Relaxed Hensel lifting of triangular sets*

by Romain Lebreton

LIRMM
Université Montpellier II

*. This document has been written using the GNU $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ text editor (see www.texmacs.org).

Context: root lifting

Previous work. [Berthomieu, L. '12]

- Relaxed algorithm for lifting **one** root of a polynomial system over $k[[x]], \mathbb{Z}_p$
- Comparison to existing Newton iteration:

  $\rightsquigarrow$ Complexity improvements

Today's talk.

- Relaxed algorithms for lifting **all** roots of a polynomial system over $k[[x]], \mathbb{Z}_p$
- Comparison to existing Newton iteration

Representation of all roots

Notations.

– $\mathcal{I}$ zero-dimensional ideal spanned by $\boldsymbol{f} = (f_1, \ldots, f_r) \in K[X_1, \ldots, X_r]$

– Quotient algebra $\mathbb{A} := K[X_1, \ldots, X_r]/\mathcal{I}$

Definition.

Triangular representation	Univariate representation
$\mathbf{T} = (T_1, \ldots, T_r)$ $T_i \in K[X_1, \ldots, X_i]$ monic T_i reduced w.r.t. $(T_1, \ldots, T_{i-1})$	*Primitive* linear form Λ Minimal polynomial Q of Λ Parametrizations (S_i)
$K[X_1]/(T_1)$ $\vdots$ $K[X_1, \ldots, X_j]/(T_1, \ldots, T_j)$ $\vdots$ $\mathbb{A} := K[X_1, \ldots, X_r]/(T_1, \ldots, T_r)$	$\mathbb{A} \simeq k[T]/(Q)$ $X_i \longmapsto S_i(T)$ $\Lambda \longleftarrow T$

Lifting of triangular representations

Example of lifting.

Let $\boldsymbol{f} = (f_1, f_2)$ in $\mathbb{Z}[X_1, X_2]$ with

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

Input: Triangular set $\boldsymbol{T}_0$ modulo 7 of $\boldsymbol{f} \in \mathbb{Z}[X_1, X_2]$ with

$$\boldsymbol{T}_0 := (X_1^2 + 5\, X_1,\quad X_2^2 + 3\, X_1\, X_2 + 2\, X_2 + 4 X_1 + 6)$$

Output: Triangular set $\boldsymbol{T}_2$ modulo 7^3 of $\boldsymbol{f}$

$$(X_1^2 + (5 + 5 \cdot 7 + 6 \cdot 7^2)\, X_1 + (7 + 7^2),\quad X_2^2 + (3 + 2 \cdot 7 + 7^2)\, X_1\, X_2 + (2 + 3 \cdot 7 + 5 \cdot 7^2)\, X_2 + (4 + 5 \cdot 7^2)\, X_1 + (6 + 3 \cdot 7 + 6 \cdot 7^2))$$

Remark. The precision is enough to recover the triangular set of $\boldsymbol{f}$

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

Lifting of triangular representations

General context.

R ring, (p) principal ideal.

R_p: completion of R for p-adic valuation.

Input.

- $\boldsymbol{f}$ polynomial system in $R[X_1, \ldots, X_r]$ given as an *s.l.p.*
- Triangular representation $\boldsymbol{T}_0$ of $\boldsymbol{f}$ over $R/(p)$

+ Hensel hypothesis: $\mathrm{Jac}\,\boldsymbol{f}_0$ is invertible modulo $\boldsymbol{T}_0$

Output.

- Triangular representation $\boldsymbol{T}$ of $\boldsymbol{f}$ over R_p at precision N

In this talk, $R = k[x]$, $p = (x)$, $R_p = k[[x]]$

Contribution

Existing body of work.

Newton-like iteration for triangular representation lifting over R_p [Schost '02]

My contribution.

- First **relaxed** algorithm for triangular representation lifting over R_p
- Complexity improvements compared to Newton's iteration
- Implementation in the computer algebra software Mathemagix

Applications of lifting.

