Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

A new Gröbner basis conversion method based on stabilization techniques

Kiyoshi Shirayanagi[a,*], Hiroshi Sekigawa[b]

[a] School of Science, Tokai University, 1117 Kitakaname, Hiratsuka-shi, Kanagawa, 259-1292, Japan

[b] NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation, 3-1 Morinosato-Wakamiya, Atsugi-shi, Kanagawa, 243-0198, Japan

ARTICLE INFO

Keywords:
Basis conversion
Gröbner basis
Algorithm stabilization
Floating-point Gröbner basis
Symbolic and numeric computation

ABSTRACT

We propose a new method for converting a Gröbner basis w.r.t. one term order into a Gröbner basis w.r.t. another term order by using the algorithm stabilization techniques proposed by Shirayanagi and Sweedler. First, we guess the support of the desired Gröbner basis from a floating-point Gröbner basis by exploiting the supportwise convergence property of the stabilized Buchberger's algorithm. Next, assuming this support to be correct, we use linear algebra, namely, the method of indeterminate coefficients to determine the exact values for the coefficients. Related work includes the FGLM algorithm and its modular version. Our method is new in the sense that it can be thought of as a floating-point approach to the linear algebra method. The results of Maple computing experiments indicate that our method can be very effective in the case of non-rational coefficients, especially the ones including transcendental constants.

1. Introduction

Symbolic and numeric computation is one of the most active research areas of computational algebra. Starting with approximate algebra [17,13], symbolic-numeric algebra for polynomials and structured matrices [15,3,23], this research resulted in a number of articles and collections of articles [23,22,26,16].

Sweedler and one of the authors of the present paper proposed stabilization techniques for algebraic algorithms [19], which have since been applied to various computations by other researchers. While approximate algebra assumes that there is noise or errors in input and aims to find a meaningful output from the scope of errors, the stabilization techniques assume that the input is exact and attempt to arrive at the exact output through approximate computations.

Throughout this paper, a *precision* denotes the number of decimal digits in the mantissa of a floating-point representation. When using an algorithm stabilized by the stabilization techniques, if we increase the precision of the floating-point coefficients of the input, the output will converge to the true output. This is called coefficientwise convergence. In fact, we can establish a supportwise convergence that is stronger than coefficientwise convergence if we slightly modify the stabilized algorithm. In this paper, using this supportwise convergence, we devise a new method of converting a Gröbner basis w.r.t. one term order into a Gröbner basis w.r.t. another term order. First, we guess the support of the desired Gröbner basis from a floating-point Gröbner basis, by using the supportwise convergence of the stabilized Buchberger's algorithm. Next, assuming this support to be correct, we use linear algebra, i.e., the method of indeterminate coefficients, to determine the exact values of the coefficients. This method can be applied to arbitrary dimensional ideals.

Related studies using linear algebra include the FGLM algorithm [6] and Noro–Yokoyama's algorithm [14]. While the latter is a modular approach, our method can be thought of as a floating-point approach. In this sense, the method is new.

* Corresponding author.
E-mail addresses: shirayan@ss.u-tokai.ac.jp (K. Shirayanagi), sekigawa@theory.brl.ntt.co.jp (H. Sekigawa).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.09.007

We give some results from Maple computing experiments, which show that the method can be very effective in the case of non-rational coefficients, especially the ones including transcendental constants.

This paper is organized as follows: Section 2 reviews the theory and gives examples of the stabilization techniques and remarks on applications. Section 3 describes the proposed method of basis conversion based on the stabilization techniques. Section 4 gives five experimental results and discusses the effectiveness of the method. The conclusion mentions future work.

2. Review of the stabilization techniques

2.1. Theory

Let us begin with a brief review of the theory of the stabilization techniques, whereby we restrict our discussion to the following class of algorithms. For the details in a more general setting, see [19].

- Data is from the polynomial ring $R[x_1, \dots, x_m]$, where R is a subfield of the reals.
- Operations between data are addition, subtraction, multiplication, and division in $R[x_1, \dots, x_m]$.
- If a predicate on data has a discontinuous point, this point should be 0.

