An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions

Changbo Chen, Marc Moreno Maza

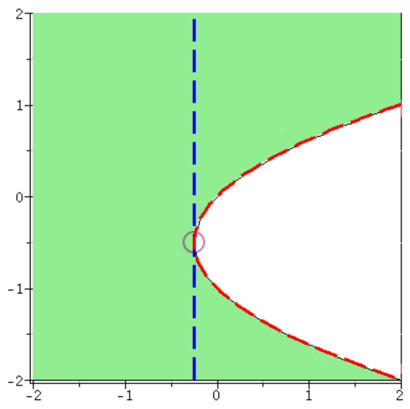
ORCCA, University of Western Ontario, Canada

Oct. 27, 2012
ASCM 2012, Beijing, China

Cylindrical Algebraic Decomposition (CAD) of $\mathbb{R}^n$

A CAD of $\mathbb{R}^n$ is a partition of $\mathbb{R}^n$ such that each cell in the partition is a connected semi-algebraic subset of $\mathbb{R}^n$ and all the cells are cylindrically arranged.

Two subsets A and B of $\mathbb{R}^n$ are called cylindrically arranged if for any $1 \leq k < n$, the projections of A and B on $\mathbb{R}^k$ are either equal or disjoint.



Cylindrical algebraic decomposition (CAD)

Invented by G.E. Collins in 1973 for solving Real Quantifier Elimination (QE) problems.

Previous work on CAD

Adjacency and clustering techniques (D. Arnon, G.E. Collins and S. McCallum 84), Improved projection operator (H. Hong 90; S. McCallum 88, 98; C. Brown 01), Partially built CADs (Collins and Hong 91, A. Strzeboński 00), Improved stack construction (G.E. Collins, J.R. Johnson, and W. Krandick), Efficient projection orders (A. Dolzmann, A. Seidl and T. Sturm 04), Making use of equational constraints (G.E. Collins 98; C. Brown and S. McCallum 05), Computing CAD via triangular decompositions (C. Chen, M. Moreno Maza, B. Xia and L. Yang 09), Preprocessing input by Gröbner bases (B. Buchberger and H. Hong 91; D.J. Wilson, R.J. Bradford, and J.H. Davenport 12), Set-theoretical operations by CAD (A. Strzeboński 10)$\cdots$

Software

QEPCAD, Mathematica, Redlog, SyNRAC, TCAD (Since Maple 14).

Outline

1. First Idea: Introduce Case Discussion
2. Second Idea: Compute the Decomposition Incrementally
3. Third Idea: Compute CAD of a Variety
4. Implementation and Benchmark

Outline

1. First Idea: Introduce Case Discussion
2. Second Idea: Compute the Decomposition Incrementally
3. Third Idea: Compute CAD of a Variety
4. Implementation and Benchmark

CAD based on projection-lifting scheme (PCAD)

Projection

- Let $Proj$ be a projection operator.
- Repeatedly apply $Proj$:

$$F_n(x_1, \ldots, x_n) \xrightarrow{Proj} F_{n-1}(x_1, \ldots, x_{n-1}) \xrightarrow{Proj} \cdots \xrightarrow{Proj} F_1(x_1).$$

Lifting

- The real roots of the polynomials in F_1 plus the open intervals between them form an F_1-invariant CAD of $\mathbb{R}^1$.
- For each cell C of the F_{k-1} invariant CAD of $\mathbb{R}^{k-1}$, isolating the real roots of the polynomials of F_k at a sample point of C, produces all the cells of the F_k-invariant CAD of $\mathbb{R}^k$ above C.

CAD based on triangular decompositions (TCAD)

Motivation: potential drawback of Collins' scheme

- The projection operator is a function defined independently of the input system.
- As a result, a strong projection operator (Collins-Hong operator) usually produces much more polynomials than needed.
- A weak projection operator (McCalumn-Brown operator) may fail for non-generic cases.

