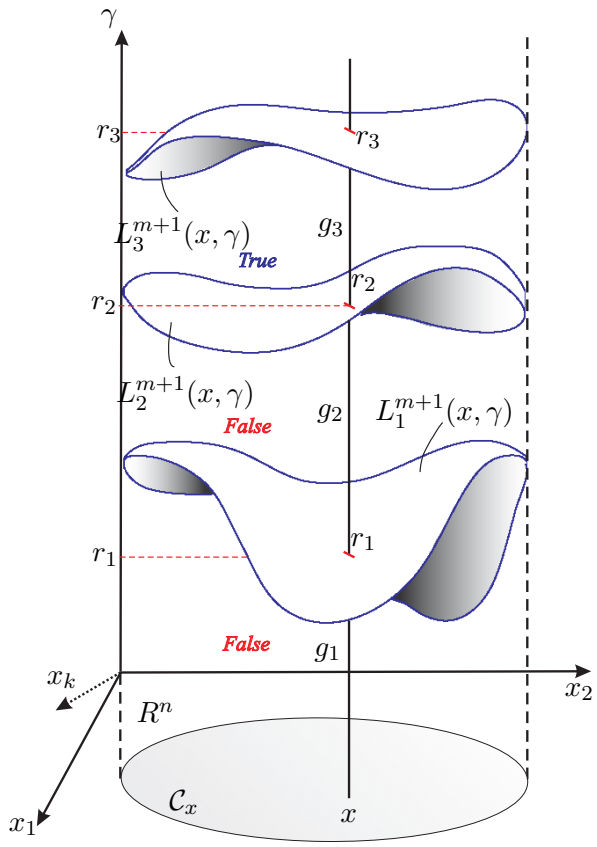

Figure 2. Lifting to the γ space. The optimal cost γ^* lies among roots $\{r_1, r_2, r_3\}$. The Figure is based on Christopher Brown's ISSAC 2004 CAD tutorial slides.

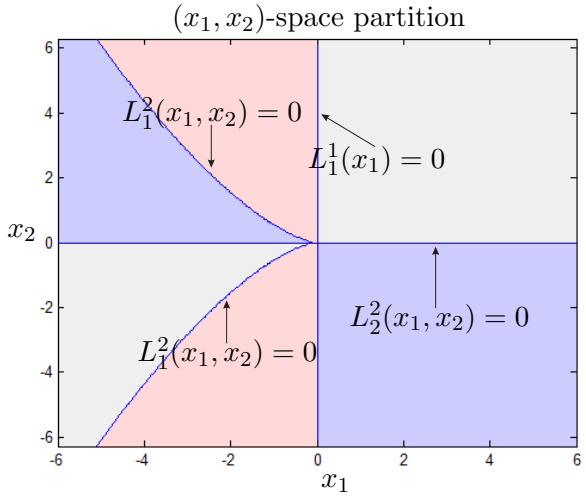

Figure 3. Level factors partitioning the x space of problem (16).

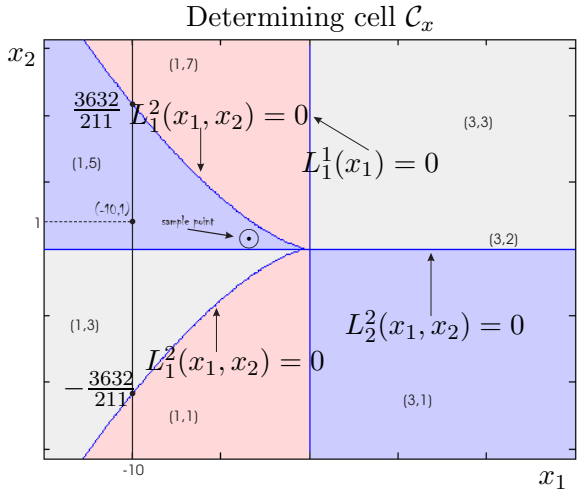

Figure 4. Point $(-10,1)$ lies in cell $(1,5)$. The sample point of this cell is labelled with a $\odot$ sign. The level factor polynomials define the boundaries of the cells.

After the construction of the CAD, we obtain the following level factors:

$$L_1^4(x_1,x_2,\gamma,u) = u^4 + x_1u^2 + x_2u - \gamma + 1$$

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

$$\begin{aligned} L_1^2(x_1,x_2) \quad &= 27x_2^2 + 8x_1^3 \\ L_2^2(x_1,x_2) \quad &= x_2 \end{aligned}$$

$$L_1^1(x_1) \quad = x_1 \ .$$

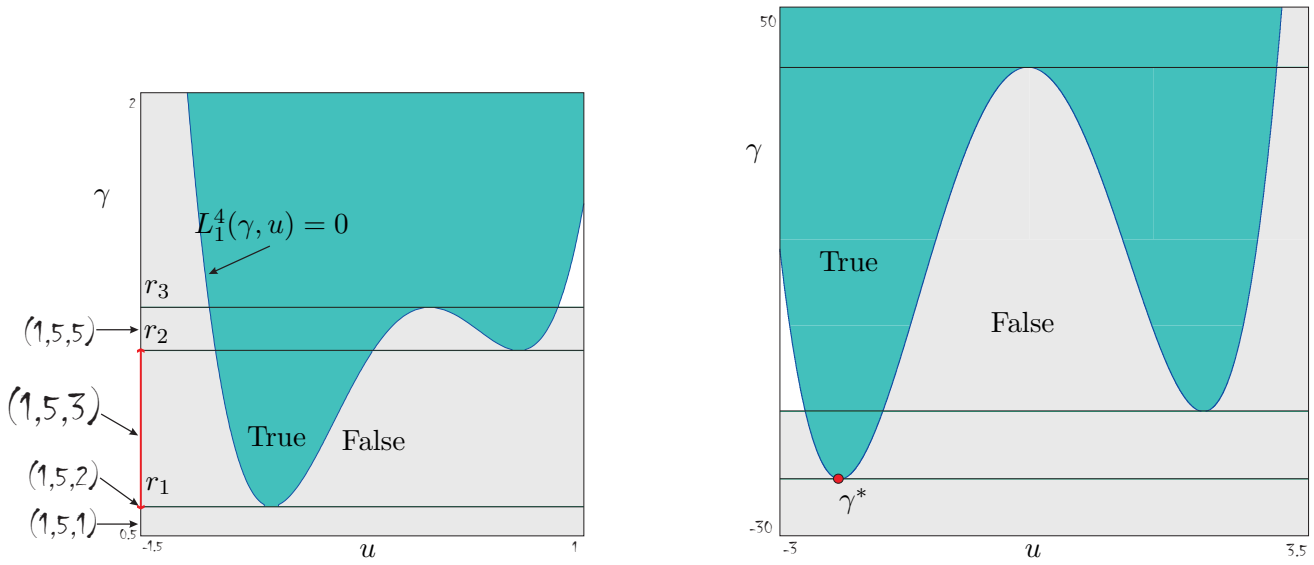

Figure 5. On the left side, sample point $(-1, \frac{1}{4})$ is lifted to the (γ, u)-space. On the right side, point $(-10, 1)$ is lifted to the same space. It is easily observed that the two figures are topologically the same. This is because both points $(-1, \frac{1}{4})$ and $(-10, 1)$ lie in the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

We note that the uppermost level factor is the input formula of the Boolean expression (17). The factor polynomials of the first two levels partition the (x_1, x_2)-space as seen in Figure 3. In the following, the algorithm is presented step by step, for the *particular* choice of $x_1 = -10$ and $x_2 = 1$.

