Absolute Factoring of bidegree Bivariate Polynomials*

G. Chèze, M. Elkadi, A. Galligo, M. Weimann

September, 2008

Abstract

We describe an efficient algorithm and an implementation for computing an absolute factorization of a bivariate polynomial with a given bidegree. Results of experimentation and an illustrative example are provided.
This algorithm is a generalization of the previous one by Rupprecht-Galligo-Chèze which works after a generic change of coordinates. It relies on a general algorithmic approach based on a study of the curve defined by the polynomial to factorize in a toric surface.

1 Introduction

Polynomial absolute factorization has been considered from many points of view but during the last decade two main strategies have been quite successful. An algebraic approach which relies on the study of Ruppert-Gao matrix improved in [Lec07] to provide an algorithm with a quasi-optimal complexity. And a geometric approach based on a zero-sum criterion (à la Sasaki), provides very efficient semi-numerical probabilistic algorithms able to deal with polynomials having degree up to 200 [Chez04].

The zero-sum criterion can be considered as a consequence of Wood's theorem on algebraic interpolation of a family of analytic germs of curves. A generalization of this theorem is adapted to the factorization of polynomials with given Newton polytopes [EGW08]. Here we concentrate on the bidegree case.

In our model of computation the input is a polynomial with integer coefficients and the output is a list of polynomials with coefficients in an algebraic extension of $\mathbb{Q}$ which should also be computed. In order to determine these coefficients the strategy consists in embedding $\overline{\mathbb{Q}}$ in $\mathbb{C}$ and representing approximations of these coefficients by bigfloats. Then conjugacy relations are used to recognize an algebraic presentation of an extension of $\mathbb{Q}$ together with an exact algebraic presentation of the coefficients.

We assume that the polynomial $f \in \mathbb{Z}[x,y]$ is irreducible, that a generic translation was performed toghether with a homothetic transformation such that the constant term in f is equal to 1, and the bidegree of f is (m,n).

2 Representation and interpolation

Proposition 1. *Let $f \in \mathbb{Z}[x,y]$ be an irreducible polynomial with constant term equal to 1. Denote by $f = f_1 \dots f_q$ its absolute factorization. Then the irreducible absolute factors f_i of f are conjugated over $\mathbb{Q}$ and their constant term is equal to 1.*

The absolute factorization of f is completely determined by the number of absolute factors q, an irreducible monic univariate polynomial $g(t) \in \mathbb{Z}[t]$ defining a finite extension $\mathbb{K} = \mathbb{Q}[t]/\big(g(t)\big)$, and the coefficients of f_1 (which are algebraic integers) in $\mathbb{K}$. The bidegree of each factor f_i is $(m_1,n_1) = (\frac{m}{q}, \frac{n}{q})$.

The absolute irreducible factors of f are in one-to-one correspondence with the irreducible components of the affine curve $C = \{(x,y) \in \mathbb{C}^2 : f(x,y) = 0\}$. This curve intersects transversely the line $y = 0$ (resp. $x = 0$) in m (resp. n) distinct points denoted by $M_i = (x_i, 0)$ (resp. $N_j = (0, y_i)$), and moreover they are all distinct from the origin. The germs of C at these points form $m+n$ curves which can be parameterized. An interesting way to achieve that parameterization is to cut C by the family of conics $xy - a = 0$, a varies near 0. The intersection points are denoted by $\mathbb{M}_1(a), \dots, \mathbb{M}_m(a)$ and $\mathbb{N}_1(a), \dots, \mathbb{N}_n(a)$.

Let $I \subset \{1, \dots, m\}$ and $J \subset \{1, \dots, n\}$. We define the trace of the coordinate function y with respect to I, J as

$$(\mathrm{Tr}_C\, y)_{I,J}(a) := \sum_{i \in I} y(\mathbb{M}_i(a)) + \sum_{j \in J} y(\mathbb{N}_j(a)).$$

The curve C is absolutely irreducible iff this function is affine in a only for $I = \{1, \dots, m\}$ and $J = \{1, \dots, n\}$ [EGW08].