Here, a discontinuous point of a predicate is said to be 0 if the algorithm branches depending on whether the evaluation value of the predicate is 0 or not, such as in "If $C = 0$ then ... else ...". Instead of "If $C = 0$", "If $C \geq 0$" or "If $C > 0$" can also be permitted. We refer to an algorithm belonging to the above class as an algebraic algorithm with discontinuity at 0. Most algorithms in computational algebra are algebraic algorithms with discontinuity at 0 or can be transformed into the ones without changing semantics.

Now, there are three key points for stabilization:

- Keep the structure of the algorithm unchanged.
- Change each coefficient into an interval coefficient in the data set.
- Perform *zero rewriting* prior to predicate evaluation.

Here, an interval consists of an approximate value and its error bound, which is often said to be a "circular disc interval". A different type of interval, such as a "rectangular interval", also works well. *Zero rewriting* means rewriting every interval containing zero into the zero interval $[0, 0]$.

More specifically, the stabilized algorithm has the following features.

Interval Domain The interval domain is the set of interval coefficient polynomials, where an interval coefficient is $[A, \alpha]$ with $A \in R$ and non-negative real α. $[A, \alpha]$ represents the set $\{x \in R : |x - A| \leq \alpha\}$.

Interval Arithmetic For binary operations $* \in \{+, -, \times, /\}$, $[A, \alpha] * [B, \beta] = [A * B, \gamma_*]$. Here, γ_* satisfies that $|x - A| \leq \alpha, |y - B| \leq \beta \Rightarrow |x * y - A * B| \leq \gamma_*$.

Zero Rewriting Prior to the evaluation of a predicate with discontinuity at 0: for each interval coefficient $[C, \gamma]$,

if $|C| \leq \gamma$, then rewrite $[C, \gamma]$ into $[0, 0]$

if $|C| > \gamma$, then keep $[C, \gamma]$ unchanged.

For more details on intervals and interval arithmetic, see [1].

If we write an input $f \in R[x_1, \dots, x_m]$ as $f = \sum_{\alpha} a_{\alpha} x^{\alpha}$ using multi-index notation, an approximation sequence $\{\mathrm{Int}(f)_j\}_j$ for f can be defined as

$$\mathrm{Int}(f)_j = \sum_{\alpha} [(a_{\alpha})_j, (\epsilon_{\alpha})_j]\, x^{\alpha},$$

where for all α and for any j, we have $|a_{\alpha} - (a_{\alpha})_j| \leq (\epsilon_{\alpha})_j$, and $(\epsilon_{\alpha})_j \to 0$ as $j \to \infty$. In this case, we simply write $\mathrm{Int}(f)_j \to f$. Most typically, as in the algorithm and examples which will appear in this paper, $(a_{\alpha})_j$ is a floating-point approximation of a_{α} with precision j.

Now, let $Stab(\mathcal{A})$ denote the stabilization of algorithm $\mathcal{A}$. The fundamental theorem is as follows.

Theorem 1 (*Coefficientwise Convergence*). *Let algorithm $\mathcal{A}$ be an algebraic algorithm with discontinuity at 0, and let it terminate normally for an input $f \in R[x_1, \dots, x_m]$. For any approximation sequence $\{\mathrm{Int}(f)_j\}_j$ for f, there exists n such that if $j \geq n$, $Stab(\mathcal{A})$ terminates normally for $\mathrm{Int}(f)_j$, and*

$$Stab(\mathcal{A})(\mathrm{Int}(f)_j) \to \mathcal{A}(f).$$

In many cases, *supportwise convergence* is more important than coefficientwise convergence. The *support* of polynomial $f = \sum_{\alpha} a_{\alpha} x^{\alpha}$ is the set $\{x^{\alpha} \mid a_{\alpha} \neq 0\}$, denoted by $\mathrm{Supp}(f)$.