Input: Let us choose $x_1 = -10$ and $x_2 = 1$.

First step: *(Determining $\mathcal{C}_x$)* 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, which are part of the cylinder $Z((-\infty, 0))$. As shown in Figure 4, where all the cells of interest have been labelled with their indexes, the given x point lies in cell $(1, 5)$, because $0 < 1 < \frac{3632}{211}$.

Second step: *(Finding J^*)* We now use the sample points to facilitate the optimal cost computation as follows. Once cell $\mathcal{C}_x$ is determined, its sample point is obtained 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 then consider the cylinder above the cell $\mathcal{C}_x$ in the (γ, u)-space, an operation called *lifting*. The level factors partition the (γ, u)-space into sections and sectors, as depicted in the left part of Figure 5. Lifting point (x_1, x_2) produces similar results as shown in the right part of Figure 5.

We observe that both parts of Figure 5 are topologically equivalent. This is always the case. The topology of the stack of cells built upon a point x depends only on cell $\mathcal{C}_x$, not on its actual coordinates. By topological equivalence, we mean that we can obtain the stack over one point by deforming, twisting and stretching the stack over the other (sample) point. Tearing, however, is not allowed. This property is closely related to a central quality of CAD called *delineability* (Caviness and Johnson, 1998).

Based on this topological equivalence, the lifting for the sample points of *all* the cells the x space is decomposed into, is done off-line. From this operation, we construct a function M that maps every cell $\mathcal{C}_x$ to the index i_γ that corresponds to the cell labelled $(i_{x_1}, \ldots, i_{x_n}, i_\gamma)$, which is the "lowest" true cell of $\mathbb{R}^{n+1}$ in the cylinder above x – see also Figures 2 and 6. We denote this cell with $\mathcal{G}$. If no such cell exists, we set $i_\gamma = -1$ to denote infeasibility of the problem. If the cell exists but is unbounded from below, then for the

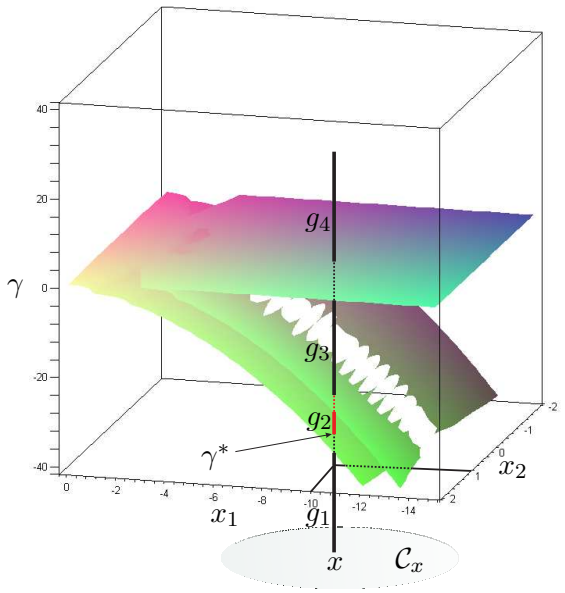

Figure 6. Determining γ^*. Point x is lifted to the (x, γ)-space and the "lowest true" cell $\mathcal{G}$ is identified.

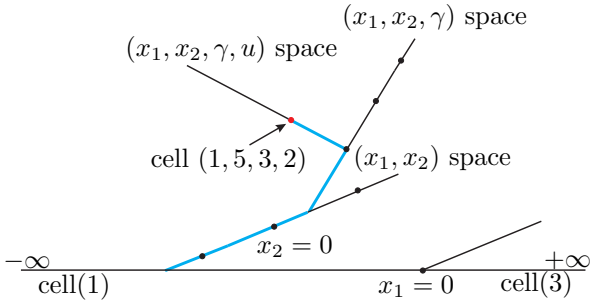

Figure 7. Traversing the cell tree. Once cell $\mathcal{C}_x$ has been identified, function M dictates the way the tree has to be traversed to reach the optimal solution. The indexes in the figure correspond to example (16).

given value of x, the problem is unbounded (from below). In all other cases the minimum is attained and its value J^* is the minimum value that γ takes in the set $\mathcal{G}$, once the value of the parameter x is fixed.

Remark 3 The function M can be extended to the u-space, giving the corresponding index information for the optimizer u^*. Thus, we obtain the following map:

$$M \; : \quad \mathcal{C}_x \longmapsto (i_\gamma, i_{u_1}, \ldots, i_{u_r}) \in \mathbb{N}^{r+1} \; . \tag{18}$$

The indexes i_{u_1} to i_{u_r} are used by the third step of the algorithm, when the optimizer u^* is determined.

Specifically in this example, $i_\gamma = 3$. Cell $(1, 5, 3)$ is clearly marked on the left part of Figure 5. It is also represented in Figure 6, where it is the three-dimensional cell on which segment g_2 lies. By solving the equation $L_1^3(-10, 1, \gamma) = 0$ we obtain three real roots $\gamma \in \{-26.25, -21.78, 1.03\}$. The minimum cost J^* is among these roots. Since the function M gives the index $i_\gamma = 3$, we know that γ^* is the *first* root, namely the smaller one – see also Figure 5.

Third step: *(Finding u^*)* To determine the optimizer $u^* \in \mathbb{R}^r$ we have to lift the $(x, \gamma) \in \mathbb{R}^{n+1}$ point in the space of the decision variables. First, we substitute the values of x and γ^* in the level factors $\{L_i^{n+2}(x, \gamma, u_1)\}_{i=1\ldots t_{n+2}}$. What we obtain is t_{n+2} univariate polynomials in u_1 which we denote by $\{s_i^1(u_1)\}_{i=1\ldots t_{n+2}}$. We solve them to calculate their real roots set $\mathcal{A}_1$. We then use the precomputed CAD cell information together with the level factor signs of each cell, i.e. function M, to determine which root in $\mathcal{A}_1$ corresponds to the optimizer u_1^*. This root will be part of the optimizer vector $u^* = (u_1^*, \ldots, u_r^*)$. This can be formalized as follows:

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

Similarly, we now substitute optimizer u_1^* in the level factors $\{L_i^{n+3}(x, \gamma, u_1, u_2)\}_{i=1\ldots t_{n+3}}$ to obtain the polynomials $\{s_i^2(u_2)\}_{i=1\ldots t_{n+3}}$. By solving them we similarly obtain the next optimizer u_2^*. Same procedure is followed until we have calculated all optimizer elements up to u_r.

Specifically, substituting x and γ^* in L_1^4 and solving equation $L_1^4(-10, 1, -26.25, u_1) = 0$ gives $\mathcal{U}_1 = \{-2.26\}$. 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 $u^4 - 10u^2 + u + 1$. If it happens that we have more than one real roots, then we accordingly use the index information (18) to select the correct one.

Return. Finally, following values are returned: $J^* = -26.25$ and $u^* = -2.26$.

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 8. We observe that the optimizer is discontinuous along the line $x_2 = 0$. Such discontinuities are characteristic of nonlinear parametric optimization problems. A summary of the illustrated approach can be seen in Algorithm *1*.

The off-line computations in this example have been carried out with the help of the software tool QEPCAD B (Brown, 2003).

Algorithm *1* *(On-line)* The CAD level factors, cells and function M have already been computed and are available.

Input: Value of the parameter x (state measurement taken in real time).
Output: Optimal cost J^* and optimizer u^*.