Solution: make case discussion during projection

- Case discussion is common for algorithms computing triangular decomposition.
- At ISSAC'09, we (with B. Xia and L. Yang) introduced case discussion (as in triangular decomposition of polynomials systems) into CAD computation. As a result, the projection phase in classical CAD algorithm is replaced by computing a complex cylindrical tree.

Complex cylindrical tree

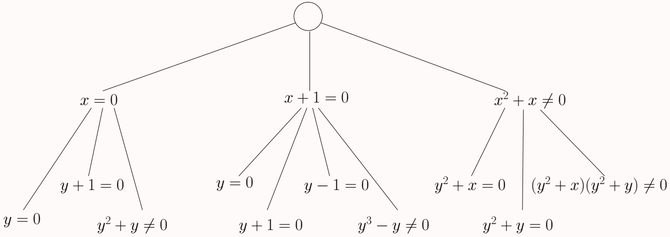
- let $\alpha = (\alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{C}^{n-1}$
- define $\mathbb{C}[x_1, \ldots, x_n] \xrightarrow{\Phi_\alpha} \mathbb{C}[x_n]$, where $p(x_1, \ldots, x_n) \mapsto p(\alpha, x_n)$

Separation

Let $S \subset \mathbb{C}^{n-1}$ and $P \subset \mathbf{k}[x_1, \ldots, x_{n-1}, x_n]$ be a finite set of level n polynomials. We say that P separates above S if for each $\alpha \in S$:

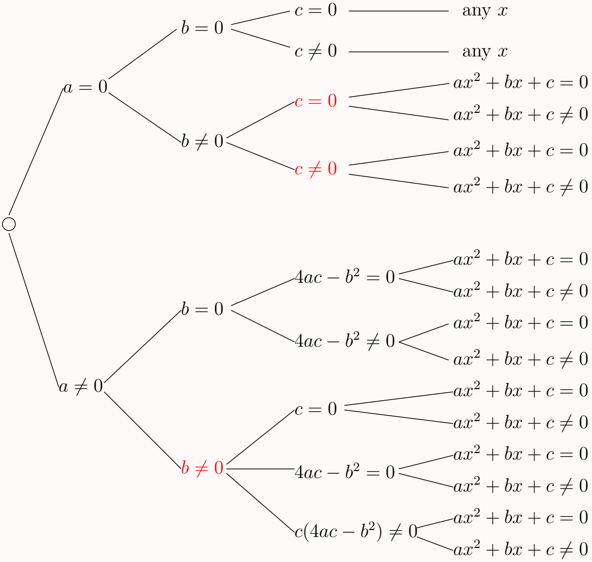
- for each $p \in P$, $\Phi_\alpha(\text{leading coefficient of p w.r.t. } \mathrm{x_n}) \neq 0$
- the polynomials $\Phi_\alpha(p)$ are squarefree and pairwise coprime.

A $\{y^2 + x, y^2 + y\}$-sign invariant complex cylindrical tree

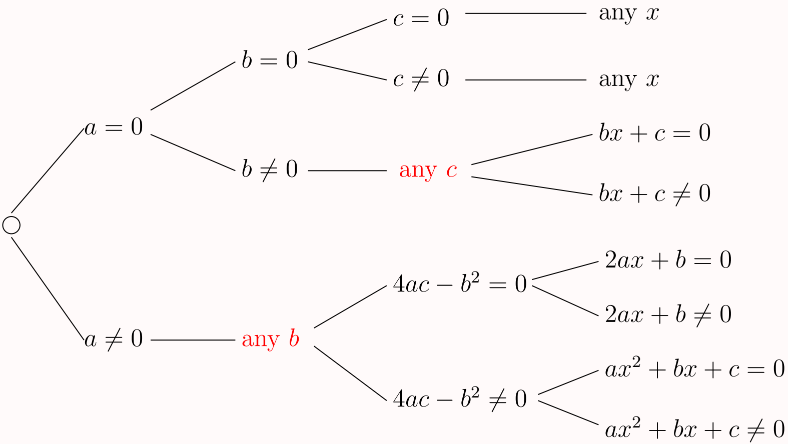


Rethink classical CAD in terms of complex cylindrical tree

The projection factors are $a, b, c, 4ac - b^2, ax^2 + bx + c$.



