A Practical Implementation of a Symbolic-Numeric Cylindrical Algebraic Decomposition for Quantifier Elimination

Iwane, Hidenao
Fujitsu Laboratories Ltd

Yanami, Hitoshi
Fujitsu Laboratories Ltd

Anai, Hirokazu
Kyushu University | Fujitsu Laboratories Ltd

Yokoyama, Kazuhiro
Rikkyo University

出版情報：MI Preprint Series. 2009-13, 2009-03-24. 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

A Practical Implementation of a Symbolic-Numeric Cylindrical Algebraic Decomposition for Quantifier Elimination

H. Iwane, H. Yanami, H. Anai & K. Yokoyama

MI 2009-13

(Received March 24, 2009)

Faculty of Mathematics
Kyushu University
Fukuoka, JAPAN

A Practical Implementation of a Symbolic-Numeric Cylindrical Algebraic Decomposition for Quantifier Elimination

Hidenao Iwane
Fujitsu Laboratories Ltd
4-1-1 Kamikodanaka,
Nakahara-ku, Kawasaki
211-8588, Japan
iwane@jp.fujitsu.com

Hitoshi Yanami
Fujitsu Laboratories Ltd
4-1-1 Kamikodanaka,
Nakahara-ku, Kawasaki
211-8588, Japan
yanami@labs.fujitsu.com

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

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

ABSTRACT

Recently quantifier elimination (QE) has been of great interest in many fields of science and engineering. In this paper an effective symbolic-numeric cylindrical algebraic decomposition (SNCAD) algorithm and its variant specially designed for QE are proposed based on the authors' previous work and our implementation of those is reported.

Based on analysing experimental performances, we are improving our design/synthesis of the SNCAD for its practical realization with existing efficient computational techniques and several newly introduced ones.

The practicality of the SNCAD is now examined by experimentation on real computer, which also reveals the quality of the implementation.

1. INTRODUCTION

Cylindrical algebraic decomposition (CAD) is a general-purpose symbolic method aiming for quantifier elimination (QE) which is a powerful tool to resolve non-convex and parametric optimization problems exactly. However, QE based on CAD is not considered to be practical on real computers, since CAD usually consists of many purely symbolic computations and has a bad computational complexity in nature.

To circumvent the inherent computational complexity of QE algorithm based on CAD, several researchers have focused on QE algorithms specialized to particular types of input formulas; see [35, 21, 24, 37, 19]. This direction is quite promising in practice since a number of important problems in engineering have been successfully reduced to such particular input formulas and resolved by using the specialized QE algorithms. See concrete successful applications in [36, 34, 13, 18, 1, 3]. Moreover, using discriminant varieties seems another promising direction; see [17].

However, there still remain many significant problems in engineering that cannot be recast as such particular formulas. Therefore, it is strongly desired to develop an efficient algorithm for CAD. One of the crucial parts of CAD construction lies in the lifting phase. Computational difficulties in the lifting phase stem from symbolic computation over towers of algebraic extensions and combinatorial explosion in CAD construction of $\mathbb{R}^n$. An effective way for efficient CAD construction is to utilize numerical computation, instead of symbolic treatment, with derived numerical information on algebraic numbers as far as possible without violating correctness of the results. So far there have been some attempts to introduce numerical computation into CAD construction: for example, Hong, 1993 [20], Strzeboński, 1999 [31], Collins et al., 2002 [12], Ratchan, 2002 [28], Anai and Yokoyama, 2005 [5], and Strzeboński, 2006 [32].

In this paper, we employ a scheme for SNCAD roughly introduced in [5], which uses in the lifting phase certified numerical computation over algebraic extensions and the dynamic evaluation technique in [15] with successive representations of algebraic extensions. As far as the authors know, the scheme was first proposed by [5] and has not yet implemented until the present. Here we propose an advanced SNCAD scheme by incorporating several new devices nicely in a concrete way into the original scheme in [5].

Moreover, to examine our new scheme, we implemented the whole algorithms for SNCAD and QE based on SNCAD on Maple (as a part of SyNRAC [39]). Finally, we report the details of our implementation and its experimentation on real computer to evaluate the effectiveness of the proposed scheme.

In order to accomplish more effective and practical implementation of algebraic algorithms, a research direction towards

hybrid methods combining computer algebra with numerical verification methods have been pursued [29, 30]. Viewing the symbolic-numeric CAD methods from another side, one can see that such approach provides new validated numerical methods for non-convex optimization problems which are a difficult task for ordinary numerical methods, with the help of symbolic computation. Both aspects are of surely practical importance. Thus our work would be one of a successful example of such direction.

The paper is organized as follows. Our scheme for an efficient SNCAD algorithm and QE based on SNCAD are proposed in Section 2. Then the computational flow and several key algorithms for realizing our SNCAD scheme are presented in Section 3. Section 4 is devoted to some remarks on our implementation on Maple and the experimental results are given in Section 5. Discussion and some concluding remarks are made in Section 6.

2. A SCHEME FOR EFFICIENT SYMBOLIC-NUMERIC CAD COMPUTATION

2.1 Standard CAD algorithms

We briefly sketch the basic ideas of CAD; see [10] for details. Assume that we are given an input formula φ with m free variables $u_1, \ldots, u_m$ and n quantified variables $x_1, \ldots, x_n$

$$\varphi(u_1, \ldots, u_m) \equiv \mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n \ \psi(u_1, \ldots, u_m, x_1, \ldots, x_n),$$

where $\mathsf{Q}_i \in \{\exists, \forall\}$ and ψ is a quantifier-free formula. We can assume, by transporting terms if necessary, each atomic formula in ψ is represented in the form $f\,\rho\,0$, where f is a polynomial on $u_1, \ldots, u_m, x_1, \ldots, x_n$ and $\rho \in \{\leq, <, =, \neq\}$. Let F be the set of polynomials appearing in ψ as the left hand sides of atomic formulas. A subset $C \subseteq \mathbb{R}^{m+n}$ is said to be *sign-invariant* for F if every polynomial in F has a constant sign on all points in C. Then $\psi(c)$ is either "true" or "false" for all $c \in C$.

Suppose we have a finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$ for F which has the following properties:

1. Each $\mathcal{D}_i$ is a partition of $\mathbb{R}^i$ into finitely many connected semi-algebraic sets called *cells*. For $1 \leq j \leq n$ each $\mathcal{D}_{m+j}$ is labeled with Q_j.