$f_j \to f$ is supportwise convergent if it is coefficientwise convergent and there exists a finite integer n such that if $j \geq n$, then $\mathrm{Supp}(f_j) = \mathrm{Supp}(f)$.

We can establish supportwise convergence in the following manner:

After running $Stab(\mathcal{A})$, perform zero rewriting for each interval coefficient of the result.

We hereby denote this algorithm by $Stab(\mathcal{A})_R$. The following theorem holds:

Theorem 2 (*Supportwise Convergence*). *Let algorithm $\mathcal{A}$ be an algebraic algorithm with discontinuity at* 0*, and let it terminate normally for an input $f \in R[x_1, \ldots, x_m]$. For any approximation sequence $\{\mathrm{Int}(f)_j\}_j$ for f, there exists n such that if $j \geq n$, $Stab(\mathcal{A})_R$ terminates normally for $\mathrm{Int}(f)_j$ and $Stab(\mathcal{A})_R(\mathrm{Int}(f)_j)$ supportwise converges to $\mathcal{A}(f)$.*

Note that the input does not have to be only a single polynomial. It can also be a finite set of polynomials.

The proofs of the above theorems in a more general setting and specifically for Buchberger's algorithm that computes Gröbner bases are given in [19,18], respectively.

The aim of this paper is to compute an exact coefficient Gröbner basis using this notion of supportwise convergence.[1]

2.2. Examples

We illustrate some examples to which the stabilization techniques have been applied by computer. Interval description and interval arithmetic are all done by using floating-point approximations. The references cited in the table describe the details of the corresponding example.

Algorithm	Output	Experiments (who and when)
Buchberger	Gröbner basis	(Shirayanagi 93) [18], (Ozaki 94), (Hiyoshi 97), (Traverso–Zanoni 02) [25], (Krandick 05) [10]
Sturm	Number of real roots	(Sekigawa 95)
Graham etc.	2d and 3d convex hull	(Sekigawa 95&98)
Elementary divisor method	Smith normal form	(Niitsuma 96)
Greville	Generalized inverse	(Minakuchi 96–98) [12], (Murakami 01)
Gauss	Triangular matrix	(Murakami 01)
Wu	Characteristic set	(Notake 00), (Shiraishi 01)
Lazard–Rioboo–Trager	Rational integration	(Khungurn 05) [7]

Here, Traverso and Zanoni do not directly use the stabilization techniques, but they do incorporate the idea of zero rewriting in the computation of numerical Gröbner bases. Note that all the algorithms in the table are algebraic algorithms with discontinuity at 0.

The results of these experiments indicate that the stabilization techniques are useful when exact computation of the original algorithm is slow *due to the growth of coefficients*.

2.3. Two categories of applications

We believe that there are two application categories for the stabilization techniques. The first category is one in which we estimate the precision for stabilization in advance, and with that precision run the stabilized algorithm to get the correct result. The second category is one in which we repeatedly run the stabilized algorithm while increasing the precision until the correct result is obtained. The application in the present paper belongs to the second category, but let us briefly mention the recent progress on the first category. This is about a precision problem, which is one of the important unsolved problems of the theory.

[1] We previously proposed another method to compute exact coefficient Gröbner basis using supportwise convergence, called the *log method* [20,21], and showed that it was effective in some cases.

Convergence is guaranteed by the theorems on coefficientwise and supportwise convergence for a sufficiently large precision. However, it is very difficult to estimate which precision will give a stabilized result. We refer to this problem as the precision problem. In fact, the difficulty of this problem comes from the fact that the desired precision depends on not only the algorithms but also the inputs.

Recently, however, Khungurn and the authors of this paper solved the precision problem for GCD computing algorithms [8,9]. Theoretically, there is a precision at and beyond which the execution path of the stabilized algorithm run with floating-point computations coincides with that of the original algorithm run with exact computation. We refer to the minimum value of such a precision as the *minimum converging precision* (*MCP*). We proved that MCP is generally incomputable for some class of GCD computing algorithms. Furthermore, in some specific domain such as $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}[\xi]$ (ξ is an algebraic integer), we gave an upper bound of the MCP for the Euclidean algorithm. Although bounding the MCP is still an open problem, we derived an upper bound for the *minimum degree precision* and the *minimum correct support precision* in the case of GCD computing algorithm based on QR decomposition of the Sylvester matrix.