The complex cylindrical tree constructed by TCAD



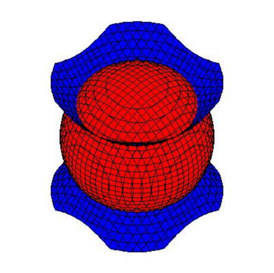
Outline

1. First Idea: Introduce Case Discussion
2. Second Idea: Compute the Decomposition Incrementally
3. Third Idea: Compute CAD of a Variety
4. Implementation and Benchmark

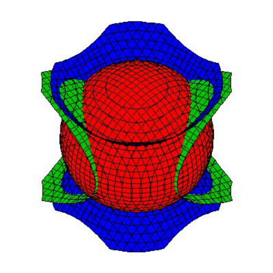
Incremental solving

- $p_1 := x^2 + y^2 + z^2 - 4$
- $p_2 := x^2 + y^2 - z^2 - 1$
- $p_3 := z^3 + xy - 1$

$W(T) := V(p_1) \cap V(p_2)$



$V(p_3) \cap W(T)$



Algorithms using incremental strategy: Triangular Decomposition (D. Lazard, 91; M^3, 00; C. Chen & M^3, 11); Lifting Fibers (G. Lecerf, 2003); Diagonal Homotopy (A.J. Sommese, J. Verschelde, C. W. Wampler, 08).

The refinement operation

Input

- A $y^2 + x$ sign invariant complex cylindrical tree

$$T := \begin{cases} x = 0 & \begin{cases} y = 0 & : \quad y^2 + x = 0 \\ y \neq 0 & : \quad y^2 + x \neq 0 \end{cases} \\ \\ x \neq 0 & \begin{cases} y^2 + x = 0 & : \quad y^2 + x = 0 \\ y^2 + x \neq 0 & : \quad y^2 + x \neq 0 \end{cases} \end{cases}$$

- A polynomial $y^2 + y$.

Output

The tree T is refined into a new tree, above each path of which both $y^2 + x$ and $y^2 + y$ are sign invariant.

Refine the first path of the tree with $y^2 + y$

$$\begin{cases} x = 0 & \begin{cases} y = 0 & : \quad y^2 + x = 0 \\ y \neq 0 & : \quad y^2 + x \neq 0 \end{cases} \\ & \\ x \neq 0 & \cdots \end{cases}$$

$$\Rightarrow \quad \begin{cases} x = 0 & \begin{cases} y = 0 & : \quad y^2 + x = 0 \wedge y^2 + y = 0 \\ y \neq 0 & : \quad y^2 + x \neq 0 \end{cases} \\ & \\ x \neq 0 & \cdots \end{cases}$$

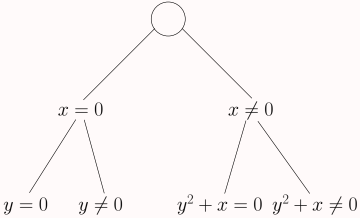
Refine the next path of the tree with $y^2 + y$

$$\begin{cases} x = 0 & \begin{cases} y = 0 & : \quad y^2 + x = 0 \wedge y^2 + y = 0 \\ y \neq 0 & : \quad y^2 + x \neq 0 \end{cases} \\ \\ x \neq 0 & \cdots \end{cases}$$

