Companion to the Tutorial *Cylindrical Algebraic Decomposition* Presented at ISSAC 2004

Christopher W. Brown[*]
Computer Science Department, Stop 9F
United States Naval Academy
572C Holloway Road
Annapolis, MD 21402
wcbrown@usna.edu

June 30, 2004

1 Introduction

This handout is a companion to the ISSAC 2004 tutorial *Cylindrical Algebraic Decomposition*. It is not intended to stand on its own, rather it is written for someone who has attended the tutorial and taken in the background, motivation, and explanations given there. It is also not intended to thoroughly cite all works pertaining to CAD. It *is* intended to be a guide to someone who might want to read further on this subject. It *is* intended give quick recaps of the ideas and definitions arising in the talk. I hope it is able to serve that role.

2 Definitions

cylindrical algebraic decomposition A cylindrical decomposition of $\mathbb{R}^n$ into semi-algebraic sets is a *cylindrical algebraic decomposition*.

cylindrical decomposition A decomposition of $\mathbb{R}^n$ into finitely many connected regions is *cylindrical* if for any two regions a and b of the decomposition and any k, $1 \leq k \leq n$, the projections of a and b onto $\mathbb{R}^k$ are either identical or disjoint. A cylindrical decomposition D of $\mathbb{R}^n$ induces cylindrical decompositions of $\mathbb{R}^k$ for every $k \leq n$. The *induced* cylindrical decomposition of $\mathbb{R}^k$ consists of the set of all projections onto $\mathbb{R}^k$ of D's partition regions.

delineability A polynomial p is *delineable* over region S if the variety of p in $S \times \mathbb{R}$ consists of finitely many disjoint sections. A set of polynomials is delineable over S if each polynomial is either nullified everywhere in R or delineable over R.

[*]This work is supported in part by the National Science Foundation grant number CCR-0306440.

level The level of a non-constant polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ is the largest k such that the degree in x_k of p is positive.

nullified An n-level polynomial p is *nullified* at a point $\alpha \in \mathbb{R}^{n-1}$ if $p(\alpha_1, \ldots, \alpha_{n-1}, x)$ is the zero polynomial.

projection definable CAD A CAD with truth values representing a set S is *projection definable* if there is a defining Tarski formula for S using only the projection factors. This is easily detected from the CAD data structure.

projection factor set A finite set of polynomials $P \subset \mathbb{R}[x_1, \ldots, x_n]$ is a *projection factor set* if the natural algebraic decomposition of $\mathbb{R}^n$ into maximal connected regions in which the elements of P are sign-invariant is a CAD. This definition holds in the context of the Collins or Collins-Hong projections, but with the McCallum and Brown-McCallum projections or when a specialized CAD construction is used we need something a bit less restrictive.

quantifier elimination Let F be a Tarski formula in the variables $x_1, \ldots, x_n$. Let $G = Q_1 x_{k+1} \cdots Q_{n-k} x_n [F]$, where $Q_i \in \{\exists, \forall\}$. Tarski proved that there is a Tarski formula H in the variables $x_1, \ldots, x_k$ that is equivalent to G over the reals [33]. Constructing this equivalent formula is called *quantifier elimination.*

section of a polynomial Let p be an element of $\mathbb{R}[x_1, \ldots, x_n, z]$ and S a connected region in $\mathbb{R}^n$. Suppose a continuous function $f : S \longrightarrow \mathbb{R}$ satisfies $p(\overline{x}, f(\overline{x})) = 0$ for all $\overline{x} \in S$. The graph of f over S is called a *section* of p.

semi-algebraic set The *semi-algebraic sets* in $\mathbb{R}^n$ are defined recursively by:

1. the set of points satisfying $p \ \sigma \ 0$, where $p \in \mathbb{R}[x_1, \ldots, x_n]$ and $\sigma \in \{=, \neq, <, \leq, >, \geq\}$, is semi-algebraic
2. the complement of a semi-algebraic set, and the union or intersection of finitely many semi-algebraic sets are semi-algebraic.

stack Let D be a CAD of $\mathbb{R}^n$, and let c be a cell in D's induced CAD of $\mathbb{R}^{k-1}$. The *stack* over c is the set of cells in the induced CAD of $\mathbb{R}^k$ whose projection onto $\mathbb{R}^{k-1}$ is c. The cell c is called the *base* of the stack.

Tarski formula A *Tarski Formula* is boolean combination of polynomial equalities and inequalities, e.g. $x_1^2 + x_2^2 - 1 = 0 \wedge x > 0 \wedge y > 0$.

3 Basic Cylindrical Algebraic Decomposition

The best reference to cite for Collins' original introduction of CAD is [14]. This paper is complete, precise and analyzes everything, but it's not the easiest read. [5] gives a description of CAD that is much easier to read — it's a good place to start. Mishra's textbook on computer algebra [30] has a nice presentation of CAD as well. Collins gives a summary of the development of CAD from the early 70's up to the early 90's, as well as some ideas for future progress, in [15].