3. Basis conversion based on the stabilization techniques

Let us consider the following problem:

> **Problem.** Given a Gröbner basis w.r.t. one term order of a polynomial ideal in $R[x_1, \ldots, x_m]$, compute a Gröbner basis w.r.t. another term order of the ideal.

Typically, while the first term order is graded reverse lexicographic order or graded lexicographic order with which the Gröbner basis computation is rather fast, the second term order is lexicographic order with which the computation is often slow.

The problem can be solved by using linear algebra such as the FGLM algorithm [6] and its modular version [14].[2] While the FGLM algorithm can only be executed for zero-dimensional ideals, the modular approach can be applied to arbitrary dimensional ideals. The FGLM algorithm prepares a set of monomials and the associated coefficients of indeterminates, and to determine the coefficients in the underlying field, it solves the linear system derived from the normal forms modulo the given Gröbner basis being zero. Particularly in the case of integer coefficients, the modular method chooses a suitable prime p and executes Buchberger's algorithm in $\mathbb{Z}/p\mathbb{Z}$, and when preparing a set of monomials, it adopts the support of the obtained Gröbner basis.

The approach we propose in this paper is to use a floating-point approximation with precision μ in place of the reduction mod p. This method can be applied to arbitrary dimensional ideals in any subfield R of $\mathbb{R}$. The outline is as follows:

> **Method.** Use $Stab(Buchberger)_R$ to get the support of the Gröbner basis with floating-point coefficients with an initial precision μ. Assuming this support to be correct, use the method of indeterminate coefficients. If there is no solution in R, increase precision and repeat the above process. Here, *Buchberger* denotes Buchberger's algorithm.

Let us describe an algorithmic form of this method on the next page. Let F be a finite subset of $R[x_1, \ldots, x_m]$ so that $I = \langle F \rangle$ is an arbitrary dimensional ideal.

In **BaseConv_Stab** (Basis conversion algorithm based on stabilization), $NF_{G_0,<_0}(\sum_{t\in \mathrm{Supp}(g)} a_t t)$ denotes the normal form of $\sum_{t\in \mathrm{Supp}(g)} a_t t$ modulo G_0 w.r.t. $<_0$. The indeterminate coefficients a_t's except for the leading monomial t are computed by exact linear algebra, so that an exact solution is obtained. Note that as the floating-point Gröbner basis computed by $Stab(Buchberger)_R$ is reduced, the resulting Gröbner basis will also be reduced. By the uniqueness of monic reduced Gröbner bases, the linear system L_g eventually will have a unique solution.

In the last "if" statement, the check to see if G is the true Gröbner basis of $\langle G_0 \rangle$ w.r.t. $<$ can be done by executing the following two steps:

(1) First, confirm that G is a Gröbner basis by checking that the S-polynomial of each pair of G reduces to 0 by G w.r.t. $<$.
(2) Second, confirm that $\langle G \rangle \supseteq \langle G_0 \rangle$ by checking that every element of G_0 reduces to 0 by G w.r.t. $<$.

In fact, we already have that $\langle G \rangle \subseteq \langle G_0 \rangle$ because every element of G reduces to 0 by G_0 w.r.t. $<_0$, from the structure of the **BaseConv_Stab** algorithm. Therefore, by (2), we have that $\langle G \rangle = \langle G_0 \rangle$. That is, the correctness of **BaseConv_Stab** is verified. Termination also immediately follows from Theorem 2 on the correct support from a finite precision.

