Relaxed Hensel lifting of triangular sets

Romain Lebreton

Relaxed Hensel lifting of triangular sets

Extended abstract

Abstract

In this paper, we present a new lifting algorithm for triangular sets over power series. Our contribution is to give, for any power series triangular set, a shifted algorithm of which the triangular set is a fixed point. Then we can apply the relaxed recursive power series framework and deduce a relaxed lifting algorithm for this triangular set.

We compare our algorithm to the existing techniques. Our algorithm always improves the asymptotic cost in the precision for the special case of univariate representations.

Our goal in this paper is to extend a growing body of work on *relaxed algorithms* to the context of lifting techniques for *univariate representations* and *triangular sets*.

It is well-known that, under some regularity conditions, techniques such as Newton iteration can be used to compute a power series root of an equation such as $f(t, x(t)) = 0$, with f in $k[t, X]$, or a p-adic root of an equation of the form $f(x) = 0$ with f in $\mathbb{Z}[X]$.

Relaxed methods, introduced by van der Hoeven [Hoe02, Hoe11], offer an alternative to Newton iteration. The case of computing one power series root, or one p-adic root, of a *system* of polynomial equations was worked out in [BL12]; for this problem, the relaxed algorithm was seen to behave better than Newton iteration in some cases, for instance for multivariate systems with a large number of equations.

In this paper, we go beyond the case of lifting a single root of a multivariate system: we deal with all roots at once, introducing relaxed algorithms that deal with objects such as univariate and triangular representations.

Example 1. We consider the polynomial system $\boldsymbol{f} = (f_1, f_2)$ in $\mathbb{Z}[X_1, X_2]$ with

$$\begin{aligned} f_1 &:= 33\,X_2^3 + 14699\,X_2^2 + 276148\,X_1 + 6761112\,X_2 - 11842820 \\ f_2 &:= 66\,X_1\,X_2 + X_2^2 - 94\,X_1 - 75\,X_2 - 22. \end{aligned}$$

Let $\mathfrak{m} = (7)$, let $k = \mathbb{Z}/7\mathbb{Z}$ and let $\boldsymbol{T}_1$ be the triangular set of $k[X_1, X_2]$, that is a lexicographical Gröbner basis for $X_1 < X_2$, given by

$$\boldsymbol{T}_1 := (X_1^2 + 5\,X_1, \quad X_2^2 + 3\,X_1\,X_2 + 2\,X_2 + 4X_1 + 6).$$

We lift the triangular set $\boldsymbol{T}_1$ defined modulo 7 to triangular sets $\boldsymbol{t}$ defined modulo $7^2, 7^3, \ldots$. At the first step, we have

$$\boldsymbol{T}_2 = (X_1^2 + (5 + 5\cdot 7)\,X_1 + 7, X_2^2 + (3 + 2\cdot 7)\,X_1\,X_2 + (2 + 3\cdot 7)\,X_2 + 4\,X_1 + (6 + 3\cdot 7))$$

in $(\mathbb{Z}_7/7^2\,\mathbb{Z}_7)[X_1, X_2]$. We iterate again and find

$$\begin{aligned} \boldsymbol{T}_3 \;=\; & (X_1^2 + (5 + 5\cdot 7 + 6\cdot 7^2)\,X_1 + (7 + 7^2), \\ & X_2^2 + (3 + 2\cdot 7 + 7^2)\,X_1\,X_2 + (2 + 3\cdot 7 + 5\cdot 7^2)\,X_2 + (4 + 5\cdot 7^2)\,X_1 + \\ & (6 + 3\cdot 7 + 6\cdot 7^2)) \end{aligned}$$

in $(\mathbb{Z}_7/7^3\,\mathbb{Z}_7)[X_1, X_2]$. The precision is enough to recover the triangular set

$$\boldsymbol{T} := (X_1^2 - 9\,X_1 + 56,\ X_2^2 + 66\,X_1\,X_2 - 75\,X_2 - 94\,X_1 - 22) \in \mathbb{Z}[X_1, X_2].$$

