EXACT BIVARIATE POLYNOMIAL FACTORIZATION OVER $\mathbb{Q}$ BY APPROXIMATION OF ROOTS*

FENG Yong · WU Wenyuan · ZHANG Jingzhong · CHEN Jingwei

DOI: 10.1007/s11424-014-2170-5
Received: 22 August 2012

Abstract Factorization of polynomials is one of the foundations of symbolic computation. Its applications arise in numerous branches of mathematics and other sciences. However, the present advanced programming languages such as C++ and J++, do not support symbolic computation directly. Hence, it leads to difficulties in applying factorization in engineering fields. In this paper, the authors present an algorithm which use numerical method to obtain exact factors of a bivariate polynomial with rational coefficients. The proposed method can be directly implemented in efficient programming language such C++ together with the GNU Multiple-Precision Library. In addition, the numerical computation part often only requires double precision and is easily parallelizable.

Keywords Factorization of multivariate polynomials, interpolation methods, minimal polynomial, numerical continuation.

1 Introduction

Polynomial factorization plays a significant role in many problems including the simplification, primary decomposition, factorized Gröbner basis, solving polynomial equations and some engineering applications, etc. It has been studied for a long time and some high efficient algorithms have been proposed. There are two types of factorization approaches. One is the traditional polynomial factorization for exact input relying on symbolic computation, and the other is approximate polynomial factorization for inexact input.

The traditional polynomial factorization methods follow Zassenhaus' approach[1, 2]. First, multivariate polynomial factorization is reduced to bivariate factorization due to Bertini's theorem and Hensel lifting[3, 4]. Then one of the two remaining variables is specialized at random.

FENG Yong · WU Wenyuan (Corresponding author) · ZHANG Jingzhong · CHEN Jingwei
Chongqing Key Lab of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences, Chongqing 400714, *China.*
Email: yongfeng@cigit.ac.cn; wuwenyuan@cigit.ac.cn; zjz101@yahoo.com.cn; chenjingwei@cigit.ac.cn.
*This work was partly supported by the National Natural Science Foundation of China under Grant Nos. 91118001 and 11170153, the National Key Basic Research Project of China under Grant No. 2011CB302400, and Chongqing Science and Technology Commission Project under Grant No. cstc2013jjys40001.
◇ *This paper was recommended for publication by Editor ZHI Lihong.*

The resulting univariate polynomial is factored and its factors are lifted up to a high enough precision. At last, the lifted factors are recombined to get the factors of the original polynomial. We refer to a series work of Lecerf[5–9] for the recent progress of this routine. In addition, Weimann[10] presented a factorization method without using Hensel lifting. For factoring sparse bivariate polynomials, we refer to [11–14] and references therein.

Approximate factorization is a natural extension of conventional polynomial factorization. It adapts factorization problem to linear algebra first, then applies numerical methods to obtain an approximate factorization in complex which is the exact absolute factorization of a nearby problem. In 1985, Kaltofen and Yagati presented an algorithm for computing the absolute irreducible factorization by floating point arithmetic[15]. Historically, the concept of approximate factorization appeared first in a paper on control theory[16]. The algorithm is as follows: 1) Represent the two factors G and H of the polynomials F with unknown coefficients by fixing their terms; 2) Determine the numerical coefficients so as to minimize $\|F - GH\|$. Huang, et al.[17] pursued this approach, but it seems to be rarely successful, unless G or H is a polynomial of several terms. In 1991, Sasaki, et al. proposed a modern algorithm[18], which use power-series roots to find approximate factors. This algorithm is successful for polynomials of small degrees. Subsequently, Sasaki, et al. presented another algorithm[19] which utilizes zero-sum relations. The zero-sum relations are quite effective for determining approximate factors. However, computation based on zero-sum relations is practically very time-consuming. In [20], Sasaki presented an effective method to get as many zero-sum relations as possible by matrix operations so that approximate factorization algorithm is improved. Meanwhile, Corless, et al. proposed an algorithm for factoring bivariate approximate polynomial based on the idea of decomposition of affine variety in [21]. However, it is not so efficient to generalize the algorithm to multivariate case. Another numerical approach to factorize multivariate complex polynomials is due to the work of numerical algebraic geometry by Sommese, et al.[22]. The authors used the technique of linear trace and monodromy to decompose the complex variety of the input polynomial which leads to absolute factorization by witness sets. A major breakthrough in approximate factorization is due to Kaltofen, et al.[23, 24] who extended Gao's work[25] from symbolics to numerics based on Ruppert matrix and Singular Value Decomposition.

Symbolic factorization has been implemented in many Computer Algebra System. However, it is difficult to implemented directly in Programming Language such as C++ and J++, because most of Programming Language standards do not support symbolic basic operators, and the compilers do not implement the symbolic computation, on which symbolic factorization is based. It restricts exact factorization from being applied in many engineering fields. Compared with symbolic factorization, approximate factorization can be implemented more easily in the popular programming languages. However it only gives approximate results even the input is exact. Fortunately, some work in the direction of symbolic-numeric methods for factorization problem has been done. Rupprecht proposed a numerical strategy with a good practical behavior to output exact result[26]. In [27], Chéze and Galligo have shown an exact absolute factorization can be reduced to an approximate one if the accuracy is good enough. Following this direction, in this paper, we propose an almost completely numerical algorithm, which is not only implemented directly in the programming languages, but also achieve exact results.

Except classic symbolic methods, some approaches have been proposed to obtain exact output by approximation[28, 29]. The idea of obtaining exact polynomial factorization is from the connect between an approximate root of a given polynomial and its minimal polynomial in $\mathbb{Q}$. Certainly, the minimal polynomial is a factor of the given input. Based on lattice basis reduced algorithm LLL and Integer Relation algorithm PSLQ of a vectors respectively, there are two algorithms for finding exact minimal polynomial of an algebraic number from its approximation. One is a numerical algorithm[28, 30] for factorization of a univariate polynomial was provided by Transcendental Evaluation and high-degree evaluation, and the other for factorization of bivariate polynomial is based on LLL[31, 32]. But they are not efficient.

In this paper, we only discuss squarefree polynomials since factoring one polynomial that is not squarefree can be reduced to factoring a squarefree one by computing gcd. Relying on LLL algorithm, we present an almost-completely numerical method for exact factoring polynomial with rational coefficient in $\mathbb{Q}$. First, we choose a sample point in $\mathbb{Q}^{n-1}$ at random. After specialization (i.e., substitution) at the point, the roots of the resulting univariate polynomial can be found very efficiently up to arbitrarily high accuracy. Then applying minimal polynomial algorithm to these roots yields an exact factorization of the univariate polynomial in $\mathbb{Q}$. Next we shall move the sample point in "good direction" to generate enough number of points by using numerical continuation. Especially, for the rest sample points, the corresponding exact factorization can be found by using the same combination of roots as found in the first step. And these roots give more univariate polynomials for the next step. Finally, the multivariate factorization can be obtained by interpolation.