4. Experiments

We give a brief report of the experiments on **BaseConv_Stab** that we conducted in Maple. We used Maple 10 loaded on a Dell Dimension DC051 PC (Intel(R) Pentium 4 CPU: 3.00 GHz, RAM: 2.99 GHz, 0.99 GB). Here we give five illustrative examples. For every example F of polynomial sets, we first computed the Gröbner basis of $\langle F \rangle$ w.r.t. tdeg in advance in Maple, and then computed the Gröbner basis of $\langle F \rangle$ w.r.t. plex by using our method. Here, tdeg and plex denote graded reverse lexicographic order and lexicographic order, respectively.

[2] For other approaches, we refer the reader to [11,24,5,2].

(1) $F = \{f_1, f_2, f_3\}$, where $f_1 = \frac{1}{7}x^2 - \frac{326548390854652}{272974017239}x + \frac{1263781236281}{712638126}y^2 + \frac{26872672361827}{7263188218281}z^2$, $f_2 = \frac{3}{8}xy + \frac{12367812638123}{763812368213132}yz - \frac{63812638126}{77263812831}y$, $f_3 = \frac{4}{9}x + \frac{327091270979304}{24122375460421}y + \frac{18467031595309203}{318405459032}z - \frac{356318063693141319}{6436561806418109}$.
(2) $F = \{(\sqrt{2}+\sqrt{5})x^3y + \sqrt{3}xy + \sqrt{7}, (\sqrt{3}-\sqrt{2})x^2y^2 - \sqrt{7}xy + \frac{\sqrt{11}}{11}\}$.
(3) $F = \{ex + \sqrt{2}y + \sqrt{3}z, exy + \sqrt{5}yz + \sqrt{3}zx, xyz - e\}$, where e is Napier's number (2.71828 ...).
(4) $F = \{\sqrt{2}ex^2 + xy^2 - z + 1/4, \sqrt{3}x + y^2z + 1/2, \sqrt{5}ex^2z - 1/2\,x - y^2\}$.
(5) $F = \{\sqrt{2}e/\pi x^3y + (\sqrt{3}+\pi)xy + \sqrt{7}/(e-\pi), (1-e\sqrt{3})/e \cdot \pi x^2y^2 - (\sqrt{7}-e)xy + e/\sqrt{11}\}$.

Examples 1 and 5 are from the paper [18] about floating-point Gröbner bases. Example 2 is a modification of another example in the same paper. Examples 3 and 4 are modifications of cyclic3 and an example from paper [4], respectively.

```
BaseConv_Stab (Basis conversion algorithm based on stabilization)
Input: Gröbner basis $G_0$ of $I$ w.r.t. term order $<_0$, another term order $<$
Output: Reduced Gröbner basis of $I$ w.r.t. $<$

$\mu := M$ (Initial value of precision of floating-point approximation)
again
loop
   $G_\mu \leftarrow$ Floating-point reduced Gröbner basis of $\langle G_0 \rangle$ w.r.t. $<$
         with precision $\mu$ computed by $Stab(Buchberger)_R$
   $G \leftarrow \emptyset$
   for $g \in G_\mu$ do
      for $t \in \mathrm{Supp}(g)$ do
         if $t$ is $LM(g)$ (the leading monomial of $g$ w.r.t $<$)
         then $a_t := 1$
         else $a_t :=$ Indeterminate corresponding to the floating-point
                  coefficient of $t \in \mathrm{Supp}(g)$
         endif
      endfor
      $L_g :=$ Linear system in $a_t$'s (except for $t = LM(g)$) made from
            $NF_{G_0,<_0}(\sum_{t \in \mathrm{Supp}(g)} a_t t) = 0$ by setting each coefficient
            w.r.t. $x_1, \ldots, x_m$ to zero.
      Solve $L_g$ in $R$
      if there exists a unique solution $\tilde{a}_t \in R$ for $a_t$ (except for $t = LM(g)$)
      then $G \leftarrow G \cup \{\sum_{t \in \mathrm{Supp}(g)} \tilde{a}_t t\}$
      else increase $\mu$ and goto again
      endif
   endfor
   if $G$ is the true Gröbner basis of $\langle G_0 \rangle$ w.r.t. $<$
   then return $G$ else increase $\mu$ goto again
   endif
endloop
```

We assume that we have a means of computing arbitrarily good approximations to the given transcendental constants. In this experiment, we used *evalf* in Maple.

