数式処理 *J.JSSAC (2006)*
Vol. 12, No. 3, pp. 21 - 29
奨励賞受賞論文

Using Coefficient-wise Tolerance in Symbolic-Numeric Algorithms for Polynomials*

Kosaku Nagasaka†

Faculty of Human Development, Kobe University

(RECEIVED 2005/9/26)

1 Introduction

Symbolic-numeric algorithms have been studied by several researchers since early 1990s in the world. With recent studies of approximate factorizations, approximate GCDs and others for numerical or empirical polynomials, we can operate such computations with non-exact coefficients. Those algorithms work well on several Computer Algebra Systems (Mathematica and Maple, for example). Especially on Maple, we can use SNAP (Symbolic Numeric Algorithms for Polynomials) package to compute approximate GCDs.

In various implementations of such algorithms, bounding errors on coefficients are provided by polynomial norms (1-norm, 2-norm and ∞-norm, for example) in general. Because operating with non-exact coefficients involves computing numerical properties (eg. singular values) and polynomial norms have high affinities for matrix norms and operations. In fact, lots of algorithms for empirical polynomials use the singular value decomposition. However, from a practical point of view, tracking or treating errors as polynomial norms is not applicable as much as floating-point numbers, since for floating-point numbers, we have a few kind of significance arithmetics (interval numbers, for example).

In this paper, we review known symbolic numeric algorithms in terms of significance arithmetics, and introduce an absolute irreducibility testing for bivariate polynomials and a relatively prime testing for univariate polynomials. Basically, there is no difference from those original algorithms, but they would be more natural for the people who use significance arithmetics and give more efficient error bounds.

We note that this paper is involved with the authors' presentation at the 14th Workshop of JSSAC (Japan Society for Symbolic and Algebraic Computation), 2005.

* This research is partly helped by Grants-in-Aid of MEXT, JAPAN, #16700016.

†nagasaka@main.h.kobe-u.ac.jp

2 Coefficient-wise and Polynomial Norm Tolerance

In this section, we review polynomial norms and a coefficient-wise tolerance and their relation briefly. Let $f(x_1, \ldots, x_r)$ be the given polynomial as follows.

$$f(x_1, \ldots, x_r) = \sum_{x_1^{e_1} \cdots x_r^{e_r} \in \mathrm{supp}(f)} c_{e_1,\ldots,e_r} x_1^{e_1} \cdots x_r^{e_r} \in \mathbf{C}[x_1, \ldots, x_r],$$

where $\mathrm{supp}(f)$ denotes the set of power products of terms whose coefficients are not zero. In most algorithms, the following definitions are used for polynomial 1-norm, 2-norm and ∞-norm, respectively.

$$\begin{aligned} \|f\|_1 &= \textstyle\sum_{x_1^{e_1} \cdots x_r^{e_r} \in \mathrm{supp}(f)} |c_{e_1,\ldots,e_r}|, \\ \|f\|_2 &= \sqrt{\textstyle\sum_{x_1^{e_1} \cdots x_r^{e_r} \in \mathrm{supp}(f)} |c_{e_1,\ldots,e_r}|^2}, \\ \|f\|_\infty &= \max_{x_1^{e_1} \cdots x_r^{e_r} \in \mathrm{supp}(f)} |c_{e_1,\ldots,e_r}|. \end{aligned}$$

For example, the 1-norm, 2-norm and ∞-norm of $4x_1^2 - 3x_1x_2 + 2x_1 + 1$ are 10, $\sqrt{30}$ and 4, respectively. If this polynomial has numerical errors on their coefficients, for example, using a polynomial norm, we say that this polynomial has an error part whose coefficients have numerical errors at most 10^{-2} in 2-norm. This means that the above polynomial can be $4x_1^2 - 3x_1x_2 + 2x_1 + 1.01$, $4x_1^2 - 3.001x_1x_2 + 2x_1 - 0.999$ or the like. To represent this, we denote numerical or empirical polynomials as polynomial sets as follows.

$$P_*(f, \varepsilon) = \{\tilde{f} \mid \tilde{f} \in \mathbf{C}[x_1, \ldots, x_r], \ \deg_{x_i} \tilde{f} \le \deg_{x_i} f, \ \|f - \tilde{f}\|_* \le \varepsilon\},$$

where $*$ denotes 1, 2 or ∞ and ε plays the error bound of the given polynomial $f(x_1, \ldots, x_r)$.

