A Geometric Approach to Factoring Bivariate Approximate Polynomials

Andre Galligo*
Universite de Nice (and INRIA)
Laboratoire de Mathematiques
Parc Valrose 06108 Nice cedex 02, France
galligo@unice.fr

Mark van Hoeij†
Department of Mathematics
Florida State University
Tallahassee, FL 32306-4510, USA
hoeij@math.fsu.edu

ABSTRACT

A geometric approach is presented for designing algorithms to factor bivariate approximate polynomials over $\mathbb{C}[x, y]$. Given a perturbed composite polynomial, stably square-free, satisfying a genericity hypothesis (one that occurs often in applications) two algorithms are described that construct a nearby composite polynomial and its irreducible factors. The time to produce the factors is polynomial in the degree of the problem. A preliminary implementation in Maple is illustrated by several examples with comments on efficiency and numerical robustness. We also discuss on the possibility to get rid of the hypothesis.

Categories and Subject Descriptors

J.2 [**Mathematics**]; I.1.2 [**Computing methodologies**]: Symbolic and Algebraic Manipulation—*Algebraic Algorithms*

General Terms

Algorithms,Theory

Keywords

Approximate Factorization, Algebraic Geometry, Algorithms, Maple Code

1. INTRODUCTION

1.1 The problem

Over the past ten years symbolic-numeric algorithms for approximate polynomials (computation of greatest common divisors, functional decompositions, find zeros of multivariate systems, test primality, factorization) have been studied by many authors (see e.g. the proceedings of the SNC conferences and the references inside). The main common feature of the produced algorithms is to propose a strategy to find a nearby object with required properties. This can be often re-interpreted as a reverse engineering task: recognize a perturbed situation by a "small" distortion of a representation. However, the size of this small distortion is usually not easy to formalize and many authors present the efficiency of their algorithms by their effectiveness on some benchmarks.

An important problem of this family which meets a renewed interest is the factoring of approximate multivariate polynomials, and the bivariate case captures its essential issues. There exist several methods, algorithms and implementations for factoring multivariate polynomials in an exact setting and most have a polynomial complexity. See e.g. [2] or [10] or [4] and their bibliography. Here we are interested in the corresponding approximate problem (see e.g. [16]) which can be stated as follows (for two factors):

Given $f \in \mathbb{C}[x, y]$ of total degree n, and $\varepsilon \in \mathbb{R}$, find g and $h \in \mathbb{C}[x, y]$ such that $g \times h = f + \Delta f$, where $\deg g, \deg h > 0, ||\Delta f|| < \varepsilon$ and $||\cdot||$ denotes an appropriate polynomial norm.

This problem has been already addressed by several authors as indicated below. The proposed methods find factors of an approximate polynomial if the given polynomial is "sufficiently close" to being factorable.

Let us assume, in the sequel, that f is "approximately square-free", that is, all polynomials near f are square-free. This can be determined in practice by taking a random complex floating-point value for x and using a fast algorithm to certify that the approximate GCD of $f(x, y)$ and $f_y(x, y)$ is 1. If this is not the case, then an algorithm to remove the multivariate approximate GCD such as described e.g. in [19, 26] could be used.

1.2 Some previous works

For a history of early algorithms see [14], [15]. The paper [1] is the first algorithmic paper using monodromy group action (as in section 2 below). The paper [9] considers point combinations, an exponential search, while the paper [12] uses an optimization method exponential in the degree of the factor recovered. The paper [13], is also of complexity exponential in the degree of the input. The papers [22, 21, 20] discuss another interesting algorithm based on zero-sum identities of power-series solutions of $f(x, y) = 0$. This algorithm is numerically stable and of polynomial complexity. The paper [5] presented a polynomial-time method and adopted a backward error analysis point of view. Similarly for the papers [17], [11] which applied Svd on the Ruppert-

*Supported by french ANR grant "GECKO"
†Supported by NSF grant 0511544

ISSAC '07 Waterloo, Canada

Gao matrix.

In this paper we focus on a geometric variation of the cited problem: control the size of the distance between f and $f + \Delta f$ not only via their coefficients but also via discrete geometric data, namely discriminant and critical loci and the monodromy group attached to the polynomial (see section 2). This point of view can be related to the computation of the approximate GCD of two univariate polynomials f and g by pairing nearby (controlled by a distance) roots of f and g which are merged. Here the situation is more involved as the zero-sets are curves. We chose to rely on a finite set of characteristic points: the critical locus of the projection of the associated curve on the x-axis. Indeed this characterizes an algebraic curve for a given degree $n > 2$.

Denote by X the (possibly singular) curve $X = f^{-1}(0) \subset \mathbb{C}^2$, by p its projection on the x-axis $p : X \to \mathbb{C}$ and choose a generic (i.e. random) fiber $E = p^{-1}(a)$, which has n points. We denote by $\Delta \in \mathbb{C}$ the discriminant locus of p. The action of the fundamental group $\pi(\mathbb{C} - \Delta)$ on E defines the monodromy group G, and can be explicitly calculated; the algcurves package in Maple contains a monodromy program that we will use and extend. For details on this program see section 2.

When f is irreducible, G acts transitively on E, while when f is composite: $f = f_1 \cdots f_s$, the orbits of G provide the s-partition of E by the subsets formed by the roots of the factors f_i. This is the key combinatorial information which allows to recover the factorization of f by a continuation method. See e.g. [6, 24, 2]. In the exact setting, an early polynomial time algorithm for computing the absolute factorization of a bivariate polynomial using the monodromy was presented by [1], monodromy plays also an important role in the factorization algorithms presented in [9, 7, 19, 24, 25, 2, 3].