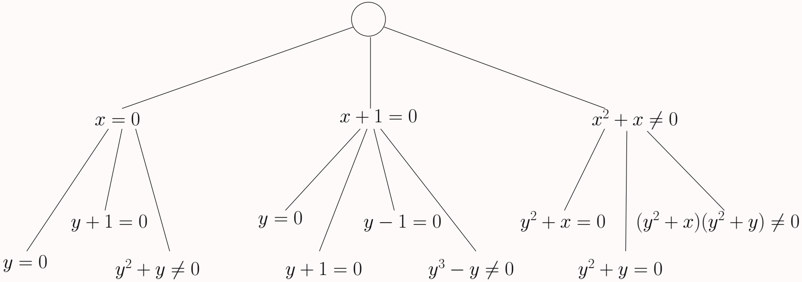
$$\Rightarrow \quad \begin{cases} x = 0 & \begin{cases} y = 0 & : \quad y^2 + x = 0 \wedge y^2 + y = 0 \\ y = -1 & : \quad y^2 + x \neq 0 \wedge y^2 + y = 0 \\ \text{otherwise} & : \quad y^2 + x \neq 0 \wedge y^2 + y \neq 0 \end{cases} \\ \\ x \neq 0 & \cdots \end{cases}$$

The $\{y^2+x, y^2+y\}$ sign invariant cylindrical tree of $\mathbb{C}^2$

$$
\begin{cases}
x=0 & \begin{cases} y=0 & : \; y^2+x=0 \wedge y^2+y=0 \\ y=-1 & : \; y^2+x\neq 0 \wedge y^2+y=0 \\ \text{otherwise} & : \; y^2+x\neq 0 \wedge y^2+y\neq 0 \end{cases} \\
x=-1 & \begin{cases} y=-1 & : \; y^2+x=0 \wedge y^2+y=0 \\ y=1 & : \; y^2+x=0 \wedge y^2+y\neq 0 \\ y=0 & : \; y^2+x\neq 0 \wedge y^2+y=0 \\ \text{otherwise} & : \; y^2+x\neq 0 \wedge y^2+y\neq 0 \end{cases} \\
\text{otherwise} & \begin{cases} y^2+x=0 & : \; y^2+x=0 \wedge y^2+y\neq 0 \\ y^2+y=0 & : \; y^2+x\neq 0 \wedge y^2+y=0 \\ \text{otherwise} & : \; y^2+x\neq 0 \wedge y^2+y\neq 0 \end{cases}
\end{cases}
$$



$x = 0$
$x \neq 0$
$y = 0$
$y \neq 0$
$y^2 + x = 0$
$y^2 + x \neq 0$



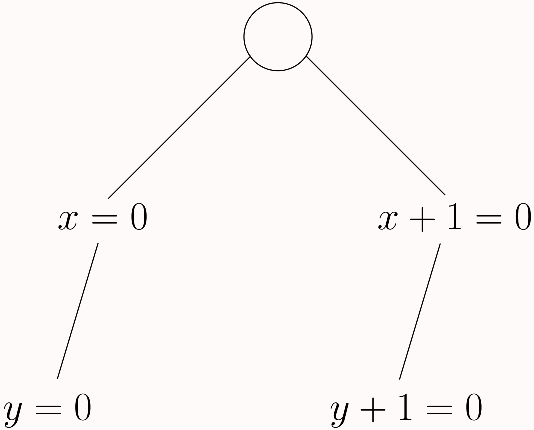
$x = 0$
$x + 1 = 0$
$x^2 + x \neq 0$
$y + 1 = 0$
$y = 0$
$y^2 + y \neq 0$
$y = 0$
$y - 1 = 0$
$y + 1 = 0$
$y^3 - y \neq 0$
$y^2 + x = 0$
$(y^2 + x)(y^2 + y) \neq 0$
$y^2 + y = 0$

Outline

1. First Idea: Introduce Case Discussion
2. Second Idea: Compute the Decomposition Incrementally
3. Third Idea: Compute CAD of a Variety
4. Implementation and Benchmark