1: Determine cell $\mathcal{C}_x$
2: Specialize parameter x in level factor polynomials $\{L_i^{n+1}\}$ and solve resulting univariate equations to obtain roots $\{r_k\}$
3: using CAD information i_γ select the root that corresponds to the optimal solution J^*
4: **for all** $j = 1, \ldots, r$ **do**
5: specialize x and solve $L_i^{n+1+j} = 0$ to obtain candidate optimizers
6: using CAD index information i_{u_j} select $u^*(j)$
7: **end for**
8: **return:** optimal cost J^* and optimizer $u^* = [u_1^*, \ldots, u_r^*]^T$

Remark 4 The on-line algorithm is in effect the traversal of the cell tree shown in Figure 7 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 (9). Following the same line of reasoning, the function

$$M \; : \quad \mathcal{C}_x \longmapsto (i_\gamma, i_{u_1}, \ldots, i_{u_r}) \in \mathbb{N}^{r+1}$$

segments in effect the continuous space of x into discrete components (the cells), converting the continuous optimization problem into a discrete decision problem plus solving univariate polynomial equations. Once it is determined in which component we start, the indexes $(i_\gamma, i_{u_1}, \ldots, i_{u_r})$ serve as "road signs" towards the optimal solution and optimizer of the original problem (8). This segmentation of the parameter space into regions is in analogy with the polytopic region partition that occurs in the linear off-line MPC case (Bemporad et al., 2002).

Remark 5 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. Nevertheless, current research aims at exploiting the structure of the problem to render the algebraic manipulations in the proposed approach more efficient, thus significantly extending the practical relevance of the algorithm.

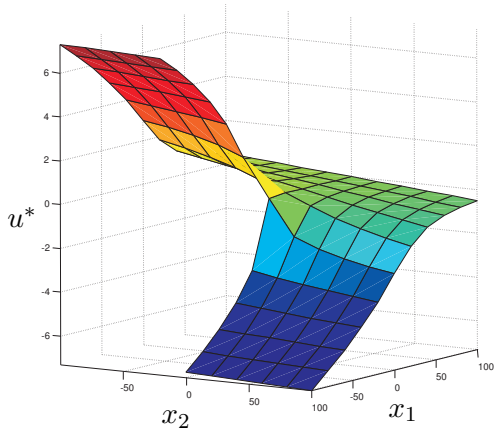

Figure 8. Optimizer for problem (16). Discontinuities with respect to the parameter are characteristic of nonlinear parametric optimization problems.

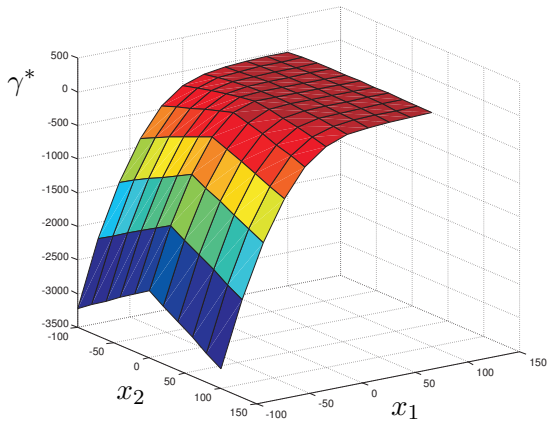

Figure 9. Optimal value for problem (16).

4 The eigenvalue method and parametric optimization

In this section we choose a different angle of attack of the same problem. Firstly, we present some fundamental concepts, namely ideals, varieties, Gröbner bases and related ideas. Secondly, we derive the Karush-Kuhn-Tucker (KKT) optimality conditions for the optimal control problem of a polynomial system. Furthermore, we show how we can use the eigenvalue method for solving systems of polynomial equations to parametrically solve the resulting KKT system and thus perform parametric optimization. The same method has also been used in control for model reduction (Hanzon et al., 1998). Finally, the method is illustrated by means of an application example.

4.1 *Posing the problem*

The new approach is based on the fact that the foundation of continuous constrained optimization are the KKT conditions. In our case, they demand that any local and global minima for problem (8) (satisfying certain constraint qualifications) occur at the so-called "critical points" (Boyd and Vandenberghe, 2004), namely the solution set of the system

$$\begin{aligned} \nabla_u J(u,x) + \textstyle\sum_{i=1}^{q} \mu_i \nabla_u g_i(u,x) &= 0 \\ \mu_i g_i(u,x) &= 0 \\ \mu_i &\geq 0 \\ g(u,x) &\leq 0\ , \end{aligned} \tag{19}$$

where $J(u,x)$ and $g(u,x)$ are as in (8), $g_i(u,x) \in \mathbb{R}[x_1,\ldots,x_n]$ for $i = 1,\ldots,q$ being the polynomial elements of the constraint vector $g(u,x)$. The Lagrange multipliers are denoted in (19) with $\mu_i \in \mathbb{R}_+$ for $i = 1,\ldots,q$. Therefore, performing parametric optimization for optimal control can be reduced to, roughly speaking, solving system (19) parametrically modulo a search over the solution set.

For the class of systems we consider, the two first relations of system (19) form a *square system of polynomial equations*. Various methods have been proposed in the literature for solving systems of polynomial equations, both numerical and symbolic (Sturmfels, 2002), (Parrilo and Sturmfels, 2000), (Manocha, 1994). Related work can also be found in (Nie et al., 2005). Here we consider symbolic methods since our aim is to solve the optimization problem parametrically, i.e. for any given value of the parameter x in the region of interest. It should be mentioned again that the underlying philosophy is moving as much as possible of the computational burden of solving the nonlinear program (8) – and as a consequence KKT system (19)– off-line, leaving an easier task for the on-line implementation.

4.2 *Ideals and varieties*

Consider a system of n polynomial equations f_i in n variables y_i

$$\begin{aligned} f_1(y_1,\ldots,y_n) &= 0 \\ &\vdots \\ f_n(y_1,\ldots,y_n) &= 0 \end{aligned} \tag{20}$$

with coefficients in a field K. A field is a set where we can define addition, substraction, multiplication and division. Furthermore, a field contains additive and multiplicative inverse and identity for all for of its elements. Well known examples are the field of the real numbers $\mathbb{R}$ and the field of complex numbers $\mathbb{C}$. Before we look at the object we are interested in, namely the solution of the polynomial system (20) it is worth describing the concept of *algebraically closed* fields. If we consider the field of complex numbers $\mathbb{C}$ as opposed to the field of real numbers $\mathbb{R}$, we can define equations with coefficients in the field $\mathbb{R}$ that do not have a solution in this field, for instance

$$y^2 + 1 = 0\ . \tag{21}$$

In the field of complex numbers $\mathbb{C}$ this is not possible. The field $\mathbb{C}$ is therefore called algebraically closed. Every field K has an *algebraic closure* which is the smallest algebraically closed field $\overline{K}$ containing K. For example, the algebraic closure of $\mathbb{R}$ is $\mathbb{C}$. The reason for which the statements to follow are posed in the more general framework of any field K (and not specifically $\mathbb{C}$ or $\mathbb{R}$), is the fact that later we have to perform computations in the (more general) field of rational functions in the parameters $x_1 \ldots x_n$.

Since we are interested in the solution of the polynomial system (20), we can state the following question: What are the solution points of (20) in the algebraic closure of the field K? We call this set of points the *variety over* the algebraic closure $\overline{K}$ of K and denote it with

$$\mathcal{V}(I) = \{s \in \overline{K}^n :\ f_1(s) = 0, \ldots, f_n(s) = 0\}\ . \tag{22}$$

In optimal control, we are interested in a subset of these points, say in the real solutions or the solutions satisfying certain constraints $g_i(s) \leq 0$. This is an issue that we address separately – see Section 4.4. If we now relax the conditions on the properties a field has to fulfill, namely if we do not require multiplicative inverse, the resulting algebraic structure is that of a *commutative ring*. The ring consisting of all polynomials $K[y_1,\ldots,y_n]$ with coefficients in the field K is an example of such a ring.

