

# Fast reversion of formal power series

Fredrik Johansson  
LFANT, INRIA Bordeaux

RAIM, 2016-06-29, Banyuls-sur-mer

## Reversion of power series

$$F = \exp(x) - 1 = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

$$G = \log(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$$

$$F(G(x)) = G(F(x)) = x, \quad G = F^{-1} \text{ and } F = G^{-1}$$

Applications:

- ▶ Combinatorics, recurrences (example: counting binary trees)
- ▶ Numerical approximation of  $F^{-1}$
- ▶ Asymptotic analysis

## Operations in $R[[x]]/\langle x^n \rangle$

$$F = F(x) = \sum_{k=0}^{n-1} f_k x^k, \quad [x^k]F(x) = f_k \in R$$

Addition  $F + G = \sum_{k=0}^{n-1} (f_k + g_k) x^k$

Multiplication  $FG = \sum_{k=0}^{n-1} \left( \sum_{i=0}^k f_i g_{k-i} \right) x^k$

Reciprocal If  $f_0$  is a unit, find  $G$  such that  $FG = 1$

Composition If  $g_0 = 0$ ,  $F(G) = \sum_{k=0}^{n-1} f_k G^k$

Reversion If  $f_0 = 0$  and  $f_1$  is a unit, find  $G$  s.t.  $F(G) = x$

## Computational complexity

Counting operations in the base ring  $R$ .

$M(n)$ : multiplying two length- $n$  polynomials

Classical  $M(n) = O(n^2)$

Karatsuba  $M(n) = O(n^{\log_2(3)}) = O(n^{1.59})$

Fast Fourier Transform  $M(n) = O(n \log^{1+o(1)} n) = O(n^{1+o(1)})$

$MM(n) = O(n^\omega)$ : multiplying two  $n \times n$  matrices

Classical  $MM(n) = O(n^3)$

Strassen  $MM(n) = O(n^{\log_2(7)}) = O(n^{2.81})$

Coppersmith-Winograd-Le Gall  $MM(n) = O(n^{2.3728639})$

## Newton iteration for power series

If  $\varphi(G_k) = 0 \pmod{x^n}$ , then (under natural assumptions)

$$\varphi(G_{k+1}) = 0 \pmod{x^{2n}}, \quad G_{k+1} = G_k - \frac{\varphi(G_k)}{\varphi'(G_k)}$$

$O(M(n))$  computation of  $1/F$ : take  $\varphi(G) = \frac{1}{G} - F$ , giving:

$$G_{k+1} = 2G_k - FG_k^2.$$

## Reversion using Newton iteration

Given  $F = f_0 + f_1x + \dots$  with  $f_0 = 0$  and  $f_1$  invertible.

Take  $\varphi(G) = F(G) - x$ , giving:

$$G_{k+1} = G_k - \frac{F(G) - x}{F'(G)}.$$

Brent and Kung, 1978: up to constant factors:

composition  $\Leftrightarrow$  reversion

## Fast composition: elementary functions

Assume that  $1, 2, \dots, n-1$  are invertible in  $R$ .

$$F' = \sum_{k=0}^{n-2} (k+1) f_{k+1} x^k, \quad \int F = \sum_{k=1}^{n-1} \left(\frac{1}{k}\right) f_{k-1} x^k$$

$$\log(1+F) = \int \frac{F'}{1+F}, \quad \operatorname{atan}(F) = \int \frac{F}{1+F^2}$$

Newton iteration:  $\log \rightarrow \exp$ ,  $\operatorname{atan} \rightarrow \tan$ .

$\sin$ ,  $\cos$ ,  $\sinh$ ,  $\cosh$ ,  $\operatorname{asin}$  ... using algebraic transformations.

Cost:  $O(M(n)) = O(n^{1+o(1)})$ .

Generalization: differential equations (hypergeometric, ...)

# Composition of general power series

Horner's rule

$$F(G) = (\dots (f_{n-1} \cdot G + f_{n-2}) \cdot G + \dots + f_1) \cdot G + f_0$$

Complexity:  $O(nM(n))$

Brent-Kung 2.1: baby-step, giant-step version of Horner's rule

Complexity:  $O(n^{1/2}M(n) + n^{1/2}MM(n^{1/2}))$

Brent-Kung 2.2: divide-and-conquer Taylor expansion

Complexity:  $O((n \log n)^{1/2}M(n))$



## Brent-Kung algorithm 2.1

Example with  $n = 9$ ,  $m = \lceil \sqrt{n} \rceil = 3$

Computing  $F(G)$  where  $F = f_0 + f_1x + \dots + f_8x^8$

$$(f_0 + f_1G + f_2G^2) + (f_3 + f_4G + f_5G^2)G^3 + (f_6 + f_7G + f_8G^2)G^6$$

$(m \times m) \times (m \times m^2)$  matrix multiplication:

$$\begin{pmatrix} f_0 & f_1 & f_2 \\ f_3 & f_4 & f_5 \\ f_6 & f_7 & f_8 \end{pmatrix} \times \begin{pmatrix} \text{---} & \text{---} & \text{---} & 1 & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} & G & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} & G^2 & \text{---} & \text{---} \end{pmatrix}$$

Final combination using Horner's rule.