2. $\mathcal{D}_{i-1}$, $1 < i \leq m+n$, consists exactly of the projections of all cells in $\mathcal{D}_i$ along the coordinate of the i-th variable in $(u_1, \ldots, u_m, x_1 \ldots x_n)$. For each cell $C \in \mathcal{D}_{i-1}$ we can determine its preimage $S(C) \subseteq \mathcal{D}_i$ under the projection.

3. Each cell $C \in \mathcal{D}_{m+n}$ is sign-invariant for F. Moreover for each cell $C \in \mathcal{D}_{m+n}$ we are given a *sample point* t_C in such a form that we can determine the sign of $f(t_C)$ for each $f \in F$ and thus evaluate $\varphi(t_C)$.

Then the partition $\mathcal{D}_{m+n}$ of $\mathbb{R}^{m+n}$ for F is called an *F-invariant cylindrical algebraic decomposition* of $\mathbb{R}^{m+n}$. The CAD algorithm computes such a sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$ and it consists of three phases; *projection phase*, *base phase* and *lifting phase*.

Projection phase: We first construct from $F \subset \mathbb{R}[u_1, \ldots, u_m, x_1, \ldots, x_n]$ a new finite set $F' \subset \mathbb{R}[u_1, \ldots, u_m, x_1, \ldots, x_{n-1}]$ that satisfies a special condition called "delineability", where the order of the real roots of all polynomials in F as univariate polynomials in x_n does not change above each connected cell in $\mathcal{D}_{m+n-1}$.

The step constructing F' from F is called a *projection* and denoted by $F' := PROJ(F)$. We call polynomials in F' *projection polynomials* and those irreducible factors *projection factors*. Iterative application of $PROJ$ operator leads to a finite sequence

$$F_{m+n}, \ldots, F_1, \quad \text{where} \quad F_{m+n} := F, \ F_i := PROJ(F_{i+1})$$

for $1 \leq i < m+n$. The $PROJ$ operator, in general, computes certain coefficients, discriminants, resultants, and subresultant coefficients derived from polynomials in F_{i+1} and their higher derivatives by regarding those as univariate polynomials in their last variable. The final set F_1 consists of univariate polynomials in u_1.

Base phase: Then we construct a partition $\mathcal{D}_1$ of the real line $\mathbb{R}^1$ into finitely many intervals that are sign-invariant for F_1. This step is called *base phase* and achieved by isolating the real zeros of the univariate polynomials in F_1.

Lifting phase: The partitions $\mathcal{D}_i \subseteq \mathbb{R}^i$ for $2 \leq i < m+n$ are computed recursively: The roots of all polynomials in F_i as univariate polynomials in their last variable are delineated above each connected cell C in $\mathcal{D}_{i-1}$. Thus we can cut the *cylinder* above C into finitely many connected semi-algebraic sets (cells). This is done by real root isolation of the univariate polynomials derived through specializing the polynomials in F_i by a sample point of C. Then $\mathcal{D}_i$ is a collection of all such cells obtained from all cylinders above the cells of $\mathcal{D}_{i-1}$.

A finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$ for F has a tree-structure: The first level of nodes under the root of the tree corresponds to the cells in $\mathcal{D}_1$. The second level of nodes stands for the cells in $\mathcal{D}_2$, i.e., the cylinders over the cells of $\mathbb{R}^1$. The leaves represent the cells of $\mathcal{D}_{m+n}$, i.e., a CAD of $\mathbb{R}^{m+n}$. A sample point of each cell is stored in its corresponding node or leaf. At each level of the tree there are a number of projection polynomials F_i whose signs define a cell when evaluated over a sample point.

In the subsequent sections we propose a scheme for an efficient CAD algorithm which incorporates certified numerical computation effectively. Moreover we mention further improvement of the proposed scheme for executing QE.

2.2 Our SNCAD scheme

In this section we summarize the fundamental concept of our SNCAD scheme (particularly for the lifting phase) that is an improved version of the scheme proposed in [5]. For a projection operator $PROJ$, we employ an existing one. (In our implementation we use the projection in [7].)

As far as the authors know, the core idea of the scheme was first suggested by [5] and had not been completely implemented till this time. The features of our scheme are listed below. We note that we in fact exploit some new organized combinations among the listed items for efficient CAD construction based on symbolic-numeric computation, in which the main novelty of this work consists. The technical details will be explained in §3.

1. **Symbolic-numeric computation with switchover strategy:** In the lifting phase, we replace heavy symbolic computations related to the arithmetic of algebraic numbers with numerical arithmetic, by which the total efficiency should be greatly improved.

   In our principle "switchover strategy", we use certified numerical computations based on interval computation as long as we can ensure the accuracy of the results. If numerical computations cannot confirm the validity of the output or turn out to be ill-conditioned for those verification, then we switch those to symbolic counterparts, that is, symbolic computations over algebraic extension fields, or improve the precision of intervals with the help of *symbolically computed parts*.

   For this reconstruction, we devise a special way of expressing algebraic numbers and make good use of efficient representation of algebraic extension fields.

2. **Expression of algebraic numbers:** Each algebraic number is expressed by a pair of its defining polynomial (symbolic part) and an isolating interval (numerical part). The defining polynomial is a triangular system of polynomials where the main variable in the last polynomial corresponds to a specified algebraic number. This expression corresponds quite naturally to our successive representation of algebraic extension fields.

   For numerical arithmetics with algebraic numbers, we use corresponding isolating intervals for those values instead of those symbolic expressions. Consequently, real root isolation of a polynomial over an algebraic extension can be reduced to that of a polynomial with interval coefficients over the base field $\mathbb{Q}$. If we fail to do the root isolation of such a polynomial, we improve the precision of the intervals using defining polynomials and try it again.

   We also use interval arithmetic for sign determination of algebraic numbers, while for zero determination we use "symbolic reconstruction" which consists of exact, symbolic methods such as polynomial division and GCD computation. Details will be given in §2.4

3. **Successive representation with dynamic evaluation:** In the lifting phase, each algebraic number appears successively as a zero of a polynomial over an extension field generated by adjoining already computed algebraic numbers. Thus, in the mathematical sense, we have to deal with algebraic extension fields which are generated successively. For handling those, we use successive representation of algebraic extension fields, as it is suited naturally to procedures in the lifting phase. Also, from a computational point of view, in general, successive representation of algebraic extension fields is superior to representation by primitive elements, since the latter is frequently suffered from overwhelming coefficient swells which make the computational efficiency very bad.

   We utilize dynamic evaluation for dealing with algebraic extensions efficiently in combination with successive representation. By dynamic evaluation technique, we can avoid heavy computation of algebraic factorization of polynomials and we only need square-free computation of polynomials which can be computed rather efficiently by a fast method for GCD computation.

