

On the complexity of integer matrix multiplication

David Harvey, Joris van der Hoeven

▶ **To cite this version:**

David Harvey, Joris van der Hoeven. On the complexity of integer matrix multiplication. 2014. hal-01071191

HAL Id: hal-01071191
https://hal.science/hal-01071191

Preprint submitted on 3 Oct 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On the complexity of integer matrix multiplication

David Harvey

School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia

Email: d.harvey@unsw.edu.au

Joris van der Hoeven

LIX, CNRS, École polytechnique, 91128 Palaiseau Cedex, France

Email: vdhoeven@lix.polytechnique.fr

Abstract. Let $\mathsf{M}(n)$ denote the bit complexity of multiplying n-bit integers, let $\omega \in (2,3]$ be an exponent for matrix multiplication, and let $\lg^* n$ be the iterated logarithm. Assuming that $\log d = O(n)$ and that $\mathsf{M}(n)/(n \log n)$ is increasing, we prove that $d \times d$ matrices with n-bit integer entries may be multiplied in

$$O(d^2\,\mathsf{M}(n) + d^\omega\, n\, 2^{O(\lg^* n - \lg^* d)}\,\mathsf{M}(\lg d)/\lg d)$$

bit operations. In particular, if n is large compared to d, say $d = O(\log n)$, then the complexity is only $O(d^2\,\mathsf{M}(n))$.

Keywords: matrix multiplication, complexity, algorithm, FFT, Bluestein reduction

A.C.M. subject classification: G.1.0 Computer-arithmetic

A.M.S. subject classification: 68W30, 68Q17, 68W40

1. Introduction

In this paper we study the complexity of multiplying $d \times d$ matrices whose entries are integers with at most n bits. We are particularly interested in the case that n is very large compared to d, say $d = O(\log n)$. All complexity bounds refer to deterministic bit complexity, in the sense of the multi-tape Turing model [Pap94].

Matrices with large integer coefficients appear naturally in several areas. One first application is to the efficient high precision evaluation of so-called holonomic functions (such as exp, log, sin, Bessel functions, and hypergeometric functions) using a divide and conquer technique [CC90, HP97, Hoe99, Hoe01, Hoe07]. Another application concerns recent algorithms for computing the L-series of algebraic curves [Har14, HS14]. The practical running time in these applications is dominated by the multiplication of matrices with large integer entries, and it is vital to have a highly efficient implementation of this fundamental operation. Typical parameters for these applications are n around 10^8 bits, and d around 10.

In this paper, we focus mainly on theoretical bounds. We write $\mathsf{M}_d(n)$ for the cost of the $d \times d$ matrix multiplication, and $\mathsf{M}(n) := \mathsf{M}_1(n)$ for the cost of multiplying n-bit integers. We will also write $\mathsf{M}_{R,d}(n)$ for the algebraic complexity of multiplying $d \times d$ matrices whose entries are polynomials of degrees $<n$ over an abstract effective ring R, and $\mathsf{M}_R(n) := \mathsf{M}_{R,1}(n)$.

Schönhage and Strassen [SS71] used fast Fourier transforms (FFTs) to prove that $\mathsf{M}(n) = O(n \log n \log\log n)$ for large n. Fürer [Für09] improved this to $\mathsf{M}(n) = O\big(n \log n\, 2^{O(\lg^* n)}\big)$ where $\lg^*$ is the iterated logarithm, i.e.,

$$\begin{aligned} \lg n &:= \lceil \log_2 n \rceil, \\ \lg^* n &:= \min\{k \in \mathbb{N} : \lg^{\circ k} n \leqslant 1\}, \\ \lg^{\circ k} &:= \lg \circ \underset{k\times}{\cdots} \circ \lg, \end{aligned}$$

and this was recently sharpened to $\mathsf{M}(n) = O(n \log n\, 8^{\lg^* n})$ [HHL14a]. The best currently known bound [CK91] for $\mathsf{M}_R(n)$ is $\mathsf{M}_R(n) = O(n \log n \log\log n)$; if R is a ring of finite characteristic this may be improved to $\mathsf{M}_R(n) = O(n \log n\, 8^{\lg^* n})$ [HHL14b].

The algebraic complexity of $d \times d$ matrix multiplication is usually assumed to be of the form $O(d^\omega)$, where ω is a so-called exponent of matrix multiplication [vzGG03, Ch. 12]. Classical matrix multiplication yields $\omega = 3$, and Strassen's algorithm [Str69] achieves $\omega = \log 7 / \log 2 \approx 2.807$. The best currently known exponent $\omega < 2.3728639$ was found by Le Gall [Gal14, CW87].