The paper is organized as follows. Section 2 gives a brief introduction of the preparation knowledge. Minimal polynomial algorithm will be discussed in Section 3. Then we present our method in Sections 4, 5, and 6.

2 Preparation

In this section, we briefly introduce the background knowledge and related topics.

2.1 Homotopy Continuation Methods

Homotopy continuation methods play a fundamental role in Numerical Algebraic Geometry and provide an efficient and stable way to compute all isolated roots of polynomial systems. These methods have been implemented in many software packages, e.g., Hom4PS[33], Bertini[34], PHCpack[35].

The basic idea is to embed the target system into a family of systems continuously depending on parameters. Then each point in the parameter space corresponds to a set of solutions. Suppose we know the solutions at a point. Then we can track the solutions from this starting point to the point representing the target system we want to solve.

First let us look at the simplest case: A univariate polynomial $f(z)$ with degree d. We know that $f(z)$ has d roots in $\mathbb{C}$ (counting multiplicities). Of course we can embed $f(z)$ into the family $a_dz^d + a_{d-1}z^{d-1} + \cdots + a_0$, where the a_i are parameters. Now choose a start point corresponding to $z^d - 1$ in this parameter space, whose roots are

$$z_k^0 = \mathrm{e}^{2k\pi\sqrt{-1}/d}, \quad k = 0, 1, \cdots, d-1. \tag{1}$$

Then we use a real straight line in the parameter space to connect $z^d - 1$ with $f(z)$:

$$H(z,t) := tf(z) + (1-t)(z^d - 1). \tag{2}$$

This form is a subclass of the family depending on only one real parameter $t \in [0,1]$.

When $t = 0$ we have the start system $H(z,0) = z^d - 1$ and when $t = 1$ we have our target system $H(z,1) = f(z)$. An important question is to show how to track individual solutions as t changes from 0 to 1. Let us look at the tracking of the solution z_k (the k-th root of $f(z)$). When t changes from 0 to 1, it describes a curve, which is function of t, denoted by $z_k = z_k(t)$. So $H(z_k(t), t) \equiv 0$ for all $t \in [0,1]$. Consequently, we have

$$0 \equiv \frac{dH(z_k(t),t)}{dt} = \frac{\partial H(z,t)}{\partial z}\frac{dz_k(t)}{dt} + \frac{\partial H(z,t)}{\partial t}. \tag{3}$$

This problem is reduced to an ODE for the unknown function $z_k(t)$ together with an algebraic constraint $H(z_k(t),t) \equiv 0$. The initial condition is the start solution $z_k(0) = z_k^0$ and $z_k(1)$ is a solution of our target problem $f(z) = 0$.

Remark 2.1 In [36], Blum, et al. showed that on average an approximate root of a generic polynomial system can be found in polynomial time. Also application of the polynomial cost method for numerically solving differential algebraic equations, Ilie, et al.[37] gave polynomial cost method for solving homotopies.

But there is a prerequisite for the continuous tracking: $\frac{\partial H(z,t)}{\partial z} \neq 0$ along the curve $z = z_k(t)$. If the equations $z - z_k(t) = 0$ and $tf'(z) + d(1-t)z^{d-1} = 0$ have intersection at some point $(t, z_k(t))$, then we cannot continue the tracking. There is way to avoid this singular case, called the "gamma trick" that was first introduced in [38]. We know two complex curves almost always have intersections at complex points, but here t must be real. So if we introduce a random complex transformation to the second curve, the intersection points will become complex points and such a singularity will not appear when $t \in [0,1)$. Let us introduce a random angle $\theta \in [-\pi, \pi]$ and modify the homotopy (2) to

$$H(z,t) := tf(z) + \mathrm{e}^{\mathrm{i}\theta}(1-t)(z^d - 1). \tag{4}$$

Obviously, the k-th starting solution is still z_k^0 in (1) and $z_k(1)$ is still a root of $f(z)$.

2.2 Genericity and Probability One

In an idealized model where paths are tracked exactly and the random angle can be generated to infinite precision, the homotopy (4) can be proved to succeed "with probability one". To clarify this statement, it is necessary to use a fundamental concept in algebraic geometry: Genericity.

Definition 2.2 Let X be an irreducible algebraic variety. We say a property P holds generically on X, if the set of points of X that do not satisfy P are contained in a proper subvariety Y of X. The points in $X \backslash Y$ are called generic points.

The set $X\backslash Y$ is called a Zariski open set of X. Roughly speaking, if Y is a proper subvariety of an irreducible variety X and p is a random point on X with uniform probability distribution, then the probability that $p \notin Y$ is one. So we can consider a random point as a generic point on X without a precise description of Y. Many of the desirable behaviors of homotopy continuation methods rely on this fact.

2.3 Coefficient-Parameter Homotopy

There are several versions of the Coefficient-Parameter theorem in [39]. Here we only state the basic one.

Theorem 2.3 *Let $F(z;q) = \{f_1(z;q), f_2(z;q), \cdots, f_n(z;q)\}$ be a polynomial system in n variables z and m parameters q. Let $\mathcal{N}(q)$ denote the number of nonsingular solutions as a function of q:*

$$\mathcal{N}(q) := \#\left\{z \in \mathbb{C}^n : F(z;q) = 0,\ \det\left(\frac{\partial F}{\partial z}(z;q)\right) \neq 0\right\}. \tag{5}$$

Then,

1) *there exists N, such that $\mathcal{N}(q) \leq N$ for any $q \in \mathbb{C}^m$. Also $\{q \in \mathbb{C}^m : \mathcal{N}(q) = N\}$ is a Zariski open set of $\mathbb{C}^m$. The exceptional set $Y = \{q : \mathcal{N}(q) < N\}$ is an affine variety contained in a variety with dimension $m-1$.*

2) *the homotopy $F(z;\phi(t)) = 0$ with $\phi(t) : [0,1) \to \mathbb{C}^m\backslash Y$ has N continuous non-singular solution paths $z(t)$.*

3) *when $t \to 1^-$, the limit of $z_k(t)$, $k = 1, 2, \cdots, N$ includes all the non-singular roots of $F(z;\phi(1))$.*

An important question is how to choose a homotopy path $\phi(t)$ which can avoid the exceptional set Y. The following lemma gives an easy way to address this problem.

Lemma 2.4 (see [39]) *Fix a point q and a proper algebraic set Y in $\mathbb{C}^m$. For a generic point $p \in \mathbb{C}^m$, the one-real-dimensional open line segment $\phi(t) := (1-t)\, p + t\, q, t \in [0,1)$ is contained in $\mathbb{C}^m\backslash Y$.*

2.4 Reductions

Before factorization of a given polynomial, we shall first apply certain reductions to the input to obtain a square-free polynomial over $\mathbb{Q}$, which can remove multiplicities and ease the computation of the roots. Also we can assume each factor involves all the variables and has more than one term. Otherwise, we can compute the GCD to reduce the problem. For example, let $F = f(x,y)g(y)$. Then $F_x = f_x g$ and $\gcd(F, F_x) = g$ which gives us the factor $g(y)$.

