New algorithms for exact and approximate polynomial decomposition

Mark Giesbrecht and John May

Abstract. Computing a decomposition of a polynomial $f(x)$ as a functional composition $g(h(x))$ of polynomials $g(x)$ and $h(x)$, is an important and well-studied problem, both for exact and approximate inputs. In this paper, we re-examine the original (exponential-time) algorithm of Barton and Zippel for this task, which looks for special factors of an associated *separated* bivariate polynomial. We demonstrate algorithms using this approach which are reasonably fast (i.e., run in a polynomial number of operations) for exact computation, and provide an effective new approach for the decomposition of approximate polynomials. For approximate polynomials we exhibit rigorous lower bounds on the distance to the nearest decomposable polynomial, as well as robust numerical algorithms.

1. Introduction

Given a polynomial $f \in \mathsf{K}[x]$ of degree n over a field K, the problem of polynomial decomposition asks if there exist polynomials $g, h \in \mathsf{K}[x]$ such that $f(x) = g(h(x))$ with $1 < \deg g, \deg h < n$. This problem has been studied for exact polynomials and rational functions by many authors, including Alonso, Gutiérrez, and Recio (1995), Gutiérrez, Recio, and de Velasco (1988), Kozen and Landau (1989), von zur Gathen (1990), and Zippel (1991).

In Corless, Giesbrecht, Jeffrey, and Watt (1999), the problem of the functional decomposition of approximate polynomials is examined. That is, for a given polynomial $f \in \mathbb{C}[x]$, polynomials $g, h \in \mathbb{C}[x]$ are sought such that there exists a "small" f_Δ with $f(x) + f_\Delta(x) = g(h(x))$. Here "small" is measured by the coefficient 2-norm: for

$$u = \sum_{0\le i\le m} u_i x^i \in \mathbb{C}[x], \text{ the coefficient 2-norm is defined by } \|u\|^2 = \sum_{0\le i\le m} u_i \overline{u_i},$$

where $\overline{u_i}$ is the complex conjugate of u_i.

In Section 1 of this paper we reconsider the original algorithm of Barton and Zippel (1976) for decomposing a polynomial $f \in \mathbb{C}[x]$. Their algorithm is based upon the fact that, for any right composition factor h of f, $h(x) - h(y)$ divides $f(x) - f(y)$. We show that, except for a particular easily handled class of polynomials, their algorithm in fact runs in polynomial time. In Section 2 we exhibit a reduction from the problem of decomposition to that of (structured) bivariate factoring to look at approximate polynomial decomposition. We demonstrate algorithms to provide rigorous lower bounds on the distance to an indecomposable polynomials (using the techniques of Kaltofen and May (2003)). In Section 3, we show how to use recent approximate bivariate factoring algorithms for polynomials to compute approximate decompositions of polynomials. Finally, in Section 4, we perform an empirical analysis on our new algorithm as compared to the techniques of Corless et al. (1999).

2. Barton and Zippel's algorithm revisited

Given a polynomial $f \in \mathsf{K}[x]$, over a field K, we first consider the problem of determining if f can be functionally decomposed in $\mathsf{K}[x]$, that is, determining if there exist $g, h \in \mathsf{K}[x]$, of degrees r, $s \geq 2$ respectively (with $n = r \cdot s$), such that $f(x) = g(h(x)) = (g \circ h)(x)$. The polynomial h is called a right composition factor of f. If f does decompose, we compute a decomposition.

The first known algorithm to compute the functional decomposition of a polynomial was given in Barton and Zippel (1976), and relied upon the following theorem of Fried and MacRae:

Fact 2.1 (Fried and MacRae, 1969). *Let x, y be independent indeterminates over a field K and $f, h \in \mathsf{K}[x]$. Then $h(x) - h(y)$ divides $\Phi_f = (f(x) - f(y))/(x - y)$ if and only if $f(x) = g(h(x))$ for some $g \in \mathsf{K}[x]$.*

Polynomials of the form Φ_f are called separated polynomials, and for any polynomial $h \in \mathsf{K}[x]$, we write Φ_h for $(h(x) - h(y))/(x - y)$. The above fact leads directly to the decomposition algorithm from Barton and Zippel (1976, 1985):

Algorithm 2.2.

```
    INPUT: A polynomial $f \in \mathsf{K}[x]$.
    OUTPUT: $g, h \in \mathsf{K}$ such that $f = g \circ h$, or $f$ is indecomposable.
1. Form $\Phi_f = (f(x) - f(y))/(x - y)$;
2. Factor $\Phi_f$ completely over $\mathbb{C}[x, y]$; if $\Phi_f$ is irreducible, $f$ is indecomposable;
3. Examine all factors of $\Phi_f$, looking for factors of the form $\Phi_h = ((h(x)-h(y))/(x-y)$ for some $h \in \mathsf{K}[x]$.
4. If no factors of the form $\Phi_h$ exist then $f$ is indecomposable, otherwise, we have
   found a factor $h$ from which we can compute $g$;
```

Note that in Step 4, computing g from a given h is simply a matter of solving the system of equations

$$f - \sum_{i=0}^{r} g_i \, h^i = 0, \tag{2.1}$$

which are linear in $g_0, g_1, \ldots, g_r \in \mathsf{K}$, where $g = \sum_{0\le i\le r} g_i x^i$, and $r = \deg g = n/\deg h$.