4 Implementations

I know of four CAD implementations that are reasonably complete.

- QEPCAD — This system is due primarily to Hoon Hong, but with contributions from many others, including George Collins. [16] is the usual citation for it. Hong's thesis [21] has a fairly thorough explanation of it. It is a stand-alone, command-line, interactive program written in C and based on the SACLIB library of computer algebra functions.
- QEPCAD B — This system forked off of QEPCAD, it's been improved and extended by me. It's C++ rather than C, and has a simple API through which it can be called. It and SACLIB are available from `www.cs.usna.edu~qepcad`. The page also includes documentation. The best reference for it is [12].
- RLCAD — This is an implementation of CAD in Reduce as part of the Redlog system by Andreas Seidl and Thomas Sturm. It is incorporated into the most recent release of Reduce, version 3.8 (see `www.reduce-algebra.com/`). The best reference for it I know is [31].
- Mathematica — There is an implementation of CAD in the Mathematica kernel due to Adam Strzebonski. [32] is a JSC article that gives some information about it. The functionality it offers is described in Mathematica's documentation.

QEPCAD is not really being kept up with. If you're interested you should just use QEPCAD B instead. I'd also like to point out that while implementing CAD in its full generality is difficult, implementing CAD for full-dimensional cells only is easy!

4.1 Projection

The goal of projection is, roughly speaking, to take a set A of polynomials and produce a set P of polynomials, where $A \subseteq P$, such that the natural algebraic decomposition of Euclidean space into maximal connected regions in which the elements of P are sign-invariant is a CAD. This set P, the projection factor set, provides an *implicit* description of the CAD that will ultimately be represented *explicitly* by the CAD data-structure.

There are two basic models for projection: Collins-Hong, which is based on producing a CAD in which projection factors are *sign-invariant*, and Brown-McCallum, which is based on producing a CAD in which projection factors are *order-invariant*.

Collins original paper on CAD [14] describes, of course, his projection operator, along with everything else. Hong's improvement of Collins' projection is presented in [20]. This is probably the best source if you are interested in the Collins-Hong approach to projection.

McCallum's projection operator is presented in two articles. [23] presents the case of three dimensions, [25] presents the general case. The former is easier to understand, and it is probably best to read it first, in order to get the intuition behind the method. Brown's improvement of McCallum's projection is presented in [9], but it really requires reading McCallum's papers first. The McCallum and Brown-McCallum projections produce smaller projection factor sets, but require a somewhat more complicated lifting algorithm. Moreover, both may fail to produce a CAD, though possible failure is always detected.

[10] gives a more refined test for possible failure than given in McCallum's original paper, and provides a more efficient and general modified lifting algorithm than McCallum. [28] discusses another way of refining the test for possible failure.

4.2 Lifting (a.k.a. Stack Construction)

There hasn't been much done with lifting as its own topic, as it a) depends on the projection operator used, and b) mostly just relies on good underlying algorithms for computing with real algebraic numbers. A literature survey on that, of course, would be a big paper of its own.

The description of basic CAD construction in [5] is very nice, and describes lifting for the Collins-Hong projection model quite well. Arnon [4] described what he called *clustering*, which is basically a strategy for avoiding lifting over some cells by making use of information about which cells are adjacent to which others. For a variety of reasons, nobody seems to talk about this much any more and it is not implemented in any current CAD programs as far as I know. [17] describes the use of floating-point interval arithmetic in lifting. At ACA 2002, Strzebonski talked about more elaborate use of floating-point in CAD construction, but I don't know of any publication about this.

4.3 Solution Formula Construction

A *solution formula* for a CAD representing a set S is a defining Tarski formula for S. One of the most important uses of CAD is to produce *simple* defining formulas for semi-algebraic sets. Especially since many tools for quantifier elimination and other problems concerning semi-algebraic sets tend to produce large output formulas.

In Collins' original paper on CAD, the method he used for solution formula construction was to construct a defining formula for each true cell in the CAD. However, in almost all cases it is impossible to construct a defining Tarski formula for a single cell of a CAD using only the projection factors, so Collins' method also used all derivatives of projection factors. In order to use these derivatives, he needed to be sure that additional properties beyond the delineability of projection factors held, and this required a larger projection, which he called the *augmented projection*. This approach has two drawbacks: 1) it does not produce simple formulas, 2) the augmented projection is typically too large to use.

Hong [22] showed how to construct simple solution formulas for projection definable CADs, i.e. when a solution formula can be constructed solely from the projection factors. He reduced the problem to the combinatorial optimization problem of boolean formula minimization. [8] discusses many improvements to this method, including a method (also described in [7]) for adding projection factors to a CAD in order to make it projection definable, which QEPCAD B implements.