An algebraic object of central importance in polynomial system solving is the notion of polynomial *ideal*. An ideal I is a subset of polynomials in the ring $K[y_1,\ldots,y_n]$, that is $I \subseteq K[y_1,\ldots,y_n]$, where K denotes an arbitrary field:

$$I = \langle f_1,\ldots,f_m\rangle := \{p_1 f_1 + \ldots + p_s f_s\ :\ p_i \in K[y_1,\ldots,y_n] \text{ for } i = 1,\ldots,s\}\ . \tag{23}$$

All common roots of system (20) in the algebraic closure $\overline{K}$, that is all $y \in \mathcal{V}$ are exactly the points we are interested in. The set of points for which the polynomials in (20) vanish, i.e. the variety $\mathcal{V}$, is not uniquely defined by the polynomials f_i. It can be also defined by other sets of polynomials. Consider for example the following system of polynomials:

$$\begin{aligned} f_1 &= y_2 - y_1 y_2 \\ f_2 &= (y_2 - 1)(y_1 + 2) \\ &= y_1 y_2 + 2y_2 - y_1 - 2\ . \end{aligned} \tag{24}$$

The associated ideal is

$$I = \langle f_1, f_2\rangle = \{p_1(y_2 - y_1 y_2) + p_2(y_1 y_2 + 2y_2 - y_1 - 2)\ :\ p_i \in K[y_1, y_2] \text{ for } i = 1, 2\}\ .$$

Another representation of the same set I is given by

$$I = \langle \tilde{f}_1, \tilde{f}_2 \rangle = \{ p_1(-3y_2 + y_1 + 2) + p_2(y_2^2 - y_2) \ : \ p_i \in K[y_1, y_2] \text{ for } i = 1, 2 \} \ .$$

In other words,

$$I = \langle f_1, f_2 \rangle = \langle \tilde{f}_1, \tilde{f}_2 \rangle \ ,$$

where

$$\begin{aligned} \tilde{f}_1 &= -3y_2 + y_1 + 2 \\ \tilde{f}_2 &= y_2^2 - y_2 \ . \end{aligned}$$

We say that both sets of polynomials $\{f_i\}$ and $\{\tilde{f}_i\}$ generate the same ideal. Equivalently, we say that f_i and $\tilde{f}_i$ form a basis of the ideal. Later, we will discuss some particular bases with nice properties (the Gröbner bases). In the example above, the associated variety $\mathcal{V}(I)$ contains two points, namely:

$$\mathcal{V}(I) = \mathcal{V}(\langle f_1, f_2 \rangle) = \mathcal{V}(\langle \tilde{f}_1, \tilde{f}_2 \rangle) = \{ \, (-2, 0), (1, 1) \, \} \ . \tag{25}$$

The *variety of an ideal* has been defined in (22). One could pose the question now, if the *ideal of a variety* can be defined. The answer is affirmative, but one has to pay attention. Consider for example, the set of solution points of the following polynomial system of one equation:

$$f_1 = y_1^2 \ .$$

This set is the same as the set of solution points of system

$$f_1' = y_1 \ .$$

However, the ideal generated by the second system $I' = \langle f_1' \rangle$ is different from the ideal defined by the first system $I = \langle f_1 \rangle$. This is because I' contains the polynomial y_1 whereas I does not. We define the *ideal of a variety* as the *largest* subset of $K[y_1 \dots y_n]$ such that all the polynomials of this subset vanish on the variety. In the example above, the ideal of the variety $\mathcal{V} = \{0\}$ is I', since $I' \supset I$, and we write $I(\mathcal{V}) = I'$. Additionally, we say that an ideal I is *radical* if $I = I(\mathcal{V}(I))$, equivalently if for every $p^l \in I$ also $p \in I$.

4.3 *Gröbner bases*

As seen in the previous Section, the representation of an ideal in terms of a basis is not unique. In order to find a basis with some nice algorithmic properties, we first have to introduce an ordering on the monomials.

An ordering needs to have the following properties:

- *Total ordering:* Every two non-equal monomials can always be compared, i.e. one is always "greater" than the other.
- *Compatibility with multiplication:* The relative order of two monomials remains the same after multiplication with another fixed monomial.
- *Well ordering:* Every strictly decreasing sequence of monomials terminates.

In order to introduce the various term orders, we define the multi-index notation

$$y^\alpha = y_1^{\alpha_1} y_2^{\alpha_2} \dots y_n^{\alpha_n} \ , \text{where } \alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}_{>0}^n \ . \tag{26}$$

The notation in (26) induces also an ordering on the variables y_i, meaning that by $y^{(1,2)}$ we unambiguously understand $y_1y_2^2$ and not $y_1^2y_2$. Furthermore, we call

$$d = |\alpha| = \sum_{i=1}^{n} \alpha_i \tag{27}$$

the total degree of the monomial y^α. A very important monomial term order is the *lexicographic* $\succ_{lex}$ term-order. In this ordering, two monomials $y^\alpha \succ_{lex} y^\beta$ if the leftmost nonzero entry of the exponent vector difference $\alpha - \beta$ is positive. For instance,

$$y_1y_2 \succ_{lex} y_2^3$$

because the difference of the exponent vectors (also called *supports*) has a positive leftmost entry:

$$(1,1) - (0,3) = (\underline{1}, -2) \, .$$

In the sequel, we mainly use a different term-order, the so-called *graded reverse lexicographic order* $\succ_{grevlex}$. We say $y^\alpha \succ_{grevlex} y^\beta$, if $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, or if $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and the leftmost entry of the difference $\alpha - \beta$ is negative. For the example above, we have

$$y_1y_2 \prec_{grevlex} y_2^3 \, ,$$

because

$$|\alpha| = 2 < 3 = |\beta| \, .$$

With the definition of a total ordering of the monomials, we can now define the *initial term* (or leading term) $in_\prec(f) = x^\alpha$ for every polynomial $f \in K[y_1, \ldots, y_n]$ as the largest monomial x^α (with respect to a give term-order). For example, the underlined term in the polynomials

$$\begin{aligned} f_1 &= -\underline{y_1y_2} + y_2 \\ f_2 &= \underline{y_1y_2} + 2y_2 - y_1 - 2 \, , \end{aligned} \tag{28}$$

is the leading term with respect to the graded reverse lexicographic term order. The coefficient of the leading term is called *leading coefficient*. Moreover, for an ideal $I \in K[y_1, \ldots, y_n]$ we can define its initial ideal as

$$in_\prec(I) = \langle in_\prec(f) \rangle \, , \; f \in I \, ,$$

namely the ideal generated by all the initial terms of all the polynomials in the ideal I.

As stated before, the basis of an ideal I is not uniquely defined. Among all possible bases for an ideal I, there is a class of bases with desired algorithmic properties, the Gröbner bases.