- Computation of triangular representations over $k(x)$ or $\mathbb{Q}$
- Repercutions on the complexity of
  - the geometric resolution algorithm [Giusti *et al.* '01], [Heintz *et al.* '01]
  - the triangular set change of order algorithm [Dahan *et al.* '08]

Relaxed / Online algorithms

Definition of *relaxed* (or *on-line*) algorithms. [Hennie '66]

The ith coefficient of the output is written before the $i+1$th, $i+2$th, ... coefficients of the inputs are read.

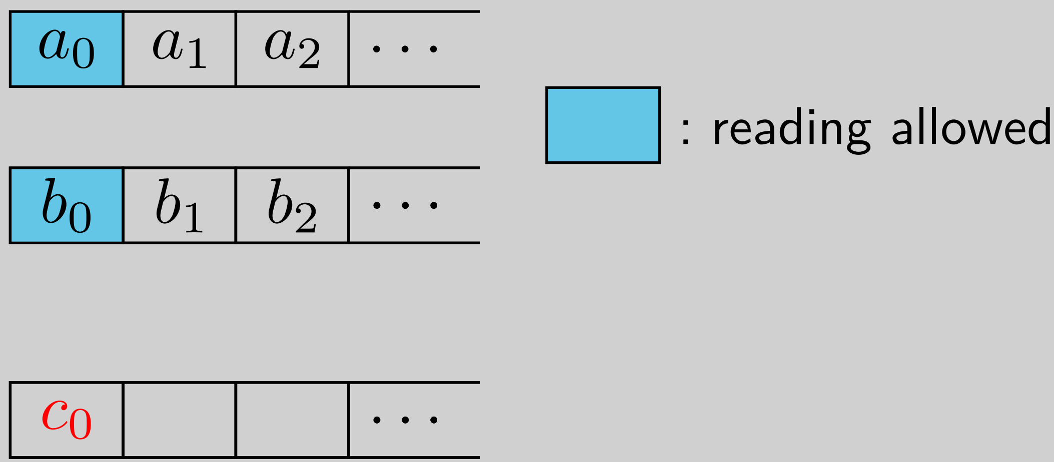
That is

$$a = \sum_{i \geqslant 0} a_i x^i$$

$$b = \sum_{i \geqslant 0} b_i x^i$$

$$\downarrow f$$

$$c = f(a,b) = \sum_{i \geqslant 0} c_i x^i$$



Definition of *relaxed* (or *on-line*) algorithms. [Hennie '66]

The ith coefficient of the output is written before the $i+1$th, $i+2$th, ... coefficients of the inputs are read.

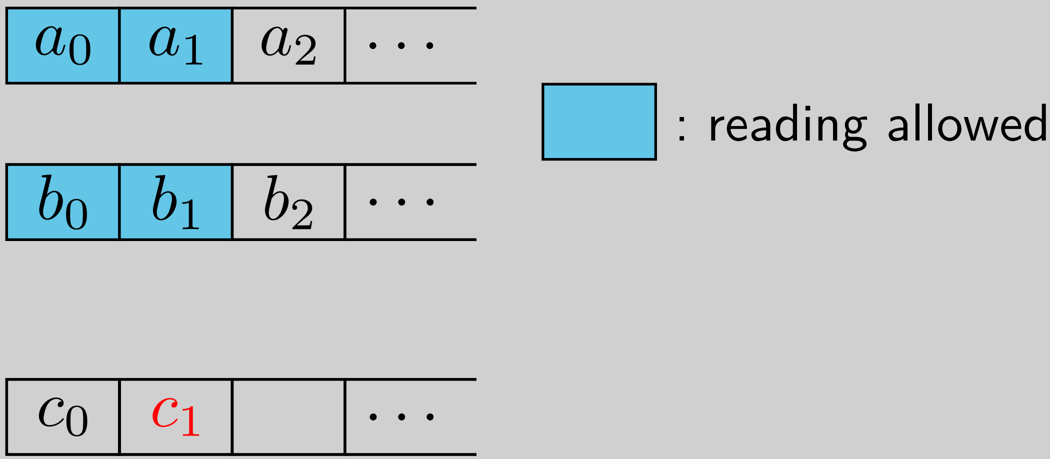
That is

$a = \sum_{i \geqslant 0} a_i x^i$

$b = \sum_{i \geqslant 0} b_i x^i$

$\downarrow f$

$c = f(a,b) = \sum_{i \geqslant 0} c_i x^i$



Relaxed / Online algorithms

Definition of *relaxed* (or *on-line*) algorithms. [Hennie '66]