For simplicity, we deal only with lifting power series (so our equations will involve a parameter t). Similar techniques apply as well to p-adic lifting, but the simpler case of power series is useful in itself (see below) and shows all useful ideas. Consider a system of polynomial equations $\boldsymbol{f} = (f_1, ..., f_n) \in k[t][X_1, ..., X_n]$, where k is a field. Denote by $\mathcal{I}$ the ideal generated by $(f_1, ..., f_n)$ in $k(t)[X_1, ..., X_n]$. In what follows, we make the following assumptions, denoted (H):

- the algebraic set $V = V(\mathcal{I}) \subset \overline{k(t)}^n$ has dimension zero;
- the Jacobian determinant of $(f_1, ..., f_n)$ vanishes nowhere on V.

Equivalently, one may consider the curve defined by $\boldsymbol{f}$ in $\overline{k}^{n+1}$; the zero-dimensional set V describes its generic points over $k(t)$.

Let d be cardinality of V; due to our non-vanishing assumption on the Jacobian of $\boldsymbol{f}$, the extension $k(t) \to A = k(t)[X_1, ..., X_n]/\mathcal{I}$ is a product of separable field extensions. As a result, A has dimension d over $k(t)$.

In order to describe V, we will consider two data structures, which we briefly describe now.

An element P of $A = k(t)[X_1, ..., X_n]/\mathcal{I}$ will be called *primitive* if the K-algebra $k(t)[P] \subset A$ spanned by P in A is equal to A itself. If Λ is a primitive linear form in A, a *univariate representation* of A consists of polynomials $\mathbf{P} = (Q, S_1, ..., S_n)$ in $k(t)[Z]$, where Z is a new variable, with $\deg(S_i) < \deg(Q)$ for all i and such that we have a K-algebra isomorphism

$$\begin{array}{ccc} A = k(t)[X_1, ..., X_n]/\mathcal{I} & \to & k(t)[Z]/(Q) \\ X_1, ..., X_n & \mapsto & S_1, ..., S_n \\ \Lambda & \mapsto & Z. \end{array}$$

In particular, Q has degree d in Z; we will say that $\mathbf{P}$ has degree d.

Such representations, or slight modifications thereof (using for instance a rational form for the S_i's) have been used for decades in computer algebra, under the names of Shape Lemma, Rational Univariate Representation, Geometric Resolution, etc [GM89, GH91, GHH+97, Rou99, GLS01, HMW01].

On the other side, one finds triangular representations. A *triangular set* is a set of n polynomials $\mathbf{T} = (T_1, ..., T_n) \subseteq k(t)[X_1, ..., X_n]$ such that for all i, T_i is in $k(t)[X_1, ..., X_i]$, monic and reduced with respect to $(T_1, ..., T_{i-1})$. This is thus a reduced Gröbner basis for the lexicographic order $X_1 < \cdots < X_n$. The notion of triangular set comes from [Rit66] in the context of differential algebra. Many similar notions were introduced afterwards [Wu84, Laz91, Kal93, ALM99]; although all these notions do not coincide in general, they are the same for zero-dimensional ideals.

Assume that $\mathcal{I}$ admits a triangular family of generators $\boldsymbol{T}$, and write $d_i = \deg(T_i, X_i)$ for all i. Then, we have the equality $d = d_1 \cdots d_n$; we will say that $\boldsymbol{T}$ has multi-degree $(d_1, ..., d_n)$.

Both kinds of data structures have found numerous applications, and it is not our purpose here to compare their respective merits. Univariate representations always exist, provided the base field is large enough. On the other hand, the ideal $\mathcal{I}$ may not admit a triangular family of generators: such issues are handled using *triangular decompositions* of $\mathcal{I}$ [DMS+05]. We will not enter this discussion here, and we will simply suppose when needed that $\mathcal{I}$ admits one or the other such representation.