1.3 Our approach

Our work is still exploratory, we do not pretend to compete with other strategies and implementations. Our aim is to present another point of view of geometric nature which can complement previous approaches and show some other aspects of the hardness of the task. In this paper, we restrict our study to the following situation (which is the one encountered in many application and most benchmark examples): The polynomial F to be factored is a small perturbation of a product $f = f_1 \cdots f_s$ such that the curves $X_i = f_i^{-1}(0)$ are all smooth and intersect transversally in double points (nodes). The general idea is that the input equation F has been corrupted by some noise creating a relative error on the data of order α, estimated by the user, that we can use in our program to estimate some threshold; but we can do computations with rational numbers or bigfloats with a higher precision. As we can start with a "generic" change of coordinates, we can suppose that f and its perturbation F are monic in y of degree n and total degree n and that the projections of the critical points on the x-axis are all distinct, therefore we expect that the only singularities of the discriminant are double roots corresponding to the projections of intersections of two components.

The proposed factoring method is based on three main ideas which leads to two procedures that we called algorithms 1 and 2.

First, as the X_i are smooth and cut transversally, the discriminant points of f are either (simple) branching points (of one X_i) or double points (corresponding to projections of intersection points of two components X_i and X_j). By a (small) perturbation the double points give rise to a pair of nearby branching points which generate the same transposition of the fiber E at a base point, i.e. there is a path looping around only these two branching points which induces the identity on E. This property allows a recognition process that we call algorithm 1 (see section 3). We rely on the description of the generators of the monodromy group of the input (approximate) polynomial F given by the commands in the package algcurves, and we diminish the number of these generators iteratively (this corresponds geometrically to decreasing the genus of the curve) until the monodromy group does not act transitively on the fiber E.

Second, following the previous analysis, as long as we don't pass between any such cluster of branchpoints, the monodromy induced by F is similar to the monodromy induced by f, then we can rely on the observation of [23] that only several large loops suffice in general to determine the orbits of the monodromy. So we compute all expected clusters of two nearby approximate singular points of the curve defined by F (i.e. approximate solutions of a system of three equations in two unknowns). We project these clusters on the x-axis on clusters of points of the discriminant, and delimit zones that should not be crossed by the "large loops", so we adapt the cited strategy to our approximate setting to get an early detection of the partition of E.

We observed that a main task in this second method is to locate the clusters formed by the deformation of the nodes. This led us to the third idea that an important preprocessing for the approximate factorization can be achieved if we can locate "centers" for these clusters and "project" F on the subvariety (in a parameter space) of the polynomials with the same terms than F having singularities at these points. We implemented that idea as algorithm 2 (see section4).

Once the partition is determined, we construct the approximate factors using interpolation. There are several ways to do this, see section 6. Alternatively we can adapt the algorithms from [5] for continuations and interpolation.

We discuss briefly the case when X may admit higher singularities and suggest some direction of research to address these cases (section 5).

1.4 An illustrative example

We will use throughout the paper the following illustrative example: f_1 and f_2 are two random dense bivariate polynomial of degree 3, f is their product divided by 100, so that the coefficient are about 50 and F is a perturbation of f with a relative error of 10^{-2}.

```
f1 := 92*y^3+44*y^2*x+40*y*x^2-67*x^3
      +8*y^2+66*x*y+68*x^2-95*y-62*x-18:
f2 := 68*y^3+39*y^2*x+20*y*x^2+45*x^3
      -65*y^2-67*y*x+93*x^2+43*y+8*x+6:
f:=f1*f2/100:  F:=evalf(f,3):
```

The relative perturbation on the coefficients of the discriminant of f is about $5 * 10^{-2}$. The relative perturbation on its roots is about 10^{-1} for the double points and about 10^{-2} for the simple points.

2. MONODROMY GROUP

In this section, we briefly define our main tool, the monodromy group: it is represented and its calculation is implemented in the package algcurves of Maple that we will use. More details can be found in [6].

Let $f(x,y)$ be a polynomial of degree n in y, that we suppose monic in y of degree n, this hypothesis simplifies the presentation and is always satisfied after a generic (random) change of coordinates. Let $X = f^{-1}(0)$ be the defined curve in $\mathbb{C}^2$ and $p : X \to \mathbb{C}$ the projection on the x-axis. The discriminant locus Δ of f is the zero-set of $D := \mathrm{Res}_y(f, f'_y)$, it contains the simple branching points which are the projections of the points with a vertical tangent and the projections of the singularities of X (which are the solutions of $f = f'_y = f'_x = 0$).

To define the monodromy, first select a base point $x = a$ in the complex x-plane minus the discriminant locus. Let E be the fiber of p above a (i.e., the n distinct y-values for which $f(a,y) = 0$). These y-values are now assigned an order, $(y_1, y_2, \ldots, y_n)$. This ordering of the n y-values labels the sheets of the covering $X - p^{-1}(\Delta)$ of $\mathbb{C} - \Delta$.