The ith coefficient of the output is written before the $i+1$th, $i+2$th, ... coefficients of the inputs are read.

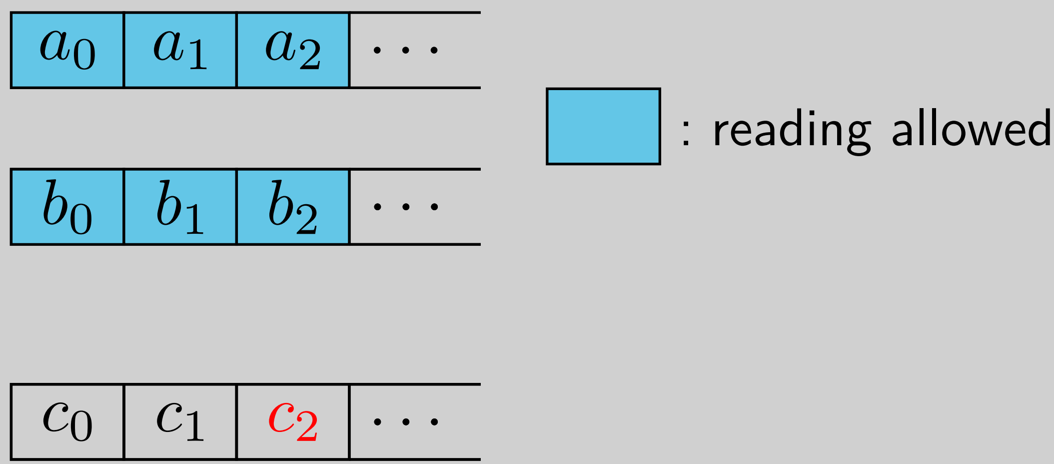
That is

$a = \sum_{i \geqslant 0} a_i x^i$

$b = \sum_{i \geqslant 0} b_i x^i$

$\downarrow f$

$c = f(a, b) = \sum_{i \geqslant 0} c_i x^i$



Relaxed / Online algorithms

Definition of *relaxed* (or *on-line*) algorithms. [Hennie '66]

The ith coefficient of the output is written before the $i+1$th, $i+2$th, $\ldots$ coefficients of the inputs are read.

Definition of *shifted* algorithms.

The ith coefficient of the output is written before the ith, $i+1$th, $\ldots$ coefficients of the inputs are read.

Relaxed arithmetic

Relaxed addition.

The naive addition algorithm is *relaxed*:

$\texttt{for}\ i\ \texttt{from}\ 0\ \texttt{to}\ N\ \texttt{do}\ c_i := a_i + b_i$

Problem.

Fast polynomial multiplication algorithms (*e.g.* Karatsuba, FFT) are *not* relaxed.

Relaxed multiplication. [Fischer, Stockmeyer '74]

Let

– $\mathsf{M}(N)$: cost of $a \times b$ in $k[[x]]$ at precision N.

– $\mathsf{R}(N)$: cost of $a \times b$ in $k[[x]]$ at precision N by a relaxed algorithm.

Then

$$\mathsf{R}(N) = \mathcal{O}(\mathsf{M}(N) \log N) = \tilde{\mathcal{O}}(N).$$

Relaxed recursive power series

Definition.