2.3 Sensible use of SNCAD for efficient QE

For performing quantifier elimination, we do not have to compute a full CAD. Therefore, saving CAD construction is of interest for improving the efficiency. Of course, we can use the same ideas of "partial CAD" proposed in [11], which exploits the logical structure of an input QE problem systematically.

Furthermore, we are able to strategize how to avoid unnecessary computations in the lifting phase by using numerical information on algebraic numbers. The details of such strategies which we actually employ are shown in §3.2.

It is noted that we employ an existing algorithm for solution formula construction. (We actually use the algorithm proposed in [6] for our implementation.)

2.4 Remarks on symbolic reconstruction

Using numerical computation for handling algebraic numbers can greatly improve the efficiency of CAD construction because we can avoid symbolic computations over algebraic extension fields and also prune unnecessary branches of a CAD tree effectively by numerical values of algebraic numbers, while it may cause uncertainty of the computed results depending on the accuracy of numerical computation. Therefore, along our *switchover strategy*, we also use such numerical computation with a machinery for validating numerically computed results by *symbolic reconstruction* which reconstructs them exactly with a smaller number of symbolic computations.

Actually computations over algebraic extension fields are required in the lifting phase to compute respective isolating intervals of algebraic numbers over successive extension fields. What we need to do is "sign determination" of polynomials

substituted those variables with given algebraic numbers, i.e., to determine exactly whether they are zero or not and their sign if nonzero. We may usually expect that sign determinations are mostly properly done by only numerical computation. We check again the sign exactly by symbolic computation only for unreliable numerical results This is the ground why our method would improve the efficiency of *standard* CAD construction.

Moreover the reconstruction procedure is improved by dynamic evaluation technique integrated with successive representations of algebraic extensions as follows: An algebraic extension of $\mathbb{Q}$ is expressed by a residue class ring $\mathbb{Q}[X]/\mathcal{M}$, where X is a set of variables and $\mathcal{M}$ is a maximal ideal in $\mathbb{Q}[X]$. Usually, computation of maximal ideals tends to be very hard. Instead of $\mathbb{Q}[X]/\mathcal{M}$, we utilize "lazy representation" $\mathbb{Q}[X]/\mathcal{J}$, where $\mathcal{J}$ is not necessarily a maximal ideal but an easily computable radical one. $\mathbb{Q}[X]/\mathcal{J}$ may not be a domain, i.e., it could have zero-divisors. In the computation over $\mathbb{Q}[X]/\mathcal{J}$, if a given algebraic number does not correspond to a zero-divisor, then it is not equal to zero and we come to check its sign by using a certain numerical method. If we meet some zero-divisors $ab = 0$, then we can split the ideal $\mathcal{J}$ as follows:

$$\mathcal{J} = (\mathcal{J} + \langle a \rangle) \cap (\mathcal{J} + \langle b \rangle).$$

This decomposition is achieved by GCD computation or Gröbner basis computation. Thus, by using lazy representation successively for towers of extensions, we do not require any algebraic factorization or primitive element computation for a simple extension which is often difficult especially in the case of tall towers of extensions. Furthermore the most crucial point is that we can easily choose one necessary branch, say, $\mathcal{J} + \langle a \rangle$ in the decomposition by virtue of numerical information "isolating intervals" of algebraic numbers and hence can prune unnecessary branches for the further computation of towers of extensions.

3. ALGORITHMIC DETAILS

In this section we first summarize the computational flow in the whole lifting phase and then explain some key algorithms concretely for realizing our SNCAD scheme in an effective manner. Moreover, we also mention some improvements for efficient QE by direct evaluation of input formulas based on interval arithmetic.

3.1 SNCAD

3.1.1 Whole computational flow of lifting phase

Now we present the whole procedure of our lifting phase LiftingSNCAD(). First let us note again that each sample point, which is an algebraic number in general, is given by a pair of an isolating interval and its defining polynomial. A function IsEqualSymbolic() checks whether two isolating intervals stand for the same algebraic number in a symbolic sense by using their defining polynomials.

A function IntervalRealRootIsolation() is a significant portion of the whole procedure. Its essential point is that we achieve real root isolation for *interval polynomials* which are derived from projection factors by substituting isolating intervals into the corresponding algebraic numbers appearing in their coefficients. From now on, we call such a polynomial with interval coefficients an *interval polynomial*. In §3.1.2 we will explain how we compute isolating intervals of real roots for interval polynomials concretely.

LiftingSNCAD($[X_1, \ldots, X_n], [f_1, \ldots, f_m]$)
Input: sample points $X_1, \ldots, X_n \in \mathbb{R}^k$,
projection factors $f_1, \ldots, f_m \in \mathbb{Q}[x_1, \ldots, x_k, x]$
Output: lifted sample points $\in \mathbb{R}^{k+1}$

```
for i = 1 to n do
    for j = 1 to m do
        Y_ij ← IntervalRealRootIsolation(f_j, X_i)
    for j = 1 to m − 1,  l = j + 1 to m do
        if (the intervals Y_ij and Y_il overlap) then
            if IsEqualSymbolic(Y_ij, Y_il) then
                remove Y_il from the list
            else
                increase precision of Y_ij, Y_il until separated

return {[X_i, Y_ij]}
```

3.1.2 Real root isolation for interval polynomials

Next we explain details on the function IntervalRealRootIsolation() which executes real root isolation for an interval polynomial concretely. In the function, actually in RealRootIsolation(), the well-known Krawczyk method [22, 25, 26] is used for isolating real roots of polynomials.

However, as the Krawczyk method fails in isolating real roots for non square-free polynomials, we provide two basic functions IntervalRealRootIsolationSqrfree() and SymbolicSqrfree() in the function, where IntervalRealRootIsolationSqrfree() isolates real roots for square-free interval polynomials and SymbolicSqfree() computes the square-free decomposition of polynomials in a symbolic manner.

Also IntervalRealRootIsolationSqrfree()
fails if the precision for intervals is insufficient. In such case, we increase the precision by using defining polynomials.

In our scheme since we use dynamic evaluation for representing algebraic extensions, the defining polynomial of an algebraic number to be newly adjoined to a base field is needed to be only square-free. This is in fact suitable for our scheme. We just need symbolic square-free decomposition SymbolicSqrfree() and thus we can avoid expensive symbolic computation such as an irreducibility check or irreducible factorization of polynomials.

The algorithm shown below is for isolating positive real roots. Actually the same algorithm is applied to $f(-x)$ for isolating negative real roots.