By the Hilbert Irreducibility theorem, we can further reduce the problem to univariate case by random specialization of one variable to a rational number. More precisely, if $f(x,y)$ is irreducible in $\mathbb{Q}[x,y]$, then for a random rational number y_0, $f(x,y_0)$ is also irreducible in the ring $\mathbb{Q}[x]$ with a high probability[40]. However, it is difficult to know if y_0 is a good specialization point or not in advance.

For univariate polynomial, there are symbolic methods to preform exact factorization in $\mathbb{Q}$. Here we are more interested in numeric methods, i.e., from approximate roots to exact factors.

3 Minimal Polynomial by Approximation

Here we recall some material of the paper[30]. There are two methods to compute the minimal polynomial of an algebraic number from its approximation. One is based on the LLL algorithm of the basis reduction[28], and another is based on PSLQ[30]. The later one is more efficient than the former one. However, it can only compute the minimal polynomial of a real algebraic number while the former one can find minimal polynomial of a complex algebraic number. Hence, we introduce the former algorithm which is more suitable for this paper here. We refer the reader to the paper[28] for more details.

Let $p(x) = \sum_{i=0}^{n} p_i x^i$ be a polynomial. The length $||p||$ of $p(x)$ is defined as the Euclidean norm of the vector $(p_0, p_1, \cdots, p_n)$, and the height $||p||_\infty$ as the L_∞-norm of the vector $(p_0, p_1, \cdots, p_n)$. The degree and height of an algebraic number are defined as the degree and height, respectively, of its minimal polynomial.

Suppose that we have upper bound d and H on the degree and height respectively of an algebraic number with $|\alpha| \le 1$, and a complex rational number $\overline{\alpha}$ approximating α such that $|\overline{\alpha}| \le 1$ and $|\alpha - \overline{\alpha}| < 2^{-s}/(4d)$, where s is the smallest positive integer such that $2^s > 2^{d^2/2}(d+1)^{(3d+4)/2}H^{2d}$.

Algorithm 1 (MiniPoly)

Input: An approximation $\overline{\alpha}$ to α (unknown) satisfying the above error control
an upper bound, d, on the degree of α
an upper bound, H, on the height of α

Output: The exact minimal polynomial of α.

For $n = 1, 2, \cdots, d$ in succession, do the following steps

1) Construct

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 2^s \cdot \mathrm{Re}(\overline{\alpha}_0) & 2^s \cdot \mathrm{Im}(\overline{\alpha}_0) \\ 0 & 1 & 0 & \cdots & 0 & 2^s \cdot \mathrm{Re}(\overline{\alpha}_1) & 2^s \cdot \mathrm{Im}(\overline{\alpha}_1) \\ 0 & 0 & 1 & \cdots & 0 & 2^s \cdot \mathrm{Re}(\overline{\alpha}_2) & 2^s \cdot \mathrm{Im}(\overline{\alpha}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 2^s \cdot \mathrm{Re}(\overline{\alpha}_n) & 2^s \cdot \mathrm{Im}(\overline{\alpha}_n) \end{pmatrix}, \tag{6}$$

where $\mathrm{Re}(a)$ and $\mathrm{Im}(a)$ stand for the real part and imaginary part, respectively, of complex a, $\alpha_0 = 1$ and $|\overline{\alpha_i} - \overline{\alpha}^i| \le 2^{-s-1/2}$ for $i = 1, 2, \cdots, n$. Note $\overline{\alpha_i}$ can be computed by rounding the powers of $\overline{\alpha}$ to s bits after the binary points.

2) Denote by b_i the row $i+1$ of the matrix in (6). Apply the basic reduction algorithm to lattice $L_s = (b_0, b_1, \cdots, b_n)$, and obtain the reduced basis of the lattice.

3) If the first basis vector $\widetilde{v} = (v_0, v_1, \cdots, v_n, v_{n+1}, v_{n+2})$ in the reduced basis satisfies $|\widetilde{v}| \le 2^{d/2}(d+1)H$, then return polynomial $v(x) = \sum_{i=0}^{n} v_i x^i$ as the minimal polynomial of algebraic number α.

Note that it is no major restriction to consider α with $|\alpha| \le 1$ only. In fact, if $|\alpha| > 1$ satisfies the polynomial $h(x) = \sum_{i=0}^{d} h_i x^i$, then $1/\alpha$ satisfies the polynomial $\sum_{i=0}^{d} h_{d-i} x^i$. Therefore, if $\sum_{i=0}^{d} h_{d-i} x^i$ is computed, the $h(x)$ is obtained. Furthermore, an ε-approximation $\overline{\alpha}$ to α with $|\alpha| > 1$ easily yields a 3ε-approximation $\overline{\beta}$ to $\beta = 1/\alpha$. This can be easily verified.

The following theorem shows the computation amount of calculating the minimal polynomial of an algebraic number[28].

Theorem 3.1 *Let α be an algebraic number, and let d and H be upper bounds on the degree and height, respectively, of α. Suppose that we are given an approximation $\overline{\alpha}$ to α such that $|\alpha - \overline{\alpha}| \le 2^{-s}/(12d)$, where s is the smallest positive integer such that*

$$2^s > 2^{d^2/2}(d+1)^{(3d+4)/2}H^{2d}.$$

Then the minimal polynomial of α can be determined in $O(n_0 \cdot d^4(d + \log H))$ arithmetic operations on integers having $O(d^2 \cdot (d + \log H))$ binary bits, where n_0 is the degree of α.

4 Finding More Polynomials by Continuation

In the previous stage, we have the factors after specialization , which are univariate polynomials. To construct the factor of two variables by using interpolation, we need more information, i.e. specializations at more points. The main tool is the homotopy continuation method.

4.1 Applying Numerical Continuation to Factorization

Suppose an input polynomial $F(x, y)$ is reducible. Geometrically, if $\mathcal{C}$ denotes the zero set of f, i.e., the union of many curves in $\mathbb{C}^2$, removing the singular locus of $\mathcal{C}$ from each curve $\mathcal{C}_i$, the regular sets $\mathcal{S}_i$ are connected in $\mathbb{C}^2$. Moreover, the singular locus has lower dimension, consequently it is a set of isolated points.

Suppose $f(x, y)$ is an irreducible factor of F in $\mathbb{Q}$. Let y_0, y_1 be random rational numbers. By the Hilbert Irreducibility theorem the univariate polynomials $f_0 = f(x, y_0)$ and $f_1 = f(x, y_1)$ are irreducible as well. Suppose we know the roots of f_0. Then we can choose a path to connect y_0 and y_1 avoiding the singular locus which has measure zero. By the Coefficient-Parameter theorem, all the roots of f_1 can be obtained by the following homotopy continuations:

$$\begin{cases} f(x, y) = 0, \\ (1-t)(y - y_0) + t(y - y_1)\gamma = 0. \end{cases} \tag{7}$$

Moreover any generic complex number γ implies that the homotopy path can avoid the singular locus by Lemma 2.4 when we track the path.

