Optimizing a particular real root of a polynomial by a special cylindrical algebraic decomposition

Gandy, Silvia
Tokyo Institute of Technology

Kanno, Masaaki
CREST, Japan Science and Technology Agency

Anai, Hirokazu
Fujitsu Laboratories Ltd |Kyushu University

Yokoyama, Kazuhiro
Rikkyo University

出版情報：MI Preprint Series. 2008-8, 2008-11-10. 九州大学大学院数理学研究院
バージョン：
権利関係：

MI Preprint Series

Kyushu University
The Grobal COE Program
Math-for-Industry Education & Research Hub

Optimizing a particular real root of a polynomial by a special cylindrical algebraic decomposition

S. Gandy,
M. Kanno, H. Anai &
K. Yokoyama

MI 2008-8

(Received November 10, 2008)

Faculty of Mathematics
Kyushu University
Fukuoka, JAPAN

Optimizing a particular real root of a polynomial by a special cylindrical algebraic decomposition

Silvia Gandy, Masaaki Kanno, Hirokazu Anai and Kazuhiro Yokoyama

Abstract. We study the problem of optimizing over parameters a particular real root of a polynomial with parametric coefficients. We propose an efficient symbolic method for solving the optimization problem based on a special cylindrical algebraic decomposition algorithm, which asks for a semi-algebraic decomposition into cells in terms of Number-of-Roots-invariant.

Mathematics Subject Classification (2000). Primary 26C10; Secondary 49N05.

Keywords. optimization of a real root, maximal real root, cylindrical algebraic decomposition (CAD), Number-of-Roots(NoR)-invariant.

1. Introduction

In this work, an algorithm is presented to optimize a particular real root of a polynomial with the help of a special cylindrical algebraic decomposition. This real root optimization problem appears in a wide range of problems.

One example is the $\mathcal{H}_\infty/\mathcal{L}_\infty$-norm computation of parametric dynamical systems, where we want to optimize the $\mathcal{H}_\infty/\mathcal{L}_\infty$-norm of a system matrix $G(s)$, $\|G(s)\|_\infty$, or its frequency-restricted version $\|G(s)\|_{\infty,[\omega_1,\omega_2]}$:

$$\begin{aligned} \|G(s)\|_\infty &:= \sup_{\omega\in\mathbb{R}} \sigma_{\max}\big[G(j\omega)\big], \\ \|G(s)\|_{\infty,[\omega_1,\omega_2]} &:= \sup_{\omega\in[\omega_1,\omega_2]} \sigma_{\max}\big[G(j\omega)\big]. \end{aligned}$$

Here, j is the imaginary unit (i.e., $j=\sqrt{-1}$) and $\sigma_{\max}\big[G(j\omega)\big]$ denotes the maximal singular value of $G(j\omega)$. The maximal singular value is defined as the square root of the maximal eigenvalue of $G^T(-j\omega)G(j\omega)$:

$$\sigma_{\max}\big[G(j\omega)\big] = \text{ maximal real root of } \det\big(\lambda^2 I - G^T(-j\omega)G(j\omega)\big).$$

Thus, we may transform the problem into an optimization problem of the maximal real root of a polynomial.

Another example is the solution of Linear Matrix Inequalities (LMI) [7]. In order to verify the positive-definiteness of a certain symmetric matrix, all eigenvalues need to be positive. Thus, solving

$$X \succ 0 \ , \quad \underbrace{AX + XA^T + BB^T}_{\text{symmetric}} \succ 0,$$

is equivalent to showing that the minimal eigenvalue of $AX + XA^T + BB^T$ is positive for all "parameter values" X satisfying $X \succ 0$.

The $\mathcal{H}_2$ control problem is also part of this problem class. Recently, Kanno et al. [4] used the sum of roots, denoted by σ, to estimate the optimal cost of the $\mathcal{H}_2$ control problem via parametric polynomial spectral factorization. The quantity σ can be represented as the maximal real root of a polynomial with parametric coefficients. Additionally, the cost function of the $\mathcal{H}_2$ optimal control problem is a polynomial in σ. More details will be given in Example 5.5 (Section 5, Computational Results). Therefore, our method is applicable to a wide collection of parametric control problems and possibly to various fields of science and engineering.

The straightforward approach to solve the problem using a numerical approach can fail when the problem is ill-conditioned and will not be able to give an exact solution. Therefore a symbolic approach is of interest. We will present a specialized cylindrical algebraic decomposition (CAD) that avoids the usage of the Sturm-Habicht sequence that is usually being used when transforming the problem setting into a quantifier elimination problem. To this end, a decision criterion, the Number-of-Roots(NoR)-invariant, will be introduced.

The algorithm was implemented in the computer algebra system Maple[1] and then tested against a conventional QE approach using the Sturm-Habicht sequence implemented on QEPCAD B [1].

This paper is organized as follows. In Section 2, we will introduce a mathematical description of the optimization problem under consideration. Then, the CAD algorithm is reviewed, together with its conventional application to solve the quantifier elimination problem. In Section 4, we will present our algorithm and introduce the NoR-invariant. Then, we will discuss examples in Section 5 and compare the computation times of the QEPCAD B and of our Maple implementation. A discussion of the connection of the Sturm-Habicht sequence and real root counting can be found in the Appendix.

2. Problem Setting

In this paper, we study the following optimization problem:

[1]Maple is a registered trade mark of Waterloo Maple Inc. (Maplesoft).

Problem 2.1. We consider a multivariate polynomial f:

$$f(x;\mathbf{q}) \in \mathbb{Q}[x,\mathbf{q}], \quad f : \mathbb{R} \times \mathbb{R}^\ell \mapsto \mathbb{R}.$$

We will interpret f as an univariate polynomial in x being parameterized by the parameter vector $\mathbf{q}$, $\mathbf{q} = (q_1, q_2, \ldots, q_\ell)$. The parameter domain is taken as the ℓ-dimensional interval $D_\mathbf{q} := [\underline{\mathbf{q}}, \overline{\mathbf{q}}] = [\underline{q_1}, \overline{q_1}] \times [\underline{q_2}, \overline{q_2}] \times \ldots \times [\underline{q_\ell}, \overline{q_\ell}] \subset \mathbb{R}^\ell$. However, we need to impose that the vertices of $D_\mathbf{q}$ are rational numbers, $\underline{\mathbf{q}}, \overline{\mathbf{q}} \in \mathbb{Q}^\ell$, to ensure that the roots of f are algebraic numbers for all vertices of $D_\mathbf{q}$. In fact, we can extend our approach without difficulties to a more general case where the parameter domain is a closed connected set that is defined algebraically. However, for the brevity of our discussion and also due to the fact that the ℓ-dimensional interval frequently appear in practical engineering problems, we confine ourselves to such interval.

The task is to determine the range of a particular real root (of f in x), for example, the range of the maximal real root, in dependence of the parameter vector $\mathbf{q}$. The minimal real root or some k-th largest/smallest real root could also be chosen. In particular, the minimal and the maximal value of the chosen real root within the interval $D_\mathbf{q}$ are of special interest.

Now, we will start with the introduction of some notation which will be necessary to identify a certain real root.

First, let us evaluate $f(x;\mathbf{q})$ at a certain parameter vector $\hat{\mathbf{q}} \in D_\mathbf{q}$. Then, the multivariate polynomial f becomes a univariate polynomial, $f_{\hat{\mathbf{q}}}(x) := f(x;\hat{\mathbf{q}})$. This polynomial has a certain degree in x, which we will denote by $n(\hat{\mathbf{q}})$.

So, it has $n(\hat{\mathbf{q}})$ roots $\lambda_1, \ldots, \lambda_{n(\hat{\mathbf{q}})}$ (multiplicities counted), whereof $r(\hat{\mathbf{q}})$ shall be real roots. We can sort the $n(\hat{\mathbf{q}})$ roots of $f_{\hat{\mathbf{q}}}(x)$ in such a way that $\lambda_{\pi(i)} \in \mathbb{R}$ for $i \in \{1, \ldots, r(\hat{\mathbf{q}})\}$ and $\lambda_{\pi(1)} \geq \lambda_{\pi(2)} \geq \cdots \geq \lambda_{\pi(r(\hat{\mathbf{q}}))}$ where π describes the index permutation. We are not interested in the non-real roots of $f_{\hat{\mathbf{q}}}(x)$, which will be labelled as $\lambda_{\pi(r+1)}, \ldots, \lambda_{\pi(n(\hat{\mathbf{q}}))}$.

By $\alpha_k(\mathbf{q}) := \lambda_{\pi(k)}$ $(k = 1, \ldots, n(\mathbf{q}))$, we define a description of the roots point-wisely. Then, we can consider $\alpha_k(\mathbf{q})$ as a representation the k-th largest real root for $1 \leq k \leq r(\mathbf{q})$. FIGURE 1 shows an example for the definition of the functions α_k.