The running times of Steps 1, 2, and 4 of Algorithm 2.2 are clearly polynomial in the degree of f. However, the possibility of having to check what could be exponentially many combinations of factors of Φ_f in Step 3 prevents the algorithm from having a running time which is polynomial in the degree.

We can avoid this exponential-time step, at least for *tame* polynomials as follows. All $f \in \mathsf{K}[x]$ are tame when the characteristic of K is zero. When the characteristic char K of K is $p > 0$, then a polynomial is tame if $p > \deg f$ (a more inclusive definition of tame polynomials is given in (Turnwald, 1995, Definition 4.1) and von zur Gathen (1990)). Of course, Barton and Zippel's (1985) algorithm works over any field, whereas we confine ourselves to the tame case in the remainder of this paper.

The following theorem of Turnwald (1995) is a refinement of Fried's (1970) breakthrough solution of "Schur's conjecture":

Fact 2.3 (Turnwald, 1995). *Let $f \in \mathsf{K}[x]$ be tame and indecomposable of degree $n > 1$. Suppose that n is not an odd prime and it is not the case that $f(x) = \alpha D_n(a, x+b) + \beta$ for $\alpha, \beta, a, b \in \mathsf{K}$, where $a = 0$ if $n = 3$. If $f(x)$ is indecomposable, then $(f(x) - f(y))/(x-y)$ is absolutely irreducible.*

The notation $D_n(a, x)$ refers to a Dickson polynomial. The Dickson polynomials can be defined as the compositions of Chebyshev polynomials, linear polynomials, and polynomials of the form x^m, or more concretely as

$$\begin{aligned} D_n(x,a) = &\sum_{i=0}^{\lfloor n/2\rfloor} \frac{n}{n-i}\binom{n-i}{i}(-a)^i x^{n-2i} \\ =&x^n - nax^{n-2} + n(n-3)/2 \cdot a^2 x^{n-4} + \cdots \end{aligned}$$

The implications of Fact 2.3 for decomposition are that if f is not of prime degree and Φ_f factors in $\mathsf{K}[x,y]$ then f decomposes. Also, if h is an indecomposable composition factor of f then Φ_h is irreducible in $\mathsf{K}[x,y]$ unless h is a Dickson polynomial. Thus, searching combinations of factors of Φ_f is not necessary unless f has only Dickson polynomials as right composition factors. We show that this latter case is not difficult to handle, as it turns out that we can detect if a polynomial f has Dickson polynomials as right composition factors simply by examining the three highest-order coefficients of f.

Lemma 2.4. *Suppose $f \in \mathsf{K}[x]$ is monic and has a right composition factor of prime degree $q \ge 3$ which is a Dickson polynomial $D_q(a, x+b)$. Then f has the form:*

$$f = x^n + nb\,x^{n-1} + \left(\frac{n(n-1)}{2}\,b^2 - n\,a\right)x^{n-2} + \cdots$$

Proof. If $f \in \mathsf{K}[x]$ is monic and $f = g \circ D_q(a, x+b)$ then g is monic as well (since D_q is monic). If $q \ge 3$ is prime, then

$$D_q(a, x+b) = x^q + q\,b\,x^{q-1} + \left(\frac{q(q-1)}{2}b^2 - q\,a\right)x^{q-2} + \cdots,$$

so

$$g(D_q(a, x+b)) = \left(x^q + q\,b\,x^{q-1} + \left(\frac{q(q-1)}{2}b^2 - q\,a\right)\,x^{q-2} + \cdots\right)^k + \cdots$$

$$= x^{q\,k} + k\,(q\,b\,x^{q-1})\,(x^q)^{k-1} + \frac{k(k-1)}{2}\,(q\,b\,x^{q-1})^2\,(x^q)^{k-2}$$
$$+ k\,\left(\frac{q(q-1)}{2}b^2 - q\,a\right)\,x^{q-2}\,(x^q)^{k-1} + \cdots$$
$$= x^n + nb\,x^{n-1} + \left(\frac{n(n-1)}{2}\,b^2 - n\,a\right)x^{n-2} + \cdots$$

□

This yields the following improved version of Barton and Zippel's algorithm.