4.2 Control of the Precision

Let $\{x_1, x_2, \cdots, x_m\}$ be the exact roots of f_1 and g be the primitive polynomial of f_1. Then

$$g = \alpha \prod_{i=1}^{m} (x - x_i) \in \mathbb{Z}[x], \tag{8}$$

for some integer number α.

Note that we only have the approximate roots $\{\widetilde{x}_1, \widetilde{x}_2, \cdots, \widetilde{x}_m\}$.

Proposition 4.1 *Let* $p = \prod_{i=1}^{m}(x - x_i)$ *and* $\widetilde{p} = \prod_{i=1}^{m}(x - \widetilde{x}_i)$. *Let* $\delta = \max_{i=1,2,\cdots,m}\{|x_i - \widetilde{x}_i|\}$ *and* $r = \max_{i=1,2,\cdots,m}\{|\widetilde{x}_i|\}$. *If* δ *is sufficiently small. Then*

$$||\widetilde{\boldsymbol{p}} - \boldsymbol{p}||_\infty \le \left(\max_{i=1,2,\cdots,m} \left\{ r^{i-1} \binom{m-1}{i-1} \right\} m + 1 \right) \cdot \delta. \tag{9}$$

Proof Let $x_i = \widetilde{x}_i + \delta_i$. Thus, $|\delta_i| \le \delta$. The left hand side $||\widetilde{\boldsymbol{p}} - \boldsymbol{p}||_\infty = ||\prod_{i=1}^{m}(x - x_i + \delta_i) - \prod_{i=1}^{m}(x - x_i)|| = ||\sum_{j=1}^{m}\prod_{i \ne j}(x - x_j)\delta_j|| + o(\delta)$. An upper bound of the coefficients of $\prod_{i \ne j}(x - x_j)$ with respect to x^{m-i} is $\binom{m-1}{i-1} r^{i-1}$. Hence, $||\widetilde{\boldsymbol{p}} - \boldsymbol{p}||_\infty \le \max_{i=1,2,\cdots,m} \binom{m-1}{i-1} r^{i-1} \cdot m\delta + \delta$. ∎

Remark 4.2 A similar precision bound result was given in [27]. For some examples the required precision might be high. But note that the homotopy continuation can still work in double precision, instead of the precision bound, and the only assumption is the condition number of the Jacobian matrix is good enough. But approximate roots given by homotopy often need the Newton refine step where high precision must be satisfied. Secondly, we shall point out that the homotopy paths are independent, so this stage is naturally parallizable.

Now let us consider how to find α. Suppose the input polynomial is $F(x, y)$ and f is a factor of F. The primitive polynomial of $f(x, y_1)$, which is g, must be a factor of the primitive polynomial of $F(x, y_1)$. Thus, the leading coefficient of g must be a factor of the leading coefficient of the primitive polynomial of $F(x, y_1)$. Let α be the leading coefficient of the primitive polynomial of $F(x, y_1)$. Then let $p = \alpha \prod_{i=1}^{m}(x - x_i) \in \mathbb{Z}[x]$. Note that itself may not be primitive, but its primitive polynomial is g.

Let $M = \max_{i=1,2,\cdots,m}\{i\ r^{i-1}\binom{m}{i}\} + 1$ and $\widetilde{p} = \alpha \prod_{i=1}^{m}(x - \widetilde{x}_i)$. Thus, if $\delta < \frac{1}{2\alpha M}$ then $||p - \widetilde{p}||_\infty < 0.5$. It means that we can round each coefficient of $\widetilde{p}$ to the nearest integer to obtain exact polynomial p which gives g.

4.3 Detecting the Degrees of Factors

After specialization at $y = y_0$, we obtain the information about the number of factors and the degree of each factor with respect to x. The degrees with respect to y of factors provide the bound of the number of interpolation nodes. Certainly, we can use the degree of the input $\deg_y(F)$ as the bound. However, the degrees with respect to y of factors are usually much less than $\deg_y(F)$, especially when there are many factors. Therefore, for high efficiency, it is better to know the degree with respect to y of each factor.

We define the notation of 2-tuple degree to be

$$\deg(F) = [\deg_x(F), \deg_y(F)].$$

Suppose $\deg(F) = [m, n]$ and F has r factors. Applying an approach of univariate polynomial solving to $F(x, y_0)$ and $F(x_0, y)$ yields points on the curve $A = \{(x_1, y_0), (x_2, y_0), \cdots, (x_m, y_0)\}$ and $B = \{(x_0, y_1), (x_0, y_2), \cdots, (x_0, y_n)\}$, respectively. In addition, we also know the decomposition of two points sets in r groups with cardinalities $\{a_1, a_2, \cdots, a_r\}$ and $\{b_1, b_2, \cdots, b_r\}$ by minimal polynomials. Moreover $\sum a_i = m$ and $\sum b_i = n$ and $m \geq r, n \geq r$. One way to obtain the degree information is to apply numerical continuation to pair the groups, which is similar to "point membership test"[39].

Choose one point from each group of the first set A. Starting from these points, we track the homotopy path

$$\begin{cases} F(x, y) = 0, \\ (1-t)(y - y_0) + t(x - x_0)\gamma = 0. \end{cases} \tag{10}$$

Because of the random choices of y_0, x_0, and γ, the path avoids the singular locus. When $t = 1$, the finite endpoint must belong to the second set B. For example if the starting point of the first group of A and its end point belongs to the ith group of B. Then we know the first factor has degree $[a_1, b_i]$. Similarly, the degrees of other factors can be detected in the same way. However, if $\deg_y(F)$ and $\deg_x(F)$ are less than the total degree of F, the end point may diverge. In our examples, it works very well, but there is no theoretical guarantee for the success.

An alternative way which can easily avoid group pairing is the change of coordinates

$$\begin{pmatrix} x \\ y \end{pmatrix} = A \cdot \begin{pmatrix} \widehat{x} \\ \widehat{y} \end{pmatrix}, \tag{11}$$

where A is a random 2×2 matrix over $\mathbb{Q}$.

Then the degree of each factor with respect to $\widehat{x}$ and $\widehat{y}$ must be same and is equal to the total degree of this factor. For instance, if $F(x, y_0)$ has r factors in $\mathbb{Q}$ with degree $d_1, d_2, \cdots, d_r$ respectively. Then immediately we know that the degree of the factor $f_i(x, y)$ must have the 2-tuple degree (d_i, d_i) without any cost. Because we only consider general polynomials and ignore the sparse structure, the change of coordinates is suitable in this article.