Thus, for every parameter vector $\mathbf{q}$, we obtain the following decomposition of $f(x;\mathbf{q})$:

$$f(x;\mathbf{q}) = \mathrm{lc}(\mathbf{q}) \prod_{i=1}^{n(\mathbf{q})} (x - \alpha_i(\mathbf{q})), \tag{2.1}$$

with $\alpha_i(\mathbf{q}) : D_\mathbf{q} \mapsto \mathbb{C}$ and $\mathrm{lc}(\mathbf{q})$ denotes its leading coefficient ($\mathrm{lc}(\mathbf{q})$ is a piecewise defined polynomial in $\mathbf{q}$).

In the following, we will discuss the efficient computation of the maximal and the minimal value that a certain root can attain in the parameter domain $D_\mathbf{q}$. Therefore, we will choose (fix) a real root, say the k-th largest real root α_k (multiplicities counted), and focus on it. The same approach can be used when

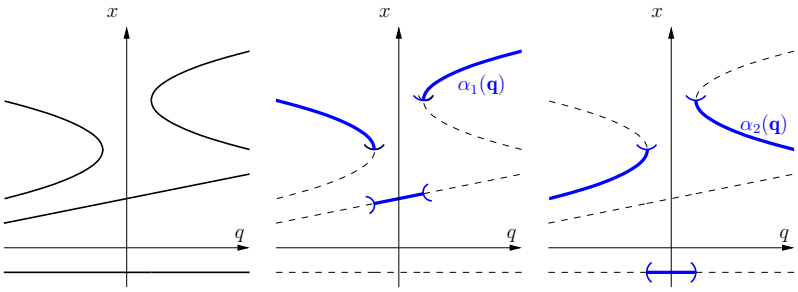


FIGURE 1.
Example showing the point-wise defined functions $\alpha_k(\mathbf{q})$.
Let $f(x;\mathbf{q}) = \frac{1}{5}\left(x^2 - 8x + 17 + \mathbf{q}\right)\left(x^2 - 12x + 37 - \mathbf{q}\right)(x+1)(5x - \mathbf{q} - 10)$.
Left: Trace of $f(x;\mathbf{q}) = 0$.
Middle: Maximal real root, described by α_1.
Right: Second largest real root, described by α_2.

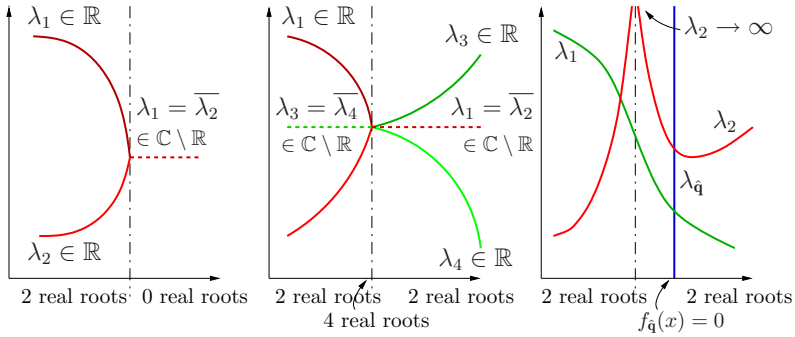


FIGURE 2.
Left: Root behaviour, that does not allow the definition of functions $\alpha_1(\mathbf{q})$ and $\alpha_2(\mathbf{q})$.
Middle: Continuous functions $\alpha_1(\mathbf{q})$ and $\alpha_2(\mathbf{q})$ can be defined, yet $\alpha_3(\mathbf{q})$ and $\alpha_4(\mathbf{q})$ are not continuous/can not be defined.
Right: Root behaviour that does not fulfill Assumptions (1) and (2): a) $\lambda_2 \to \infty$ and b) $\exists \hat{\mathbf{q}} \in D_{\mathbf{q}}$ s.t. $f_{\hat{\mathbf{q}}}(x) = 0$.

focusing on the k-th smallest real root of $f(x;\mathbf{q})$ as this root corresponds to the k-th largest real root of $f(-x;\mathbf{q})$.

Assumptions: For our approach to the calculation of the largest and smallest values of $\alpha_k(\mathbf{q})$, f need to fulfill the following:

(1) $\alpha_k(\mathbf{q})$ is a continuous function of $\mathbf{q}$ in $D_{\mathbf{q}}$, and

(2) $\deg(f(x;\mathbf{q}), x)$ is constant in $D_{\mathbf{q}}$.

The assumption (1) implies that, in the interval $D_{\mathbf{q}}$, no real roots that are larger than $\alpha_k(\mathbf{q})$ become complex (in a locally continuous representation of the roots) and that any complex-conjugated root pair that becomes real results in real roots that are smaller than or equal to $\alpha_k(\mathbf{q})$. FIGURE 2 illustrates some scenarios. On the left, two roots become a conjugated-complex root pair and thus, no continuous functions $\alpha_1(\mathbf{q})$ and $\alpha_2(\mathbf{q})$ can be defined for the entire parameter domain. The middle figure shows what happens when two roots become real at the same point where two other roots become complex. In such a setting, we can define the (continuous) functions $\alpha_1(\mathbf{q})$ and $\alpha_2(\mathbf{q})$, but no continuous $\alpha_3(\mathbf{q})$ and $\alpha_4(\mathbf{q})$ can be defined.
As $D_{\mathbf{q}}$ is a closed connected set, the k-th largest root will only take values in a line segment. Also, the maximum and minimum of $\alpha_k(\mathbf{q})$ are attained on $D_{\mathbf{q}}$, and therefore it is possible to calculate the corresponding parameter vectors $\mathbf{q}$.

The assumption (2) forbids the leading coefficient $\mathrm{lc}(\mathbf{q})$ to become zero in the admissible domain, i.e., $\mathrm{lc}(\mathbf{q}) \neq 0,\ \forall \mathbf{q} \in D_{\mathbf{q}}$. This assumption is imposed because a vanishing leading coefficient corresponds to a drop in the degree of the polynomial $f(x;\mathbf{q})$, which needs to be excluded. So, the polynomial $f(x;\mathbf{q})$ has a constant degree (in x); thus, no root can go to ∞. Is is also not possible for $f_{\hat{\mathbf{q}}}$ to become identically 0 for some $\hat{\mathbf{q}} \in D_{\mathbf{q}}$. (Due to the constant degree assumption, $f(x;\mathbf{q})$ would have to be identically zero for all $\mathbf{q} \in D_{\mathbf{q}}$ then.) This is illustrated in the right part of FIGURE 2.
Using the notations for the real roots of $f(x;\mathbf{q})$ together with the assumptions (1) and (2), we can formalize Problem 2.1 as follows:

Problem 2.2. Let $f(x;\mathbf{q})$ be as in Eqn. (2.1), fulfilling the assumptions (1) and (2). For a given $k \in \mathbb{N}$, determine the largest/smallest value the k-th largest root of $f(x;\mathbf{q})$ can attain and determine the corresponding parameter vector $\mathbf{q}$.

In other words:
Determine $\mathbf{q}_{\max}$, $\mathbf{q}_{\min} \in D_{\mathbf{q}}$ and $\alpha_k(\mathbf{q}_{\max})$, $\alpha_k(\mathbf{q}_{\min}) \in \mathbb{R}$ that fulfill

$$\begin{aligned} \alpha_k(\mathbf{q}_{\max}) &= \max_{\mathbf{q}\in D_{\mathbf{q}}} \alpha_k(\mathbf{q}), \\ \alpha_k(\mathbf{q}_{\min}) &= \min_{\mathbf{q}\in D_{\mathbf{q}}} \alpha_k(\mathbf{q}). \end{aligned}$$

Remark. Due to the assumptions (1) and (2), $\alpha_k(\mathbf{q})$ is a continuous function defined on a closed domain $D_{\mathbf{q}}$. Therefore, the extremal values (minimum and maximum) exist and are attained on $D_{\mathbf{q}}$.

The assumptions seem to impose serious restrictions. However, recall the function from FIGURE 1. In this example, $\alpha_1(\mathbf{q})$ is not continuous, but we can restrict the parameter range $D_{\mathbf{q}}$ such that $\alpha_1(\mathbf{q})$ becomes continuous on the restricted parameter range. Indeed, there is a systematic computation algorithm to decompose the parameter range into cells in each of which $\alpha_1(\mathbf{q})$ is continuous with respect to $\mathbf{q}$. This is in fact Cylindrical Algebraic Decomposition reviewed in Section 4.

The representation as Problem 2.2 will be used by our proposed method. As we want to compare our method to the conventional quantifier elimination (QE) approach, we also state the following problem formulation, which is a direct translation of Problem 2.1:

Problem 2.3 (Formulation as quantifier elimination (QE) problem). Find all values x that fulfill

$$\exists\, \mathbf{q}\, [\ f(x;\mathbf{q}) = 0\ \wedge\ \psi(x;\mathbf{q})\ \wedge\ \phi(\mathbf{q})\ ]\,,$$

where $\psi\,(x;\mathbf{q})$ is a formula that specifies the desired real root of f and $\phi(\mathbf{q})$ is a formulation stating that $\mathbf{q} \in D_{\mathbf{q}}$.

