Approximate Factorization of Multivariate Polynomials via Differential Equations*

Shuhong Gao
Dept. of Mathematical Sciences
Clemson University
Clemson, South Carolina
29634-0975, USA
sgao@ces.clemson.edu

Erich Kaltofen, John May
Dept. of Mathematics
North Carolina State University
Raleigh, North Carolina
27695-8205, USA
{kaltofen,jpmay}@math.ncsu.edu

Zhengfeng Yang, Lihong Zhi
Key Lab of Mathematics
Mechanization,
AMSS, Beijing 10080 China
{zyang,lzhi}@mmrc.iss.ac.cn

ABSTRACT

The input to our algorithm is a multivariate polynomial, whose complex rational coefficients are considered imprecise with an unknown error that causes f to be irreducible over the complex numbers $\mathbb{C}$. We seek to perturb the coefficients by a small quantitity such that the resulting polynomial factors over $\mathbb{C}$. Ideally, one would like to minimize the perturbation in some selected distance measure, but no efficient algorithm for that is known. We give a numerical multivariate greatest common divisor algorithm and use it on a numerical variant of algorithms by W. M. Ruppert and S. Gao. Our numerical factorizer makes repeated use of singular value decompositions. We demonstrate on a significant body of experimental data that our algorithm is practical and can find factorizable polynomials within a distance that is about the same in relative magnitude as the input error, even when the relative error in the input is substantial (10^{-3}).

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algebraic Algorithms*; G.1.2 [**Mathematics of Computing**]: Numerical Analysis—*Approximation*

General Terms

algorithms, experimentation

Keywords

multivariate polynomial factorization, multivariate gcd, approximate factorization, approximate gcd, singular value decomposition, symbolic/numeric hybrid method

*Gao was supported in part by the National Science Foundation under Grant DMS-0302549, the National Security Agency under Grant MDA904-02-1-0067, and the Office of Naval Research under Grant N00014-00-1-0565 (DoD MURI). Kaltofen and May were supported in part by the National Science Foundation under Grants CCR-0113121 and CCR-0305314. Yang and Zhi were supported in part by a grant from the Chinese National Science Foundation.

ISSAC'04, July 4–7, 2004, Santander, Spain.

1. INTRODUCTION

We consider the problem of approximately factoring a polynomial $f(x, y) \in \mathbb{C}[x, y]$, where the actual coefficients of f are real or complex numbers. We do not assume that f is reducible over $\mathbb{C}$. Irreducibility of f is the case, for instance, if the coefficients of f are imprecise due to perturbations caused by physical measurements or by floating point computation. More generally, we do not assume that f is near a factorizable polynomial. By $\|f\|$ we denote the Euclidean length of the coefficient vector of f. By $f^{[\min]}$ we denote a factorizable polynomial over $\mathbb{C}$ with $\deg(f^{[\min]}) \le \deg(f)$ such that $\|f - f^{[\min]}\|$ is minimized, that is, $f^{[\min]}$ is a nearest reducible polynomial. We present new algorithms that can find a factorization $\widetilde{f} = f_1 \cdot f_2 \cdots f_r$ in $\mathbb{C}[x, y]$ with $\deg(\widetilde{f}) \le \deg(f)$ such that $\|\widetilde{f} - f^{[\min]}\|$ is small.

In [17, Example 2] it was discovered that $f^{[\min]}$ is dependent on the degree notion. Our bounds such as $\deg(\widetilde{f}) \le \deg(f)$ limit the degrees in the individual variables, that is $\deg_x(\tilde{f}) \le \deg_x(f)$ and $\deg_y(\tilde{f}) \le \deg_y(f)$ (rectangular polynomials). Our algorithms are based on the exact algorithms in [7]. All our methods are numerical and we execute our procedures with floating point scalars. We use the singular value decomposition (SVD) to determine the number of factors and approximate nullspace vectors in the arising Ruppert matrices. Furthermore, we have designed a new approximate bivariate polynomial GCD algorithm for the last step in Gao's approach. Our approximate GCD algorithm again makes use of the SVD on bivariate and univariate Sylvester matrices.

We present a substantial body of experimental evidence that our approach numerically computes approximate factorizations. The difficulty of a satisfying numerical analysis of any of our algorithms are the notions of "near" and "small". Our experiments show that our algorithms perform well even for polynomials with a relatively large irreducibility radius [22, 17, 23]. In section 2 we provide an initial analysis for the approximate GCD algorithm by proving that the approximate GCD converges to the exact GCD as the perturbation error goes to 0. However, our worst case error bounds, which must make use of bivariate factor coefficient bounds, seem unrealistically large.

There is an extensive literature on the problem of factoring multivariate polynomials over the real or complex num-

bers. In [14] one of the first polynomial-time algorithms is given for input polynomials with exact rational or algebraic number coefficients, and the problem of approximate factorization is already discussed there [14, section 6]. Approximate factorization algorithms suppose that the input coefficients are perturbed and the input polynomial consequently irreducible over $\mathbb{C}$ under exact interpretation of its coefficients. However, if the input polynomial is near its factorizable counterpart, say within machine floating point precision, one can attempt to run exact methods with floating point arithmetic, such as Hensel lifting or curve interpolation. The work reported in [28, 27, 6, 13, 26, 4, 2, 25, 30] studies recovery of approximate factorization from the numerical intermediate results. A somewhat related topic are algorithms that obtain the exact factorization of an exact input polynomial by use of floating point arithmetic in a practically efficient way [1].

A different line of methods bounds from below the distance from the input polynomial to the nearest factorizable polynomial, that is, the irreducibility radius [22, 17]. Not only do such bounds help in declaring inputs numerically irreducible, they also provide insight in the quality of a computed approximate factorization.

No polynomial-time algorithm is known for computing the nearest factorizable polynomial $f^{[\min]}$, which is open problem 1 in [15]. In [12] a polynomial-time algorithm is given for computing the nearest polynomial with a complex factor of constant degree. In practice, that algorithm is much slower than any of the numerical solutions—and the same may be expected of a future solution to the open problem—but for polynomials of degree 2 or 3 one can obtain an actual optimal answer, with which one can further gauge the output of the fast but non-optimal numerical procedures.

With the algorithms presented in this paper, we have successfully found approximate factorizations of all benchmark examples presented in the literature, including those with significant irreducibility radii (see section 3.4). The latter seems to distinguish our method from all previous ones. Our algorithms were designed to accomplish accurate factorizations rather than running at high speed, and we appear to meet or surpass the backward errors reported for all previous algorithms from the literature. In addition, our timings seem competitive, given that actual timings have been reported quite sparsely throughout the literature. Last not least, we present approximate factorizations for a list of new benchmark examples, thereby not only establishing the versatility of our method but also giving a set of new test problems to future algorithm designers.