5 Interpolation

Polynomial interpolation is a classical numerical method. It is studied very well for univariate polynomials in numerical computation. Polynomial interpolation problem is to determine a polynomial $f(x) \in F[x]$ with degree not greater than $n \in \mathbb{N}$ for a given pairs $\{(x_i, f_i), i = 0, 1, \cdots, n\}$ satisfying $f(x_i) = f_i$ for $i = 0, 1, \cdots, n$, where F is a field and $x_i, f_i \in F$. In general, there are four types of polynomial interpolation method: Lagrange

Interpolation, Neville's Interpolation, Newton's Interpolation, and Hermite Interpolation. Lagrange interpolation and Newton's Interpolation formula are suited for obtaining interpolation polynomial for a given set $\{(x_i, f_i), i = 0, 1, \cdots, n\}$. Neville's interpolation method aims at determining the value of the interpolating polynomial at some point. If the interpolating problem prescribes at each interpolation point $\{x_i, i = 0, 1, \cdots, n\}$ not only the value but also the derivatives of desired polynomial, then the Hermite formula is preferred.

Different from the traditional interpolation problem above, our problem is to construct a bivariate polynomial from a sequence of univariate polynomials at chosen nodes. It is important to point out that the univariate polynomials are constructed by roots, which may not be equal to the polynomials by substitutions. But the only difference for each polynomial is just a scaling constant.

More precisely, in this paper, we aim to solve a special polynomial interpolation problem: Given a set of nodes and square free polynomials $\{(y_i \in F, f_i(x) \in F[x]), i = 0, 1, \cdots, k\}$, compute a square free polynomial $f(x, y) \in F[x, y]$ of degree with respect to x not greater than n, where F is a field, such that $f(x, y_i)$ and $f_i(x)$ have the same roots.

5.1 Illustrative Examples

Example 5.1 Let $f = x^2 + y^2 - 1$. Since its degree with respect to y is two, we need three interpolation nodes which are $y = -1/2, 0, 1/2$. Suppose we know the roots at each node, then the interpolating polynomials are $\{f_0 = x^2 - 3/4, f_1 = x^2 - 1, f_2 = x^2 - 3/4\}$. To construct original polynomial f, we can use Lagrange method. Let $\ell_1 = \frac{y(y-1/2)}{(-1/2-0)(-1/2-1/2)} = 2y^2 - y$. Similarly, $\ell_2 = -4y^2 + 1$ and $\ell_3 = 2y^2 + y$. It is easy to check that $(x^2 - 3/4)\ell_1 + (x^2 - 1)\ell_2 + (x^2 - 3/4)\ell_3 = f$.

In the example above, the coefficient of f with respect to x^2 is a constant 1. Making the interpolating polynomials given by (8) monic, we can construct f correctly by Lagrange basis. However, if the coefficient is nonconstant, i.e., a polynomial of y, then it is not straightforward to find f. The example below shows this problem.

Example 5.2 Let $f = xy - 1$. The nodes are $y = 2, 3$. We know the roots are $1/2, 1/3$ respectively at the nodes. Then the monic interpolating polynomials are $\{x-1/2, x-1/3\}$. If we still apply Lagrange basis $\ell_1 = -y+3$, $\ell_2 = y-2$, it gives $(x-1/2)(-y+3)+(x-1/3)(y-2) = x + 1/6\,y - 5/6$ which is totally different from the target polynomial $xy - 1$.

The basic reason is that the interpolating polynomials are not the polynomials after specializations, and the only difference is certain scaling constants. To find these constants, we need more information.

Now we use one more node: When $y = 4$, the monic interpolating polynomial is $x - 1/4$. By multiplying a scaling constant to f we can assume $f(x, 4) = x - 1/4$, then there exist a, b such that $f(x, 2) = a(x - 1/2)$ and $f(x, 3) = b(x - 1/3)$. The corresponding Lagrange bases are $\ell_1 = (y - 3)(y - 4)/2$, $\ell_2 = -(y - 2)(y - 4)$, $\ell_3 = (y - 2)(y - 3)/2$. Then $f = a(x - 1/2)\ell_1 + b(x - 1/3)\ell_2 + (x - 1/4)\ell_3$. The coefficient of f with respect to y^2 must be zero.

Consequently we have

$$\frac{1}{2}\left(x-\frac{1}{2}\right)a+\left(x-\frac{1}{3}\right)b+\frac{1}{2}x-\frac{1}{8}=0, \tag{12}$$

which implies a linear system

$$\frac{1}{2}a-b+\frac{1}{2}=0, -\frac{1}{4}a+\frac{1}{3}b-\frac{1}{8}=0.$$

The solution is $a=1/2, b=3/4$. Substituting them back to two nodes interpolation formula yields the polynomial we need, up to a constant $1/4$:

$$\frac{1}{2}\left(x-\frac{1}{2}\right)(-y+3)+\frac{3}{4}\left(x-\frac{1}{3}\right)(y-2)=\frac{1}{4}(xy-1).$$

5.2 Interpolation with Indeterminates

To extend the idea in Example 5.2, we present a method to construct desired bivariate polynomial by using monic univariate interpolating polynomials.

Suppose f is irreducible and its degrees with respect to x and y are m and n, respectively. Consider x as the main variable, we can express this polynomial by $f=\sum_{i=0}^{m}c_i(y)x^i$, where c_i are polynomials of y of degree less than or equal to n. We can consider each c_i as a vector in monomial basis. Suppose there are r linearly independent coefficients. If $r=1$, then $c_i(y)=a_ic_0(y)$ for some constant a_i and $f=(\sum_{i=0}^{m}a_ix^i)\cdot c_0(y)$. It contradicts the assumption that f is irreducible. Hence, $r\geq 2$.

Now we consider how to construct f by using the interpolating polynomials

$$\{f_0(x), f_1(x), \cdots, f_k(x)\}$$

at $k+1$ nodes $\{y_0, y_1, \cdots, y_k\}$ respectively chosen at random.

Let C be a $(k+1)\times(m+1)$ matrix $[\boldsymbol{c}_0, \boldsymbol{c}_1, \cdots, \boldsymbol{c}_m]$, where $\boldsymbol{c}_i$ is the column vector in monomial basis $\{y^k, y^{k-1}, \cdots, 1\}$ of the polynomial c_i. Let V be the Vandermonde matrix

$$\begin{pmatrix} y_0^k & y_0^{k-1} & \cdots & 1 \\ y_1^k & y_1^{k-1} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ y_k^k & y_k^{k-1} & \cdots & 1 \end{pmatrix}.$$

Let A be a $(k+1)\times(m+1)$ matrix, where A_{ij} is the coefficient of the ith interpolating polynomial with respect to x^j. To make the solution unique, we may fix $f(x,y_0)=f_0$ and suppose $f(x,y_i)=\lambda_i f_i$ and $\lambda_i\neq 0$ for $i=1,2,\cdots,k$. Let $\Lambda=\mathrm{diag}(1,\lambda_1,\lambda_2,\cdots,\lambda_k)$.

Therefore,

$$V\cdot C=\Lambda\cdot A. \tag{13}$$

Since $\{y_i\}$ are distinct, the Vandermonde matrix has inverse and consequently $C=V^{-1}\cdot\Lambda\cdot A$. By our assumption, the degree with respect to y is n. It means that the first $k-n$ rows of C

must be zero. The zero at the ith row and jth column corresponds an equation. Thus, it leads to a linear system

$$\mathrm{Row}(V^{-1}, i) \cdot \Lambda \cdot \mathrm{Col}(A, j) = 0, \tag{14}$$

for $1 \le i \le k - n$ and $1 \le j \le m + 1$ with k unknowns.