When a CAD isn't projection definable, you can add projection factors to make it projection definable, or you can give a formula in an extended language in which all CADs are projection definable. Both QEPCAD B and Mathematica's CAD implementations offer this second alternative. Both extend the language of Tarski formulas by allowing reference to a particular root of a polynomial by its index (its position in a sorted list of the polynomial's roots), which is of course unambiguous over a region in which a polynomial is delineable. Mathematica's version counts multiplicities, QEPCAD B's does not. In fact, as described in [8], CAD allows quantifier elimination for this extended language just as easily as for Tarski formulas, and since the extended language never requires adding polynomials to the projection factor set, it is a more efficient way of storing intermediate results in CAD-based calculations.

4.4 Other CAD operations

When the CAD data structure is augmented with information about which cells are adjacent to one another, you have complete topological information about the varieties of projection factors and, in particular, about any set you can represent with that CAD. [6] presents an algorithm for calculating adjacencies in a CAD of $\mathbb{R}^2$. [3] described how this could be used to produce topological correct plots of algebraic curves, and in fact most of the CAD plots from the tutorial were produced by QEPCAD B using adjacency information. [29] gives algorithms for computing adjacencies in CADs of $\mathbb{R}^3$ and $\mathbb{R}^4$. As far as I know, neither has been implemented, which is a shame, since this would be quite useful.

Adjacency information allows you to perform some interesting operations on semi-algebraic sets quite easily — constructing the closure of a set, for instance. To determine boundedness, in general, you need adjacency information as well. Things like computing the number of connected components also become easy.

5 Optimizing CAD and using it effectively

There are a variety of things one can do to solve problems more effectively with CAD. Here are a few.

5.1 Constructing less than the full CAD

In [16], Hong and Collins introduced the notion of a *partial* CAD, which simply means that over some cells in induced CADs of lower dimension, we may not bother to construct stacks (lift). If, for example, you're trying to construct a CAD representation of the set $x_1 > 0 \wedge x_2 > 0 \wedge x_3^2 + x_2^2 + x_1^2 = 1$, there is no need to lift over cells in $\mathbb{R}^1$ with negative sample points, or cells in $\mathbb{R}^2$ with sample points whose x_2-coordinate is negative. Though the full CAD of $\mathbb{R}^3$ is not constructed, we are still able to represent the original set. In doing this, we save time and space by not constructing all the cells in the CAD of $\mathbb{R}^3$ defined by the projection factor set.

As partial CAD is presented in [16], there are two ways of avoiding stack constructions. First, for a cell in an induced CAD of lower dimension, you can partially evaluate the input formula, whether quantified or not, at the sample point for that cell. This partial assignment of values to variables may already yield true or false, in which case you don't need to lift over that cell. Second, suppose that x_k is existentially quantified. As soon as any cell in the stack above a cell c in $(k-1)$-space is assigned the value true, c can be assigned the value true and no further lifting over cells in the stack is required. A similar observation holds for universal quantification.

A very important special case of partial CAD is that of only lifting over full dimensional cells — i.e. never lifting over section cells. Not only do you construct fewer cells, but you never need to do any computations with algebraic numbers ([24] notes this) and the Brown-McCallum projection is always valid ([32] notes this). The previous two references look at problems for which exact answers can be obtained from the full dimensional cells alone. In other situations the full dimensional cells might not be enough to get an exact answer, but it may be close enough. After all, any error can only occur in the cells of lower dimension and these constitute a measure zero subset of the space of variables. If variables are physical quantities, for example, lower dimensional subsets are unrealizable, so we lose nothing by ignoring them. Seidl and Sturm [31] take a slightly different approach with their *generic CAD*. Instead of never lifting over any section cells, they identify certain projection factors whose non-vanishing would allow them to reduce the size of the projection factor set. Cells that are sections of these chosen projection factors are never lifted over. Essentially they get the advantage of the simplified projection but not of eliminating algebraic number computations. However, they are then able to give an answer that is exact under the assumption that the chosen projection factors are non-zero.

5.2 Variable Ordering

The variable ordering you use can have a huge impact on CAD construction. In general, variables that appear in few terms and to low degree in the input polynomials ought to be eliminated first. However, the problem may constrain the ordering you use. For example, quantified variables in Q.E. problems must be eliminated before free variables, and while variables within a block of identical quantifiers can be rearranged, the order of quantifier blocks cannot.

With n variables, there are $n!$ possible variable orders, so trying to evaluate each of them is difficult ... especially if “evaluating” means going through some or all of CAD construction! [19, 18] presents a greedy algorithm for choosing a good projection order. It basically tries using each variable for the first projection and whichever gives the “smallest” projection is chosen as the first variable to be eliminated. The process then repeats with the remaining variables. The article derives a good metric for projection factor sets. While obviously much less sophisticated, the following simple heuristic works reasonably well and can be done by hand quite easily:

1. Descending order by degree of variable, breaking ties with
2. Descending order by highest total-degree term in which the variable appears, breaking ties with
3. Descending order by number of terms containing the variable

Note that my convention is that the last variable in the order is the first to be eliminated.