2. SVD BASED APPROXIMATE MULTIVARIATE GCD

2.1 Algorithm Description

One of the main problems in converting Gao's factorization algorithm for use in the approximate case is the lack of a good multivariate GCD algorithm which will handle the case when the given polynomials are quite far from having a common factor.

We first make some general comments on GCDs of arbitrary multivariate polynomials. The following simple lemma is the key to our approximate GCD algorithm.

LEMMA 2.1. *Let $g, h \in \mathbb{C}[x_1, \ldots, x_n]$, both nonzero. Let $g_1 = g/\gcd(g,h)$ and $h_1 = h/\gcd(g,h)$. Then all the solutions $u, v \in \mathbb{C}[x_1, \ldots, x_n]$ to the equation*

$$ug + vh = 0 \tag{1}$$

must be of the form

$$u = h_1 q, \quad v = -g_1 q, \tag{2}$$

where $q \in \mathbb{C}[x_1, \ldots, x_n]$.

The proof of the lemma is trivial as $\mathbb{C}[x_1, \ldots, x_n]$ is a unique factorization domain. Note that the equation (1) is a linear system for the coefficients of u and v. To make it a finite system, we need to restrict the degrees of u and v. There are several ways to do this. For example, one can consider the individual degrees for each variable, or consider any weighted degree. We choose to consider the total degree (tdeg) and require that

$$\operatorname{tdeg}(u) \le \operatorname{tdeg}(h) - 1, \quad \operatorname{tdeg}(v) \le \operatorname{tdeg}(g) - 1. \tag{3}$$

Then $\gcd(g,h) = 1$ iff (1) and (3) have no nonzero solution for u and v.

In general, there is an explicit relation between the total degree of $\gcd(g,h)$ and the dimension of the solution space. To see this, note that the number of terms $x_1^{i_1} \cdots x_n^{i_n}$ with total degree $\le d$ is the binomial number

$$\beta(d,n) = \tbinom{d+n}{n}.$$

Hence u has $\beta(\operatorname{tdeg}(h)-1, n)$ coefficients and v has $\beta(\operatorname{tdeg}(g)-1, n)$ coefficients. Thus the number of variables for the linear system is

$$m = \beta(\operatorname{tdeg}(g) - 1, n) + \beta(\operatorname{tdeg}(h) - 1, n).$$

By (2) and (3), all the solutions for u and v are determined by $q \in \mathbb{C}[x_1, \ldots, x_n]$ with

$$\operatorname{tdeg}(q) \le \ell - 1,$$

where ℓ is the total degree of $\gcd(g,h)$. Hence the dimension of the solution space for u and v is exactly

$$\beta(\ell - 1, n).$$

Therefore one can compute the rank p of the coefficient matrix and then determine ℓ directly from $m - p = \beta(\ell - 1, n)$.

For univariate polynomials, the coefficient matrix for the linear system (1) and (3) is nothing but the well-known Sylvester matrix for g and h. In [3] the Sylvester matrix is used to get an approximate algorithm for the univariate GCD. For multivariate polynomials, we still call the coefficient matrix corresponding to (1) and (3) the Sylvester matrix for g and h. We shall apply it to multivariate GCDs. Note that the cofactors g_1 and h_1 appear as the solution with the smallest degree; they are the solution we are looking for.

We will denote the Sylvester matrix of g and h as S_1. To find the GCD we need to find a basis for the nullspace of S_1. In the case of exact arithmetic, the cofactors g_1 and h_1 can be found by performing Gaussian elimination on the nullspace basis. The GCD can then be found by division. Doing this numerically, we face two difficulties: First, S_1 may be full rank in exact arithmetic, and second, recovering the smallest degree polynomial from the nullspace using Gaussian elimination is not stable numerically.

To overcome the first difficulty, we use singular value decomposition to find the nearest matrix with (lower) rank p. The last $m-p$ singular vectors form a basis for the nullspace

of this nearest low rank matrix. To determine what p should be, we look at the singular values of S_1. Typically, when determining rank numerically, one would specify a tolerance ϵ and find a gap in the singular values:

$$\sigma_m \geq \cdots \geq \sigma_{m-p-1} > \epsilon \geq \sigma_{m-p} \geq \cdots \geq \sigma_1.$$

We do not wish to specify ϵ in advance, so we try to infer the "best" ϵ from the largest gap (i.e. the largest ratio σ_{i+1}/σ_i) in the singular values. In [5], it is shown that when given a tolerance ϵ it is possible to certify the degree of the approximate GCD using gaps in the sequence $\tau_i = \sigma_1(S_i(g,h))$ instead of the singular values of S_1. The size of the gap needed to certify the degree is very large however, and in practice the largest gap in the τ_is seems to given the same degree as the largest gap in the σ_is.

Another technique is to evaluate g and h at all of their variables but one, and find the "best" degree of the univariate GCD. This seems to work well when a small tolerance is given (in this case the tolerance could be inferred from the factorization problem), but does not work as well when inferring a tolerance, as above. The advantage of evaluating to find the degree is that the univariate Sylvester matrix is much smaller, so even if we have to compute SVDs for several evaluations, it will be much faster than computing the SVD of the multivariate Sylvester matrix.

The second difficulty above can be handled by removing rows and columns from S_1. Once we know what rank S_1 should be, we can take a submatrix of S_1 found by using stronger degree restrictions (on the unknown polynomials u and v) in the linear system $ug+vh = 0$ so that we have a new Sylvester matrix S_k which has a nullspace of dimension 1, where k is the degree of $\gcd(g,h)$. In this case, the single basis vector for the nullspace will give a constant multiple of the cofactors g_1 and h_1. This null vector can be computed numerically without computing the full SVD by using an iterative method. For our implementation we use the method given in [19]. Once we have our approximations of g_1 and h_1 we can compute an approximate GCD by doing least squares approximate division.

A very similar multivariate approximate GCD algorithm was proposed in [33] but a tolerance ϵ is required there, and an additional Gauss-Newton iteration step is used to improve the GCD further.

AMVGCD: Approximate Multivariate GCD.
Input: g and h in $\mathbb{C}[x_1, \ldots, x_n]$.
Output: d, a non-constant approximate GCD of g and h.

