Faster integer multiplication using short lattice vectors

David Harvey and Joris van der Hoeven

ANTS XIII, University of Wisconsin, Madison, July 2018

University of New South Wales / CNRS, École Polytechnique



Main result

$\mathrm{M}(n) :=$ time needed to multiply n-bit integers.

("time" $=$ # steps on a multi-tape Turing machine.)

Theorem (H., van der Hoeven, 2018)

There is an integer multiplication algorithm achieving

$$\mathrm{M}(n) = O(n \log n \, 4^{\log^* n}).$$

$\log^* x$ is the iterated logarithm:

$$\log^* x := \begin{cases} 0 & \text{if } x \leq 1, \\ 1 + \log^*(\log x) & \text{if } x > 1. \end{cases}$$

Example: $\log^*(e^{(e^{(e^{(e^{e})})})}) = 6.$

History of bounds for $\mathsf{M}(n)$

$<$ 1962	?	n^2
1962	Karatsuba	$n^{\log 3/\log 2} \approx n^{1.58}$
1963	Toom	$n\,2^{5\sqrt{\log_2 n}}$
1966	Schönhage	$n\,2^{\sqrt{2\log_2 n}}(\log n)^{3/2}$
1969	Knuth	$n\,2^{\sqrt{2\log_2 n}}\log n$
1971	Schönhage–Strassen	$n \log n \log\log n$
2007	Fürer	$n \log n\, K^{\log^* n}$ for some $K > 1$

Fürer's algorithm recurses from size n to size n' exponentially smaller than n. Number of recursion levels is $\log^* n + O(1)$.

The const-  K measures the "expansion factor" at each level.

Progress on the value of K

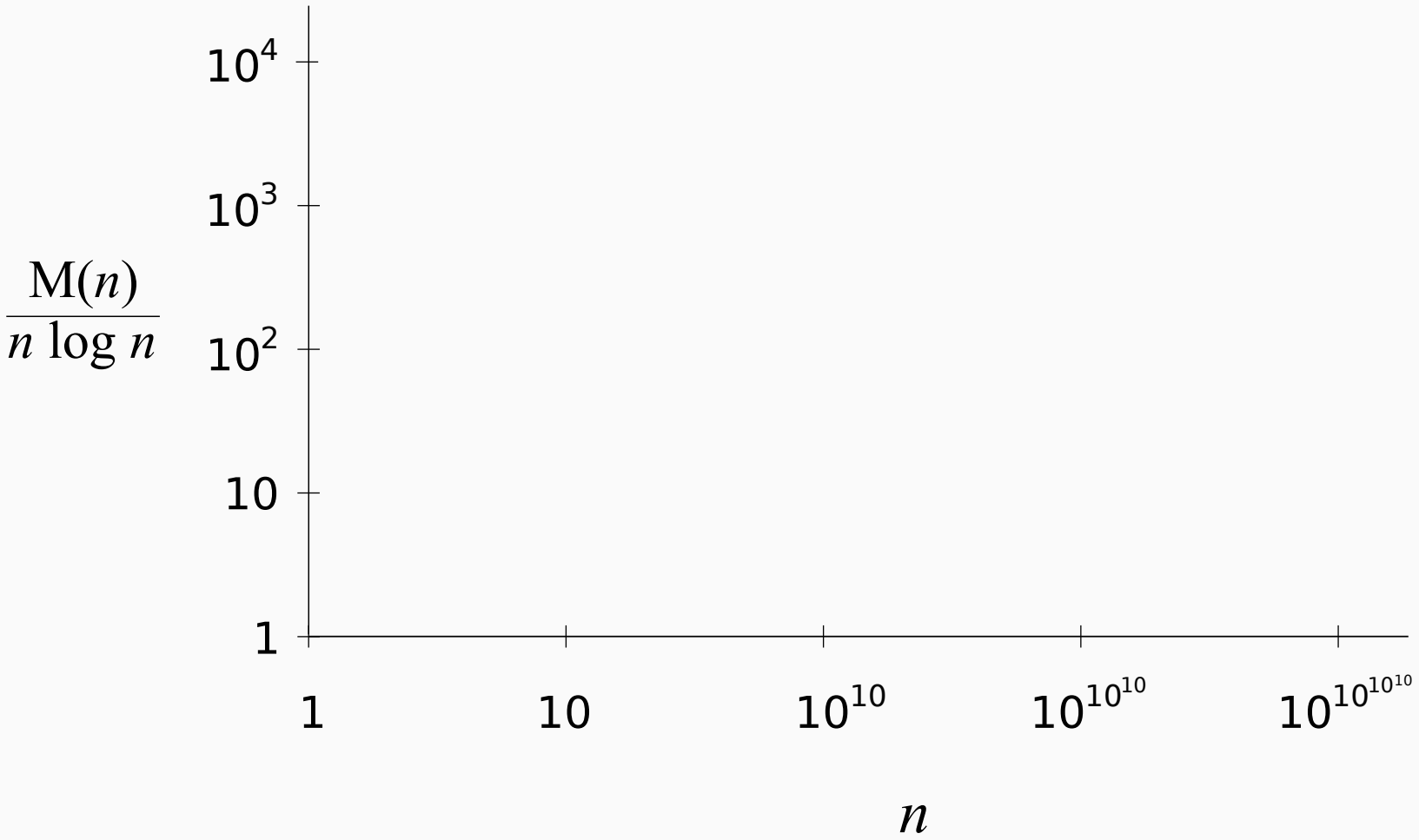
2014	H.–van der Hoeven–Lecerf	$K = 8$
†2014	H.–van der Hoeven–Lecerf	$K = 4$
†2015	Covanov–Thomé	$K = 4$
†2016	H.–van der Hoeven	$K = 4$
*2017	H.	$K = 6$
**2017	H.–van der Hoeven	$K = 4\sqrt{2} \approx 5.7$
2018	H.–van der Hoeven	$K = 4$