5.3 Preparing Input

CAD makes a poor “black box” for many problems, since it doesn’t take advantage of special things that may appear in the input formula. For example, if $x = 2y+1$ is conjoined with the input formula one should substitute $2y+1$ for x rather than forcing CAD construction to commence with one more polynomial and one more variable than needed. These kind of simple substitutions should be done by hand or by some other program before CAD is used.

Breaking up input into smaller pieces, when possible, is also important. For example, instead of solving $\exists x[F(x) \vee G(x)]$ with one CAD computation, you are better off solving it with two as $\exists x[F(x)] \vee \exists x[G(x)]$ unless F and G contain the same polynomials. Even then splitting may be good, because you may be able to optimize each of the two smaller problems. This kind of splitting is even a good idea if the results of the quantifier elimination are to be combined and subjected to more quantifier elimination. For example, instead of solving $\forall y \exists x[F(x,y) \vee G(x,y)]$ as a single application of CAD, it is typically better to perform Q.E. on $\exists x[F(x,y)]$ and $\exists x[G(x,y)]$ separately and take their results, call them $F'(y)$ and $G'(y)$, and perform Q.E. on the combined formula $\forall y[F'(y) \vee G'(y)]$. Note, however, that doing this generally requires that F' and G' are in one of the extended languages in order to be efficient! The "extra" polynomials one may need to add in order to produce Tarski formulas would slow down the second quantifier elimination step.

[11] describes how the divide and conquer technique of splitting input into small pieces, solving and combining results can be used to simplify very large formulas using CAD. This is implemented in the program SLFQ (`www.cs.usna.edu/~qepcad/SLFQ`). The same article proves that converting non-prenex formulas to prenex form in order to apply CAD is less efficient than solving the non-prenex parts separately and substituting the results back into the original formula.

5.4 Make use of the special properties of CAD

A CAD representing a semi-algebraic set tells you a lot about the set and about the polynomials in the projection factor set. Quantifier elimination doesn't exploit all of this extra information, so it is often better not to phrase a problem as a Q.E. problem and then try to apply CAD to solve the Q.E. problem, but rather to apply CAD directly to the problem in a way that exploits all that extra information. Here's a quick example: Suppose you want to characterize all the monic quartic polynomials that have four distinct positive real roots. Phrasing this as a Q.E. problem requires different variables for each of the four roots you're looking for. This is wasteful! Instead, you can construct a CAD for $p = x^4 + ax^3 + bx^2 + cx + d$ and $q = x$ with x as the first variable to be eliminated, and from this CAD you can deduce all that you need. Over each cell in the CAD of $\mathbb{R}^4$ you can count the distinct positive zeros of p and assign truth values accordingly. This CAD defines the set of values for a, b, c, d for which p has four distinct, real, positive roots. Maybe the CAD alone is enough to answer your questions, otherwise you might construct a defining Tarski from it. In any event, by making use of CAD directly, we've saved ourselves three variables.

One may also try to optimize CAD construction to specific classes of problems. As previously discussed, if the problem allows you to avoid lifting over cells with algebraic coordinates you can save a lot of time. [9] describes how projection may be simplified for Q.E. problems for which it is known a priori that the set to be projected is bounded between continuous functions. [1] describes a version of Q.E. by CAD that is optimized to solve semidefinite programming problems. The modified CAD takes advantage of special properties of this problem in many different ways, in projection, lifting, and even in assigning truth values to cells. They do not construct a CAD from the set of polynomials in the formula defining the set of interest, rather than construct a simpler CAD that they know will suffice to represent the set (i.e. each cell is either entirely in or entirely out of the set) and simply evaluate the defining formula at sample points to assign truth values. This technique, which applies in other circumstances as well, takes advantage of the fact that the projection factor set for a CAD defining a set may often be smaller than a projection factor set containing the polynomials in a defining formula, which is essentially the reverse of the previously mentioned problem of adding polynomials to a projection factor set in order to make a CAD projection definable.

5.5 Equational constraints

In his “20 Years of Progress” paper [15], Collins proposed the use of *equational constraints* in projection and lifting. Recall that a previous section discussed taking advantage of inputs in which an equation of the form $x_k = f(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots x_n)$, where f is a polynomial or a rational function with non-vanishing denominator, is conjoined with the rest of the input formula by doing a simple substitution to eliminate x_k. The equational constraints optimization is to take advantage of equations $p = 0$ that are conjoined with the input formula (i.e. equational constraints), even when we can’t solve for any of the variables. The basic idea is that only the constraint polynomial needs to be delineable, the other projection factors merely need to be sign-invariant within the sections of the constraint polynomial. This insight led Collins to propose a reduced projection and a lifting scheme that avoided raising stacks over many cells.