IntervalRealRootIsolation(f, X)
Input : sample point $X \in \mathbb{R}^k$,
projection factor $f(x_1, \ldots, x_k, x) \in \mathbb{Q}[x_1, \ldots, x_k, x]$
Output: isolating intervals of the roots in $x > 0$ of $f(X, x)$,
square-free part of f

$g \leftarrow f$
$F \leftarrow g(X, x)$
$r \leftarrow IntervalRealRootIsolationSqrfree(F)$
if error then
  $g \leftarrow SymbolicSqrfree(g)$
  $F \leftarrow g(X, x)$
  while
    $r \leftarrow IntervalRealRootIsolationSqrfree(F)$
    if not error then
      break
    increase precision of X
return r, g

Finally we illustrate how IntervalRealRootIsolationSqrfree() accomplishes real root isolation of an interval polynomial (derived from a square-free polynomial).

Let $f(x_1, \ldots, x_k, x) \in \mathbb{Q}[x_1, \ldots, x_k, x]$ be a projection factor, $X \in \mathbb{R}^k$ a sample point, $[l, h]$ an interval and $F(x) = \sum_{i=0}^{d} [l_i, h_i] x^i$ an interval polynomial which derived from $f(x_1, \ldots, x_k, x)$ by substituting the interval expression of each component X for its corresponding variable in f.

We set $F_h(x) = \sum_{i=0}^{d} h_i x^i$ and $F_l(x) = \sum_{i=0}^{d} l_i x^i$. Then we have $F_l(x) \leq F(x) \leq F_h(x)$ when $x > 0$. We denote the positive real roots of $F_h(x)$ by $\alpha_1 < \cdots < \alpha_{t_h}$ and those of $F_l(x)$ by $\beta_1 < \cdots < \beta_{t_l}$. When $F_h(x)$ and $F_l(x)$ have the same number of positive real roots, i.e., $t_h = t_l$, the positive real roots of $F(x)$ may be given by $[\min(\alpha_1, \beta_1), \max(\alpha_1, \beta_1)]$, ..., $[\min(\alpha_{t_h}, \beta_{t_l}), \max(\alpha_{t_h}, \beta_{t_l})]$.

If the precisions of interval coefficients of $F(x)$ are not sufficiently small, there might occur the following ill-conditioned cases for real root isolation:

1. The numbers of real roots t_h of $F_h(x)$ is not equal to that t_l of $F_l(x)$,
2. $t_h = t_l$ but $\max(\alpha_i, \beta_i) \geq \min(\alpha_{i+1}, \beta_{i+1})$ for some i,
3. $t_h = t_l$ and $\max(\alpha_i, \beta_i) < \min(\alpha_{i+1}, \beta_{i+1})$ for every i, but the interval $[\min(\alpha_i, \beta_i), \max(\alpha_i, \beta_i)]$ contains two or more real roots of $f(X, x)$.

In order to check the third case we need to verify whether f has only one real root in the given interval or not. This is done by a function CheckOnlyOneRoot(), which is efficiently done by the following manner: First it checks if values of derivatives of f in the given interval is constant. If so, it is shown that f has a unique root in the give interval. Otherwise, it counts the number of real roots based on *Descartes' rule of sign*, where the given interval is transformed to $(0, \infty)$.

It is noted that if the sign of each coefficient of an input interval polynomial is definite (i.e., $l \cdot h > 0$ or $l = h = 0$, this is easily accomplished), we can expect that real root isolation is more efficient, because extension of intervals is generally-restrained.

IntervalRealRootIsolationSqrfree(F)
Input : interval polynomial $F(x) = \sum [l_i, h_i] x^i$
Output: isolating intervals of the roots of $F(x)$ $(x > 0)$

$F_h \leftarrow \sum h_i x^i, \quad F_l \leftarrow \sum l_i x^i$
$a_h \leftarrow RealRootIsolation(F_h)$
$a_l \leftarrow RealRootIsolation(F_l)$
if $\#(a_l) \neq \#(a_h)$ then error
for $i = 1$ to $\#(a_l)$ do
  $b[i] \leftarrow [\min(a_l[i, 1], a_h[i, 1]), \max(a_l[i, 2], a_h[i, 2])]$
for $i = 2$ to $\#(b)$ do

if $b[i-1,2] > b[i,1]$ then error
for $i = 1$ to $\#(b)$ do
if $CheckOnlyOneRoot(F, b[i]) =$ false then error
return b

3.2 QE based on SNCAD

Since our final goal is to perform efficient QE, we generally do not need to compute a full CAD. Therefore, saving CAD construction is of interest for improving the efficiency. Now we show our strategy by avoiding unnecessary computations via interval arithmetic. Though such strategy cannot be always applicable, possible situations are rather often the case with real-world problems in science and engineering. In fact they are applied to most of our examples in §5.

(1) We exploit some features in the structure of input polynomials for simplifying the problem before CAD construction. For example, when we have a polynomial in which each of the variables is bounded by an interval (e.g. $0 < a < 1$), we might have a chance to determine its sign by using interval arithmetic easily.

Example: Consider the following QE problem over $\mathbb{R}$;

$$\forall a \forall b (0 < a < 1 \wedge 0 < b < 1 \Rightarrow (a - b - 2)F(a, b) \geq 0).$$

We can easily see $-3 < a - b - 2 < -1$ provided that $0 < a < 1$ and $0 < b < 1$. Therefore the input formula is equivalent to

$$\forall a \forall b (0 < a < 1 \wedge 0 < b < 1 \Rightarrow F(a, b) \leq 0).$$

(2) There might be cases where we can determine an interval (possibly two or more intervals) whose truth value is readily computed from an input formula by using numerical arithmetic before CAD construction. Then we do not generate the cells which lies in the interval and hence we can omit not only to check if a pair of algebraic numbers are the same and but to lift such cells.

Example: Consider the following QE problem over $\mathbb{R}$;

$$\forall x \forall y \ \left(x \geq 0 \ \vee \ x^2 + y^2 \geq 1 \vee f(x, y) < 0\right).$$

We obtain the intervals $[-\infty, -1], [0, \infty]$ for x where the formula is true along the following:

$$x^2 + y^2 \geq 1 \Rightarrow x^2 + [0, \infty] \geq 1 \Rightarrow x^2 \geq [-\infty, 1].$$

4. REMARKS ON IMPLEMENTATION

We have implemented all algorithms presented in the previous sections §2 and §3 (except some advanced ones in §2.4) on Maple as a part of SyNRAC. As for projection phase and solution formula construction, we employ the projection function proposed by [7] and the formula construction algorithm shown in [6]. We list up some remarks on our implementations.