†: depends on unproved number-theoretic conjecture

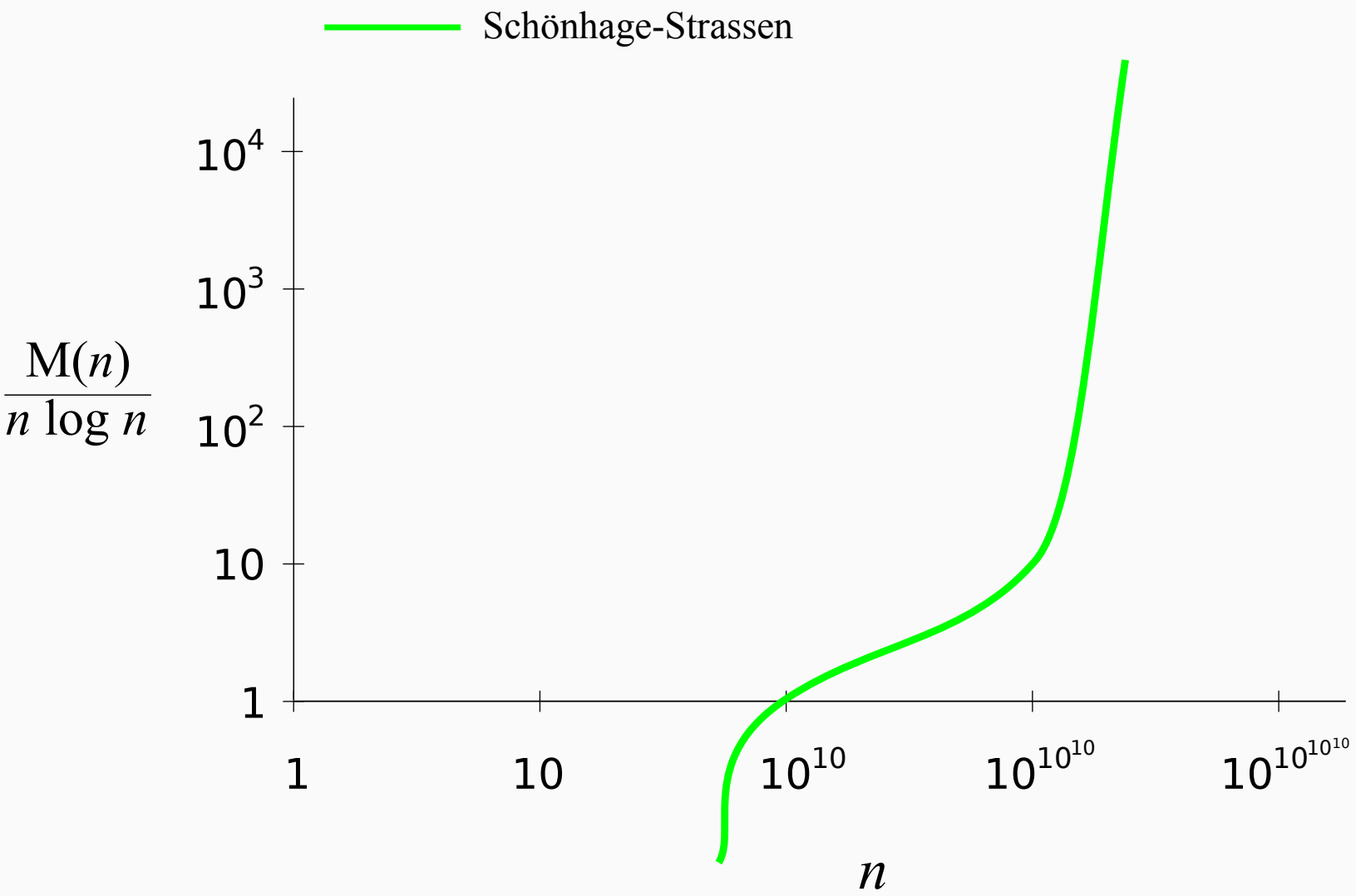
*: submitted, under review

**: preprint, do not  -icipate