Compute partial cylindrical tree

A partial cylindrical tree induced by the $F := \{y^2 + x = 0, y^2 + y = 0\}$ is



Transform a complex cylindrical decomposition to a real one

$$
\text{Complex}: \left\{\begin{array}{ll}
x = 0 & \left\{\begin{array}{lcl} y = 0 & : & y^2 + x = 0 \\ y \neq 0 & : & y^2 + x \neq 0 \end{array}\right. \\
 & \\
x \neq 0 & \left\{\begin{array}{lcl} y^2 + x = 0 & : & y^2 + x = 0 \\ y^2 + x \neq 0 & : & y^2 + x \neq 0 \end{array}\right.
\end{array}\right.
$$

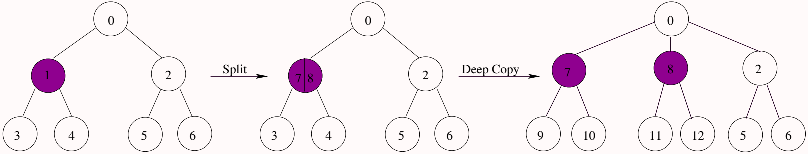
$$
\text{Real}: \left\{\begin{array}{ll}
x < 0 & \left\{\begin{array}{lcl} y < -\sqrt{|x|} & : & y^2 + x > 0 \\ y = -\sqrt{|x|} & : & y^2 + x = 0 \\ y > -\sqrt{|x|} \wedge y < \sqrt{|x|} & : & y^2 + x < 0 \\ y = \sqrt{|x|} & : & y^2 + x = 0 \\ y > \sqrt{|x|} & : & y^2 + x > 0 \end{array}\right. \\
 & \\
x = 0 & \left\{\begin{array}{lcl} y < 0 & : & y^2 + x > 0 \\ y = 0 & : & y^2 + x = 0 \\ y > 0 & : & y^2 + x > 0 \end{array}\right. \\
 & \\
x > 0 & \text{for any } y : \quad y^2 + x > 0
\end{array}\right.
$$

Outline

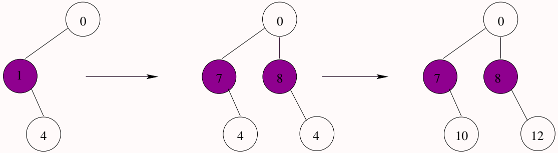
1. First Idea: Introduce Case Discussion
2. Second Idea: Compute the Decomposition Incrementally
3. Third Idea: Compute CAD of a Variety
4. Implementation and Benchmark

Implementation in Maple

The universe tree is always up-to-date



A sub-tree evolves with the universe tree



A typical sub-algorithm

- p, f : polynomials of level n.
- S: the subresultant chain S of p and f w.r.t. x_n.
- s: the principle subresultant coefficients of p and f
- d, i : two non-negative integers such that s_d is invertible modulo Γ and s_j is zero modulo Γ, for all $0 \leq j < i$.
- T: a cylindrical tree of $\mathbf{k}[x_1 < \cdots < x_{n-1}]$; Γ: a path of T.

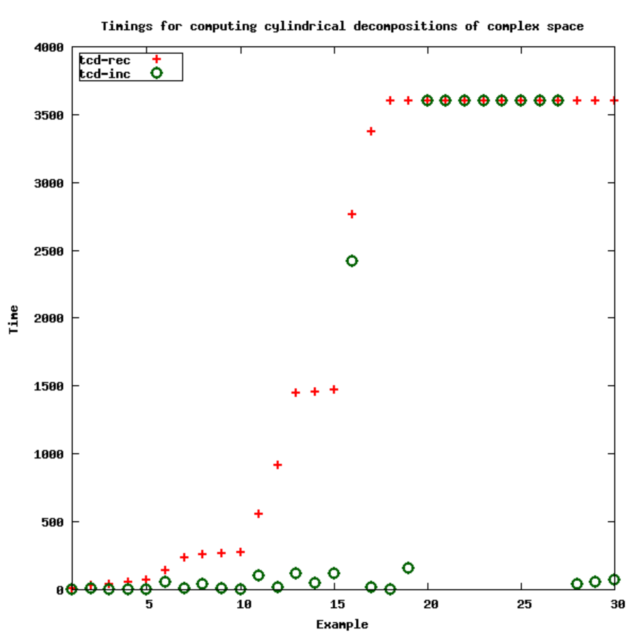
A refined tree T such that above each path C of T derived from Γ, $C.leaf.Gcd[p, f]$ is a GCD of p and f modulo C.