Interval computation: We have developed our own package for interval computations, since we aim at making a more efficient and effective interval arithmetic for our purposes. Moreover, as the Maple interval package "INTERVAL" sometimes outputs wrong results which might cause serious problems in our computation, we have provided our original one. In the below, we show an example of wrong outputs, where the number of significant digits exceeds the setting for the Digits environmental variable.

```
> Digits, Rounding;
                10, nearest
> evalr(INTERVAL(0.3333333333333..0.3333333333334)*3);
                0.9999999999
```

Real root isolation: For isolating real roots of a polynomial we use Krawczyk's method. We use its implementation in C language by us which is available within double precision. Usually Krawczyk's method is more efficient than a Maple command "realroot". As Krawczyk's method is called many times, this greatly improves the efficiency.

Boolean expressions: Since we definitely need an efficient logic package suited for our purposes, we have been developing our own commands for boolean expressions (see [38, 39]). Such boolean functions are _ALL, _EX, _OR, and _AND whose sample outputs are shown in the following:

```
> syn_qe(_ALL(x, a*x^2+b*x+c>0), [a,b,c,x]);
  _OR(_AND(b = 0, 0 < c, 0 <= 4*a*c-b^2),
      _AND(0 <= c, 0 < 4*a*c-b^2))

> syn_qe(_EX(y, _AND(x^2+y^2=1, x+y<0)), [a,b,c,x]);
  _OR(2 x^2  < 1, _AND(x <= 0, 0 <= x + 1))

> syn_qe(_EX([z],_AND(x^2+y^2+z^2=1,x+2*y+z<1)),[x,y,z]);
  _OR(_AND(0 <= 1+x, 5*x < -3, x^2+y^2 <= 1),
      _AND(0 <= 1+x, x <= 1, 2*x^2+5*y^2+4*x*y-2*x-4*y < 0),
      _AND(0 <= 1+x, x <= 1, 5*y+2*x < 2, x^2+y^2 <= 1))
```

5. EXPERIMENTATION

In this section we show our experiment on several benchmark problems including some engineering problems to demonstrate the effectiveness and efficiency of our SNCAD and QE by SNCAD and our implementation.

Comparisons with some QE implementations based on CAD (SyNRAC, QEPCAD B 1.50 [8], REDLOG 3.0 [14] and Mathematica 6) are shown in Table 1. All the computation is executed on a PC with Intel(R) Core Solo CPU U1500 1.33 GHz and 1 GB memory. All timing data are given in second. QEPCAD was executed with option "+N40000000 +L10000".

In Table 1, symbols SyN, QEP, RED and MATH stand for SyNRAC, QEPCAD B, REDLOG and Mathematica, respectively and SyN2 means SyNRAC implementation without a dynamic evaluation.

Table 2 shows the number of cells produced in CAD for each problem. We could not know the numbers of cells produced by REDLOG and Mathematica had constructed. In the first four examples the numbers of cells produced by Mathematica are taken from [32].

Table 3 tells us the numbers of occurrences of important key events in our algorithm. The first column "precision" shows the number of times raising the precision. The second one "zero check" represents "the number of the case the result is actually zero / the number of zero checks executed". The last column "sqrfree" shows "the number of worked square-free computations / the number of executed square-free computations / the number of calls of IntervalRealRootIsolation()".

problem	SyN	SyN2	QEP	RED	MATH
adam1	0.01	0.01	2.11	2.78	0.23
adam2-1	31	2293	529	>3600	2.75
adam2-2	282	>3600	1595	>3600	3.31
adam3	129	2092	>3600	>3600	21.58
pl01	1.19	0.61	1.01	0.08	0.06
lass	3.07	3.93	1.03	1.84	0.16
simple	2.05	4.07	1.04	5.79	0.15
makepdf	0.31	0.26	0.98	-	0.02
makepdf2	1.63	1.65	0.98	-	0.05
collision	0.14	0.12	1.03	0.02	0.02
consistency	0.13	0.08	1.03	0.14	0.02
termination	0.69	0.84	0.99	0.13	0.04
candj	2.55	3.81	1.04	1.18	0.15
dandh	2.44	2.74	0.99	0.62	0.08
xaxis	14.51	21.60	1.14	4.86	0.13
ono	0.88	0.74	24.70	1.73	0.72

Table 1: Computing time (sec)

problem	SyN	QEP	MATH
adam1	13	58	81
adam2-1	3027	6835	4853
adam2-2	9799	11653	5053
adam3	7661	-	32606
pl01	123	475	-
lass	899	1328	-
simple	741	863	-
makepdf	57	56	-
makepdf2	247	265	-
collision	47	519	-
consistency	33	277	-
termination	319	310	-
candj	567	685	-
dandh	1091	1215	-
xaxis	2791	3029	-
ono	69	239	-

Table 2: Number of cells

We note that Mathematica's implementation of CAD offers essentially "generic CAD", that is, it may miscategorize some points, which are measure-zero subsets of the free variable space, in the output formula [8], by which it can drastically decrease

problem	precision	zero check	sqrfree
adam1	0	0/1	0/0/7
adam2-1	7	8/155	68/70/1640
adam2-2	1696	117/913	346/355/8340
adam3	49	12/1054	160/170/5678
pl01	0	6/10	0/0/92
lass	0	10/50	2/4/678
simple	0	21/33	22/26/394
makepdf	0	2/2	0/0/18
makepdf2	0	9/28	0/1/115
collision	0	0/0	0/0/12
consistency	0	0/0	0/0/10
termination	0	3/10	0/0/134
candj	0	2/48	0/1/398
dandh	0	12/104	0/0/635
xaxis	1	36/348	15/15/1723
ono	0	0/0	0/0/76

Table 3: Numbers of occurrences

the time and space requirements of the computation. In our experiments, all the implementations except for Mathematica performed QE computation without omitting measure-zero subsets.

By tables, the authors think that the following points are confirmed from the computational results:

- Table 1 shows that SyN is more efficient than the others for relatively large problems. We can say that our proposed scheme and techniques in this paper works well, in particular, on large size problems.
- The significant effect of dynamic evaluation can be seen by comparing SyN and SyN2 in Table 1.
- Table 2 shows that SyN produces a smaller number of cells than the others by virtue of the techniques presented in §3.2.
- From Table 3 it can be shown that raising precision indeed occurs for some problems and checking symbolic zero is also required and actually worked a lot. Moreover it turns out that we can avoid quite a lot of square-free computations as an effect of numeric computation while there still remain necessary square-free computation.

The problems we deal with here are shown in the sequel. Four example problems in the first block of the tables are quoted from [32] concerning with stability study and control design. The second block consists of typical control design problems which are very obstinate for semidefinite programming [27]. The exemplary examples for solution formula construction [6] are in the third block. The last block adds some more well-known examples from previous QE related papers.