Let us for instance suppose that $\mathcal{I}$ admits a univariate representation $\mathbf{P}$. Let further $\mathfrak{m} = (t - t_0)$ be a maximal ideal in $k[t]$, with residual field k, for some $t_0 \in k$. We make the following assumptions, denoted by $(\mathsf{H}')_{\boldsymbol{P},t_0}$:

- none of the denominators appearing in $\boldsymbol{P}$ vanishes modulo $\mathfrak{m}$; we write $\boldsymbol{P}_1$ for the reduction of $\boldsymbol{P}$ modulo $\mathfrak{m}$;
- the polynomials $\boldsymbol{f} \bmod \mathfrak{m}$ still satisfy (H);
- $\boldsymbol{P}_1$ is a univariate representation of $k[X_1, \ldots, X_n]/(\boldsymbol{f} \bmod \mathfrak{m})$.

Then, given $\boldsymbol{P}_1 = \boldsymbol{P} \bmod \mathfrak{m}$ and $\boldsymbol{f}$, our objective is to compute objects of the form $\boldsymbol{P}_N = \boldsymbol{P} \bmod \mathfrak{m}^N$, for higher powers of N. Note that, without loss of generality, we could assume $t_0 = 0$ (by shifting the origin in t of the input polynomials).

Similar questions can be asked for triangular representations: suppose that $\mathcal{I}$ admits a triangular representation $\boldsymbol{T}$, and that the natural analogue $(\mathsf{H}')_{\boldsymbol{T},t_0}$ of $(\mathsf{H}')_{\boldsymbol{P}}$ holds; we will show how to compute $\boldsymbol{T}_N = \boldsymbol{T} \bmod \mathfrak{m}^N$, for high powers of N, starting from $\boldsymbol{T}_1 = \boldsymbol{T} \bmod \mathfrak{m}$ and $\boldsymbol{f}$.

Remark that for both the univariate and the triangular case, assumption $(\mathsf{H}')_{\boldsymbol{P},t_0}$, resp. $(\mathsf{H}')_{\boldsymbol{T},t_0}$, holds for all values of t_0 except finitely many: this is analyzed in detail in [Sch03a, Sch03b]; thus, these are very mild assumptions.

Let us say a few words about applications of this kind of techniques. One direct application is naturally to solve polynomial systems that depend on a parameter t: the techniques discussed here allow one to compute a description of V over $k(t)$ by computing it modulo $(t - t_0)$ (which we expect to be an easier problem) and lifting it to a sufficient precision (and possibly applying rational reconstruction to recover coefficients in $k(t)$ from their power series expansion).

Such lifting techniques are also at the core of many further algorithms: for instance, it is used in the geometric resolution algorithm [GLS01, HMW01] that computes univariate representations. This algorithm relies on an iterative lifting / intersection process: the lifting part involves lifting univariate representation, as explained in this paper, while the intersection part uses resultant computations. On the triangular side, a similar lifting / intersection process is used in the change of order algorithm of [DJMS08].

In order to analyze the cost of our lifting algorithms, we will need the following notations for polynomial and power series arithmetic.

- We denote by $\mathsf{M}\colon \mathbb{N} \to \mathbb{N}$ a function such that one can multiply polynomials in $k[t]$ of degree at most N in $\mathsf{M}(N)$ operations in k; we also ask that M satisfies the super-linearity conditions of [GG03]. Using FFT techniques, one can take $\mathsf{M}(N) = \mathcal{O}(N \log(N) \operatorname{loglog}(N))$.

- We denote by $\mathsf{R}: \mathbb{N} \to \mathbb{N}$ a function such that *online* (or equivalently, *relaxed*) multiplication of power series over k can be done at precision N using $\mathsf{R}(N)$ operations in k. Using the algorithm of [FS74, Hoe02], one can take $\mathsf{R}(N) = \mathcal{O}(\mathsf{M}(N)\log(N))$.

Recall that online power series multiplication computes the product of two power series $\sum_{i\geqslant 0} c_i\, t^i := \sum_{i\geqslant 0} a_i\, t^i \times \sum_{i\geqslant 0} b_i\, t^i$, with the constraint that for all i, the coefficient c_i of the output is written before the coefficients a_{i+1}, a_{i+2}, ... or b_{i+1}, b_{i+2}, ... of the inputs are read [Hen66, FS74, Hoe02].