Definition 4.1 A *Gröbner basis* is a finite subset $G = \{\gamma_1, \ldots, \gamma_t\}$ of I with the following properties:

(i) it generates the ideal I, i.e. $I = \langle \gamma_1, \ldots, \gamma_t \rangle$, and
(ii) all initial terms of the Gröbner basis generate the initial ideal $in_\prec(I)$ of I, i.e

$$in_\prec(I) = \langle in_\prec(\gamma_1), \ldots, in_\prec(\gamma_t) \rangle \, .$$

For example (28) with $I = \langle -\underline{y_1y_2} + y_2, \underline{y_1y_2} + 2y_2 - y_1 - 2 \rangle$, we can see that $f_1 = -\underline{y_1y_2} + y_2$ and $f_2 = \underline{y_1y_2} + 2y_2 - y_1 - 2$ do not form a Gröbner basis. This is because the second condition of Definition 4.1

is not fulfilled: The leading term of the polynomial $f = f_1 + f_2 = \underline{-y_1} + 3y_2 - 2$ is $-y_1$, that is y_1, belongs to the initial ideal $in_{\prec}(I)$. However, the leading terms of f_1 and f_2 ($-y_2y_1$ and y_2y_1) cannot generate it, because it is not divisible by any one of them. Expressed differently, the leading terms do not generate the initial ideal. The initial ideal of $I = \langle f_1, f_2 \rangle$ for graded reverse lexicographic order contains monomials y_2^2, y_1 and all monomials that are divisible by any one of them. A (graded reverse lexicographic) Gröbner basis for ideal I can then be

$$\begin{aligned} \gamma_1 &= \underline{y_1} - 3y_2 + 2 \\ \gamma_2 &= \underline{y_2^2} - y_2 \ . \end{aligned} \tag{29}$$

Another Gröbner basis for the same ideal is

$$\begin{aligned} \tilde{\gamma}_1 &= \underline{y_1} - 3y_2 + 2 \\ \tilde{\gamma}_2 &= \underline{y_2^2} - y_2 \\ \tilde{\gamma}_2 &= -\underline{y_1 y_2} + y_2 \ , \end{aligned}$$

or any other set of polynomials $\hat{\gamma}_i \in I$ containing γ_1, γ_2. We therefore realize that a Gröbner basis is not unique, nor does it have any minimality requirement that the word "basis" intuitively invokes. However, among all previous instances, it becomes clear that (29) has the property that it minimally generates the corresponding ideal. This property is that the leading coefficients are all one and no trailing term of any γ_i lies in the initial ideal $in_{\prec}(I)$. We call such a basis the *reduced Gröbner basis* and it is unique. Consequently, we have the following definition:

Definition 4.2 A *reduced Gröbner basis* (with respect to a particular term order) is a Gröbner basis $G = \{\gamma_1, \ldots, \gamma_t\}$ that satisfies the following conditions:

(i) The leading coefficients of all elements γ_i are equal to one, and
(ii) No trailing (non-leading) term of any element γ_i lies in the initial ideal $in_{\prec}(I)$.

If we now look at the set $\mathcal{B}$ of monomials not contained in the initial ideal $in_{\prec}(I)$, we come to another interesting structure, the so called *quotient ring* $K[y_1, \ldots, y_n]/I$ of an ideal I. For example (28), the only monomials not contained in $in_{\prec}(I)$, and therefore not divisible by any of the leading terms of the Gröbner basis, are $\mathcal{B} = \{1, y_2\}$. It turns out that any polynomial $f \in K[y_1, \ldots, y_n]$ can be uniquely written in the following form:

$$f = \sum_{i=1}^{l} p_i \gamma_i + \overline{f}^G, \ \gamma_i \in G \ , p_i \in K[y_1, \ldots, y_m] \ , \tag{30}$$

where G is *the* reduced Gröbner basis. The algorithm that takes the polynomial f and writes it in form (30) is the *division algorithm*. The polynomial $\overline{f}^G$ lies in $K[y_1, \ldots, y_n]/I$ and is called the *remainder* of f with respect to the Gröbner basis G, hence the $\overline{(\cdot)}^G$ notation. The remainder $\overline{f}^G$ contains only monomials that are not divisible by any of the leading terms of the γ_i's. Namely, it contains only monomials in $\mathcal{B}$. The monomials in $\mathcal{B}$ build a basis for the quotient ring. Thus, every remainder $r = \overline{f}^G$ can be written as a linear combination of $b_i \in \mathcal{B}$. That is, for

$$\mathcal{B} = \{b_1, \ldots, b_l\} \ , \tag{31}$$

we have

$$r = a^T \cdot b \ , \tag{32}$$

where $b = [b_1, \ldots b_l]^T$ is the vector of the *standard monomials*, the same monomials appearing in basis $\mathcal{B}$.

Definition 4.3 A monomial is *standard* if it is not divisible by any leading monomial of a polynomial in the Gröbner basis.

In the generic case, namely for a square system of polynomials with coefficients random enough, the system of polynomials (20) has a finite number of solutions and consequently, the quotient ring has the structure of a finite-dimensional vector space. In addition, the corresponding ideal is called *zero-dimensional*. Our primary interest is in this finite set of solutions. The quotient ring associated with system (20) can be used to transform the original problem to a linear algebra problem. Namely, we can define certain matrices whose eigenvalues give exactly the solutions of system (20). Let us use again the polynomials in (28) to illustrate this connection.

Suppose that we have the ideal $I = \langle \gamma_1 = \underline{y_1} - 3y_2 + 2, \gamma_2 = \underline{y_2^2} - y_2 \rangle$ generated by the Gröbner basis (29) with $\mathcal{B} = \{1, y_2\}$. After division with this Gröbner basis, the arbitrary polynomial $f = \underline{y_1 y_2^2} + y_1^2 - 4y_1 y_2 + y_1 + 4y_2 + 5$ can be written as

$$f = (y_1 - 1)\gamma_1 + (y_1)\gamma_2 + \overline{f}^G \ ,$$

where

$$\overline{f}^G = y_2 + 7 = [7 \ \ 1] \cdot b \quad (= [7 \ \ 1] \cdot \left[\begin{smallmatrix} 1 \\ y_2 \end{smallmatrix}\right]) \ .$$

A remainder $\overline{f}^G \neq 0$ shows that $f \notin I$. The polynomial f can in turn be multiplied with another polynomial function $h \in K[y_1, \ldots, y_n]$, say $h = y_1 = \gamma_1 + 3y_2 - 2$, and their product expressed as follows:

$$f \cdot h = (y_1^2 - y_1 + y_2 + 7)\gamma_1 + (y_1^2 + 3)\gamma_2 + 22y_2 - 14 \ ,$$

which means that

$$\overline{f \cdot h}^G \left(= \overline{\overline{f} \cdot \overline{h}}^G \right) = [-14 \ \ 22] \cdot \left[\begin{smallmatrix} 1 \\ y_2 \end{smallmatrix}\right] \ .$$

Moreover, we can define a mapping $m_h : \ K[y_1, \ldots, y_n]/I \to K[y_1, \ldots, y_n]/I$ with $m_h(f) = \overline{\overline{f}^G \cdot \overline{h}^G}^G$, that can in turn be applied to any remainder $\overline{f}^G$.