For each point $b \in \Delta$, one chooses a path γ_b in the complex x-plane which starts and ends at $x = a$, encircles only $x = b$ counterclockwise and avoids all points of Δ. The n-tuple $(y_1, y_2, \ldots, y_n)$ is then analytically continued around this path γ_b. When one returns to $x = a$, a new n-tuple is found, which has the same entries as $(y_1, y_2, \ldots, y_n)$, but ordered differently: $(y_{\sigma_b(1)}, y_{\sigma_b(2)}, \ldots, y_{\sigma_b(n)})$, where σ_b is a permutation acting on the set of labels $\{1, 2, \ldots, n\}$. We will say that the permutation σ_b is attached to the path γ_b. Note that for different choices of γ_b, we obtain different permutations. We will see later, how they can be compared.

Here are some typical situations. If $x = b$ is a simple branching point, then σ_b is a transposition. If $x = b$ is the projection of a simple double point (a node), then σ_b is the identity. If $x = b$ is the projection of a cusp singularity like the one with local equation $x^2 - y^3 = 0$, then σ_b is the cyclic permutation of order 3.

For the effective calculation of the monodromy some labeling and ordering should be done. The Maple implementation (algcurves) that we will use (see [6]) and complete with new procedures, made the following choices.

1. **Base point and circles:** With every discriminant point b_i, a radius $r(b_i)$ is associated: (ρ denotes the distance)

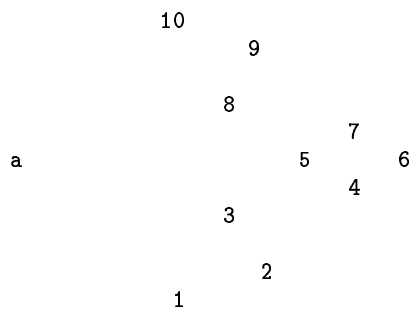
   $$r(b_i) = \frac{2}{5}\rho(b_i, \{b_1, b_2, \ldots, b_m\} - \{b_i\}),$$

   Let $C(b_i, r(b_i))$ denote the circle with center b_i and radius $r(b_i)$. Then the circles $C(b_i, r(b_i))$ do not intersect each other.

   Now a base point a is chosen, such that the real part of a is smaller than the real parts of any of the b_i. By this choice, the arguments of $b_i - a$ are between $-\pi/2$ and $\pi/2$.

2. **Labeling of the sheets:** At the base point $x = a$ there are n distinct finite y-values, they form the fiber E of p. These are determined numerically as the solutions of $f(a,y) = 0$. Let these n y-values be assigned an order $(y_1, y_2, \ldots, y_n)$, this labels the corresponding sheet of the covering $y(x)$ which contains y_i.

3. **Ordering of the discriminant points:** We order these points according to their argument with respect to the base point: if $\arg(b_i - a) < \arg(b_j - a)$, then b_i precedes b_j in the ordering, where $\arg(\cdot)$ denotes the argument function. If $\arg(b_i - a) = \arg(b_j - a)$, then b_i precedes b_j if $|b_i - a| < |b_j - a|$. For example:



4. **Choice of the paths:** The simplest path $L(b_i)$ around b_i consists of one line segment from a to $b_i - r(b_i)$. This is followed by $C(b_i, r(b_i))$, starting at $b_i - r(b_i)$. Successively, a line segment is followed from $b_i - r(b_i)$, back to a. The algorithm chooses a path that is composed of line segments and semi-circles, and that is equivalent to the simplest path.

5. **Analytic continuation:** The path γ_b is discretized in small segments $x_i x_{i+1}$ so that when a path is followed in the complex x-plane from x_1 to x_2, the n entries of $f^{-1}(x_1)$ i.e. the roots of $f(x_1, y) = 0$, follow paths on the Riemann surface to the roots of $f(x_2, y) = 0$, by a numerical integration of a vector field defined by derivatives of f (see [6] or [5] for more details) . This gives rise to an n-tuple, whose ordering is induced by the ordering of $f^{-1}(x_1)$. The accepted size of $|x_{i+1} - x_i|$ depends on the separation of the entries of $f^{-1}(x_i)$.

6. **Monodromy group:** Consider a closed path starting from $x = a$ and returning there after encircling one branch point $x = b$. After analytic continuation of E along this path, the entries of E are recovered, but they are shuffled by the permutation σ_b.

   The collection of all σ_b generates the monodromy group, which is represented here as a subgroup of S_n, the group of permutations of $\{1, 2, \ldots, n\}$. Note that this representation depends on the choice of the labeling of the y-values at $x = a$, so it is only unique up to conjugation.

   The point $x = \infty$ might also be a branch point. The corresponding permutation σ_∞ does not need to be computed by analytic continuation, since it can be determined from the other σ_b.

7. **Example:** We apply the procedure monodromy to the perturbation F of f in our illustrative example.

   ```
   with(algcurves):
   m := monodromy(convert(F,rational),x,y);
   ```

   The output is a list with three entries. The first entry m[1] is the basepoint $x = a$ that the algorithm chose. The second entry m[2] is the fiber above a (i.e. our list E), it consists of n complex numbers. The third entry m[3] is a list, each entry contains a branchpoint b_i with its permutation σ_{b_i} (given as a product of disjoint cycles).

3. ALGORITHM 1

In this section, given a polynomial F supposed a small perturbation of a composite polynomial f, and a "generic" base point a in $\mathbb{C}$, we propose an algorithm to recover the partition of the fiber of f above a. We denote by X the curve defined by f and by Y the "perturbed" curve defined by F. We first analyze the situation.