The formula $\psi(x;\mathbf{q})$ is obtained by utilizing the Sturm-Habicht sequence [3] of f and consists of a number of polynomial inequalities in x and $\mathbf{q}$. See appendix A for a brief review of the Sturm-Habicht sequence. The formula $\phi(\mathbf{q})$ is a reformulation of the constraints on the range of q_i, i.e., $q_i \in [\underline{q_i}, \overline{q_i}]$, as inequalities.

By using a software tool for the solution of the QE problem, such as QEPCAD B [1], we can obtain an equivalent description of this problem, where all the quantifiers concerning the parameters q_i have been eliminated. The obtained equivalent formulation is therefore a quantifier-free formula in x that describes the admissible region of x for $\mathbf{q} \in D_{\mathbf{q}}$. Thus, we can determine the maximum and minimum of the chosen real root of f.

If the considered optimization problem 2.1 is formulated as the QE problem 2.3 and solved, then the exact minimum/maximum can be obtained irrespective of the convexity of the optimization problem. From this point of view, an optimization algorithm using QE is a very powerful tool. However, as it is a method that uses symbolic (exact) computation and verifies exhaustively all possibilities derived from algebraic procedures, the computation cost tends to be high. Consequently the size/class of problems that one can realistically solve is extremely limited.

In the sequel, we develop an algebraic approach for Problem 2.2. The approach introduces the notion of *Number-of-Roots-invariant* and avoids the use of the Sturm-Habicht sequence so as to improve the efficiency without sacrificing the exactness of the obtained result. As preparation, we will review in the next section an algebraic tool, the Cylindrical Algebraic Decomposition, which is often used to solve QE problems. We then devise our approach in Section 4 based on the Number-of-Roots criterion.

3. CAD - Cylindrical Algebraic Decomposition (Sign-invariant)

An approach to solve Problem 2.3 is given by cylindrical algebraic decomposition (CAD) [2]. This is a very fundamental concept in quantifier elimination theory. We briefly review CAD in the sequel (See [2] for further details of the CAD).

The algorithm for performing CAD takes as input a set of multivariate polynomials in n variables, say $F \subset \mathbb{R}[y_1, \ldots, y_n]$, and calculates a decomposition of $\mathbb{R}^n$ into components, so called *sign-invariant cells*, over which all the polynomials

in F have constant signs. The algorithm for CAD also provides a point in each cell, called a *sample point*, which can be used to determine the sign of the polynomials in the cell.

3.1. CAD construction

The construction of CAD, in general, consists of three phases: the *projection phase*, the *base phase*, and the *lifting phase*.

In the projection phase, sets of polynomials in $n-1, n-2, \ldots, 1$ variables are computed successively: Given the set of polynomial $F = \{f_i(\mathbf{y})\} \subset \mathbb{R}[y_1, \ldots, y_n]$, compute a new set of polynomials $\mathrm{PROJ}_k(F)$ by eliminating one variable at a time in each step $k = 1, \ldots, n-1$. Consequently, the new set of polynomials at each step lies in $\mathbb{R}[y_1, \ldots, y_{n-k}]$. Some concrete realizations of the projection operator PROJ_k have been proposed, see [10]. Then, the base phase computes a decomposition of $\mathbb{R}$ at the lowest level of projection, namely sample points for a decomposition of $\mathbb{R}$. This can be achieved by using a real root isolation method [12]. Finally, in the lifting phase, the decomposition of $\mathbb{R}$ is first lifted to a decomposition of $\mathbb{R}^2$, and then lifting $\mathbb{R}^2$ to $\mathbb{R}^3$,..., $\mathbb{R}^{n-1}$ to $\mathbb{R}^n$ successively results in a decomposition of the full $\mathbb{R}^n$.

3.2. Solving Problem 2.3 by using CAD

Here, we explain in brief how CAD construction can be utilized for solving Problem 2.3.

In our case, we consider the set $S \subset \mathbb{Q}[x, \mathbf{q}]$ of polynomials appearing in the formula of Problem 2.3. The conventional CAD algorithm determines a decomposition of the $(x, \mathbf{q})$-space into cells where the polynomials appearing in S have constant signs. Thus, a cell boundary describes the zero-set of some polynomial in the set.

In the projection phase, we eliminate parameters $\mathbf{q}$ by successive projections, for example, in the projection order $q_1, q_2, \ldots, q_\ell$. The result is a resultant chain of the following form:[2]

$$(h_0(x; (q_1, \ldots, q_\ell)), h_1(x; (q_2, \ldots, q_\ell)), \ldots, h_{\ell-1}(x; q_\ell), h_\ell(x)), \tag{3.1}$$

where $h_0(x; q) = f(x; \mathbf{q})$, and h_i denotes the result of the i-th projection. The set of real roots of $h_\ell(x)$ is the set X_{cand} of candidate values that includes the extremal values of the desired real root of f. We note that the base phase usually computes sample points for $\mathbb{R}$ (i.e., x-axis), which constits of the candidate points X_{cand} and the intermediate points between adjoining two roots in X_{cand}. A parameter vector $\mathbf{q}$ which realizes a candidate point is provided by the lifting phase. The above resultant chain can be used to lift a candidate x-value to the parameter

[2]This form is only attained in the general case where no resultant $h_{i-1}(x; (q_i, \ldots, q_\ell))$ becomes independant of q_i. In the singular case where some resultant is free from q_i, one can eliminate/ignore the parameter q_i and continue as in the general case. As the resultant does not depend on q_i, there are no restrictions on q_i that can be deduced, therefore it can be choosen freely in $[\underline{q_i}, \overline{q_i}]$.

space. Such a $(x;\mathbf{q})$-tuple will solve the equation $f(x;\mathbf{q}) = h_0(x;\mathbf{q}) = 0$, thereby describing the cell boundaries. The lifting phase takes the following steps:

- Substitute x into Eqn. (3.1). Thus, we obtain

$$\left(h_0^x(q_1,\ldots,q_\ell), h_1^x(q_2,\ldots,q_\ell),\ldots,h_{\ell-1}^x(q_\ell),\text{const.}\right),$$

- Determine a parameter vector $\mathbf{q}$ that solves $f_x(\mathbf{q}) = h_0^x(q1,\ldots,q_\ell) = 0$. Such a solution has the property to be a solution of all equations $h_i^x = 0$ simultaneously. Therefore, we can use the structure of these equations to determine the set of solutions $q_i,\ i = \{\ell,\ell-1,\ldots,1\}$ recursively that solve $h_{i-1}^x(q_i) = 0$, where all q_j with $j > i$ are already fixed and substituted. Thus, the task has been reduced to finding the roots of a set of univariate polynomials instead of the roots of a multivariate polynomial.

4. Algorithms - Specialized CAD (NoR-invariant)

In the following, we will present an approach to reduce the computation cost without losing the exactness of the optimal solution.

For this purpose, we will make use of the fact that we do not focus on all the real roots but only on a particular real root of our interest. In order to be able to take advantage of this, we will need the previously introduced conditions on the behaviour of the real roots.

The approach that we will present solves Problem 2.2 and gives algebraic expressions for $\alpha_k(\mathbf{q}_{\max})$ and $\alpha_k(\mathbf{q}_{\min})$. The main point will be to use a specialized CAD algorithm that uses a Number-of-Root criterion in the lifting phase of CAD construction.

The steps of the algorithm are as follows:

1. **Projection & Base:** Determine a finite set of candidate points X_{cand} that contains the extremal values of $\alpha_k(\mathbf{q})$ (i.e., $X_{\text{cand}} \subset \mathbb{R}$, $\alpha_k(\mathbf{q}_{\max})$, $\alpha_k(\mathbf{q}_{\min}) \in X_{\text{cand}}$).
2. **Partial lifting:** Check from which parameter vector $\mathbf{q}$ the candidate point originates and determine whether the candidate represents an extremum. Avoid checking all candidates.

4.1. Projection & Base

There are two possibilities where an extremum of $f(x;\mathbf{q})$ can be attained:

1. In the interior of the interval $D_{\mathbf{q}}$, $\text{int}(D_{\mathbf{q}})$.
2. For parameter vectors $\mathbf{q}$ on the boundary hyperplanes of the interval $D_{\mathbf{q}}$, $\overline{D_{\mathbf{q}}} \setminus \text{int}(D_{\mathbf{q}})$. (We speak of hyperplanes, although we mean the intersection of the hyperplane with $D_{\mathbf{q}}$.)

