Numerical computation of Gröbner bases

Aleksey Kondratyev[1], Hans J. Stetter[2], and Franz Winkler[1]

[1] RISC-Linz
J. Kepler University, Linz, Austria
Aleksey.Kondratyev@risc.uni-linz.ac.at, Franz.Winkler@jku.at
[2] Institute for Applied and Numerical Mathematics
Technical University, Vienna, Austria
stetter@tuwien.ac.at

Abstract. In this paper we deal with the problem of numerical computation of Gröbner bases of zero-dimensional polynomial systems. It is well known that the computation of a Gröbner basis cannot be generally executed in floating-point arithmetic by a standard approach. This, however, would be highly desirable for practical applications. We present an approach for computing Gröbner bases numerically. It is an elaboration of the idea of a stabilized Gröbner basis computation initially proposed by Hans Stetter. Our implementation of the algorithm based on the presented results is available online.

1 Introduction

It is known that the Buchberger algorithm for computing a Gröbner basis of a polynomial system is generally unstable under small changes in the coefficients of the system. Hence it can not be generally executed in floating-point arithmetic and/or with inexact input data. This problem has received considerable attention from different points of view [2], [4], [5], [10], [13], [14], [15], [16]. From our point of view it has not been sufficiently realized in the aforementioned related research that the approximate computation of a Gröbner basis is a more fundamental challenge than an appropriate modification of the Buchberger algorithm: The instability of Gröbner bases is not just a peculiarity of the Buchberger algorithm but rather a problem with the standard definition of a Gröbner basis. Our approach and the corresponding algorithm for zero-dimensional polynomial systems use a modified notion of a Gröbner basis. This is an *extended Gröbner basis* [11] which provides a stable representation of the initially stable polynomial system in the case when its genuine Gröbner basis is unstable. Our algorithm is a variation of the Buchberger algorithm which recognizes the "danger of instability" at run-time and changes the course of computation to maintain the stability resulting in an extended Gröbner basis. If no "danger" is encountered then the algorithm performs as a slightly overcomplicated version of the Buchberger algorithm resulting in a Gröbner basis in the conventional sense.

Our main interest is solving a zero-dimensional polynomial system with numerically specified coefficients with a limited meaningful accuracy. The method of Gröbner bases is a standard tool in computer algebra for solving polynomial systems. As we have already mentioned, this tool is not readily suited for the numerical setting. Indeed, the computation of a Gröbner basis by the Buchberger algorithm is essentially an elimination process like Gaussian elimination in a linear system; but in place of the variables in a linear system we have terms, i.e. power products in the variables, and this Buchberger type elimination generally requires linear combinations of shifts with terms, the so-called S-polynomials.

For the machinery of Gröbner bases, the term order is the key ingredient as it introduces a well-defined "direction" of elimination of the terms in the polynomial system in place of the linear order of the variables in Gaussian elimination. But it is well understood in numerical analysis that a numerically stable Gaussian elimination must take into account the magnitude of the coefficients employed in the elimination. This mechanism is called pivoting. In the computation of a Gröbner

This paper contains results from the Ph.D. Thesis of A. Kondratyev; his work was partially supported by the Austrian Scientific Research Foundation FWF under the grants P 13266−MAT and SFB F013/F1304.

basis, the place of pivots is taken by the coefficients of the leading terms which are used for S-polynomials and reductions. Once a term order is specified, the leading term and hence the leading coefficient of any polynomial are uniquely defined. So the elimination algorithm leaves no freedom for an alternative selection of pivots which could take into account the magnitude of the coefficients. This limitation makes the customary Gröbner basis algorithm numerically unstable. Thus, it cannot generally be executed in floating-point arithmetic, which would be highly desirable in many practical situations.

But there is another, mathematically more striking drawback, concerning not the course but the result of the computation: The totally reduced Gröbner basis of a polynomial system may depend discontinuously on the system: A slight change in the coefficients of the system may change the *structure* of the Gröbner basis and the associated normal set. As the coefficients of the system approach a discontinuous situation, some coefficients of the Gröbner basis polynomials diverge to infinity. Hence the Gröbner basis may be a very ill-conditioned representation of the ideal of the system even if the system is perfectly well-conditioned itself. In this situation, the Gröbner basis is naturally unsuitable for whatever subsequent numerical computation.

The potential occurrence of such artificial discontinuities introduced in Gröbner bases by the strict adherence to a term order makes it hard to design and develop a general-purpose black-box floating-point software for solving polynomial systems with coefficients of limited accuracy via Gröbner bases. The analysis of the origin of these discontinuities and a concept for a *stabilized Gröbner basis algorithm* which permits a safe use of floating-point arithmetic by locally overriding the demands of a term order in favor of numerical stability have been developed by H.J. Stetter [11]. The result of such a stabilized Gröbner basis algorithm applied to a polynomial system is generally not a Gröbner basis in the usual computer algebra sense but a so-called *extended Gröbner basis*, a slight perturbation of the genuine Gröbner basis of a system in close vicinity of the input system. This is an appropriate representation of the ideal of the system which avoids the artificial representation discontinuities possibly introduced by a genuine Gröbner basis.

The proposed idea of a stabilized Gröbner basis computation works well for sufficiently simple cases and generates the expected results but it has not been clear so far how it can be turned into a general stabilized Gröbner basis algorithm to be executed in floating-point arithmetic which, given a system of polynomials $\mathcal{F}$,

– recognizes the closeness to a discontinuity of the genuine Gröbner basis $\widehat{\mathcal{G}}$ and the subsequent ill-conditioning which is to be expected; in this case, delivers an extended Gröbner basis $\mathcal{G}$ which does not share the ill-conditioning of $\widehat{\mathcal{G}}$;
– delivers the genuine Gröbner basis $\widehat{\mathcal{G}}$ of $\mathcal{F}$ if no discontinuity of $\widehat{\mathcal{G}}$ occurs near $\mathcal{F}$, in this case no deviation from the standard approach is necessary.

In this paper we present such an algorithm.[1] The algorithm is based on an encoding of the order of magnitude of coefficients by a new variable e and an extension of the term order to an order of the terms with e as explained in Section 2. The application of this technique within a Gröbner basis computation is shown in Section 2.2. In Section 2.3 we explain how the resulting Gröbner basis in the extended variable range is reinterpreted in the original variables which leads to either a genuine or an extended Gröbner basis. In Section 2.4, we explain some techniques in the linear algebra of the elimination which enhance the numerical stability in floating-point execution of the algorithm. The interpretation of an "approximate Gröbner basis" is discussed in Section 2.5. Finally, we exhibit an example of the use of the algorithm in Section 3 and indicate some potential future developments in Section 4.

