Journal of Complexity 34 (2016) 78–128

Contents lists available at ScienceDirect

Journal of Complexity

journal homepage: www.elsevier.com/locate/jco

A softly optimal Monte Carlo algorithm for solving bivariate polynomial systems over the integers

Esmaeil Mehrabi, Éric Schost*

Computer Science Department, Western University, London, ON, Canada

ARTICLE INFO

Article history:
Received 27 February 2015
Accepted 21 September 2015
Available online 28 November 2015

Keywords:
Bivariate system
Complexity
Algorithm

ABSTRACT

We give an algorithm for the symbolic solution of polynomial systems in $\mathbb{Z}[X, Y]$. Following previous work with Lebreton, we use a combination of lifting and modular composition techniques, relying in particular on Kedlaya and Umans' recent quasi-linear time modular composition algorithm.

The main contribution in this paper is an adaptation of a deflation algorithm of Lecerf, that allows us to treat singular solutions for essentially the same cost as the regular ones. Altogether, for an input system with degree d and coefficients of bit-size h, we obtain Monte Carlo algorithms that achieve probability of success at least $1 - 1/2^{\mathcal{P}}$, with running time $d^{2+\varepsilon}\tilde{O}(d^2 + dh + d\mathcal{P} + \mathcal{P}^2)$ bit operations, for any $\varepsilon > 0$, where the $\tilde{O}$ notation indicates that we omit polylogarithmic factors.

1. Introduction

Overview. Newton iteration is one of the most popular components of polynomial system solvers, from either the numeric or symbolic points of view. The usual version of this procedure handles situations without multiplicities only, since it requires that the Jacobian matrix of the given system be invertible at the roots we are looking for. To handle singular roots, various forms of *deflation techniques* have been developed (we will review some of them below).

* Corresponding author.
E-mail addresses: emehrab@uwo.ca (E. Mehrabi), eschost@uwo.ca (É. Schost).

http://dx.doi.org/10.1016/j.jco.2015.11.009
0885-064X/

In this paper, we are interested in applying such techniques to the symbolic solution of bivariate polynomial systems $F = G = 0$, with F and G in $\mathbb{Z}[X, Y]$. This is in the continuation of previous work with Lebreton [29], where Newton iteration techniques were used to handle solutions without multiplicities of the system $F = G = 0$. In this work, using results and ideas from [29], as well as Lecerf's deflation algorithm [31], we extend this approach to all solutions.

Motivated by applications to computational topology or computer graphics, recent years have witnessed the publication of a large body of work on bivariate systems. While some algorithms rely mostly on numerical techniques such as subdivision [3], many recent results involve symbolic elimination techniques, possibly in combination with real or complex root isolation [5,7–9,17–19,23, 25,47]; we will discuss some these results further below.

Our interest here is on complexity of the symbolic side of such algorithms. In a nutshell, our main result says that bivariate systems with integer coefficients can be solved "symbolically" in essentially optimal time by Monte Carlo algorithms.

Over an arbitrary field. Let us first discuss known results for solving a bivariate system $F = G = 0$ over $\mathbb{K}[X, Y]$, where $\mathbb{K}$ is an arbitrary perfect field. Suppose that the zero-set $V(F, G)$ of F and G in an algebraic closure $\overline{\mathbb{K}}$ of $\mathbb{K}$ is finite. In this case, if F and G have total degree at most d, the Bézout theorem implies that the system $F = G = 0$ has at most d^2 solutions.

Several approaches exist to describe the solutions of our system: Gröbner bases, triangular representations, or descriptions based on univariate polynomials. For instance, in [29], together with Lebreton, we relied on a canonical description of a zero-dimensional variety, called the equiprojectable decomposition [14], using triangular sets; this is close to the decomposition based on subresultant calculations used in [23], but as a geometric notion, it does not take into account multiplicities in the input system.

Although it would be natural to use this kind of description here as well, the techniques we rely on are slightly easier to apply when working in generic coordinates. Indeed, if we are in generic coordinates, the zeros of $F = G = 0$ can simply be described by a pair of polynomials in $\mathbb{K}[X]$, of the form

$$P(X) = 0, \qquad Y = S(X); \tag{1}$$

the polynomials (P, S) will be called a *lexicographic basis* of F, G, since $(P, Y - S)$ is indeed a Gröbner basis of the radical of $\langle F, G\rangle$, for the lexicographic order $Y > X$. Remark that for such a description to make sense, no two points on $V(F, G)$ should have the same abscissa; this is precisely what is ensured once we are in generic coordinates. In such an output, our choice is to take P squarefree; in other words, our representation of the solutions does not reflect multiplicities (as a matter of fact, if $\mathbb{K}$ were not perfect, P could still have multiple roots in $\overline{\mathbb{K}}$ while being squarefree in $\mathbb{K}[X]$).

The input polynomials F and G have degree d in two variables; the polynomials P and S have degree at most d^2 in one variable. Thus, representing both input and output involves only $O(d^2)$ elements in $\mathbb{K}$. One would then naturally hope that P and S could be computed within $O^{\sim}(d^2)$ operations in $\mathbb{K}$, where the $O^{\sim}(\ )$ notation omits polylogarithmic factors (see [51, Definition 25.8]).

However, no such result is known; the very close problem of computing the resultant of F and G using $O^{\sim}(d^2)$ operations is given as a research problem in [51, Problem 11.11]. For the latter resultant problem, the best algorithm known so far [45] uses $O^{\sim}(d^3)$ operations in $\mathbb{K}$.

Systems over the integers. In this paper, we are going to work in the particular case where $\mathbb{K} = \mathbb{Q}$. In such cases, it becomes crucial to take into account the bit-size of the input and output as well; cost estimates will then be given in a boolean model (explicitly, a RAM with logarithmic cost, see [2, Section 1.2]).

For a nonzero integer a, we write $\mathrm{len}(a) = \lceil \log(|a|) \rceil$, and we call this the *length* of a; for $a = 0$, we write $\mathrm{len}(0) = 1$. This quantity essentially represents the amount of bits needed to store a (one may also work with the *height* of a, written $\mathrm{ht}(a) = \log(|a|)$, but the fact that $\mathrm{len}(a)$ takes integer values will be useful to us). It will be convenient to introduce a notion of length for polynomials with coefficients in $\mathbb{Q}$ as well: if P is such a polynomial, the length $\mathrm{len}(P)$ denotes the maximum of the lengths $\mathrm{len}(d)$ and $\mathrm{len}(n_i)_{i\in I}$, where d is a minimal common denominator for all coefficients of P and