Algorithm 2.5.

INPUT: A tame polynomial $f = x^n + f_{n-1}\,x^{n-1} + f_{n-2}\,x^{n-2} + \ldots \in \mathsf{K}[x]$.
OUTPUT: $g, h \in \mathsf{K}$ such that $f = g \circ h$, or a message that f is indecomposable.

1. Form $\Phi_f = (f(x) - f(y))/(x - y)$;
2. Factor Φ_f completely in $\mathsf{K}[x, y]$; if Φ_f is irreducible, f is indecomposable;
3. Attempt to find an irreducible factor of Φ_f of the form Φ_h for some $h \in \mathsf{K}[x]$;
4. If such an h was found in Step 3, compute g from the system (2.1) so that $f(x) = g(h(x))$;
5. If no such h was found in Step 3, for each prime number q which divides $n = \deg f$, determine if $h(x) = D_q(a, x+b)$ is a right composition factor of f, where

$$b = f_{n-1}/n, \quad a = \frac{1}{n}\left(\frac{n(n-1)}{2}\,(f_{n-1}/n)^2 - f_{n-2}\right);$$

   This can be done by attempting to compute a left composition factor g by solving the system (2.1), and testing that $f(x) = g(h(x))$.

Algorithm 2.5 works for similar reasons to Barton & Zippel's, except now we know that the Φ_h corresponding to a non-Dickson factor must be irreducible by Fact 2.3 (and Dickson factors are dealt with in Step 5). It clearly requires only a polynomial number of operations since the factor combination of Algorithm 2.2 has been eliminated and Step 5 involves trying fewer than n possibilities for q.

While Algorithm 2.5 is interesting as a theoretical and historical artifact, in that a slight modification of the first algorithm of Barton and Zippel in fact runs in polynomial-time, it is really not competitive for exact decomposition of polynomials. A faster polynomial time algorithm from Kozen and Landau (1989), and a nearly linear time algorithm in von zur Gathen (1990), have been known for over 15 years. However, Algorithm 2.5 lends itself well to approximate decomposition, as we shall see in the next section.

3. Approximate Decomposability Testing

The relationship between decomposition and bivariate factorization presented in the previous section make it straightforward to transform decomposition into a linear problem. This provides a useful approach to approximate decomposition using recent approximate bivariate decomposition and irreducibility testing algorithms.

For a given $f = \sum_{i=0}^{n} f_i\, x^i \in \mathbb{R}[x]$ (with n not prime), we know f decomposes if and only if

$$\Phi_f = \frac{f(x) - f(y)}{x - y} = \sum_{i=1}^{n} f_i \left(\sum_{j=0}^{i-1} x^{i-j-1} y^j \right) \tag{3.1}$$

factors in $\mathbb{C}[x, y]$. Kaltofen and May (2003) provide a method for computing a lower bound on how far an irreducible bivariate polynomial is from a reducible polynomial. Their method employs (and extends) the linearization of Ruppert (1999). Stated in a somewhat non-standard way, Ruppert shows that for any n, m positive integers, there exist matrices $R_{ij} \in \mathbb{Z}^{4mn \times 2mn+n-1}$ for $0 \le i < m$ and $0 \le j < n$, with the following properties. Let

$$w = \sum_{0 \le i \le n} \sum_{0 \le j \le m} w_{ij} x^i y^j \in \mathbb{R}[x, y].$$

Then

$$\mathrm{Rup}(w) = \sum_{0 \le i \le n} \sum_{0 \le j \le m} w_{ij} R_{ij} \in \mathbb{R}^{4mn \times 2mn+n-1} \tag{3.2}$$

has full rank if and only if w is absolutely irreducible. Kaltofen and May (2003) show that if w is irreducible and $\tilde{w}$ does not have degree greater than w in either variable, then

$$\|w - \tilde{w}\|_2 < \frac{\sigma(\mathrm{Rup}(w))}{\max\{m, n\}\sqrt{2mn - n}}$$

implies that $\tilde{w}$ is irreducible, where $\sigma(\mathrm{Rup}(w))$ is the smallest singular value of $\mathrm{Rup}(w)$. They also show that $\|R_{ij}\| < \max\{n, m\}$. Note that Kaltofen and May (2003)'s result is stronger than an immediate application of Ruppert's theorem as it allows the degree of $\tilde{w}$ to be smaller than that of w.

Returning to the decomposition problem, Φ_f has factors if and only if the matrix $\mathrm{Rup}(\Phi_f)$ does not have full rank. Thus, we can bound the distance of an indecomposable f to a decomposable polynomial by bounding the distance of $\mathrm{Rup}(\Phi_f)$ to a matrix of lower rank, as done in Kaltofen and May (2003) through the singular value decomposition.