Unfortunately, a rigorous realization of this scheme has proven to be difficult. McCallum has made some progress in this [26, 27] area, but it is tricky to apply equational constraints in a way that is guaranteed to be correct. In fact, the second of McCallum’s papers on the subject contains a subtle error. Making effective use of equational constraints would be a big improvement in CAD-based methods for a common class of problems, i.e. those with equational constraints.

6 Case Studies

This section will look at many problems that can be solved by CAD. Some apply CAD in a straightforward way to do Q.E. or formula simplification, but the others demonstrate this idea of specializing CAD or optimizing it for certain kinds of problems.

6.1 A problem from epidemiology

Andreas Weber and his colleagues have been working on applying symbolic tools to investigations of epidemiological models. The following example will appear in a paper to be given at CASC 2004. We consider the SEIT model, used to model tuberculosis in [34]. It uses a system of ODEs to model the movement of disease through a population. The model contains many parameters, and the question is this: For what parameters is there an *endemic equilibrium*, i.e. an equilibrium that doesn’t have the disease dying out? Here’s the model:

$$\begin{aligned}
S' &= d - dS - \beta_1 IS \\
E' &= \beta_1 IS + \beta_2 IT - (d + \nu + r_1)E + (1 - q)r_2 I \\
I' &= \nu E - (d + r_2)I \\
T' &= -dT + r_1 E + q r_2 I - \beta_2 TI
\end{aligned}$$

S susceptibles	β_1, β_2 transmission parameters for S and T
E exposed (not yet infectious)	d birth and death rate (assumed equal)
I infectious	ν rate of change from exposed to infectious
T under treatment	r_1, r_2 treatment rates for E and I
	q fraction of infectious successfully treated

An endemic equilibrium satisfies $0 = S', E', I', T'$ and $0 < S, E, I, T$. So, we want to know for which parameter values there is a solution to

$$
\begin{aligned}
0 &= d - dS - \beta_1 IS \\
0 &= \beta_1 IS + \beta_2 IT - (d + \nu + r_1)E + (1 - q)r_2 I \\
0 &= \nu E - (d + r_2)I \\
0 &= -dT + r_1 E + qr_2 I - \beta_2 TI
\end{aligned}
$$

satisfying $0 < S, E, I, T$, assuming all parameters positive. This can be phrased as the quantifier elimination problem:

$$\exists S, E, I, T[\text{system is satisfied} \wedge 0 < S, E, I, T \wedge 0 < d, s, \beta_1, \beta_2, \nu, q, r1, r2]$$

However, constructing a CAD for this takes way too long (a nice theory of equational constraints to apply might ameliorate this!). But it's easy to solve for E,I and T in terms of S using three of the equations. Substituting the results into the fourth gives $P = 0$, where

$$
\begin{aligned}
P = &-\nu S^2\beta_1^2 + \beta_1\nu S^2\beta_2 + d\beta_1 Sr_2 - d^2\beta_2 S + d^2\beta_1 S + \beta_1 Sr_1r_2 - d\nu S\beta_2 \\
&+ \nu\beta_1 Sqr_2 - d\beta_2 r_2 S + d\nu S\beta_1 - \beta_1 S\nu\beta_2 + \beta_1 Sr_1 d + \beta_2 d^2 + \nu\beta_2 d + \beta_2 dr_2
\end{aligned}
$$

The condition $0 < S, E, I, T$ is easily seen to be equivalent to $0 < S < 1$. So there is an endemic equilibrium for any assignment of positive parameter values for which there is a a real value S such that $P(S) = 0 \ \wedge \ 0 < S < 1$, i.e.

$$\exists S\,[P(S) = 0 \ \wedge \ 0 < S < 1]$$

assuming that all parameters are positive. Qepcad b is able to construct a CAD for this quickly, but the solution formula is rather large — it filled up an entire slide in the tutorial, and that at a small font size. Certainly the solution is not very illuminating. However, there are some more conditions that the model really assumes the parameters satisfy, such as $\beta_1 > \beta_2$. When this condition is added to the assumptions, the solution formula is much nicer: $\nu\beta_1 - r_1r_2 - dr_2 - \nu qr_2 - dr_1 - d^2 - \nu d > 0$. This example shows several important things:

- the importance of preparing input (recall the substitutions made to eliminate E, I, and T),
- the importance of variable order (variables orderturns out to be crucial for this problem, in the computations the earlier simple heuristic was used),
- that you can only *get* a simple defining formula if the set your Q.E. problem describes *has* a simple defining formula.
- that sometimes constructing CADs of high dimension (8 variables for this problem) is feasible.