1. Determine k, the degree of the approximate GCD of g and h in one of two ways below:
   (a) Form $S = \mathrm{S}_1(g,h)$, the matrix of the linear system $ug+vh = 0$, where $g, h \in \mathbb{C}[x,y]$ with $\mathrm{tdeg}(u) < \mathrm{tdeg}(h)$ and $\mathrm{tdeg}(v) < \mathrm{tdeg}(g)$. Finding the largest gap in the singular values of S and inferring the degree from the numerical rank of S.
   (b) Computing the degrees of the GCDs of several random univariate projections of g and h by looking for the numerical rank of the corresponding univariate Sylvester matrices.
2. Reform S as $\mathrm{S}_k(g,h)$ that is, use $\mathrm{tdeg}(u) = \mathrm{tdeg}(h) - k$ and $\mathrm{tdeg}(v) = \mathrm{tdeg}(g) - k$ as the constraints on u and v in the linear system in the first step. This new S will have a dimension 1 nullspace.
3. Compute a basis for the nullspace of S by computing the singular vector corresponding the smallest singular value of S. This vector gives a solution $[u,v]^T$.
4. Find a d that minimizes $\|h - d\,u\|_2^2$; alternately minimize $\|h - d\,u\|_2^2 + \|g + d\,v\|_2^2$, both using least squares.

If one wishes to specify a tolerance, then only the first step is affected. It is possible that in this step we could find that $k = 0$, in which case the method would return $d = 1$, declaring g and h to be approximately relatively prime to the given tolerance.

2.2 Convergence of the Algorithm

In this section we restrict to bivariate case ($n = 2$) for ease of notation. Let us start with $\tilde{g}$ and $\tilde{h}$ relatively prime and normalized so that $\|\tilde{h}\|_2 = \|\tilde{g}\|_2 = 1$. Suppose that $\gcd(g,h) = d \neq 1$, $\mathrm{tdeg}(g) = \mathrm{tdeg}(\tilde{g})$, $\mathrm{tdeg}(h) = \mathrm{tdeg}(\tilde{h})$, $\|g - \tilde{g}\|_2 = \epsilon_1$, and $\|h - \tilde{h}\|_2 = \epsilon_2$. We will show that as $\epsilon_1, \epsilon_2 \to 0$ that the computed approximate GCD for $\tilde{g}$ and $\tilde{h}$ converges to d.

Let $S = \mathrm{S}_k(g,h)$ where k is chosen so that S has rank p, and rank deficiency 1. Then $w = [u,v]$ is a basis for the nullspace of S, where $u = h/d$, and $v = -g/d$, and without loss of generality we can assume that u is unit length. Let $\tilde{S} = \mathrm{S}_k(\tilde{g},\tilde{h})$. We can bound the distance between these two Sylvester matrices:

$$\|S - \tilde{S}\|_2^2 \leq \|S - \tilde{S}\|_F^2 = a_1\epsilon_1^2 + a_2\epsilon_2^2 = \epsilon_3^2$$

where

$$a_1 = \beta(\mathrm{tdeg}(h) - k, 2) \text{ and } a_2 = \beta(\mathrm{tdeg}(g) - k, 2)$$

depend only on k and the degrees of g and h.

We can use the SVD to find M so that:

$$\min_{\mathrm{Rank}(M)=\mathrm{Rank}(S)} \|\tilde{S} - M\|_2 = \sigma_{m-p}(\tilde{S}) \leq \epsilon_3.$$

Note that M is not a Sylvester matrix and $\|M - S\|_2 \leq 2\epsilon_3$. Now, M has a dimension 1 nullspace, so let $\widetilde{w} = [\tilde{u}, \tilde{v}]$ be the vector which spans the nullspace of M with $\|\tilde{u}\|_2 = 1$. Theorem 6.4 in [31] (reformulated for our purpose in [11, section 8]) bounds the distance between w and $\widetilde{w}$ in terms of ϵ_3 so that as $\epsilon_3 \to 0$, $\epsilon_4 = \|w - \widetilde{w}\|_2 \to 0$. Thus for sufficiently small ϵ_1 and ϵ_2 we have $\mathrm{tdeg}(\tilde{u}) = \mathrm{tdeg}(u)$.

In the following we will make repeated use of the multivariate factor coefficient bound found in [8, pages 134-139]: $\|f_1\|_2\|f_2\|_2 \leq 2^{\sum_i(\deg_{x_i} f)}\|f\|_2$, where $f = f_1 f_2$.

Now, using least squares division, we compute $\tilde{d}$ as the polynomial that minimizes

$$\epsilon_5 = \min_{\bar{d}:\ \mathrm{tdeg}(\bar{d}) \leq \mathrm{tdeg}(d)} \|\bar{d}\,\tilde{u} - \tilde{h}\|_2.$$

We can bound ϵ_5:

$$\begin{aligned}
\epsilon_5 = \|\tilde{d}\,\tilde{u} - \tilde{h}\|_2 &\leq \|d(\tilde{u} - u) - (\tilde{h} - h)\|_2 \\
&\leq \|d(\tilde{u} - u)\|_1 + \epsilon_2 \leq \|d\|_1\|\tilde{u} - u\|_1 + \epsilon_2 \\
&\leq \tbinom{k+2}{2}^{1/2}\|d\|_2\tbinom{\mathrm{tdeg}(h)-k+2}{2}^{1/2}\|\tilde{u} - u\|_2 + \epsilon_2 \\
&\leq a_4\,\epsilon_4 + \epsilon_2
\end{aligned}$$

where

$$a_4 = 2^{\deg_x(h)+\deg_y(h)}\,\|h\|_2\tbinom{k+2}{2}^{1/2}\tbinom{\mathrm{tdeg}(h)-k+2}{2}^{1/2}$$

via the multivariate factor coefficient bound. Now,

$$\begin{aligned}\|(d-\tilde{d})u\|_2 &= \|h-\tilde{d}\,\tilde{u}-\tilde{d}(u-\tilde{u})\|_2\\ &\le \|h-\tilde{h}+(\tilde{h}-\tilde{d}\,\tilde{u})\|_2+\|\tilde{d}(u-\tilde{u})\|_2\\ &\le \|h-\tilde{h}\|_2+\|\tilde{h}-\tilde{d}\,\tilde{u}\|_2+\|\tilde{d}(u-\tilde{u})\|_1\\ &\le \epsilon_2+\epsilon_5+\|\tilde{d}\|_1\|u-\tilde{u}\|_1\\ &\le 2\epsilon_2+a_4\epsilon_4+\tbinom{k+2}{2}^{1/2}\|\tilde{d}\|_2\tbinom{\mathrm{tdeg}(h)-k+2}{2}^{1/2}\epsilon_4,\end{aligned}$$

where $\|\tilde{d}\|_2$ is bounded as follows:

$$\begin{aligned}\|\tilde{d}\|_2 &\le 2^{\deg_x(h)+\deg_y(h)}\|\tilde{d}\,\tilde{u}\|_2\\ &\le 2^{\deg_x(h)+\deg_y(h)}(\|\tilde{h}\|_2+\|\tilde{d}\,\tilde{u}-\tilde{h}\|_2)\\ &\le 2^{\deg_x(h)+\deg_y(h)}(\|\tilde{h}\|_2+\epsilon_2+a_4\epsilon_4).\end{aligned}$$