The projection function is the usual projection function of the CAD algorithm. However, the inputs for the two cases described above are as follows:

1) The candidate points from the interior of the interval are obtained by projecting $f(x;\mathbf{q})$ iteratively onto the elements of $\mathbf{q} = (q_1, \ldots, q_\ell)$. Thus, after (in general) ℓ projection steps, we arrive at a univariate polynomial in x. The set of real roots of this polynomial is the set of candidate values X_{cand}. Here it is important to note that this operation gives all candidate points as given by $f(x;\mathbf{q})$, without any restriction on the parameters. Therefore, the set of candidate values also contains values that correspond to parameter vectors $\mathbf{q} \notin D_{\mathbf{q}}$. Candidates originating from other roots (irrelevant extrema of real roots other than $\alpha_k(\mathbf{q})$) are included in the set as well.

The set of real roots is represented as a set of isolating intervals of a specified (small) width, thus introducing interval arithmetics. However, once the extremum is identified, the value can be represented as a certain root of the computed univariate polynomial in x, and an algebraic expression is thus obtained.

2) A boundary hyperplane is defined by the condition that one of the parameter vector coefficients $q_i,\ i \in \{1, \ldots, \ell\}$ takes an extremal value $\hat{q}_i \in \{\underline{q_i}, \overline{q_i}\}$. Thus, the parameter space is reduced to $\ell - 1$ dimensional. In order to obtain the candidate points that lie on these hyperplanes, we specialize f to these hyperplanes and define the 2ℓ functions $f_{(i,\hat{q}_i)} := f(x;\mathbf{q})\,|_{q_i=\hat{q}_i}$. The candidate points are the projection results after projecting each $f_{(i,\hat{q}_i)}$ $\ell - 1$ times.

Apart from these points, we evaluate $f(x;\mathbf{q})$ at a particular set of points in order to obtain a good starting point for partial lifting. For this, we evaluate at each extremal $\mathbf{q}$-values $\mathbf{q}_{\text{vertex}}$, i.e., the vertices of $D_{\mathbf{q}}$, $\mathbf{q}_{\text{vertex}} \in Q_{\text{vertex}} := \{\mathbf{q} = (q_1, q_2, \ldots, q_\ell) \mid q_i \in \{\underline{q_i}, \overline{q_i}\},\ \forall i\}$. The vertices are chosen, because, in many cases, the maximal value and the minimal value of $\alpha_k(\mathbf{q})$ are attained at these vertices. Thus, we evaluate at 2^ℓ extra points corresponding to all the 2^ℓ vertices.

4.2. Partial Lifting

We can obtain a set of candidate points $X_{\text{cand}} \subset \mathbb{R}$ as the union of the set of roots of the projection factors and the set of roots of polynomials arising from the vertex points. This candidate point set has the advantageous property that it contains the optimal values $\alpha_k(\mathbf{q}_{\max})$ and $\alpha_k(\mathbf{q}_{\min})$ that are being searched. Therefore, we can limit the search (optimization) on this set of candidate points. In order to find the optimal values efficiently, the idea is to further discard as many candidate points as possible prior to checking them.

As we have determined the values of the real roots at the vertex points, we can determine the maximal (minimal) value of $\alpha_k(\mathbf{q}_{\text{vertex}})$. Then, all candidates smaller (larger) than this value can be discarded. Thus, we can start the search of the optimum from the the maximum (minimum) of the sample points at the vertices and devide the search space $(-\infty, \infty)$ into

$$(-\infty, \min_{\mathbf{q}\in Q_{\text{vertex}}} (\alpha_k(\mathbf{q}))] \quad \text{and} \quad [\max_{\mathbf{q}\in Q_{\text{vertex}}} (\alpha_k(\mathbf{q})), \infty).$$

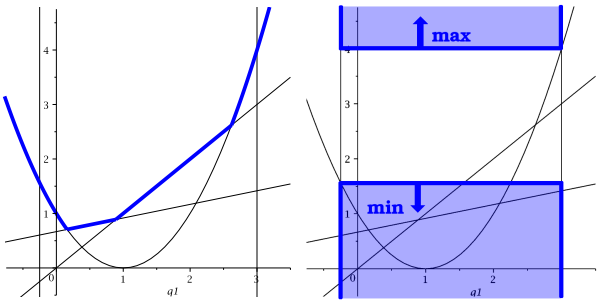


FIGURE 3. Left: Trace of the maximal real root $\alpha_1(\mathbf{q})$.
Right: Reduction of the search space due to the sample points at the vertices of $D_{\mathbf{q}}$.

This division of the search space is shown exemplarily (FIGURE 3) for the calculation of the maximal real root.

Due to the continuity assumption of $\alpha_k(\mathbf{q})$, the selected root can only change within a line segment. Thus, $\alpha_k(\mathbf{q})$ has the range

$$[\alpha_k(\mathbf{q}_{\min}),\ \alpha_k(\mathbf{q}_{\max})]\,.$$

Combining these two range conditions, we obtain

$$\left[\alpha_k(\mathbf{q}_{\min}),\ \min_{\mathbf{q}_{\text{vertex}}}\left(\alpha_k(\mathbf{q})\right)\right] \quad \text{and} \quad \left[\max_{\mathbf{q}_{\text{vertex}}}\left(\alpha_k(\mathbf{q})\right),\ \alpha_k(\mathbf{q}_{\max})\right],$$

for the minimal value and the maximal value of $\alpha_k(\mathbf{q})$, respectively.

So, what we need is a criterion to decide whether a candidate point lies in the inside or on the outside of these intervals. Then, we can start from the known interval boundary and – by subsequently testing the candidate points according to size – determine the other interval boundary. We will use the following approach (compare FIGURE 4):

$$\alpha_k(\mathbf{q}_{\min}) = \min_{x_{\text{cand}}}\left\{x_{\text{cand}} \geq \alpha_k(\mathbf{q}_{\min})\right\},$$

$$\alpha_k(\mathbf{q}_{\max}) = \min_{x_{\text{cand}}}\left\{x_{\text{cand}} \geq \alpha_k(\mathbf{q}_{\max})\right\}.$$

The introduction of the Number-of-Roots(NoR)-invariant in the next subsection will provide such a test criterion. Thus, we can use the following search pattern to obtain the maximal value of $\alpha_k(\mathbf{q})$ in $D_{\mathbf{q}}$:

Search pattern (maximum):

- Check the candidate points x_{cand} in increasing order, starting with the candidate point $\max_{\mathbf{q}\in Q_{\text{vertex}}}\left(\alpha_k(\mathbf{q})\right)$
- Is $x_{\text{cand}} \geq \alpha_k(\mathbf{q}_{\max})$? If yes, stop. The maximal value is found.

The search pattern for the minimum follows the same line of reasoning.

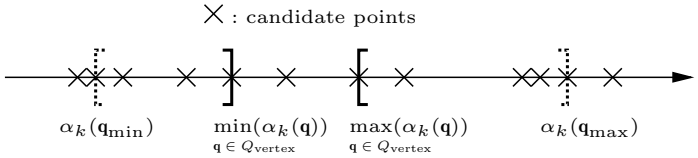


FIGURE 4. Intervals bounded by the searched values $\alpha_k(\mathbf{q}_{\min})$ and $\alpha_k(\mathbf{q}_{\max})$ (dotted interval boundaries).

Search pattern (minimum):

- Check the candidate points x_{cand} in decreasing order, starting with the candidate point $\min_{\mathbf{q}\in Q_{\text{vector}}}(\alpha_k(\mathbf{q}))$
- Is $x_{\text{cand}} < \alpha_k(\mathbf{q}_{\max})$? If yes, stop. The minimal value is found. It is the last valid candidate that was checked in this process.

4.3. Decision Criterion: Number-of-Roots (NoR)

We will now introduce the quantity on which we will base our algorithm. This quantity, which we call the *number of roots*, is invariant on each cell of the decomposition such that its value does not depend on the chosen sample point in each cell.

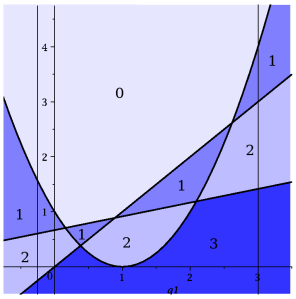
Definition 4.1 (Number-of-Roots, NoR). Let $f(x;\mathbf{q})$ be a multivariate polynomial depending on a parameter vector $\mathbf{q}$ (as in Problem 2.1). When evaluating $f(x;\mathbf{q})$ at some $\hat{\mathbf{q}} \in D_{\mathbf{q}}$, this polynomial only depends on x, i.e., for a fixed $\hat{\mathbf{q}}$, we have that $f_{\hat{\mathbf{q}}}(x) = f(x;\hat{\mathbf{q}})$ is a univariate polynomial in x. This polynomial $f_{\hat{\mathbf{q}}}(x)$ has a certain number of real roots in $\mathbb{R}$.
Define $\text{NoR}(z,\hat{\mathbf{q}})$ to be the number of real roots of $f_{\hat{\mathbf{q}}}(x)$ in the open interval (z,∞), multiplicities of the roots being counted. This definition of $\text{NoR}(z,\hat{\mathbf{q}})$ is valid for each $\hat{\mathbf{q}} \in D_{\mathbf{q}}$. Thus, $\text{NoR}(z,\mathbf{q})$ becomes a function in z and $\mathbf{q}$, being defined point-wisely for all $\mathbf{q} \in D_{\mathbf{q}}$.