EXAMPLE 1. *(adam1 [32]) Stability of Dormand-Prince fifth-order embedded seven-stage method (Example 4.4 from Hong, et al. (1997)).*

$$\forall x \forall y((x < 0 \wedge x^2 + y^2 < \frac{99438}{100000}) \Rightarrow R(x+iy)R(x-iy) < 1),$$

where

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{z^6}{600}.$$

EXAMPLE 2. *(adam2 [32]) Stability of a six-point upwind-based second-order accurate scheme for approximating a two-dimensional advection equation (Example 5.4 from Hong, et al. (1997))*

$$2-1 \qquad \forall\alpha\forall\beta\forall C_2((\alpha \geq 0 \wedge \beta \geq 0 \wedge 4(\alpha^2+\beta^2) < 1) \Rightarrow (B \leq 0 \vee D \leq 0)),$$

$$2-2 \qquad \forall\alpha\forall\beta\forall C_2((0 \leq \alpha \leq 1 \wedge 0 \leq \beta \leq 1) \Rightarrow A \leq 0 \wedge C \leq 0 \wedge (B \leq 0 \vee D \leq 0)),$$

where

$$
\begin{aligned}
A &= C_2^4(\alpha-\beta+1)(\alpha-\beta-1)(\alpha-\beta)^2\\
B &= 2C_2^4\beta(3\alpha^2\beta-2\alpha^2-2\alpha\beta^2+\alpha+\beta^3-\beta)+4C_2^3\alpha\beta(\alpha^2-\alpha+\beta^2-\beta)+2C_2^2\alpha(\alpha^3-2\alpha^2\beta+3\alpha\beta^2-\alpha-2\beta^2+\beta)\\
C &= C_2^4\beta^2(\beta^2-1)+4C_2^3\alpha\beta^2(\beta-1)+\alpha^2(\alpha^2-1)+2C_2^2\alpha\beta(3\alpha\beta-2\alpha-2\beta+1)+4C_2\alpha^2\beta(\alpha-1)\\
D &= C_2^2R+2C_2S+T\\
R &= 8\alpha^2\beta^2-12\alpha^2\beta+5\alpha^2-8\alpha\beta^3+8\alpha\beta^2+2\alpha\beta-4\alpha+4\beta^4-4\beta^3-3\beta^2+4\beta\\
S &= 4\alpha^3\beta-2\alpha^3-4\alpha^2\beta^2-2\alpha^2\beta+\alpha^2+4\alpha\beta^3-2\alpha\beta^2+2\alpha\beta-2\beta^3+\beta^2\\
T &= 4\alpha^4-8\alpha^3\beta-4\alpha^3+8\alpha^2\beta^2+8\alpha^2\beta-3\alpha^2-12\alpha\beta^2+2\alpha\beta+4\alpha+5\beta^2-4\beta.
\end{aligned}
$$

EXAMPLE 3. *(adam3 [32]) Robust multi-objective feedback design (Example 4.2 from Dorato, el at. (1997)). Find the set of $\frac{n}{d}$ satisfying.*

$$
\begin{array}{l}
\exists q_1\exists q_2\forall w \quad (q_1>1 \ \wedge \ q_2>0 \ \wedge \ \frac{n}{d}>0 \wedge\\
\quad (\frac{n}{d}-q_1^2)w^4+(\frac{n}{d}((q_1+1)^2-2q_2)-(q_1^2+q_2^2))w^2+(\frac{n}{d}-1)q_2^2\geq 0 \wedge\\
\quad (\frac{n}{d}-q_1^2)w^4+(\frac{n}{d}((q_1-1)^2-2q_2)-(q_1^2+q_2^2))w^2+(\frac{n}{d}-1)q_2^2\geq 0)
\end{array}
$$

EXAMPLE 4. *(pl01 [27])*

$$
\begin{array}{ll}
\forall t_1\forall t_2 & ((-1\leq t_1\leq 1\wedge -1\leq t_2\leq 1)\\
\quad\Rightarrow & f\leq t_1^2t_2^4+t_1^4t_2^2+1-3t_1^2t_2^2)
\end{array}
$$

EXAMPLE 5. *(lass [23])*

$$
\begin{array}{ll}
\forall t_1\forall t_2\forall\rho & ((0\leq\rho<1 \ \wedge \ -\rho\leq t_1\leq\rho \ \wedge -\rho\leq t_2\leq\rho)\\
\quad\Rightarrow & f\leq t_1^2t_2^4+t_1^4t_2^2+1-t_1^2t_2^2)
\end{array}
$$

EXAMPLE 6. *(simple [6])*

$$\exists z \quad (10x-10y-19z>0 \ \wedge(x^2+y^2+(z-3)^2<9 \ \vee \ 2x+10y+19z\geq 11))$$

EXAMPLE 7. *(makepdf [6])*

$$\exists y \quad (x^2+y^2=1 \ \wedge \ x+y<0)$$

EXAMPLE 8. *(makepdf2 [6])*

$$\exists z \quad (x^2+y^2+z^2=1 \ \wedge \ x+2y+z<1)$$

EXAMPLE 9. *(collision: Collision Problem [11])*

$$
\begin{array}{lll}
\exists t\exists x\exists y(& & (15/16)t\geq 6\wedge(15/16)t\leq 10\\
& \wedge & x-(15/16)t\geq -1\wedge x-(15/16)/t\leq 1\\
& \wedge & y-(15/16)t\geq -9\wedge y-(15/16)/t\leq -7\\
& \wedge & (x-t)^2+y^2\leq 1
\end{array}
$$

EXAMPLE 10. *(consistency: Consistency in Strict Inequalities [11])*

$$\exists x\exists y\exists z \quad (x^2+y^2+z^2<1\wedge x^2+(y+z-2)^2<1)$$

EXAMPLE 11. *(termination: Termination of Term Rewrite System [11])*

$$\exists r\forall x\forall y \quad (x>r\wedge y>r\Rightarrow x^2(1+2y)^2>y^2(1+2x^2))$$

EXAMPLE 12. *(candj: Collins and Johnson 1989b [11, 24])*

$$
\begin{array}{ll}
\exists r & (0<r<1 \ \wedge \ b>0 \ \wedge \ a\geq 1/2 \wedge\\
& 3a^2r+3b^2r-2ar-a^2-b^2<0\\
& 3a^2r+3b^2r-4ar+r-2a^2-2b^2+2a>0
\end{array}
$$

EXAMPLE 13. *(dandh: Davenport and Heintz [11])*

$$\exists c\forall b\forall a \quad ((a=d\wedge b=c)\vee(a=c\wedge b=1)\Rightarrow a^2=b)$$