Theorem 3.1. *If $f \in \mathbb{R}[x]$ is an indecomposable polynomial, and $\tilde{f} \in \mathbb{R}[x]$ is a decomposable polynomial with $\tilde{f}(0) = f(0)$ and $\deg \tilde{f} \le \deg f$ then*

$$\|f - \tilde{f}\|_2 \ge \frac{\sigma(\mathrm{Rup}(\Phi_f))}{n^2\sqrt{2n^2 - n}}.$$

Proof. Since Φ_f is irreducible and $\Phi_{\tilde{f}}$ is not, and $\deg_x \Phi_f = \deg_y \Phi_f = d$ we have, from Theorem 1 of Kaltofen and May (2003),

$$\|\Phi_f - \Phi_{\tilde{f}}\|_2 \geq \frac{\sigma(\mathrm{Rup}(\Phi_f))}{n\sqrt{2n^2 - n}}.$$

Looking at (3.1) it is easy to see that:

$$\|\Phi_f - \Phi_{\tilde{f}}\|_2 = \|\Phi_{f-\tilde{f}}\|_2 \leq n\,\|(f - \tilde{f}) - (f - \tilde{f} \bmod x)\|_2 \leq n\,\|f - \tilde{f}\|_2.$$

Thus

$$\|f - \tilde{f}\|_2 \geq \frac{\sigma(\mathrm{Rup}(\Phi_f))}{n^2\sqrt{2n^2 - n}}.$$

□

We now have the ability to compute a radius of indecomposability about any indecomposable polynomial and, as with irreducibility, this gives a simple algorithm to test the indecomposability of an approximate polynomial when a tolerance on the coefficients is specified.

Note that, as with factorization, if we omit the degree bound, it is possible to find a decomposable polynomial which is arbitrarily close to f, namely $f \circ (\epsilon x^2 + x)$. It may be that the degree bound in Theorem 3.1) is too tight; approximate decompositions up to degree $2 \deg f - 1$ may be meaningful. However, we will consider only approximate decompositions of the same degree.

Example 3.2. We begin with a decomposable monic polynomial with a large noise term added to it (making it indecomposable):

$$f = (x^2 - x) \circ (x^2 + 3\,x) + .02\,x^3 = x^4 + 6.02\,x^3 + 8\,x^2 - 3\,x.$$

Then we compute

$$\Phi_f = x^3 + x^2y + xy^2 + y^3 + 6.02\,(x^2 + xy + y^2) + 8.0\,(x + y) - 3.0,$$

and the matrix $\mathrm{Rup}(\Phi_f)$ which is $27{\times}20$. Computing the largest coefficient of $\|\mathrm{Rup}(\Phi)\|^2$ (where Φ is a polynomial with symbolic coefficients of the same form as Φ_f) we get 200 (compared to the bound $d^2\,(2d^2 - d) = 7168$ from Theorem 3.1). Computing the smallest singular value of $\mathrm{Rup}(\Phi_f)$, we find a lower bound on the distance from f to the nearest polynomial which decomposes (in the 2-norm) is 4.32207×10^{-5}. Thus, any polynomial with constant coefficient 0 which is closer than the bound must also be indecomposable.

If we want to compute a better bound to separate f from decomposable monic polynomials, we can consider the largest coefficient of $\|\mathrm{Rup}(\Phi)\|^2$ after substituting 0 for the symbol corresponding to coefficient of the highest total degree terms. In that case we get 100 which leads to a slightly larger bound of 5.10581×10^{-5}. □

4. Approximate Decomposition

In this section we describe how to use the established connection between the decomposability of f and the irreducibility of Φ_f as a basis for decomposition algorithms. For many polynomials it is still probably best to use one of the algorithms described in Corless, Giesbrecht, Jeffrey, and Watt (1999). However, if those algorithms perform badly, we would like an alternate approach, which in some sense has a firmer theoretical underpinning. Ultimately neither of these algorithms provides a guaranteed solution in all cases.

Approximate decomposition using approximate bivariate factorization

We can build an algorithm to compute approximate decompositions of polynomials using the reduction to approximate bivariate polynomial factorization. The straightforward application of the approximate factoring algorithm described in Gao, Kaltofen, May, Yang, and Zhi (2004) to Φ_f creates an approximate version of Algorithm 2.5, which we explore now.

We first note that an algorithm which finds the nearest separated reducible polynomial $\Phi_{\tilde{f}} \in \mathbb{R}[x, y]$, of the form $(\tilde{f}(x) - \tilde{f}(y))/(x - y)$, for some $\tilde{f} \in \mathbb{R}[x]$, would actually find the nearest decomposable polynomial $\tilde{f}$ to f. However, this is the problem of looking for nearby reducible *structured* (i.e., separated) polynomials, which we will address in the next subsection. In this subsection we work to reconstruct structured factors from the factorization of a nearby unstructured polynomial.