6.2 The external trisector problem

Consider the triangle ABC from Figure 1. We define *the external trisector of B with respect to A* as the segment connecting vertex B with the intersection of the ray from A through C and the external trisector of ϕ shown in the figure. Of course the external trisector of B with respect to A does not always exist, and the problem we consider is to characterize in terms of the side lengths a, b and c the triangles for which it does exist.[1] We can do this quite nicely with formula simplification.

[1]This question arose from work with George Nakos.

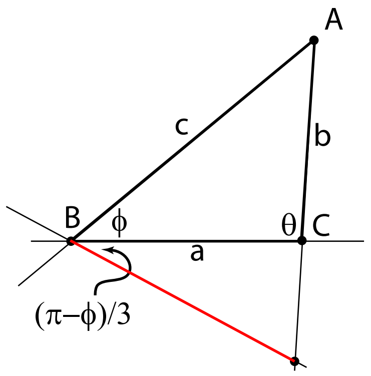

Figure 1: The external trisector of B with respect to A.

It is clear from the picture that the external trisector of B with respect to A exists if and only if $(\pi - \phi)/3 < \theta$. However, we'd like a characterization in terms of side lengths not angles. We derive an equivalent statement to $(\pi - \phi)/3 < \theta$ in terms of a, b and c using little more than the law of cosines:

Case 1: Assuming $\theta \leq \pi/3$ we derive the following:

$$\begin{array}{rcl}
\pi - \phi & < & 3\theta, \text{ note that both sides are in } [0,\pi] \\
-\cos(\pi - \phi) & < & -\cos(3\theta) \\
\cos\phi & < & -\cos(3\theta) \\
\cos\phi & < & -4\cos^3\theta + 3\cos\theta \\
\frac{a^2+c^2-b^2}{2ac} & < & -4\left(\frac{a^2+b^2-c^2}{2ab}\right)^3 + 3\left(\frac{a^2+b^2-c^2}{2ab}\right) \\
a^2b^3\left(a^2+c^2-b^2\right) & < & -c\left(a^2+b^2-c^2\right)^3 + 3a^2b^2c\left(a^2+b^2-c^2\right)
\end{array}$$

Case 2: Assuming $\theta > \pi/3$ we see immediately that $\pi - \phi < 3\theta$ holds.

We are in Case 1 exactly when

$$\begin{array}{rcl}
\theta & \leq & \pi/3, \text{ note that both sides are in } [0,\pi] \\
\cos\theta & \geq & \cos\frac{\pi}{3} \\
\frac{a^2+b^2-c^2}{2ab} & \geq & \frac{1}{2} \\
a^2+b^2-c^2 & \geq & ab,
\end{array}$$

and so are in Case 2 when $a^2 + b^2 - c^2 < ab$. Putting it all together, the external trisector of B with respect to A exists if and only if the side lengths a, b, c satisfy

$$\underbrace{a^2+b^2-c^2 \geq ab}_{\text{Case 1}} \wedge \underbrace{a^2b^3\left(a^2+c^2-b^2\right) < -c\left(a^2+b^2-c^2\right)^3 + 3a^2b^2c\left(a^2+b^2-c^2\right)}_{\text{Derived condition for Case 1}} \vee \underbrace{a^2+b^2-c^2 < ab}_{\text{Case 2}}$$

This formula is a characterization of the existence of the external trisector in terms of the side lengths, but not a very nice characterization. We can use CAD (through QEPCAD B, in this case) to simplify this formula under the assumption that a, b and c are actually side lengths of a non-degenerate triangle, i.e. that all three are positive and satisfy the triangle inequalities. These assumptions are implicit in the law of cosines. The resulting characterization, $c^2 + bc - a^2 > 0$, is quite a bit simpler, gives more insight, and provides better input for further computation.

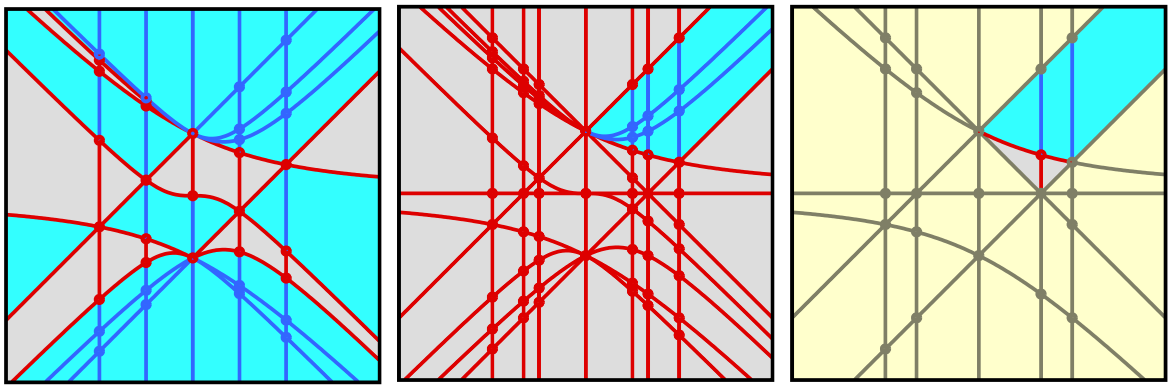

Figure 2: Simplifying the condition for the existence of the external trisector of A w.r.t. B.