3.1 Effect of a perturbation

With the hypothesis on f described in the introduction, the discriminant Δ of f admits only a subset of simple roots s_1, and a subset of double roots s_2. By a small deformation F of f, each element of s_2 is deformed into a "cluster" of 2 roots of the discriminant Δ_p of F, while each element of s_1 deforms to a root of Δ_p. Our task is thus to separate in these two categories the elements of Δ_p as they all appear as simple roots of the discriminant of F. The ones in the first category come in pairs and form a set of clusters. So we will look for nearby points of Δ_p.

This is done as follows. The user specifies a level of distortion of the coefficients of f, that we use to estimate probabilistically a level of deformation for each type of roots, as follows. Let the level of relative approximation for the coefficients of f be α. Then the expectation for the level of relative error for D, β, is about $2n\alpha$. The expectation for the level for the simple roots is also about β, while the one for the double roots is about $\sqrt{\beta}$. In our illustrative example $\alpha = 5 * 10^{-2}$, so we roughly expect perturbations of order 10^{-2} for the points in s_1 and perturbation of order 10^{-1} for the points in s_2.

We note that each point in s_2 is the projection of a double point of X which is also deformed in a cluster of two critical points of the projection of Y. So we can improve the previous estimation of vicinity by considering not only the distance between two points of Δ_p but also the distance between the corresponding critical points in Y. They should be in the same level of magnitude.

As these estimation of levels are not precise, we will use them only as a guide for starting iterations.

A next observation is the following "conservation law". The monodromy (on any generic fiber of f) defined by a small circle looping only around a double point of Δ is the identity; because each path in the Riemann surface above the circles remains in the same layer (as for two crossing lines in $\mathbb{C}^2$). This feature is conserved by deformation: the monodromy defined by a path γ looping only around a pair of two roots of Δ_p, obtained by deformation of a double point of D, is the identity (the simplest example is a couple of crossing lines deformed in an hyperbola).

If we are able to recognize all the pairs of roots of Δ_p, obtained by deformation of an element of s_2, we can merge the two points of each such pair by creating "cuts" in the complex plane. Then we require that the paths γ used to calculate the monodromy do not cross these cuts. Once this is done, the remaining roots of Δ_p define (a large number of) permutations of the fiber E whose s orbits give the target partition of E. The classes of this partition are subsets of cardinal n_i with $\sum n_i = n$ and $\sum_{i<j} n_i n_j$ should be the number of merged elements.

Finally, together with the partition into layers, this process provides the coordinates (x, y) of many points in each layer. So we can interpolate each F_i, $1 \leq i \leq s$ (see next section 6 for more on this) and we expect that $F - F_1 \cdots F_s$ is small.

Here are some easy enumerative data. If f admits two factors of degree $n/2$, then there are $n^2/4$ double points in the discriminant. The extreme cases are when f admits a factor of degree 1 and one of degree $n-1$, then there are $n-1$ double points; while when f admits n factors of degree 1, there are $n(n-1)/2$ double points.

3.2 Recognition process

In order to detect the pairs of roots (p, q) to be merged, we could perform analytic continuation over the shortest path around p and q for every pair of roots of the discriminant. However, that would be $O(n^4)$ paths. To reduce the number of paths over which we perform analytic continuation, we order the points as described in the previous section 2, and for each point, compute a path around it and the basepoint. This requires $O(n^2)$ paths. Then we use two procedures given below to detect which pairs may be merged.

Take the triangle: basepoint - p - q. Consider all the points inside that triangle, in the order in which they appear in the output M of algcurves[monodromy] (see in section 2 the subsection on the sorting of discriminant points). The only points that could be inside that triangle have ordering between p and q, and must appear on the same side of L as the basepoint, where L is the line through p and q. Finding these points is done with this Maple procedure:

```
PointsInTriangle := proc(M,p,q)  local L,t,i;
  # {L | t in R} is the line through p and q:
  L := t * M[3,p,1] + (1-t) * M[3,q,1];
  # Now {L | t in C} = C.  Two points in C are on
  # the same side of line {L | t in R} iff the
  # imaginary parts of their corresponding t-values
  # are either both positive, or both negative.
  # This lists i's for which M[3,i,1] (point #i)
  # is on the same side as M[1] (the basepoint):
  [seq(`if`(Im(solve(L=M[1]    ,t)) *
            Im(solve(L=M[3,i,1],t)) > 0,
  i, NULL), i=p+1..q-1)]
end
```

Then we multiply the permutations corresponding to the points inside that triangle, to get a permutation g. Now, in order to compare the permutation of point p with that of point q, we first have to conjugate with g, and then multiply and check for identity. See the Maple code below for more details.

We start with the relatively closest pair of points with matching permutations. We make a cut in the complex plane joining the two points of the pair. Then we proceed iteratively, up to some threshold for the distance, we should replace a path crossing a cut (and producing a permutation γ) by a path avoiding that cut. However, we don't compute any new analytic continuations, we just calculate the appropriate conjugate of γ and obtain the same resulting permutation.

When the deformation is small enough, we should find the clusters corresponding to the intersections between the components.

We choose to start with a subset of pairs of roots of Δ_p for which the distance between the two points of the pairs is smaller than the threshold provided by the estimation described in the previous subsection. Then we apply the procedure MatchingPermutations to see which pairs should

be merged. Then we perform the merging and check if the monodromy action is no longer transitive.

Remark that if we remove more points and create more cuts than necessary, it is likely that there remain enough points and paths to recover the targeted partition of E (the fiber over the base point).

3.3 Maple procedures

Here are some Maple procedure that implement the previous approach.