publication

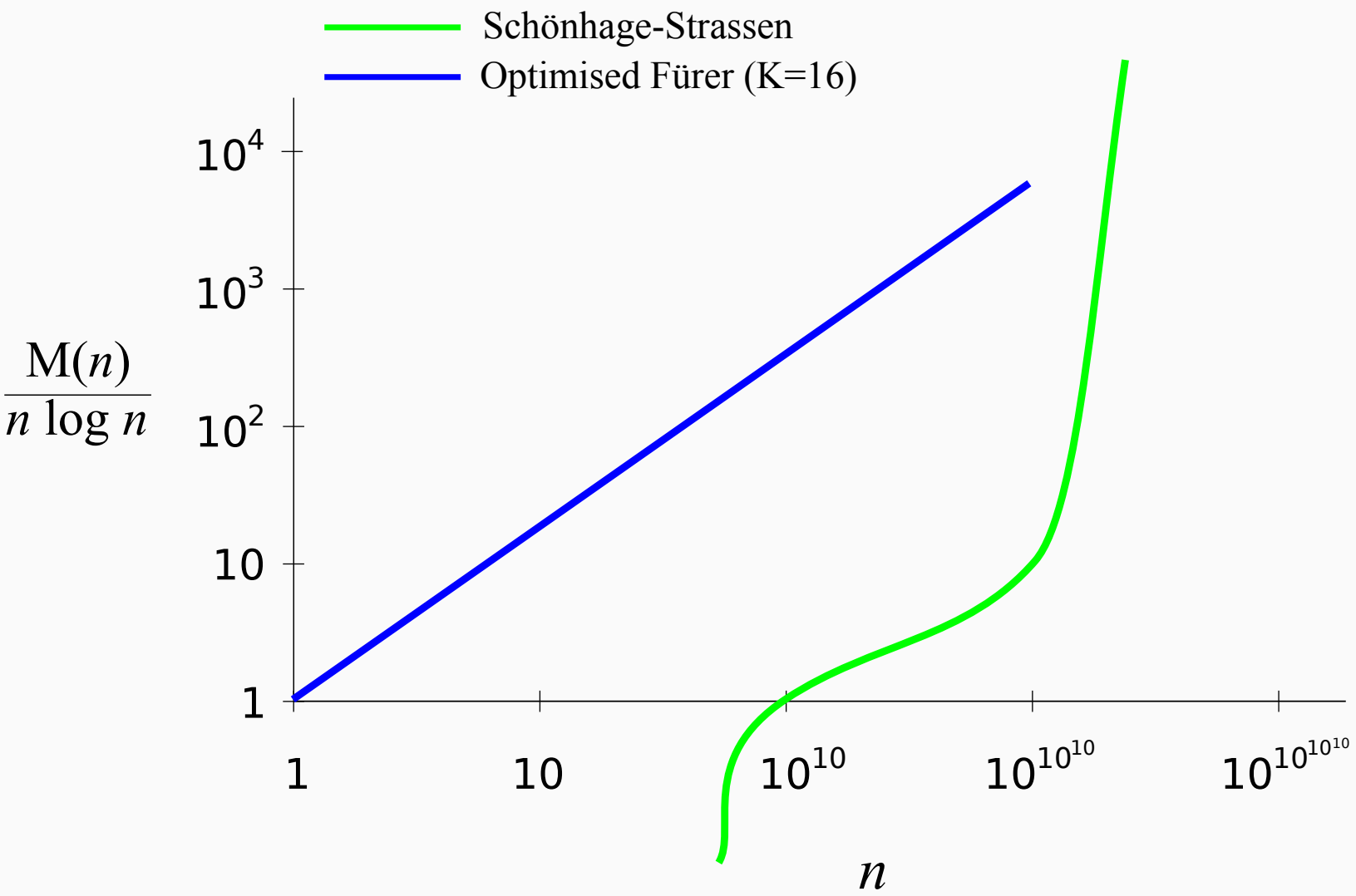
Implementation and performance



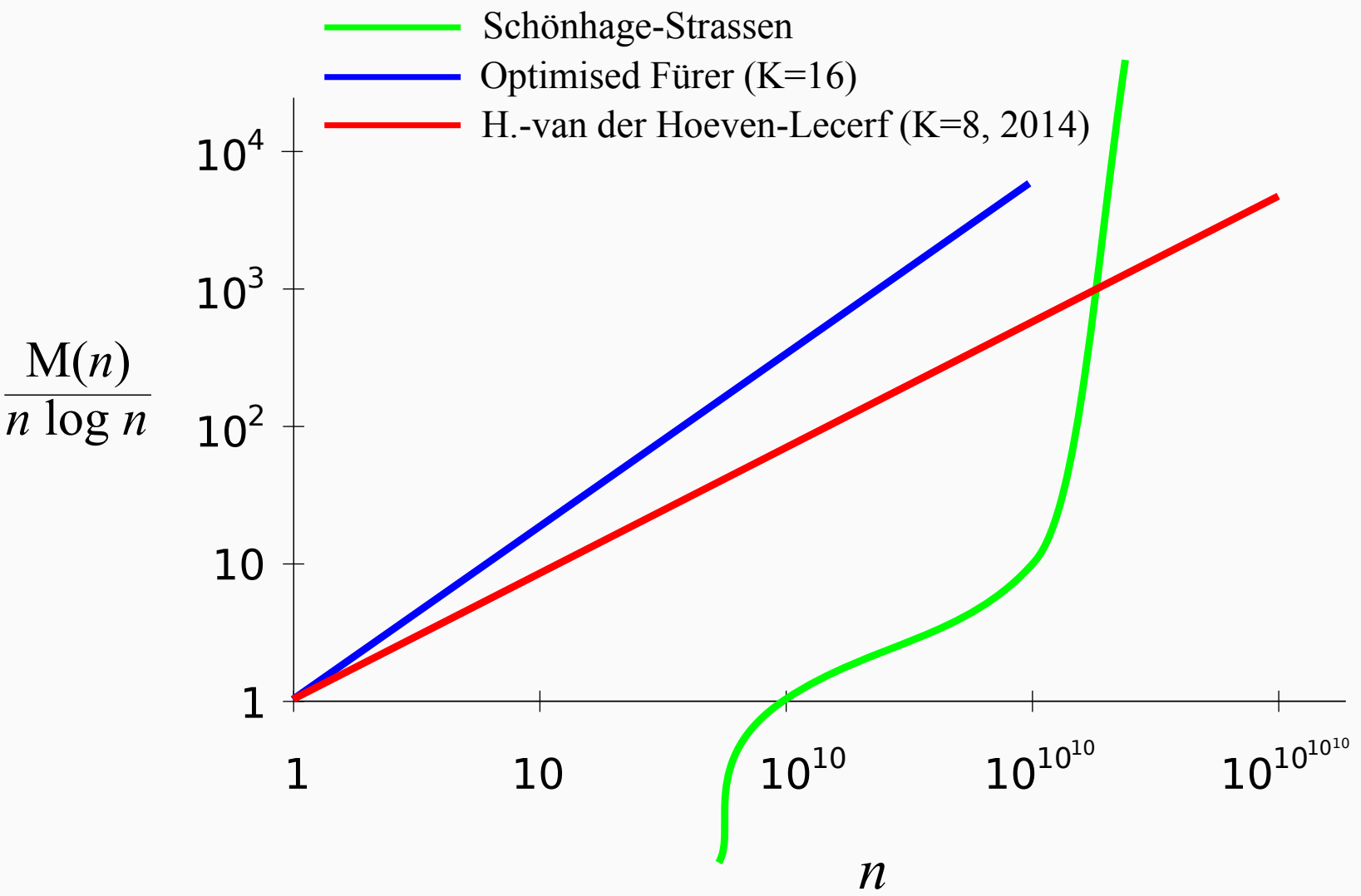
Implementation and performance