Algorithm 1: RegularGcd$(p, f, S, d, i, \Gamma, T)$

```
if i = d then
    Γ.leaf.Gcd[p, f] := S_i; return;
IntersectPath_{n-1}(s_i, Γ, T);
while C := NextPathToDo_{n-1}(Γ) ≠ ∅ do
    if C.leaf.signs[s_i] = 1 then
        if i = 0 then  C.leaf.Gcd[p, f] := 1 ;
        else  C.leaf.Gcd[p, f] := S_i
    else  RegularGcd(p, f, S, d, i + 1, C, T)
```

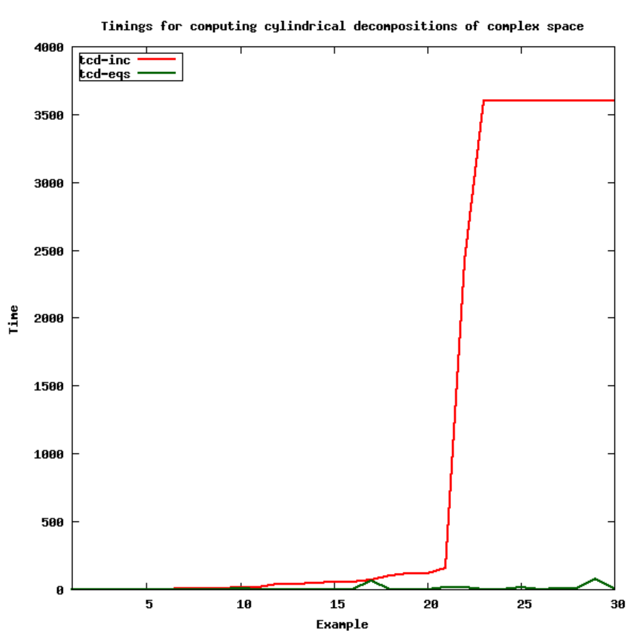
tcd-rec : our ISSAC'09 recursive algorithm