On the other hand, we can bound numerical errors on a coefficient basis, as follows. We call it coefficient-wise tolerance. Let consider that each coefficient $c_{e_1,\ldots,e_r}$ involves numerical error which is bounded by $\varepsilon_{e_1,\ldots,e_r} \in \mathbf{R}_{\ge 0}$. This means, for example, $c_{0,0}$ can be represented by an interval number $[0.999, 1.001]$, for $c_{0,0} = 1.0$ and $\varepsilon_{0,0} = 10^{-3}$. Therefore, using these coefficient-wise tolerances, we can denote numerical or empirical polynomials as polynomial sets as follows.

$$\begin{aligned} P_{cw}(f) = \{\tilde{f} \mid \tilde{f} \in \mathbf{C}[x_1, \ldots, x_r], \ \mathrm{supp}(\tilde{f}) \subseteq \mathrm{supp}(f), \ |\tilde{c}_{e_1,\ldots,e_r} - c_{e_1,\ldots,e_r}| \le \varepsilon_{e_1,\ldots,e_r}, \\ \tilde{f} = \textstyle\sum_{x_1^{e_1} \cdots x_r^{e_r} \in \mathrm{supp}(\tilde{f})} \tilde{c}_{e_1,\ldots,e_r} x_1^{e_1} \cdots x_r^{e_r}\}. \end{aligned}$$

We note that this representation can be defined by using a weighted ∞-norm for polynomials. A weight is depending on the given coefficients. Hence, in the future, this coefficient-wise tolerance will be useless if we use weighted norms easily.

Remark 1
Some computer algebra systems have significance arithmetics other than interval numbers (see [4, 11]). Those error tracking systems are useful for users who want both speed and accuracy. However, most of such significance arithmetics sacrifice accuracy to gain speed and not to tend to become over-estimated. This means that the results might be incorrect mathematically in some cases.

3 Coprimality Testing

For exact computations, coprimality testing can be done by calculating the rank of the Sylvester matrix of the given two polynomials. We review this briefly. Let g and h be the following univariate

polynomials, respectively.

$$g(x) = \sum_{i=0}^{n} g_i x_1^i, \quad h(x) = \sum_{i=0}^{m} h_i x_1^i \ \in \ \mathbf{C}[x_1].$$

We have the well-known lemma that g and h are coprime if the following Sylvester matrix of g and h is of full rank (see usual text books for details).

$$\mathrm{Syl}(g,h) = \begin{pmatrix}
g_n & & & & h_m & & & & \\
g_{n-1} & g_n & & & h_{m-1} & h_m & & & \\
\vdots & g_{n-1} & \ddots & & \vdots & h_{m-1} & \ddots & & \\
\vdots & \vdots & \ddots & g_n & h_1 & \vdots & \ddots & \ddots & \\
\vdots & \vdots & \vdots & g_{n-1} & h_0 & h_1 & \vdots & \ddots & h_m \\
g_0 & \vdots & \vdots & \vdots & & h_0 & \ddots & \vdots & h_{m-1} \\
 & g_0 & \vdots & \vdots & & & \ddots & \ddots & \vdots \\
 & & \ddots & \vdots & & & & \ddots & h_1 \\
 & & & g_0 & & & & & h_0
\end{pmatrix}.$$

For numerical or empirical polynomials, the given coefficients may be not correct, so the rank of the Sylvester matrix also may change. We have to think its alternative way for non-exact polynomials. In general, this is done by using singular value decomposition in part, in various symbolic numeric algorithms [1, 9, 14, 13], because singular values have the following property for example, so they are very useful to operate with perturbations.

Lemma 2 (Corollary 8.6.2 in [3])
For any matrices A and $E \in \mathbf{C}^{n\times m}$ and any positive integer $k \leq \min\{n, m\}$, we have

$$|\sigma_k(A + E) - \sigma_k(A)| \leq \|E\|_2,$$

where $\sigma_k(A)$ denotes the k-th largest singular value of A.