When working over the integers and taking into account the growth of coefficients, the general bound for matrix multiplication specialises to

$$\mathsf{M}_d(n) \;=\; O(d^\omega\, \mathsf{M}(n + \lg d)).$$

Throughout this paper we will enforce the very mild restriction that $\log d = O(n)$. Under this assumption the above bound simplifies to

$$\mathsf{M}_d(n) \;=\; O(d^\omega\, \mathsf{M}(n)).$$

The main result of this paper is the following improvement.

Theorem 1. *Assume that $\mathsf{M}(n)/(n \log n)$ is increasing. Let $C > 1$ be a constant. Then*

$$\mathsf{M}_d(n) \;=\; O(d^2\, \mathsf{M}(n) + d^\omega\, n\, 2^{O(\lg^* n - \lg^* d)}\, \mathsf{M}(\lg d)/\lg d), \tag{1}$$

uniformly for all $d \geqslant 1$, $n \geqslant 2$ with $\lg d \leqslant C n$.

In particular, if n is large compared to d, say $d = O(\log n)$, then (1) simplifies to

$$\mathsf{M}_d(n) \;=\; O(d^2\, \mathsf{M}(n)). \tag{2}$$

This bound is essentially optimal (up to constant factors), in the sense that we cannot expect to do better for $d = 1$, and the bound grows proportionally to the input and output size as a function of d.

The new algorithm has its roots in studies of analogous problems in the algebraic complexity setting. When working over an arbitrary effective ring R, a classical technique for multiplying polynomial matrices is to use an evaluation-interpolation scheme. There are many different evaluation-interpolation strategies [Hoe10, Sections 2.1–2.3] such as Karatsuba, Toom–Cook, FFT, Schönhage–Strassen and general multi-point. The efficiency of a particular evaluation-interpolation strategy can be expressed in terms of two quantities: the complexity $\mathsf{E}_R(n)$ of evaluation/interpolation and the number $\mathsf{N}_R(n)$ of evaluation points. In terms of these quantities, we have

$$\mathsf{M}_{R,d}(n) = O(d^2\, \mathsf{E}_R(n) + d^\omega\, \mathsf{N}_R(n)). \tag{3}$$

If R admits many primitive 2^p-th roots of unity, then the FFT provides an efficient evaluation-interpolation strategy that achieves $\mathsf{E}_R(n) = O(n \log n)$ and $\mathsf{N}_R(n) = O(n)$. Moreover, when using the TFT [Hoe04], one may take $\mathsf{N}(n) = 2n - 1$, which is optimal. If R is a field of characteristic zero, or a finite field with sufficiently many elements, then Bostan and Schost proved [BS05, Thm. 4] that one may achieve $\mathsf{E}_R(n) = O(\mathsf{M}_R(n))$ and $\mathsf{N}_R(n) = 2n - 1$ by evaluating at geometric sequences. Thus, in this situation they obtain

$$\mathsf{M}_{R,d}(n) = O(d^2\, \mathsf{M}_R(n) + d^\omega\, n). \tag{4}$$

In the setting of integer coefficients, a popular evaluation-interpolation strategy is Chinese remaindering with respect to many small primes of bit length $O(\log n)$. Still assuming that $\log d = O(n)$, this yields the bound (see [Sto00, Lemma 1.7], for instance)

$$\mathsf{M}_d(n) \;=\; O(d^2\, \mathsf{M}(n) \log n + (n/\log n)\, \mathsf{M}_d(\lg n)),$$

and recursive application of this bound yields

$$\mathsf{M}_d(n) \;=\; O(d^2\, \mathsf{M}(n) \log n + d^\omega\, n\, 2^{O(\lg^* n - \lg^* d)}\, \mathsf{M}(\lg d)/\lg d).$$

Comparing with the algebraic bound (4), we notice an extra factor of $\log n$ in the first term. This factor arises from the cost of computing a certain product tree (and remainder tree) in the Chinese remaindering algorithm.