2 Computation of an extended Gröbner basis

Let $\mathbb{C}$ be the field of complex numbers, $\mathbb{C}^*{=}\mathbb{C}\backslash\{0\}$, and X a finite set of indeterminates (variables). Denote by $\mathcal{P}{=}\mathbb{C}[X]$ a polynomial algebra equipped with an admissible term order $<_{\mathbb{T}}$ on the terms $\mathbb{T}$ in X. Consider a polynomial system $\mathcal{F}\subseteq\mathcal{P}$. Assume $\mathcal{F}$ is zero-dimensional, i.e. the variety $Z(\mathcal{F})$

[1] The implementation of the algorithm in C++ is available online via anonymous ftp from `ftp://ftp.risc.uni-linz.ac.at/pub/people/akondra/ Numerical_Groebner_bases.tar.gz`.

of $\mathcal{F}$ consists of finitely many isolated points. It is well-known that for some specific configurations of the solutions, a small perturbation of the coefficients may cause a structural change in the geometry of the solutions e.g. change the number of solutions or blow up points into curves. We assume that the geometry of the solutions remains structurally unchanged in a neighborhood of $\mathcal{F}$. Our goal is to find an extended Gröbner basis of $\mathcal{F}$ w.r.t. $<_{\mathbb{T}}$ under this restriction.

Definition 1 (cf. Definition 4.1 **in [11]).** *Let $\mathcal{F}$ be zero-dimensional.* **An extended Gröbner basis** $\mathcal{G}$ $(\subseteq\mathcal{P})$ *of $\mathcal{F}$ w.r.t. a specified term order $<_{\mathbb{T}}$ is*

$$\mathcal{G} = \Big\{ g + \sum_{t\in N} \beta_t(g)\cdot t \;\Big|\; g\in\widetilde{\mathcal{G}} \Big\} \textit{ with } \beta_t(g)\in\mathbb{C} \textit{ for } t\in N,\ g\in\widetilde{\mathcal{G}},$$

where $\widetilde{\mathcal{G}}$ is the genuine totally reduced Gröbner basis of a slightly perturbed $\mathcal{F}$, N is the normal set associated with $\widetilde{\mathcal{G}}$ and all coefficients $\beta_t(g)$, $t\in N$, $g\in\widetilde{\mathcal{G}}$, are sufficiently small.

Note that, for a specified term order, the genuine Gröbner basis of $\mathcal{F}$ is a special case of an extended Gröbner basis.

The main idea of our approach is to avoid small leading coefficients by locally overriding of the demands of a term order in the computation of a Gröbner basis. This is done by *encoding* the order of magnitude of the coefficient of a term in a polynomial by a power of a new variable e. This new variable is added to X and the term order $<_{\mathbb{T}}$ is extended to an order on terms in $X\cup\{e\}$ such that a smaller magnitude of a coefficient results in a smaller term. This way we achieve a stabilization of the course as well as of the result of computation.

Assume that the system $\mathcal{F}\subseteq\mathcal{P}$ whose (extended) Gröbner basis we wish to find is given with limited numerical precision $0<\delta\ll 1$. Consider a small real number $1\gg\varepsilon>\delta$ and and an *e-bound*: the smallest integer $\mathcal{B}>1$ such that $\varepsilon^{\mathcal{B}}\leq\delta$ i.e. $\varepsilon^{\mathcal{B}}$ is indistinguishable from 0 in the given precision; $\mathcal{B}$ is typically 2 or 3. We extend the set of variables X by a new variable e such that $e^{\mathcal{B}}=0$. The sets

$$\mathbb{T}(e)^* = \big\{ t\cdot e^i \;\big|\; t\in\mathbb{T}, 0\leq i<\mathcal{B} \big\},\ T(e) = T(e)^*\cup\{0\}$$

and the algebra

$$\mathcal{P}(e) := \mathbb{C}[X,e]/_{\langle e^{\mathcal{B}}\rangle}$$

generated by $X\cup\{e\}$ are called the *set of non-zero extended terms*, the *set of extended terms* and the *algebra of extended polynomials* respectively. The original variables X are called the *main variables*. It is quite clear that each non-zero extended term s can be uniquely represented as $s=t\cdot e^d$ with $t\in\mathbb{T}$, $0\leq d<\mathcal{B}$: t and d in the unique representation of s are referred to as the *main part*, denoted $\mathrm{T}(s)=t$, and the *e-degree*, denoted $\deg_e(s)=d$, of s, respectively.

It is also easy to see that $\mathbb{T}(e)^*$ is a $\mathbb{C}$-linear basis of $\mathcal{P}(e)$. This is due to the fact that $\mathcal{P}(e)$ is a polynomial algebra with a sole monomial relation $e^{\mathcal{B}}=0$. Hence every $f\in\mathcal{P}(e)$ can be written as the unique finite linear combination of terms $\mathrm{supp}\,(f)\subseteq\mathbb{T}(e)^*$. For $F\subseteq\mathcal{P}(e)$, $\mathrm{supp}\,(F) := \bigcup_{f\in F}\mathrm{supp}\,(f)$.

The major steps taken in our approach can be summarized as follows.

1. *Preprocessing:* The input system $\mathcal{F}$ is *translated* to the system $F\subseteq\mathcal{P}(e)$ in the algebra of extended polynomials by encoding the orders of magnitude of the coefficients of F by the powers of e.
2. *Computation:* An admissible term order $<_{\mathbb{T}}$ on T is extended to an *extended term order* $<$ on terms $T(e)$. This extended term order coincides with $<_{\mathbb{T}}$ on $\mathbb{T}$ and has to satisfy certain requirements similar to an admissible term order; cf. Section 2.2. A numerical version of a Gröbner basis algorithm in $\mathcal{P}(e)$ is used to compute a totally reduced Gröbner basis $G\subseteq\mathcal{P}(e)$ w.r.t. $<$. This algorithm is based on our theory of Gröbner bases in $\mathcal{P}(e)$ which is a variation of the standard theory developed by B. Buchberger.
3. *Postprocessing:* The attained basis G is reinterpreted as a system $\mathcal{G}\subseteq\mathcal{P}$ by setting $e{:=}1$. If G satisfies certain conditions then $\mathcal{G}$ is an extended Gröbner basis w.r.t. $<_{\mathbb{T}}$; cf. Section 2.3.

In the next sections we consider the aforementioned steps in more detail. A complete description is available in Chapter 3 of [6].

2.1 Preprocessing

A degree of e is used in our approach to encode the size of the coefficient in a monomial $\alpha{\cdot}t{\in}\mathcal{P}$ with $\alpha{\in}\mathbb{C}^*$ and $t{\in}\mathbb{T}$ by the correspondence Ω defined as

$$\Omega(\alpha{\cdot}t) = \alpha{\cdot}\big(t{\cdot}e^{\mathcal{O}(\alpha)}\big),$$

where $\mathcal{O}\colon \mathbb{C}^*{\to}\mathbb{N}_o$ encodes the magnitude of the absolute value of a coefficient, such that for any $a, b{\in}\mathbb{C}^*$

$$\begin{aligned} |a|{\leq}|b| &\text{ implies } \mathcal{O}(a){\geq}\mathcal{O}(b),\\ |a|{\approx}O(1) &\text{ implies } \mathcal{O}(a){=}0,\\ |a|{<}\delta &\text{ implies } \mathcal{O}(a){\geq}\mathcal{B}\ \big[\text{hence } a{\cdot}t{\cdot}e^{\mathcal{O}(a)}{\equiv}0 \text{ in } \mathcal{P}(e)\big]. \end{aligned}$$

Note that a smaller coefficient is encoded by a higher degree of e. The map Ω is extended to polynomials by linearity.