Thus, using the multivariate factor coefficient bound again, we have that

$$\|d-\tilde{d}\|_2 \le 2^{\deg_x(h)+\deg_y(h)}(2\epsilon_2+a_5\epsilon_4)$$

with a bound a_5 derived from the previous two estimates. So, as $\epsilon_1,\epsilon_2\to 0$, $\|d-\tilde{d}\|_2\to 0$ since $\epsilon_4\to 0$, and a_5 is bounded in terms of the degrees and norms of $\tilde{g}$ and $\tilde{h}$. It should be noted that in practice, the $\tilde{d}$ computed is much closer than the given bound suggests it might be.

3. THE FACTORIZATION ALGORITHM AND EXPERIMENTS

In this section, we propose a numerical algorithm for factoring approximate bivariate polynomials over $\mathbb{C}$. The algorithm relies on the singular value decomposition of the Ruppert matrix and approximate GCDs of bivariate polynomials. We have implemented the algorithm in Maple 9. A set of examples is tested.

3.1 Ruppert Matrix

We consider the problem of the factorization of a bivariate polynomial f in $\mathbb{C}[x,y]$. The coefficients of f may be known to only a fixed precision. Our algorithms are based on the exact algorithms in [7]. However, our procedures are designed to take as input polynomials with imprecise floating point coefficients. We use the numerical singular value decomposition to determine the number of approximate factors and approximate nullspace vectors in the arising Ruppert matrices. The approximate factors can then be obtained by computing approximate GCDs of f and polynomials formed by the null vectors of the Ruppert matrix.

We assume that f is non-constant and $\gcd(f,f_x)=1$ where $f_x=\partial f/\partial x$, which makes f both squarefree and with no factor in $\mathbb{C}[y]$. Suppose that f factors as

$$f=f_1f_2\cdots f_r, \tag{4}$$

where $f_i\in\mathbb{C}[x,y]$ are distinct and irreducible over $\mathbb{C}$. Define

$$E_i=\frac{f}{f_i}\frac{\partial f_i}{\partial x}\in\mathbb{C}[x,y],\ 1\le i\le r. \tag{5}$$

Then

$$f_x=E_1+E_2+\cdots+E_r \text{ and } E_iE_j\equiv 0 \mod f \text{ for all } i\ne j.$$

THEOREM 3.1. *[24] Suppose $f\in\mathbb{C}[x,y]$ with bi-degree (m,n), i.e., $\deg_x f=m$, $\deg_y f=n$. Then f is absolutely irreducible if and only if the equation*

$$\frac{\partial}{\partial y}\left(\frac{g}{f}\right)=\frac{\partial}{\partial x}\left(\frac{h}{f}\right), \tag{6}$$

has no nonzero solution $g,h\in\mathbb{C}[x,y]$ with $\deg g\le(m-1,n),\deg h\le(m,n-2)$.

Since differentiation is linear over $\mathbb{C}$, the equation (6) gives a linear system for the coefficients of g and h, whose coefficient matrix we call the Ruppert matrix $R(f)$. The matrix $R(f)$ of f is full rank if and only if f is absolutely irreducible. Using Ruppert's criterion, [17] provides some separation bounds for testing whether a numerical polynomial is absolutely irreducible, given a certain tolerance on its coefficients. When these bounds are small, one may suspect the polynomial f to be near a reducible polynomial. In the following, we start with an explanation of some results in [7] for the factorization of polynomials with exact coefficients.

First, let us note that in [7] the degree conditions on g and h are changed to:

$$\deg g\le(m-1,n),\deg h\le(m,n-1), \tag{7}$$

which allows for the solution $(g,h)=(f_x,f_y)$ even when f is irreducible. Again, $R(f)$ denotes the coefficient matrix.

THEOREM 3.2. *[7] Let $f\in\mathbb{C}[x,y]$ be a non-constant polynomial of bi-degree (m,n) with $\gcd(f,f_x)=1$. Define*

$$G=\{g\in\mathbb{C}[x,y]:(6)\text{ and }(7)\text{ hold for some }h\in\mathbb{C}[x,y]\} \tag{8}$$

Suppose f has the factorization into irreducible polynomials as in (4). Then G is a vector space over $\mathbb{C}$ of dimension r and each $g\in G$ is of the form $g=\sum\lambda_iE_i$ where $\lambda_i\in\mathbb{C}$.

THEOREM 3.3. *[7] Suppose that $g_1,\ldots,g_r$ form a basis for G over $\mathbb{C}$. Select $s_i\in S\subset\mathbb{C}$ uniform randomly and independently for all $1\le i\le r$, and let $g=\sum_{i=1}^r s_ig_i$. There is a unique $r\times r$ matrix $A=[a_{i,j}]$ over $\mathbb{C}$ such that*

$$gg_i\equiv\sum_{j=1}^r a_{i,j}g_jf_x \pmod f \quad \text{in } \mathbb{C}(y)[x]. \tag{9}$$

Furthermore, let $E_g(x)=\det(Ix-A)$, the characteristic polynomial of A. Then the probability that

$$f=\prod_{\lambda\in\mathbb{C}:\ E_g(\lambda)=0}\gcd(f,g-\lambda f_x) \tag{10}$$

gives a complete factorization of f over $\mathbb{C}$ is at least $1-r(r-1)/(2|S|)$.

When we extend Gao's exact algorithm to the factorization of the polynomials with approximate coefficients, we have to consider four main problems.

1. Reduce the polynomial f so that $\gcd(f,f_x)=1$ approximately.
2. Determine the numerical dimension of G.
3. Compute an E_g that has no cluster of roots.
4. Compute the approximate GCDs of bivariate polynomials: $\gcd(f,g-\lambda_if_x)$.

The previous section provides us a robust algorithm to compute the approximate GCDs of bivariate polynomials.

We can use the algorithm AMVGCD to compute the approximate GCD of f and f_x. By computing $f/\gcd(f, f_x)$, we may reduce f to the case where $\gcd(f, f_x) = 1$ approximately.