## Complexity of composition algorithms

Horner's rule:  $O(nM(n))$

BK 2.1:  $O(n^{1/2}M(n) + n^{1/2}MM(n^{1/2}))$

BK 2.2:  $O((n \log n)^{1/2}M(n)) = O(n^{3/2+o(1)})$

$M(n)$	$MM(n)$	Horner	BK 2.1	BK 2.2
$n^2$	$n^3$	$n^3$	$n^{2.5}$	$n^{2.5} \log^{0.5} n$
$n^{1.59}$	$n^3$	$n^{2.59}$	$n^{2.09}$	$n^{2.09} \log^{0.5} n$
$n \log n$	$n^3$	$n^2 \log n$	$n^2$	$n^{1.5} \log^{1.5} n$
$n \log n$	$n^{2.3728639}$	$n^2 \log n$	$n^{1.687}$	$n^{1.5} \log^{1.5} n$
$(n \log n)$	$(n^2)$	$(n^2 \log n)$	$(n^{1.5} \log n)$	$(n^{1.5} \log^{1.5} n)$

BK 2.1 wins for small  $n$ , BK 2.2 wins for large  $n$

BK 2.1 wins with hypothetical  $O(n^2)$  matrix multiplication

# The Lagrange inversion theorem

The coefficients of  $F^{-1}$  are given by

$$[x^k]F^{-1} = \frac{1}{k}[x^{k-1}] \left( \frac{x}{F} \right)^k.$$

$O(nM(n))$  algorithm: compute  $H = x/F$  using Newton iteration for the reciprocal, then compute  $H, H^2, H^3, \dots, H^{n-1}$

Better than classical algorithms ( $O(n^3)$ )

Similar to Newton iteration + Horner's rule

Worse than Newton iteration + BK2.1 or BK2.2

## Fast Lagrange inversion

We can extract a single coefficient in  $H^a = H^b H^c$  using  $O(n)$  operations:

$$[x^k]H^a = \sum_{i=0}^k ([x^i]H^a) \cdot ([x^{k-i}]H^c)$$

Compute  $H, H^2, \dots, H^{m-1}$ , then  $H^m, H^{2m}, H^{3m}, \dots, H^{\lfloor n/m \rfloor m}$ .

Choose  $m \approx \sqrt{n} \Rightarrow$  need  $O(\sqrt{n})$  polynomial multiplications, plus  $O(n^2)$  coefficient operations.

## Fast Lagrange inversion, first version

```
1:  $m \leftarrow \lfloor \sqrt{n} \rfloor$ 
2:  $H \leftarrow x/F$ 
3: for  $1 \leq i < m$  do
4:    $H^{i+1} \leftarrow H^i \cdot H$ 
5:    $b_i \leftarrow \frac{1}{i} [x^{i-1}] H^i$ 
6: end for
7:  $T \leftarrow H^m$ 
8: for  $i = m, 2m, 3m, \dots, \ell m < n$  do
9:    $b_i \leftarrow \frac{1}{i} [x^{i-1}] T$ 
10:  for  $1 \leq j < m$  while  $i + j < n$  do
11:     $b_{i+j} \leftarrow \frac{1}{i+j} \sum_{k=0}^{i+j-1} ([x^k] T) \cdot ([x^{i+j-k-1}] H^j)$ 
12:  end for
13:   $T \leftarrow T \cdot H^m$ 
14: end for
15: return  $b_1 x + b_2 x^2 + \dots + b_{n-1} x^{n-1}$ 
```

## Using matrix multiplication

Example:  $n = 8$ ,  $m = \lceil \sqrt{8-1} \rceil = 3$ . We want the coefficients  $b_1, b_2, \dots, b_7$  of  $1, x, \dots, x^6$  in the respective powers of  $H$ .