Algorithm 4.1.

```
INPUT: An indecomposable polynomial $f \in \mathbb{R}[x]$ of degree $n$.
OUTPUT: $g, h \in \mathbb{R}[x]$ such that $f \approx g \circ h$.
1. Form $\Phi_f = (f(x) - f(y))/(x - y)$;
2. Compute an approximate factorization of $\Phi_f$ over $\mathbb{C}[x, y]$; discard all factors $p$ such
   that $\mathrm{tdeg}\, p + 1$ does not divide $\deg f$ – if no factors remain, skip to Step 5;
3. For each remaining factor compute its distance to a factor of the form $\Phi_h$;
    4.1 For all terms of the same total degree, compute the standard deviation of their
        coefficients;
    4.2 Find the maximum of the deviations over all sets of terms, and set this as the
        distance to a separated factor (see (3.1));
5. Choose the factor $\Upsilon$ of $\Phi$ with the smallest distance to a separated polynomial, and
   form $h = \sum_{i=1}^{n} f_i x^i$ where $f_i$ is the average of the coefficients of the terms of $\Upsilon$
   with total degree $i - 1$; use $h$ to compute a least squares solution $g$ to the system
   (2.1), so that $\|f - g \circ h\|$ is minimized;
6. Find $a$ and $b$ as in Algorithm 2.5 Step 5, and compute a corresponding $g$ by least
   squares for each possible choice of $q$;
7. Improve each approximate decomposition with Gauss-Newton iteration as in Corless
   et al. (1999);
8. Return the $g$, $h$ pair with the smallest value of $\|f - g \circ h\|_2$;
```

In practice, if f is a slightly perturbed decomposable polynomial then the approximate factors of Φ_f tend to contain polynomials very close to the form Φ_h. However, there is no guarantee on how close the approximate factors will come to having this form.

If one finds that an approximate factorization of Φ_f does not have any factors of the correct degree, it is possible to modify the approximate GCD algorithm used in the approximate factorization to produce factors of a predetermined degree (by choosing what the numerical rank of the Sylvester matrix will be, rather than trying to compute what it should be). In this way one can always find an approximate decomposition without a Dickson polynomial as a right decomposition factor though the backward error may be quite bad in some cases.

In Step 6 we suggest a simple least squares heuristic to compute the nearest polynomial with a right Dickson factor. In fact, since the Dickson polynomials are defined by only two parameters ($a, b \in \mathbb{R}$), it is easy to show that the absolutely nearest polynomial to f with a Dickson right factor can be found by minimizing a bivariate rational function of degree $O((\deg f)^4)$. This can be solved in polynomial time in $\deg f$ and $\log \|f\|$ by exact methods (see, e.g., Renegar 1992). While we make no claim that this method is at all practical, it is interesting to note that it exists.

In the following example, Step 6 of the algorithm is omitted.

Example 4.2. Beginning with the same polynomial as in example 3.2

$$f = (x^2 - x) \circ (x^2 + 3\,x) + .02\,x^3,$$

we feed Φ_f into an approximate factorization algorithm and get the following factorization:

$$\begin{aligned}\Phi_f \approx\ & (4.9047250 + 1.6439030\,x + 1.6439030\,y)\\ & \cdot (-0.61172559 + 1.836216\,x + 1.836216\,y + 0.6115972\,x^2\\ & \qquad - 0.003454426\,xy + 0.6115972\,y^2)\end{aligned}$$

The first factor is closer to the form Φ_h than the second, and the best fit h is

$$h(x) = 3.00099703989216\,x + x^2,$$

which we made monic to give a neater decomposition. Using least squares to solve for the best corresponding monic g, we get

$$g = -1.00029893310899\,x + x^2.$$

Composing, we obtain

$$g(h(x)) = -3.00189413726736\,x + 8.00568430033252\,x^2 + 6.00199407978432\,x^3 + x^4,$$

which has distance 0.0189766 from the original f. □

As with factoring, if the smallest singular value of $\mathrm{Rup}(\Phi_f)$ is quite large then it may be trivial to find a closer decomposition than the one produced by the algorithm. Most trivial approximate decompositions have relative backward error of about 1. For example, setting the coefficients of all the odd power terms to 0 will given a polynomial which has a right composition factor of x^2. For a randomly generated polynomial, not close to one that decomposes, this may be the best we can do.