In short, $\text{NoR}(z,\mathbf{q})$ gives the number of real roots of the polynomial $f_{\mathbf{q}}(x)$ that are larger that z.

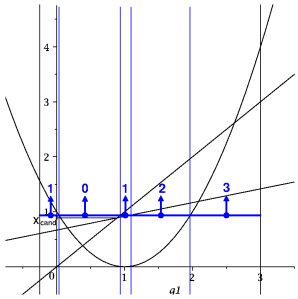
Example:
Consider $f(x;\mathbf{q}) = (x-1)(x-q_1)^2(x+q_1q_2)(x^2+1)$ and fix $\mathbf{q} = (q_1,q_2) = (2,3)$. Then, $f_{\mathbf{q}}(x)$ evaluates to $f_{(2,3)}(x) = (x-1)(x-2)^2(x+6)(x^2+1)$. Thus,

$$\text{NoR}(z,(2,3)) = \begin{cases} 4 & z \in (-\infty,-6] \\ 3 & z \in (-6,1] \\ 2 & z \in (1,2] \\ 0 & z \in (2,\infty) \end{cases}$$

$\text{NoR}(z,\mathbf{q})$ can be determined via the evaluation of the sign variations of the Sturm sequence [11] of $f_{\mathbf{q}}(x)$. Note that a minor change is necessary to accommodate the counting of the multiplicity of a root.



NoR in the different cells



Check of $\mathrm{NoR}(x_{\mathrm{cand}} + \epsilon, \mathbf{q}_{\mathrm{sp}})$, for each cell sample point $\mathbf{q}_{\mathrm{sp}}$

FIGURE 5. Cell decomposition showing $\mathrm{NoR}(z, \mathbf{q})$ in each cell ($f(x; \mathbf{q})$ is taken from Example 5.1 in Section 5)

For every cell of the decomposition, we can determine the number of roots that lie at higher values of x. The left part of FIGURE 5 illustrates this. The lines correspond to the traces of the roots of the polynomial $f(x; \mathbf{q})$ under consideration. The cells of the decomposition are bounded by these traces and contain the number of roots that lie above the cell.
Due to the continuity assumption of $\alpha_k(\mathbf{q})$, we will be able to show that the decision is independent of the sample point. Therefore, we need to show why $\mathrm{NoR}(z, \mathbf{q})$ cannot change within a cell and how it is related to $\alpha_k(\mathbf{q})$.

Lemma 4.2. *If $\alpha_k(\mathbf{q})$ is continuous in $D_{\mathbf{q}}$, then $\alpha_i(\mathbf{q})$ is continuous in $D_{\mathbf{q}}$ for $i \in \{1, 2, \ldots, k-1\}$.*

This is due to $f(x; \mathbf{q})$ being a multivariate polynomial in x and $\mathbf{q}$. Note: a change in the NoR-value within a cell occurs, when two roots become complex/real.

Lemma 4.3. *For $\mathbf{q} \in D_{\mathbf{q}}$, we immediately get from the definition of $\alpha_k(\mathbf{q})$ that*

$$\mathrm{NoR}(z, \mathbf{q}) \geq k, \quad \forall z < \alpha_k(\mathbf{q}).$$

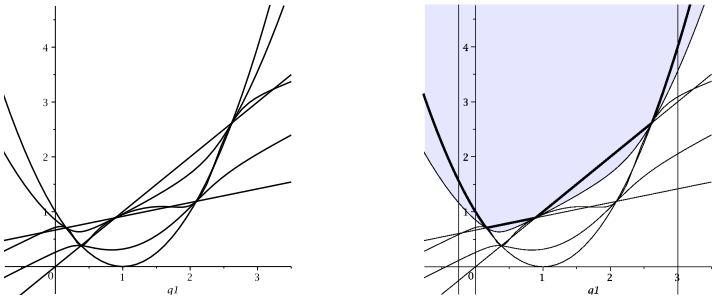
Also,

$$\alpha_k(\mathbf{q}_{\max}) = \min\{z \mid \mathrm{NoR}(z, \mathbf{q}) < k \ \forall \mathbf{q} \in D_{\mathbf{q}}\}. \tag{4.1}$$

Therefore, the first candidate x_{cand} which is contained in the right hand side of Eqn. (4.1) corresponds to the maximum that is being searched, i.e., $x_{\mathrm{cand}} = \alpha_k(\mathbf{q}_{\max})$. Remember that the candidate points are checked in increasing order.

An example for the evaluation of $\mathrm{NoR}(z, \mathbf{q})$ (in the case of the computation of the minimum, $k = 1$) is shown in the right part of FIGURE 5.

This concludes the description of our proposed NoR-based special cylindrical algebraic decomposition. Using the described NoR criterion to check the individual candidate points during the lifting, we obtain an algorithm that does not need the conditions of the Sturm-Habicht sequence.



SH conditions added Extraction of the maximal real root

FIGURE 6. Left: Trace of the input polynomial and the Sturm-Habicht (SH) conditions of Example 5.1 (Sect. 5).
Right: The shaded area (originating from the SH-conditions) extracts the maximal real root.

4.4. Advantage over the conventional QE approach

The proposed algorithm (NoR criterion based) uses as input only the polynomial $f(x;\mathbf{q})$ to construct the projection factors that lead to the set of candidate points. Thereby, the algorithm constructs a (sub-)set of the cells (regions) shown in the left part of FIGURE 5. The direct QE-approach, however, employs a set of additional polynomial conditions to identify the root of interest. These conditions originate from the Sturm-Habicht sequence of input polynomial f. An outline of the Sturm-Habicht sequence can be found in the Appendix. In short, the number of polynomial conditions increases, because we need to identify the chosen real root. As a consequence, the number of candidate points will increase considerably. To give an example, see FIGURE 6, which derives from the same input polynomial as the example in FIGURE 5 (input polynomial as in Example 5.1 of Section 5). The largest real root α_1 is chosen in this example. On the left side, you see the input polynomial together with the additional conditions, dividing the plane into a far greater number of cells than in FIGURE 5. On the right part of the figure, the area restrictions from the Sturm-Habicht conditions are coloured. The only part of the root traces that lies in this area is exactly the maximal real root α_1. This shows examplarily the increase in candidate points due to the increase of cells in the decomposition of the parameter vector space.

The main advantage of our proposed method is to solve the problem without increasing the number of conditions (polynomials) prior to the projection phase. The additional computation to check the individual candidates is by far outweight by this advantage. The examples computed in Section 5 support this asserton.

5. Computational Results

In order to confirm the improvement of the aforementioned specialization of the CAD, we developed a Maple-based prototype program implementing our approach

as described in Section 4. By comparing the computation times of this prototype to those resulting from an existing QEPCAD B implementation, we can confirm that the computation time is reduced significantly, as soon as the problem reaches a size where implementation issues can be ignored.

The computation times for a set of 6 examples were measured. The computations were executed on a personal computer equipped with an Intel Core Solo CPU U1500, 1.33 GHz processor, with 2 GB of memory. The existing implementation uses a Linux based QEPCAD B (ver. 1.50), whereas the prototype uses a Linux based Maple 9.5.

The computation times in TABLE 1 are the times spent to calculate the smallest and the largest value of the specified real root of the polynomial $f_k(x;\mathbf{q})$ in $D_{\mathbf{q}}$. The input of the QEPCAD B program consisted of three parts, the condition $f_k(x;\mathbf{q}) = 0$, the conditions $\psi(x;\mathbf{q})$ coming from the Sturm-Habicht sequence, and the conditions $\phi_i(q_i)$ indicating the domains of the q_i's. The input of the prototype consisted only the two condition sets $f_k(x;\mathbf{q}) = 0$ and $\phi_i(q_i)$.

The QEPCAD B input parameter "N" that controls the memory allocation in the initialization process of the program was chosen as the minimal value where the memory suffices and no garbage collection occurs during computation. We used as measured computation time the "System time after the initialization" that appears in the QEPCAD B output.

We will now present the examples we used, Examples 5.1-5.6. Note that the examples were not used in factorized form when used as input to the program, although being presented in factorized form below to facilitate understanding.

Example 5.1. The first example is a simple example of a polynomial containing only one parameter q_1:

$$f_1(x;q_1) := (x-q_1)(x-(q_1-1)^2)(x-(\frac{q_1}{4}+\frac{2}{3}))\ , \quad q_1 \in [-1,3].$$

From the factorization of $f_1(x;q_1)$, it is evident that the 3 roots have the values

$$\lambda_1 = q_1,\ \lambda_2 = q_1^2,\ \lambda_3 = \frac{q_1}{4}+\frac{2}{3}.$$

The maximal real root takes values in the interval

$$\left[\frac{1}{96}(91-\sqrt{537}),4\right] \approx [0.7065,4].$$

The minimum is attained for $q_{1\min} = \frac{9}{8} - \frac{1}{24}\sqrt{537} \approx 0.159$ and the maximum for $q_{1\max} = -1$.