"mp" computes the product of (several) permutations, "invperm" inverses a permutation and "orbit" computes the orbits of a subgroup of permutations.

With the previous notations, the following procedures returns true if the shortest path around points p,q has trivial permutation $= [\,]$.

```
MatchingPermutations := proc(M,p,q) local g,i;
g := [];
for i in PointsInTriangle(M,p,q) do
  # mp = multiply permutations
  g := mp(M[3,i,2], g)
od;
evalb([] = mp(invperm(g), M[3,q,2], g, M[3,p,2]))
end:
```

Then let us denote by S the set of pairs that we discard and by P the subset of discriminant points $1, 2, \ldots,$ nops(M[3]) that we keep.

```
S := NULL;
for p to nops(M[3])-1 do
  for q from p+1 to nops(M[3]) do
    d := abs(M[3,p,1]-M[3,q,1]);
    if d > 0.3 then next fi;
    if MatchingPermutations(M,p,q) then
      S := S, [p,q, d]
    fi
  od
od:
```

The number 0.3 was used as a threshold, if no such threshold is available then that line should be deleted. To check if the monodromy is transitive for a set of points P, we use the following procedure.

```
TransitiveGroup := proc(M, P) local G,i;
G := permgroup(n, {seq(M[3,i,2], i=P)});
# output true if the curve is still irreducible:
evalb(nops(orbit(G, 1)) = nops(M[2]))
end:
```

Instead of deleting all pairs with distance below a certain threshold, we can also work without a threshold and simply start deleting grouped branchpoints (starting with the relatively closest pair, then the next closest, and so on) and stop as soon as the group becomes reducible (i.e. not transitive). If the distance of the last pair was d, then we see that merging only pairs with distance $< d$ does not suffice to reach a reducible polynomial. One can now try to compute a number δ such that no δ-perturbation of F can bring a pair of distance d together. Such δ would then be a lower bound for the distance from F to the nearest reducible polynomial. In the final version of this paper we hope to compute a lower bound in this way and compare it to alternatives like [17] or the paragraph at the end of section 5.

3.4 Illustrative example

If a pair p,q of discriminant points of F comes from a double point of f, then we assume that the path around p,q corresponds to a path around that double point (and hence has trivial monodromy). The above program MatchingPermutations is based on this assumption. However, it is possible that this assumption fails; during the deformation, the points p,q move away from each other, starting at the double point. However, if during this process a third branchpoint r moves between p,q, then the effect of that will be that σ_p (or σ_q) will be conjugated by σ_r, after which $\sigma_p\sigma_q$ no longer needs to be trivial.

This seemingly unlikely situation actually happened in one of our examples, the example from section 1.4, because of a coincidence that happens with the projection on the x-axis. As a result, the code only finds 8 out of the 9 pairs that should have been matched. An easy fix is that switching to another projection will likely make the problem go away. Another fix would be to check σ_p,σ_q *and* σ_r whenever a pair p,q is so close to another branchpoint r that this situation may have occurred (for additional robustness this could be combined with the computation of the pseudo singular points in section 4.2, because such a point should be found near the center between p,q if p,q originated from a double point). For this example we simply switched to another projection by exchanging the roles of x and y.

We order the 30 roots of Δ_p as said before, and take the set S of the pairs under the threshold 0.3 that satisfy the previous matching test. Then we get 9 elements:

$$S := [[15, 16, .037], [18, 19, .075], [12, 13, .075], [23, 25, .12],$$

$$[6, 8, .12], [28, 29, .13], [2, 3, .13], [21, 26, .26], [5, 10, .26]]$$

So $30 - 9 \cdot 2 = 12$ permutations remain. The generated subgroup is not transitive and provides the partition in two subsets (with 3 elements each) of the fiber E. The corresponding factors will be computed by interpolation.

4. ALGORITHM 2

In this section we present a new method, based on a geometric analysis, for diminishing the distance between a perturbed composite polynomial satisfying our genericity hypothesis and the set of composite polynomials. Then the "preprocessed" input can be treated by various factorization algorithms including the following one that we may call an "early detection" of the monodromy.

4.1 Early detection

In this subsection, we sketch a probabilistic method to reach by an early detection the targeted partition of the fiber E.

In the previous section, algorithm 1 explores all points of the discriminant set Δ_p and determines iteratively a set of pairs of points of Δ_p which should be discarded together with their associated permutations of E. Thus the set of generators of the monodromy group decreases. The process stops when the orbit of the monodromy group splits, hence provides a partition of E.

Here we will proceed in the opposite direction: we start with an empty set of generators of the monodromy group and iteratively enlarge it, so we will coagulate the orbits to get the partition.

4.2 Pseudo singular points

We first need to locate the clusters of points on $Y = F^{-1}(0)$ created by the deformation of the singular points of $X = f^{-1}(0)$. We can call them pseudo singular points. They are the solutions of a system of three "approximate equations" (F, F'_x, F'_y) in (x, y). We expect the same order of magnitude ϵ for the derivatives of f as for f. By a deformation of f, a common solution A to the three equations (f, f'_x, f'_y) produces a cluster of three points (A_{12}, A_{13}, A_{23}). A_{12} (respectively A_{13}, and A_{23}) is the common solution of (F, F'_x), respectively of (F, F'_y) and (F'_x, F'_y). We expect that the order of magnitude of the distance between A and A_{12}, or A_{13} is about $\sqrt{\epsilon}$ while the order of magnitude of the distance between A and A_{23} is only about ϵ. A practical way for locating such clusters is as follows.