It is natural to define $\mathcal{O}(\cdot)$ in terms of the ε-weighted measure of magnitude given as

$$\mathcal{O}(\alpha){=}\big\lfloor \log_\varepsilon(|\alpha|)\big\rfloor \quad \text{for} \quad \alpha{\neq}0,\ |\alpha|{\leq}1.$$

Then Ω is defined for any polynomial p such that the absolute value of any coefficient in p is in the interval $(0, \dots, 1]$. The latter restriction on coefficients is not burdensome for our algorithm and can easily by achieved by an appropriate scaling of the polynomials. As a useful side effect, the scaling turns $\mathcal{O}$ into a relative order of magnitude of the coefficients in a polynomial.

2.2 Computation

The classical computation of a Gröbner basis in $\mathcal{P}(e)$ would require an admissible term order on $\mathbb{T}(e)$. Unfortunately this is not possible: the compatibility of an order with multiplication with terms cannot be satisfied due to the existence of zero-divisors in $\mathbb{T}(e)$. Therefore we use a weaker notion of a term order.

Definition 2. *An* **extended term order** $<$ *is a total well-order on* $\mathbb{T}(e)$ *such that* 0 *is the smallest term,* $<$ *coincides with* $<_{\mathbb{T}}$ *on* $\mathbb{T}$ *and*

$$a_1{<}a_2,\ b_1{\leq}b_2,\ a_2b_2{\neq}0$$

implies

$$a_1b_1{<}a_2b_2$$

for all $a_1, b_1{\in}\mathbb{T}(e)$*,* $a_2, b_2{\in}\mathbb{T}(e)^*$*.*

The following two *extended term orders* are currently used in our approach.

$<_{\mathbb{T}_e}$: The lexicographical product of the order $<_{\mathbb{T}}$ on $\mathbb{T}$ and the degree anti-compatible order on the (finitely many) powers of e with priority given to the main variables and "0" being the smallest term:
For any $s_1, s_2{\in}\mathbb{T}(e)$ we define $\mathbf{s_1}{<_{\mathbb{T}_e}}\mathbf{s_2}$ if and only if

$$\begin{aligned} s_2{\neq}0 \text{ and } & a)\ s_1 = 0 \text{ or}\\ & b)\ s_1 \neq 0 \text{ and } b_1)\mathrm{T}(s_1){<_{\mathbb{T}}}\mathrm{T}(s_2) \text{ or}\\ & \qquad\qquad\quad\ b_2)\,\mathrm{T}(s_1){=}\mathrm{T}(s_2) \text{ and}\\ & \qquad\qquad\qquad\ \deg_e(s_2){<}\deg_e(s_1). \end{aligned}$$

$<_{e\mathbb{T}}$: The lexicographical product of the degree anti-compatible order on the (finitely many) powers of e and the order $<_{\mathbb{T}}$ on $\mathbb{T}$ with priority given to the variable e and "0" being the smallest term:
For any $s_1, s_2{\in}\mathbb{T}(e)$ we define $\mathbf{s_1}{<_{e\mathbb{T}}}\mathbf{s_2}$ if and only if

$$\begin{aligned} s_2{\neq}0 \text{ and } & a)\ s_1{=}0 \text{ or}\\ & b)\ s_1{\neq}0 \text{ and } b_1)\deg_e(s_2){<}\deg_e(s_1) \text{ or}\\ & \qquad\qquad\quad\ b_2)\,\deg_e(s_2){=}\deg_e(s_1)\\ & \qquad\qquad\qquad\ \text{and } \mathrm{T}(s_1){<_{\mathbb{T}}}\mathrm{T}(s_2). \end{aligned}$$

Due to the modified notion of a term order, the standard theory of Gröbner bases is not readily suitable to provide an algorithm for computing Gröbner bases in $\mathcal{P}(e)$. Therefore we have developed a variation of the theory of Gröbner bases and established a correspondingly modified Buchberger algorithm for computing such bases in $\mathcal{P}(e)$ using an extended term order.

In the following, we fix an extended term order $<$ on $\mathbb{T}(e)$ and an e-bound $1{<}\mathcal{B}{\in}\mathbb{N}$. Let $\ell\mathrm{c}(f)$ and $\ell\mathrm{t}(f)$ denote the leading coefficient and the leading term of $0{\neq}f{\in}\mathcal{P}(e)$. For $F{\subseteq}\mathcal{P}(e)$, $\ell\mathrm{t}(F){:=}\{\ell\mathrm{t}(f) \mid 0{\neq}f{\in}F\}$.

The definition of a Gröbner basis in $\mathcal{P}(e)$ is essentially the same as in the standard setting.

Definition 3. *Let $I{\lhd}\mathcal{P}(e)$ be an ideal in $\mathcal{P}(e)$. A finite set $G{\subseteq}I\backslash\{0\}$ such that $\langle\ell\mathrm{t}(G)\rangle{=}\langle\ell\mathrm{t}(I)\rangle$ is called a* **Gröbner basis of** *I.*

We follow the usual convention where an arbitrary finite set $G{\subseteq}\mathcal{P}(e)\backslash\{0\}$ is called a Gröbner basis if and only if it is a Gröbner basis of the ideal which it generates.

The reason to keep the definition the same is the essentially same structure of monomial ideals (the ideals which admit a monomial basis) in $\mathcal{P}(e)$. Here it is crucial that the terms $\mathbb{T}(e)^*$ are linearly independent. Then it is easy to show that for any monomial ideal $M{\lhd}\mathcal{P}(e)$ there exists a finite generating set $S{\subseteq}\mathbb{T}(e)^*$ and a term $v{\in}\mathbb{T}(e)^*$ is in M if and only if $s \mid v$ for some $s{\in}S$.

These properties of monomial ideals allow to establish the following basic fact about Gröbner bases in $\mathcal{P}(e)$, the proof parallels exactly the respective proof in the standard theory of Gröbner bases; cf. Lemma 2.3.22 and Theorem 2.3.2 in [6].

Theorem 1. *Let $I{\lhd}\mathcal{P}(e)$ be an ideal in $\mathcal{P}(e)$. Then there exists a Gröbner basis $G{\subseteq}I\backslash\{0\}$ of I. Furthermore, G is a Gröbner basis of I if and only if for any $f{\in}I\backslash\{0\}$ there exists $g{\in}G$ such that $\ell\mathrm{t}(g) \mid \ell\mathrm{t}(f)$.*

Reduction The reduction of $g{\in}\mathcal{P}(e)$ modulo $F{\subseteq}\mathcal{P}(e)$ w.r.t. $<$ is essentially the same as in the standard theory of Gröbner bases. Also the notion of a reduced form of g modulo F remains unchanged.

However it is important to note that this is a numerically stable reduction being considered as a part of our approach. Indeed, assume that g and F are both obtained according to the translation described in Section 2.1 and $<$ is either $<_{\mathbb{T}_e}$ or $<_{{}_e\mathbb{T}}$. Let $f{\in}F$. Here it is crucial that the translation encodes a smaller order of magnitude of coefficients by a higher power of e: Let $s{\in}\operatorname{supp}(g)$ and $\alpha{\in}\mathbb{C}$ be the coefficient at s in g. Assume $\ell\mathrm{c}(f)$ is considerably smaller than α, such that $\mathcal{O}(\alpha){<}\mathcal{O}(\ell\mathrm{c}(f))$. Then $\deg_e(s){=}\mathcal{O}(\alpha) < \mathcal{O}(\ell\mathrm{c}(f)){=}\deg_e\big(\ell\mathrm{t}(f)\big)$ according to the translation mechanism. Hence $\ell\mathrm{t}(f){\nmid}s$ and therefore s can not be reduced modulo f provided that $\ell\mathrm{c}(f)$ is considerably smaller than α.