Only r linearly independent columns in A, so there are $(k - n)\ r$ equations and k unknowns. The existence of the solution is due to the origination of the interpolating polynomials $f(x, y_i) = \lambda_i f_i$ for $i = 1, 2, \cdots, k$. The linear system has unique solution implies that $(k - n)r \ge k$. Thus, $k \ge rn/(r - 1)$. Let μ be the smallest integer greater than or equal to $\frac{rn}{r-1}$, namely

$$\mu = \left\lceil \frac{rn}{r - 1} \right\rceil. \tag{15}$$

Thus, to determine the scaling constants $\{\lambda_i\}$, we need at least μ more interpolation nodes.

To find an upper bound for the number of nodes, let us consider f as a monic polynomial with rational function coefficients. All the coefficients $\{c_m, c_{m-1}, \cdots, c_0\}$ can be uniquely determined by rational function interpolation of $x^m + c_{m-1}/c_m x^{m-1} + \cdots + c_0/c_m$ at $2n+1$ nodes. Therefore, it requires $2n$ nodes except the initial one. Thus, we have $\mu \le k \le 2n$.

But this upper bound is often overestimated, and for some special case the polynomial f can be constructed by using less nodes.

Proposition 5.3 *Let f be a polynomial in $\mathbb{Q}[x, y]$ and $\deg(f) = [m, n]$. Suppose $m \ge n$ and f has $n + 1$ linearly independent coefficients. Then f can be uniquely determined by $n + 2$ monic interpolating polynomials $\{f_0(x), f_1(x), \cdots, f_{n+1}(x)\}$ up to a scaling constant.*

Proof Suppose the first $n+1$ columns of A are linearly independent. By Equation (13), we construct $n + 1$ equations: $\mathrm{Row}(V^{-1}, 1) \cdot \Lambda \cdot \mathrm{Col}(A, j) = 0$, for $j = 1, 2, \cdots, n + 1$. Let B be the transpose of the submatrix consisting of the first $n+1$ columns of A and $\boldsymbol{v} = (v_1, v_2, \cdots, v_{n+2})^t$ be the transpose of $\mathrm{Row}(V^{-1}, 1)$. Thus,

$$\mathbf{0} = B \cdot \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_{n+2}) \cdot \boldsymbol{v} = B \cdot \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_{n+2}) \cdot (\lambda_1, \lambda_2, \cdots, \lambda_{n+2})^t.$$

Because the first $n + 1$ coefficients of f are linearly independent, the evaluations of them at $n + 2$ random points must be linearly independent. So the rank of B is $n + 1$. Here v can be expressed by explicit form of the Vandermonde matrix[27] which is a vector of polynomials of $\{y_0, y_1, \cdots, y_{n+1}\}$. For generic choice of $\{y_0, y_1, \cdots, y_{n+1}\}$, each $v_i \ne 0$. Hence, the rank of $B \cdot \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_{n+2})$ is still $n + 1$ and its nullity equals one. We can choose any solution $\{\lambda_i\}$ to construct f by Lagrange basis: $\sum_{i=0}^{n} \lambda_i f_i \ell_i$.

Remark 5.4 In our algorithm, we compute $\{\lambda_i\}$ starting from μ more nodes (together with the initial node y_0), and we add incrementally more nodes if necessary. But interestingly, the experimental results show that the lower bound μ is often enough. We test 5 groups of random polynomials, 1000 polynomials in each group, generated by the following Maple code.
[> randpoly($[x, y]$, degree=5)+rand($-100..100$)();
The success portion of correct interpolations by using μ more nodes is 82.86% on average.

However, there exist some polynomials, for instance $29\,x^3+\left(60-11\,y^3\right)x^2+52\,y^3x-5-31\,y^3$, where $\mu=4$. But the upper bound is reached, i.e., $2n=6$ more nodes are necessary for the correct interpolation.

Algorithm 2 (Interpolation)

```
Input: A set of polynomials $\{f_0(x), f_1(x), \cdots, f_k(x)\} \subset \mathbb{Z}[x,y]$
       a set of rational numbers $\{y_0, y_1, \cdots, y_k\}$
       an integer $n$ the degree of $f$ with respect to $y$
Output: $f \in \mathbb{Z}[x,y]$, such that $f(x,y_i) = f_i(x)$.
1) Let $A$ be the matrix consisting of the coefficient row vectors of the input univariate
   polynomials.
2) Let $r = \mathrm{Rank}(A)$. If $k < \mu$, then it needs more interpolation nodes.
3) Solve the homogenous linear system (14) to obtain the scaling constants $\{\lambda_1, \lambda_2, \cdots, \lambda_k\}$.
4) If the solution is not unique, then it needs more interpolation nodes.
5) Else $f = \sum_{i=0}^{k} \lambda_i f_i \ell_i \in \mathbb{Q}[x,y]$.
6) Return the primitive polynomial of $f$.
```

Note that we can also use a reversible linear transformation of coordinates (see the end of Section 4) to make the leading coefficient of the resulting polynomial in x be a rational number, rather than a polynomial in y and the resulting polynomial can be reduced to be monic in x by dividing the leading coefficient. Then the interpolation becomes the naive one. For the polynomial $xy-1$ of Example 5.2, we can use a linear transformation

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} \widehat{x} \\ \widehat{y} \end{pmatrix}, \tag{16}$$

which leads to a new polynomial $\widehat{x}^2 + \frac{3}{2}\,\widehat{x}\widehat{y} + \frac{1}{2}\widehat{y}^2 - \frac{1}{2}$ and it can be interpolated by using 3 nodes with the naive method. Then, $xy-1$ can be recovered from (16). Along this way, the number of interpolating nodes can be fixed as the total degree of the original polynomial plus 1. However, our new interpolating method needs fewer nodes in general but, unfortunately, we can only give a range (15), rather than an exact formula on the number of nodes, as indicated in Remark 5.4.

6 Combination of Tools

Now, we combine the tools introduced in previous sections to obtain a new factorization algorithm. A preliminary version of the algorithm is implemented in Maple. For the efficiency, it requires a more sophisticate version in C++. Especially, the polynomials at the interpolation nodes for each factor can be computed independently. Therefore, a parallel program for interpolation can be implemented on a more powerful machine, e.g., a computer cluster. But in this paper we mainly focus on the theoretical aspect of the algorithm.

6.1 Main Steps of the Algorithm

We now describe the main algorithm in this article.

Algorithm 3 (Factorization)

Input: f, a squarefree primitive polynomial $f \in \mathbb{Z}[x, y]$ such that $\gcd(f, f_x) = 1$.
Output: F, a set of primitive polynomials $\{f_1, f_2, \cdots, f_r\} \subset \mathbb{Z}[x, y]$, such that $f = \prod f_i$.