1. Compute approximately the set $\mathcal{A}$ of the $(n-1)^2$ common solutions of (F'_x, F'_y) in $\mathbb{C}^2$. This can be done via resultants of a rational approximation of F.

2. Take the subset $\mathcal{B} \subseteq \mathcal{A}$ consisting of those $P \in \mathcal{A}$ for which $F(P)$ is close to 0 (tolerance $O(\epsilon)$).

If we know the desired degrees of the factors, then we can work without a threshold. For example, if f should have two factors of degree $n/2$ then we can sort the points P in $\mathcal{B}$ (where the P for which $F(P)$ is relatively closest to 0 come first) and take the first $n^2/4$ elements.

4.3 Forbidden zones

We project $\mathcal{B}$ on the x-axis, and consider the set of m small disks (or squares) centered at the m projected points and of radius about $\sqrt{\epsilon}$, in such a way that each of them contains at least two points of the discriminant set Δ. Let U denote the union of these disks.

Then we choose a base point a outside of U and of Δ, and consider a set of random paths γ_j starting and ending at a avoiding U and of Δ and encircling about a third or half of the points of Δ. The paths are constructed iteratively and are concatenations of lines and portions of circles. To each such paths γ_j we associate via continuation, as explained in section 2, a permutation σ_j of the fiber E above a.

Let G_J be the group generated by a family $\{\sigma_j; j \in J\}$ of such permutations. When we increase J, the orbits of G_J coagulate and eventually they converge to the targeted partition of E.

4.4 Other strategies

If we already computed the monodromy as explained in section 2, we can use this information. Indeed we don't need to compute new continuations, but we can use a concatenation of the already computed paths in order to get the permutations associated to homotopic paths that avoid the forbidden zones. So we fully control the choice of the paths. We can also consider a mix of the two algorithms.

An opposite strategy would be to travel blind and construct the loops without computing in advance all the pseudo singular points: The forbidden zones are constructed dynamically when the path detects a nearby potential pseudo singular point via the size of a normalized gradient of F. Such a continuation algorithm was presented in [5].

4.5 Example

We applied the procedure of computation of the pseudo singular points to our illustrative example. With a threshold equal to 0.1 or to 0.2, we obtain 11 pseudo singular points, (X has only 9 singular points). However, one of the pseudo singular point of Y very near to a singular point of X is obtained with distance larger than the two "extra" ones. This shows that we may have to accept more pseudo singular points of Y than singular points of X in order not to miss one. For the lucky case that we do have the correct number of pseudo singular points we implemented a TrivialFactoring program [8] that simply interpolates those points to find factors (see also section 6).

5. WHAT HAPPENS WITH HIGHER SINGULARITIES

In the previous sections we made the important hypothesis that F was a small deformation of a composite polynomial $f = f_1 \cdots f_s$ such that $X_i = f_i^{-1}(0)$ are all smooth. Therefore the only singularities of f are a fixed number of nodes (double points).

In this section, we discuss some issues and sketch some possibilities of solutions when f has more complicated singularities.

First of all let us recall the classical results on perturbation of multiple roots of an univariate polynomial: the typical behavior is the one of $x^d - \epsilon$ with ϵ small, so the root 0 is perturbed at a distance of d^{th}-root of ϵ.

This extends to systems of two bivariate polynomials: the more complicated the singularity is, the further the roots move away. Indeed, it suffices to analyze the projections on two lines, and clusters of several roots are not easy to locate.

One strategy used in one variable, and which can be generalized in two (or more) variables to locate the clusters of d-roots of a polynomial P is the following. First compute the zeros of P, P', P", ..., $P^{(d)}$. Then for each zero x_1 of $P^{(d)}$, such that $P^{(d+1)}(x_1)$ is big enough, there are two zeros of $P^{(d-1)}$ at a distance about $\sqrt{\epsilon}$, then three zeros of $P^{(d-2)}$ at a distance about the third root of ϵ, and so on till d zeros of P at a distance about the d^{th}-root of ϵ. The situation is more complicated in two variables because one has to consider not the totally ordered iterations of derivations of P but the the only partially ordered iteration of the partial derivations of $F(x, y)$. This was studied by several authors including T. Ojika, and more recently G. Lecerf see references in e.g. [18].

In our setting such singularities may show up in three different ways: For one or more i, $X_i = f_i^{-1}(0)$ admits singularities, but

- We still have the property that for any two i and j, X_i and X_j intersect transversally (and their small deformations also intersect transversally) and form only nodes at these intersections. Then the strategy presented in the previous sections can be adapted and re-used with small changes, because we focus on the intersections.

- We still have the property that for any two i and j, the intersection point of X_i and X_j is a smooth point of X_i and a smooth point of X_j, but they are tangent. Then we can try to adapt the previous strategy but the analysis and the algorithm will be a bit more intricate.

- For two i and j, the intersection point of X_i and X_j is not a smooth point of X_i. Then the situation is even more complicated.

In all cases, a general strategy for obtaining an approximate factorization of F is to first approximate F by the nearby more singular polynomial.

Let us illustrate this paradigm on a case that we encountered when we experimented our algorithms. We created a benchmark of small degree polynomial with two factors, choosing two random polynomials f_1 and f_2 of degree 4 and 5 via the Maple command randpoly, and then deforming f into F. However, one such pair represented two curves X_1 and X_2 almost tangent at a point. Instead of finding the expected $4 \times 5 = 20$ common solutions to F'_x and F'_y, on which F almost vanishes, we found 21 even for quite small perturbations.