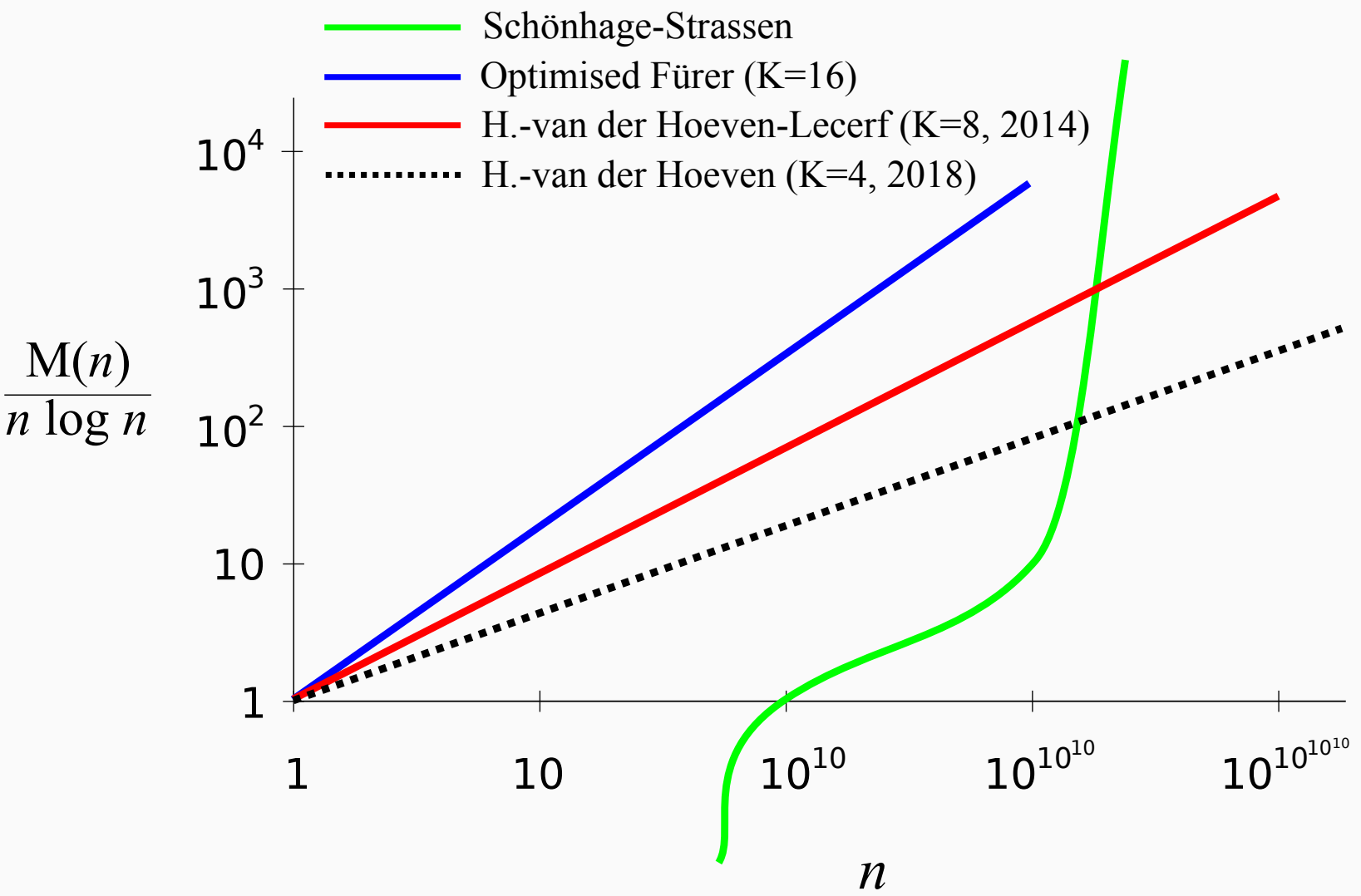
Implementation and performance



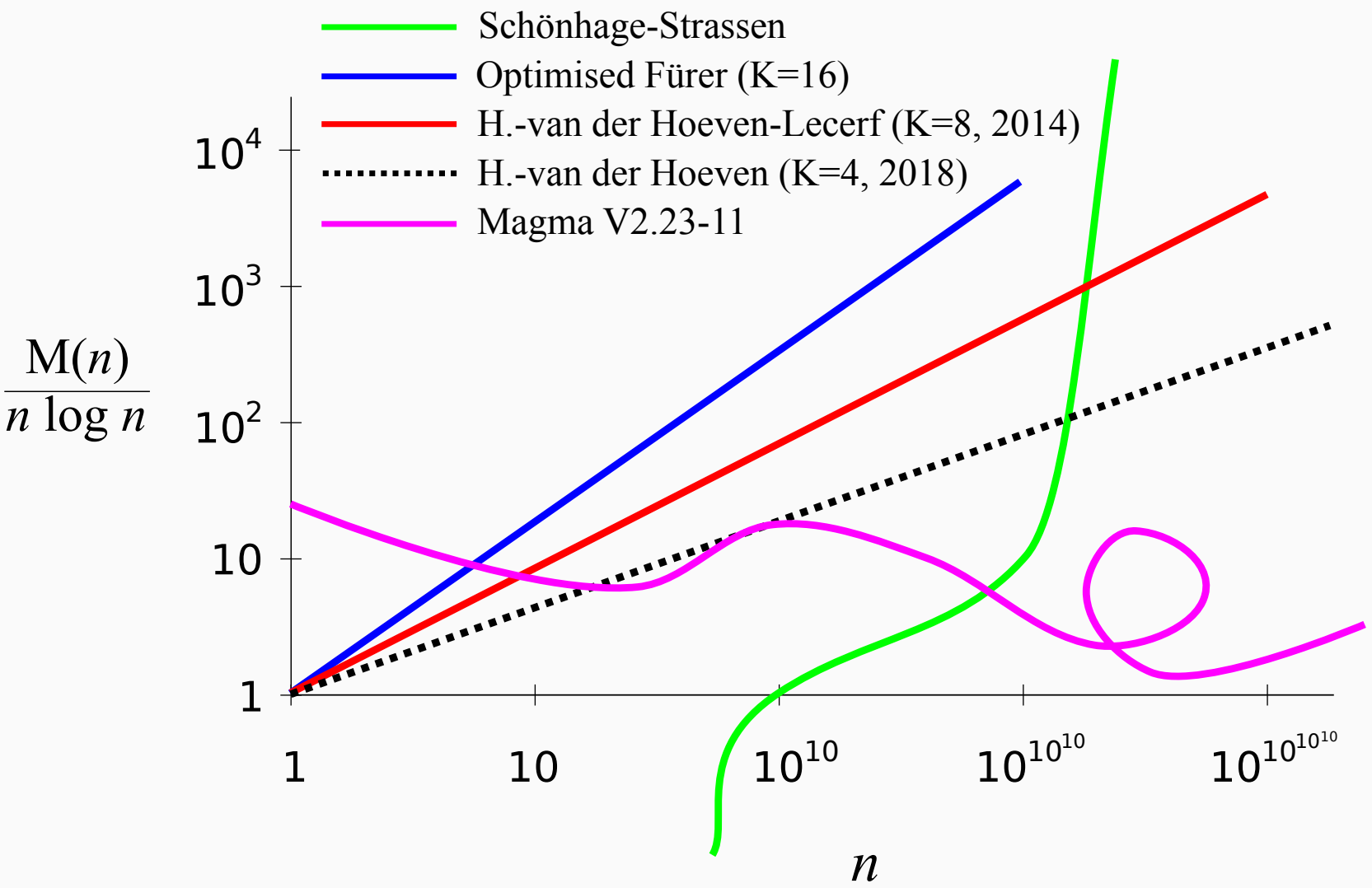
Implementation and performance



Implementation and performance



Implementation and performance



Overview of new algorithm

Same overall structure as 2014 algorithm.

