Relaxed algorithms, p-adic lifting and polynomial system solving*

by Romain Lebreton

ARITH Team, LIRMM
Université Montpellier II

BIPOP-CASYS Seminar

Laboratoire Jean Kuntzmann, Grenoble
March 7, 2013

*. This document has been written using the GNU TeXmacs text editor (see www.texmacs.org).

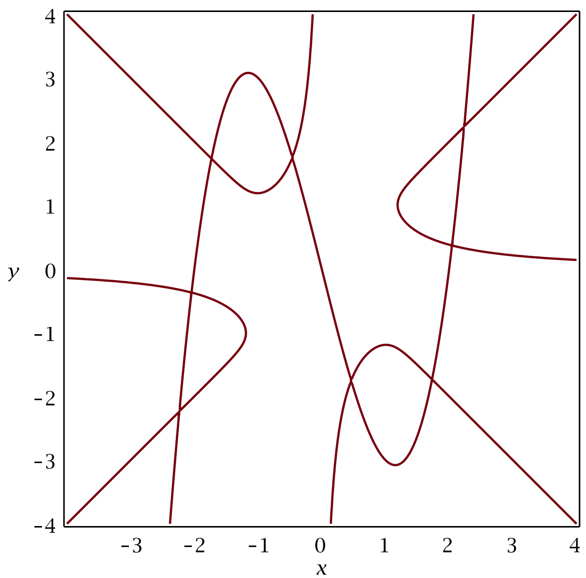
Computer algebra

1. Why is p-adic lifting interesting?
2. Relaxed p-adic lifting
    a. On-line algorithms
    b. Recursive p-adic
    c. Application to:
        i. Linear systems
        ii. Linear q-differential systems
        iii. Polynomial root lifting
        iv. Univariate representation lifting
3. Conclusion

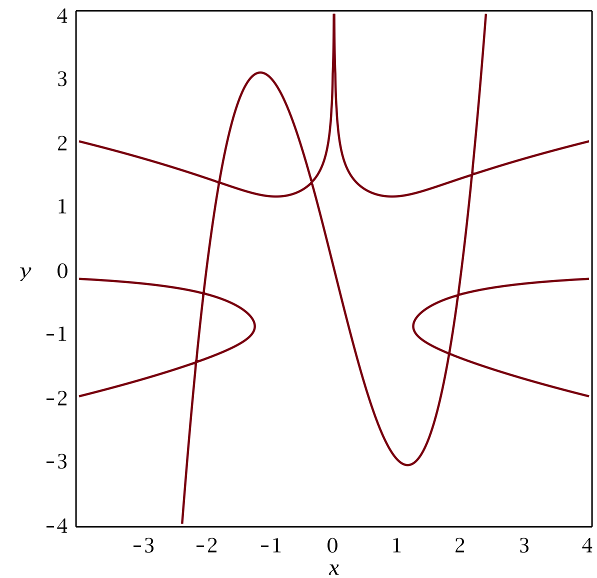
Why modular computation?

$$P_1 = x^5 y^8 + (-x^8 + 4x^6) y^7 - x^7 y^6 + (x^{10} - 4x^8) y^5 + y - x^3 + 4x$$

$$P_2 = -x^2 y^8 + (x^5 - 4x^3) y^7 + x^4 y^4 + (-x^7 + 4x^5) y^3 + y - x^3 + 4x$$



Solutions of $P_1 = 0$



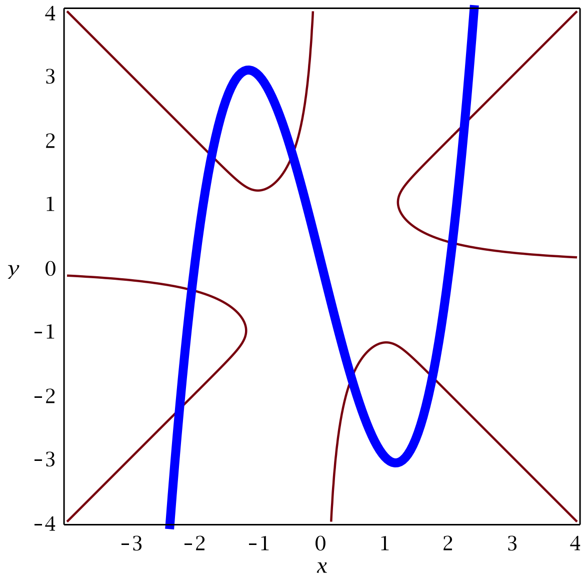
Solutions of $P_2 = 0$

Do they have a curve in common?

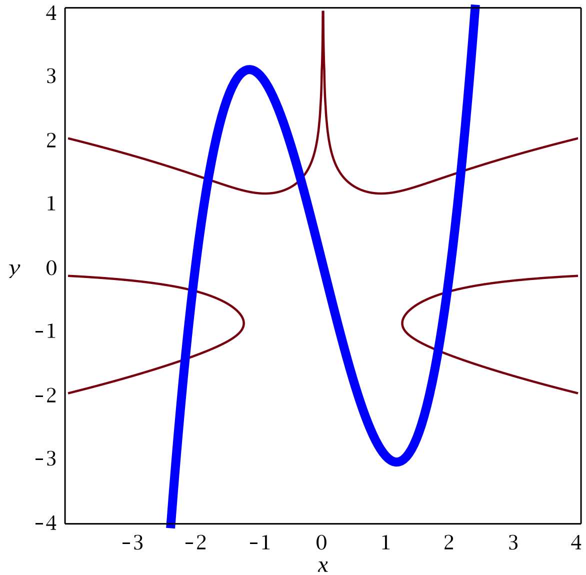
Why modular computation?

$$P_1 = x^5 y^8 + (-x^8 + 4x^6) y^7 - x^7 y^6 + (x^{10} - 4x^8) y^5 + y - x^3 + 4x$$

$$P_2 = -x^2 y^8 + (x^5 - 4x^3) y^7 + x^4 y^4 + (-x^7 + 4x^5) y^3 + y - x^3 + 4x$$



Solutions of $P_1 = 0$



Solutions of $P_2 = 0$

Do they have a curve in common?

YES !

Why modular computation?

$$P_1 = x^5 y^8 + (-x^8 + 4x^6) y^7 - x^7 y^6 + (x^{10} - 4x^8) y^5 + y - x^3 + 4x$$

$$P_2 = -x^2 y^8 + (x^5 - 4x^3) y^7 + x^4 y^4 + (-x^7 + 4x^5) y^3 + y - x^3 + 4x$$

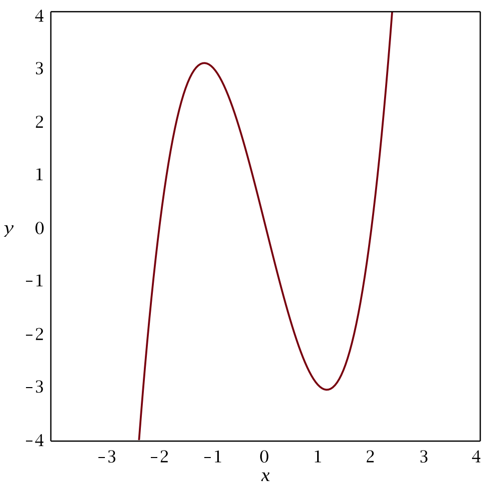
$\downarrow$

Euclid's algorithm
(in $\mathbb{Q}(x)[y]$)

$\downarrow$

$R = y - x^3 + x$

Solutions of $R = 0$



Why modular computation?

$$P_1 = x^5 y^8 + (-x^8 + 4x^6) y^7 - x^7 y^6 + (x^{10} - 4x^8) y^5 + y - x^3 + 4x$$

$$P_2 = -x^2 y^8 + (x^5 - 4x^3) y^7 + x^4 y^4 + (-x^7 + 4x^5) y^3 + y - x^3 + 4x$$

↓

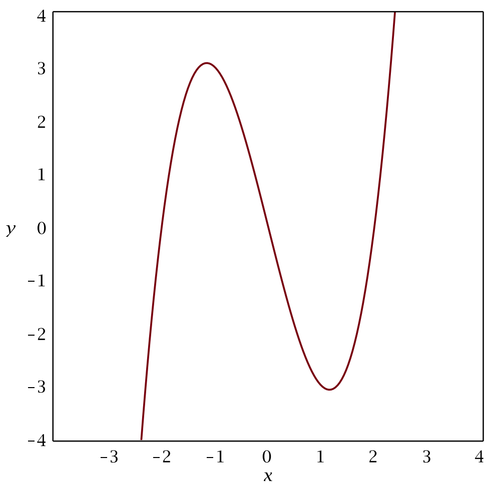
Euclid's algorithm (in $\mathbb{Q}(x)[y]$)

↓

$$R_1 = y^2 + (-x^{34} + x^{33} + 5x^{32} - 5x^{31} - 5x^{30} + 5x^{29} + x^{28} - x^{27} - 4x^{23} + 4x^{22} + 4x^{21} - 13x^{20} + 10x^{19} + 12x^{18} - 18x^{17} + 2x^{16} + 15x^{15} - 13x^{14} - 11x^{13} + 11x^{12} - 5x^{11} - 11x^{10} + 8x^9 + 2x^8 - 6x^7 + 6x^6 + 4x^5 - 4x^4 + 4x^3 + x^2 - x + 1)/(x^{31} - x^{30} - x^{29} + x^{28} - x^{22} + x^{21} + x^{20} - 3x^{19} + 2x^{18} + 3x^{17} - 4x^{16} + 3x^{14} - 3x^{13} - 3x^{12} + 2x^{11} - 2x^{10} - 2x^9 + x^8 - x^7)\, y + (x^{32} - x^{31} - 5x^{30} + 5x^{29} + 4x^{28} - 3x^{27} - x^{26} - 5x^{25} + 7x^{24} + 2x^{23} - 14x^{22} + 10x^{21} + 8x^{20} - 9x^{19} + x^{18} + 4x^{17} - 5x^{16} + x^{15} + 3x^{14} - 8x^{13} + 8x^{12} + 12x^{11} - 22x^{10} + 22x^9 + 18x^8 - 28x^7 + \cdots$$

$$R = y - x^3 + x$$

Solutions of $R = 0$

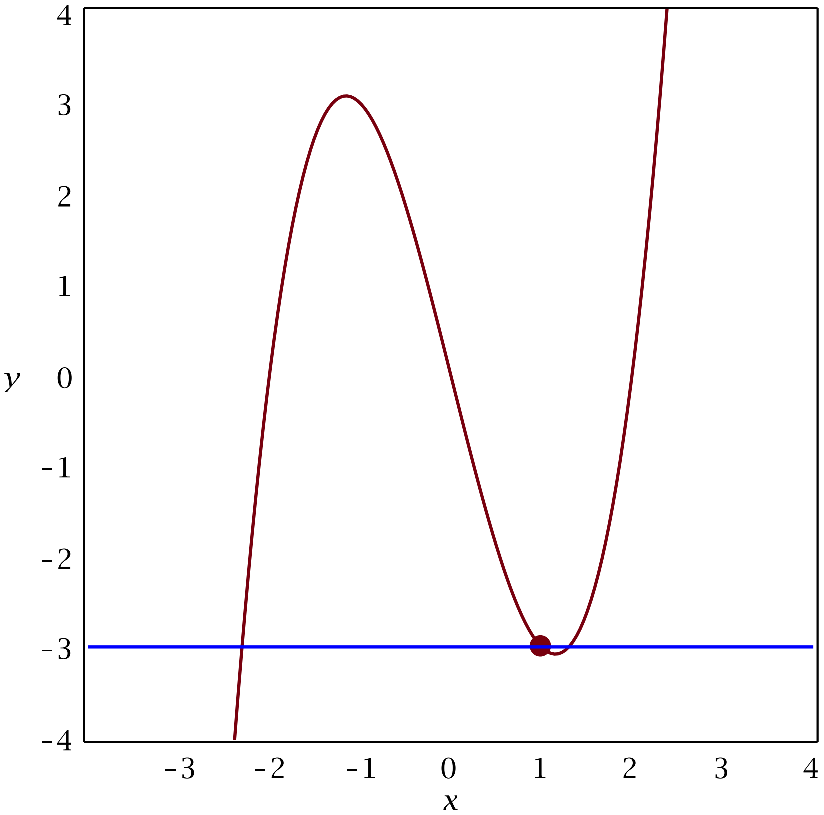


High precision modular computation

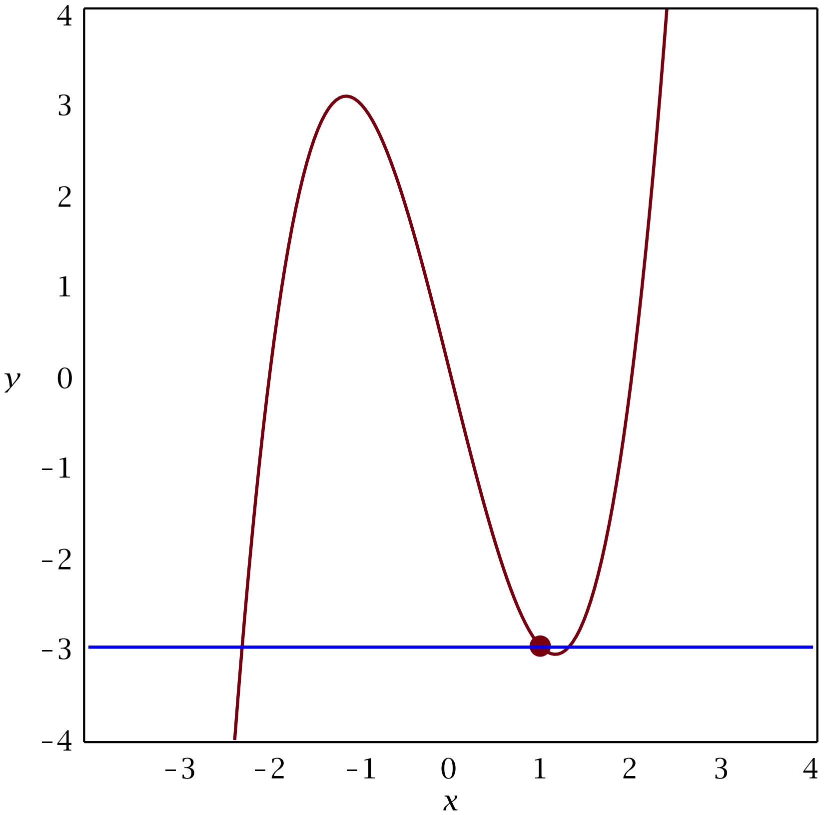
Two different approaches:

multi-modular lifting	Hensel lifting

Precision 1

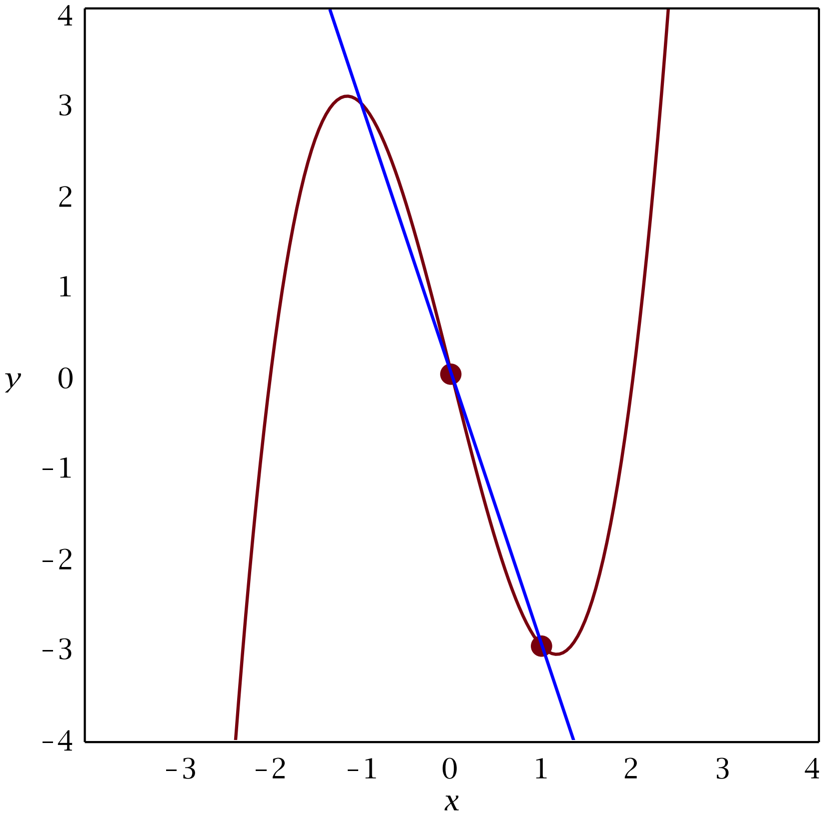
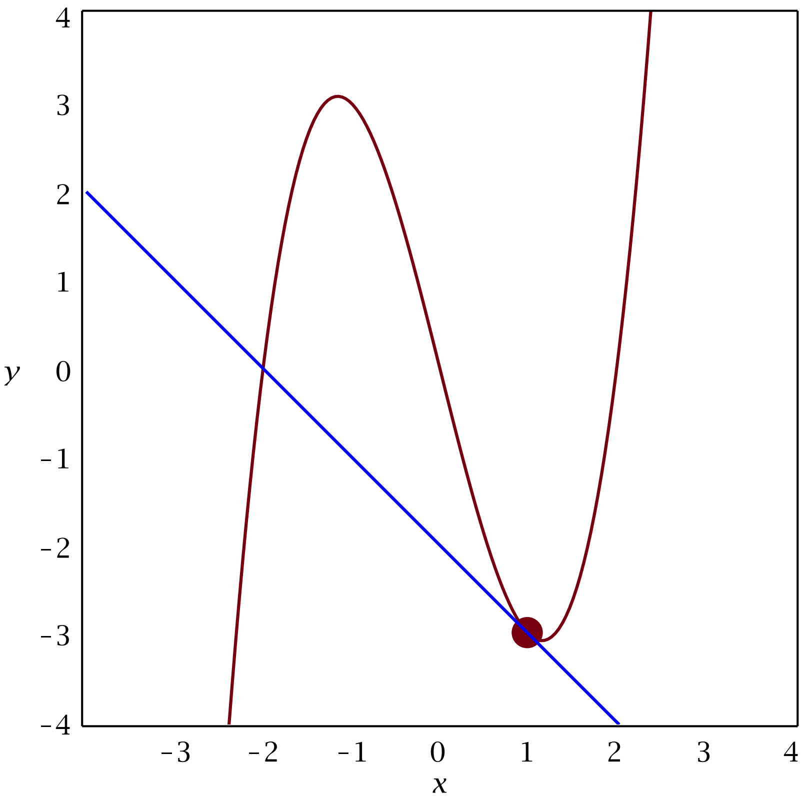


Precision 1

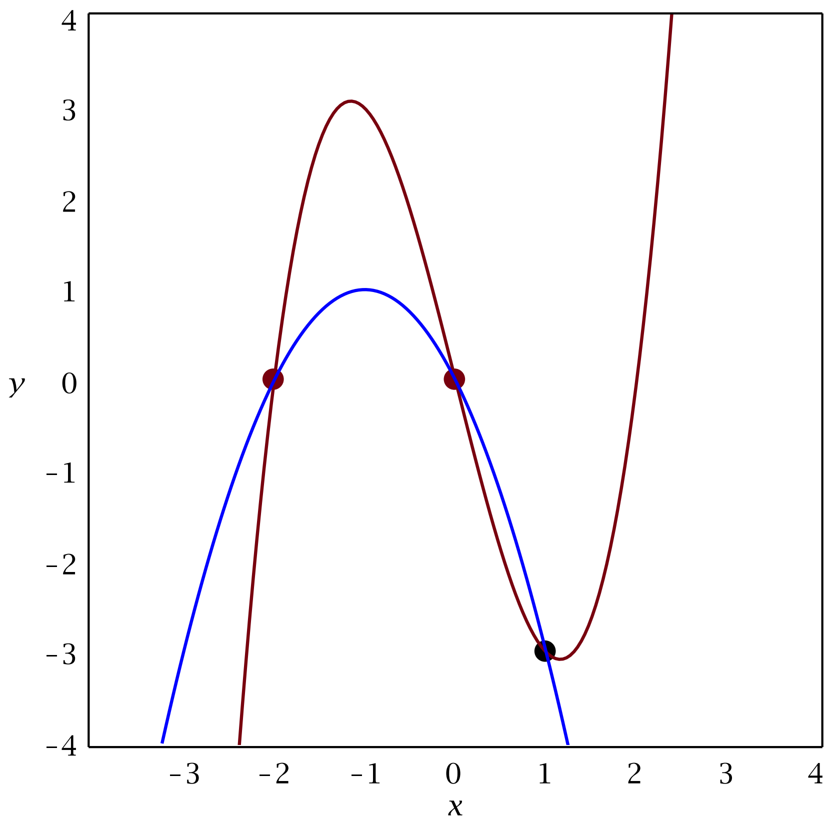
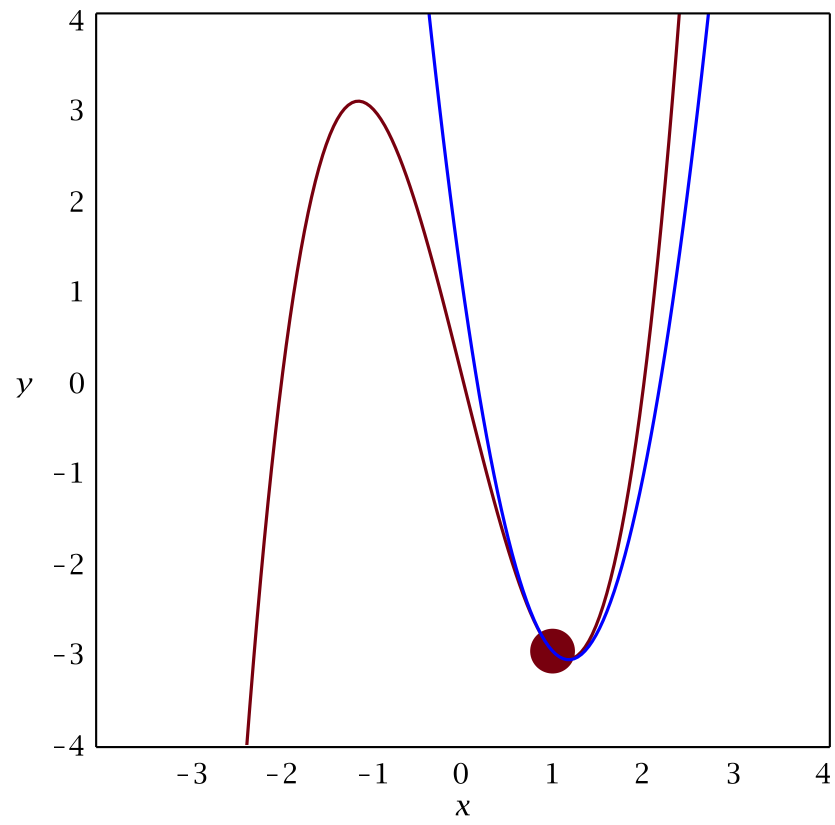


High precision modular computation

Two different approaches:

multi-modular lifting	Hensel lifting
Precision 2	Precision 2
	

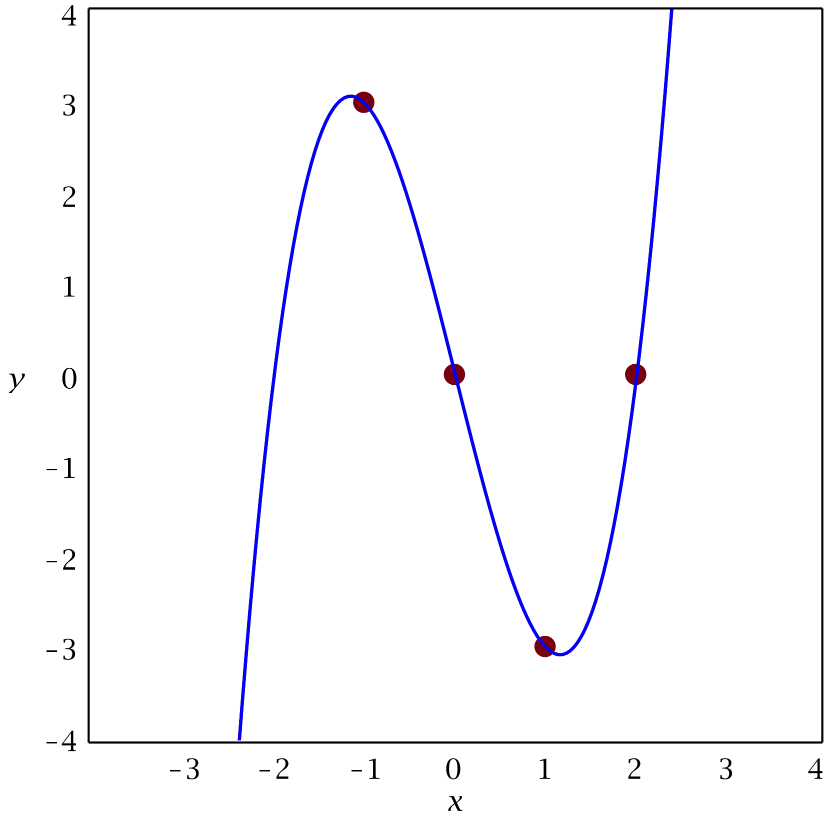
Two different approaches:

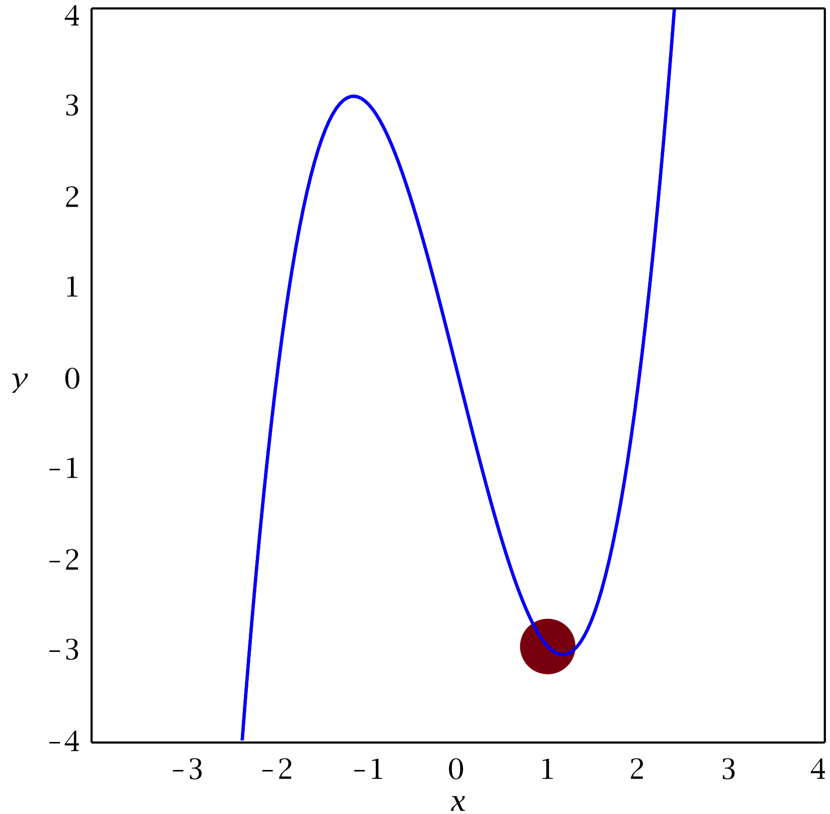
multi-modular lifting	Hensel lifting
Precision 3	Precision 3
	

High precision modular computation

Two different approaches:

multi-modular lifting	Hensel lifting
Precision 4	Precision 4





Plan

1. Why is p-adic lifting interesting?
2. Relaxed p-adic lifting
   a. On-line algorithms
   b. Recursive p-adic
   c. Application to:
      i. Linear systems
      ii. Linear q-differential systems
      iii. Polynomial root lifting
      iv. Univariate representation lifting
3. Conclusion

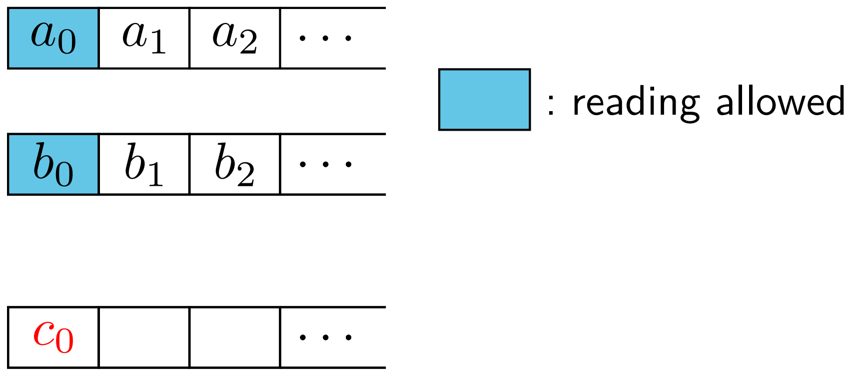
Definition. **(*on-line* or *relaxed* algorithm)** [Hennie '66]

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

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

$\downarrow f$

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



Relaxed algorithms

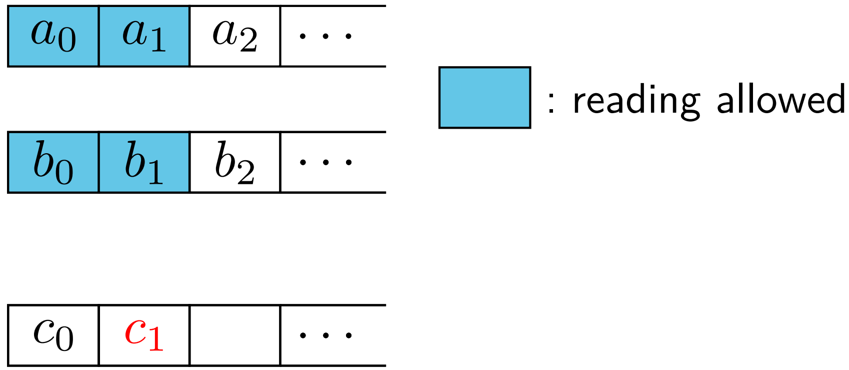
Definition. (*on-line* or *relaxed* algorithm) [Hennie '66]

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

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

$\downarrow f$

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



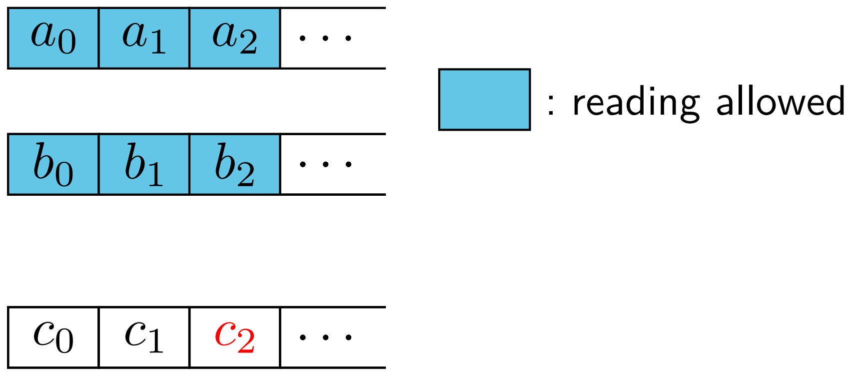
Definition. (*on-line* or *relaxed* algorithm) [Hennie '66]

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

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

$$\downarrow f$$

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



Relaxed algorithms

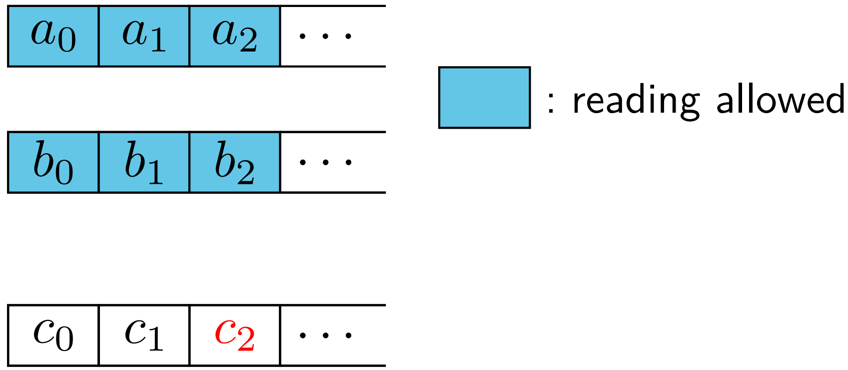
Definition. (*on-line* or *relaxed* algorithm) [Hennie '66]

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

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

$\downarrow f$

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



Half-line *or* semi-relaxed *algorithm* : *condition on one input.*

Off-line *or* zealous *algorithm* : *condition not met.*

Relaxed algorithms

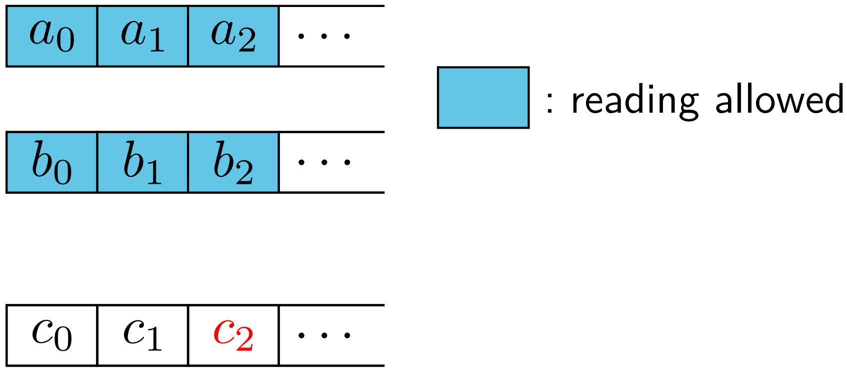
Definition. (*on-line* or *relaxed* algorithm) [Hennie '66]

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

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

$\downarrow f$

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



Half-line *or* semi-relaxed *algorithm* : *condition on one input.*

Off-line *or* zealous *algorithm* : *condition not met.*

Example.

- The naive addition algorithm is on-line:

  for i from 0 to N do $c_i := a_i + b_i$

- The naive multiplication algorithm is on-line:

  for i from 0 to N do $c_i := a_i b_0 + a_{i-1} b_1 + \cdots + a_0 b_i$

Challenge. Find a *quasi-optimal on-line* multiplication algorithm

Relaxed multiplication $c = a \times b$ - step 0

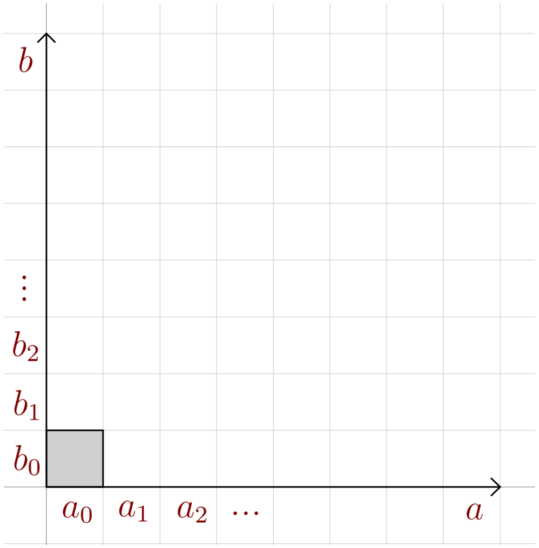


Figure. What we must compute

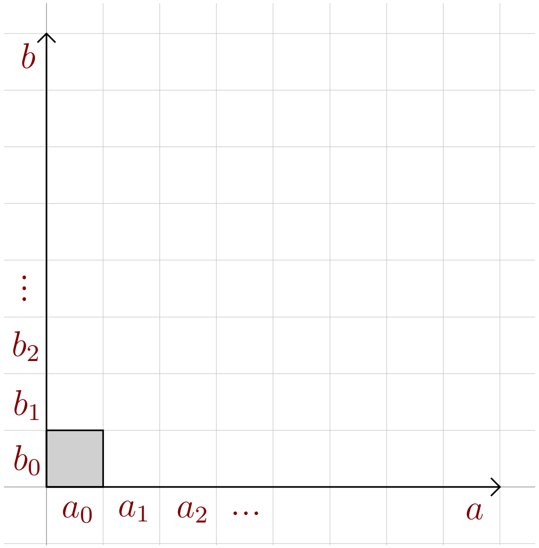


Figure. Minimum knowledge on the input

Figure. What we compute

step 0: $c \; = \; a_0 \, b_0$

Relaxed multiplication $c = a \times b$ - step 1

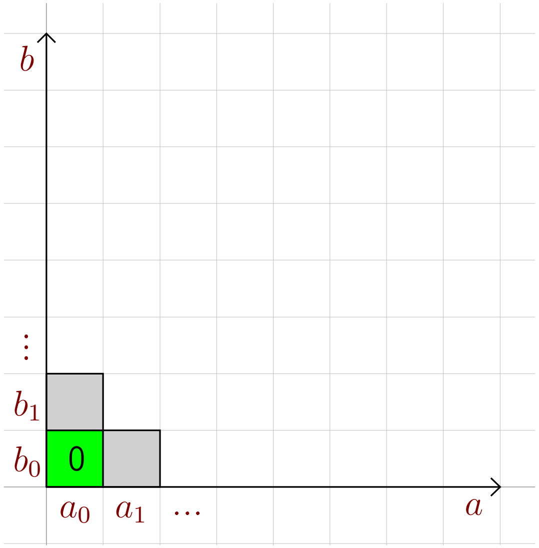


Figure. What we must compute

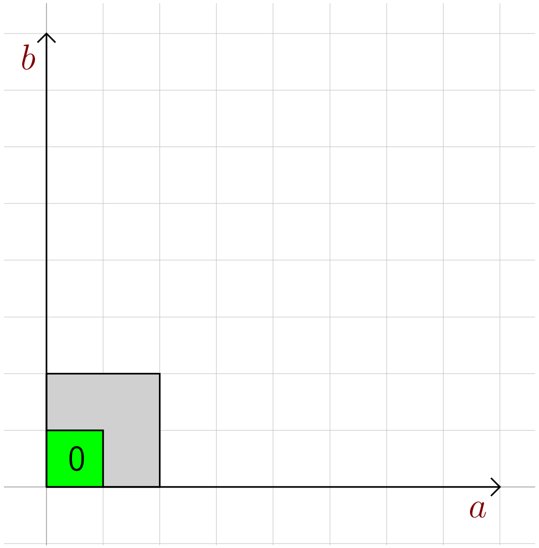


Figure. Minimum knowledge on the input

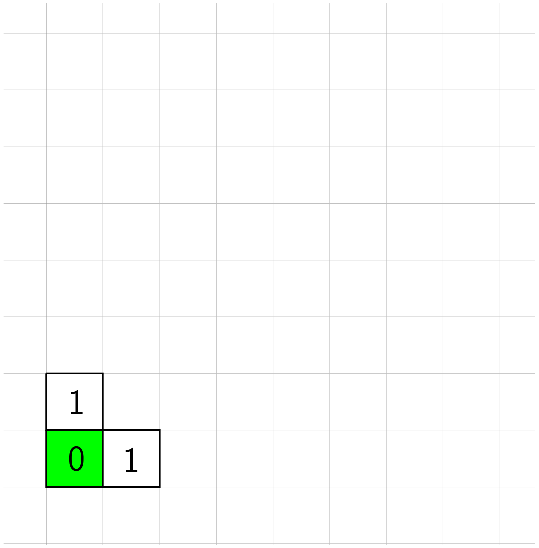


Figure. What we compute

step 0: $c \;=\; a_0\, b_0$
step 1: $c \mathrel{+}= z\;(a_0\, b_1 + a_1\, b_0)$

Relaxed multiplication $c = a \times b$ - step 2

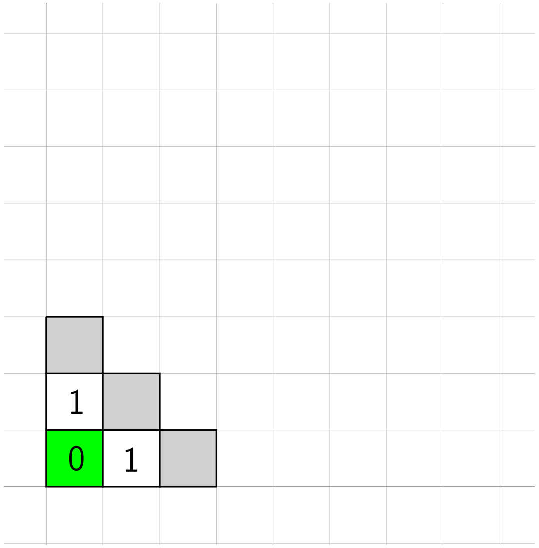


Figure. What we must compute

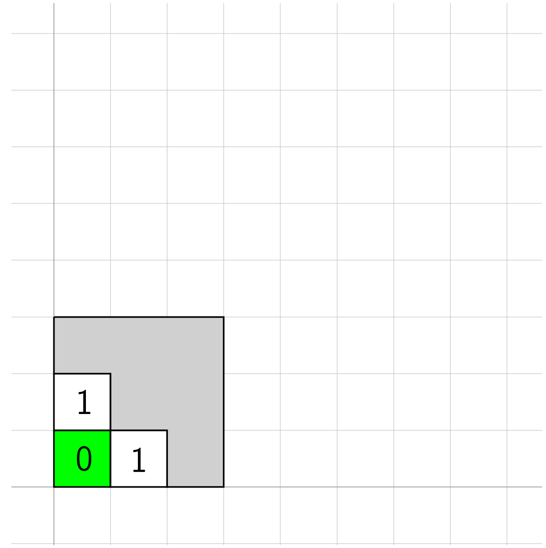


Figure. Minimum knowledge on the input

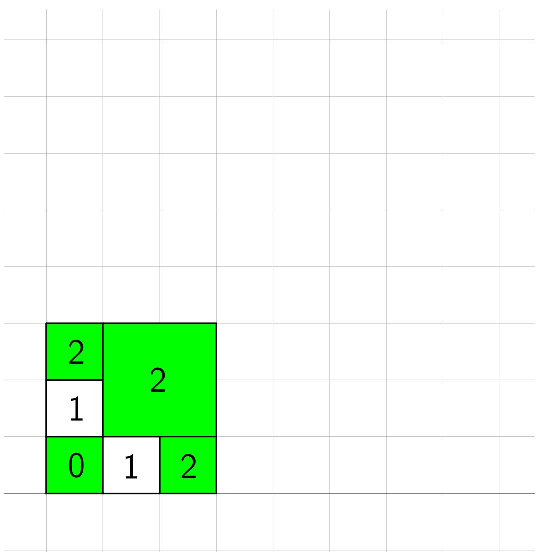


Figure. What we compute

step 0: $c \;=\; a_0\, b_0$

step 1: $c \mathrel{+}= z\, (a_0\, b_1 + a_1\, b_0)$

step 2: $c \mathrel{+}= z^2\, (a_0\, b_2 + a_2\, b_0 + (a_1 + a_2\, z)\, (b_1 + b_2\, z))$

Relaxed multiplication $c = a \times b$ - step 3

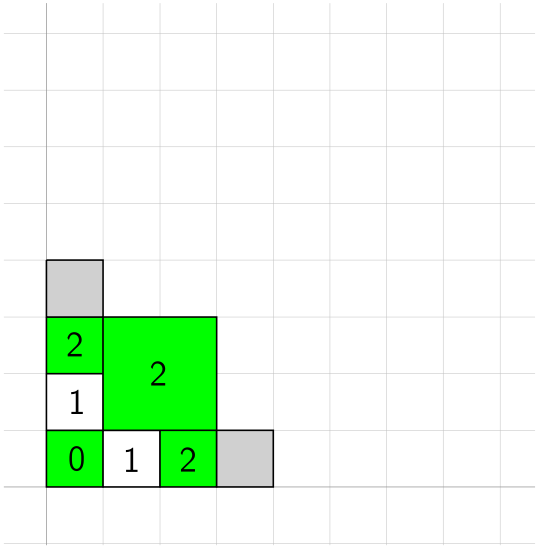


Figure. What we must compute

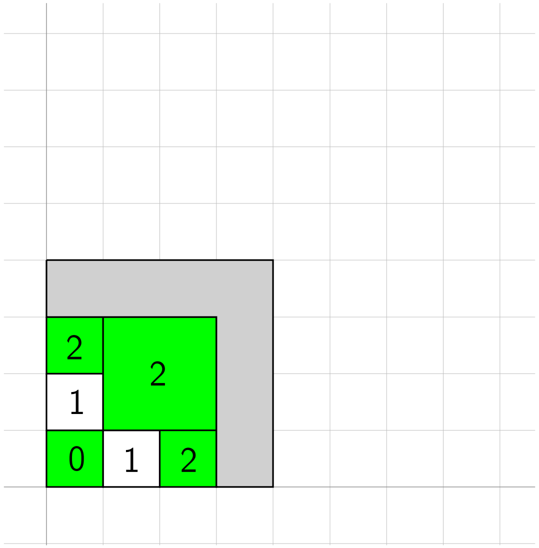


Figure. Minimum knowledge on the input

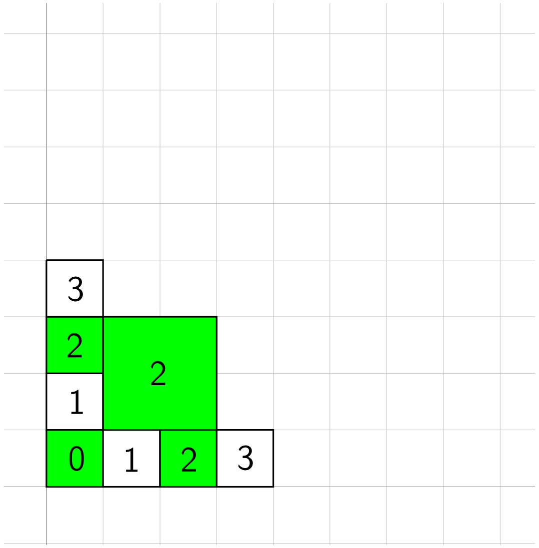


Figure. What we compute

step 0: $c \;=\; a_0\, b_0$

step 1: $c \mathrel{+}= z\ (a_0\, b_1 + a_1\, b_0)$

step 2: $c \mathrel{+}= z^2\, (a_0\, b_2 + a_2\, b_0 + (a_1 + a_2\, z)\, (b_1 + b_2\, z))$

step 3: $c \mathrel{+}= z^3\, (a_0\, b_3 + a_3\, b_0)$

Relaxed multiplication $c = a \times b$ - step 4

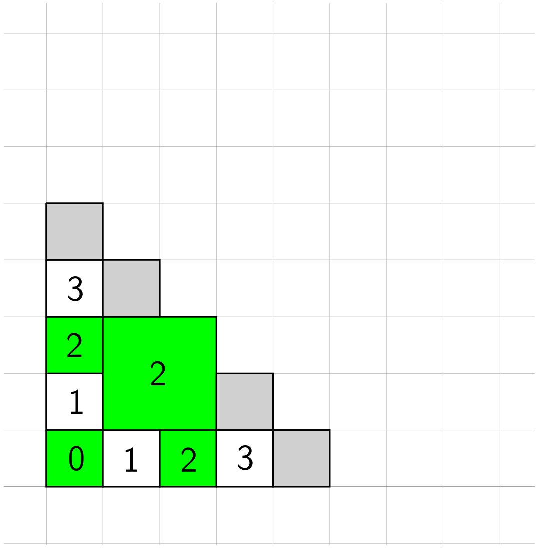


Figure. What we must compute

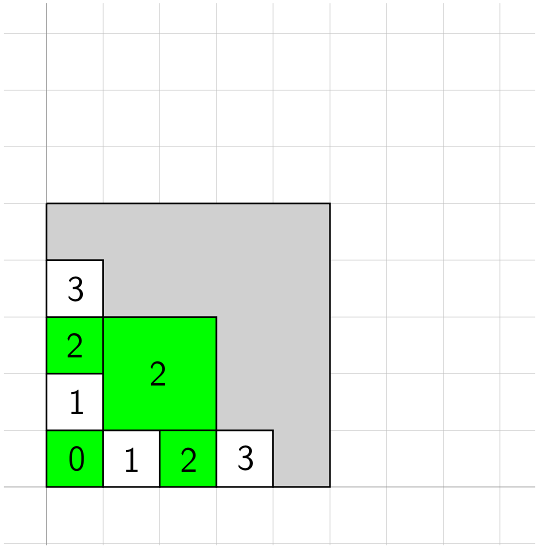


Figure. Minimum knowledge on the input

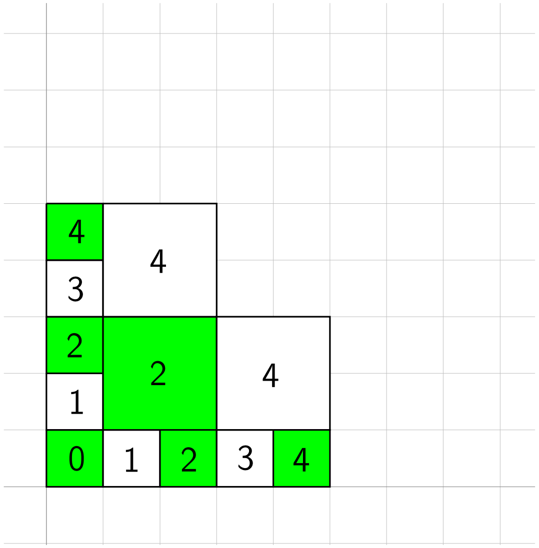


Figure. What we compute

step 0: $c \;=\; a_0\, b_0$

step 1: $c \mathrel{+}= z\, (a_0\, b_1 + a_1\, b_0)$

step 2: $c \mathrel{+}= z^2\, (a_0\, b_2 + a_2\, b_0 + (a_1 + a_2\, z)\, (b_1 + b_2\, z))$

step 3: $c \mathrel{+}= z^3\, (a_0\, b_3 + a_3\, b_0)$

step 4: $c \mathrel{+}= z^4\, (a_0\, b_4 + a_4\, b_0 + (a_1 + a_2\, z)\, (b_3 + b_4\, z)$
$+ (a_3 + a_4\, z)\, (b_1 + b_2\, z))$

Relaxed multiplication $c = a \times b$ - step 5

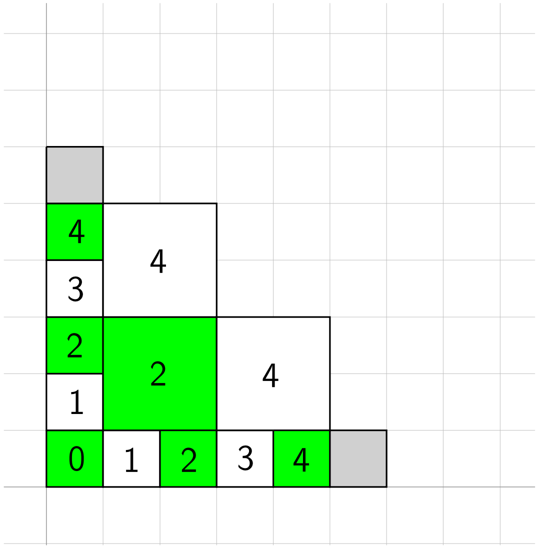


Figure. What we must compute

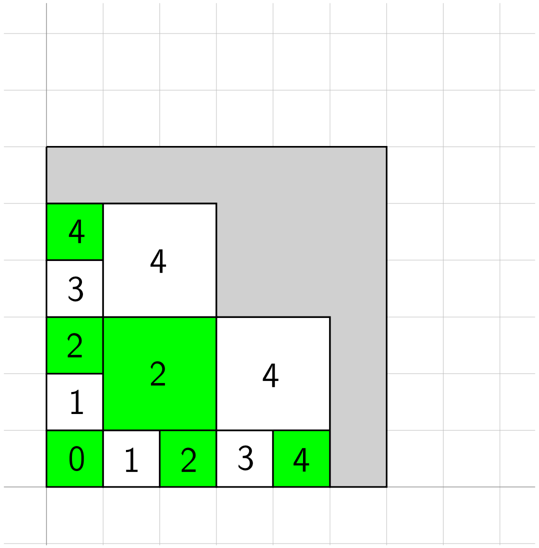


Figure. Minimum knowledge on the input

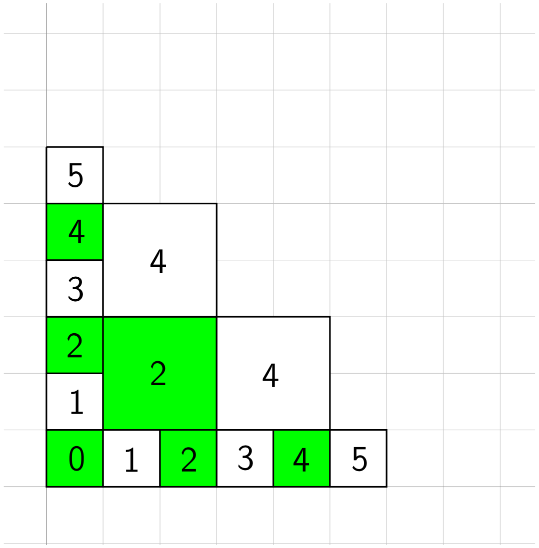


Figure. What we compute

step 0: $c \;=\; a_0\, b_0$

step 1: $c \mathrel{+}= z\;(a_0\, b_1 + a_1\, b_0)$

step 2: $c \mathrel{+}= z^2\,(a_0\, b_2 + a_2\, b_0 + (a_1 + a_2\, z)\,(b_1 + b_2\, z))$