EXAMPLE 14. *(xaxis: The x-axis ellipse problem [6])*

$$\begin{array}{l}\forall x \forall y \quad (0 < a \leq 1 \ \wedge \ 0 < b \leq 1 \ \wedge \\ 0 \leq c < 1 - a \ \wedge (c - a < x < c + a \ \wedge \\ (b^2(x-c)^2 + a^2y^2 - a^2b^2 = 0) \ \Rightarrow \ x^2 + y^2 \geq 1))\end{array}$$

EXAMPLE 15. *(ono: Ono's inequality [33])*

$$\forall a \forall b \forall c \quad ((t_1 \geq 0 \wedge t_2 \geq 0 \wedge 3t \geq 0) \Rightarrow (n \geq 0)),$$

where

$$\begin{array}{rcl} s & = & (a+b+c)/2 \\ k & = & s(s-a)(s-b)(s-c) \\ t_1 & = & a^2 + b^2 - c^2 \\ t_2 & = & b^2 + c^2 - a^2 \\ t_3 & = & c^2 + a^2 - b^2 \\ n & = & (16k)^3 - 27t_1^2t_2^2t_3^2. \end{array}$$

6. DISCUSSION AND CONCLUDING REMARKS

In this paper we reported our implementation for effective symbolic-numeric CAD and QE based on SNCAD on real computer which have been evaluated the practicality and quality by computational experiments on several examples including practical engineering problems.

Since CAD consists of some sub-procedures, the efficiencies of the subprocedures (with certified numerical computations) influence the total efficiency. Moreover, the choice of strategy (i.e., combination of sub-procedures, switching between numerical and symbolic computations) also affects the overall efficiency. From the experimental results, it can be confirmed that our SNCAD and QE based on SNCAD have significant effects. In particular, our implementation is effective to relatively large problems. This is because a lot of heavy symbolic computations can be avoided by using the following organically:

- switchover machinery of numerical and symbolic computations,
- combination of dynamic evaluation and interval computations, and
- reduction techniques of the number of necessary cells.

Of course, as to our SNCAD scheme, there is more room for improvement not only in lifting phase but also in solution formula construction by elaborately exploiting numerical information. Moreover, it seems to be very promising to combine SNCAD with the ideas of discriminant variety for solving large practical problems.

7. ACKNOWLEDGMENTS

This work has been partially supported by CREST, Japan Science and Technology Agency and Global COE Program "Education-and-Research Hub for Mathematics-for-Industry" of Kyushu University. We would like to thank Prof. H. Hong, Prof. C. W. Brown, Dr. M. Kanno and Dr. K. Kimura, for their valuable advises.

8. REFERENCES

[1] H. Anai and S. Hara. Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In *Proceedings of American Control Conference 2000*, pages 1312–1316, 2000.

[2] H. Anai and P. A. Parrilo. Convex quantifier elimination for semidefinite programming. In *Proceedings of the 6th International Workshop on Computer Algebra in Scientific Computing 2003*. pp 3-11, 2003.

[3] H. Anai, H. Yanami, K. Sakabe, and S. Hara. Fixed-structure robust controller synthesis based on symbolic-numeric computation: design algorithms with a CACSD toolbox (invited paper). In *Proceedings of CCA/ISIC/CACSD 2004 (Taipei, Taiwan)*, pages 1540–1545, 2004.

[4] H. Anai and K. Yokoyama. Numerical cylindrical algebraic decomposition with certificated reconstruction. In *Proceedings of 11th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (Fukuoka, Japan)*, page 31, 2004.

[5] H. Anai and K. Yokoyama. Cylindrical algebraic decomposition via numerical computation with validated symbolic reconstruction. In *Proceedings of the A3L 2005*, pages 25–30, 2005.

[6] C. W. Brown. Solution Formula Construction for Truth-Invariant CADs. PhD thesis, 1999.

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

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

[9] B. Caviness and J. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Texts and monographs in symbolic computation. Springer-Verlag, 1998.

[10] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition–preliminary report. *SIGSAM Bull.*, 8(3):80–90, 1974.

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

[12] G. E. Collins, J. R. Johnson, and W. Krandick. Interval arithmetic in cylindrical algebraic decomposition. *J. Symb. Comput.*, 34(2):145–157, 2002.

[13] A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. In B. H. Matzat, G.-M. Greuel, and G. Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, Berlin, 1998.

[14] A. Dolzmann, A. Seidl, and T. Sturm. Efficient projection orders for CAD. In Proceedings of ISSAC 2004, pages 111–118, 2004.

[15] J. D. Dora, C. Dicrescenzo, and D. Duval. About a new method for computing in algebraic number fields. In B. F. Caviness, editor, *European Conference on Computer Algebra (2)*, volume 204 of *Lecture Notes in Computer Science*, pages 289–290. Springer, 1985.

[16] D. Duval. Algebraic numbers: an example of dynamic evaluation. *Journal of Symbolic Computation*, 18:429–445, 1994.

[17] J.-C. Faugère, G. Moroz, F. Rouillier and M.S. El Din. Classification of the perspective-three-point problem, discriminant variety and real solving polynomial systems of inequalities. In Proceedings of ISSAC 2008, pp 79-86, 2008.

[18] L. González-Vega. Applying quantifier elimination to the Birkhoff interpolation problem. *Journal of Symbolic Computation*, Vol. 22, Issue 1, pages 83-103 1966.

[19] L. González-Vega. A combinatorial algorithm solving some quantifier elimination problems. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 365–375. Springer, Wien, New York, 1998.

[20] H. Hong. Efficient method for analyzing topology of plane real algebraic curves. In *the International Symposium on Symbolic and Algebraic Computation (ISSAC 93)*, pages 264–274, 1993.

[21] H. Hong. Quantifier elimination for formulas constrained by quadratic equations. In Manuel Bronstein, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 93)*, pages 264–274, Kiev, Ukraine, ACM, ACM Press, July 1993.

[22] R. Krawczyk. Newton-Algorithmen zur Bcstimmung yon Nullstellen mit Fehlerschranken. Computing, Vol. 4, pp. 187-201, 1969.

[23] J.B. Lasserre. Global optimization with polynomials and the problems of moments. *SIAM Journal on Optimization*, 11 796–817, 2001.

[24] R. Loos and V. Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, Special issue on computational quantifier elimination, 36(5):450–462, 1993.

[25] R. E. Moore. A Test for Existence of Solutions to Nonlinear Systems. SlAM J.Numer. Anal., Vol. 14, No. 4, pp. 611- 615, Sept. 1977.