0) Start with multiplication problem of size n.

Overview of new algorithm

Same overall structure as 2014 algorithm.

0) Start with multiplication problem of size n.
1) Reduce to convolution over "nice" coefficient ring, coefficient size expon-  -ially smaller than n.

Overview of new algorithm

Same overall structure as 2014 algorithm.

0) Start with multiplication problem of size n.
1) Reduce to convolution over "nice" coefficient ring, coefficient size expon-  -ially smaller than n.
2) Reduce to "long" DFTs, transform length comparable to n.

Overview of new algorithm

Same overall structure as 2014 algorithm.

0) Start with multiplication problem of size n.
1) Reduce to convolution over "nice" coefficient ring,
   coefficient size expon-  -ially smaller than n.
2) Reduce to "long" DFTs, transform length comparable to n.
3) Use Cooley–Tukey to reduce to "short" DFTs,
   transform length exponentially smaller than n.

Overview of new algorithm

Same overall structure as 2014 algorithm.

0) Start with multiplication problem of size n.
1) Reduce to convolution over "nice" coefficient ring,
   coefficient size expon-  -ially smaller than n.
2) Reduce to "long" DFTs, transform length comparable to n.
3) Use Cooley–Tukey to reduce to "short" DFTs,
   transform length exponentially smaller than n.
4) Use Bluestein to reduce to "short" multiplications,
   size exponentially smaller than n.

Overview of new algorithm

Same overall structure as 2014 algorithm.

0) Start with multiplication problem of size n.
1) Reduce to convolution over "nice" coefficient ring, coefficient size expon-  -ially smaller than n.
2) Reduce to "long" DFTs, transform length comparable to n.
3) Use Cooley–Tukey to reduce to "short" DFTs, transform length exponentially smaller than n.
4) Use Bluestein to reduce to "short" multiplications, size exponentially smaller than n.
5) Recurse!

Steps 1–3 are same as Fürer. Step 4 eliminates need for Fürer's "fast roots of unity".

Why did we get $K = 8$ in the 2014 algorithm?

Three factors of 2 from different sources:

a) *FFT multiplication*. Need to recurse into both forward and inverse DFTs.

Why did we get $K = 8$ in the 2014 algorithm?

Three factors of 2 from different sources:

a) *FFT multiplication*. Need to recurse into both forward and inverse DFTs.

b) *Coefficient growth.* If f and g have integer coefficients with k bits, then the coefficients of fg have roughly $2k$ bits.

Why did we get $K = 8$ in the 2014 algorithm?

Three factors of 2 from different sources:

a) *FFT multiplication*. Need to recurse into both forward and inverse DFTs.

b) *Coefficient growth.* If f and g have integer coefficients with k bits, then the coefficients of fg have roughly $2k$ bits.

c) *Truncated product problem.* The algorithm works over $\mathbf{C}$. When multiplying complex numbers with k-bit m-  -issa, need to compute product with $2k$ bits and then truncate.

Why did we get $K = 8$ in the 2014 algorithm?

Three factors of 2 from different sources:

a) *FFT multiplication*. Need to recurse into both forward and inverse DFTs.

b) *Coefficient growth.* If f and g have integer coefficients with k bits, then the coefficients of fg have roughly $2k$ bits.

c) *Truncated product problem.* The algorithm works over $\mathbf{C}$. When multiplying complex numbers with k-bit m-  -issa, need to compute product with $2k$ bits and then truncate.

All of the $K = 4$ algorithms (both conditional and unconditional) attack (c) by replacing $\mathbf{C}$ with $\mathbf{Z}/q\mathbf{Z}$ for a suitable integer q.

Primes with cyclic structure

This may be adv-  -ageous if q has *cyclic structure* that maps efficiently onto FFTs.

- H.–van der Hoeven–Lecerf (2014), Mersenne primes:

$$q = 2^p - 1 \quad \Longrightarrow \quad x^n - 1.$$

Primes with cyclic structure

This may be adv-  -ageous if q has *cyclic structure* that maps efficiently onto FFTs.

- H.–van der Hoeven–Lecerf (2014), Mersenne primes:

$$q = 2^p - 1 \quad \Longrightarrow \quad x^n - 1.$$

- Covanov–Thomé (2015), generalised Fermat primes:

$$q = r^{2^\lambda} + 1 \quad \Longrightarrow \quad x^{2^\lambda} + 1.$$

Primes with cyclic structure

This may be adv-  -ageous if q has *cyclic structure* that maps efficiently onto FFTs.