We will further need linear algebra operations. We denote by ω the constant such that one can multiply matrices of size n over a ring R using $\mathcal{O}(n^\omega)$ operations $\{+, -, \times\}$ in R. Similarly, we denote Ω the constant such that one can compute the determinant and the adjoint matrix using $\mathcal{O}(n^\Omega)$ operations $\{+, -, \times\}$ in R. Using the algorithms of [CW90, Sto10, Vas11] for the multiplication and of [Ber84, Kal92, KV04] for the determinant and the adjoint matrix, one can take $\omega \leqslant 2.38$ and $\Omega \leqslant 2.70$. The computation of the determinant and the adjoint matrix is used to compute an unevaluated inverse of a matrix A^{-1} via the formula $A^{-1} = 1/\det(A)\,\mathrm{Adjoint}(A)$, *i.e.* the division by the determinant has yet to be performed.

The bulk of the computations will rely on *modular* arithmetic: we will need to perform arithmetic operations, mostly additions / multiplications, modulo either a univariate polynomial (in the case of univariate representations) or a whole triangular set.

- In the former case, it is known that if $Q \in R[Z]$ has degree d (for some ring R), additions and multiplications modulo Q can be done using $\mathcal{O}(\mathsf{M}(d))$ operations in R [GG03]. When R is a field, inversion modulo Q, when possible, can be done in time $\mathcal{O}(\mathsf{M}(d)\log(d))$ [GG03].

- If $\boldsymbol{T} = (T_1, ..., T_n)$ is a triangular set in $R[X_1, ..., X_n]$ with degrees $d_1, ..., d_n$, where R is a ring, we will let $\mathsf{MT}(d_1, ..., d_n)$ denote an upper bound on the number of operations in R needed to perform multiplication modulo $\boldsymbol{T}$. If R is a field, we let $\mathsf{IT}(d_1, ..., d_n)$ denote an upper bound for the cost of inversion modulo $\boldsymbol{T}$ (when possible).

  These operations are less straightforward than in the univariate case. Known upper bounds are $\mathsf{MT}(d_1, ..., d_n) = \mathcal{O}(4^n\, d \log(d)\, \mathrm{loglog}(d))$ where $d = d_1 \cdots d_n$ and $\mathsf{IT}(d_1, ..., d_n) = \mathcal{O}(c^n\, d \log(d)^3)$, for some (large) constant c [DMSX06, DJMS08, LMS09].

We seize the occasion to state our first result. We propose an algorithm, which modifies slightly the algorithm of [LMS09], that saves a recursive call.

Proposition 2. *The multiplication modulo a triangular set $\boldsymbol{T}$ of multidegree $(d_1, ..., d_n)$ can be done in $\mathsf{MT}(d_1, ..., d_n) = \mathcal{O}(3^n\, d \log(d)\, \mathrm{loglog}(d))$ operations in R.*

Finally, we will make the following assumption on the representation of the input system: we assume that $\boldsymbol{f}$ is given by a straight-line program with inputs $t, X_1, ..., X_n$ and n outputs corresponding to $f_1, ..., f_n$, using operations in $\{+, -, \times\}$. We let L be the size of this straight-line program.

With this notation being introduced, we can state our first result, which deals with lifting univariate representations.

Theorem 3. *Let $\boldsymbol{f} = (f_1, \ldots, f_n)$ be a polynomial system in $k[t][X_1, \ldots, X_n]$ given by a straight-line program Γ of size L. Suppose that $\boldsymbol{f}$ satisfies assumption* (H) *and admits a univariate representation $\boldsymbol{P}$ with coefficients in $k(t)$ and degree d.*

Let further t_0 be in k, let $\mathfrak{m} = (t - t_0)$, and suppose that assumption $(\mathsf{H}')_{\boldsymbol{P}, t_0}$ holds. Given $\boldsymbol{P}_1 = \boldsymbol{P} \bmod \mathfrak{m}$, one can compute $\boldsymbol{P}_N = \boldsymbol{P} \bmod \mathfrak{m}^N$ in time

$$\mathcal{O}((\mathsf{M}(d) \log(d) + n^\Omega \mathsf{M}(d)) + (L \mathsf{R}(N) + (n^2 + n L) N) \mathsf{M}(d)).$$

In the running time, the first term corresponds to the inversion of the Jacobian matrix of $\boldsymbol{f}$ modulo $\boldsymbol{P}_1$, with coefficients in k; this allows us to initialize the lifting. Then, to reach precision N, most of the work consists in relaxed multiplications for power series in $k[[t]]$ at precision N, coupled with polynomial multiplication in the variable Z in degree d. The overall cost is quasi-linear in N.