The values for x and q_1 are algebraic numbers, so we represent them internally by intervals. These intervals are obtained as the isolating intervals of the real roots of a defining polynomial. The tolerance, i.e., the size of the isolating intervals, can be adjusted and was set to 2^{-27} in the example runs. In the discussion of the examples, we will only give the decimal representation of the results as the defining polynomials would take up too much space.

The computation time was 116 ms for the QEPCAD B program and 180 ms for our Maple program. The QEPCAD B program is faster, but as it is a compiled program (whereas Maple uses an interpreted language), we should regard the result as the consequence of implementation issues rather than the complexities of the algorithms.

Example 5.2. The next example is a more complicated version of Example 5.1. We introduce an additional parameter q_2 and added its square to each root of $f_1(x; q_1)$. So, we define $f_2(x; (q_1, q_2)) := f_1(x - q_2^2; q1)$, which factorizes as

$$f_2(x; (q_1, q_2)) = (x - (q_1 + q_2^2))(x - ((q_1 - 1)^2 + q_2^2))(x - (\frac{q_1}{4} + \frac{2}{3} + q_2^2)) \ .$$

We consider the following domain:

$$q_1 \in [-1, 3], \ q_2 \in [-1, 1].$$

The maximal real root takes values in the interval $[0.7065, 5]$. The minimum is attained for $(q_1, q_2)_{\min} = (0.159, 0)$, which lies in the interior of $D_{(q_1,q_2)}$. The maximum is attained for $(q_1, q_2)_{\max} = (-1, -1)$, a vertex point of $D_{(q_1,q_2)}$.

The computation time for this example was 5.036 s for the QEPCAD B program vs. 4.384 s for our Maple implementation. This time, we can observe the positive effect of our approach. On the one hand, the usage of the NoR criterion makes it unnecessary to project any additional polynomials due to the Sturm-Habicht sequence. On the other hand, the maximum is attained at a vertex point, which enables us to efficiently use partial lifting.

Example 5.3. In the second example, Example 5.2, the roots of x depended on the square of q_2. In this example, we change the parametrization by substituting β for q_2^2. By inserting this change, we will see the strong dependency of the computation time w.r.t. the parametrization:

$$\begin{aligned} f_3(x; (q_1, \beta)) \quad &:= \quad (x - (q_1 + \beta))(x - ((q_1 - 1)^2 + \beta))(x - (\frac{q_1}{4} + \frac{2}{3} + \beta)) \ , \\ &\quad\ \ q_1 \in [-1, 3], \ \beta \in [0, 1] \end{aligned}$$

The behaviour of the maximal real root does not change due to the change of parametrization. It still takes values in the interval $[0.7065, 5]$. The minimum is now attained for $(q_1, \beta)_{\min} = (0.159, 0)$ which lies on the boundary hyperplane, whereas the maximum is attained for $(q_1, \beta)_{\max} = (-1, 1)$, a vertex point of $D_{(q_1,\beta)}$.

This change of parametrization greatly affects the computation times. Two effects can be observed. First, the degree of f_3 in the second parameter changed from degree 2 (q_2^2) to degree 1 (β). Therefore, the projection and base phase contains polynomials of lower degree, speeding up the computation. Second, the size of the parameter domain decreases and the optimal point moves from the interior of $D_{(q_1,q_2)}$ to the boundary of $D_{(q_1,\beta)}$. In this particular example, the number of candidate points that are checked for the minimum thereby decreased and the partial lifting is very efficient.

The computation time for this example was 4.816 s for the QEPCAD B program, whereas we measured 1.220 s for our Maple implementation. So, an efficient parametrization can make a great difference for the computations. In this setting, our prototype finishes after one fourth of the time spent by the QEPCAD B program.

Example 5.4 (Computation of the second largest root α_2). The next example is an artificially constructed problem, which contains 3 parameters and a varying leading coefficient, $\mathrm{lc}(\mathbf{q}) = (q_1 - 10)$. Essentially, it is identical to Example 5.3 with two additional roots and a varying leading coefficient. The leading coefficient becomes zero at $q_1 = 10$ which lies outside of the parameter domain $D_{(q_1,\beta,\gamma)}$.

We will consider the problem of calculating the second largest real root of $f_4(x; (q_1, \beta, \gamma))$. One of the two additional roots is a constant root $x = 6$, which is larger than all real roots of f_3. The other root we added, γ, does not interfere with the largest real root of f_3. Thus, the range of the second largest root of f_4 equals the range of the maximal real root of f_3:

$$\begin{aligned} f_4(x; (q_1, \beta, \gamma)) \quad := \quad & (q_1 - 10)\,(x - 6)\,(x - \gamma)(x - (q_1 + \beta)) \\ & \times\, (x - ((q_1 - 1)^2 + \beta))\,(x - (\frac{q_1}{4} + \frac{2}{3} + \beta))\ , \\ & q_1 \in [-1, 3],\ \beta \in [0, 1],\ \gamma \in [-1, 1] \end{aligned}$$

So, the second largest root takes values in the interval $[0.7065, 5]$. The minimum is attained for $(q_1, \beta, \gamma)_{\min} = (0.159, 0, 0)$ and the maximum for $(q_1, \beta, \gamma)_{\max} = (-1, 1, -1)$.

The computation time for this example was 7.860 s for our Maple implementation. We didn't prepare the input for the QEPCAD B program containing the condition "second largest root", and thus we skipped this computation.

Example 5.5 ($\mathcal{H}_2$ regulation problem). The next example originates from the $\mathcal{H}_2$ regulation problem [6]. It describes the optimization of tuning parameters of a plant from an admissible region to minimize the optimal performance level. A detailed description can be found in [5]. In FIGURE 7, $P(s)$ is the transfer function of the SISO continuous-time, linear, time-invariant plant, whereas $K(s)$ is the transfer function of the controller. The signals $r(t),\ u(t),\ y(t),\ d(t)$, and $e(t) := r(t) - u(t)$ are the reference input, the control input, the control output, the disturbance input, and the error signal, respectively. In the case of the $\mathcal{H}_2$ regulation problem, it is assumed that there is no reference input, i.e., $r(t) \equiv 0$, and that the disturbance input takes the form of an impulse signal, i.e., $d(t) = \delta(t)$. The controller is designed to regulate the plant output to zero. The function

$$E(P, K) := \int_0^\infty \left(|y(t)|^2 + |u(t)|^2\right) dt$$

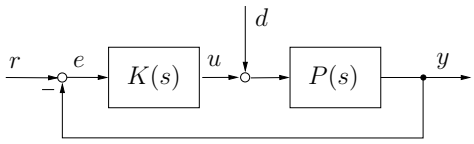


FIGURE 7. Unity feedback system configuration

is used to measure its performance. The best (minimal) performance level is thus given by

$$E^*(P) := \inf_{K \text{ stabilizing}} E(P,K).$$

In [5], it is shown that the best $\mathcal{H}_2$ regulation performance E^* can be expressed explicitly in terms of two *sums of roots*, when the plant P is a) strictly proper, and b) minimum phase (yet possibly unstable).

The *sum of roots* expression for $E^*(P)$ can then be used to design an optimal plant P^*, which – when used in conjunction with its optimal controller $K_{\text{opt}}(P^*)$ – achieves the best possible $E^*(P)$, i.e., $P^* = \text{argmin}_P\{E^*(P)\}$.
Here we consider the following plant:

$$P(s) := \frac{(3-2q_1)(1+q_2^2)}{s(s-2q_1^2+q_2)}, \quad q_1 \in [0,1],\ q_2 \in \left[\frac{9}{10}, \frac{11}{10}\right].$$

The minimal performance level is given by the value of the maximal real root of the polynomial $f_5(x;(q_1,q_2))$ [5], as defined below:

$$\begin{aligned} f_5(x;(q_1,q_2)) \quad := \quad & x^4 + (-8\,q_1^2 + 4\,q_2)\,x^3 + (16\,q_1^4 - 16\,q_1^2\,q_2 + 4\,q_2^2)x^2 - 36 + 96\,q_1\,q_2^2 \\ & +48\,q_1 - 72\,q_2^2 - 16\,q_1^2 - 36\,q_2^4 + 48\,q_1\,q_2^4 - 32\,q_1^2\,q_2^2 - 16\,q_1^2\,q_2^4, \\ & q_1 \in [0,1],\ q_2 \in \left[\frac{9}{10}, \frac{11}{10}\right] \end{aligned} \tag{5.1}$$

The maximal real root takes values in the interval $[2.301, 3.298]$. The minimum is attained for $(q_1,q_2)_{\min} = (0.393, \frac{9}{10})$ and the maximum for $(q_1,q_2)_{\max} = (1, \frac{9}{10})$.