step 3: $c \mathrel{+}= z^3\,(a_0\, b_3 + a_3\, b_0)$

step 4: $c \mathrel{+}= z^4\,(a_0\, b_4 + a_4\, b_0 + (a_1 + a_2\, z)\,(b_3 + b_4\, z)$
$+ (a_3 + a_4\, z)\,(b_1 + b_2\, z))$

step 5: $c \mathrel{+}= z^5\,(a_0\, b_5 + a_5\, b_0)$

Relaxed multiplication $c = a \times b$ - step 6

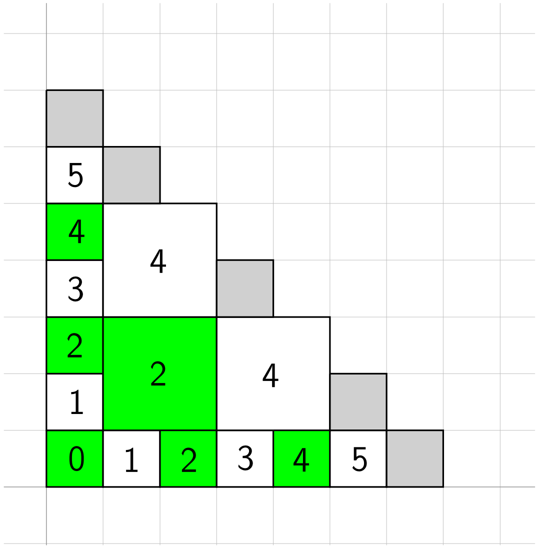


Figure. What we must compute

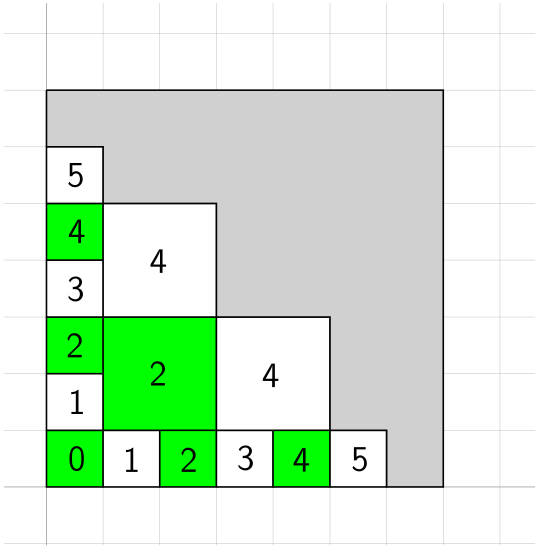


Figure. Minimum knowledge on the input

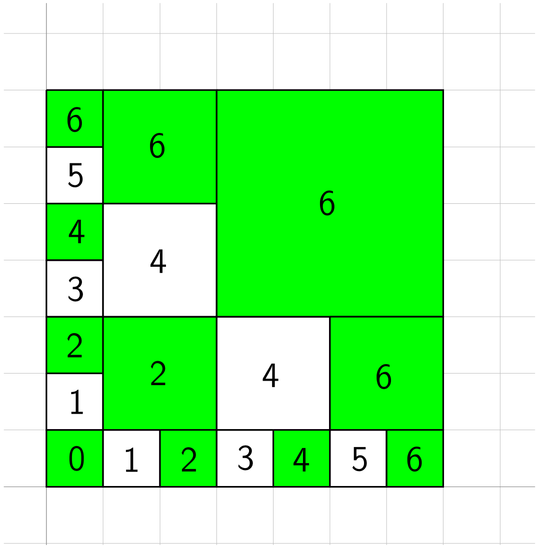


Figure. What we compute

step 0: $c = a_0 b_0$

step 1: $c += z\,(a_0 b_1 + a_1 b_0)$

step 2: $c += z^2 (a_0 b_2 + a_2 b_0 + (a_1 + a_2 z)(b_1 + b_2 z))$

step 3: $c += z^3 (a_0 b_3 + a_3 b_0)$

step 4: $c += z^4 (a_0 b_4 + a_4 b_0 + (a_1 + a_2 z)(b_3 + b_4 z)$
$+ (a_3 + a_4 z)(b_1 + b_2 z))$

step 5: $c += z^5 (a_0 b_5 + a_5 b_0)$

step 6: $c += z^6 (a_0 b_6 + a_6 b_0 + (a_1 + a_2 z)(b_5 + b_6 z)$
$+ (a_5 + a_6 z)(b_1 + b_2 z)$
$+ (a_3 + \cdots + a_6 z^3)(b_3 + \cdots + b_6 z^3))$

Relaxed multiplications

Theorem. [Fischer, Stockmeyer '74], [Schröder '97], [van der Hoeven '97], [Berthomieu, van der Hoeven, Lecerf '11], [L., Schost '12]

$\mathsf{M}(N)$: *cost of* $a \times b$ *in* $k[[x]]$ *at precision* N *by an off-line algorithm.*
$\mathsf{R}(N)$: *cost of* $a \times b$ *in* $k[[x]]$ *at precision* N *by an on-line algorithm. Then*

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

Relaxed multiplications

Theorem. [Fischer, Stockmeyer '74], [Schröder '97], [van der Hoeven '97], [Berthomieu, van der Hoeven, Lecerf '11], [L., Schost '12]

$\mathsf{M}(N)$: *cost of* $a \times b$ *in* $k[[x]]$ *at precision* N *by an off-line algorithm.*
$\mathsf{R}(N)$: *cost of* $a \times b$ *in* $k[[x]]$ *at precision* N *by an on-line algorithm. Then*

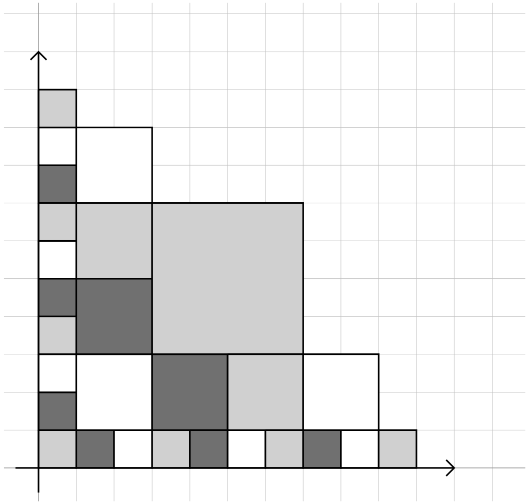
$$\mathsf{R}(N) = \mathcal{O}(\mathsf{M}(N) \log N) = \tilde{\mathcal{O}}(N).$$

Theorem. [van der Hoeven '07, '12]

$$\mathsf{R}(N) = \mathsf{M}(N) \log (N)^{o(1)}.$$

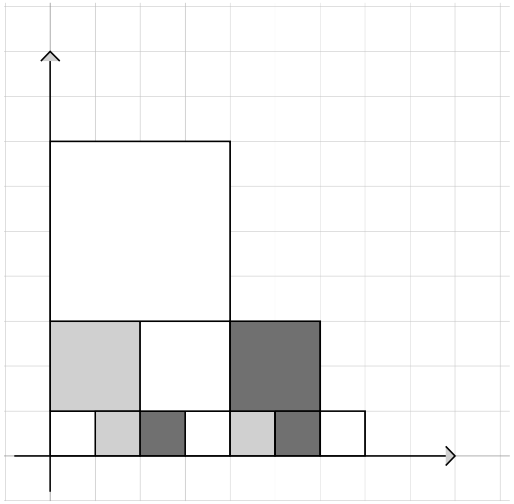
Seems not to be used yet in practice.

Relaxed multiplications



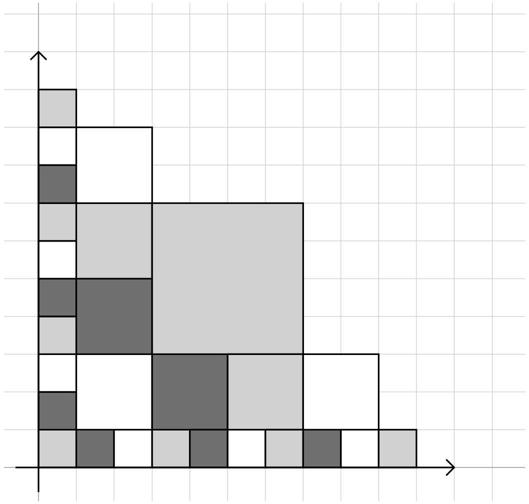
Relaxed multiplication [Fischer, Stockmeyer '74]

Relaxed multiplications

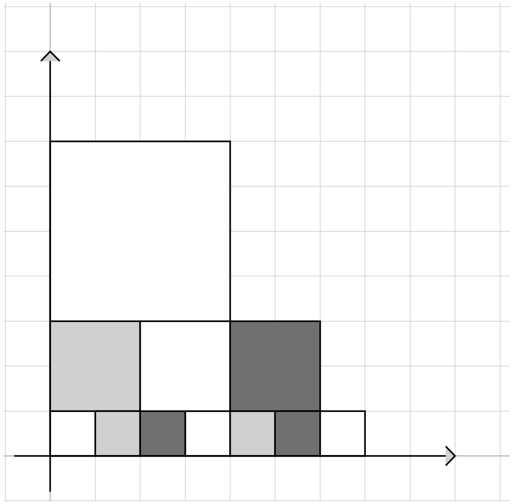


Semi-relaxed multiplication [Fischer, Stockmeyer '74]

$\longrightarrow$

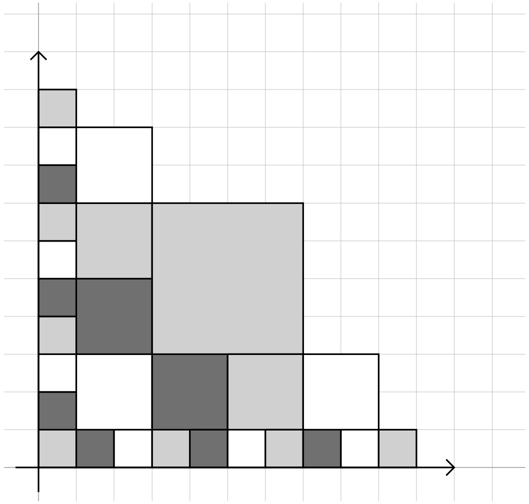


Relaxed multiplication [Fischer, Stockmeyer '74]

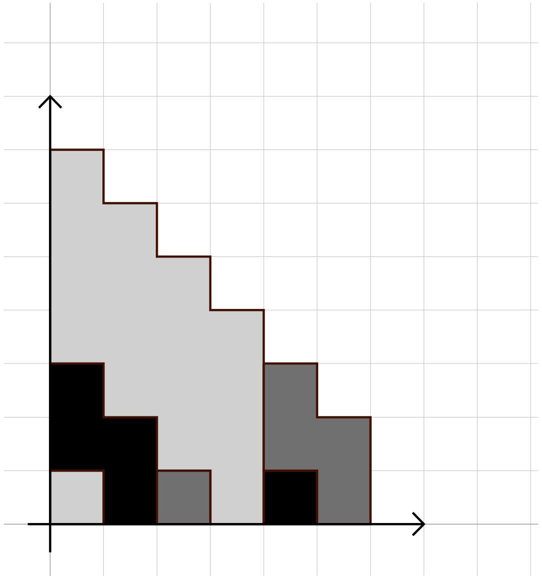


Semi-relaxed multiplication [FISCHER, STOCKMEYER '74]

⟶



Relaxed multiplication [FISCHER, STOCKMEYER '74]

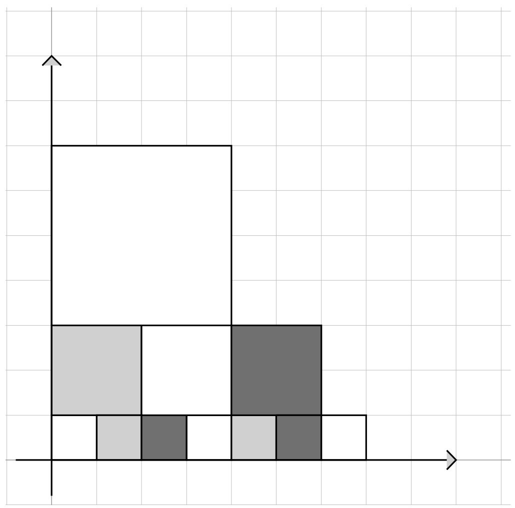


Semi-relaxed multiplication [VAN DER HOEVEN '03]

⟶

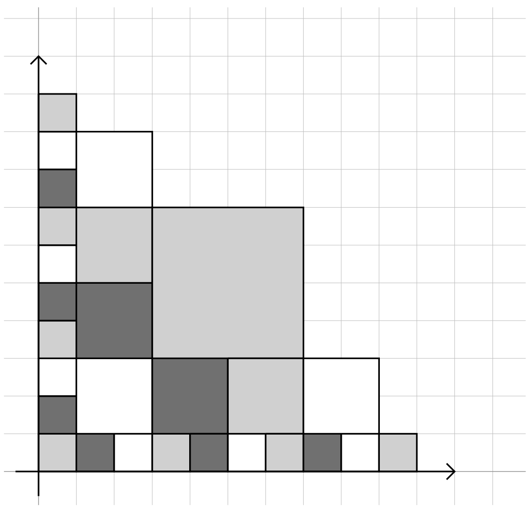
?