The other known algorithm to perform this kind of task is a suitable version of Newton iteration, introduced in [GLS01, HMW01]. To compute the same output, it runs in time

$$\mathcal{O}((\mathsf{M}(d) \log(d) + n^\Omega \mathsf{M}(d)) + (n L + n^\omega) \mathsf{M}(N) \mathsf{M}(d)).$$

As the term $n^\omega \ \mathsf{M}(N) \ \mathsf{M}(d)$ suggests, this algorithm requires one to do matrix multiplications, with entries that are univariate polynomials of degree d, having themselves power series coefficients of precision N. Our solution requires no such matrix multiplication.

On the other hand, Newton's iteration can use plain power series multiplication, which is slightly faster than the online product used in our algorithm. Thus, there exists a trade-off between the dependencies in n and N for both algorithms.

Let us next turn to the case of triangular representations. Our main result for this problem is the following.

Theorem 4. *Let $\boldsymbol{f} = (f_1, \ldots, f_n)$ be a polynomial system in $k[t][X_1, \ldots, X_n]$ given by a straight-line program Γ of size L. Suppose that $\boldsymbol{f}$ satisfies assumption* (H) *and admits a triangular representation $\boldsymbol{T}$ with coefficients in $k(t)$ and multidegree $(d_1, \ldots, d_n)$.*

Let further t_0 be in k, let $\mathfrak{m} = (t - t_0)$, and suppose that assumption $(\mathsf{H}')_{\boldsymbol{T}, t_0}$ holds. Given $\boldsymbol{T}_1 = \boldsymbol{T} \bmod \mathfrak{m}$, one can compute $\boldsymbol{T}_N = \boldsymbol{T} \bmod \mathfrak{m}^N$ in time

$$\mathcal{O}((\mathsf{IT}(d_1, \ldots, d_n) + n^\Omega \mathsf{MT}(d_1, \ldots, d_n)) + (n L \mathsf{R}(N) + n^2 N) \mathsf{MT}(d_1, \ldots, d_n)).$$

Once more, this is quasi-linear in the precision N. For comparison, an adaptation of Newton's iteration for triangular sets [Sch02] computes the same output in time

$$\mathcal{O}((\mathsf{IT}(d_1, \ldots, d_n) + n^\Omega \mathsf{MT}(d_1, \ldots, d_n)) + (n L + n^\omega) \mathsf{M}(N) \mathsf{MT}(d_1, \ldots, d_n)).$$

Thus, as in the univariate case, the relaxed approach avoids matrix multiplication with multivariate entries; the price to pay is that we have to replace classical power series multiplication by its relaxed variant.

A univariate representation can be seen as a particular case of a triangular one, through the introduction of an extra unknown Z, and the equation expressing Z as a linear form in $X_1, \ldots, X_n$. However, the result in our first theorem is not obtained by specializing the triangular case to the univariate one; further considerations are actually needed.

Let us now give the outline of the proofs of these two theorems. We apply the fundamental theorem concerning relaxed recrusive power series that states that the computation of a recursive power series $y = \Phi(y)$ at precision N takes the time necessary to evaluate Φ at y at the same precision by a shifted algorithm [Wat89, Hoe02, BL12]. The idea behind this theorem is simple : at the Nth step, we know the coefficients $y_0, \ldots, y_{N-1}$ of y and we have already computed the coefficients $\Phi(y)_i$ for $0 \leqslant i < N$. Because a shifted algorithm for Φ reads at most the coefficients $y_0, \ldots, y_{N-1}$ to compute $\Phi(y)_N$, we can compute one more coefficient of $\Phi(y)$. Then we write the new coefficient $y_N = \Phi(y)_N$ of the input. And we can continue the process.