A well-known method that attempts to avoid the spurious $\log n$ factor is to use FFTs. For example, suppose that we are using the Schönhage–Strassen integer multiplication algorithm. This works by cutting up the integers into into chunks of about $\sqrt{n}$ bits, and then performs FFTs over a ring of the form $S = \mathbf{Z}/(2^{2^k}+1)\,\mathbf{Z}$ where $2^k \sim \sqrt{n}$. We can multiply integer matrices the same way, by cutting up each entry into chunks of about $\sqrt{n}$ bits, performing FFTs over S, and then multiplying the *matrices* of Fourier coefficients. When n is much larger than d, the latter step takes negligible time, and the bulk of the time is spent performing FFTs. Since each matrix entry is only transformed once, this leads to a bound of the type $O(d^2\,\mathsf{M}(n))$, without the extraneous $\log n$ factor. This method is very efficient in practice; both [HS14] and Mathemagix [HLM+02, HLQ14] contain implementations based on number-theoretic transforms (i.e., FFTs modulo word-sized prime numbers).

However, the theoretical question remains as to whether the $\log n$ overhead can be removed unconditionally, independently of the "internal structure" of the currently fastest algorithm for integer multiplication. Our Theorem 1 shows that this is indeed the case. More precisely, we reduce integer matrix multiplication to the multiplication of matrix polynomials over $\mathbf{Z}/p^\lambda\,\mathbf{Z}$ for a suitable prime power p^λ. The multiplication of such polynomials is done using FFTs. However, instead of using a classical algorithm for computing the FFT of an individual polynomial, we reduce this problem back to integer multiplication using Bluestein's trick [Blu70] and Kronecker substitution [vzGG03, Ch. 8]. This central idea of the paper will be explained in section 2. In section 3, we prove our main complexity bound (1).

We stress that Theorem 1 is a theoretical result, and we do not recommend our algorithm for practical computations. For any given FFT-based integer multiplication algorithm, it should always be better, by a constant factor, to apply the same FFT scheme to the matrix entries directly, as outlined above. See Remark 6 for further discussion about the implied big-O constant.

Remark 2. The observation that the Bluestein–Kronecker combination leads to a particularly efficient FFT algorithm was announced previously in [HHL14a]. We mention as a historical remark that the development of the main ideas of the present paper actually preceded [HHL14a].

2. Bluestein–Kronecker reduction

We begin with a lemma that converts a certain polynomial evaluation problem to integer multiplication.

Lemma 3. *Assume that $\mathsf{M}(n)/n$ is increasing. Let p be an odd prime, let $\lambda \geq 1$ be an integer, and let $\zeta \in (\mathbf{Z}/p^\lambda\,\mathbf{Z})^*$ be an element of order $p-1$. Given as input $F \in (\mathbf{Z}/p^\lambda\,\mathbf{Z})[x]$, with $\deg F < p-1$, we may compute $F(1), F(\zeta), \ldots, F(\zeta^{p-2}) \in \mathbf{Z}/p^\lambda\,\mathbf{Z}$ in time*

$$O(\mathsf{M}(\lambda\, p \lg p)).$$

Proof. Let $S = \mathbf{Z}/p^\lambda\,\mathbf{Z}$ and let $F = \sum_{j=0}^{p-2} F_j\, x^j \in S[x]$. We first use Bluestein's trick [Blu70] to convert the evaluation problem to a polynomial multiplication problem. Namely, from the identity $ij = \binom{i}{2} + \binom{-j}{2} - \binom{i-j}{2}$ we obtain

$$F(\zeta^i) = \sum_{j=0}^{p-2} F_j\, \zeta^{ij} = H_i \sum_{j=0}^{p-2} F'_j\, G_{i-j} \tag{5}$$

where

$$H_k = \zeta^{\binom{k}{2}}, \qquad F'_k = \zeta^{\binom{-k}{2}}\, F_k, \qquad G_k = \zeta^{-\binom{k}{2}}.$$

Since $H_{k+1} = \zeta^k\, H_k$ and $G_{k+1} = \zeta^{-k}\, G_k$, we may easily compute $H_0, \ldots, H_{p-2}$ and $G_{-p+2}, \ldots, G_{p-2}$ from ζ and $\zeta^{-1} = \zeta^{p-2}$ using $O(p)$ ring operations in S. Similarly we may obtain the F'_k from the F_k using $O(p)$ ring operations. The sum $\sum_{j=0}^{p-2} F'_j\, G_{i-j}$ in (5) may be interpreted as the coefficient of x^i in the product of the (Laurent) polynomials

$$F' = \sum_{k=0}^{p-2} F'_k\, x^k \qquad \text{and} \qquad G' = \sum_{k=-p+2}^{p-2} G_k\, x^k.$$