$$h_{k,i} = [x^i]H^k$$

$$A = \begin{pmatrix} h_{0,2} & h_{0,1} & h_{0,0} & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{3,5} & h_{3,4} & h_{3,3} & h_{3,2} & h_{3,1} & h_{3,0} & 0 & 0 & 0 \\ - & - & h_{6,6} & h_{6,5} & h_{6,4} & h_{6,3} & h_{6,2} & h_{6,1} & h_{6,0} \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 & h_{1,0} & h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} & h_{1,6} \\ 0 & h_{2,0} & h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & h_{2,5} & h_{2,6} & - \\ h_{3,0} & h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} & h_{3,6} & - & - \end{pmatrix}$$

$$AB^T = \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & - & - \end{pmatrix}$$

## Fast Lagrange inversion, matrix version

- 1:  $m \leftarrow \lceil \sqrt{n-1} \rceil$
- 2:  $H \leftarrow x/F$
- 3: {Assemble  $m \times m^2$  matrices  $B$  and  $A$  from  $H, H^2, \dots, H^m$  and  $H^m, H^{2m}, H^{3m}, \dots$ }
- 4: **for**  $1 \leq i \leq m, 1 \leq j \leq m^2$  **do**
- 5:    $B_{i,j} \leftarrow [x^{i+j-m-1}] H^i$
- 6:    $A_{i,j} \leftarrow [x^{im-j}] H^{(i-1)m}$
- 7: **end for**
- 8:  $C \leftarrow AB^T$
- 9: **for**  $1 \leq i < n$  **do**
- 10:    $b_i \leftarrow C_i/i$  ( $C_i$  is the  $i$ th entry of  $C$  read rowwise)
- 11: **end for**
- 12: **return**  $b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1}$

## Comparison with BK 2.1

Let  $m = \sqrt{n}$ .

Fast Lagrange inversion for reversion:

1.  $2m + O(1)$  polynomial multiplications
2. One  $(m \times m^2)$  times  $(m^2 \times m)$  matrix multiplication
3.  $O(n)$  additional operations

“Fast Horner’s rule” (BK 2.1) for composition:

1.  $m$  polynomial multiplications, each with cost  $M(n)$
2. One  $(m \times m)$  times  $(m \times m^2)$  matrix multiplication
3.  $m$  polynomial multiplications and additions

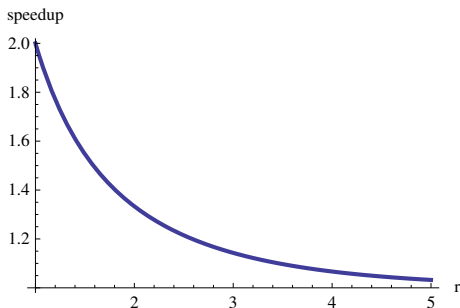
Both  $O(n^{1/2}(M(n) + MM(n^{1/2})))$ , same constant factor.



## Speedup by avoiding Newton iteration

If composition costs  $C(n) \sim cn^r$ , reversion via Newton costs

$$C(n) + C(n/2) + C(n/4) + \dots = cn^r \left( \frac{2^r}{2^r - 1} \right)$$



Classical polynomial multiplication:  $\frac{4}{31}(8 + \sqrt{2}) \approx 1.214$

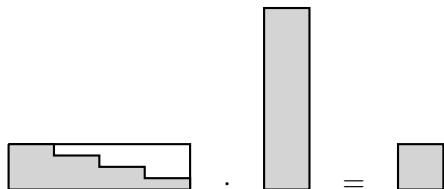
FFT polynomial multiplication, classical matrix multiplication:  $4/3$

# Faster matrix multiplication

Brent-Kung 2.1:



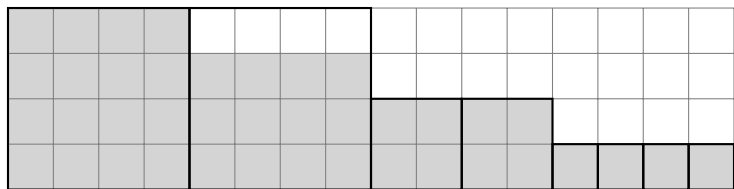
Fast Lagrange inversion:



Can we exploit the structure of  $A$  when  $\omega < 3$ ?

## Faster matrix multiplication, first algorithm

Decompose each  $m \times m$  block of  $A$  into  $(m/k) \times (m/k)$  blocks.