We review its basic idea for polynomial sets $P_2(g, \varepsilon_g)$ and $P_2(h, \varepsilon_h)$, where $P_2(*, *)$ is defined in the previous section. Since Frobenius norm is always greater than or equal to 2-norm, we have the following inequality for any polynomial pair $\tilde{g} \in P_2(g, \varepsilon_g)$ and $\tilde{h} \in P_2(h, \varepsilon_h)$,

$$\|\mathrm{Syl}(\tilde{g}, \tilde{h}) - \mathrm{Syl}(g, h)\|_2 \leq \sqrt{m\varepsilon_g^2 + n\varepsilon_h^2}.$$

Hence, by the above lemma, for any positive integer $k \leq n + m$, we have

$$|\sigma_k(\mathrm{Syl}(\tilde{g}, \tilde{h})) - \sigma_k(\mathrm{Syl}(g, h))| \leq \sqrt{m\varepsilon_g^2 + n\varepsilon_h^2}.$$

Therefore, any polynomial pair $\tilde{g} \in P_2(g, \varepsilon_g)$ and $\tilde{h} \in P_2(h, \varepsilon_h)$ are coprime if

$$\sigma_{n+m}(\mathrm{Syl}(g, h)) > \sqrt{m\varepsilon_g^2 + n\varepsilon_h^2}. \tag{1}$$

If the given two polynomials are not close to polynomials that have common factors, coprimality testing using the inequality (1) works well. However, the right hand side of the inequality may be over-estimated for most cases, so the inequality does not work for polynomials that are close to polynomials that have common factors. Especially for polynomials with coefficients whose errors are bounded by coefficient-wise, the right formula easily computes over-estimated values. For such polynomials, we should use coefficient-wise error bounds directly.

We rewrite two polynomial sets of g and h as follows.

$$P_{cw}(g) = \{\tilde{g} \mid \tilde{g} \in \mathbf{C}[x_1],\ \deg_{x_1} \tilde{g} \le \deg_{x_1} g,\ |\tilde{g}_i - g_i| \le \varepsilon_{g,i},\ \tilde{g} = \textstyle\sum_{i=0}^{n} \tilde{g}_i x_1^i\},$$
$$P_{cw}(h) = \{\tilde{h} \mid \tilde{h} \in \mathbf{C}[x_1],\ \deg_{x_1} \tilde{h} \le \deg_{x_1} h,\ |\tilde{h}_i - h_i| \le \varepsilon_{h,i},\ \tilde{h} = \textstyle\sum_{i=0}^{m} \tilde{h}_i x_1^i\}.$$

We have $\|\mathrm{Syl}(\tilde{g}, \tilde{h}) - \mathrm{Syl}(g, h)\|_2 \le \|E\|_2$ where

$$E = \begin{pmatrix}
\varepsilon_{g,n} & & & & \varepsilon_{h,m} & & & & \\
\varepsilon_{g,n-1} & \varepsilon_{g,n} & & & \varepsilon_{h,m-1} & \varepsilon_{h,m} & & & \\
\vdots & \varepsilon_{g,n-1} & \ddots & & \vdots & \varepsilon_{h,m-1} & \ddots & & \\
\vdots & \vdots & \ddots & \varepsilon_{g,n} & \varepsilon_{h,1} & \vdots & \ddots & \ddots & \\
\vdots & \vdots & \vdots & \varepsilon_{g,n-1} & \varepsilon_{h,0} & \varepsilon_{h,1} & \vdots & \ddots & \varepsilon_{h,m} \\
\varepsilon_{g,0} & \vdots & \vdots & \vdots & & \varepsilon_{h,0} & \ddots & \vdots & \varepsilon_{h,m-1} \\
 & \varepsilon_{g,0} & \vdots & \vdots & & & \ddots & \ddots & \vdots \\
 & & \ddots & \vdots & & & & \ddots & \varepsilon_{h,1} \\
 & & & \varepsilon_{g,0} & & & & & \varepsilon_{h,0}
\end{pmatrix}.$$

Therefore, any polynomial pair $\tilde{g} \in P_{cw}(g)$ and $\tilde{h} \in P_{cw}(h)$ are coprime if

$$\sigma_{n+m}(\mathrm{Syl}(g, h)) \ > \ \|E\|_2. \tag{2}$$

This inequality is much better than the inequality (1). When we implement recent algorithms, we should implement also coefficient-wise version of algorithms in this point of view.

Remark 3
For numerical computations, root finding can be done by several numerical methods. Terui and Sasaki [12] studied computing error bounds (or existence domains) of zeros of empirical polynomials in our coefficient-wise tolerance fashion.