A power series $y = \sum_{i \geqslant 0} y_i x^i \in k[[x]]$ is **recursive** if there exists $\Phi \in k[Y]$ such that

- $y = \Phi(y)$
- $\Phi(y)_n$ only depends on $y_0, \ldots, y_{n-1}$

Fundamental Theorem. [Watt '88], [Hoeven '02], [Berthomieu, L. '12]

Let $y \in k[[x]]$ such that $y = \Phi(y)$.

Given $y_0 \in k$ and $\Phi \in k[Y]$, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$ by a **shifted algorithm**.

Reduction to the relaxed framework

Work remaining to have a relaxed lifting algorithm.

- Express the triangular representation $\boldsymbol{T}$ as a **recursive** power series:

$$\boldsymbol{T} = \Phi(\boldsymbol{T}) := \underbrace{\boldsymbol{T}_0 + [B_0^{-1}(\boldsymbol{f} - (B - B_0)(\boldsymbol{T} - \boldsymbol{T}_0)) \operatorname{rem} \boldsymbol{T}_0]}_{\text{the coefficient in } x^n \text{ involves only } \boldsymbol{T}_0, \ldots, \boldsymbol{T}_{n-1}}$$

where $B \in \mathcal{M}_r(k(x)[X_1, \ldots, X_r])$ is the *quotient matrix* s.t. $\boldsymbol{f} = B \cdot \boldsymbol{T}$.

- Giving a shifted algorithm for Φ:

$$\boldsymbol{T} = \Phi(\boldsymbol{T}) = \boldsymbol{T}_0 + [B_0^{-1}(\boldsymbol{f} - x^2(\delta(B) \cdot \delta(\boldsymbol{T}))) \operatorname{rem} \boldsymbol{T}_0]$$

where $\delta(B) := (B - B_0)/x$.

$\Rightarrow$ Relaxed lifting algorithm for $\boldsymbol{T}$ over R_p

Complexity results

Complexity improvements for univariate representations.

	Newton iteration	Relaxed algorithms
Univariate representations	$\mathcal{O}(r\,L + r^{\omega})\,\mathsf{M}(N)\,\mathsf{M}(d) + \mathcal{O}(N)$	$\mathcal{O}(L)\,\mathsf{R}(N)\,\mathsf{M}(d) + \mathcal{O}(N)$

r number of unknowns,

N precision of the output,

L complexity of evaluation,

d degree of the univariate representation

Timings

Implementation in the computer algebra software MATHEMAGIX.

N	KATSURA-3 Newton	KATSURA-3 relaxed	KATSURA-4 Newton	KATSURA-4 relaxed	KATSURA-5 Newton	KATSURA-5 relaxed	KATSURA-6 Newton	KATSURA-6 relaxed
2	0.02	0.007	0.08	0.02	0.25	0.06	0.8	0.17
4	0.03	0.01	0.10	0.03	0.35	0.08	1.1	0.22
8	0.05	0.02	0.17	0.05	0.55	0.13	1.7	0.36
16	0.08	0.04	0.29	0.09	0.9	0.24	1.9	0.70
32	0.14	0.07	0.51	0.20	1.7	0.53	5.2	1.5
64	0.26	0.16	1.0	0.44	3.3	1.2	10	3.6
128	0.51	0.36	1.9	1	6.6	2.8	21	8.6

Table. Timings in seconds of lifting of univariate representations over $\mathbb{F}_{16411}[[x]]$ for KATSURA-n.

Conclusion

Two paradigms for lifting.

Newton iteration	Relaxed algorithms
implicit equations $\boldsymbol{P}(\boldsymbol{Y}) = 0$	recursive equations $\boldsymbol{Y} = \boldsymbol{\Phi}(\boldsymbol{Y})$

Contributions.

- Relaxed algorithms for lifting **all** roots of a polynomial system over $k[[x]], \mathbb{Z}_p$
- Complexity improvements w.r.t. existing Newton's iteration
- Implementation in Mathemagix

Perspectives.

- Lift of a polynomial root under more general hypotheses (multiplicities);
- Study hybrid Newton / relaxed algorithms

Thank you for your attention ;-)