Now let us look at the second and third problems. Similarly to the discussion in section 2, we can determine the numerical dimension of G by the singular value decomposition of the Ruppert matrix $R(f)$. If a tolerance ϵ is given, then the numerical dimension of G is the r such that

$$\cdots \geq \sigma_{r+2} \geq \sigma_{r+1} > \epsilon \geq \sigma_r \geq \cdots \geq \sigma_1.$$

However, if we do not know the relative error in the coefficients of f, it is difficult to provide a tolerance ϵ that is consistent with the error in the data. In that case we may look for the biggest gap in the singular values. Once we have determined that the numerical dimension of G is r, then the singular value σ_r tells us something about how far f is to a polynomial $\tilde{f}$ that has r absolutely irreducible factors:

$$\min_{\substack{\deg \tilde{f}=(m,n) \\ \dim \text{Nullspace}(R(\tilde{f}))=r}} \|R(f) - R(\tilde{f})\|_2 \geq \sigma_r.$$

Thus σ_r can be used as a tolerance when we estimate the degree of bivariate approximate GCD by projecting to univariate GCD problems.

Suppose we have obtained the numerical dimension r of G. The coefficients of the approximate basis $g_1, \ldots, g_r$ of G can be determined from the singular vectors corresponding to the last r small singular values $v_1, \ldots, v_r$. It is easy to see that $\|R(f)v_i\|_2 \leq \sigma_i \leq \sigma_r$. So the g_i's form an approximate basis for G with tolerance σ_r. We choose $s_1, \ldots, s_r \in S \subset \mathbb{C}$ uniform randomly and let $g = \sum_{i=1}^{r} s_i g_i$. As pointed out in [7, pg. 818, Comment 2], we can substitute an arbitrary value of $\alpha \in \mathbb{C}$ for y with the property that $f(x, \alpha)$ remains squarefree. The matrix A can be formed in the following two steps: First, we reduce the polynomial gg_i and $g_j f_x$ with respect to f at $y = \alpha$ for $1 \leq i, j \leq r$ by using approximate division of univariate polynomials [34]; then we solve the least squares problem:

$$\min \|\text{rem}(gg_i - (a_{i,1}g_1 f_x + \cdots + a_{i,r}g_r f_x),\ f)\|_2$$

at $y = \alpha$ to find the value of unknown elements $a_{i,j}$. Let $E_g(x) = \det(Ix - A)$, the characteristic polynomial of A. We compute all the numerical roots $\lambda_1, \ldots, \lambda_r$ of the univariate polynomial E_g over $\mathbb{C}$, and check the smallest distance between these roots:

$$\text{distance} := \min\{|\lambda_i - \lambda_j|,\ \ 1 \leq i < j \leq r.\}$$

If the distance is small, i.e., E_g has a cluster of roots, we should choose another set of s_i's and try to find a separable E_g.

In [7], the absolutely irreducible factors are obtained by computing the GCDs over algebraic extension fields formed by the irreducible factors of E_g. Since we are working over $\mathbb{C}$, all the roots of E_g are already in $\mathbb{C}$, hence there is no need to deal with field extension. So we compute the bivariate approximate GCDs $\tilde{f}_i = \gcd(f, g - \lambda_i f_x)$ according to the method in Section 2 for each numerical root λ_i of E_g. Suppose we obtain a proper approximate factorization of f over $\mathbb{C}$: $f \approx \prod_{i=1}^{r} \tilde{f}_i$. We can check the backward error of the approximate factorization: $\min_{c \in \mathbb{C}} \|f - c\prod_{i=1}^{r} \tilde{f}_i\|_2 / \|f\|_2$. If the backward error is reasonable small compared with the relative error in the coefficients of f, we have found a valid factorization of f. Otherwise, we can form a minimization problem as in [13], and try to improve the approximate factors.

3.2 Algorithm

AFBP: Approximate Factoring Bivariate Polynomials.
Input: A polynomial $f \in \mathbb{C}[x, y]$ such that f and f_x are approximately relatively prime, that is, f is approximately squarefree and has no approximate factors in $\mathbb{C}[y]$ (see section 3.3 below). Let $\deg_x(f) = n > 1$, $\deg_y(f) = m > 1$ and let S be a finite set $S \subset \mathbb{C}$ with $|S| \geq mn$.
Output: A list of approximate factors of f.

Step 1. [Ruppert matrix]

- 1.1 Form the Ruppert matrix from the linear equations (6) and (7).
- 1.2 Compute the singular value decomposition of the Ruppert matrix, and find the last $\text{tdeg}(f)+1$ singular values σ_i.
- 1.3 Find the biggest gap in these singular values
$$\max_i\{\sigma_{i+1}/\sigma_i\}$$
and decide the numerical dimension r of the Ruppert matrix, suppose $r \geq 2$.
- 1.4 Form a basis $g_1, \ldots, g_r$ from the last r right singular vectors $v_1, \ldots, v_r$.

Step 2. [Separable E_g]

- 2.1 Pick $s_i \in S$, uniform randomly and independently, and set $g := \sum_{i=1}^{r} s_i g_i$.
- 2.2 Select randomly a proper value for variable $y = \alpha$ which does not change the degree or the squarefree property of f.
- 2.3 For $y = \alpha$, compute $a_{i,j}$ that minimize the norm of the univariate remainder:
$$\min \|\text{rem}(gg_i - \sum_{j=1}^{r} a_{i,j} g_j f_x,\ f)\|_2.$$
- 2.4 Let $E_g(x) = \det(Ix - A)$, where $A = [a_{i,j}]$. We compute the numerical roots $\lambda_i, 1 \leq i \leq r$ of E_g (the numerical eigenvalues of A) and distance := $\min_{1 \leq i < j \leq r}\{|\lambda_i - \lambda_j|\}$.
- 2.5 If the distance is small then go to Step 2.1.

Step 3. [Approximate GCD] Compute $f_i = \gcd(f, g - \lambda_i f_x)$ over $\mathbb{C}[x, y]$ for $1 \leq i \leq r$.

Step 4. [Backward error and correction]

- 4.1 Compute $\min_{c \in \mathbb{C}} \|f - c\prod_{i=1}^{r} f_i\|_2 / \|f\|_2$.
- 4.2 Improve the approximate factors if necessary.

Step 5. Output the approximate factors $f_1, \ldots, f_r$ and c.

Remark 3.4. Algorithm AFBP has been generalized to polynomials in $n > 2$ variables. Instead of a single PDE as in (6), one has $n - 1$ such equations:

$$\frac{\partial}{\partial x_i}\frac{g_1}{f} - \frac{\partial}{\partial x_1}\frac{g_i}{f} = 0, \forall\, 2 \leq i \leq n$$
$$\deg g_i \leq \deg f, \quad \deg_{x_i} g_i \leq \deg_{x_i} f - 1, \forall\, 1 \leq i \leq n;$$

see the journal version of [17].