The condition we derived actually defines a complicated set. The assumptions, however, carve out a simple piece of that set, which is why we ultimately arrive at a simple equivalent formula. A key component of deriving this simple characterization is CAD simplification, which allows us to discard projection factors that are not relevant to defining the set. Figure 2 shows this process. By normalizing side a to be one, we can view triangle abc as a point in the bc-plane. The figure shows the CAD of of the ab-plane representing the condition, a CAD representing the condition with the assumptions, and a simplified CAD representing the condition and the assumptions.

6.3 Sign-stack sequences

The sign stack sequence problem comes from the thesis "Signed Sequences and Rolle's Restrictions: Why Not All Real Differentiable Functions and Polynomials Satisfying Rolle's Theorem Are Constructible", by Bruce Anderson [2].

If f is a monic nth degree polynomial in x, the "sign stack sequence" for f is a sequence of $(n+1)$-tuples. Each tuple represents the signs of $(f, f', f'', .., f^{(n)})$ at a point. The sequence gives all the the distinct sign-tuples taken by f as x goes from $-\infty$ to $+\infty$, excluding points at which f or any of its derivatives are zero. We're only interested in the case that f is generic, meaning that there are no pairwise common zeros amongst f and its derivatives. There are some clear restrictions on the sign-stack sequence for f:

1. *monic* implies: The rightmost entry is always $+$, the first sign-stack is alternating $+$'s and $-$'s, and the last sign-stack is all $+$'s.
2. *generic* implies: Consecutive sign-stacks differ in only one entry.
3. *Rolle's theorem* implies: From one sign-sequence to the next, an entry may only be changed to equal the entry to its right.

A sequence satisfying these requirements is called *legal*. The most fundamental question in Anderson's thesis is this: "Are there legal sign-stack sequences that are not realized by any polynomial?"

Anderson's thesis considers this problem for polynomials up to degree 5. He enumerates all sign-stack sequences and concludes that some legal sequences are in fact not realizable by 4th and 5th degree polynomials. This enumeration was done in a time consuming ad hoc way, and in fact seems to have some errors for the 5th degree case. However, as discussed in the tutorial, a CAD for polynomial $p(a_1, \ldots, a_n, x)$ and all of its derivatives, where x is the first variable eliminated, actually gives you all realizable sign-stack sequences. Moreover, all generic sequences occur in full dimensional cells, so we can in fact answer Anderson's question by constructing the full dimensional cells alone.

I've posted a tweaked version of QEPCAD B called "Rolle" at `www.cs.usna.edu/~qepcad/ROLLE/`. It does a depth-first traversal of the CAD data structure for p and its derivatives, printing out sign-stack sequences as they are encountered. It completely solves this problem up to and including degree 5.

Anderson's thesis mentions in an appendix that this problem could be solved through quantifier elimination, and even mentions CAD-based Q.E. explicitly. However, it states (correctly) that this approach would be utterly infeasible. That's because Q.E. is not a natural language for specifying the problem. Many variables and polynomials are introduced that are artifacts of the language of quantified Tarski formulas, not of the problem. What we did here was to apply CAD directly to the problem, to take advantage of all the information contained in a CAD data structure, and to optimize CAD for this particular application.

7 Conclusion

I strongly urge people who are faced with a problem to solve and who contemplate using CAD to solve it to consider CAD in this light — as a tool to be adapted, not a black box. Adapt and specialize CAD to the problem at hand. I think that a lot of interesting research and a lot of successful applications of computer algebra could result. I have endeavored in this tutorial to give the background and intuituion necessary to see these opportunities, and in this handout to give the references needed to follow through with proofs and algorithms. If you have questions about implementation, which was really beyond the scope of the tutorial, please feel free to contact me and I'll do my best.

References

[1] ANAI, H., AND PARRILO, P. A. Convex quantifier elimination for semidefinite programming. In *Proceedings of the International Workshop on Computer Algebra in Scientific Computing (CASC)* (2003).

[2] ANDERSON, B. *Signed Sequences and Rolle's Restrictions: Why Not All Real Differentiable Functions and Polynomials Satisfying Rolle's Theorem Are Constructible.* PhD thesis, Cornell University, May 1992. Advisor: Moss Sweedler.

[3] ARNON, D. S. Topologically reliable display of algebraic curves. In *Proceedings of SIGGRAPH* (1983), pp. 219–227.

[4] ARNON, D. S. A cluster-based cylindrical algebraic decomposition algorithm. *Journal of Symbolic Computation 5*, 1,2 (1988), 189–212.