Furthermore, it is easy to see that $e{<}1$ w.r.t. $<_{\mathbb{T}_e}$ and $<_{{}_e\mathbb{T}}$. If the extended term order is $<_{{}_e\mathbb{T}}$ then the leading term function selects a term with the highest magnitude of coefficient in a polynomial which is desirable for the numerical stability of the reduction. For $<_{\mathbb{T}_e}$ this is not always true; therefore the reduction is less reliable.

Gröbner bases To compute a Gröbner basis in $\mathcal{P}(e)$ one has to deal with S-polynomials as in the usual Buchberger algorithm but also with an extra type of critical elements.

Definition 4. *A* **critical e-shift** *$E(f)$ of $0{\neq}f{\in}\mathcal{P}(e)$ is the shift $e^k{\cdot}f$ where $\mathcal{B}{\geq}k{>}0$ is the smallest degree of e such that $e^k{\cdot}\ell\mathrm{t}(f){=}0$.*

Our algorithm for computing Gröbner bases in $\mathcal{P}(e)$ is an easy adaptation of the Buchberger algorithm: reduce the S-polynomials as well as the critical e-shifts; if a remainder is not zero, add this remainder to the list of polynomials in the generating set; do this until there are "enough" polynomials to make all S-polynomials and critical e-shifts reduce to zero. The resulting Gröbner basis is auto-reduced to obtain a *totally-reduced* Gröbner basis which is uniquely defined.

Buchberger's first and second criteria for S-polynomials are easily translated to our context and used in our implementation as well as the following modification of Buchberger's second criterion for critical e-shifts: Let $f, g{\in}\mathcal{P}(e)\backslash\{0\}$. It is not necessary to reduce the critical e-shift $E(f)$ if the critical e-shift $E(g)$ as well as the S-polynomial $\mathcal{S}(g,f)$ are either reduced or excluded by another instance of whatever criteria to detect redundant critical elements and $\mathrm{T}\big(\ell\mathrm{t}(g)\big) | \mathrm{T}\big(\ell\mathrm{t}(f)\big)$.

In [6] we have established the aforementioned facts by developing a theory of Gröbner bases in $\mathcal{P}(e)$ in parallel with the standard theory of Gröbner bases but with the notion of an extended term order used in place of the admissible order of terms. This is an instructive although technically complicated example of a theory of Gröbner bases developed for a structure with zero-divisors. A detailed formal description in contained in Chapter 2 of [6]. This level of detail is not feasible here. Instead we point out a simple alternative to see the validity of the stated results: We may reinterpret the Gröbner basis computation of an ideal $\langle F\rangle$ in $\mathcal{P}(e)$ as a computation of the Gröbner basis of the ideal $\big\langle F_y \cup \{y^{\mathcal{B}}\}\big\rangle$ in $\mathcal{P}(y) = \mathbb{C}[X,y]$ with a new free variable y; an arbitrary term order $<_y$ on $\mathcal{P}(y)$ such that $\phi(w_1) < \phi(w_2)$ implies $w_1 <_y w_2$ for any terms $w_1, w_2 \in \mathcal{P}(y)$ where ϕ is the canonical epimorphism of $\mathcal{P}(y)$ on $\mathcal{P}(e)$ given as the unique homomorphic extension of the correspondence $y \to e$, the reduction strategy which keeps all polynomials totally reduced modulo $y^{\mathcal{B}}$, and F_y is a pre-image of F in $\mathcal{P}(e)$ under ϕ e.g. attained by the substitution $e \to y$ in each polynomial $f = f(X,e) \in F$. Note that due to a possibility $y <_y 1$, $<_y$ is generally a so-called *mixed ordering.* As a matter of fact, a standard theory of Gröbner bases and, in particular, the reduction algorithm have to be modified for the case of mixed orderings, cf. e.g. [9]. However, this is not necessary[2] in our case due to the relation $y^{\mathcal{B}} = 0$.

Numerical aspects The computation of a Gröbner basis in $\mathcal{P}(e)$ as a part of our approach would only be of moderate value for numerical purposes if we had no mechanism to keep a correspondence between the e-degree of a term and the true order of magnitude of the coefficient at the term in a polynomial. Let an input $F \subseteq \mathcal{P}(e) \setminus \{0\}$ for a Gröbner basis computation be obtained by the translation described in Section 2.1. Then the e-degrees in F are perfectly compliant with the true order of magnitude of the respective coefficients. The computation of a Gröbner basis of F in $\mathcal{P}(e)$ is naturally a sequence of operations with polynomials in $\mathcal{P}(e)$ where e acts as a conventional variable but the relation $e^{\mathcal{B}} = 0$ is plugged into the polynomial arithmetic. In particular, for any $s_1, s_2 \in \mathbb{T}(e)^*$

$$\begin{aligned} &\deg_e(s_1) + \deg_e(s_2) \geq \mathcal{B} \quad \text{if and only if } s_1 \cdot s_2 = 0, \\ &\deg_e(s_1 \cdot s_2) = \deg_e(s_1) + \deg_e(s_2) \quad \text{ for } s_1 \cdot s_2 \neq 0. \end{aligned}$$

Such a behavior of e-degrees is quite nice regarding the correspondence between the e-degree of a term and the true order of magnitude of the coefficient at the term in a polynomial during the computation of a Gröbner basis: if $\alpha_1 \cdot s_1$ and $\alpha_2 \cdot s_2$ are the monomials in some polynomials that have appeared in the flow of computation and have been multiplied then the e-degree $\deg_e(s_1) + \deg_e(s_2)$ of $\deg_e(s_1 \cdot s_2)$ clearly simulates the true order of magnitude of $\alpha_1 \cdot \alpha_2$; in particular, $\deg_e(s_1) + \deg_e(s_2) \geq \mathcal{B}$ indicates that we have hit the given numerical precision and then the term $s_1 \cdot s_2 \equiv 0$ is removed from the result of multiplication.

But it is also clear that the order of magnitude of coefficients generally behaves in a more complex way under multiplication than the degrees of e. Furthermore, the *addition* of polynomials in $\mathcal{P}(e)$ can create new small coefficients and this phenomenon is not reflected at all in the e-degrees. As an example consider the reduction of

$$f = x + (0.5 + \alpha) \cdot y + 2 \in \mathcal{P}(e) \text{ with } \alpha \in \mathbb{C}^*,\ 0 < |\alpha| \ll 1,\ \mathcal{O}(\alpha) > 0$$

modulo $g = x + 0.5 \cdot y + 1 \in \mathcal{P}(e)$ with $x > y > 1$. The result of the reduction $h = f - g = \alpha \cdot y + 1$ acquires a very small leading coefficient α which is not reflected by $\deg_e\big(\ell\mathrm{t}(h)\big) = \deg_e(y) = 0$.