4 Absolute Irreducibility Testing

Absolute irreducibility testing for empirical bivariate polynomials is done by Kaltofen and May's algorithm [5] and its variant algorithms [7, 8]. Those algorithms and the primality testing noted in the previous section have the same property that they use matrices whose elements are directly generated from coefficients. This means that we can easily extend those absolute irreducibility testing algorithms to coefficient-wise tolerance, by the same way for the coprimality testing in the previous section.

At first, we show their original algorithms briefly. In those algorithms [5, 7], we compute a separation bound $B(f) \in \mathbf{R}_{>0}$ for the given polynomial $f \in \mathbf{C}[x_1, x_2]$, such that any $\tilde{f} \in \mathbf{C}[x_1, x_2]$

satisfying $\|f - \tilde{f}\|_2 < B(f)$ (and $\deg(\tilde{f}) \leq \deg(f)$) is absolutely irreducible. Such a bound can be computed by using Ruppert matrix $R(f)$ which is the coefficient matrix of a certain linear system and the size of Ruppert matrix $R(f)$ is $(4nm) \times (2nm + m - 1)$ where $n = \deg_{x_1}(f)$ and $m = \deg_{x_2}(f)$.

Let $f(x_1, x_2)$ be the following bivariate polynomial

$$f(x_1, x_2) = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} x_1^i x_2^j, \quad c_{i,j} \in \mathbf{C}.$$

The Ruppert matrix $R(f)$ can be written as

$$R(f) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j} c_{i,j}, \quad R_{i,j} \in \mathbb{Z}^{(4nm)\times(2nm+m-1)},$$

where each elements of $R_{i,j}$ is an integer and defined as the following figure where $\delta_{i,j}$ denotes Kronecker delta and the block matrices G_i and H_i are the matrices of sizes $2m \times (m+1)$ and $2m \times (m-1)$, respectively. A separation bound can be computed by the following expression, and we can test its absolute irreducibility by comparing to its error bound in polynomial 2-norm.

$$B(f) = \sqrt{6}\, \sigma_{2nm+m-1}(R(f)) / \sqrt{n(m(m+1)(2m+1) + (m-1)(n+1)(2n+1))}. \tag{3}$$

The concept of separation bounds is based on treating perturbations by polynomial 2-norm, so it may be over-estimated if we can use coefficient-wise tolerance for the given polynomial. In fact, numerical numerator and denominator on the right hand side of the expression (3) is a due to use polynomial 2-norm. Hence, we extend it to coefficient-wise tolerance.

We rewrite the given polynomial $f(x_1, x_2)$ with perturbations as follows.

$$\begin{aligned} P_{cw}(f) = \{\tilde{f} \mid \tilde{f} \in \mathbf{C}[x_1, x_2], \quad & \deg_{x_i} \tilde{f} \leq \deg_{x_i} f, \\ & |\tilde{f}_{i,j} - f_{i,j}| \leq \varepsilon_{i,j}, \quad \tilde{f} = \textstyle\sum_{i=0}^{n} \sum_{j=0}^{m} \tilde{f}_{i,j} x_1^i x_2^j\}. \end{aligned}$$

We have $\|R(\tilde{f}) - R(f)\|_2 \leq \|\sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j} \varepsilon_{i,j}\|_2$ for any polynomial $\tilde{f} \in P_{cw}(f)$. We note that the above separation bounds are based on Ruppert criterion [10] which guarantees the given polynomial is absolutely irreducible if its Ruppert matrix is of full rank. Therefore, any polynomial $\tilde{f} \in P_{cw}(f)$ is absolutely irreducible if

$$\sigma_{2nm+m-1}(R(f)) \; > \; \|\sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j} \varepsilon_{i,j}\|_2. \tag{4}$$

Remark 4
The above extension is only available for absolute irreducibility testing. Since we have to use the smallest singular value of the Ruppert matrix, we can not bound each element of the matrix. Hence, this extension to coefficient-wise tolerance can not be applied to separation bounds.