THEOREM 4.4 *The mapping m_h is linear (Cox et al., 1998).*

Because of this, and due to the vector space structure of $K[y_1, \ldots, y_n]/I$, m_h can be represented by a matrix M_h. To compute the matrix M_h, we compute for each element of the basis $b_i \in \mathcal{B}$ the remainder r_i of the polynomial $h \cdot b_i$ with respect to the Gröbner basis G:

$$\overline{h \cdot b_i}^G = r_i, \quad \forall \ b_i \in \mathcal{B} \ . \tag{33}$$

All $r_i \in K[y_1, \ldots, y_n]/I$ can be in turn expressed as an inner product

$$r_i = a_i^T \cdot b$$

with respect to the basis $\mathcal{B}$, as in (32). By collecting all vectors a_i for all basis elements (Cox et al., 1998), we can construct a representation of the map m_h with respect to basis $\mathcal{B}$, i.e. calculate the matrix M_h as follows:

$$M_h \equiv [a_{ij}] = \begin{bmatrix} a_1^T \\ \vdots \\ a_l^T \end{bmatrix} . \tag{34}$$

For our example, we obtain

$$M_h = \begin{pmatrix} -2 & 3 \\ 0 & 1 \end{pmatrix} .$$

The matrix M_h is called the *generalized companion matrix* of the function h.

Evaluating polynomial functions on a variety

Consider a polynomial function $h \in \mathbb{R}[y_1, \ldots, y_n]$. The amazing fact about the matrix M_h is that the set of its eigenvalues is exactly the values of h over the variety $\mathcal{V}(I)$ defined by the ideal I. More precisely, $\mathcal{V}(I)$ is the set of all solution points in complex n-space $\mathbb{C}^n$ of the system (20). The following theorem holds.

THEOREM 4.5 *Let $I \subset \mathbb{C}[y_1, \ldots, y_n]$ be a zero-dimensional ideal, let $h \in \mathbb{C}[y_1, \ldots, y_n]$. Then, for $\lambda \in \mathbb{C}$, the following are equivalent:*

(i) λ is an eigenvalue of the matrix M_h
(ii) λ is a value of the function h on the variety $\mathcal{V}(I)$.

The proof can be found in (Cox et al. (1998), p. 54).

To obtain the coordinates of the solution set of (20), we evaluate the functions

$$\begin{aligned} h_1 &: y \longmapsto y_1 \\ &\vdots \\ h_m &: y \longmapsto y_n \end{aligned} \tag{35}$$

on the variety $\mathcal{V}(I)$ defined by the ideal I, where y above denotes the vector $(y_1, \ldots, y_n)$. This can be done by means of the associated companion matrices of the functions h_i. The following theorem taken from (Sturmfels (2002), p. 22) is the basis for the calculation of these point coordinates.

THEOREM 4.6 *The complex zeros of the ideal I are the vectors of joint eigenvalues of the companion matrices $M_{y_1} \ldots M_{y_n}$, that is*

$$\mathcal{V}(I) = \left\{ (y_1, \ldots, y_n) \in \mathbb{R}^n : \exists\, v \in \mathbb{R}^n \; \forall\, i \; : \; M_{y_i} v = y_i v \right\} .$$

It has to be noted that any vector-valued polynomial function $h : \mathbb{R}^n \longrightarrow \mathbb{R}$ can be evaluated over a zero-dimensional variety in the same way.

4.4 *The algorithm*

In this section, we present an algorithm, which consists of two parts: the off-line part, where the generalized companion matrices for the optimization problem are constructed, and the on-line part where this precomputed information is used and given the value of the parameter x, the optimal solution is efficiently extracted.

Idea

Under certain regularity conditions, if J^* (defined in (10)) exists and occurs at an optimizer u^*, the KKT system (19) holds at u^*. Consequently, J^* is the minimum of $J(u, x)$ over the semi-algebraic set defined by the KKT equations and inequalities (19). These conditions can be separated in a set of inequalities and a square system of polynomial equations. We assume that the ideal generated by the KKT system (19) is zero-dimensional, in order to be able to use the method of eigenvalues for solving systems of

polynomial equations, as described in the previous paragraphs. If we have a positive dimensional variety of critical points, we can always perturb the system in order to get a zero-dimensional variety. The stability and convergence issues related to such perturbations is a research issue on its own right and one should keep it in mind. Recently, (Hanzon and Jibetean, 2003) have proposed such a perturbation scheme with convergence guarantees.

By ignoring the inequalities, a superset of all critical points is computed and in a second step, all infeasible points are removed. It remains to be searched among the feasible candidate points for those with the smallest cost function value.

Off-line Part

We define the KKT ideal

$$I_{KKT} = \langle \nabla_u J(u,x) + \sum_{i=1}^{q} \mu_i \nabla_u g_i(u,x),\ \mu_i g_i(u,x) \rangle \tag{36}$$

containing all the equations within the KKT-system (19). All critical points for the optimization problem (19) and fixed x are the subset of real points on the KKT-variety

$$\mathcal{V}_{KKT}^{\mathbb{R}} \subseteq \mathcal{V}_{KKT} = \mathcal{V}(I_{KKT})\ . \tag{37}$$

These points can be computed by means of generalized companion matrices. We can use Gröbner bases computation for the ideal I_{KKT} and compute the corresponding companion matrices M_{u_i} and M_{μ_i} directly, as shown in (34). Since we want to calculate the companion matrices parametrically, we do all the computations over the field K of rational functions in the parameters $x_1 \dots x_n$, denoted with $\mathbb{R}(x_1 \dots, x_n)$. This means that all coefficients of the polynomials

$$\begin{gathered} f_1(u_1, \dots, u_r, \mu_1, \dots, \mu_q) \\ \vdots \\ f_{r+q}(u_1, \dots, u_r, \mu_1, \dots, \mu_q) \end{gathered}$$

are generally rational functions of the parameter x. Consider following example

$$\begin{aligned} f_1 &= u_2 - x_1 u_2 u_1 \\ f_2 &= (u_2 - x_2)(u_1 + 2)\ , \end{aligned}$$

where $x = [x_1 \ \ x_2]$ is the parameter. The solution set can be easily found to be $\mathcal{V} = \{(-2, 0)\,, (\frac{1}{x_1}, x_2)\}$. However, to illustrate the method, we solve the system also using eigenvalues. The calculation of a Gröbner basis with graded reversed lexicographic ordering $\succ_{gradrev}$ in u_1 and u_2 leads to

$$\begin{aligned} \gamma_1 &= \underline{x_1 x_2 u_1} - 2x_1 u_2 - u_2 + 2x_1 x_2 \\ \gamma_2 &= \underline{u_2^2} - x_2 u_2\ , \end{aligned} \tag{38}$$

for almost all values of the parameters x_1, x_2. Almost all means that (38) is a Gröbner basis, if none of the leading terms vanish, i.e. provided that $x_1 x_2 \neq 0$. For $x_1 x_2 = 0$ we have to calculate a Gröbner basis separately, which leads to

$$\begin{aligned} \gamma_1^{x_1 x_2 = 0} &= u_2 \\ \gamma_2^{x_1 x_2 = 0} &= u_1 + 2\ . \end{aligned}$$

Let us exclude these cases and do the remaining calculations with (38) assuming $x_1 x_2 \neq 0$. The set of standard monomials is given by $\mathcal{B} = \{1, u_2\}$ and the multiplication matrices are

$$M_{u_1} = \begin{pmatrix} -2 & \frac{1+2x_2}{x_1 x_2} \\ 0 & \frac{1}{x_1} \end{pmatrix}, \qquad M_{u_2} = \begin{pmatrix} 0 & 1 \\ 0 & x_2 \end{pmatrix}. \tag{39}$$

In this simple case it is even possible to calculate the eigenvalues parametrically. Generally speaking, however, computing eigenvalues and checking the feasibility of the solution points has to be done after specializing the parameter x. We readily see that the eigenvalue set of matrices (39) are exactly the coordinates of the variety $(-2, 0)$ and $(\frac{1}{x_1}, x_2)$. The matching information is obtained for free, since the companion matrices share the same eigenvectors – see Theorem 4.6. A summary of the off-line algorithm appears in Algorithm *1*.