Relaxed multiplications

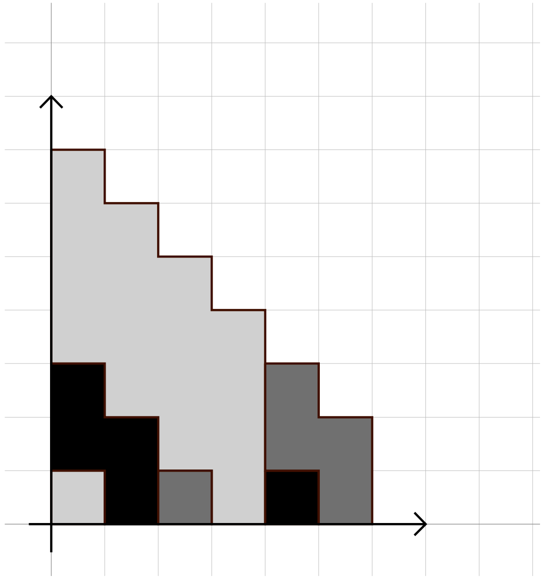


Semi-relaxed multiplication [Fischer, Stockmeyer '74]

→

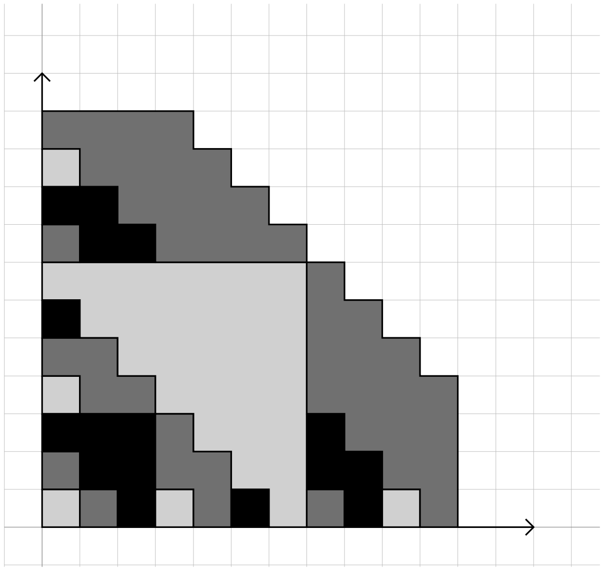


Relaxed multiplication [Fischer, Stockmeyer '74]



Semi-relaxed multiplication [van der Hoeven '03]

→



Relaxed multiplication [L., Schost '12]

Timings of relaxed multiplications

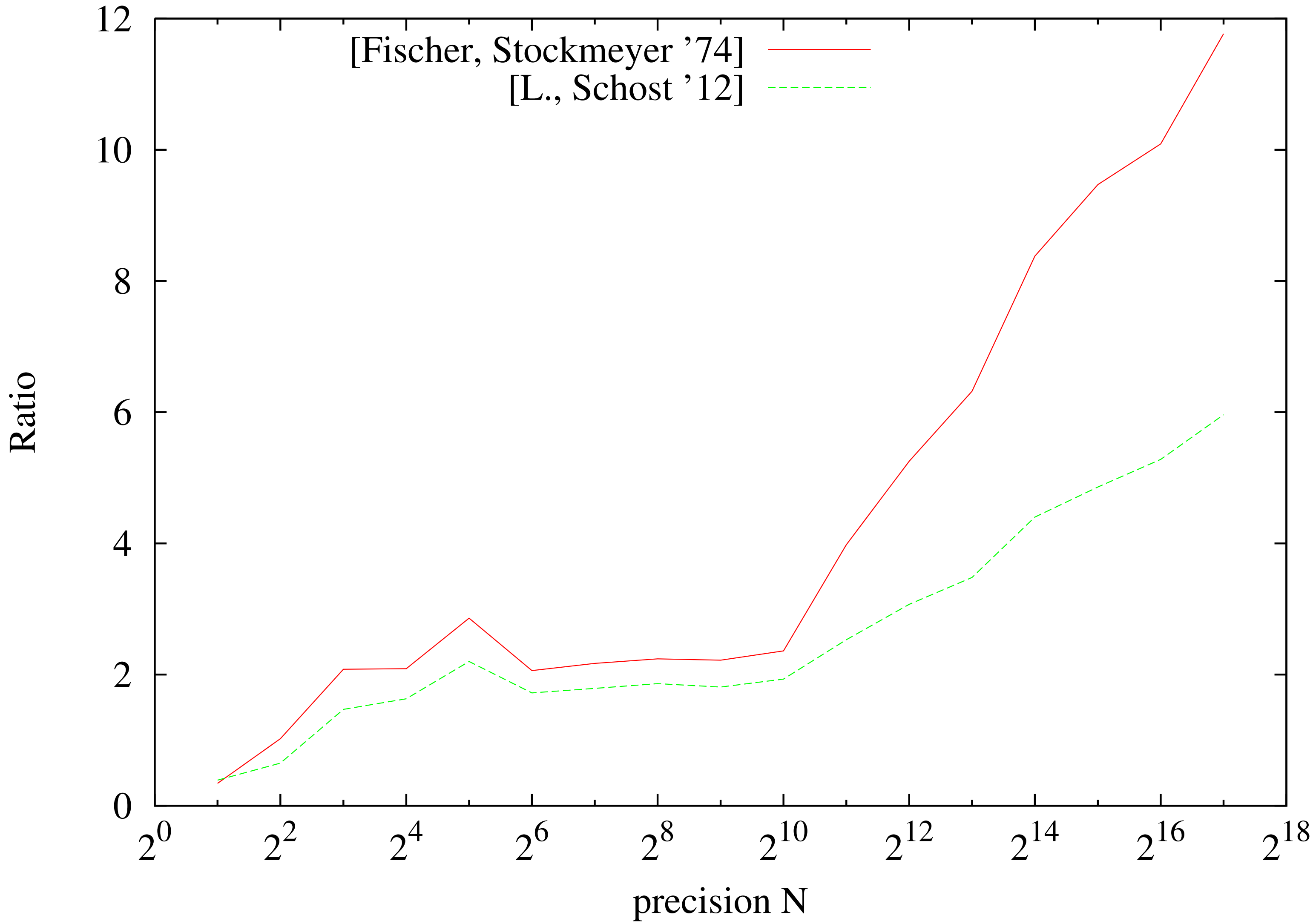


Figure. Ratio $\mathsf{R}(N)/\mathsf{M}(N)$ for different relaxed multiplication algorithms over $\mathbb{F}_{268435459}[[x]]$.

Plan

1. Why is p-adic lifting interesting?
2. Relaxed p-adic lifting
    a. On-line algorithms
    b. Recursive p-adic
    c. Application to:
        i. Linear systems
        ii. Linear q-differential systems
        iii. Polynomial root lifting
        iv. Univariate representation lifting
3. Conclusion

Relaxed recursive power series

Definition. *A power series $y \in k[[x]]$ is recursive if there exists Φ such that*

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

Relaxed recursive power series

Definition. *A power series* $y \in k[[x]]$ *is recursive if there exists* Φ *such that*

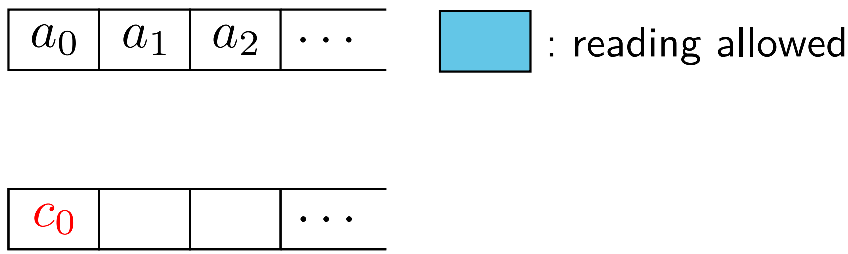
- $y = \Phi(y)$
- $\Phi(y)_n$ *only depends on* $y_0, \ldots, y_{n-1}$

Definition. (*Shifted algorithm*)

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

$$\downarrow f$$

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



Relaxed recursive power series

Definition. *A power series $y \in k[[x]]$ is recursive if there exists Φ such that*

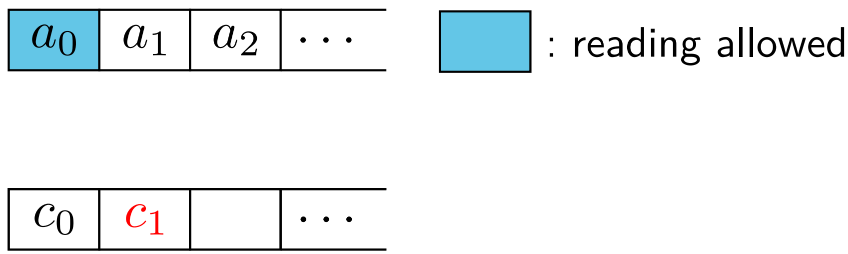
- $y = \Phi(y)$
- $\Phi(y)_n$ *only depends on* $y_0, \ldots, y_{n-1}$

Definition. **(*Shifted algorithm*)**

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

$$\downarrow f$$

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



Relaxed recursive power series

Definition. *A power series* $y \in k[[x]]$ *is* *recursive* *if there exists* Φ *such that*

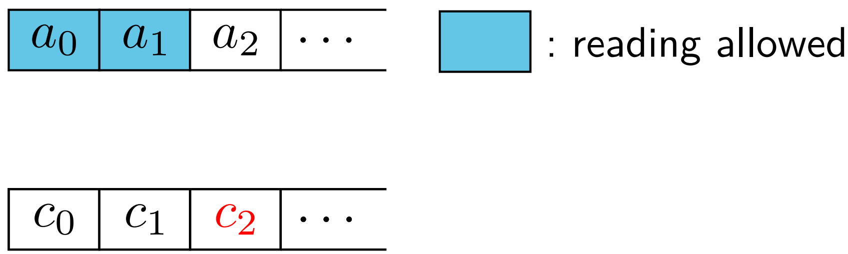
- $y = \Phi(y)$
- $\Phi(y)_n$ *only depends on* $y_0, \ldots, y_{n-1}$

Definition. (*Shifted algorithm*)

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

$$\downarrow f$$

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



Relaxed recursive power series

Definition. *A power series $y \in k[[x]]$ is recursive if there exists Φ such that*

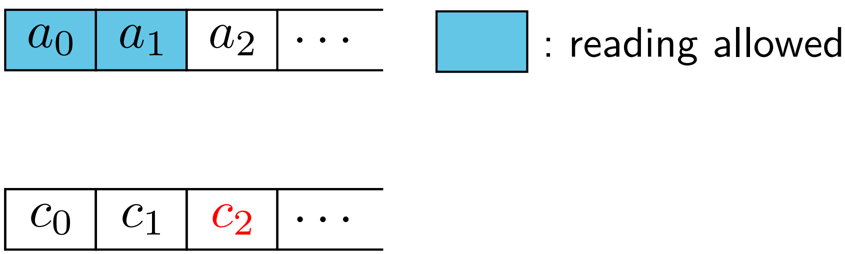
- $y = \Phi(y)$
- $\Phi(y)_n$ *only depends on* $y_0, \ldots, y_{n-1}$

Definition. (***Shifted algorithm***)

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

$$\downarrow f$$

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



Theorem. [Watt '88], [van der Hoeven '02]

Let $y \in k[[x]]$ such that $y = \Phi(y)$. Given y_0 and Φ, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$.