5 Sparsity of Empirical Polynomials

There is a large difference between coefficient-wise and polynomial norm tolerances especially for sparse polynomials. For example, we consider the following sparse polynomial and its polynomial sets.

$$\begin{aligned} & f(x_1) = x^7 + 1, \\ & P_{cw}(f) = \{\tilde{f} \mid \tilde{f} \in \mathbf{C}[x_1], \ |\tilde{c}_7 - 1| \leq 10^{-2}, \ |\tilde{c}_0 - 1| \leq 10^{-2}, \ \tilde{f} = \tilde{c}_7 x^7 + \tilde{c}_0\}, \\ & P_2(f, \sqrt{2} \cdot 10^{-2}) = \{\tilde{f} \mid \tilde{f} \in \mathbf{C}[x_1], \ \deg_{x_1} \tilde{f} \leq \deg_{x_1} f, \ \|\tilde{f} - f\|_2 \leq \sqrt{2} \cdot 10^{-2}\}. \end{aligned}$$

$$
R_{i,j} =
\begin{pmatrix}
\delta_{i,n}G_n & \cdots & 0 & 0\cdot\delta_{i,n}H_n & 0 & \cdots \\
\delta_{i,n-1}G_{n-1} & \ddots & \vdots & -\delta_{i,n-1}H_{n-1} & \delta_{i,n}H_n & \ddots \\
\vdots & \ddots & 0 & \vdots & 0\cdot\delta_{i,n-1}H_{n-1} & \ddots \\
\delta_{i,1}G_1 & \ddots & \delta_{i,n}G_n & (1-n)\delta_{i,1}H_1 & \vdots & \ddots \\
\delta_{i,0}G_0 & \ddots & \delta_{i,n-1}G_{n-1} & -n\delta_{i,0}H_0 & (2-n)\delta_{i,1}H_1 & \ddots \\
0 & \ddots & \vdots & 0 & (1-n)\delta_{i,0}H_0 & \ddots \\
\vdots & \ddots & \delta_{i,1}G_1 & \vdots & 0 & \ddots \\
0 & \cdots & \delta_{i,0}G_0 & 0 & \cdots & 0
\end{pmatrix}
$$

$$
H_i =
\begin{pmatrix}
0 & \cdots & 0 \\
\delta_{j,m} & \ddots & \vdots \\
\delta_{j,m-1} & \ddots & 0 \\
\vdots & \ddots & \delta_{j,m} \\
\delta_{j,1} & \ddots & \delta_{j,m-1} \\
\delta_{j,0} & \ddots & \vdots \\
 & \ddots & \delta_{j,1} \\
0 & & \delta_{j,0}
\end{pmatrix},
\qquad
\begin{pmatrix}
0 & 0 \\
\vdots & \vdots \\
0 & \vdots \\
(n-1)\delta_{i,n}H_n & 0 \\
(n-2)\delta_{i,n-1}\times H_{n-1} & nH_n \\
\vdots & (n-1)\delta_{i,n-1}\times H_{n-1} \\
0\cdot\delta_{i,1}H_1 & \vdots \\
-\delta_{i,0}H_0 & \delta_{i,1}H_1
\end{pmatrix},
$$

$$
G_i =
\begin{pmatrix}
0 & 0 & \cdots & 0 & 0 & 0 \\
\delta_{j,m-1} & -\delta_{j,m} & \ddots & \vdots & \vdots & 0 \\
2\delta_{j,m-2} & 0 & \ddots & 0 & \vdots & \vdots \\
\vdots & \delta_{j,m-2} & \ddots & (2-m)\,\delta_{j,m} & 0 & \vdots \\
\vdots & \vdots & \ddots & \vdots & (1-m)\,\delta_{j,m} & 0 \\
m\,\delta_{j,0} & \vdots & \ddots & \vdots & \vdots & -m\,\delta_{j,m} \\
0 & (m-1)\,\delta_{j,0} & \ddots & 0 & \vdots & \vdots \\
\vdots & 0 & \ddots & \delta_{j,1} & -\delta_{j,2} & \vdots \\
0 & \vdots & \ddots & 2\delta_{j,0} & 0 & -2\delta_{j,2} \\
0 & 0 & \cdots & 0 & \delta_{j,0} & -\delta_{j,1}
\end{pmatrix}.
$$

The set $P_{cw}(f)$ is absolutely smaller than another set $P_2(f, \sqrt{2} \cdot 10^{-2})$ as follows.