The computation time for this example was 18.241 s for the QEPCAD B program vs. 1.220 s for our Maple implementation.

Example 5.6 ($\mathcal{H}_\infty$-norm computation). As described in Section 1, the $\mathcal{H}_\infty$-norm computation problem can be reformulated as a root optimization problem.

Consider the transfer function matrix

$$G(s) := \begin{bmatrix} \dfrac{2\left(1-50\,\alpha^2\right)}{s^2+1/10\,s+1} & \dfrac{2\,s+5+\alpha}{s^2+1/10\,s+1} \\ \dfrac{1}{s^2+(1/4+\alpha)\,s+10} & \dfrac{30\ (1+7\,\alpha)}{s^2+(1/4+\alpha)\,s+10} \end{bmatrix}.$$

We want to calculate the frequency-restricted $\mathcal{H}_\infty$-norm of $G(s)$:

$$\|G(s)\|_{\infty,[0,3]} = \sup_{\omega\in[0,3]} [G(j\omega)]\,,$$

where $\sigma_{\max}$ is the maximal real root of $\det\big(\lambda^2 I - G^T(-j\omega)G(j\omega)\big)$.

To reduce the degree, we set $\tilde{\lambda} = \lambda^2$ and write $f_6(\tilde{\lambda};(\alpha,\omega)) := \det\big(\tilde{\lambda}I - G^T(-j\omega)G(j\omega)\big)$. Thus we get:

$$\begin{aligned} f_6(\tilde{\lambda};(\alpha,\omega)) \;:=\; & \big(160000 - 350300\,\omega + 1600\,\omega^3\,\alpha^2 + 800\,\omega^3\,\alpha + 800\,\alpha\,\omega \\ & \quad +1600\,\alpha^2\,\omega - 1592\,\alpha\,\omega^2 - 3184\,\alpha^2\,\omega^2 \\ & \quad +225081\,\omega^2 - 35084\,\omega^3 + 1600\,\omega^4\big)\cdot\tilde{\lambda}^2 \\ & +\big(319638400\,\alpha^4\,\omega + 303200\,\omega\,\alpha^3 - 16000000\,\alpha^6\,\omega + 127631900\,\alpha^2\,\omega \\ & \quad -8000000\,\alpha^5\,\omega - 1600000000\,\alpha^4 + 3153884\,\omega - 6400\,\omega^3 \\ & \quad -69928000\,\alpha^2\,\omega^2 - 20179200\,\alpha\,\omega^2 + 40414200\,\alpha\,\omega - 21760000\,\alpha \\ & \quad -16000000\,\alpha^4\,\omega^2 - 6720000\,\alpha^2 - 6081600 - 1360400\,\omega^2\big)\cdot\tilde{\lambda} \\ & +4840000 - 13756800000\,\alpha^4 + 201600000000\,\alpha^5 + 73744000\,\alpha \\ & \quad -7718400000\,\alpha^3 + 705600000000\,\alpha^6 + 6400\,\omega - 247102400\,\alpha^2. \end{aligned}$$

We optimize the maximal real root of $f_6(\tilde{\lambda};(\alpha,\omega))$ for parameter values $\alpha \in [-0.15, 0.15]$ and $\omega \in [0,3]$ to compute the largest and smallest values for $\|G(s)\|^2_{\infty,[0,3]}$. The trace of the roots of $f_6(\tilde{\lambda};(\alpha,\omega))$ can be seen in the left part of FIGURE 8. In the right part, the logarithm of the roots is plotted to show more clearly the two surfaces formed by the two roots of $f_6(\tilde{\lambda};(\alpha,\omega))$. Also, the roots are shown for a wider parameter range, $\omega \in [0,9]$.

With the use of our prototype program, we can determine the range of the maximal root of $f_6(\tilde{\lambda};(\alpha,\omega))$. This is equal to the range of the square of the frequency-restricted $\mathcal{H}_\infty$-norm of $G(s)$,

$$\text{range}(\|G(s)\|^2_{\infty,[0,3]}) = [\min(\sigma^2_{\max}), \max(\sigma^2_{\max})] = [8.878, 3324.7].$$

In fact, the output of our prototype gives the minimum value $\min(\sigma^2_{\max}) = 8.878$ as belonging to the interval $\min(\sigma^2_{\max}) \in [\frac{892459525321}{268435456}, \frac{446229762661}{134217728}]$. An algebraic description as some root of a defining polynomial is also available. The same is valid for the maximum of $\max(\sigma^2_{\max})$.
The corresponding parameter values are $(\alpha,\omega)_{\min} = (-0.15, 3)$ for $\min(\sigma^2_{\max})$ and $(\alpha,\omega)_{\max} = (0.0146, 0.996)$ for $\max(\sigma^2_{\max})$.

For this example, the computation with the QEPCAD B program exits with the error "prime list exhausted". This is an error occurring in the saclib2.2.0 library when too many cells are needed.
The computation with our Maple implementation obtained the answer in 65.632 s. When comparing this time with the computation time from Examples 5.1-5.5, the growth is significant. The long computation time is due to the high order dependency of $f_6(\lambda;(\alpha,\omega))$ on the parameters α and ω.

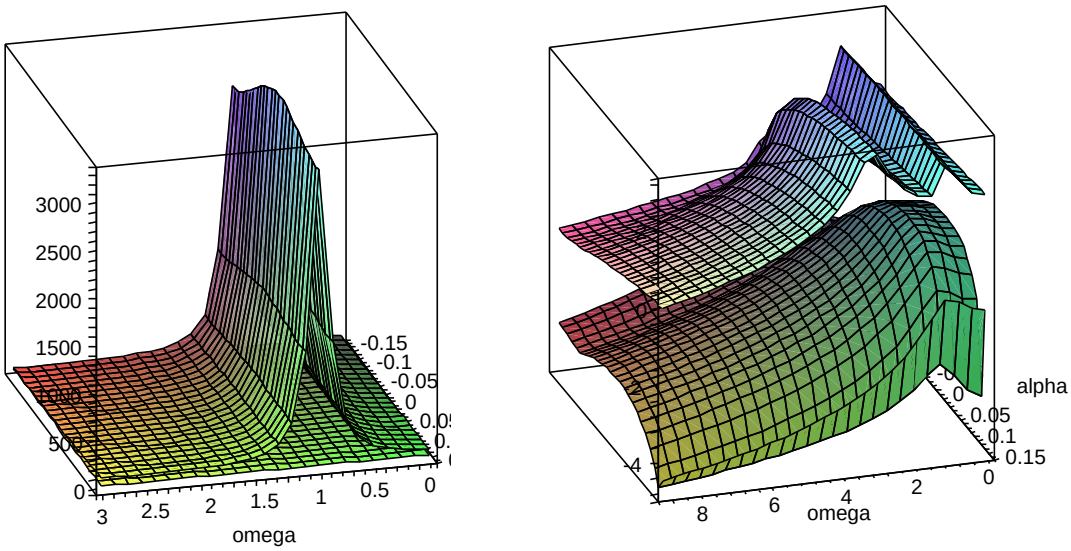


FIGURE 8. Graph of the roots of $f_6(\tilde{\lambda};(\alpha,\omega))$
Left: Trace of the roots of $f_6(\tilde{\lambda};(\alpha,\omega))$. The maximum of the largest real root attains a value > 3000, whereas the minimum is close to 0.
Right: Logarithm ($\log_{10}$) of the left graph, showing the function for a larger parameter range. The two roots of $f_6(\tilde{\lambda};(\alpha,\omega))$ form two surfaces in the parameter domain.

	QEPCAD B	Prototype
Example 5.1	0.116	0.180
Example 5.2	5.036	4.384
Example 5.3	4.816	1.220
Example 5.4	not computed	7.860
Example 5.5	18.241	0.844
Example 5.6	failed	65.632

TABLE 1. Comparison of computation times (sec)

The computation times measured for Examples 5.1-5.6 are summarized in TABLE 1. Our prototype program computed the optimal values faster than the QEPCAD B in all examples apart from Example 5.1 (due to implementation issues) and Example 5.4 (not computed with the QEPCAD B program, so no comparison

possible). Therefore, we can say that in spite of an unfavourable implementation, our prototype program outperformed the conventional approach that employs conditions from the Sturm-Habicht sequence.

6. Concluding Remarks

In this work, we presented an algorithm to optimize a particular real root of a polynomial. We used the fact, that we do not focus on all real roots but only on one particular real root. Via introducing the Number-of-Root (NoR) criterion, we were able to propose a specialized cylindrical decomposition that does not require any additional condition sets obtained from the Sturm-Habicht sequence. Combining this approach with partial lifting techniques, we reduced the problem size to a far more manageable size. We implemented the algorithm in Maple and tested it against a conventional implementation in QEPCAD B .

Our prototype program outperformed the conventional approach that employs conditions from the Sturm-Habicht sequence. Therefore, our approach represents a significant advancement; we were able to cut down the computation time while enlarging the size of the problems that are tangible for our algorithm. Also, the exactness of the problem solution remains guaranteed.