- H.–van der Hoeven–Lecerf (2014), Mersenne primes:

$$q = 2^p - 1 \quad \Longrightarrow \quad x^n - 1.$$

- Covanov–Thomé (2015), generalised Fermat primes:

$$q = r^{2^\lambda} + 1 \quad \Longrightarrow \quad x^{2^\lambda} + 1.$$

- H.–van der Hoeven (2016), plain vanilla FFT primes:

$$q = a \cdot 2^m + 1 \quad \Longrightarrow \quad x^m + a.$$

The new algorithm

The idea of the new algorithm is to manufacture cyclic structure for an almost *arbitrary* prime q.

Goal: reduce arithmetic in $\mathbf{Z}/q\mathbf{Z}$ to arithmetic in $\mathbf{Z}[x]/(x^m + 1)$, assuming that $q = 1 \pmod{2m}$.

The new algorithm

The idea of the new algorithm is to manufacture cyclic structure for an almost *arbitrary* prime q.

Goal: reduce arithmetic in $\mathbf{Z}/q\mathbf{Z}$ to arithmetic in $\mathbf{Z}[x]/(x^m + 1)$, assuming that $q = 1 \pmod{2m}$.

In the rest of the talk I will demonstrate the method for

$$q = 31415926535897932384626833, \qquad m = 4.$$

The condition $q = 1 \pmod 8$ guar-  -ees existence of a primitive 8-th root of unity modulo q:

$$\theta = 254253343156690445092273 5 \bmod q.$$

θ-representations

A *θ-representation* for $u \in \mathbf{Z}/q\mathbf{Z}$ is an expression

$$u = a_{m-1}\theta^{m-1} + \cdots + a_1\theta + a_0,$$

where the a_i are integers such that $|a_i| \leq mq^{1/m}$.

θ-representations

A *θ-representation* for $u \in \mathbf{Z}/q\mathbf{Z}$ is an expression

$$u = a_{m-1}\theta^{m-1} + \cdots + a_1\theta + a_0,$$

where the a_i are integers such that $|a_i| \leq mq^{1/m}$.

For inst-  :

$$\begin{aligned} u &= 271828182845904523536028 \pmod{q} \\ &= 3366162 \cdot \theta^3 + 951670 \cdot \theta^2 - 5013490 \cdot \theta - 3202352. \end{aligned}$$

θ-representations

A *θ-representation* for $u \in \mathbf{Z}/q\mathbf{Z}$ is an expression

$$u = a_{m-1}\theta^{m-1} + \cdots + a_1\theta + a_0,$$

where the a_i are integers such that $|a_i| \leq mq^{1/m}$.

For inst-  :

$$\begin{aligned} u &= 271828182845904523536028 \pmod{q} \\ &= 3366162 \cdot \theta^3 + 951670 \cdot \theta^2 - 5013490 \cdot \theta - 3202352. \end{aligned}$$

Are θ-representations unique? Not necessarily:

$$u = -4133936 \cdot \theta^3 + 1849981 \cdot \theta^2 - 5192184 \cdot \theta + 1317423.$$

θ-representations

A *θ-representation* for $u \in \mathbf{Z}/q\mathbf{Z}$ is an expression

$$u = a_{m-1}\theta^{m-1} + \cdots + a_1\theta + a_0,$$

where the a_i are integers such that $|a_i| \leq mq^{1/m}$.

For inst-  :

$$\begin{aligned} u &= 271828182845904523536028 \pmod{q} \\ &= 3366162 \cdot \theta^3 + 951670 \cdot \theta^2 - 5013490 \cdot \theta - 3202352. \end{aligned}$$

Are θ-representations unique? Not necessarily:

$$u = -4133936 \cdot \theta^3 + 1849981 \cdot \theta^2 - 5192184 \cdot \theta + 1317423.$$

Notice that the total bitsize of $(a_0, \ldots, a_{m-1})$ is about the same as the bitsize of q.

Proposed ANTS by-laws

⋮

By-law #691: *any speaker whose presentation includes, in the expert opinion of the Steering Committee, a distastefully excessive number of low-resolution bitmaps, cartoons, or puns, pertaining to formic acid, colony social behaviour, or any species of the order Hymenoptera (but not including termites, which are members of the order Blattodea and are more closely related to cockroaches), shall be banned from ANTS for a period of not more than twenty-four months; or in the case that the subsequent ANTS is held in New Zealand, eighteen months.*

⋮

Arithmetic on θ-representations

How do we multiply elements of $\mathbf{Z}/q\mathbf{Z}$ in θ-representation?

Suppose we want to multiply

$$\begin{aligned} u &= 1414213562373095048801689 \pmod{q} \\ &= 3740635 \cdot \theta^3 + 3692532 \cdot \theta^2 - 3089740 \cdot \theta + 4285386 \end{aligned}$$

by

$$\begin{aligned} v &= 1732050807568877293527447 \pmod{q} \\ &= 4629959 \cdot \theta^3 - 4018180 \cdot \theta^2 - 2839272 \cdot \theta - 3075767. \end{aligned}$$

Arithmetic on θ-representations

First multiply as polynomials in θ, using the relation $\theta^m = -1$:

$$uv = 102668685436\,25 \cdot \theta^3 - 371231948042\,09 \cdot \theta^2 - 4729783170300 \cdot \theta + 2658245912907\,8.$$

Arithmetic on θ-representations

First multiply as polynomials in θ, using the relation $\theta^m = -1$:

$$uv = 102668685436\,25 \cdot \theta^3 - 371231948042\,09 \cdot \theta^2 - 4729783170300 \cdot \theta + 2658245912907\,8.$$

Problem: this is not a θ-representation.