$$\begin{array}{lll} x^7 + 1.001 & \in P_{cw}(f), & \in P_2(f, \sqrt{2} \cdot 10^{-2}), \\ 1.01x^7 + 0.999 & \in P_{cw}(f), & \in P_2(f, \sqrt{2} \cdot 10^{-2}), \\ x^7 + 0.00000001x^6 + 1 & \notin P_{cw}(f), & \in P_2(f, \sqrt{2} \cdot 10^{-2}), \\ x^7 + 0.01x^6 - 0.01x + 1 & \notin P_{cw}(f), & \in P_2(f, \sqrt{2} \cdot 10^{-2}). \end{array}$$

Hence, we should use coefficient-wise tolerance for such sparse polynomials. However, one may think that empirical polynomials may have tiny coefficients so we can not use their sparsity. This is true in some cases, for example, computing Lagrange interpolating polynomials from empirical data. On the other hand, this is not true in some cases, for example, computing interpolating polynomials that have only a few terms by least squares. Therefore, we should distinguish exactly zero coefficients and approximately zero coefficients to operate with empirical polynomials.

>From the above point of view, we can use another algorithm for computing separation bounds. The author [8] introduced the sparse version of the algorithms noted in the previous section. In the algorithm (see [8] for details), we compute a separation bound $\bar{B}(f) \in \mathbf{R}_{>0}$ for the given polynomial $f \in \mathbf{C}[x_1, x_2]$, such that any $\tilde{f} \in \mathbf{C}[x_1, x_2]$ satisfying $\mathcal{P}(\tilde{f}) \subseteq \mathcal{P}(f)$ and $\|f - \tilde{f}\|_2 < \bar{B}(f)$ is absolutely irreducible, where $\mathcal{P}(p)$ means the Newton polytope of a polynomial p. We note that the Newton polytope of a polynomial $p = \sum_{i,j} a_{i,j} x^i y^j$ is defined as the convex hull in the Euclidean plane $\mathbf{R}^2$ of the exponent vectors (i, j) of all the nonzero terms of p. Hence, we are encouraged to use this algorithm to compute separation bounds of empirical polynomials which have exactly zero coefficients in part.

Remark 5
We note that the algorithm [8] uses another certain matrix $\mathcal{R}(f)$ which is similar to Ruppert matrix and have the same shape of Ruppert matrix but some columns are zeros (the integer matrices $R_{i,j}$ are the same). The criterion due to Gao and Rodrigues [2] requires computing the $(\rho - 1)$-th largest singular value while the criterion due to Ruppert requires the $(2nm + m - 1)$-th largest value, where $\rho - 1$ denotes a certain integer that is smaller than $2nm + m - 1$ mostly. Hence, for absolute irreducibility testing, we are also encouraged to use this algorithm for sparse empirical polynomials.

6 Examples

In this section, examples showing that coefficient-wise tolerance is better than polynomial norms, are presented. We note that numbers in this section are rounded at the fifth significant digit.

At first, we consider the following two polynomials and their coprimality.

$$g(x_1) = x_1^7 + 1.1, \quad h(x_1) = x_1^2 - 1.$$

We suppose that any non-presented coefficient is exactly zero and any presented coefficient can be perturbed within 0.01 at most such that $\varepsilon_{g,7} = \varepsilon_{g,0} = \varepsilon_{h,2} = \varepsilon_{h,0} = 0.01$ and $\varepsilon_g = \varepsilon_h = 0.01\sqrt{2}$. In this case, we can not determine their coprimality if we use the polynomial norm tolerance. Because we have

$$\sigma_{n+m}(\mathrm{Syl}(g, h)) = 0.02137, \quad \sqrt{m\varepsilon_g^2 + n\varepsilon_h^2} = 0.04243.$$

However, we can say that they are coprime if we use the coefficient-wise tolerance. Because we have

$$\sigma_{n+m}(\mathrm{Syl}(g, h)) = 0.02137, \quad \|E\|_2 = 0.02000.$$

Next, we consider the absolute irreducibility of the following bivariate polynomial.

$$f(x_1, x_2) = x^3 - y^3 + 0.001.$$

We suppose that any non-presented coefficient is exactly zero and any presented coefficient can be perturbed within 0.00045 at most such that $\varepsilon_{3,0} = \varepsilon_{0,3} = \varepsilon_{0,0} = 0.00045$ and $\varepsilon_f = 0.00045\sqrt{3} \approx 0.0007794$. In this case, we can not determine its absolute irreducibility if we use the polynomial norm tolerance. Because we have the following separation bound $B(f)$ which is less than ε_f, so there are possibilities that its Ruppert matrix $R(f)$ becomes a rank deficient matrix.