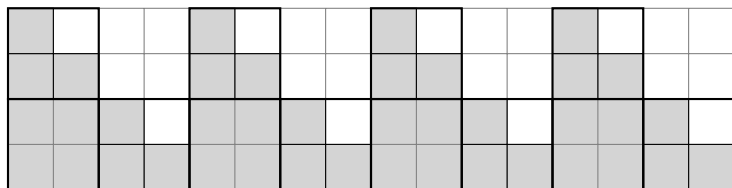


Asymptotic speedup  $s$ :

$$s = \frac{m^{\omega+1}}{\sum_{k=1}^{\infty} \left( \frac{m}{k} - \frac{m}{k+1} \right) k^2 \left( \frac{m}{k} \right)^{\omega}} > \frac{1}{\sum_{k=0}^{\infty} \frac{2^{k-1}}{2^{k\omega}}} = 2 - 2^{2-\omega} > 1$$

## Faster matrix multiplication, second algorithm

Write  $AB^T = (AP)(P^{-1}B^T)$  where  $P$  is a permutation matrix that makes each  $m \times m$  block in  $A$  lower triangular.

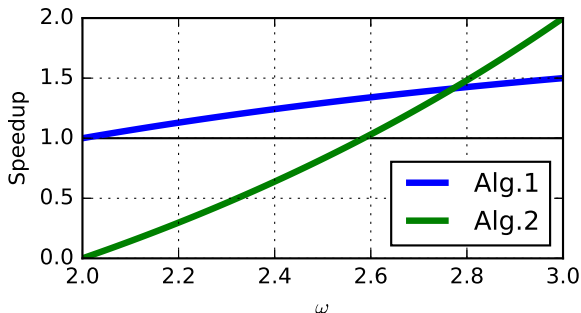


Recursive triangular multiplication:

$$R(k) = 4R(k/2) + 2(k/2)^\omega + O(k^2)$$

Speedup:  $s = 2^{\omega-1} - 2$ . Note:  $s > 1$  when  $\omega > \log_2 6 \approx 2.585$

## Matrix multiplication speedup



Crossover at  $\omega = 1 + \log_2(2 + \sqrt{2}) \approx 2.771$

$s = 2$  with classical matrix multiplication ( $\omega = 3$ )

$s = 1.5$  with Strassen multiplication

$s = 1.228$  with Coppersmith-Winograd-Le Gall

## Using fast rectangular matrix multiplication

$MM(x, y, z)$ :  $(x \times y)$  by  $(y \times z)$  matrix multiplication

Huang and Pan (1998):

$$MM(m, m, m^2) = O(n^{1.667}) < mMM(m, m, m) = O(n^{1.687})$$

By a transposition argument,

$$MM(m, m^2, m) = (1 + o(1))MM(m, m, m^2)$$

Open problem: can we get a speedup  $> 1$ ?

## Total speedup

Theoretical speedup of fast Lagrange inversion over BK 2.1 by both avoiding Newton iteration and speeding up matrix multiplication.

Dominant operation	Complexity	Newton	Matrix	Total
Polynomial, classical	$O(n^{5/2})$	1.214	1	1.214
Polynomial, Karatsuba	$O(n^{1/2+\log_2 3})$	1.308	1	1.308
Matrix, classical	$O(n^2)$	1.333	2.000	2.666
Matrix, Strassen	$O(n^{(1+\log_2 7)/2})$	1.364	1.500	2.047
Matrix, Cop.-Win.-LG	$O(n^{1.687})$	1.451	1.228	1.781
Matrix, Huang-Pan	$O(n^{1.667})$	1.458	1?	1.458?
(Polynomial, FFT)	$O(n^{3/2} \log^{1+o(1)} n)$	1.546	1	1.546

# Implementation

Composition and reversion implemented in FLINT:

- ▶  $\mathbb{Z}/p\mathbb{Z}$ ,  $p < 2^{64}$
- ▶  $\mathbb{Z}$
- ▶  $\mathbb{Q}$

In Arb (interval coefficients):

- ▶  $\mathbb{R}$
- ▶  $\mathbb{C}$



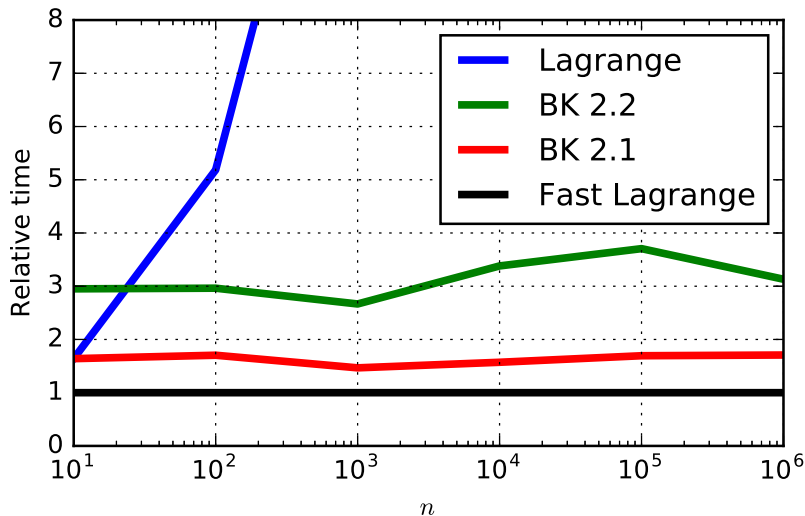
# Reversion over $\mathbb{Z}/p\mathbb{Z}$ ( $p = 2^{63} + 29$ )