Thus it suffices to compute the product $F' \cdot (x^{p-2} G')$ in $S[x]$. To compute this product, we lift the problem to $\mathbf{Z}[x]$, and use Kronecker substitution [vzGG03, Ch. 8] to convert to an integer multiplication problem. The coefficients of F' and $x^{p-2} G'$ are bounded by p^λ, and their degrees by $2\,p-4$, so the coefficients of their product in $\mathbf{Z}[x]$ have at most $\lg(2\,p^{2\lambda+1}) = O(\lambda \lg p)$ bits. Therefore the integers being multiplied have at most $O(\lambda\ p\ \lg\ p)$ bits, leading to the desired $O(\mathsf{M}(\lambda\ p\ \lg\ p))$ bound. The remaining work consists of $O(p)$ ring operations in S, contributing a further $O(p\,\mathsf{M}(\lambda \lg p)) = O(\mathsf{M}(\lambda\, p \lg p))$ bit operations since $\mathsf{M}(n)/n$ is increasing. □

3. Integer matrix multiplication

Proposition 4. *Assume that $\mathsf{M}(n)/(n \log n)$ is increasing. Let $C > 1$ be a constant. Then*

$$\mathsf{M}_d(n) \;=\; O\Big(d^2\,\mathsf{M}(n) + \tfrac{n}{\lg n + \lg d}\,\mathsf{M}_d(\lg n + \lg d)\Big)$$

uniformly for all $d \geqslant 1$, $n \geqslant 2$ with $\lg d \leqslant C\,n$.

Proof. The input consists of matrices $A = (A_{ij})$ and $B = (B_{ij})$, where $1 \leq i, j \leq d$ and $A_{ij}, B_{ij} \in \mathbf{Z}$, $|A_{ij}| < 2^n$, $|B_{ij}| < 2^n$. We wish to compute the product $A\,B$.

Let $b := \lg n + \lg d$ and $m := \lceil n/b \rceil$. Note that $m = O(n/b)$ since we assumed that $b = O(n)$. We split the entries A_{ij} into chunks of b bits, choosing $P_{ij} \in \mathbf{Z}[x]$ so that $P_{ij}(2^b) = A_{ij}$ with $\deg P_{ij} < m$, and such that the coefficients of P_{ij} are bounded in absolute value by 2^b. Similarly choose $Q_{ij} \in \mathbf{Z}[x]$ for B_{ij}. Let $P = (P_{ij})$ and $Q = (Q_{ij})$ be the corresponding $d \times d$ matrices of polynomials. The coefficients of the entries of PQ are bounded in absolute value by $2^{2b}\,d\,m$, and thus have bit size bounded by $2\,b + \lg d + \lg m = O(b)$. The product $A\,B = (PQ)(2^b)$ may be deduced from PQ in time $O(d^2\,m\,b) = O(d^2\,n)$. Thus we have reduced to the problem of computing PQ.

The degrees of the entries of PQ are less than $2\,m$. Let p be the least odd prime such that $p \geq 2\,m$. By [BHP01] we have $p = 2\,m + O(m^{0.525}) = O(n/b)$. We may find p by testing candidates successively; the cost is $o(n)$, since each candidate requires $O((\log p)^{O(1)})$ bit operations [AKS04].

To compute PQ, it suffices to compute PQ modulo p^λ, where $\lambda \geq 1$ is the least positive integer for which $p^\lambda > 2 \cdot 2^{2b}\,d\,m$. Since $p^\lambda \leqslant 2 \cdot 2^{2b}\,d\,m\,p$ we have $\lambda \lg p \leqslant 2\,b + \lg d + \lg m + \lg p + 1 = O(b)$. Our plan is to compute PQ over $S := \mathbf{Z}/p^\lambda\,\mathbf{Z}$ by means of an evaluation-interpolation scheme, using Lemma 3 for the evaluations. The lemma requires a precomputed element $\zeta \in S^*$ of order $p-1$. To find ζ, we first compute a generator of $(\mathbf{Z}/p\,\mathbf{Z})^*$ in (deterministic) time $O(p^{1/4+\epsilon}) = o(n)$ [Shp96], and then lift it to a suitable $\zeta \in S^*$ in time $O(\mathsf{M}(\lambda \lg p) \lg p) = O(\mathsf{M}(b) \lg n)$ [vzGG03, Ch. 15]. This last bound lies in $O(d^2\,\mathsf{M}(n))$ (one may check the cases $\lg n \leqslant d^2$ and $\lg n \geqslant d^2$ separately).