In our implementation the e-degrees of the terms in intermediate polynomials are dynamically adjusted in the course of a Gröbner basis computation such that $\mathcal{O}(\alpha) \leq \deg_e(s)$ for any monomial $\alpha \cdot s$, $\alpha \in \mathbb{C}^*$, $s \in \mathbb{T}(e)$, of any intermediate polynomial; cf. Section 2.1. The corresponding mechanism is called the *trimming of e-degrees.* This mechanism may cause the flow of computation to deviate from the course described in Section 2.2. This is done for the sake of numerical stability. The trimming of e-degrees occurs as a part of the reduction process.

[2] Despite this fact, our implementation uses a carefully adjusted definition of *ecart* ([9]) in the reduction strategy as it allows to compute reduced forms in fewer steps.

Surely we have to follow certain restrictions to keep the algorithm operational.

First of all it is unconditionally forbidden to decrease the e-degree of any term. This restriction is stipulated by the reduction process in $\mathcal{P}(e)$: Indeed, consider a monomial $\alpha{\cdot}s$, $\alpha{\in}\mathbb{C}^*$, $s{\in}\mathbb{T}(e)^*$ in some intermediate polynomial $f{\in}\mathcal{P}(e)$ that is being reduced at some point. Both supported extensions ($<_{\mathbb{T}_e}$ and $<_{\underset{e}{\mathbb{T}}}$) of the term order assign a negative weight to e so that decreasing the e-degree of s we create a higher term in the order. But the trimming of e-degrees is a part of the reduction process which is performed as a successive elimination of the reducible terms modulo the current basis by replacing them with linear combinations of strictly smaller terms. Thus, if we decrease the e-degree of s during reduction then we break the important principle that a reduction process can never introduce any higher terms than have originally been present in f; this would generally break the termination property of the reduction process provided by the well-ordering of an extended term order.

There is also a numerical reason against decreasing the e-degree of s: if s has a positive e-degree (otherwise we can not decrease the e-degree anyway) then the coefficient α of s in f stems from a small coefficient which is the origin of the positive e-degree of s either initially present in F or being assigned by the trimming of e-degrees at an earlier point of computation. In this case α is an amplified small coefficient whose numerical value should not be trusted; hence we should not decrease the e-degree of s creating a term higher in the extended term order and thus giving α an extra chance to become the leading coefficient in the result of reduction of f.

Two other restrictions on the trimming of e-degrees are the following:

It is forbidden to change the e-degree of a term which is reducible modulo the current basis. Here the idea is transparent: if it is possible to reduce a term then we should do it instead of applying extra mechanisms to the term!

It is forbidden to change the e-degrees in the terms of the polynomials that have been already added to the current basis, it is particularly important not to change the e-degrees in their leading terms. This restriction keeps the interpretation of a Gröbner basis computation as the successive extension of the ideal of leading terms of the current basis which is crucial for the termination argument of a Gröbner basis algorithm.

2.3 Postprocessing

The last step of our approach is the reinterpretation in the original variables of a totally reduced Gröbner basis G in $\mathcal{P}(e)$. This is done by setting $e{:=}1$ in all polynomials of G. If certain conditions described below are satisfied then the result of the reinterpretation is an extended Gröbner basis in $\mathcal{P}$ w.r.t. $<_{\mathbb{T}}$. To reveal the meaning of this transition let us look at the situation from a slightly more general point of view. For $a{\in}\mathbb{C}$ denote the *partial evaluation at $e{=}a$* by

$$\Phi_a : \begin{cases} \quad P(e) \longrightarrow P \\ f(X,e) \longrightarrow f(X,a). \end{cases}$$

Note that Φ_a is not a homomorphism of rings unless $a{=}0$. The crucial statement is given in the following theorem.

Theorem 2. *Let $<$ be $<_{\mathbb{T}_e}$ or $<_{\underset{e}{\mathbb{T}}}$. If $G{\subseteq}\mathcal{P}(e)\backslash\{0\}$ is a Gröbner basis in $\mathcal{P}(e)$ w.r.t. $<$ and $\deg_e\big(\ell\mathrm{t}(g)\big){=}0$ for each $g{\in}G$ then $\Phi_o(G)$ is a Gröbner basis in $\mathcal{P}$ w.r.t. $<_{\mathbb{T}}$ and then $\ell\mathrm{t}(G){=}\ell\mathrm{t}\big(\Phi_o(G)\big)$.*

Proof. For the order $<_{\underset{e}{\mathbb{T}}}$ the proof is straightforward (see Theorem 2.6.2 in [6]).

For $<_{\mathbb{T}_e}$ it is considerably more complicated; the complete proof is the main result of Section 2.5.1 in [6]. Here it is not possible to provide details. Instead we list the major steps of the proof.

- G is a Gröbner basis in $(\mathbb{C}[e])[X]$ w.r.t. $<_{\underset{e}{\mathbb{T}}}$. This statement is similar to Theorem 4.1.18 in [1].
- There exists a **strong Gröbner basis** $\widetilde{G}{\supseteq}G$ of $\langle G\rangle$ in $(\mathbb{C}[e])[X]$. The existence and construction of a strong Gröbner basis in $(\mathbb{C}[e])[X]$ mimics the known construction for the case of a polynomial ring over a principal ideal domain, cf. e.g. Section 4.5 in [1], Section 4 in [7].
- $\Phi_o(\widetilde{G})$ is a Gröbner basis in $\mathcal{P}$ w.r.t. $<_{\mathbb{T}}$. This statement parallels a series of results in the literature which deal with Gröbner bases in a polynomial ring over a principal ideal domain [8], [17].

– By construction of $\widetilde{G}$, for any $f{\in}\Phi_o(\widetilde{G})\backslash\Phi_o(G)$ there exists $g{\in}\Phi_o(G)$ such that $\ell\mathrm{t}(g)\,|\,\ell\mathrm{t}(f)$. Then

$$\Phi_o(G) = \Phi_o(\widetilde{G}) \setminus \big(\Phi_o(\widetilde{G})\backslash\Phi_o(G)\big)$$

is a Gröbner basis in $\mathcal{P}$ w.r.t. $<_{\mathbb{T}}$. Here we apply a simple observation conventionally utilized for construction of a so-called minimal Gröbner basis. □

Now we are ready to present the meaning of the result of a computation carried out with our approach as formulated in the following lemma.