1) Apply a numerical solver to approximate the roots of $f(x, y_0) = 0$ and $f(x_0, y) = 0$ at generic points $x_0, y_0 \in \mathbb{Q}$.

2) Apply miniPoly to roots above and decompose the solutions. And generate the minimal polynomials for them and we have grouping information of roots. In this step, it needs Newton iteration to refine the roots up to desired accuracy.

3) Apply homotopy (10) to obtain the degrees of each factor.

4) For group i (corresponding to the factor f_i), $i = 1, 2, \cdots, r$, use Homotopy (7) to generate its approximate roots at random rational numbers $\{y_1, y_2, \cdots, y_k\}$.

5) For each set of roots at y_j, refine the roots to the accuracy given by Proposition 4.1, then make the product and construct the polynomial $f_i(x, y_j)$.

6) Call interpolate with the interpolating polynomials $\{f_i(x, y_0), f_i(x, y_1), \cdots, f_i(x, y_k)\}$ to construct $f_i(x, y)$.

Remark 6.1 As mentioned before, if y_0 is not a good specialization point for the Hilbert Irreducible theorem, then the factorization is incorrect which can be easily checked by polynomial division. Therefore, the algorithm fails. One obvious way is to recompute by a different y_0. Or, we have to solve a recombination problem which is difficult and it is beyond the scope of this paper.

6.2 A Simple Example

Let us consider a polynomial $f = (x\,y - 2)\,(x^2 + y^2 - 1)$. First, we choose a sequence of random rational numbers $\{97/101, 1, 104/101, 123/101, 129/101, \cdots\}$. Substituting $y = 97/101$ into f yields Mignotte bound of the coefficients of factors 9170981 and the digits required to produce the minimal polynomial is 110 by Theorem 3.1. Then compute the approximate roots of $f(x, 97/101)$ up to 110 digits accuracy. The miniPoly subroutine gives two groups of points: $[[1, 2], [3]]$ and the corresponding minimal polynomials $[-792 + 10201\,x^2, -202 + 97\,x]$. By the Hilbert Irreducibility theorem, there are two factors. On the other hand, fix the value of x and obtain the univariate polynomials $[-202 + 97\,y, -792 + 10201\,y^2]$ and $[[3], [1, 2]]$.

Starting from the first point of group one, the Homotopy (10) path ends at a point which satisfies $-792 + 10201\,y^2$. It implies that $-792 + 10201\,x^2$ and $-792 + 10201\,y^2$ are from the same factor of degree $[2, 2]$. By Equation (15), we need $\mu = \lceil \frac{rn}{r-1} \rceil = 4$ more interpolating polynomials which are produced by Homotopy (7). Thus, there are five polynomials $[-792 + 10201\,x^2, x^2, 615 + 10201\,x^2, 4928 + 10201\,x^2, 6440 + 10201\,x^2]$. The scaling constants $[\lambda_1 =$

$1, \lambda_2 = 10201, \lambda_3 = 1, \lambda_4 = 1, \lambda_5 = 1]$ are obtained by System (14). Consequently, the Lagrange interpolation formula gives the correct factor $-1 + x^2 + y^2$.

Since the degree of the other factor is $[1, 1]$, it needs $\mu = 2$ more polynomials and they are $[-202 + 97\,x, x - 2, -101 + 52\,x]$. The corresponding scaling constants are $[\lambda_1 = 1/2, \lambda_2 = 101/2, \lambda_3 = 1]$ and the resulting factor is $x\,y - 2$.

6.3 Experiments

While our implementation is still young and the parallel algorithm is not implemented in this paper, the current version is not as efficient as Maple's factor in general. Here, we only consider the following examples.

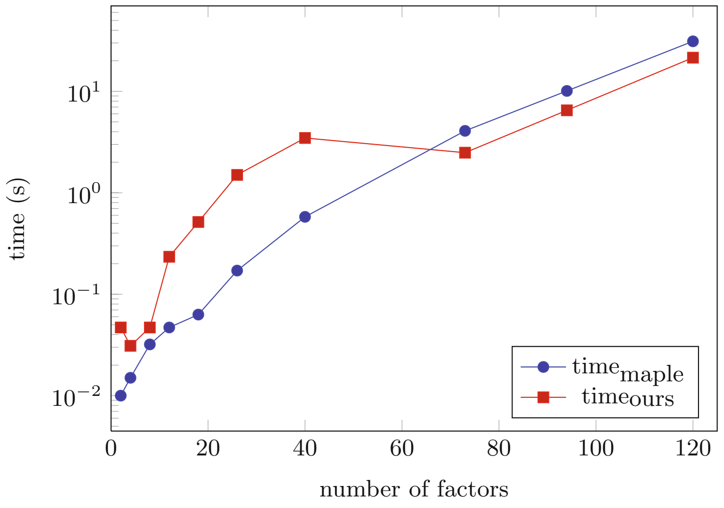

Figure 1 Running time (in seconds) of Algorithm 3 and Maple's factor

We have tested Algorithm 3 to compare its practical performance with that of Maple's factor. We use polynomials of form $\prod_{i=1}^{s}(x+y+i)$ for some s in the range of 2 to 120. For $s = 120$, the corresponding polynomial has degree [120, 120] and height $2.1313\cdots \times 10^{201}$. Figure 1 shows the running times, suggesting that Algorithm 3 seems slightly faster than Maple's built-in function factor when s grows large.

We note that the running times of Algorithm 3 for general polynomials are slower than that of factor. The main reason is that the performance of miniPoly depends on multi-precision numerical operations, which influences on the efficiency heavily. However, it is easy to find that the Steps 2), 4), 5), and 6) of Algorithm 3 can be parallelized. Therefore, we believe that our algorithm will show its real advantage with a parallel implementation in an efficient programming language.

7 Conclusion

A new numerical method to factorize bivariate polynomials exactly is presented in this article. We preliminarily implemented our algorithm in Maple to verify the correctness. More importantly, the main components of our algorithm, miniPoly, and numerical homotopy continuation can be implemented directly in C++ or J++ with existing multi-precision packages, e.g., GNU MP library. Furthermore, these two numerical components are naturally parallelizable. Therefore, it gives an alternative way to exact factorization which can take the advantages of standard programming languages and parallel computation techniques widely used by industries.

As an initial step towards the new direction of obtaining exact result from approximation, our method, currently, is not comparable with well developed symbolic factorization methods, e.g., Hensel lifting technique. But being a different approach, it needs more work to explore its advantages and minimize its disadvantages.

In this article, we mainly focus on bivariate case. It is quite straightforward to extend to multivariate case. However, the number of the interpolation nodes grows exponentially as the increasing of the number of monomials. A more practical way to deal with such difficulty is to exploit the sparsity if the factors are sparse. It desires the further study in our future work.

References

[1] Zassenhaus H, On hensel factorization, I, *Journal of Number Theory*, 1969, **1**(3): 291–311.

[2] Zassenhaus H, A remark on the hensel factorization method, *Mathematics of Computation*, 1978, **32**(141): 287–292.