The coefficients are too big.

We need a *reduction algorithm* to make the coefficients small again, without changing the value modulo q.

Arithmetic on θ-representations

Key idea: precompute a polynomial $P(x)$ giving a nontrivial representation of zero:

$$0 = P(\theta) = -394297 \cdot \theta^3 - 927319 \cdot \theta^2 + 1136523 \cdot \theta - 292956.$$

Arithmetic on θ-representations

Key idea: precompute a polynomial $P(x)$ giving a nontrivial representation of zero:

$$0 = P(\theta) = -394297 \cdot \theta^3 - 927319 \cdot \theta^2 + 1136523 \cdot \theta - 292956.$$

Finding $P(x)$ is equivalent to the problem of finding a short nonzero vector in the lattice

$$\Lambda := \{(a_3, \ldots, a_0) \in \mathbf{Z}^4 : a_3\theta^3 + \cdots + a_0 = 0 \pmod{q}\}.$$

In general, Minkowski's theorem guarantees the existence of such a vector with $|a_i| \leq q^{1/m}$ for all i.

(I used LLL to compute the example above.)

Arithmetic on θ-representations

Now we want to subtract an appropriate multiple of

$$0 = P(\theta) = -394297 \cdot \theta^3 - 927319 \cdot \theta^2 + 1136523 \cdot \theta - 292956$$

from

$$uv = 10266868543625 \cdot \theta^3 - 37123194804209 \cdot \theta^2 - 4729783170300 \cdot \theta + 26582459129078.$$

to make the coefficients small.

For technical reasons, we use a Montgomery-style modular reduction algorithm.

Arithmetic on θ-representations

Precompute an auxiliary prime r (around the size of $q^{1/m}$) such that $P(\theta)$ is invertible modulo r, and let $J(\theta)$ be its inverse.

In our example:

$$\begin{aligned} r &= 42602761, \\ J(\theta) &= 17106162 \cdot \theta^3 + 6504907 \cdot \theta^2 + 30962874 \cdot \theta + 8514380, \end{aligned}$$

so that

$$\begin{aligned} J(\theta)P(\theta) &= -29688032222177 \cdot \theta^3 + 32133728922904 \cdot \theta^2 \\ &\qquad + 19033763340253 \cdot \theta - 3695193078095 \\ &= 0 \cdot \theta^3 + 0 \cdot \theta^2 + 0 \cdot \theta + 1 \pmod{r} \\ &= 1 \pmod{r}. \end{aligned}$$

Arithmetic on θ-representations

Now we can compute the "Montgomery quotient":

$$\begin{aligned} Q(\theta) &:= (uv)J(\theta) \pmod{r} \\ &= 3932274 \cdot \theta^3 - 14729381 \cdot \theta^2 + 20464841 \cdot \theta - 11934644, \end{aligned}$$

Arithmetic on θ-representations

Now we can compute the "Montgomery quotient":

$$\begin{aligned} Q(\theta) &:= (uv)J(\theta) \pmod{r} \\ &= 3932274 \cdot \theta^3 - 14729381 \cdot \theta^2 + 20464841 \cdot \theta - 11934644, \end{aligned}$$

and then the "Montgomery remainder":

$$\begin{aligned} uv - Q(\theta)P(\theta) = 424307736538\,43 \cdot \theta^3 - 773147238\,13102 \cdot \theta^2 \\ + 169907905\,39259 \cdot \theta + 331448628\,52478. \end{aligned}$$

Arithmetic on θ-representations

Now we can compute the "Montgomery quotient":

$$\begin{aligned} Q(\theta) &:= (uv)J(\theta) \pmod{r} \\ &= 3932274 \cdot \theta^3 - 14729381 \cdot \theta^2 + 20464841 \cdot \theta - 11934644, \end{aligned}$$

and then the "Montgomery remainder":

$$\begin{aligned} uv - Q(\theta)P(\theta) = 42430773653843 \cdot \theta^3 - 77314723813102 \cdot \theta^2 \\ + 16990790539259 \cdot \theta + 33144862852478. \end{aligned}$$

By construction, this last polynomial is divisible by r.

Dividing by r, we get

$$\frac{uv}{r} = 995963 \cdot \theta^3 - 1814782 \cdot \theta^2 + 398819 \cdot \theta + 777998 \pmod{q}.$$

Finally, we have found a θ-representation for uv/r.

Arithmetic on θ-representations

Still need to remove the extraneous factor of r. Same issue as in standard Montgomery reduction; various workarounds are possible.

Arithmetic on θ-representations

Still need to remove the extraneous factor of r. Same issue as in standard Montgomery reduction; various workarounds are possible.

One also needs algorithms for addition/subtraction in θ-representation, and for conversions between standard and θ-representation. All straightforward with the tools already described.

Thank you!