Algorithm *1* *(Off-line)* Computing the companion matrices

Input: Objective function $J(x, u)$ and constraints $g_i(x, u) \leq 0$.
Output: Generalized companion matrices M_{u_i} and M_{μ_i} for candidate optimizer.

```
construct I_KKT
calc. Gröbner basis G for I_KKT
if G =< 1 > then
    KKT-variety V_KKT is empty
else
    Compute generalized companion matrices M_{u_i} and M_{μ_i} for all variables u_i, μ_i
end if
return: M_{u_i} and M_{μ_i}.
```

Remark 1 Due to the structure of the polynomials in (36), there is an alternative way of calculating the generalized companion matrices. Namely, one could decompose the varieties in sub-varieties and use case enumeration. For clarity of exposition, however, we choose not to elaborate on this approach. The interested reader is referred to Fotiou et al. (2006b) and the references therein.

On-line Part

In order to evaluate the point coordinates of the KKT variety, we need to compute eigenvectors and eigenvalues for the companion matrices. As already mentioned, eigenvalue computation cannot be done parametrically. The parameter x has to be fixed to a numerical value and the computation is done on-line.

Given the precomputed generalized companion matrices M_{u_i} and M_{μ_i}, the online algorithm takes the value of the parameters x to compute the optimum J^* and the optimizer u^*. The three main steps of the algorithm are:

(i) calculate all critical points (by means of eigenvalue computations),
(ii) remove infeasible solutions (e.g. those for which $\mu_i < 0$),
(iii) find the feasible solution u^* with the smallest objective function value $J^* = J(u^*)$.

Since all companion matrices have been computed parametrically, the on-line algorithm mainly involves linear algebra, which can be done fast and efficiently.

First, a set of right eigenvectors $\{v\}$ is computed for the companion matrices M_{u_i} – see Theorem 4.6. All companion matrices commute pairwise, see (Cox et al., 1998), and from linear algebra we know that such matrices all share the same eigenvectors. Therefore, it suffices to calculate the eigenvectors for a single arbitrary matrix or a linear combination of companion matrices. To avoid computational problems, we choose a matrix M_{rand} as a random linear combination of the companion matrices associated with the

decision variables M_{u_i}, i.e.

$$M_{rand} = c_1 M_{u_1} + \cdots + c_r M_{u_r} + c_{r+1} M_{\mu_1} + \cdots + c_{r+q} M_{\mu_q} , \tag{40}$$

where $c_i \in \mathbb{C}$ are randomly chosen scalars. Provided that all the roots of the original system have a multiplicity of one, this ensures with a low probability of failure that the eigenvalues of M_{rand} will all have algebraic multiplicity of one (Cox et al. (1998), Chapter 2, §4). The case of root multiplicities greater than one is studied in Möller and Stetter (1995).

The sets of eigenvectors $\{v\}$ can now be used to compute all candidate critical points u_i^{cand} and their Lagrange multipliers μ_i for the variety $\mathcal{V}_{KKT}$. To avoid unnecessary computations, we first calculate the candidate Lagrange multipliers μ_i. In this way, complex or infeasible candidate points with $\mu_i < 0$ can be immediately discarded before the candidate optimizers u_i^{cand} are computed.

For all non-discarded candidate solutions, it remains to be checked whether they are feasible, i.e. $g(u_i^{cand}, x_i) \leq 0$. To achieve that, a set of feasible local candidate optimizers $\mathcal{S} = \{u_i^{cand}\}$ is initially calculated by collecting all feasible candidate optimizers. After computing the objective function value $J(u_i^{cand}, x)$ for all candidate optimizers, the optimal solution

$$J^* = \min_{u_i^{cand} \in \mathcal{S}} J(u_i^{cand}, x)$$

and the optimizer

$$u_i^* = \arg\min_{u_i^{cand} \in \mathcal{S}} J(u_i^{cand}, x)$$

can be easily obtained. A summary of the on-line algorithm can be seen in Algorithm *2*.

Algorithm *2 On-line Part:* Companion matrices M_u and M_{μ_i} for the variety $\mathcal{V}_{KKT}$ have already been computed and are available.

Input: Value of the parameter x (state measurement taken in real time).
Output: Optimal cost J^* and optimizer u_i.

specialize parameter x in M_{u_i} and M_{μ_i}
calc. a set of common eigenvectors $\{v\}$ for the companion matrix M_{rand}
solve $M_{\mu_i} v = \mu_i v$ to obtain the joint eigenvalues, i.e. candidates for μ_i
discard all eigenvectors with corresp. $\mu_i < 0$
use the remaining eigenvectors to calc. joint-eigenvalues of M_{u_i} to obtain candidates for u_i^{cand}
for all evaluated candidate points $\{u_i^{cand}\}$ **do**
  if $g_k(u_i^{cand}, x) > 0$ **then**
    discard candidate point u_i^{cand}
  else
    evaluate $J(u_i^{cand}, x)$
  end if
  compare $J(u_i^{cand}, x)$ for the calculated u_i^{cand} and choose optimal J^* and corresponding u_i^*
end for
return: optimal cost J^* and optimizer $u^* = [u_1^*, \ldots, u_r^*]^T$.

4.5 *Optimal control application*

In this section, the eigenvalue-based approach is illustrated by means of an application example. The off-line algorithm has been implemented in Maple. Then, a Maple-generated input file is used to initialize Matlab, which runs the on-line algorithm to obtain the optimizer in real time.

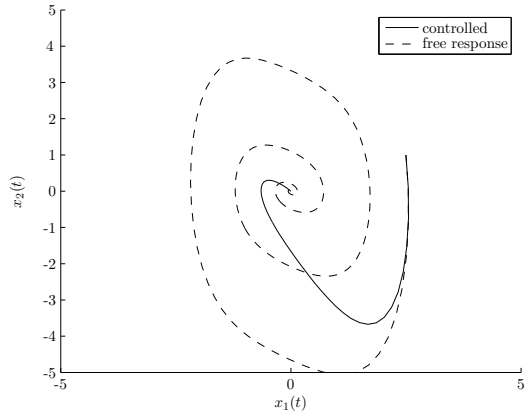

Figure 10. State-space diagram of the Duffing oscillator

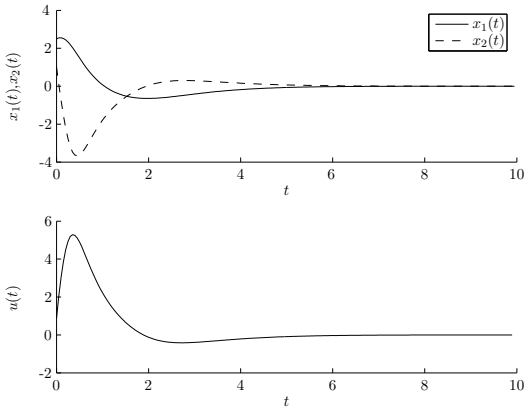

Figure 11. State and input evolution of the controlled Duffing oscillator

Consider the Duffing oscillator (Jordan and Smith, 1987), a nonlinear oscillator of second order. An equation describing it in continuous time is

$$\ddot{y}(t) + 2\zeta\dot{y}(t) + y(t) + y(t)^3 = u(t) \,,$$

where $y \in \mathbb{R}$ is the continuous state variable and $u \in \mathbb{R}$ the control input. The parameter ζ is the damping coefficient and it is known (here $\zeta = 0.3$). The control objective is to regulate the state to the origin. To derive the discrete time model, forward difference approximation is used (with a sampling period of $h = 0.05$ time units). The resulting state space model with a discrete state vector $x \in \mathbb{R}^2$ and input $u \in \mathbb{R}$ is:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & h \\ -h\,(1-2\zeta h) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ h \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ -hx_1^3(k) \end{bmatrix} \,.$$