Lemma 1. *Let $<$ be either $<_{\mathbb{T}_e}$ or $\underset{e}{<}_{\mathbb{T}}$ and G be a totally reduced Gröbner basis in $\mathcal{P}(e)$ w.r.t. $<$. Assume that for each $g{=}\sum\limits_{s\in\,\mathrm{supp}(g)}\alpha_s(g)s \in G$ we have $\deg_e\big(\ell\mathrm{t}(g)\big){=}0$ and the coefficients $\alpha_s(g)$ are sufficiently small for $s{\in}\,\mathrm{supp}\,(g)$ such that $\deg_e(s){>}0$. Then $\Phi_1(G)$ is an extended Gröbner basis in $\mathcal{P}$ w.r.t. $<_{\mathbb{T}}$.*

Proof. Denote

$$N{=}\big\{t{\in}\mathbb{T} \;\big|\; \ell\mathrm{t}(g){\nmid}t \;\text{ for any }\; g{\in}G\big\} \subseteq \mathbb{T}$$

By Theorem 2 and the fact that G is totally reduced it is easy to see that $\Phi_o(G)$ is a totally reduced Gröbner basis in $\mathcal{P}$ w.r.t. $<_{\mathbb{T}}$ with the associated normal set N. Take any $g{=}\sum\limits_{s\in\,\mathrm{supp}(g)}\alpha_s(g)s \in G$. It is clear that

$$\ell\mathrm{t}(g) \in \mathrm{supp}\,(\Phi_o(g))\,, \quad \mathrm{supp}\,(\Phi_o(g))\,\backslash\{\ell\mathrm{t}(g)\} \subseteq N$$

and

$$\Phi_1(g) = \Phi_o(g) + h_g \text{ with } \; h_g{=}\sum_{\substack{s\in\,\mathrm{supp}(g)\\ \deg_e(s)>0}}\alpha_s(g)\mathrm{T}(s).$$

Using again the fact the G is totally reduced, $\mathrm{supp}\,(h_g)\,{\subseteq}N$. Hence

$$\ell\mathrm{t}(g) \in \mathrm{supp}\,\big(\Phi_1(g)\big) \quad\text{and}\quad \mathrm{supp}\,\big(\Phi_1(g)\big)\,\backslash\big\{\ell\mathrm{t}(g)\big\} \subseteq N.$$

The claim of the lemma is now immediate from the latter statement and Definition 1 of an extended Gröbner basis with $\widetilde{\mathcal{G}}{=}\Phi_o(G)$ and $\mathcal{G}{=}\Phi_1(G)$.

According to our experiments, the statement of Lemma 1 typically holds for a result G of computation in $\mathcal{P}(e)$ if the original system $\mathcal{F}$ is zero-dimensional and the geometry of solutions does not change under a small perturbation of $\mathcal{F}$. In particular, the coefficients of the terms with a positive e-degree of polynomials in G are typically small: a positive e-degree of a term in a polynomial can be seen as a *tag* assigned either by the initial translation to $\mathcal{P}(e)$ or at some point of computation in $\mathcal{P}(e)$ as an indication that the coefficient of the term in the polynomial is small. These tags propagate to G and thus the evaluation at $e{=}1$ can be considered as just removing the tags resulting in a system in $\mathcal{P}$. The situation when the conditions of Lemma 1 are not satisfied requires further research.

2.4 Linear algebra techniques

Our implementation utilizes a rearranged version of reduction with an extensive use of linear algebra to enhance the numerical stability in floating-point execution of the algorithm. This has originally been introduced by J.-C. Faugere in his F_4 algorithm [3]. It differs from a standard approach by a simultaneous reduction of several critical elements (S-polynomials and also critical e-shifts in our case) and the subdivision of reduction into two distinct phases: the symbolic reduction and the subsequent linear elimination to the row echelon form. We have modified the original F_4 version of reduction by the use of an essentially recursive symbolic reduction and a weak analogue to complete pivoting in the linear elimination.

The idea of the rearranged reduction is straightforward: Let $p, f{\in}\mathcal{P}(e)\backslash\{0\}$, $t{\in}\,\mathrm{supp}\,(p)$ and $\ell\mathrm{t}(f)\,|\,t$. Denote $v{=}t/\ell\mathrm{t}(f)$. An elementary reduction of p modulo f by eliminating t can be seen as computation of a *shift* $h{=}vf$ and the subtraction $p{:=}p{-}\alpha{\cdot}h$ with an appropriate coefficient $\alpha{\neq}0$

to eliminate t in p. Here it is crucial that $h{\neq}0$ and $\ell\mathrm{t}(h){=}v\ell\mathrm{t}(f)$ w.r.t. an extended term order $<$ (Corollary 2.3.6 in [6]).

This idea can be generalized in the following way. Let $P, F{\subseteq}\mathcal{P}(e)\backslash\{0\}$. Our goal is to compute a partially auto-reduced $\widetilde{P}{\subseteq}\mathcal{P}(e)\backslash\{0\}$ of reduced forms modulo F of polynomials P. Then $\widetilde{P}$ can be computed in two phases. The first phase is called the symbolic reduction according to [3]. Its purpose is to compute sufficiently many shifts for the subsequent second phase, which is the linear elimination.

Simplifying only slightly, the following version of the *symbolic reduction algorithm* is used in our implementation. It takes P and F as an input. The result of symbolic reduction is the set of reducible terms $RT{\subseteq}\mathbb{T}(e)^*$ and the corresponding shifts $H{\subseteq}\mathcal{P}(e)\backslash\{0\}$.

Symbolic reduction algorithm

```
  **Input:** $P$, $F$
  **Output:** $RT$, $H$
    $H := \emptyset, RT := \emptyset,\ Done := \emptyset$
    **while** supp $(P{\cup}H) \neq Done$
      choose $s$ in supp $(P{\cup}H) \setminus Done$, $Done{:=}Done{\cup}\{s\}$
      $D := \{f{\in}F \mid \ell\mathrm{t}(f) \mid s\} \cup \{p{\in}P{\cup}H \mid \ell\mathrm{t}(p) \mid s,\ \ell\mathrm{t}(p){\neq}s\}$
      **if** $D{\neq}\emptyset$ **then**
        choose $r$ in $D$
        $H := H{\cup}\{\frac{s}{\ell\mathrm{t}(r)}{\cdot}r\}$, $RT := RT{\cup}\{s\}$
      **end if**
    **end while**
```

At this point it is convenient to introduce extra terminology.

Definition 5. *Let $s{\in}\mathbb{T}(e)^*$ and $F{\subseteq}\mathcal{P}(e)\backslash\{0\}$. If there exists $f{\in}F$ such that $\ell\mathrm{t}(f) \mid s$ and $\ell\mathrm{t}(f){\neq}s$ then s is called* **properly reducible** *modulo F. If $s{\in}\ell\mathrm{t}(F)$ then s is called* **linearly reducible** *modulo F.*

The symbolic reduction can now be seen as a construction of $U{=}P{\cup}H$ where the reducibility of terms is equivalent to linear reducibility. The precise formulation is the following.

Theorem 3. *In the terms and notation above:*

(1) *(Termination) The symbolic reduction algorithm terminates.*
(2) *(Correctness) Let $s{\in}$ supp (U). If s is reducible modulo F or properly reducible modulo U then $s{\in}\ell\mathrm{t}(H)$.*
(3) *(Minimality) $RT{=}\ell\mathrm{t}(H)$, for any $s{\in}RT$ there exists one and only one $h{\in}H$ such that $s{=}\ell\mathrm{t}(h)$.*

Proof. The termination statement is established using the well-ordering of $<$ and König's lemma. The correctness is proved by a contradiction. The details can be found in Lemma 3.4.1 in [6]. The minimality statement is quite clear from the layout of the algorithm (Lemma 3.4.2 in [6]).

The second phase of reduction applies a linear algebra elimination to U. Our implementation uses a variant of a direct row echelon elimination.