[5] ARNON, D. S., COLLINS, G. E., AND MCCALLUM, S. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing 13*, 4 (1984), 865–877.

[6] Arnon, D. S., Collins, G. E., and McCallum, S. Cylindrical algebraic decomposition II: An adjacency algorithm for the plane. *SIAM Journal on Computing 13*, 4 (1984), 878–889.

[7] Brown, C. W. Guaranteed solution formula construction. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1999), pp. 137–144.

[8] Brown, C. W. *Solution Formula Construction for Truth Invariant CAD's.* PhD thesis, University of Delaware, 1999.

[9] Brown, C. W. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation 32*, 5 (November 2001), 447–465.

[10] Brown, C. W. The McCallum projection, lifting, and order-invariance. See `www.cs.usna.edu/~wcbrown/research/techreports.html`, September 2001.

[11] Brown, C. W. Simple CAD construction and its applications. *Journal of Symbolic Computation 31*, 5 (May 2001), 521–547.

[12] Brown, C. W. QEPCAD B – a program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin 37*, 4 (December 2003), 97–108.

[13] Caviness, B., and Johnson, J. R., Eds. *Quantifier Elimination and Cylindrical Algebraic Decomposition.* Texts and Monographs in Symbolic Computation. Springer-Verlag, 1998.

[14] Collins, G. E. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Lecture Notes In Computer Science* (1975), vol. Vol. 33, Springer-Verlag, Berlin, pp. 134–183. Reprinted in [13].

[15] Collins, G. E. Quantifier elimination by cylindrical algebraic decomposition - 20 years of progress. In *Quantifier Elimination and Cylindrical Algebraic Decomposition* (1998), B. Caviness and J. Johnson, Eds., Texts and Monographs in Symbolic Computation, Springer-Verlag.

[16] Collins, G. E., and Hong, H. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation 12*, 3 (Sep 1991), 299–328.

[17] Collins, G. E., Johnson, J. R., and Krandick, W. Interval arithmetic in cylindrical algebraic decomposition. *Journal of Symbolic Computation 34*, 2 (Aug. 2002), 145–157.

[18] Dolzmann, A., Seidl, A., and Sturm, T. Efficient projection orders for cad. Technical report, FMI, Universität Passau, 2003.

[19] Dolzmann, A., Seidl, A., and Sturm, T. Efficient projection orders for cad. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (ISSAC 2004)* (Santander, Spain, July 2004), J. Gutierrez, Ed., ACM. To Appear.

[20] Hong, H. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1990), pp. 261–264.

[21] Hong, H. *Improvements in CAD–based Quantifier Elimination.* PhD thesis, The Ohio State University, 1990.

[22] Hong, H. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1992), pp. 177–188.

[23] McCallum, S. An improved projection operation for cylindrical algebraic decomposition of three-dimensional space. *Journal of Symbolic Computation 5*, 1,2 (1988), 141–161.

[24] McCallum, S. Solving polynomial strict inequalities using cylindrical algebraic decomposition. *The Computer Journal 36*, 5 (1993), 432–438.

[25] McCallum, S. An improved projection operator for cylindrical algebraic decomposition. In *Quantifier Elimination and Cylindrical Algebraic Decomposition* (1998), B. Caviness and J. Johnson, Eds., Texts and Monographs in Symbolic Computation, Springer-Verlag, Vienna.

[26] McCallum, S. On projection in CAD-based quantifier elimination with equational constraint. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1999), S. Dooley, Ed., pp. 145–149.

[27] McCallum, S. On propagation of equational constraints in CAD-based quantifier elimination. In *Proc. International Symposium on Symbolic and Algebraic Computation* (2001), B. Mourrain, Ed., pp. 223–230.

[28] McCallum, S. On order-invariance of a binomial over a nullifying cell. In *Proc. International Symposium on Symbolic and Algebraic Computation* (2003), R. Sendra, Ed., pp. 184–190.

[29] McCallum, S., and Collins, G. E. Local box adjacency algorithms for cylindrical algebraic decompositions. *Journal of Symbolic Computation 33*, 3 (Mar. 2002), 32–342.

[30] Mishra, B. *Algorithmic Algebra*. Springer-Verlag New York, Inc., 1993.

[31] Seidl, A., and Sturm, T. A generic projection operator for partial cylindrical algebraic decomposition. In *Proc. International Symposium on Symbolic and Algebraic Computation* (2003), R. Sendra, Ed., pp. 240–247.

[32] Strzebonski, A. Solving systems of strict polynomial inequalities. *Journal of Symbolic Computation 29* (2000), 471–480.

[33] Tarski, A. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 1951. second ed., rev. Reprinted in [13].

[34] van den Driessche, P., and Watmough, J. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission. *Mathematical Biosciences 180* (2002), 29–48.