An optimal control problem with prediction horizon $N = 3$, weight matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \frac{1}{10} \,,$$

and state-constraints

$$\|x(k+j)\|_\infty \leq 5 \;\; \forall j = 1 \ldots 3 \,,$$

leads to the following optimization problem:

$$\begin{aligned} J^* = \min_{u(k),u(k+1),u(k+2)} & \textstyle\sum_{i=1}^{3} \left[\, x_1(k+i)\, x_2(k+i) \,\right] Q \begin{bmatrix} x_1(k+i) \\ x_2(k+i) \end{bmatrix} + \sum_{i=0}^{2} u(k+i) R u(k+i) \\ \text{s.t.} \quad & \|x(k+j)\|_\infty \leq 5 \;\; \forall j = 1 \ldots 3 \,. \end{aligned}$$

Of these twelve constraints there are ten constraints involving $u(k+i)$, which have to be considered during the optimization. The trajectory of the controlled system starting from an initial state of $x_1(0) = 2.5$ and $x_2(0) = 1$ is shown in Figure 11. Figure 10 shows the state-space evolution of the controlled Duffing oscillator and its free response without the controller. In the uncontrolled case, a violation of the constraint $x_2(t) > -5$ can be readily observed. The technical details of the computations in this section can be found in (Fotiou et al., 2006b).

4.6 *Continuous-time systems*

The exposition throughout the paper concerns discrete-time systems. To the best of the authors' knowledge, the proposed algebraic methods cannot be directly applied to continuous-time systems. This is because the continuous-time optimization problem is an infinite-dimensional problem. Nevertheless, an approach for finding the explicit optimal LQ controller for linear time-invariant systems has been proposed in (Kojima and Morari, 2004), where the authors propose an approximation scheme involving singular value decomposition. It is proven, that the computed sub-optimal control laws converge to the optimal one as the problem approximation is refined.

On the one hand, one could argue that most physical systems are continuous-time systems. This fact alone motivates efforts for finding optimal control methodologies that operate on the continuous-time systems directly, which is an interesting research topic in itself. On the other hand, the discrete-time formulation of continuous-time problems lends itself better to computations. When it comes to implementation, however, the approach to be taken is likely to be determined by the application at hand and the respective demands it imposes.

5 Conclusions and future outlook

In this paper we presented two new algebraic methods of nonlinear parametric optimization for the optimal control of polynomial systems. The first method uses cylindrical algebraic decomposition to evaluate the map from parameter space to the corresponding optimizer and optimal value. The second method uses Gröbner bases and the eigenvalue method for solving systems of polynomial equations to perform the same task. Both methods are illustrated with examples. The algorithms presented are very general and can be applied to a broad range of problems. The common theme of the proposed approaches is the precomputation of the solution of the optimal control problem, either by means of CAD or the generalized companion matrices. Thus, a partial pre-solution of the optimization problem is achieved, moving most of the computational burden off-line and leaving an easier task for the on-line, real time receding horizon control implementation.

Further research will explore the possibility of combining algebraic geometry techniques with recently proposed sum-of-squares programming methods, based on semi-definite representations of finite varieties (Laurent, 2004; Parrilo, 2002). Moreover, exploiting the recursive structure of dynamical systems so as to extend the applicability of the proposed algebraic framework to systems with more variables is also under investigation.

6 Acknowledgements

The authors would like to express their gratitude to Prof. Bernd Sturmfels for his time, help and inspiring discussions.

References

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

Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press, Cambridge.

Brown, C. W., 2003. QEPCAD B: a program for computing with semialgebraic sets using CADs. ACM SIGSAM Bulletin 37, 97–108.

Caviness, B., Johnson, J., 1998. Quantifier Elimination and Cylindrical Algebraic Decomposition. Springer Verlag, Wien.

Chandru, V., Hooker, J. N., 1999. Optimization methods for logical inference. Series in discrete mathematics and optimization. John Wiley & Sons, New York.

Cox, D., Little, J., O'Shea, D., 1998. Using Algebraic Geometry. Springer, New York.

Fotiou, I. A., Beccuti, A. G., Papafotiou, G., Morari, M., Mar. 2006a. Optimal control of piece-wise polynomial hybrid systems using cylindrical algebraic decomposition. In: Hybrid Systems: Computation and Control. Santa Barbara, CA.

Fotiou, I. A., Parrilo, P. A., Morari, M., Dec. 2005. Nonlinear parametric optimization using cylindrical algebraic decomposition. In: Proceedings of the ECC-CDC Conference. Seville, Spain.

Fotiou, I. A., Rostalski, P., Sturmfels, B., Morari, M., Jun. 2006b. An algebraic geometry approach to nonlinear parametric optimization in control. In: American Control Conference, Preprint: arXiv:math.OC/0509288. Minneapolis, MN.

Garcia, C. E., Prett, D. M., Morari, M., 1989. Model predictive control: theory and practice - a survey. Automatica 25, 335–348.

Grieder, P., Borrelli, F., Torrisi, F., Morari, M., Apr. 2004. Computation of the constrained infinite time linear quadratic regulator. Automatica 40 (4), 701–708.

Hanzon, B., Jibetean, D., Sep. 2003. Global minimization of a multivariate polynomial using matrix methods. Journal of Global Optimization 27 (1), 1–23.

Hanzon, B., Maciejowski, J., Chou, C., Mar. 1998. Model reduction in H2 using matrix solutions of polynomial equations. Technical Report, CUED/F-INFENG/TR.314.

Jordan, D. W., Smith, P., 1987. Nonlinear Ordinary Differential Equations. Oxford Applied Mathematics and Computer Science. Oxford University Press.

Kojima, A., Morari, M., Jul. 2004. LQ control for constrained continuous-time systems. Automatica 40 (7), 1143–1155.

Laurent, M., 2004. Semidefinite Representations for Finite Varieties. Preprint, 2004. To appear in *Mathematical Programming*.

Manocha, D., 1994. Solving Systems of Polynomial Equations. IEEE Computer Graphics and Applications 14, 46–55.

Möller, H. M., Stetter, H. J., 1995. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. Numer. Math. 70 (3), 311–329.

Munro, N., 1999. Symbolic methods in control system analysis and design. IEE, London.

Nie, J., Demmel, J. W., Sturmfels, B., Mar. 2005. Minimizing polynomials via sum of squares over the gradient ideal. To appear in Mathematical Programming Series A, Preprint: arXiv:math.OC0411342.

Parrilo, P. A., Mar. 2002. An explicit construction of distinguished representations of polynomials nonnegative over finite sets. Tech. rep., Automatic Control Laboratory, ETH Zurich.
URL `http://control.ee.ethz.ch`

Parrilo, P. A., Sturmfels, B., 2000. Minimizing Polynomial Functions. Dimacs Series in Discrete Mathematics and Theoretical Computer Science.

Qin, S. J., Badgwell, T. A., 2003. A survey of industrial model predictive control technology. Control Engineering Practice 11, 733–764.

Sturmfels, B., 2002. Solving Systems of Polynomial Equations. No. 97 in CBMS Regional Conference Series in Mathematics. American Mathematical Society.

Tarski, A., 1948. A decision method for elementary algebra and geometry. University of Califormia Press.