Our contribution consists in finding a recursive equation of which the triangular set we seek to lift is a fixed point. More precisely, the challenge was to find a shifted algorithm that computes an operator Φ satisfying $y = \Phi(y)$.

The computer algebra software MATHEMAGIX [HLM+02] provides a C++ library named ALGEBRAMIX implementing the relaxed recursive power series framework [Hoe02, Hoe07, BHL11, BL12]. This representation allows the computation of recursive power series. The efficiency of this approach is partially due to the relaxed multiplication algorithm, which gives a fast method for multiplication in this representation. On this basis, we implemented the lifting of univariate representations over the power series ring $\mathbb{F}_p[[t]]$ for both our relaxed approach and Newton iteration.

Our implementation is available in the files `lift_series.hpp` and `lifting_fiber_relaxed.hpp` in the C++ library GEOMSOLVEX of MATHEMAGIX. Our relaxed lifting algorithm is connected to the implementation of the geometric resolution algorithm available in MATHEMAGIX, with the help of G. Lecerf.

We now give some implementation details:

- For the multiplication of polynomials of power series in $\mathbb{F}_p[[t]][Z]$, we first converted them to power series of polynomials in $\mathbb{F}_p[Z][[t]]$. Then, the relaxed multiplication algorithm reduces to multiplications of finite precision power series of polynomials, that is polynomials of polynomials in $\mathbb{F}_p[Z][t]$. We classically used a Kronecker substitution to reduce these products to multiplications of univariate polynomials in $\mathbb{F}_p[Z]$. Univariate polynomial multiplication uses FFT techniques.

- For the matrix multiplication used in Newton iteration, whose entries are polynomials reduced modulo a univariate Q, we delayed the reductions until after the matrix multiplication to reduce their numbers from n^3 to n^2.

We report here timings of our implementation (in milliseconds), which are measured using one core of an INTEL XEON X5650 at 2.67 GHz running LINUX 64 bits, GMP 5.0.2 [G+91] and setting $p = 16411$ (a 15-bit prime number).

Before considering lifting itself, we start by giving some comparison of timings between the relaxed and classical (zealous) product in $\mathbb{F}_p[[t]][Z]$, depending on the degree in Z and the precision N of power series.

	Degree 32 in Z		Degree 64 in Z		Degree 128 in Z		Degree 256 in Z	
N	zealous	relaxed	zealous	relaxed	zealous	relaxed	zealous	relaxed
8	0	0	0	1	1	1	2	4
16	0	1	0	3	1	5	3	11
32	0	3	1	7	2	14	6	34
64	1	8	3	18	6	41	12	100
128	3	21	6	49	12	110	30	270
256	6	56	12	130	29	300	70	700
512	12	150	30	340	71	790	170	1800
1024	29	370	71	860	170	2000	340	4500
2048	72	920	170	2100	350	4800	750	11000

Table 1. Timings of zealous and relaxed multiplication in $(\mathbb{F}_p[[t]])[Z]$

We observe that the ratio of the timings between the relaxed and zealous algorithms grows as $\log(N)$, as predicted by the estimate $\mathsf{R}(N) = \mathcal{O}(\mathsf{M}(N)\log(N))$ [FS74, Hoe02].

We applied our algorithm on two families of polynomial systems of low complexity of evaluation. The KATSURA polynomials systems comes from a problem of magnetism in physics [Kat90]. We have added some power series coefficients to the original system in order to have a non trivial power series univariate representation. The system KATSURA-n has $n+1$ unknowns $X_0, \ldots, X_n$ and $n+1$ equations:

$$\text{for } 0 \leqslant m < n, \quad \sum_{\ell=-n}^{n} X_{|\ell|}\, X_{|m-\ell|} = \alpha_m\, X_m$$

and $X_0 + 2\sum_{\ell=1}^{n} X_\ell = \alpha_n$ where the α_i are random coefficients in $\mathbb{F}_p[[t]]$.