Having selected p, λ and ζ, we now apply Lemma 3 to each matrix entry to evaluate $P_{ij}(\zeta^k) \in S$ and $Q_{ij}(\zeta^k) \in S$ for $0 \leq k < p-1$. This step takes time $O(d^2\,\mathsf{M}(\lambda\,p \lg p)) = O(d^2\,\mathsf{M}(n))$. Next we perform the pointwise multiplications $(PQ)(\zeta^k) = P(\zeta^k)\,Q(\zeta^k)$. These are achieved by first lifting to *integer* matrix products, and then reducing the results modulo p^λ. The integer products cost $O(p\ \mathsf{M}_d(\lambda\ \lg\ p)) = O((n/b)\ \mathsf{M}_d(b))$. The bit size of the entries of the products are bounded by $2\,\lambda \lg p + \lg d = O(b)$, so the reduction step costs $O(d^2\,p\,\mathsf{M}(b)) = O(d^2\,\mathsf{M}(n))$. Since the evaluation is really a discrete Fourier transform over S, the interpolation step is algebraically the same, with ζ replaced by ζ^{-1}. Thus we may recover $(PQ)_{ij}$ using Lemma 3 again, with cost $O(d^2\,\mathsf{M}(n))$. There is also a final division (scaling) by $p-1$, whose cost is subsumed into the above.

In the Turing model, we must also take into account the cost of data rearrangement. Specifically, in the above algorithm we switch between "matrix of vectors" and "vector of matrices" representations of the data. Using the algorithm in the Appendix to [BGS07], this needs only $O((d^2\ p\ \lambda\ \lg\ p)\ (\log\ n)) = O(d^2\ n\ \log\ n) = O(d^2\ \mathsf{M}(n))$ bit operations, since we assumed that $\mathsf{M}(n)/(n \log n)$ is increasing. □

Remark 5. We could replace $\mathbf{Z}/p^\lambda\,\mathbf{Z}$ by a "ring" of finite-precision approximations to complex numbers, and obtain results of the same strength. The latter approach has the disadvantage that it introduces a tedious floating-point error analysis.

Now we may prove the main theorem announced in the introduction.

Proof of Theorem 1. First consider the region $(\lg d)/C \leqslant n \leqslant d$. Proposition 4 yields

$$\begin{aligned} \mathsf{M}_d(n) &= O(d^2\,\mathsf{M}(n) + n\,\mathsf{M}_d(\lg d)/\lg d) \\ &= O(d^2\,\mathsf{M}(n) + d^\omega\, n\,\mathsf{M}(\lg d)/\lg d), \end{aligned}$$

and for such n we have $\lg^* n = \lg^* d + O(1)$, so the desired bound holds in this region.

Now consider the case $n > d$. Let $k := \min\{k \in \mathbb{N} \colon \lg^{\circ k}(n) \leqslant d\} = \lg^* n - \lg^* d + O(1)$, and let $n_j := \lg^{\circ j} n$ for $j = 0, \ldots, k$, so that $n_{k-1} > d$ and $\lg d \leqslant n_k \leqslant d$. By Proposition 4 there is a constant $K > 1$ (depending only on C) such that

$$\mathsf{M}_d(n') \leqslant K\left(d^2\,\mathsf{M}(n') + \frac{n'}{\lg n'}\,\mathsf{M}_d(\lg n')\right)$$

for any $n' > d$. Starting with $n' := n$ and iterating k times yields

$$\mathsf{M}_d(n) \leqslant K\,d^2\,n\left(\frac{\mathsf{M}(n_0)}{n_0} + \frac{K\,\mathsf{M}(n_1)}{n_1} + \cdots + \frac{K^{k-1}\,\mathsf{M}(n_{k-1})}{n_{k-1}}\right) + \frac{K^k\,n}{n_k}\,\mathsf{M}_d(n_k).$$

By the argument in the first paragraph, we may apply Proposition 4 once more (and increase K if necessary) to obtain

$$\mathsf{M}_d(n) \leqslant K\,d^2\,n\left(\frac{\mathsf{M}(n_0)}{n_0} + \frac{K\,\mathsf{M}(n_1)}{n_1} + \cdots + \frac{K^k\,\mathsf{M}(n_k)}{n_k}\right) + d^\omega\,n\,K^{k+1}\,\mathsf{M}(\lg d)/\lg d.$$