Approximate decomposition by searching the Ruppert structure manifold

Because we are searching for factors of separated polynomials, we can conduct a more focused search on the Ruppert structure manifold than is possible for Gao et al. (2004). We saw in (3.2) that the (absolute) irreducibility of

$$\Phi_f = \sum_{0\le i,j\le n} \phi_{ij} x^i y^j \in \mathbb{R}[x,y], \qquad \text{for } f = \sum_{1\le i\le n} f_i x^i \in \mathbb{R}[x]$$

is indicated by the rank deficiency of

$$R_f = \sum_{0\le i,j\le n} \phi_{ij} R_{ij},$$

for some $R_{ij} \in \mathbb{Z}^{4n^2\times 2n^2+n-1}$ dependent only upon $\deg_x \Phi_f$ and $\deg_y \Phi_f$. Since

$$\Phi_f = \sum_{1\le i\le n} f_i \left(\sum_{0\le j\le i} x^j y^{i-j} \right),$$

Φ_f is absolutely irreducible if and only if

$$R_f = \sum_{1\le i\le n} f_i \left(\sum_{0\le j\le i} R_{j,i-j} \right)$$

is rank deficient. Thus, finding the nearest decomposable polynomial $\tilde{f} = \sum_{1\le i\le n} \tilde{f}_i x^i$ to f corresponds exactly to finding the nearest vector $(\tilde{f}_1, \ldots, \tilde{f}_n)$ to $(f_1, \ldots, f_n)$ such that

$$\sum_{0\le i\le n} \tilde{f}_i T_i, \quad \text{where } T_i = \sum_{0\le j\le i} R_{j,i-j} \text{ for } 1 \le i \le n,$$

is rank deficient. This problem is an instance of the Structured Total Least Squares (STLS) problem, for which there are a number of established (albeit heuristic) algorithms. See, e.g., De Moor (1994), Lemmerling (1999). We examine a number of approximate polynomial problems and their formulations as STLS problems, as well as a particular heuristic (the Riemannian SVD), in Botting et al. (2005). We summarize some of the results for the approximate decomposition problem in the next section.

5. Numerical Experiments

In this section we compare our algorithm empirically with the algorithms presented in Corless, Giesbrecht, Jeffrey, and Watt (1999). They present two algorithms, both of which are numerical iterations starting from a potential right composition factor obtained by running the exact decomposition algorithm of Kozen and Landau (1989). This will not generally be a right composition factor, but they demonstrate it is often a good initial approximation. In particular, the starting point closely matches the high order coefficients of f, while ignoring the lower order coefficients until the iteration, which may well be wise if these high order coefficients are dominant, as is often the case. Their first algorithm is a fast (linear-time), linearly convergent method, while the second is a standard, and relatively expensive but quadratically convergent, Newton iteration.

In our tests we consider only decompositions of monic polynomials into monic polynomials since we want in some sense to have representative degrees for the composition factors. We want to avoid approximate decompositions in which the leading coefficients of the composition factors becomes very small when compared to the other coefficients, i.e., in which the numerical degree is not representative of the actual degree. We note that in fact the algorithm of Corless et al. (1999) will often produce such decompositions unless explicitly constrained not to.

We conduct tests on the method for estimating a radius of indecomposability, as described in Section 3. The lower bounds were determined for monic polynomials generated by choosing monic polynomials $g, h \in \mathbb{R}[x]$ with rational coefficients selected randomly and uniformly between -5 and 5, with 6 decimal digits (except the constant coefficient which is always 0), then composing them to form a monic $f \in \mathbb{R}[x]$. We then perturb each of the coefficients of f with uniformly distributed random noise of norm specified in the column labeled "$\|f_\triangle\|$" in the the table below (keeping the perturbed f monic with constant coefficient 0).

All results in the table are the median value of 100 runs, and are scaled by dividing through by the norm of the perturbed f. The sixth column is the median of the distance to the smallest polynomial that decomposes (computed by solving the optimization using Groebner basis methods and the SALSA package for Maple `http://fgbrs.lip6.fr/Software/`). It is interesting to note that the smallest singular value is usually quite close to that distance. This suggests that perhaps our lower bound is too small, and that is may be possible to prove a better lower bound, perhaps $\frac{1}{\sqrt{d}}\,\sigma(\mathrm{Rup}(\Phi_f))$ (or perhaps even a constant multiple of the smallest singular value).

deg g	deg h	$\|f_\triangle\|$	$\sigma(\mathrm{Rup}(\Phi_f))$	Lwr Bnd	Abs Min
2	2	10^{-3}	2.4028e-4	5.2687e-6	1.7212e-4
2	2	10^{-1}	1.7646e-2	3.8693e-4	1.3613e-2
3	2	10^{-5}	1.0934e-6	4.8793e-9	1.1643e-6
3	2	10^{-3}	9.3678e-5	4.1802e-7	1.1935e-4
3	2	10^{-1}	7.8925e-3	3.5219e-5	1.1875e-2
2	3	10^{-5}	2.7156e-6	1.2118e-8	1.7172e-6
2	3	10^{-1}	2.7610e-2	1.2321e-4	1.9851e-2

To compare the approximate factorization algorithms, we generated random monic g and h with integer coefficients in $[-10..10]$ then composed to form $f = \sum f_i x^i$ which we perturbed with $f_\triangle = \sum \delta_i x^i$ where $\delta_i/(f_i\,\epsilon) \in [-10, 10]$ and the δ_i have 5 decimal digits. This choice for $f_\triangle$ is made to compensate for the fact that some of the coefficients of composed polynomials are significantly larger than others.