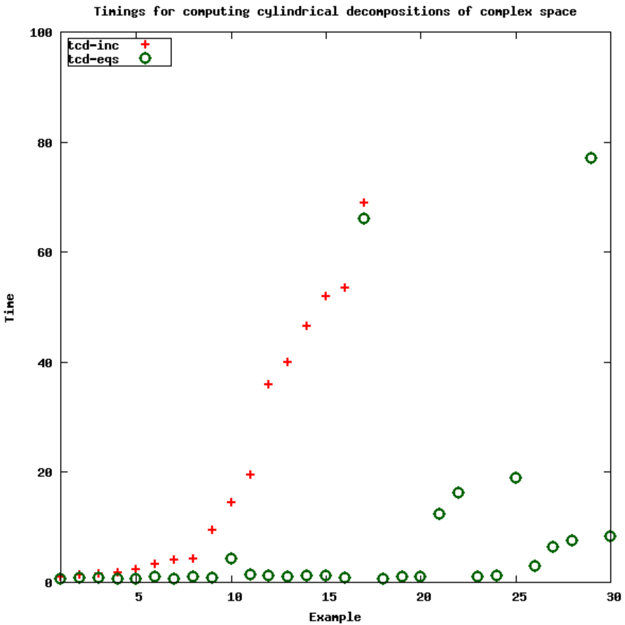
tcd-inc : the incremental algorithm

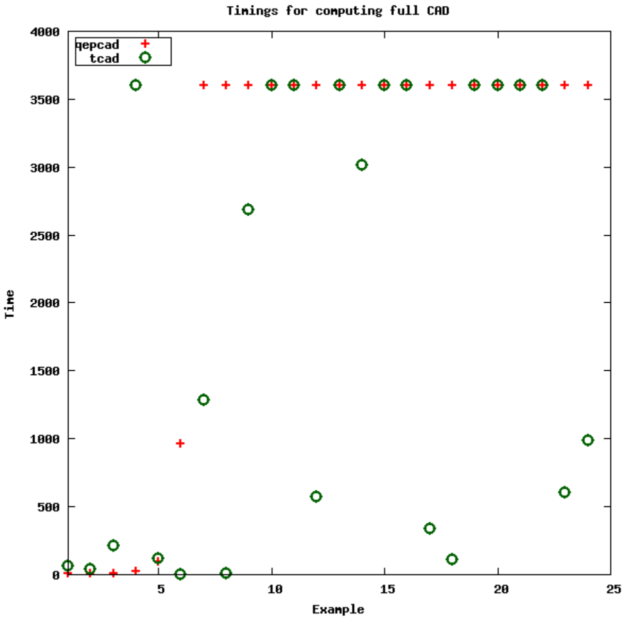


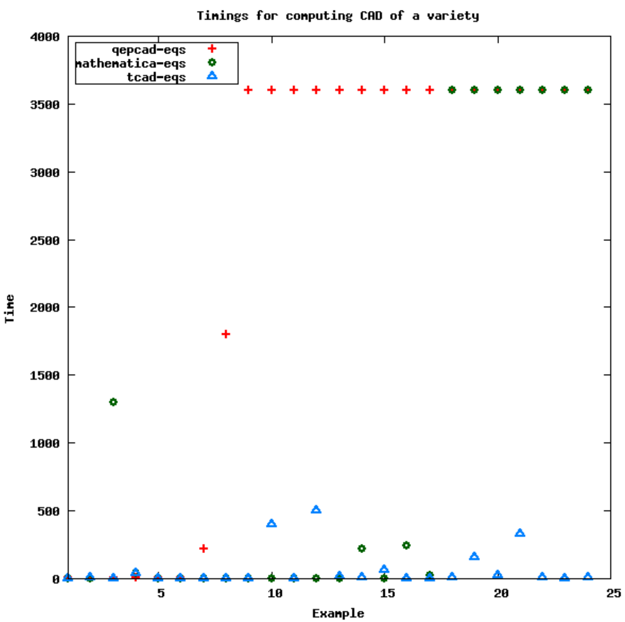
tcd-inc: the incremental algorithm with set of polynomials as input

tcd-eqs: the incremental algorithm with set of equations as input




Timings for computing cylindrical decompositions of complex space
tcd-inc
tcd-eqs
Time
Example
0
20
40
60
80
100
5
10
15
20
25
30


Timings for computing full CAD
qepcad
tcad
Time
Example
4000
3500
3000
2500
2000
1500
1000
500
0
5
10
15
20
25


Timings for computing CAD of a variety
qepcad-eqs
mathematica-eqs
tcad-eqs
Time
Example
4000
3500
3000
2500
2000
1500
1000
500
0
5
10
15
20
25

Conclusion and work in progress

Conclusion

- We presented an incremental algorithm for computing CADs.
- The core operation of our algorithm is an Intersect operation, which refines a complex cylindrical tree by means of a polynomial constraint.
- The Intersect operation provides a systematic solution for propagating equational constraints.
- For many examples, the incremental outperforms both QEPCAD and Mathematica as well as our previous recursive algorithm.

Work in progress

- We have developed a preliminary QE routine QETCAD based on TCAD.
- We are working on different optimizations for both TCAD and QETCAD.