This can be explained as follows: to simplify the notations, let suppose that the tangent point correspond to two parabolas and is represented by the two equations $f_1 := y - x^2$ and $f_2 := y + 2x^2$, so $f = y^2 + x^2 y - 2x^4$. By a small deformation, the nearby composite curve is formed by the union of two parabolas intersecting (transversally) at two points nearby 0. However, the partial derivatives F'_x and F'_y, of the deformed polynomial F have 3 common roots nearby 0, (because $f'_y = 2y + x^2$ and $f'_x = 2xy - 8x^3$ have a root in 0 of multiplicity 3) and F almost vanishes at these 3 points, so we can hardly distinguish 2 out of these 3 points.

The suggestion is to consider the second derivatives: $f"_{yy} = 2$, $f"_{xy} = 2x$ and $f'_{xx} = 2y - 24x^2$ and observe that $f"^2_{xy} - f'_{xx} f'_{yy}$ vanishes at 0 and that $f"_{xy}$ and $f"_{xx}$ have a common simple solution in 0. This last property is conserved by a small deformation and give a good estimate of the location of the cluster.

If among the pseudo singular points we correctly selected the subset $\mathcal{B}$ coming from true singularities, then we can apply a Newton iteration to bring those points closer to true singularities, as follows: Consider the space of polynomials that vanish at all $P \in \mathcal{B}$, and then replace F by the projection of F on that space. We implemented this, see NearestNpoly on [8]. If the initial errors were sufficiently small, say of size $O(\epsilon)$, then this process has quadratic convergence and should lead to an output that is no further than $O(\epsilon^2)$ away from an optimal output f (assuming transversal intersections, the distance from a $P \in \mathcal{B}$ to a singularity of f is $O(\epsilon)$, and hence $f(P) = O(\epsilon^2)$. Note that this gives another potential way to determine a lower bound for the distance from F to a nearest reducible polynomial. We hope to compute such a lower bound and compare to other approaches in the final version of this paper).

6. INTERPOLATION

The previous procedures "recognize" the partition of the fiber $E = p^{-1}(a)$, above the base point a, into s disjoint subsets I_j with $1 \leq j \leq s$ of $\{1, \ldots, n\}$ and identify several points $M_i = (x_i, y_i)$ on the corresponding layer. Then it remains to compute, by interpolation at these points, candidate factors F_j. Then this factorization can be improved further by one or more Newton steps.

The general method is to find the coefficients of each F_j, e.g. in the monomial basis of dense polynomials of degree n_j, by imposing a family of linear conditions on these coefficients. These conditions express the vanishing of F_j at the M_i corresponding to I_j. Then one solves approximately the (overdetermined) linear system using QR or Svd techniques.

There are several ways to choose the points M_i.

1. Take any subset of the points M_i used during the analytic continuations in the monodromy computation.
2. Take m points x_i in the complementary of the discriminant locus and all n points $M_{i,l}$ above each x_i organized in a grid.
3. Take all double points.
4. Take for each j, the $n_j(n_j - 1)$ critical points M_i of F corresponding to the branching points producing I_j, as well as the conditions coming from $F'_y(M_i) = 0$.

The second possibility with $m = 2n$ points widely spread in the x-axis gives satisfactory results. However, it seems that the distance (defined by the difference of the coefficients) between each candidate factor F_i to the corresponding f_i is smaller than the distance from their product to F. So Newton step(s) can improve the result. If $s = 2$ a Newton step amounts to solving approximately (Svd) in the coefficients of $(\delta F_1, \delta F_2)$ the overdetermined linear system

$$F - F_1 * F_2 - F_1 * \delta F_2 - F_2 * \delta F_1 = 0.$$

The third and the fourth possibility have the advantage of focusing on the discriminant and critical loci. So the candidate F_i will have discriminant and a critical loci near a subset of the loci of F. This is more in the spirit of a geometric approach to the factoring problem.

Remark When the polynomial F is monic in y, the second possibility has the advantage that, for each triple of x-values (x_1, x_2, x_3), we can check the zero-sum condition for each set of points corresponding to a given j, then return to the main loop if it fails, or simply if needed deform the points in a fiber (as done e.g. in [25]) before the interpolation.

6.1 Illustrative example

With our illustrative example, we took 12 points in the rectangle $[-2..1, -1.5..1.5]$ included in the x-axis and the points above them (in the corresponding fibers) divided in two sets U and V. therefore, U defines 36 homogeneous linear conditions on the ten coefficients of F_1; and respectively for F_2. Once normalized by letting F_1 and F_2 to be monic in y. We have that the (normalized) maximal difference between the coefficients of f and its perturbation F is 0.01. We obtain two candidate factors F_1 and F_2 with a maximal difference with the coefficients of f_1 and f_2 of 0.02 and 0.03 while the maximal difference between the coefficients of F and $F_1 * F_2$ is 0.03. This can be improved with only one Newton step and we get new candidate factors $\tilde{F}_1$ and $\tilde{F}_2$ such that the maximal difference between the coefficients of F and $\tilde{F}_1 \tilde{F}_2$ is 0.003 which seems satisfactory.