The experimental results are shown in the table.

Example	GB(tdeg) by Maple	Basis conversion	MP	GB(plex) by Maple
1	0.03	0.28	4	0.09
2	1.03	0.88	3	5.25
3	1.26	0.84	3	116.61
4	0.25	50.14	10	>3600
5	>3600	–	–	>3600

In this table, "GB(tdeg) by Maple" shows the cpu time in seconds for the Basis function in Maple 10 to compute the Gröbner basis w.r.t. tdeg. "Basis conversion" shows the cpu time in seconds for the execution of **BaseConv_Stab** where the initial value of floating-point precision was set to 1 and thereafter the precision was increased by 1 if a failure occurred. Practically, however, various other ways of increasing the precision such as doubling at each iteration can be considered as well. "MP" means the minimum precision which produced success. For reference, "GB(plex) by Maple" shows the cpu time in seconds for the Maple 10 Basis to compute the Gröbner basis w.r.t. plex. ">3600" and "–" respectively denote that the computation did not terminate after 3600 s and that we could not get the result.

Discussion. The experimental results indicate that our method of basis conversion is more useful in the case of non-rational coefficients than in the case of rational coefficients. In particular, when coefficients contain not only algebraic numbers but also transcendental constants, it often displays great effectiveness. The cpu time taken for checking whether the obtained result is the true Gröbner basis in the algorithm was rather small.

We believe that methods such as the FGLM algorithm or the modular version would be good in the cases of rational or integer coefficients, whereas our method would be useful in the cases with non-rational coefficients, especially the ones containing transcendental constants, because floating-point computation can display its full power.

In example 5, however, the method could not be used because an exact Gröbner basis w.r.t. tdeg was not obtained. In such a case, we might consider a Gröbner basis w.r.t. another term order, or if possible, compute a Gröbner basis w.r.t. tdeg by using another computer algebra system. Moreover, for example 5, we easily obtained a *floating-point* Gröbner basis w.r.t. plex by using the stabilization techniques, so we might try the log method instead of basis conversion.

Finally, let us discuss other possible approaches to Gröbner basis computation in the case of coefficients containing transcendental constants. First, in place of each transcendental constant, we might introduce an indeterminate and then directly compute the Gröbner basis w.r.t. plex in the coefficient field with the indeterminates adjoined. This method may be faster than direct computation in the original coefficient field, but in general it cannot avoid the growth of coefficients as well. In fact, it was faster for example 3, but for examples 4 and 5, it could not display the results within a reasonable time. Moreover, if the number of transcendental constants is more than 1, we need to be careful about the algebraic independence among them. Another approach to basis conversion would be to change each transcendental constant to an integer chosen *at random* and then guess the support by computing the Gröbner basis w.r.t. plex in the new coefficient field.[3] In most cases, it would give the correct support more efficiently, but theoretically it is probabilistic. This approach deserves probabilistic analysis.

5. Conclusion

We proposed a new Gröbner basis conversion method based on the stabilization techniques. We experimentally confirmed that it is useful in the non-rational case. Future work will include experiments in other computer algebra systems, an estimation of the precision for which the correct support can be obtained, applications to real-world problems where non-rational coefficients appear and new applications of supportwise convergence to other algebraic problems.

Acknowledgments

The authors are grateful to Victor Pan and the anonymous referees for their valuable comments on improving the paper.