[26] R. E. Moore, and S. T. Jones. Safe Starting Regions for lterative Methods. SIAM J. Numer. Anal., Vol. 14, No. 6, pp. 1051-1065, Dec. 1977.

[27] P. Parrilo, and S. Lall. Semidefinite Programming Relaxation and Algebraic Optimization in Control. *Mini-course on Polynomial Equations and Inequalities I and II*, MTNS, 2006.

[28] S. Ratschan. Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1):21–42, 2002.

[29] S. M. Rump. Algebraic computation, numerical computation and verified inclusions. In R. Janssen, editor, *Trends in Computer Algebra*, volume 296 of *Lecture Notes in Computer Science*, pages 177–197. Springer-Verlag, New York, NY, 1988.

[30] S. M. Rump. Computer-assisted proofs and self-validating methods. In B. Einarsson, editor, *Accuracy and Reliability in Scientific Computing*, volume 18 of *Software, Environments, Tools*, chapter 10, pages 195–240. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005.

[31] A. W. Strzeboński. A real polynomial decision algorithm using arbitrary-precision floating point arithmetic. *Reliable Computing*, 5(3):337–346, 1999.

[32] A. W. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *J. Symb. Comput.*, 41(9):1021–1038, 2006.

[33] T. Sturm. REDLOG Example Management & Information System. http://www.algebra.fim.uni-passau.de/ redlog/remis/.

[34] T. Sturm and V. Weispfenning. Rounding and blending of solids by a real elimination method. In Achim Sydow, editor, *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modeling, and Applied Mathematics (IMACS 97)*, volume 2, pages 727–732, Berlin. IMACS, Wissenschaft & Technik Verlag, August 1997.

[35] V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1–2):3–27, February–April 1988.

[36] V. Weispfenning. Simulation and optimization by quantifier elimination. *Journal of Symbolic Computation*, Vol. 24, pages 208, 1997.

[37] V. Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing*, 8(2):85–101, February 1997.

[38] H. Yanami and H. Anai. Development of SyNRAC – formula description and new functions. In *Proceedings of International Workshop on Computer Algebra Systems and their Applications (CASA) 2004: ICCS 2004, LNCS 3039*, pages 286–294. Springer, 2004.

[39] H. Yanami and H. Anai. The Maple package SyNRAC and its application to robust control design. *Future Generation Comp. Syst.*, 23(5):721–726, 2007.

List of MI Preprint Series, Kyushu University

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

MI

MI2008-1 Takahiro ITO, Shuichi INOKUCHI & Yoshihiro MIZOGUCHI
Abstract collision systems simulated by cellular automata

MI2008-2 Eiji ONODERA
The intial value problem for a third-order dispersive flow into compact almost Hermitian manifolds

MI2008-3 Hiroaki KIDO
On isosceles sets in the 4-dimensional Euclidean space

MI2008-4 Hirofumi NOTSU
Numerical computations of cavity flow problems by a pressure stabilized characteristic-curve finite element scheme

MI2008-5 Yoshiyasu OZEKI
Torsion points of abelian varieties with values in nfinite extensions over a p-adic field

MI2008-6 Yoshiyuki TOMIYAMA
Lifting Galois representations over arbitrary number fields

MI2008-7 Takehiro HIROTSU & Setsuo TANIGUCHI
The random walk model revisited

MI2008-8 Silvia GANDY, Masaaki KANNO, Hirokazu ANAI & Kazuhiro YOKOYAMA
Optimizing a particular real root of a polynomial by a special cylindrical algebraic decomposition

MI2008-9 Kazufumi KIMOTO, Sho MATSUMOTO & Masato WAKAYAMA
Alpha-determinant cyclic modules and Jacobi polynomials

MI2008-10 Sangyeol LEE & Hiroki MASUDA
Jarque-Bera Normality Test for the Driving Lévy Process of a Discretely Observed Univariate SDE

MI2008-11 Hiroyuki CHIHARA & Eiji ONODERA
A third order dispersive flow for closed curves into almost Hermitian manifolds

MI2008-12 Takehiko KINOSHITA, Kouji HASHIMOTO and Mitsuhiro T. NAKAO
On the L^2 a priori error estimates to the finite element solution of elliptic problems with singular adjoint operator

MI2008-13 Jacques FARAUT and Masato WAKAYAMA
Hermitian symmetric spaces of tube type and multivariate Meixner-Pollaczek polynomials

MI2008-14 Takashi NAKAMURA
Riemann zeta-values, Euler polynomials and the best constant of Sobolev inequality

MI2008-15 Takashi NAKAMURA
Some topics related to Hurwitz-Lerch zeta functions

MI2009-1 Yasuhide FUKUMOTO
Global time evolution of viscous vortex rings

MI2009-2 Hidetoshi MATSUI & Sadanori KONISHI
Regularized functional regression modeling for functional response and predictors

MI2009-3 Hidetoshi MATSUI & Sadanori KONISHI
Variable selection for functional regression model via the L_1 regularization

MI2009-4 Shuichi KAWANO & Sadanori KONISHI
Nonlinear logistic discrimination via regularized Gaussian basis expansions

MI2009-5 Toshiro HIRANOUCHI & Yuichiro TAGUCHII
Flat modules and Groebner bases over truncated discrete valuation rings

MI2009-6 Kenji KAJIWARA & Yasuhiro OHTA
Bilinearization and Casorati determinant solutions to non-autonomous 1+1 dimensional discrete soliton equations

MI2009-7 Yoshiyuki KAGEI
Asymptotic behavior of solutions of the compressible Navier-Stokes equation around the plane Couette flow

MI2009-8 Shohei TATEISHI, Hidetoshi MATSUI & Sadanori KONISHI
Nonlinear regression modeling via the lasso-type regularization

MI2009-9 Takeshi TAKAISHI & Masato KIMURA
Phase field model for mode III crack growth in two dimensional elasticity

MI2009-10 Shingo SAITO
Generalisation of Mack's formula for claims reserving with arbitrary exponents for the variance assumption

MI2009-11 Kenji KAJIWARA, Masanobu KANEKO, Atsushi NOBE & Teruhisa TSUDA
Ultradiscretization of a solvable two-dimensional chaotic map associated with the Hesse cubic curve

MI2009-12 Tetsu MASUDA
Hypergeometric τ-functions of the q-Painlevé system of type $E_8^{(1)}$

MI2009-13 Hidenao IWANE, Hitoshi YANAMI, Hirokazu ANAI & Kazuhiro YOKOYAMA
A Practical Implementation of a Symbolic-Numeric Cylindrical Algebraic Decomposition for Quantifier Elimination