Definition. *A power series $y \in k[[x]]$ is recursive if there exists Φ such that*

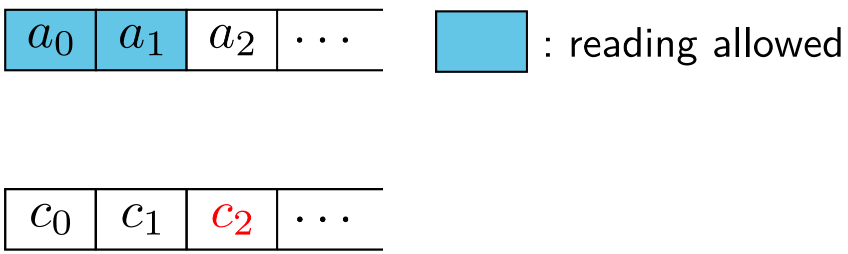
- $y = \Phi(y)$
- $\Phi(y)_n$ *only depends on* $y_0, \ldots, y_{n-1}$

Definition. **(*Shifted algorithm*)**

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

$\downarrow f$

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



Theorem. [Watt '88], [van der Hoeven '02], [Berthomieu, L. ISSAC '12]

Let $y \in k[[x]]$ such that $y = \Phi(y)$. Given y_0 and Φ, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$ by a shifted algorithm.

Relaxed recursive power series

Theorem. (Fundamental theorem)

[Watt '88], [van der Hoeven '02], [Berthomieu, L. ISSAC '12]

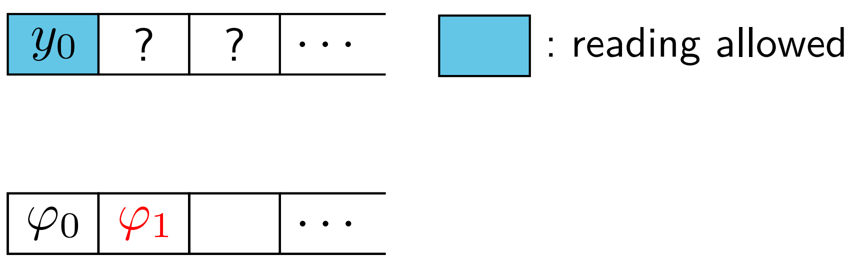
Let $y \in k[[x]]$ such that $y = \Phi(y)$. Given y_0 and Φ, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$ by a shifted algorithm.

Proof. $y = \Phi(y)$

$$y = \textstyle\sum_{i \geqslant 0} y_i x^i$$

$$\downarrow \Phi$$

$$\Phi(y) = \textstyle\sum_{i \geqslant 0} \varphi_i x^i$$



□

Theorem. (Fundamental theorem)

[Watt '88], [van der Hoeven '02], [Berthomieu, L. ISSAC '12]

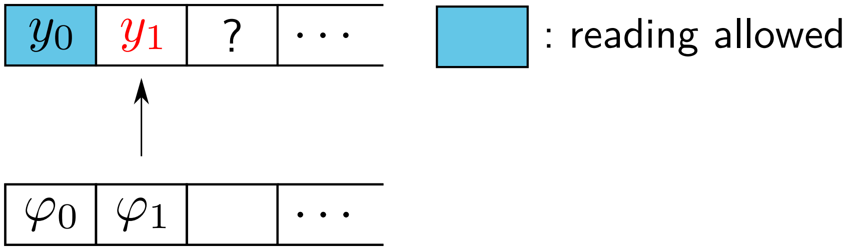
Let $y \in k[[x]]$ such that $y = \Phi(y)$. Given y_0 and Φ, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$ by a shifted algorithm.

Proof. $y = \Phi(y) \Rightarrow \varphi_1 = y_1$

$$y = \sum_{i \geqslant 0} y_i x^i$$

$$\downarrow \Phi$$

$$\Phi(y) = \sum_{i \geqslant 0} \varphi_i x^i$$



□

Theorem. (Fundamental theorem)

[Watt '88], [van der Hoeven '02], [Berthomieu, L. ISSAC '12]

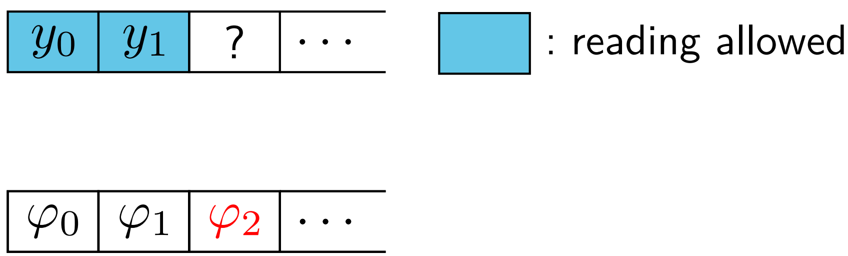
Let $y \in k[[x]]$ such that $y = \Phi(y)$. Given y_0 and Φ, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$ by a shifted algorithm.

Proof. $y = \Phi(y)$

$$y = \sum_{i \geqslant 0} y_i x^i$$

$$\downarrow \Phi$$

$$\Phi(y) = \sum_{i \geqslant 0} \varphi_i x^i$$



□

Theorem. (Fundamental theorem)

[Watt '88], [van der Hoeven '02], [Berthomieu, L. ISSAC '12]

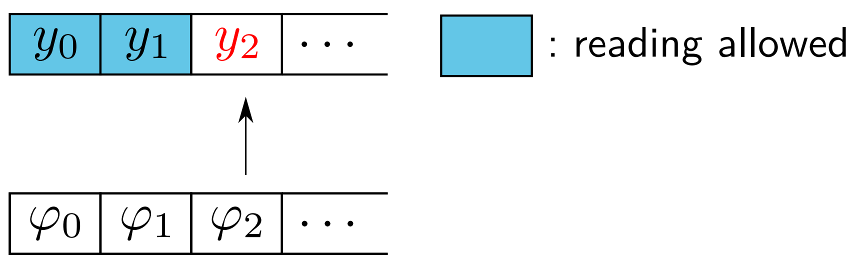
Let $y \in k[[x]]$ such that $y = \Phi(y)$. Given y_0 and Φ, we can compute y at precision N in the time necessary to evaluate $\Phi(y)$ by a shifted algorithm.

Proof. $y = \Phi(y) \Rightarrow \varphi_2 = y_2$

$$y = \sum_{i \geqslant 0} y_i x^i$$

$$\downarrow \Phi$$

$$\Phi(y) = \sum_{i \geqslant 0} \varphi_i x^i$$



□

Plan

1. Why is p-adic lifting interesting?
2. Relaxed p-adic lifting
    a. On-line algorithms
    b. Recursive p-adic
    c. Application to:
        i. Linear systems
        ii. Linear q-differential systems
        iii. Polynomial root lifting
        iv. Univariate representation lifting
3. Conclusion

Two paradigms

Lifting solutions methods:

Zealous algorithms	Relaxed algorithms
Newton operator	Fundamental theorem
implicit equations $\boldsymbol{P}(\boldsymbol{Y})=0$	recursive equations $\boldsymbol{Y}=\boldsymbol{\Phi}(\boldsymbol{Y})$

To do:

- Transform an implicit equations into recursive equations (+ shifted algorithm)

$$\boldsymbol{P}(\boldsymbol{Y})=\mathbf{0} \longrightarrow \boldsymbol{Y}=\boldsymbol{\Psi}(\boldsymbol{Y})$$

Linear systems

		Zealous algorithms	Relaxed algorithms
Linear systems	dense	$\tilde{\mathcal{O}}(r^{\omega} d + r^{\omega-1} N)$	
	structured	$\tilde{\mathcal{O}}(\alpha^2 r d + \alpha r N)$	

Example. Find $a(x), b(x), c(x) \in k[[x]]$ such that

$$\begin{cases} (1+x)\,a(x) + (3-x^2)\,b(x) + 2\,c(x) = 1 \\ (x-x^3)\,a(x) + (1+x)\,b(x) + (3-x^2)\,c(x) = x \\ (4-2\,x)\,a(x) + (x-x^3)\,b(x) + (1+x)\,c(x) = x^2 \end{cases}$$

r number of unknowns
d degree of polynomials in input
N precision of the output
α parameter associated to structured matrices $\alpha \ll r$
ω exponent of matrix multiplication $2 \leqslant \omega \leqslant 3$

Linear systems: [Moenck, Carter '79], [Storjohann '03]

Ideas

Example of system

$$A \cdot Y = B \Leftrightarrow \begin{pmatrix} (1+x) & (3-x^2) & 2 \\ (x-x^3) & (1+x) & (3-x^2) \\ (4-2\,x) & (x-x^3) & (1+x) \end{pmatrix} \cdot \begin{pmatrix} a(x) \\ b(x) \\ c(x) \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

Transformation of equation

Implicit equation	$0 = f(Y) := A \cdot Y - B$
Recursive equation	$Y = \Phi(Y) := A_0^{-1} \cdot (B - (A - A_0) \cdot Y)$
Shifted algorithm	$\Psi(Y) := A_0^{-1} \cdot \left(B - x \times \left[\left(\frac{A - A_0}{x}\right) \cdot_{\text{online}} Y\right]\right)$

		Zealous algorithms	Relaxed algorithms
Linear systems	dense	$\tilde{\mathcal{O}}(r^{\omega}\, d + r^{\omega-1}\, N)$	$\tilde{\mathcal{O}}(r^{\omega} + r^2\, N)$
	structured	$\tilde{\mathcal{O}}(\alpha^2\, r\, d + \alpha\, r\, N)$	$\tilde{\mathcal{O}}(\alpha^2\, r + \alpha\, r\, N)$

[Berthomieu, L., *ISSAC* '12], [L., Schost '12]

Example. Find $a(x), b(x), c(x) \in k[[x]]$ such that

$$\begin{cases} (1+x)\, a(x) + (3-x^2)\, b(x) + 2\, c(x) = 1 \\ (x-x^3)\, a(x) + (1+x)\, b(x) + (3-x^2)\, c(x) = x \\ (4-2\,x)\, a(x) + (x-x^3)\, b(x) + (1+x)\, c(x) = x^2 \end{cases}$$

r number of unknowns
d degree of polynomials in input
N precision of the output
α parameter associated to structured matrices $\alpha \ll r$
ω exponent of matrix multiplication $2 \leqslant \omega \leqslant 3$

Linear systems: [Moenck, Carter '79], [Storjohann '03]

Plan

1. Why is p-adic lifting interesting?
2. Relaxed p-adic lifting
    a. On-line algorithms
    b. Recursive p-adic
    c. Application to:
        i. Linear systems
        ii. Linear q-differential systems
        iii. Polynomial root lifting
        iv. Univariate representation lifting
3. Conclusion

Differential systems

	Zealous algorithms	Relaxed algorithms
Regular differential systems	$\mathcal{O}(r^{\omega})\,\mathsf{M}(N) + \mathcal{O}_{N\to\infty}(N)$	$\mathcal{O}(r^2)\,\mathsf{R}(N) + \mathcal{O}_{N\to\infty}(N)$
(q)-differential singular systems		

Example. Find $a(x) \in k[[x]]$ such that

$$(1+x)\,a(x) + (3-x^2)\,(a(2\,x) - a(x)) \;=\; x$$

Applications:

- List-decoding of folded Reed-Solomon codes: solve $Q\,(x, f(x), f\,(q\,x)) = 0$
- Composition of power series $f(g)$ with $g(0)=0$ when f solution of a singular differential equation.

Differential systems: [Brent, Kung '78], [van der Hoeven '02 '11], [Bostan, Chyzak *et al.* '07]

Differential systems

	Zealous algorithms	Relaxed algorithms
Regular differential systems	$\mathcal{O}(r^\omega)\,\mathsf{M}(N) + \mathcal{O}_{N\to\infty}(N)$	$\mathcal{O}(r^2)\,\mathsf{R}(N) + \mathcal{O}_{N\to\infty}(N)$
(q)-differential singular systems	$\mathcal{O}(r^\omega)\,\mathsf{M}(N) + \mathcal{O}_{N\to\infty}(N)$	$\mathcal{O}(r^2)\,\mathsf{R}(N) + \mathcal{O}_{N\to\infty}(N)$

[Bostan, Chowdhury, L., Salvy, Schost, *ISSAC* '12]

Example. Find $a(x) \in k[[x]]$ such that

$$(1+x)\,a(x) + (3-x^2)\,(a(2\,x) - a(x)) \;=\; x$$

r number of unknowns
N precision of the output

Differential systems: [Brent, Kung '78], [van der Hoeven '02 '11], [Bostan, Chyzak *et al.* '07]

Polynomial systems

	Zealous algorithms	Relaxed algorithms
Polynomial systems	$\mathcal{O}(r\,L + r^{\omega})\,\mathsf{M}(N) + \mathcal{O}_{N\to\infty}(N)$	
Univariate representations	$\mathcal{O}(r\,L + r^{\omega})\,\mathsf{M}(N)\,\mathsf{M}(d) + \mathcal{O}_{N\to\infty}(N)$	

Example. Find $a(x), b(x), c(x) \in k[[x]]$ such that

$$\left\{ \begin{array}{rcl} (1+x) + (4-2\,x)\,a(x)^2\,c(x)^4 + 2\,b(x) & = & 0 \\ (x-x^3) + (3-x^2)\,a(x)\,b(x)^2 + (1+x+x^2)\,c(x) & = & 0 \\ 1 + b(x)^4\,c(x)^5 + x\,a(x)^3 & = & 0 \end{array} \right.$$

r number of unknowns, N precision of the output, L complexity of evaluation

Polynomial systems: [Newton 1736], [Hensel 1918], [Lipson '76]
[Berthomieu, van der Hoeven, Lecerf '11], [van der Hoeven '11]
Univariate representations: [Giusti, Lecerf, Salvy '01], [Heintz, Matera, Waissbein '01]