REMARK 3.5. Our output polynomials are not certified to be approximately irreducible in the sense that their radii of irreducibility are large. The reason for this is that our algorithm can be run on inputs that do not lie near factorizable polynomials, and the approximate GCDs in Step 3 may place a factor near a reducible polynomial. However, one may always achieve approximate irreducibility certification by applying the algorithms in [17] to the produced factors.

3.3 Multiple Factors

In the case that f is quite close to a polynomial that is not squarefree, our factorization algorithm does not work well.

A similar but lesser problem is the removal of approximate factors in $\mathbb{C}[y]$, which essentially amounts to a univariate approximate GCD computation. Our code performs such content removal.

One method to deal with the non-squarefree case is to compute f_{sqfr}, the approximate quotient of f and the approximate GCD of f and f_x. Then compute the distinct approximate factors of $f_{\text{sqfr}} \approx f_1 \cdots f_r$ using our algorithm. Finally, determine powers for each factor by looking for gaps in the sequence $\alpha_{i,j} = \sigma_1(S_1(f_i, \partial_{x,j} f))$.

We can only definitively call f approximately squarefree if all of the nearest polynomials that factor are squarefree. We cannot compute the nearest polynomial that factors, but we can bound the distance to the nearest polynomial that factors using the singular values of $R(f)$ [17], and similarly bound the distance to the nearest polynomial that is not squarefree using the singular values of $S_1(f, f_x)$. If the two bounds are very close we have to compute the factorization both ways and use the one with smaller backwards error.

In [33] a different method is proposed, which is based entirely on multivariate approximate GCDs and which generalizes the univariate algorithm in [32]. We experimentally compare our approach to that one in the next subsection.

3.4 Experiments

EXAMPLE 3.6. [22] We illustrate our algorithm by factoring the following polynomial:

$$f := (x^2 + yx + 2y - 1)(x^3 + y^2 x - y + 7) + 0.2x.$$

Since $\deg f = (5, 3)$, the Ruppert matrix is a 37×26 matrix. The last several singular values of the matrix are:

$$\cdots, 0.0475177, 0.0180165, 0.00168900, 0.203908 \cdot 10^{-10}.$$

Starting from the second smallest singular value, the biggest gap is $0.0180165/0.00168900 = 10.6669$. So $r = 2$ and f is supposed to be close to a polynomial having two irreducible factors. A basis for G computed from the last two right singular vectors is:

$$\begin{aligned} g_1 &= -0.222225\,x^4 - 0.249186\,yx^3 + 0.157991\,y + 0.0120672 \\ &\quad - 0.541602\,yx^2 + 0.265256\,x^2 + 0..341504\,x - 0.179952\,y^3 \\ &\quad - 0.0515876\,yx - 0.00117838\,y^2x - 0.0670575\,y^3x \\ &\quad - 0.0402726\,y^2x^2 - 0.00669638\,x^3 + 0.0653902\,y^2, \\ g_2 &= 0.186723\,x^4 + 0.149379\,yx^3 - 0.112034\,x^2 + 0.261412\,y \\ &\quad - 0.0746893\,yx + 0.522825\,x + 0.0746893\,y^3 + 0.00746893 \\ &\quad - 0.0746893\,y^2 + 0.112034\,y^2x^2 + 0.0746893\,y^3x \\ &\quad + 0.224068\,yx^2 - 0.146533 \cdot 10^{-9}\,x^3. \end{aligned}$$

Take a random linear combination of $g = \frac{49}{50} g_1 + \frac{29}{40} g_2$ and set $y = 1$. $A = [a_{i,j}]$ can be computed as

$$\begin{bmatrix} -0.757129 & 1.166496 \\ 0.726481 & 0.537448 \end{bmatrix}.$$

Two eigenvalues of the matrix A are $\lambda_1 = -1.23519$, $\lambda_2 = 1.01551$. Computing $f_i = \gcd(f, g - \lambda_i f_x)$, $i = 1, 2$, we obtain two factors of f:

$$\begin{aligned} f_1 &= -0.227274\,x^3 + 0.000400937\,yx^2 + 0.224906\,y - 1.57196 \\ &\quad - 0.226899\,y^2x - 0.00914627\,yx + 0.000416589\,y^3 \\ &\quad + 0.0100343\,x^2 + 0.00747927\,x + 0.00700437\,y^2, \\ f_2 &= -0.442980\,x^2 - 0.0121738\,x - 0.447473\,yx - 0.887806\,y \\ &\quad - 0.000598981\,y^2 + 0.444049. \end{aligned}$$

We optimize the scalar multiple for computing the backward error: $\|f - 10.03713\,f_1 f_2\|_2 / \|f\|_2 = 0.00950792$. The factorization can be corrected once to

$$\begin{aligned} f_1 &= 0.226888\,x^3 + 0.00192287\,yx^2 + 0.221635\,y - 1.57177 \\ &\quad - 0.225259\,y^2x + 0..00122024\,yx + 0.0000988727\,y^3 \\ &\quad + 0.00464850\,x^2 - 0.00304222\,x + 0.000834169\,y^2, \\ f_2 &= -0.442403\,x^2 - 0.0134080\,x - 0.443501\,yx - 0.888244\,y \\ &\quad + 0.00163781\,y^2 + 0.443727. \end{aligned}$$

and the backw. err. $\|f - 10.03706\,f_1 f_2\|_2 / \|f\|_2 = 0.00102532$, which is a little bigger than the backward error 0.000753084 of the factorization given in [22]. ⊠

Ex.	$\deg(f_i)$	$\frac{\sigma_{r+1}}{\sigma_r}$	$\frac{\sigma_r}{\|R(f)\|_2}$	*coeff. error*	*backward error*	*time(sec)*
1	2,3	11	10^{-3}	10^{-2}	1.08e–2	14.631
2	5,5	10^{9}	10^{-10}	10^{-13}	8.30e–10	5.258
3	10,10	10^{5}	10^{-6}	10^{-7}	1.05e–6	85.96
4	7,8	10^{7}	10^{-8}	10^{-9}	1.41e–8	19.628
5	3,3,3	10^{8}	10^{-10}	0	1.29e–9	9.234
6	6,6,10	10^{3}	10^{-6}	10^{-5}	2.47e–4	539.67
7	9,7	486	10^{-4}	10^{-4}	2.14e–4	43.823
8	4,4,4,4,4	273	10^{-6}	10^{-5}	1.31e–3	3098
9	3,3,3	1.70	10^{-3}	10^{-1}	7.93e–1	29.25
10	12,7,5	658	10^{-6}	10^{-5}	1.56e–4	968
11	12,7,5	834	10^{-6}	10^{-5}	3.19e–4	1560
12	12,7,5	8.34	10^{-4}	10^{-3}	8.42e–3	4370
13	5,$(5)^2$	10^{3}	10^{-5}	10^{-5}	6.98e–5	34.28
14	$(5)^3$,3,$(2)^4$	10^{7}	10^{-9}	10^{-10}	2.09e–7	73.52
15	5,5	10^{4}	10^{-5}	10^{-5}	1.72e–5	332.99
15a	2,2	10^{9}	10^{-10}	10^{-4}	1.02e–9	13.009
16	18,18	10^{4}	10^{-7}	10^{-6}	3.75e–6	3173
17	18,18	10^{4}	10^{-7}	10^{-6}	4.10e–6	4266
18	6,6	10^{6}	10^{-8}	10^{-7}	2.97e–7	30.034