$n$	Lagrange	BK 2.1	BK 2.2	<b>Fast Lagrange</b>
10	$10 \cdot 10^{-6}$	$10 \cdot 10^{-6}$	$18 \cdot 10^{-6}$	$6.1 \cdot 10^{-6}$
$10^2$	0.0028	0.00092	0.0016	0.00054
$10^3$	0.690	0.066	0.120	0.045
$10^4$	110	3.3 (8%)	7.1	2.1
$10^5$	12100	144 (20%)	315	85
$10^6$	1900000	8251 (28%)	15131	4832

Time in seconds.

(Relative time spent on matrix multiplication.)

# Reversion over $\mathbb{Z}/p\mathbb{Z}$ ( $p = 2^{63} + 29$ )



## Reversion over $\mathbb{Z}$

$$F_1 = \sum_{k \geq 1} k! x^k, \quad F_2 = \frac{x}{\sqrt{1-4x}}, \quad F_3 = \frac{x+x^2}{1+x+x^2}$$

$n$	BK 2.1			Fast Lagrange		
	$F_1$	$F_2$	$F_3$	$F_1$	$F_2$	$F_3$
$10$	$10 \cdot 10^{-6}$	$10 \cdot 10^{-6}$	$10 \cdot 10^{-6}$	$4.9 \cdot 10^{-6}$	$4.3 \cdot 10^{-6}$	$4.1 \cdot 10^{-6}$
$10^2$	0.0078	0.0021	0.0021	0.0064	0.00096	0.00065
$10^3$	10	1.1	0.96	7.1	0.71	0.22
$10^4$	24356	1453	538	8903	426	152

Time in seconds.

## Reversion over $\mathbb{Q}$

$$F_1 = \exp(x) - 1, \quad F_2 = x \exp(x), \quad F_3 = \frac{3x(1 - x^2)}{2(1 - x + x^2)^2}$$

$n$	BK 2.1			Fast Lagrange		
	$F_1$	$F_2$	$F_3$	$F_1$	$F_2$	$F_3$
10	$31 \cdot 10^{-6}$	$28 \cdot 10^{-6}$	$28 \cdot 10^{-6}$	$11 \cdot 10^{-6}$	$11 \cdot 10^{-6}$	$9.1 \cdot 10^{-6}$
$10^2$	0.012	0.021	0.0088	0.0089	0.0081	0.0019
$10^3$	8.8	17	3.1	14	13	0.65
$10^4$	13812	27057	1990	35633	27823	784

Time in seconds.

## Reversion over $\mathbb{R}$

$$F_1 = \exp(x) - 1, \quad F_2 = \log(1 + x), \quad F_3 = \Gamma(1 + x) - 1$$

$n$	BK 2.1			Fast Lagrange		
	$F_1$	$F_2$	$F_3$	$F_1$	$F_2$	$F_3$
10	$26 \cdot 10^{-6}$	$24 \cdot 10^{-6}$	$52 \cdot 10^{-6}$	$17 \cdot 10^{-6}$	$17 \cdot 10^{-6}$	$39 \cdot 10^{-6}$
$10^2$	0.0042	0.059	0.0049	0.0031	0.044	0.0031
$10^3$	0.79	91	0.53	0.41	61	0.33
$10^4$	154		207	364		74

Precision doubled until the coefficient of  $x^{n-1}$  is a precise interval.  
Time in seconds.

## Conclusion and questions

Reversion algorithm analogous to BK 2.1, avoiding Newton iteration

*Usually* the fastest algorithm for reversion in practice.

Is there a Newton-free analog of BK 2.2 for reversion?

Can more be said about the structured matrix products?

Can denominators be handled better? Numerical stability?

Note: the algorithm has been published in FJ, *A fast algorithm for reversion of power series*, Math. Comp. 84, 475–484, 2015. Some more comments in my PhD thesis.