Polynomial systems

	Zealous algorithms	Relaxed algorithms
Polynomial systems	$\mathcal{O}(r\,L + r^{\omega})\,\mathsf{M}(N) + \mathcal{O}_{N\to\infty}(N)$	$\mathcal{O}(L)\,\mathsf{R}(N) + \mathcal{O}_{N\to\infty}(N)$
Univariate representations	$\mathcal{O}(r\,L + r^{\omega})\,\mathsf{M}(N)\,\mathsf{M}(d) + \mathcal{O}_{N\to\infty}(N)$	$\mathcal{O}(L)\,\mathsf{R}(N)\,\mathsf{M}(d) + \mathcal{O}_{N\to\infty}(N)$

[Berthomieu, L., *ISSAC* '12], [L. '12]

Example. Find $a(x), b(x), c(x) \in k[[x]]$ such that

$$\left\{ \begin{array}{rcl} (1+x) + (4-2\,x)\,a(x)^2\,c(x)^4 + 2\,b(x) & = & 0 \\ (x-x^3) + (3-x^2)\,a(x)\,b(x)^2 + (1+x+x^2)\,c(x) & = & 0 \\ 1 + b(x)^4\,c(x)^5 + x\,a(x)^3 & = & 0 \end{array} \right.$$

r number of unknowns, N precision of the output, L complexity of evaluation

Polynomial systems: [Newton 1736], [Hensel 1918], [Lipson '76]
[Berthomieu, van der Hoeven, Lecerf '11], [van der Hoeven '11]
Univariate representations: [Giusti, Lecerf, Salvy '01], [Heintz, Matera, Waissbein '01]

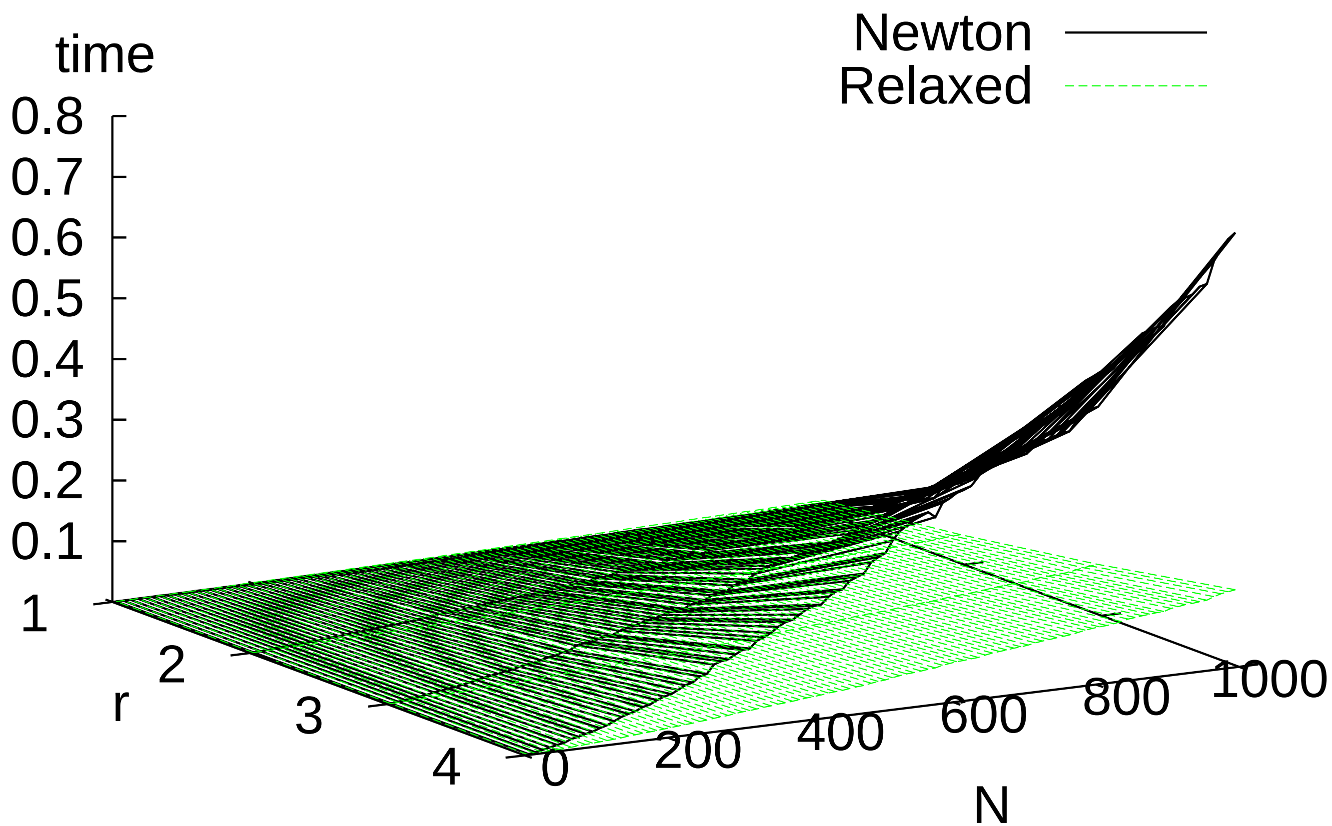


Figure. Timings with systems of singular (q)-differential equations

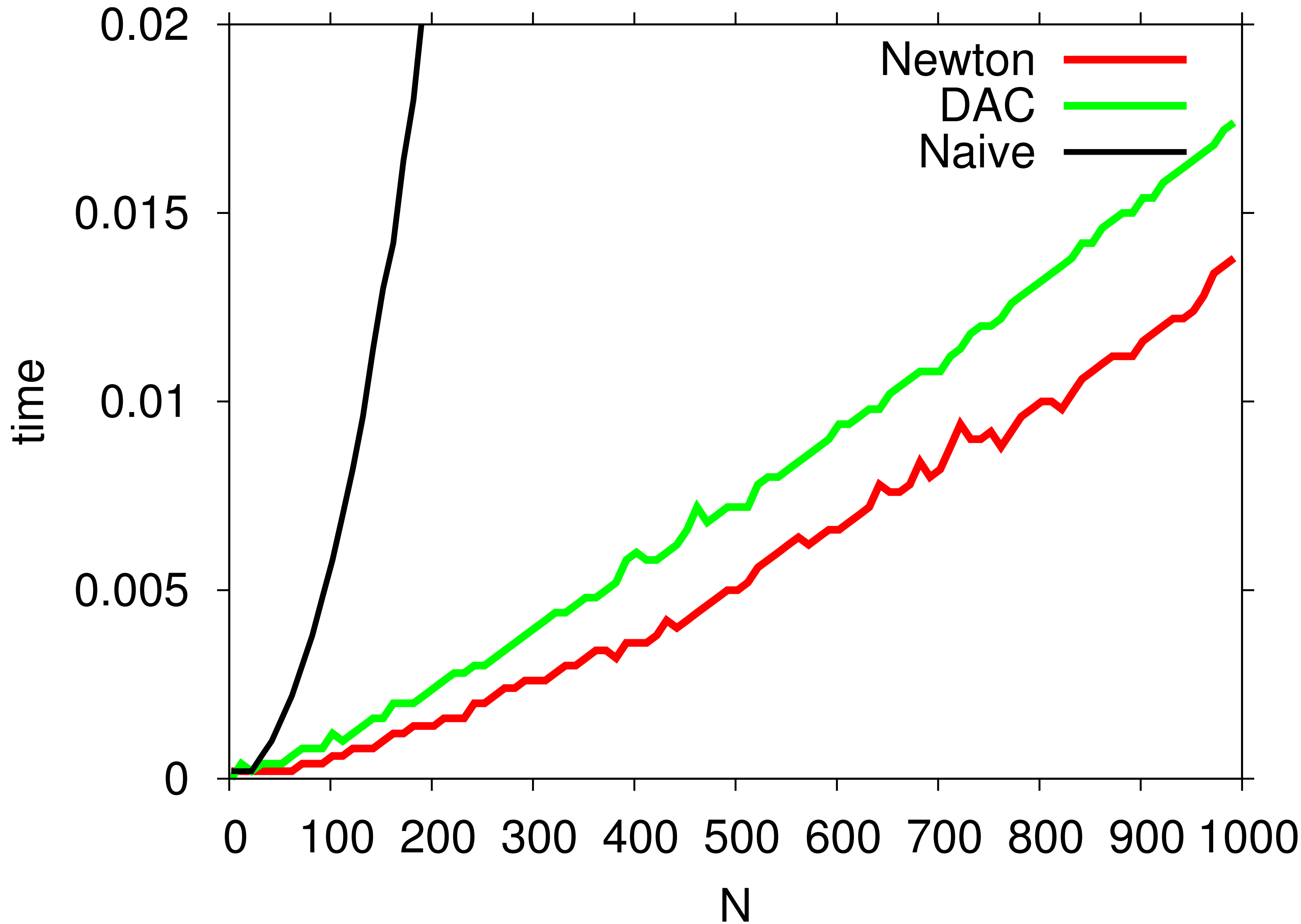


Figure. Timings with a singular (q)-differential equation

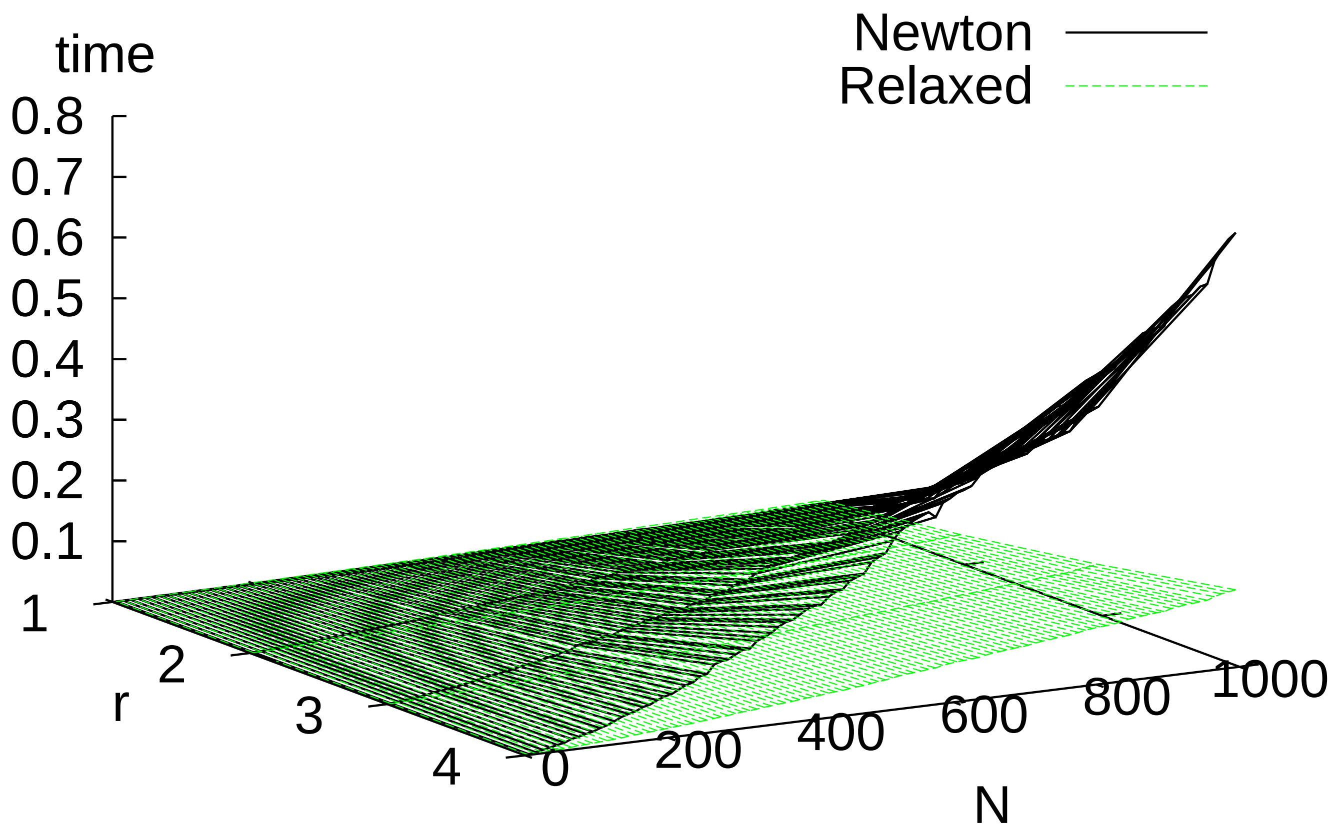


Figure. Timings with systems of singular (q)-differential equations

Timings

N	Katsura-3		Katsura-4		Katsura-5		Katsura-6	
	zealous	relaxed	zealous	relaxed	zealous	relaxed	zealous	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 general paradigms:

Newton operator	Relaxed algorithms
Solve implicit equations $P(y) = 0$	Solve recursive equations $y = \Phi(y)$
Faster for higher precision	Less on-line multiplications

Implementations:

Contributions to ALGEBRAMIX and GEOMSOLVEX packages of MATHEMAGIX;

Perspectives:

- Lift a polynomial root under more general hypotheses (multiplicities);
- Link between relaxed and divide-and-conquer algorithms
- New horizons : Relaxed Gröbner bases?

Thank you for your attention ;-)