Table 1: Algorithm performance on benchmarks

We have implemented the AFBP algorithm and its variants in Maple. In Table 1, we show the performance for some well known or randomly generated examples on Pentium 4 at 2.0 Ghz for *Digits* = 10 in Maple 8 and 9 under Windows and Linux. Here σ_r and σ_{r+1} are singular values around the biggest gap—the given values are orders of magnitude; *coeff. error* indicates the noise imposed on the input,

namely the relative 2-norm coefficient error to the original product of polynomials. The *time* is that for the entire factorization in seconds of a single run; the timings can vary significantly (up-to a factor of 4) with the randomization. Example 1 in Table 1 is from [22] and has already been explained above. Examples 2 and 3 are from [26]; Sasaki's algorithm takes 430ms and 2080ms on a SPARC 5 (CPU: microSPARC II, 70 MHz) and produced backward errors of 10^{-9} and 10^{-5}, respectively. Example 4 in Table 1 is from [4]; the backward error for their approximate factorization is reported as 0.47×10^{-4}, compared to ours of backward error 0.14×10^{-7}. Example 5 in Table 1 is from [2], which is the factorization of an exact polynomial of degree 9; here their and our backward errors are about the same. No timings were reported in the cited papers for Examples 1, 4 and 5.

Example 6 to 13 and 15 to 17 are constructed by choosing factors with random integer coefficients in the range $-5 \leq c \leq 5$ and then adding a perturbation. For noise we choose a relative tolerance 10^{-e}, then randomly choose a polynomial that has the same degree as the product, 25% as many terms (5% for Example 10 and 99% for Example 17) and coefficients in $[-10^e, 10^e]$. Finally, we scale the perturbation so that the relative error is 10^{-e}. Examples 10, 11 and 12 approximately factorize the same polynomial with perturbations of different noise level and sparseness.

Examples 16 and 17 have been speeded significantly by changing how our implementation obtains the degree of the GCD in Step 1 of Algorithm AMVGCD of section 2.1. The use of σ_r as the tolerance for the approximate GCD is not accurate due to the large norm of the projected univariate polynomials, and must be increased to avoid the time-consuming correction loop for obtaining suitable GCDs.

Example 13 and 14 have repeated factors denoted with exponents in the degrees column. Example 14 is Zeng's ASFF example in [33]. The forward errors of the factors we compute are about 10^{-8}, similar to Zeng's forward error. Forward error means the relative 2-norm coefficient vector distance of a computed approximate factor to the nearest originally chosen factor, before noise is added to the product. For our other examples our algorithm produces forward errors that are in magnitude those of the stated backward errors, with the exception of Example 9. Here the degrees of the produced approximate factors are 4 and 5, hence the forward error is ∞. In fact, the approximate factorization is poor because better backward error can obtained simply by setting terms to 0. Polynomials with 10% relative noise are not handled well by our algorithms.

Example 15 and 15a are polynomials in three variables; 15a is from [15]. Our algorithm employed the method described in Remark 3.4. Example 18 is a polynomial with complex coefficients, where the real and imaginary parts of the coefficients of the factors were chosen random integers in $[-5, 5]$. We added noise to the real and imaginary parts of all terms. The journal version of this paper will contain a full suite of benchmarks that are complex polynomials.

We have successfully found the approximate factors of the four examples that Jan Verschelde has provided us, which arise in the engineering of Stewart-Gough platforms (see [30]). The input polynomials in 2 and 3 variables of degree 12 have small absolute coefficient error, 10^{-16}, and have approximate factors of multiplicities 1, 3 and 5.. We computed the tri-variate approximate factors via sparse numerical interpolation [10], which is possible here because the forward error in the approximate factor coefficients is small.. We carried out all computations in double precision. Our running times, no more than 200 seconds with a backward error of no more than $7.62 \cdot 10^{-9}$, appear much faster than what [30] report for their solution, but we had at our disposal the code for [10].

Our 23 test cases and Maple implementation are available from `http://www.mmrc.iss.ac.cn/~lzhi/Research/appfac.html` and `http://www.math.ncsu.edu/~kaltofen/` (click on the "Software" link).

4. CONCLUDING REMARKS

The singular value decomposition was introduced to the area of hybrid symbolic/numeric algorithms in [3]. Since then, the SVD approach has been successfully applied to a variety of problems [5, 9], most recently to the univariate squarefree factorization problem [32]. The past experience indicates that straightforward application of the SVD as a singular linear system solver may not yield useful results, and that the algorithms often require modification. Such is the case for our approximate GCD and factorization algorithm. Often in the past literature, the hybrid algorithms are experimentally shown to work in lieu of a theoretical backward analysis. We present an analysis for our multivariate GCD algorithm. However, such worst case analysis gives overly pessimistic error bounds, and becomes even more difficult when the algorithms are nested, as is the case for our factorization method. The worst case analysis together with our many successful experiments thus make our methods "good heuristics" in the sense of [16].

Clearly, our work is not finished. Now that we have a universal bivariate approximate factorization algorithm, meaning that closeness to a reducible polynomial is no longer a necessary requirement, the case of three and more variables becomes easier by way of the Bertini/Hilbert irreducibility theorems. One may interpolate the remaining variables, that either with a dense or the new sparse numerical algorithms [10], together with the homotopy of [18] when needed. We have reported success with this approach in section 3.4 for the case that the irreducibility radius is small (10^{-16}). Since the generalized Sylvester and Ruppert matrices have a fast matrix times vector product, a black box iterative method for computing the singular values and vectors is possible (cf. [33]), thus allowing direct application of our method (see Remark 3.4 and Example 15 in section 3.4) for three or more variables on larger examples. In order to approximately factor substantially perturbed polynomials with 25, say, variables into sparse factors, the sparse interpolation approach requires further study.