The other family of polynomial system MULLINFORM-n has n unknowns and n equations of the form

$$(\lambda_1 X_1 + \cdots + \lambda_n X_n)\,(\mu_1 X_1 + \cdots + \mu_n X_n) = \alpha$$

where the λ_i, μ_i and α are random coefficients in $\mathbb{F}_p[[t]]$.

For both these examples, we lift a univariate representation on power series at different precision N. We indicate with a bold font the theoretical bound for the precision of power series required in the lifting step of the geometric resolution algorithm.

	KATSURA-3		KATSURA-4		KATSURA-5		KATSURA-6	
N	zealous	relaxed	zealous	relaxed	zealous	relaxed	zealous	relaxed
2	21	7	75	20	250	58	780	170
4	31	11	106	29	350	78	1100	220
8	**49**	**18**	170	48	550	130	1700	360
16	82	36	**290**	**92**	940	240	1900	700
32	140	74	510	200	**1700**	**530**	5200	1500
64	260	160	970	440	3300	1200	**10000**	**3600**
128	510	360	1900	1000	6600	2800	21000	8600
256	1000	820	4000	2400				
512	2200	1900	8600	5500				

Table 2. Timings in milliseconds of zealous/relaxed lifting of univariate representations for KATSURA-n.

	MULLINFORM-4		MULLINFORM-5		MULLINFORM-6	
N	zealous	relaxed	zealous	relaxed	zealous	relaxed
2	44	16	160	45	520	130
4	64	23	230	63	720	180
8	96	38	340	100	1000	300
16	**150**	**69**	520	180	1700	540
32	230	140	**850**	**380**	2900	1000
64	370	180	1400	780	**5200**	**2300**
128	670	580	2600	1600	9500	4800

Table 3. Zealous/relaxed lifting timings in milliseconds of univariate representations of MULLINFORM-n.

We remark that, on these examples, relaxed algorithms always perform better than zealous algorithms, especially when they are many variables. However, for a given polynomials system, the gap between the two algorithms reduces as the precision gets bigger and we expect zealous algorithms to be faster when the precision tends to infinity.

Bibliography

[ALM99] P. Aubry, D. Lazard and M. Moreno Maza. On the theories of triangular sets. *J. Symbolic Comput.*, 28(1-2):105–124, 1999. Polynomial elimination—algorithms and applications.

[Ber84] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.*, 18(3):147–150, 1984.

[BHL11] J. Berthomieu, J. van der Hoeven and G. Lecerf. Relaxed algorithms for p-adic numbers. *J. Théor. Nombres Bordeaux*, 23(3):541–577, 2011.

[BL12] J. Berthomieu and R. Lebreton. Relaxed p-adic Hensel lifting for algebraic systems. In *Proceedings of ISSAC'12*, pages 59–66. ACM Press, 2012.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comp.*, 9(3):251–280, 1990.

[DJMS08] X. Dahan, X. Jin, M. Moreno Maza and É. Schost. Change of ordering for regular chains in positive dimension. *Theoretical Computer Science*, 392(1-3):37–65, 2008.

[DMS+05] X. Dahan, M. Moreno Maza, É. Schost, W. Wu and Y. Xie. Lifting techniques for triangular decompositions. In *ISSAC'05*, pages 108–115. ACM Press, 2005.

[DMSX06] X. Dahan, M. Moreno Maza, É. Schost and Y. Xie. On the complexity of the D5 principle. In *Transgressive Computing*, pages 149–168. 2006.

[FS74] M. J. Fischer and L. J. Stockmeyer. Fast on-line integer multiplication. *J. Comput. System Sci.*, 9:317–331, 1974.

[G+91] T. Granlund et al. GMP, the GNU multiple precision arithmetic library. 1991. Version 5.0.2.

[GG03] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, Second edition, 2003.

[GH91] M. Giusti and J. Heintz. Algorithmes—disons rapides—pour la décomposition d'une variété algébrique en composantes irréductibles et équidimensionnelles. In *MEGA*, volume 94 of *Progr. Math.*, pages 169–194. Birkhäuser, 1991.