*This research was partly supported by the french ANR contrat Gecko. Elkadi and Galligo are also members of the Galaad project team, INRIA-UNS.

The implicit function theorem applied to f at points M_i (resp. N_j) gives the following Taylor expansions: $y = \frac{1}{x_i}a - \frac{c_i}{x_i^3}a^2 + O(a^3)$ (resp. $y = y_j + \frac{a_j}{y_j}a - \frac{a_j^2}{y_j^3}a^2 + \frac{b_j}{y_j^2}a^2 + O(a^3)$.

The criterion in the bidegree case is explicitly: Two subsets of intersection points between C and the two axes, indexed by $I \subset \{1,\ldots,m\}$ and $J \subset \{1,\ldots,n\}$, correspond to an absolute factor of f iff

$$\sum_{i\in I}(-\frac{c_i}{x_i^3}) + \sum_{j\in J}(-\frac{a_j^2}{y_j^3} + \frac{b_j}{y_j^2}) = 0. \tag{1}$$

3 Algorithm, implementation and experimentation

As in [Chez04], good approximations of roots x_i and y_j are computed by a Newton process. Then the LLL algorithm can be applied to solve a knapsack problem of size $m+n$ associated to the previous sums and determine the partitions of $\{1,\ldots,m\}$ and $\{1,\ldots,n\}$ in q subsets denoted by I_k and J_k, $k = 1,\ldots,q$. Each pair of such subsets should correspond to a factor f_k of f, a polynomial of bidegree $(\frac{m}{q}, \frac{n}{q})$. The solutions of $f_k(x,0) = 0$ (resp. $f_k(0,y) = 0$) are the M_i (resp. N_j) indexed by I_k (resp. J_k). As the constant term of f is 1, good approximations of $f_k(x,0)$ and of $f_k(0,y)$ can be computed.

Our algorithm consists in applying Hensel liftings with respect to y to lift the obtained approximation of $f_k(x,0)$ to an approximate factorization of f. We know that the factors must be conjugated and their coefficients are conjugated algebraic integers. So an irreducible monic polynomial $g(z)$ defining a field extension is recovered from a sufficiently good approximation by bigfloats of coefficients as in [ChGa06]. Then the exact expression of the coefficients of a factor f_1 is recovered.

The structure of this algorithm is similar to the one described in details in [ChGa06] and the costs of all steps in the two algorithms are comparable, except the lifting step. Indeed, the new algorithm needs only $n_1 = \frac{n}{q}$ linear steps (or $\log(n_1)$ quadratic steps) of Hensel liftings of polynomials of degrees $m_1 = \frac{m}{q}, m - m_1 n$ instead of polynomials of degrees $m_1 + n_1, m - m_1 - n_1$. This makes a significant difference when m is much smaller than n.

Example 1. *We apply our algorithm to a simple randomly generated example with $m = 4$, $n = 6$. The precision is set to Digits $:= 10$, approximations of roots y_j and x_i are computed. Then following the previous formulae, the coefficients to be summed are:*

$$BBy := [-2.657975570, -.2371654537 + 13.23148974 * I, -.2371654537 - 13.23148974 * I,$$
$$-1521.517910 - 226.3038301 * I, -1521.517910 + 226.3038301 * I, -1.831875847],$$
$$BBx := [-0.3884114512e-1, 1.153103449 + 2.550219536 * I, 1.153103449 - 2.550219536 * I, 3045.732635].$$

We get only two (approximate) zero sums, one of them is:

$$BBy[4] + BBy[5] + BBy[1] + BBx[1] + BBx[4] = -0.00002$$

when the other sums are much bigger than 1 in norm.

So we estimate that $q = 2$ and we build two approximate factors of $f(x,0)$ (resp. $f(0,y)$) with constant term equal to 1:

$$15.0710 * x^2 + 26.7279 * x + 1.00000 \quad \text{and} \quad .9289 * x^2 + 1.2720 * x + .9999.$$

The minimal polynomial of the first coefficient is $G(t) = (t - 15.0710)(t - .9289) = t^2 - 15.999 * t + 13.999$. So we recognize $g(t) = t^2 - 16 * t + 14$. The roots of $g(t)$ are $8 + 5 * \sqrt{2}$ and $8 - 5 * \sqrt{2}$. Hence the field is $\mathbb{K} = \mathbb{Q}(\sqrt{2})$. Then we recognize the coefficients of $f_1(x,0)$, $f_2(x,0)$, $f_1(0,y)$ and $f_2(0,y)$. For instance the first coefficient of $f_1(x,0)$ is $8 + 5 * \sqrt{2}$.