7. CONCLUDING REMARKS

We presented a new geometric approach and two algorithms for factoring bivariate approximate polynomials. A draft implementation and some test files with examples are available at [8]. The algorithms work for bivariate approximate polynomials that are close enough to exactly factorable polynomials. The presented algorithm assumes that the

input polynomial approximates a product of polynomials defining smooth curves that intersect transversally. In section 5 we discussed the effect of more complicated singularities. In the near future we will try to extend our algorithm to this more general case. We can also incorporate in our algorithm, the zero-sums tests in order to speed up the iteration step before going to the interpolation step. We will also implement the use of orbits of the monodromy attached to chosen large paths including several branching points, to speed up the computation of the targeted partition, adapting to our setting the observation of [24]. Another interesting direction of research is to compare our approach with the one using the Ruppert-Gao matrix developed in [11].

8. REFERENCES

[1] C. Bajaj, J. Canny, T. Garrity, and J. Warren. Factoring rational polynomials over the complex numbers. *SIAM J. Comput.*, 22(2):318–331, 1993.

[2] G. Chèze and A. Galligo. Four lectures on polynomial absolute factorization. In *Solving polynomial equations*, volume 14 of *Algorithms Comput. Math.*, pages 339–392. Springer, Berlin, 2005.

[3] G. Chèze and A. Galligo. From an approximate to an exact absolute polynomial factorization. *J. Symbolic Comput.*, 41(6):682–696, 2006.

[4] G. Cheze and G. Lecerf. Lifting and recombination techniques for absolute factorization. Accepted for publication in Journal of Complexity, 2006.

[5] R. Corless, M. Giesbrecht, I. K. M. van Hoeij, and S. Watt. Towards factoring bivariate approximate polynomials. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*. ACM, 2001.

[6] B. Deconinck and M. van Hoeij. Computing riemann matrices of algebraic curves. *PhysicaD*, 152:28–46, 2001.

[7] A. Galligo. Real factorization of multivariate polynomials with integer coefficients. *Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI)*, 258(Teor. Predst. Din. Sist. Komb. i Algoritm. Metody. 4):60–70, 355, 1999.

[8] A. Galligo and M. van Hoeij. Implementation: http://www.math.fsu.edu/~hoeij/files/approxfact.

[9] A. Galligo and S. Watt. A numerical absolute primality test for bivariate polynomials. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 217–224 (electronic), New York, 1997. ACM.

[10] S. Gao. Factoring multivariate polynomials via partial differential equations. *Math. Comp.*, 72(242):801–822 (electronic), 2003.

[11] S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In *ISSAC '04: Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 167–174, New York, NY, USA, 2004. ACM Press.

[12] M. Hitz, E. Kaltofen, and Y. N. Lakshman. Efficient algorithms for computing the nearest polynomial with a real root and related problems. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, pages 205–212, New York, 1999. ACM.

[13] W. Huang, Y.and Wu, H. J. Stetter, and L. Zhi. Pseudofactors of multivariate polynomials. In *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*, pages 161–168, New York, 2000. ACM.

[14] E. Kaltofen. Polynomial factorization 1982–1986. In *Computers in mathematics (Stanford, CA, 1986)*, volume 125 of *Lecture Notes in Pure and Appl. Math.*, pages 285–309. Dekker, New York, 1990.

[15] E. Kaltofen. Polynomial factorization 1987–1991. In *Proc. LATIN '92*, volume 583 of *Lect. Notes Comput. Sci*, pages 294–313, New York, 1992. Springer Verlag.

[16] E. Kaltofen. Challenges of symbolic computation: My favourite open problems. *JSC*, 29(6):891–919, 2000.

[17] E. Kaltofen and J. May. On approximate irreducibility of polynomials in several variables. In *ProcISSAC03*, pages 161–168, 2003.

[18] A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theor. Comput. Sci.*, 359(1):111–122, 2006.

[19] D. Rupprecht. *Elements de gomtrie algbrique approche: Etude du pgcd et de la factorisation.* PhD thesis, Univ. Nice Sophia Antipolis, 2000.

[20] T. Sasaki. Approximate multivariate polynomial factorization based on zero-sum relations. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, pages 284–291. ACM, 2001.

[21] T. Sasaki and M. Sasaki. A unified method for multivariate polynomial factorizations. *Japan J. Indust. Appl. Math.*, 10(1):21–39, 1993.

[22] T. Sasaki, M. Suzuki, M. Kolář, and M. Sasaki. Approximate factorization of multivariate polynomials and absolute irreducibility testing. *Japan J. Indust. Appl. Math.*, 8(3):357–375, 1991.

[23] A. Sommese, J. Verschelde, and C. Wampler. Numerical irreducible decomposition using projections from points on the components. In *Symbolic Computation: Solving Equations in Algebra, Geometry, and Engineering*, volume 286 of *Contemporary Mathematics*, pages 37–51. AMS, 2001.

[24] A. Sommese, J. Verschelde, and C. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In *Application of Algebraic Geometry to Coding Theory, Physics and Computation*, pages 297–315. Kluwer Academic Publishers, 2001. Proceedings of a NATO Conference, February 25 - March 1, 2001, Eilat, Israel.

[25] A. Sommese, J. Verschelde, and C. Wampler. Symmetric functions applied to decomposing solution sets of polynomial systems. *SIAM J. Numer. Anal.*, 40(6):2026–2046, 2002.

[26] Z. Zeng and B. H. Dayton. The approximate gcd of inexact polynomials. In *ISSAC '04: Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 320–327, New York, NY, USA, 2004. ACM Press.