[GHH+97] M. Giusti, J. Heintz, K. Hägele, J. E. Morais, L. M. Pardo and J. L. Montaña. Lower bounds for Diophantine approximations. *J. Pure Appl. Algebra*, 117/118:277–317, 1997. Algorithms for algebra (Eindhoven, 1996).

[GLS01] M. Giusti, G. Lecerf and B. Salvy. A Gröbner free alternative for polynomial system solving. *J. Complexity*, 17(1):154–211, 2001.

[GM89] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using Groebner bases. In *Applied algebra, algebraic algorithms and error-correcting codes (Menorca, 1987)*, volume 356 of *Lecture Notes in Comput. Sci.*, pages 247–257. Springer, Berlin, 1989.

[Hen66] F. C. Hennie. On-line turing machine computations. *Electronic Computers, IEEE Transactions on*, EC-15(1):35–44, 1966.

[HLM+02] J. van der Hoeven, G. Lecerf, B. Mourrain et al. Mathemagix. 2002. SVN Version 7058. Available from `http://www.mathemagix.org`.

[HMW01] J. Heintz, G. Matera and A. Waissbein. On the time-space complexity of geometric elimination procedures. *Appl. Algebra Engrg. Comm. Comput.*, 11(4):239–296, 2001.

[Hoe02] J. van der Hoeven. Relax, but don't be too lazy. *J. Symb. Comput.*, 34(6):479–542, 2002.

[Hoe07] J. van der Hoeven. New algorithms for relaxed multiplication. *J. Symbolic Comput.*, 42(8):792–802, 2007.

[Hoe11] J. van der Hoeven. From implicit to recursive equations. Technical Report, HAL, 2011. `http://hal.archives-ouvertes.fr/hal-00583125/fr/`.

[Kal92] E. Kaltofen. On computing determinants of matrices without divisions. In P. S. Wang, editor, *Proc. 1992 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'92)*, pages 342–349. New York, N. Y., 1992. ACM Press.

[Kal93] M. Kalkbrener. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comp.*, 15:143–167, 1993.

[Kat90] S. Katsura. Spin glass problem by the method of integral equation of the effective field. *New Trends in Magnetism*, :110–121, 1990.

[KV04] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Comput. Complexity*, 13(3-4):91–130, 2004.

[Laz91] D. Lazard. A new method for solving algebraic systems of positive dimension. *Discrete Appl. Math.*, 33(1-3):147–160, 1991. Applied algebra, algebraic algorithms, and error-correcting codes (Toulouse, 1989).

[LMS09] X. Li, M. Moreno Maza and É. Schost. Fast arithmetic for triangular sets: from theory to practice. *J. Symbolic Comput.*, 44(7):891–907, 2009.

[Rit66] J. F. Ritt. *Differential algebra*. Dover Publications Inc., New York, 1966.

[Rou99] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Appl. Algebra Engrg. Comm. Comput.*, 9(5):433–461, 1999.

[Sch02] É. Schost. Degree bounds and lifting techniques for triangular sets. In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 238–245. New York, 2002. ACM.

[Sch03a] É. Schost. Computing parametric geometric resolutions. *Appl. Algebra Engrg. Comm. Comput.*, 13(5):349–393, 2003.

[Sch03b] Éric Schost. Complexity results for triangular sets. *J. Symbolic Comput.*, 36(3-4):555–594, 2003. International Symposium on Symbolic and Algebraic Computation (ISSAC'2002) (Lille).

[Sto10] A. Stothers. *On the Complexity of Matrix Multiplication*. PhD thesis, University of Edinburgh, 2010.

[Vas11] V. Vassilevska Williams. Breaking the Coppersmith-Winograd barrier. 2011.

[Wat89] S. Watt. A fixed point method for power series computation. In P. Gianni, editor, *Symbolic and Algebraic Computation*, volume 358 of *Lecture Notes in Computer Science*, pages 206–217. Springer Berlin / Heidelberg, 1989.

[Wu84] W. J. Wu. Basic principles of mechanical theorem proving in elementary geometries. *J. Systems Sci. Math. Sci.*, 4(3):207–235, 1984.