In the following table, "CGJW" indicates the iterative algorithm from Corless et al. (1999), "AppFac" indicates the approximate factorization based decomposition algorithm, and "RiSVD" indicates the Riemannian SVD based method. "Error" is the median relative error of the decomposition found over 30 or 60 runs, "Best" is the number of times this method was better than both the others (the difference between the sums of the columns and 100% were two or three way ties). In some tests, indicated by a '*' in the "Best" column of "AppFac", the best decompositions produced by the approximate factorization method could have slightly larger degree than f (the degree was not bigger in all case however). For small examples "Abs" is the number of times this method found the absolute minimum, for larger examples the value is in italics and is the number of times the result of this method was the best or tied for the best.

			CGJW			AppFac			RiSVD		
deg g	deg h	ϵ	Error	Best	Abs	Error	Best	Abs	Error	Best	Abs
2	2	10^{-4}	1.53e-5	0%	100%	1.59e-5	0%	98%	1.53e-5	0%	98%
2	3	10^{-4}	1.15e-4	3%	100%	9.70e-3	0%	45%	1.26e-4	0%	97%
3	2	10^{-4}	1.90e-5	0%	96%	1.90e-5	0%	100%	1.90e-5	0%	100%
4	2	10^{-4}	1.85e-4	5%	97%	6.21e-2	2%	28%	2.20e-4	0%	88%
2	4	10^{-4}	2.99e-5	0%	98%	4.17e-5	0%	83%	2.99e-5	2%	98%
3	3	10^{-4}	1.67e-4	0%	*100%*	3.08e-4	0%	*60%*	1.67e-4	0%	*100%*
2	5	10^{-4}	4.18e-4	3%	*87%*	1.31e-1	10%	*30%*	4.18e-1	3%	*87%*
2	2	10^{-1}	8.74e-3	2%	95%	9.18e-3	2%	83%	9.52e-3	3%	87%
2	3	10^{-1}	5.24e-2	5%	70%	5.37e-2	2%	43%	5.08e-2	12%	80%
3	2	10^{-1}	1.54e-2	0%	85%	1.47e-2	8%	95%	1.54e-2	2%	87%
2	4	10^{-1}	1.21e-1	5%	50%	1.61e-1	8%*	25%	1.21e-1	7%	53%
4	2	10^{-1}	2.50e-2	2%	43%	2.28e-2	18%*	30%	2.50e-2	3%	40%
3	3	10^{-1}	1.25e-1	0%	*40%*	6.35e-2	60%*	*97%*	1.25e-1	0%	*40%*
2	5	10^{-1}	2.04e-1	0%	*87%*	2.13e-1	13%*	*60%*	2.04e-1	0%	*87%*
5	2	10^{-1}	2.99e-2	0%	*90%*	2.99e-2	10%*	*83%*	2.99e-2	0%	*90%*
2	6	10^{-1}	2.40e-1	0%	*80%*	2.28e-1	17%*	*83%*	2.40e-1	3%	*83%*
3	4	10^{-1}	1.87e-1	0%	*63%*	1.34e-1	37%*	*93%*	1.87e-1	0%	*63%*
4	3	10^{-1}	8.20e-2	0%	*83%*	7.69e-2	10%*	*80%*	7.54e-2	7%	*90%*
6	2	10^{-1}	2.42e-2	0%	*93%*	4.63e-2	7%	*63%*	2.42e-2	0%	*93%*
2	2	1	2.40e-2	0%	95%	2.40e-2	0%	93%	2.40e-2	0%	95%
2	3	1	1.83e-1	2%	75%	1.74e-1	7%*	70%	1.79e-1	7%	75%
3	2	1	1.62e-1	0%	77%	1.51e-1	7%*	77%	1.44e-1	8%	87%
2	4	1	3.78e-1	2%	50%	3.45e-1	30%*	28%	3.70e-1	8%	55%
4	2	1	2.87e-1	8%	58%	2.63e-1	22%*	40%	2.89e-1	5%	60%
3	3	1	4.98e-1	0%	*40%*	3.08e-1	43%*	*93%*	4.69e-1	7%	*57%*
2	5	1	4.02e-1	7%	*53%*	3.58e-1	33%*	*70%*	4.09e-1	13%	*57%*
5	2	1	3.80e-1	10%	*60%*	3.43e-1	20%*	*80%*	3.80e-1	0%	*70%*
2	6	1	4.53e-1	7%	*60%*	4.44e-1	23%*	*73%*	4.49e-1	17%	*70%*
3	4	1	4.54e-1	17%	*57%*	4.34e-1	27%*	*53%*	4.43e-1	17%	*57%*
4	3	1	4.13e-1	3%	*60%*	3.54e-1	33%*	*83%*	3.98e-1	7%	*63%*