At first let us size up the desired outcome of the linear elimination. Denote $N{=}$ supp (U), $IT{=}N \setminus RT$. An immediate consequence of part (3) of Theorem 3 is the direct decomposition $\langle N\rangle_{\mathbb{C}}{=}\langle H\rangle_{\mathbb{C}}{\oplus}\langle IT\rangle_{\mathbb{C}}$ (Lemma 3.4.7 in [6]) where $\langle\cdot\rangle_{\mathbb{C}}$ denotes the $\mathbb{C}$-linear vector span of a set of polynomials in $\mathcal{P}(e)$. In particular, for any $p{\in}P$ there exists the unique decomposition $p{=}\eta_p{+}\mathrm{NF}_H(p)$ with $\eta_p{\in}\langle H\rangle_{\mathbb{C}}$ and $\mathrm{NF}_H(p) \in\langle IT\rangle_{\mathbb{C}}$. Denote $\overline{P}{=}\{\mathrm{NF}_H(p) \mid p{\in}P\}$. Let $\widetilde{P}$ be $\overline{P}$ reduced to the row echelon form w.r.t. $<$. It is easy to see by parts (2), (3) of Theorem 3 and the choice of IT that $\widetilde{P}$ complies with our initial goal: it is a partially auto-reduced set of reduced forms modulo F of polynomials P.

In our implementation, $\widetilde{P}$ is obtained as a subset of U reduced to the row echelon form w.r.t. a specific order on N. This is possible due to the following fact.

Theorem 4. *Let $<_N$ be a linear order on N. Assume $<_N$ coincides with $<$ on IT and $t_1{<_N}t_2$ for any $t_1{\in}IT$, $t_2{\in}RT$. Let Q be U reduced to the row echelon form w.r.t. $<_N$. Denote $R{=}Q\cap\langle IT\rangle_{\mathbb{C}}$. Then R is equivalent to $\widetilde{P}$ up to scaling where $\widetilde{P}$ is as described above.*

Proof. Note that R and $\widetilde{P}$ are in the row echelon form by construction. Denote $K{=}\langle U\rangle_{\mathbb{C}}{\cap}\langle IT\rangle_{\mathbb{C}}$. A standard linear algebra technique (the details can be found in Lemma 3.4.8 in [6]) can be used to show the equivalence $\langle R\rangle_{\mathbb{C}}{=}K{=}\left\langle \widetilde{P}\right\rangle_{\mathbb{C}}$. Now the statement of the theorem is clear by the uniqueness (up to scaling) of a basis of a linear vector space in the row echelon form (w.r.t. $<_N$).

The statement of Theorem 4 allows an extra freedom in elimination. Namely, during the computation of Q, the order $<_N$ can be defined arbitrarily and even modified on RT. This possibility is used in the algorithm as a weak analogue to complete pivoting. We also perform the scaling of polynomials and the trimming of e-degrees in the row echelon elimination. The latter can introduce terms which are not present in N. However, this does not require a principal change in the algorithm.

In the implementation we actually compute a totally auto-reduced $\widetilde{P}$. It is done by an iteration of the described process. Also we use a trick with the treatment of S-polynomials to avoid numerical operations outside the row echelon elimination: For $f_1, f_2{\in}\mathcal{P}(e)\backslash\{0\}$, the S-polynomial $\mathcal{S}\,(f_1, f_2)$ is a linear combination of shifts $S_1{=}v_1 f_1$ and $S_2{=}v_2 f_2$ with $L{=}\mathrm{lcm}\big(\ell\mathrm{t}(f_1), \ell\mathrm{t}(f_2)\big)$, $v_1{=}L/\ell\mathrm{t}(f_1)$, $v_2{=}L/\ell\mathrm{t}(f_2)$. Instead of taking the linear combination, we pass S_1, S_2 to the symbolic reduction and set $L{\in}RT$. This can be interpreted as just a slight rearrangement of the described symbolic reduction with $S_1{\in}P$ and S_2 selected as the shift for L.

Another interesting observation comes from the proof of Theorem 4: what we actually need is the intersection $\langle U\rangle_{\mathbb{C}}{\cap}\langle IT\rangle_{\mathbb{C}}$ represented by its basis in the row echelon form (and this is the only place in the algorithm where numerical operations are performed!). From this point of view the row echelon elimination applied to U is nothing but a particular way to find it. This raises an interesting question for future research about numerically advantageous alternatives to compute this intersection of linear spaces.

2.5 Use of Floating-Point Arithmetic

To emphasize the algebraic aspects of our approach, we have so far assumed that exact computation is used in the numerical reduction to row echelon form. But, as initially stated, it has been the main goal of our endeavor to permit the use of floating-point arithmetic in the Gröbner basis computation for a polynomial system with numerical coefficients of limited accuracy.

Here, we have derived and discussed the design of an algorithm which is *numerically stable* under weak assumptions on the given system, viz. sufficient structural stability (or well-conditioning) of the zero set. Note also that, in our algorithm, reduction to zero has effectively been replaced by reduction to numerically irrelevant quantities through the relation $e^{\mathcal{B}}{=}\,0$. As is well-known in numerical analysis, the floating-point execution of a numerically stable algorithm for a well-conditioned problem leads to a result which structurally matches the result for exact computation but whose numerical data (here the coefficients of $\mathcal{G}$) deviate slightly from the exact result values.

Normally, this approximate floating-point result may immediately be interpreted as the exact result for slightly modified *initial* data. In the case of (genuine or extended) Gröbner bases, however, this is generally not possible because the representation of an ideal by a Gröbner basis is generally overdetermined.

A thorough discussion of this situation and the appropriate interpretation of an "approximate Gröbner basis" have been presented in the monograph [12] by H.J. Stetter; cf. section 9.2 of [12]. There it has also been shown how an approximate basis may be numerically refined towards the exact basis. We must refer the reader to this investigation.

3 A numerical example

The following example constructed by W. Windsteiger has been used in [11] to demonstrate the desired result of a stabilized Gröbner basis computation. Two ellipses are given as the zeros of h_1,

h_2 in $\mathbb{C}[x,y]$:

$$\begin{aligned} h_1 &= 1.027748\,y^2 - 0.467871\,xy + 2.972252\,x^2 + 0.662026\,y + 0.0785252\,x - 3.888889, \\ h_2 &= 3.958378\,y^2 + 0.701807\,xy + 1.041622\,x^2 - 0.0785252\,y + 0.662026\,x - 3.888889. \end{aligned}$$

They intersect in 4 real points, with angles of intersection not far from 90°. Thus we have a perfectly well-conditioned situation for the computation of the zeros. Fix a *lexicographic term order* $<_{\mathbb{T}}$ *with* $x <_{\mathbb{T}} y$. The *genuine Gröbner basis* of this system is (rounded)

$$\begin{aligned} g_1 &= x^4 - 0.1346456\,x^3 - 2.107266\,x^2 + 0.2423348\,x + 1.009172, \\ g_2 &= y + 1.355154 \cdot 10^{16}\,x^3 + 1.240075 \cdot 10^{16}\,x^2 - 1.55393 \cdot 10^{16}\,x - 1.3028 \cdot 10^{16} \end{aligned}$$

with the associated normal set $\{1, x, x^2, x^3\}$. It is clear from g_2 that this basis is not suited for the numerical computation of the zeros. A stable representation is given by the *extended Gröbner basis* (rounded)

$$\begin{aligned} \{&4.620929 \cdot 10^{-7}\,y + x^3 + 0.9150814\,x^2 - 1.146682\,x - 0.9613683, \\ &xy - 1.049727\,y - 4.156068\,x^2 + 0.1436151\,x + 4.428918, \\ &y^2 + 0.1662753\,y + x^2 + 0.1417843\,x - 1.767677\} \end{aligned}$$

with the associated normal set $\{1, x, x^2, y\}$.