References

[1] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Computer Science and Applied Mathematics, Academic Press, 1983.
[2] A. Basiri, J.-C. Faugère, Changing the ordering of Gröbner bases with LLL: Case of two variables, in: Proc. the 2003 International Symposium on Symbolic and Algebraic Computation, ISSAC 2003, 2003, pp. 23–28.
[3] D. Bini, V.Y. Pan, Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms, Birkhäuser, Boston, 1994.
[4] B. Buchberger, Gröbner bases: An algorithmic method in polynomial ideal theory, in: N.K. Bose (Ed.), Multidimensional Systems Theory, D. Reidel Publishing Company, 1985, pp. 184–232. (Chapter 6).
[5] S. Collart, M. Kalkbrenner, D. Mall, Converting bases with the Gröbner walk, Journal of Symbolic Computation 24 (3–4) (1997) 465–469.
[6] J. Faugère, P. Gianni, D. Lazard, T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, Journal of Symbolic Computation 16 (4) (1993) 329–344.
[7] P. Khungurn, Stabilizing an algorithm for integrating rational functions, 2005, Preprint. http://web.mit.edu/pramook/www/18.337/report.pdf.
[8] P. Khungurn, Shirayanagi–Sweedler algebraic algorithm stabilization and polynomial gcd algorithms, Master's Thesis, MIT School of Engineering, 2007.
[9] P. Khungurn, H. Sekigawa, K. Shirayanagi, Minimum converging precision of the QR-factorization algorithm for real polynomial, in: Proc. ISSAC 2007, 2007, pp. 227–234.
[10] W. Krandick, Symbolic algebraic techniques for steady state power system analysis. slides, 2005. http://www.drexel.edu/coe/news/articles/rd05/393.pdf.
[11] S. Licciardi, T. Mora, Implicitization of hypersurfaces and curves by the primbasissatz and basis conversion, in: Proc. ISSAC'94, 1994, pp. 191–196.
[12] H. Minakuchi, H. Kai, M.-T. Noda, Algorithms of generalized inverse and their stabilization, in: Electronic Proc. ATCM'98, 1998. http://www.atcm.mathandtech.org/EP/1998/ATCMP046/paper.pdf.
[13] M.-T. Noda, T. Sasaki, Approximate gcd and its application to ill-conditioned algebraic equations, Journal of Computational and Applied Mathematics 38 (1991) 335–351.
[14] M. Noro, K. Yokoyama, A modular method to compute the rational univariate representation of zero-dimensional ideals, Journal of Symbolic Computation 28 (1) (1999) 243–263.
[15] V.Y. Pan, Complexity of computations with matrices and polynomials, SIAM Review 34 (2) (1992) 225–262.
[16] J. Verschelde, S. Watt (Eds.), Proceedings of the 2007 International Workshop on Symbolic-Numeric Computation, SNC 2007, July 2007, London, Ontario, Canada, ACM Press, New York, 2007.
[17] T. Sasaki, M.-T. Noda, Approximate square-free decomposition and root finding of ill-conditioned algebraic equations, Journal of Information Processing 12 (2) (1989) 159–168.
[18] K. Shirayanagi, Floating point Gröbner bases, Mathematics and Computers in Simulation 42 (4–6) (1996) 509–528.

[3] This approach was personally communicated by Dr. Masayuki Noro.

[19] K. Shirayanagi, M. Sweedler, A theory of stabilizing algebraic algorithms, Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 92 pp., 1995. http://www.ss.u-tokai.ac.jp/~shirayan/msitr95-28.pdf.
[20] K. Shirayanagi, M. Sweedler, Automatic algorithm stabilization, in: ISSAC'96 Poster Session Abstracts, 1996, pp. 75–78.
[21] K. Shirayanagi, M. Sweedler, Remarks on automatic algorithm stabilization, Journal of Symbolic Computation 26 (6) (1998) 761–766.
[22] Special issue on algebraic and numerical algorithms, Theoretical Computer Science 315 (2–3) (2004) 307–672.
[23] Special issue on symbolic numeric algebra for polynomials, Journal of Symbolic Computation 26 (6) (1998).
[24] C. Traverso, Hilbert function and the Buchberger algorithm, Journal of Symbolic Computation 22 (1996) 355–376.
[25] C. Traverso, A. Zanoni, Numerical stability and stabilization of Groebner basis computation, in: Proc. ISSAC 2002, pp. 262–269.
[26] D. Wang, L. Zhi, Symbolic-Numeric Computation, Birkhäuser, Basel/Boston, 2007.