On average, CGJW is about 10 times faster than RiSVD which is about ten times faster than AppFac. All three algorithms find the same answer in most of the examples. Occasionally, when ϵ is relatively small, CGJW produces a very bad decomposition, while the AppFac or RiSVD still produce good results. On the other hand, when ϵ is large, the approximate factorization may not find a factor of the correct degree leading to a worse result than the other two algorithms. The approximate factorization based decomposition algorithm is also significantly slower than the other two. This suggests that in practice one should probably used the Gauss-Newton iteration algorithm from Corless et al. (1999), and revert first to RiSVD and failing that to AppFac if the backwards error of the result is not about the same order of magnitude as $\sigma(\mathrm{Rup}(\Phi_f))$.

Acknowledgment

We would like to thank Erich Kaltofen for early important discussions related to this work, and for later comments. The first author would like to thank the NSERC and MITACS. Some of this work was done while the authors were visiting the Key Laboratory of Mathematics-Mechanization, Institute of Systems Science, Chinese Academy of Sciences in Beijing. We thank them for their generous support.

References

C. Alonso, J. Gutiérrez, and T. Recio. A rational function decomposition algorithm by near-separated polynomials. *J. Symb. Comp*, 19(6):527–544, 1995.

D. R. Barton and R. Zippel. A polynomial decomposition algorithm. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 356–358, New York, NY, USA, 1976. ACM Press.

D.R. Barton and R. Zippel. Polynomial decomposition algorithms. *J. Symbolic Comput.*, 1:159–168, 1985.

B. Botting, M. Giesbrecht, and J.P. May. Using the Riemannian SVD for problems in approximate algebra. In *Proceedings of the International Workshop of Symbolic-Numeric Computation*, pages 209–219, Xi'an, China, 2005.

R.M. Corless, M. Giesbrecht, D. Jeffrey, and S.M. Watt. Approximate polynomial decomposition. In *Internat. Symp. Symbolic Algebraic Comput.*, pages 213–220. ACM Press, 1999.

B. De Moor. Total least squares for affinely structured matrices and the noisy realization problem. *IEEE Transactions on Signal Processing*, 42:3004–3113, 1994.

M.D. Fried. On a conjecture of Schur. *Mich. Math. Journal*, 17:41–55, 1970.

M.D. Fried and R.E. MacRae. On the invariance of chains of fields. *Ill. J. Math.*, 13: 165–171, 1969.

S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In *ISSAC 2004 Proc. 2004 Internat. Symp. Symbolic Algebraic Comput.*, pages 167–174, 2004.

J. von zur Gathen. Functional decomposition of polynomials: the tame case. *J. Symbolic Comput.*, 9:281–299, 1990.

J. Gutiérrez, T. Recio, and C. Ruiz de Velasco. Polynomial decomposition algorithm of almost quadratic complexity. In *Proc. Applied algebra, algebraic algorithms and error-correcting codes*, volume 357 of *Lecture Notes in Comp. Sci.*, pages 471–475, Rome, 1988. Springer, Berlin.

E. Kaltofen and J. May. On approximate irreducibility of polynomials in several variables. In *ISSAC 2003 Proc. 2003 Internat. Symp. Symbolic Algebraic Comput.*, pages 161–168, 2003.

D. Kozen and S. Landau. Polynomial decomposition algorithms. *J. Symbolic Comput.*, 7: 445–456, 1989.

P. Lemmerling. *Structured total least squares: analysis, algorithms and applications*. PhD thesis, K.U. Leuven (Leuven, Belgium), 1999.

J. Renegar. On the computational complexit and geometry of the first-order theory of the reals: parts I, II, III. *J. Symb. Comp*, 1992.

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

G. Turnwald. On Schur's conjecture. *J. Austral. Math. Soc. Ser. A*, 58(3):312–357, 1995.

R. Zippel. Rational function decomposition. In *Proc. ISSAC'91*, pages 1–6. ACM Press, 1991.

Mark Giesbrecht
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
e-mail: `mwg@uwaterloo.ca`

John May
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
e-mail: `jpmay@uwaterloo.ca`