Appendix A. Real Root Counting — Sturm-Habicht Sequence

In Problem 2.3, we introduced an algebraic expression $\psi(x;\mathbf{q})$ that describes the chosen root. In order to obtain such an algebraic expression, the Sturm-Habicht sequence $SH\,(f(x))$[3] of a polynomial $f(x)$ can be used. It is defined as

$$SH\,(f(x)) := \big\{g_0(x), \ldots, \underbrace{g_{s-1}(x)}_{f'(x)}, \underbrace{g_s(x)}_{f(x)}\big\},$$

where $g_0(x), \ldots, g_{s-2}(x)$ is the subresultant chain of f and f' (changed signs).

A.1. Sturm–Habicht Sequence and Real Root Counting

The identification of a real root is done via counting the number of real roots. This is done as follows. The function $W_{SH}(f;\alpha)$ denotes the number of sign variations of the number sequence obtained when evaluating $SH\,(f(x))$ at α. The function $W_{SH}(f;\alpha,\beta)$ then gives the difference between $W_{SH}(f;\alpha)$ and $W_{SH}(f;\beta)$. Namely,

$$\begin{aligned} W_{SH}(f;\alpha) &:= \#\text{ sign variations in } \{g_0(\alpha), \ldots, g_s(\alpha)\}, \\ W_{SH}(f;\alpha,\beta) &:= W_{SH}(f;\alpha) - W_{SH}(f;\beta);\ \alpha,\, \beta \in \mathbb{R} \cup \{-\infty, +\infty\}. \end{aligned}$$

The important result is that the value of this function, $W_{SH}(f;\alpha,\beta)$, gives the number of roots of $f(x)$ in the interval (α,β):

$$W_{SH}(f;\alpha,\beta) = \#\text{ real roots of } f(x) \text{ in } (\alpha,\beta).$$

A.2. Algebraic representation of the maximal real root

Thus, we can give the following algebraic representation of the maximal real root:

A value x_0 is the maximal real root in f

$$\Leftrightarrow \quad f(x_0) = 0 \wedge \text{ there is no other zero of } f \text{ in } (x_0, \infty)$$
$$\Leftrightarrow \quad f(x_0) = 0 \wedge W_{SH}(f; x_0, \infty) = 0$$

These Sturm-Habicht conditions tend to be complicated conditions of high degree. Thus, the introduction of these additional constraints to identify a root results in a large number of additional cells in the cylindrical algebraic decomposition. The following example will illustrate this increased complexity of the conditions.

Example A.1.

$$f(x; (q_1, q_2)) = \big(x - (q_1 + q_2^2)\big)\big(x - ((q_1 - 1)^2 + q_2^2)\big)\big(x - (q_1/4 + 2/3 + q_2^2)\big),$$
$$D_{(q_1,q_2)} = [-1, 3] \times [-1, 1].$$

The corresponding condition set

$$\exists (q_1, q_2)\big[f(x; (q_1, q_2)) = 0 \wedge \psi(x; (q_1, q_2)) \wedge \phi(q_1, q_2)\big]$$

is given as

$$\exists q_1\, \exists q_2\big[\, q_1 \geq -1 \,\wedge\, q_1 \leq 3 \,\wedge\, q_2 \geq -1 \,\wedge\, q_2 \leq 1 \,\wedge$$
$$\big(x - (q_1 + q_2^2)\big)\big(x - ((q_1 - 1)^2 + q_2^2)\big)\big(x - (q_1/4 + 2/3 + q_2^2)\big) = 0 \wedge$$
$$36x^2 + (18q_1 - 40 - 24q_1^2 - 72q_2^2)x$$
$$+ 7q_1 - 19q_1^2 + 15q_1^3 + 40q_2^2 + 8 + 36q_2^4 + 24q_1^2q_2^2 - 18q_1q_2^2 > 0 \wedge$$
$$(288q_1^4 - 1512q_1^3 + 2490q_1^2 - 1224q_1 + 224)x$$
$$- 288q_1^4q_2^2 + 1512q_1^3q_2^2 - 160 - 339q_1^3 - 1057q_1^2 + 796q_1$$
$$- 224q_2^2 - 2490q_1^2q_2^2 + 1224q_1q_2^2 - 180q_1^5 + 687q_1^4 > 0\,\big],$$

where the last two inequalities represent the condition $W_{SH}(f(x; \mathbf{q}); x, \infty) = 0$ that identifies the maximal real root of $f(x; \mathbf{q})$.

References

[1] Brown, C. W.: QEPCAD B: A program for computing with semi-algebraic sets using CADs, *ACM SIGSAM Bulletin*, Vol. 37, No. 4, pp. 97–108, 2003.

[2] Collins, G.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Proceedings Second GI Conference on Automata Theory and Formal Languages*, volume 33 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1975, 134–183.

[3] González-Vega, L., Recio, T., Lombardi, H., and Roy, M.-F.: Sturm-Habicht sequences determinants and real roots of univariate polynomials, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, (Caviness, B. F., and Johnson, J. R., eds.), Texts and Monographs in Symbolic Computation, Springer, Wien, New York, 1998, 300–316.

[4] Kanno, M., Yokoyama, K., Anai, H., and Hara, S.: Parametric optimization in control using the sum of roots for parametric polynomial spectral factorization, *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2007* (Brown, C. W., ed.), ACM, New York, 2007, 211–218.

[5] Kanno, M., Hara, S., Anai, H., and Yokoyama, K.: Sum of Roots, Polynomial Spectral Factorization and Control Performance Limitations, *Proceedings of the 46th IEEE Conference on Decision and Control* LA, USA, Dec. 12-14, 2007.

[6] Chen, J., Hara, S., and Chen, G.: Best tracking and regulation performance under control energy constraint, *IEEE Transactions on Automatic Control* 48(8):1320-1336, August 2003.

[7] Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V.: Linear Matrix Inequalities in System and Control Theory, *SIAM Studies in Applied Mathematics* vol. 15, 1994.

[8] Anai, H. and Parrilo, P. A.: Convex Quantifier Elimination for Semidefinite Programming, *Proceedings of the 6th International Workshop on Computer Algebra in Scientific Computing (CASC) 2003*, 3-11, 2003.

[9] Anai, H. and Hara, S. and Kanno, M. and Yokoyama, K.: Parametric Polynomial Spectral Factorization Using the Sum of Roots and Its Application to a Control Design Problem, *Journal of Symbolic Computation* (to appear).

[10] Quantifier Elimination and Cylindrical Algebraic Decomposition, *Texts and monographs in symbolic computation*, (Caviness, B.F., and Johnson, J.R., eds.), Springer-Verlag, 1998.

[11] Gantmacher, F. R.: *The Theory of Matrices*, Chelsea Publishing Company, 1960, vol.2, New York, NY.

[12] Akritas, A. G.: *Elements of Computer Algebra with Applications*, John Wiley & Sons, 1989, New York, NY.

Silvia Gandy
Tokyo Institute of Technology
Meguro-ku, Ookayama 2-12-1-S3-60,
152-8550 Tokyo, Japan
e-mail: gandy@comm.ss.titech.ac.jp

Masaaki Kanno
CREST, Japan Science and Technology Agency
4-1-8 Honcho,
Kawaguchi-shi, Saitama
332-0012, Japan
e-mail: M.Kanno.99@cantab.net

Hirokazu Anai
Fujitsu Laboratories Ltd/ Kyushu University
4-1-1 Kamikodanaka,
Nakahara-ku, Kawasaki
211-8588, Japan
e-mail: anai@jp.fujitsu.com

Kazuhiro Yokoyama
Rikkyo University
3-34-1 Nishi Ikebukuro,
Toshima-ku, Tokyo
171-8501, Japan
e-mail: yokoyama@rkmath.rikkyo.ac.jp

List of MI Preprint Series, Kyushu University

The Grobal COE Program
Math-for-Industry Education & Research Hub

MI

2008-1 Takahiro ITO, Shuichi INOKUCHI & Yoshihiro MIZOGUCHI
Abstract collision systems simulated by cellular automata

2008-2 Eiji ONODERA
The intial value problem for a third-order dispersive flow into compact almost Hermitian manifolds

2008-3 Hiroaki KIDO
On isosceles sets in the 4-dimensional Euclidean space

2008-4 Hirofumi NOTSU
Numerical computations of cavity flow problems by a pressure stabilized characteristic-curve finite element scheme

2008-5 Yoshiyasu OZEKI
Torsion points of abelian varieties with values in nfinite extensions over a p-adic field

2008-6 Yoshiyuki TOMIYAMA
Lifting Galois representations over arbitrary number fields

2008-7 Takehiro HIROTSU & Setsuo TANIGUCHI
The random walk model revisited

2008-8 Silvia GANDY, Masaaki KANNO, Hirokazu ANAI, & Kazuhiro YOKOYAMA
Optimizing a particular real root of a polynomial by a special cylindrical algebraic decomposition