$$B(f) = 0.0003586.$$

Moreover, we can not determine its absolute irreducibility even if we use the coefficient-wise tolerance without using its sparsity. Because we have

$$\sigma_{2nm+m-1}(R(f)) = 0.002121, \quad \|\sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}\varepsilon_{i,j}\|_2 = 0.002277.$$

This also means that there are possibilities that its Ruppert matrix $R(f)$ becomes a rank deficient matrix. However, we can say that it is absolutely irreducible if we use the coefficient-wise tolerance with using its sparsity. Because we have

$$\sigma_{\rho-1}(\mathcal{R}(f)) = 0.002121, \quad \|\sum_{i=0}^{n}\sum_{j=0}^{m} \mathcal{R}_{i,j}\varepsilon_{i,j}\|_2 = 0.001909.$$

This means that its sparse Ruppert matrix $\mathcal{R}(f)$ can not become any rank deficient matrix, so the given polynomial is absolutely irreducible within the given error bound.

7 Conclusion

It is absolutely clear that the coefficient-wise tolerance is better than the polynomial norm tolerance to operate with empirical polynomials. However, unfortunately, most of algorithms implemented in several computer algebra systems have not been implemented with both the tolerance systems together, especially not with the coefficient-wise tolerance. As shown in this paper, it is not difficult to extend the algorithms to the coefficient-wise tolerance, especially for some certain algorithms using coefficients directly. However, for other algorithms which use coefficients indirectly for example, extending to the coefficient-wise tolerance is not easy. The author thinks that all the researchers in symbolic-numeric algorithms for polynomials, should implement their algorithms using both tolerance mechanisms.

Moreover, there is a package for Mathematica, which is called "SNAP" [6], is made by the same author of this paper and it provides coprimality testing, absolute irreducibility testing and so on, using coefficient-wise tolerance also.

References

[1] Emiris, Ioannis Z., Galligo, André and Lombardi, Henri: Certified approximate univariate GCDs, *J. Pure Appl. Algebra*, **117/118**, 1997, 229–251.

[2] Gao, S. and Rodrigues, V. M.: Irreducibility of polynomials modulo p via newton polytopes, *J. Number Theory*, **101**, 2003, 32–47.

[3] Golub, G. H. and Loan, C. F. V.: *Matrix Computations Third Edition, Johns Hopkins Series in the Mathematical Sciences*, The Johns Hopkins University Press, Baltimore, 1996.

[4] Kako, F. and Sasaki, T.: Proposal of "effective floating-point number" for approximate algebraic computation, *preprint*, 1997.

[5] Kaltofen, E. and May, J.: On approximate irreducibility of polynomials in several variables, *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, 2003, 161–168.

[6] Nagasaka, K.: SNAP Package for Mathematica and Its Applications, *Proc. 9th Asian Technology Conference in Mathematics*, 2004, 308-316.

[7] Nagasaka, K.: Towards More Accurate Separation Bounds of Empirical Polynomials, *SIGSAM Bulletin, Formally Reviewed Articles*, **38**(4), 2004, 119–129.

[8] Nagasaka, K.: Towards More Accurate Separation Bounds of Empirical Polynomials II, *Proc. Computer Algebra in Scientific Computing 2005 (CASC2005), Lecture Notes in Computer Science, LNCS* **3718**, 318–329.

[9] Pan, V. Y.: Computation of approximate polynomial GCDs and an extension, *Inform. and Comput.*, **167**(2), 2001, 71–85.

[10] Ruppert, W. M.: Reducibility of polynomials $f(x, y)$ modulo p, *J. Number Theory*, **77**, 1999, 62–70.

[11] Sofroniou, M. and Spaletta. G.: Precise Numerical Computation, *J. Logic and Algebraic Programming*, **64**(1), 2005, 113–134.

[12] Terui, A. and Sasaki, T.: "approximate zero-points" of real univariate polynomial with large error terms, *IPSJ J.*, **41**, 2000, 974–989.

[13] Zeng, Z.: The approximate GCD of inexact polynomials. Part I: a univariate algorithm, manuscript, 2004.

[14] Zhi, Lihong: Displacement structure in computing approximate GCD of univariate polynomials, *Lecture Notes Ser. Comput.*, **10**, 2003, 288–298.