Acknowledgements: We thank two anonymous referees for their comments, Zhonggang Zeng for sending us [33], Jan Verschelde for files of the examples in [30] and Wen-shin Lee for help with sparse interpolation and the Maple code to [10].

5. REFERENCES

[1] Chèze, G., and Galligo, A. From an approximate to an exact absolute polynomial factorization. Paper submitted, 22 pages, 2003.

[2] Corless, R. M., Galligo, A., Kotsireas, I. S., and Watt, S. M. A geometric-numeric algorithm for absolute factorization of multivariate polynomials. In Mora [20], pp. 37–45.

[3] Corless, R. M., Gianni, P. M., Trager, B. M., and Watt, S. M. The singular value decomposition for polynomial systems. In *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95* (New York, N. Y., 1995), A. H. M. Levelt, Ed., ACM Press, pp. 96–103.

[4] Corless, R. M., Giesbrecht, M. W., van Hoeij, M., Kotsireas, I. S., and Watt, S. M. Towards factoring bivariate approximate polynomials. In Mourrain [21], pp. 85–92.

[5] Emiris, I. Z., Galligo, A., and Lombardi, H. Certified approximate univariate GCDs. *J. Pure Applied Algebra 117 & 118* (May 1997), 229–251. Special Issue on Algorithms for Algebra.

[6] Galligo, A., and Watt, S. A numerical absolute primality test for bivariate polynomials. In *ISSAC 97 Proc. 1997 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 1997), W. Küchlin, Ed., ACM Press, pp. 217–224.

[7] Gao, S. Factoring multivariate polynomials via partial differential equations. *Math. Comput. 72*, 242 (2003), 801–822.

[8] Gelfond, A. O. *Transcendental and algebraic numbers. Translated from the 1st Russian ed. by Leo F. Boron*. Dover, New York, 1960.

[9] Gianni, P., Seppälä, M., Silhol, R., and Trager, B. Riemann surfaces, plane algebraic curves and their period matrices. *J. Symbolic Comput. 26*, 6 (1998), 789–803.

[10] Giesbrecht, M., Labahn, G., and Lee, W.-s. Symbolic-numeric sparse interpolation of multivariate polynomials (extended abstract). In *Proc. Ninth Rhine Workshop on Computer Algebra (RWCA'04), University of Nijmegen, the Netherlands* (2004), pp. 127–139. Full paper under preparation.

[11] Golub, G. H., and Van Loan, C. F. *Matrix Computations*, third ed. Johns Hopkins University Press, Baltimore, Maryland, 1996.

[12] Hitz, M. A., Kaltofen, E., and Lakshman Y. N. Efficient algorithms for computing the nearest polynomial with a real root and related problems. In *Proc. 1999 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'99)* (New York, N. Y., 1999), S. Dooley, Ed., ACM Press, pp. 205–212.

[13] Huang, Y., Wu, W., Stetter, H. J., and Zhi, L. Pseudofactors of multivariate polynomials. In *Internat. Symp. Symbolic Algebraic Comput. ISSAC 2000 Proc. 2000 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2000), C. Traverso, Ed., ACM Press, pp. 161–168.

[14] Kaltofen, E. Fast parallel absolute irreducibility testing. *J. Symbolic Comput. 1*, 1 (1985), 57–67. Misprint corrections: *J. Symbolic Comput.* vol. 9, p. 320 (1989).

[15] Kaltofen, E. Challenges of symbolic computation my favorite open problems. *J. Symbolic Comput. 29*, 6 (2000), 891–919. With an additional open problem by R. M. Corless and D. J. Jeffrey.

[16] Kaltofen, E. The art of symbolic computation, Mar. 2002. Colloquium at the University of Waterloo. Transparencies at `http://www.math.ncsu.edu/~kaltofen/bibliography/02/waterloo.pdf`.

[17] Kaltofen, E., and May, J. On approximate irreducibility of polynomials in several variables. In Sendra [29], pp. 161–168.

[18] Kaltofen, E., and Trager, B. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symbolic Comput. 9*, 3 (1990), 301–320.

[19] Li, T. Y., and Zeng, Z. A rank-revealing method and its application. Manuscript available at `http://www.neiu.edu/~zzeng/papers.htm`, 2003.

[20] Mora, T., Ed. *ISSAC 2002 Proc. 2002 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2002), ACM Press.

[21] Mourrain, B., Ed. *ISSAC 2001 Proc. 2001 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2001), ACM Press.

[22] Nagasaka, K. Towards certified irreducibility testing of bivariate approximate polynomials. In Mora [20], pp. 192–199.

[23] Nagasaka, K., June 2003. Private email commun.

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

[25] Rupprecht, D. Semi-numerical absolute factorization of polynomials with integer coefficients. *J. Symbolic Comput. 37*, 5 (2004), 557–574.

[26] Sasaki, T. Approximate multivariate polynomial factorization based on zero-sum relations. In Mourrain [21], pp. 284–291.

[27] Sasaki, T., Saito, T., and Hilano, T. Analysis of approximate factorization algorithm I. *Japan J. of Industrial and Applied Mathem. 9*, 3 (Oct. 1992), 351–368.

[28] Sasaki, T., Suzuki, M., Kolář, M., and Sasaki, M. Approximate factorization of multivariate polynomials and absolute irreducibility testing. *Japan J. of Industrial and Applied Mathem. 8*, 3 (Oct. 1991), 357–375.

[29] Sendra, J. R., Ed. *ISSAC 2003 Proc. 2003 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2003), ACM Press.

[30] Sommese, A. J., Verschelde, J., and Wampler, C. W. Numerical factorization of multivariate complex polynomials. *Theoretical Comput. Sci. 315*, 2–3 (2004), 651–669.

[31] Stewart, G. W. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review 15*, 4 (Oct. 1973), 727–764.

[32] Zeng, Z. A method computing multiple roots of inexact polynomials. In Sendra [29], pp. 266–272.

[33] Zeng, Z., and Dayton, B. H. The approximate GCD of inexact polynomials part II: a multivariate algorithm. In *ISSAC 2004 Proc. 2004 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2004), J. Gutierrez, Ed., ACM Press. These proceedings.

[34] Zhi, L. Displacement structure in computing approximate GCD of univariate polynomials. In *Proc. Sixth Asian Symposium on Computer Mathematics (ASCM 2003)* (Singapore, 2003), Z. Li and W. Sit, Eds., vol. 10 of *Lecture Notes Series on Computing*, World Scientific, pp. 288–298.