Then we proceed to the Hensel liftings, taking advantage that we already know the coefficients of $f_1(0,y)$ and $f_2(0,y)$. Therefore at each step the inverse of the Sylvester matrix of $f_1(x,0)$ and $f_2(x,0)$ is used to determine m coefficients.

We implemented the steps of the algorithm in Maple and tested it on examples constructed with random data. Here are the details on the routines and the timmings corresponding to an example of total degree 100 and bidegre $(m = 20, n = 80)$. We will see that it has 4 absolute factors. As a preprocessing, we perform a generic translation and a homothetic transformation to get a polynomial $f(x,y)$ with integer coefficients and constant term equal to 1.

1. Computation of approximate roots of $f(x,0)$ and $f(0,y)$, Digits is set to 200. Then the $m+n$ coefficients *BBx* and *BBy*, to be summed, are computed by evaluation of the formulae (1). This step took about 20 seconds and used 20 MO of memory.

2. Solution of the knapsack problem. We used the LLL function from the package IntegerRelations with the option integer, then simplified the output by a row echelon reduction. We got a partition with 4 subsets. This step took about 900 seconds and used 30 MO of memory.

3. Computation of good approximations of $f_k(x,0)$, $k = 1..4$, then the construction of cofactors $g_k(x,0) := \frac{f}{f_k}$, and similarly $f_k(0,y)$ and $g_k(0,y)$.

   With the 4 first coefficients, construction of an approximate minimal polynomial of the extension field. Recognition of its integer coefficients. We then used the function OptimizedRepresentation in Magma to get a minimal polynomial of the extension field with smaller coefficients. In our example it was simply the extension $\mathbb{Q}(\sqrt{2},\sqrt{3})$. This step took 10 seconds and used 10 MO of memory.

4. Recognition of exact coefficients of $f_1(x,0)$, $g_1(x,0)$, $f_1(0,y)$ and $g_1(0,y)$ in $\mathbb{Q}(\sqrt{2},\sqrt{3})$. This step took less than 10 seconds and used no additional memory.

5. Inversion of the (m,m) Sylvester matrix of $f_1(x,0)$ and $g_1(x,0)$. This step took about 500 seconds and used 150 MO of memory.

6. n_1 linear Hensel liftings by applying the inversed matrix recursively to the vector of the coefficients of $f - d_{j-1}f * d_{j-1}g - y^j * g_1(x,0) * coeff(f_1(0,y),y,j) - y^j * f_1(x,0) * coeff(g_1(0,y),y,j),y,j)$. This step took about 5000 seconds and used 50 MO of memory.

7. In total it took less than 2 hours to factor this polynomial and most of the time was used in Hensel liftings.

4 Conclusion

The presented symbolic-numeric algorithm produces efficiently exact absolute factorization of a bivariate polynomial when its bidegree is known. The experimentations show encouraging results.

In a near future we will implement and compare several algorithms in the computer algebra system and library under development Mathemagix.

References

[Chez04] G. CHÈZE, *Absolute polynomial factorization in two variables and the knapsack problem*, in ISSAC 2004, ACM, New York, 2004, pp. 87–94.

[ChGa06] G. CHÈZE AND A. GALLIGO, *From an approximate to an exact absolute polynomial factorization*, J. Symbolic Comput., 41 (2006), pp. 682–696.

[EGW08] M. ELKADI, A. GALLIGO, M. WEIMANN, *Towards Absolute Toric Factorization*, J. Symb. Comp, (to appear).

[Lec07] G. LECERF, *Improved dense multivariate polynomial factorization algorithms*, J. Symbolic Comput., 42 (2007), pp. 477–494.