[3] von zur Gathen J, Irreducibility of multivariate polynomials, *Journal of Computer and System Sciences*, 1985, **31**(2): 225–264.

[4] von zur Gathen J and Gerhard J, *Modern Computer Algebra*, 2nd edition, Cambridge University Press, London, 2003.

[5] Lecerf G, Sharp precision in Hensel lifting for bivariate polynomial factorization, *Mathematics of Computation*, 2006, **75**(254): 921–934.

[6] Lecerf G, Improved dense multivariate polynomial factorization algorithms, *Journal of Symbolic Computation*, 2007, **42**(4): 477–494.

[7] Lecerf G, Fast separable factorization and applications, *Applicable Algebra in Engineering, Communication and Computing*, 2008, **19**(2): 135–160.

[8] Lecerf G, New recombination algorithms for bivariate polynomial factorization based on Hensel lifting, *Applicable Algebra in Engineering, Communication and Computing*, 2010, **21**(2): 151–176.

[9] Berthomieu J and Lecerf G, Reduction of bivariate polynomials from convex-dense to dense, with application to factorizations, *Mathematics of Computation*, 2012, **81**(279): 1799–1821.

[10] Weimann M, Factoring bivariate polynomials using adjoints, *Journal of Symbolic Computation*, 2013, **58**: 77–98.

[11] Abu Salem F, An efficient sparse adaptation of the polytope method over $\mathbb{F}_p$ and a record-high binary bivariate factorisation, *Journal of Symbolic Computation*, 2008, **43**(5): 311–341.

[12] Weimann M, A lifting and recombination algorithm for rational factorization of sparse polynomials, *Journal of Complexity*, 2010, **26**(6): 608–628.

[13] Chattopadhyay A, Grenet B, Koiran P, Portier N, and Strozecki Y, Factoring bivariate lacunary polynomials without heights, in *Proceedings of the* 2013 *international symposium on Symbolic and Algebraic Computation*, Boston, USA, 2013, 141–148.

[14] Wu W, Chen J, and Feng Y, Sparse bivariate polynomial factorization, *Science China Mathematics*, 2014, **57**, doi: 10.1007/s11425-014-4850-y.

[15] Kaltofen E and Yagati L, Improved sparse multivariate polynomial interpolation algorithms, in Gianni P, editor, *Symbolic and Algebraic Computation*, *Lecture Notes in Computer Science*, Springer, 1989, **358**: 467–474.

[16] Mou-Yan Z and Unbehauen R, Approximate factorization of multivariable polynomials, *Signal Processing*, 1988, **14**(2): 141–152.

[17] Huang Y, Wu W, Stetter H J, and Zhi L, Pseudofactors of multivariate polynomials, in *Proceedings of the* 2000 *International Symposium on Symbolic and Algebraic Computation*, St. Andrews, Scotland, 2000, 161–168.

[18] Sasaki T, Suzuki M, Kolář M, and Sasaki M, Approximate factorization of multivariate polynomials and absolute irreducibility testing, *Japan Journal of Industrial and Applied Mathematics*, 1991, **8**(3): 357–375.

[19] Sasaki T, Saito T, and Hilano T, Analysis of approximate factorization algorithm I, *Japan Journal of Industrial and Applied Mathematics*, 1992, **9**(3): 351–368.

[20] Sasaki T, Approximate multivariate polynomial factorization based on zero-sum relations, in *Proceedings of the* 2001 *International Symposium on Symbolic and Algebraic Computation*, London, Canada, 2001, 284–291.

[21] Corless R M, Giesbrecht M W, van Hoeij M, Kotsireas I, and Watt S M, Towards factoring bivariate approximate polynomials, in *Proceedings of the* 2001 *International Symposium on Symbolic and Algebraic Computation*, London, Canada, 2001, 85–92.

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

[23] Gao S, Kaltofen E, May J P, Yang Z, and Zhi L, Approximate factorization of multivariate polynomials via differential equations, in *Proceedings of the* 2004 *International Symposium on Symbolic and Algebraic Computation*, Santander, Spain, 2004, 167–174.

[24] Kaltofen E, May J P, Yang Z, and Zhi L, Approximate factorization of multivariate polynomials using singular value decomposition, *Journal of Symbolic Computation*, 2008, **43**(5): 359–376.

[25] Gao S, Factoring multivariate polynomials via partial differential equations, *Mathematics of Computation*, 2003, **72**(242): 801–822.

[26] Rupprecht D, Semi-numerical absolute factorization of polynomials with integer coefficients, *Journal of Symbolic Computation*, 2004, **37**(5): 557–574.

[27] Chéze G and Galligo A, From an approximate to an exact absolute polynomial factorization, *Journal of Symbolic Computation*, 2006, **41**(6): 682–696.

[28] Kannan R, Lenstra A K, and Lovász L, Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers, *Mathematics of Computation*, 1988, **50**(181): 235–250.

[29] Zhang J and Feng Y, Obtaining exact value by approximate computations, *Science in China Series A, Mathematics*, 2007, **50**(9): 1361–1368.

[30] Qin X, Feng Y, Chen J, and Zhang J, A complete algorithm to find exact minimal polynomial by approximations, *International Journal of Computer Mathematics*, 2012, **89**(17): 2333–2344.

[31] van der Hulst M-P and Lenstra A K, Factorization of polynomials by transcendental evaluation, in Caviness B F, editor, *EUROCAL* '85, *Lecture Notes in Computer Science*, 1985, **204**: 138–145.

[32] Chen J, Feng Y, Qin X, and Zhang J, Exact polynomial factorization by approximate high degree algebraic numbers, *Proceedings of the* 2009 *Conference on Symbolic Numeric Computation*, Kyoto, Japan, 2009, 21–28.

[33] Lee T L, Li T Y, and Tsai C H, HOM4PS-2.0, a software package for solving polynomial systems by the polyhedral homotopy continuation method, *Computing*, 2008, **83**(2–3): 109–133.

[34] Bates D J, Hauenstein J D, Sommese A J, and Wampler C W, Bertini, Software for numerical algebraic geometry, Available at https://bertini.nd.edu/, June 2014.

[35] Verschelde J, Algorithm 795, PHCpack, A general-purpose solver for polynomial systems by homotopy continuation, *ACM Transactions on Mathematical Software*, 1999, **25**(2): 251–276.

[36] Blum L, Cucker F, Shub M, and Smale S, *Complexity and Real Computation*, Springer, New York, 1998.

[37] Ilie S, Corless R M, and Reid G, Numerical solutions of index-1 differential algebraic equations can be computed in polynomial time, *Numerical Algorithms*, 2006, **41**(2): 161–171.

[38] Morgan A and Sommese A J, A homotopy for solving general polynomial systems that respects m-homogeneous structures, *Applied Mathematics and Computation*, 1987, **24**(2): 101–113.

[39] Sommese A J and Wampler C W, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, Singapore, 2005.

[40] Fried M, On Hilbert's rrreducibility theorem, *Journal of Number Theory*, 1974, **6**(3): 211–231.