With our algorithm it is now possible to *obtain this result automatically.* We use the extension $< := \underset{e}{<}_{\mathbb{T}}$ of $<_{\mathbb{T}}$ and $\mathcal{B}=2$. The accuracy of the system is assumed to be 2^{-23} which is the accuracy provided by the single-precision floating-point arithmetic utilized in the computation. In our setup $\mathcal{O}(\alpha) = \lfloor \log_{\frac{1}{2}}(|\alpha|)/12 \rfloor$ for $\alpha \in \mathbb{C}^*$, $|\alpha| \leq 1$.

At first the input system $\{h_1, h_2\}$ is totally auto-reduced and scaled resulting in

$$\begin{aligned} f_1 &= -0.1625218\,xy + 0.1706035\,y + 0.6754516\,x^2 - 0.02334059\,x - 0.7197958, \\ f_2 &= 0.4947973\,y^2 + 0.08227258\,y + 0.4947975\,x^2 + 0.07015449\,x - 0.8746419. \end{aligned}$$

Let $F = \{f_1, f_2\}$. Now the algorithm constructs the shifts $S_1 = y f_1$ and $S_2 = x f_2$. A linear combination of S_1 and S_2 would give an S-polynomial $\mathcal{S}(f_1, f_2)$ but it is not computed in order to avoid numerical operations at this point. Instead we perform the reduction of $P = \{S_1\}$: the first term considered by the symbolic reduction is $y^2x = \ell\mathrm{t}(S_1) = \ell\mathrm{t}(S_2)$ with S_2 selected as the shift for y^2x which leads to $RT = \{y^2x\}$ and $H = \{S_2\}$. Then the symbolic reduction proceeds in the usual way. The result is $RT = \{y^2x, y^2, yx^2, yx\}$ and the respective shifts $H = \{S_2, f_2, xf_1, f_1\}$.

At the second phase of reduction the row echelon elimination is applied to $U = P \cup H$ with $N = \operatorname{supp}(U) = \{y^2x, y^2, yx^2, yx, y, x^3, x^2, x, 1\}$, $IT = N \setminus RT = \{y, x^3, x^2, x, 1\}$. Initially the order $<_N$ coincides with $<$ on RT and IT and $t_1 <_N t_2$ for $t_1 \in RT$, $t_2 \in IT$. The elimination is performed with complete pivoting on RT and partial pivoting on IT. During this elimination, S_1 acquires a small coefficient at y. This triggers the trimming of e-degrees $y \to y \cdot e$ in S_1. Note that y remains unchanged in other polynomials! The new term ye is added to IT. Without the trimming of e-degrees the term y with the small coefficient would have been the leading term of the reduced S_1. This S_1 would have propagated to the resulting genuine Gröbner basis and made it poorly suited for the numerical determination of solutions.

After the elimination is finished, S_1 is the only polynomial in the reduced U which is a linear combination of IT:

$$S_1 := 0.6771894\,x^3 + 0.6196834\,x^2 - 0.776521\,x - 0.6510285 + 3.129244 \cdot 10^{-7}\,ye.$$

Hence it is the result of the reduction.

The S-polynomial $\mathcal{S}(S_1, f_2)$ is excluded by Buchberger's first criterion. The S-polynomial $\mathcal{S}(S_1, f_1)$ is reduced to zero using the same reduction process as before. No critical e-shifts are to be considered for S_1, f_1 and f_2 because the leading terms of all these polynomials are terms in $\mathbb{T}$. Hence $G = \{S_1, f_1, f_2\}$ is a Gröbner basis in $\mathbb{C}[x, y, e]$ and it is totally reduced. Furthermore, G satisfies the conditions of Lemma 1. Hence the partial evaluation of G at $e=1$ gives an extended Gröbner basis $\mathcal{G}$. It is straightforward to check that $\mathcal{G}$ is equivalent up to scaling to the extended Gröbner basis listed at the beginning of this section.

4 Conclusion

We have demonstrated how the machinery of Gröbner bases can be adapted to a numerical environment. The implemented algorithm based on these modifications performs well on typical examples. In classical benchmark problems such as Cyclic6 it can help to reduce the computation time. In the current version of the algorithm certain parameters of computation (like $\mathcal{B}$ and $\mathcal{O}$) need to be provided by the user. It would be desirable to make the algorithm choose these parameters automatically.

References

1. W. Adams and P. Loustaunau. *An Introduction to Grobner Bases.* AMS, 1994.
2. T. Becker. Gröbner bases versus $d-$Gröbner bases, and Gröbner bases under specialization. *AAECC*, 5:1–8, 1994.
3. J.-C. Faugere. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.
4. E. Fortuna, P. Gianni, and B. Trager. Degree reduction under specialization. *Journal of pure and applied algebra*, 164(1-2):153–164, 2001.
5. M. Kalkbrener. On the stability of Gröbner bases under specializations. *Journal of Symbolic Computation*, 24(1):51–58, 1997.
6. A. Kondratyev. *Numerical computation of Gröbner bases.* Dissertation, RISC, Johannes Kepler University, Linz, Austria, 2003. Online available at `ftp://ftp.risc.uni-linz.ac.at/pub/people/akondra/thesis.ps.gz`.
7. H. M. Möller. On the Construction of Gröbner Bases Using Syzygies. *Journal of Symbolic Computation*, 6(2&3):345–360, 1988.
8. F. Pauer. On lucky ideals for Gröbner basis computations. *Journal of Symbolic Computation*, 14(5):471–482, 1992.
9. H. Schönemann. Algorithms in singular. In *Reports On Computer Algebra*, number 02. Centre for Computer Algebra, University of Kaiserslautern, 1996. Online available at `http://www.mathematik.uni-kl.de/~zca`.
10. K. Shirayanagi. Floating point Gröbner bases. *Math. Comput. Simulation*, 42:509–528, 1996.
11. H. J. Stetter. Stabilization of polynomial systems solving with Gröbner bases. In *Proceedings ISSAC'97*, pages 117–124, New York, 1997. ACM Press.
12. H. J. Stetter. *Numerical Polynomial Algebra.* SIAM, 2004.
13. C. Traverso. Syzygies, and the stabilization of numerical buchberger algorithm. In *Proceedings LMCS 2002*, pages 244–255, RISC-Linz.
14. C. Traverso. Gröbner trace algorithms. In *Proceedings ISSAC'88*, pages 125–138, Berlin - Heidelberg - New York, 1989. Springer.
15. C. Traverso and A. Zanoni. Numerical stability and stabilization of Gröebner basis computation. In *Proceedings ISSAC'2002*, pages 262–269, New York, NY 10036, USA, 2002. ACM Press.
16. V. Weispfenning. Gröbner bases for inexact input data. In *Proceedings CASC '03*, pages 403–411, Passau, Germany, 2003.
17. F. Winkler. A p-adic approach to the computation of Gröbner bases. *Journal of Symbolic Computation*, 6(2–3):287–304, 1988.