The second term lies in $O\big(d^\omega\,n\,2^{O(\lg^* n - \lg^* d)}\,\mathsf{M}(\lg d)/\lg d\big)$. Since $\mathsf{M}(n)/(n \log n)$ is increasing, the first term is bounded by

$$K\,d^2\,\mathsf{M}(n)\left(1 + \frac{K\log n_1 + \cdots + K^k \log n_k}{\log n}\right) \leqslant K\,d^2\,\mathsf{M}(n)\left(1 + \frac{k\,K^k \log\lg n}{\log n}\right) = O(d^2\,\mathsf{M}(n)). \qquad \square$$

Remark 6. An important question is to determine the best possible big-O constant in Theorem 1. For simplicity, consider the case where n is much larger than d, and define

$$A := \operatorname*{limsup}_{d\to\infty}\ \operatorname*{limsup}_{n\to\infty} \frac{\mathsf{M}_d(n)}{d^2\,\mathsf{M}(n)}.$$

Theorem 1 shows that $A < \infty$.

After some optimisations, it is possible to achieve $A = 24$. (We omit the details. The idea is to increase the chunk size b, say from $\lg n$ to $\lg^2 n$, and use the fact that Bluestein's trick actually produces a negacyclic convolution.)

We can do even better if we change the basic problem slightly. Define $\mathsf{M}'(n)$ to be the cost of an n-bit cyclic integer convolution, i.e., multiplication modulo $2^n - 1$. This kind of multiplication problem is of interest because all of the fastest known multiplication algorithms, i.e., based on FFTs, actually compute convolutions. (In this brief sketch we ignore the difficulty that such algorithms typically only work for n belonging to some sparse set.) Let $\mathsf{M}'_d(n)$ be the cost of the corresponding $d \times d$ matrix multiplication (convolution) problem, and let A' be the corresponding constant defined as above. Then by mapping the Bluestein convolution directly to integer convolution, one saves a factor of two, obtaining $A' = 12$.

We conjecture that in fact one can achieve $A = 1$. This conjecture can be proved for all integer multiplication algorithms known to the authors, and it is also consistent with measurements of the performance of the implementation described in [HS14, HLQ14]. The point is that the implementation transforms each matrix entry exactly once, and the time spent on the small-coefficient matrix multiplications is negligible if n is large.

Remark 7. It is tempting to interpret the bound in Theorem 1 as an analogue of (3) in the case of integer coefficients. However, several technical problems arise if one wants to make this more precise. Indeed, most "evaluation-interpolation" strategies for integers (apart from Chinese remaindering) involve cutting the integers into several chunks, which prevents the evaluation mappings from being algebraic homomorphisms. Moreover, due to carry management, we have to include an additional parameter for the target precision of our evaluations. Thus, in the case of matrix multiplication, we really should be looking for bounds of the form

$$\mathsf{M}_d(n) = O(d^2\,\mathsf{E}(n,p) + \mathsf{N}(n,p)\,\mathsf{M}_d(p)),$$

where p stands for the precision at our evaluation points and $p \geqslant \lceil \lg d \rceil$. In terms of $\mathsf{E}(n) = \mathsf{E}(n, q)$ and $\mathsf{N}(n) = \mathsf{N}(n, q)$ for some small fixed precision $q \leqslant p$, we have

$$\begin{aligned} \mathsf{E}(n, p) &\leqslant \mathsf{E}(n) \\ \mathsf{N}(n, p) &\sim \mathsf{N}(n/p). \end{aligned}$$

Reformulated in this way, our new evaluation-interpolation strategy achieves

$$\begin{aligned} \mathsf{E}(n) &\sim O(\mathsf{M}(n)) \\ \mathsf{N}(n) &= n\, 2^{O(\log^* n)}, \end{aligned}$$

and it can be applied to several other problems, such as the multiplication of multivariate polynomials or power series with large integer coefficients.

Acknowledgments

The authors thank Arne Storjohann for helpful conversations. The first author was supported by the Australian Research Council, DECRA Grant DE120101293.

Bibliography

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793. MR 2123939 (2006a:11170)

[BGS07] Alin Bostan, Pierrick Gaudry, and Éric Schost, *Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator*, SIAM J. Comput. **36** (2007), no. 6, 1777–1806. MR 2299425 (2008a:11156)

[BHP01] R. C. Baker, G. Harman, and J. Pintz, *The difference between consecutive primes. II*, Proc. London Math. Soc. (3) **83** (2001), no. 3, 532–562. MR 1851081 (2002f:11125)

[Blu70] L. Bluestein, *A linear filtering approach to the computation of discrete fourier transform*, Audio and Electroacoustics, IEEE Transactions on **18** (1970), no. 4, 451–455.

[BS05] Alin Bostan and Éric Schost, *Polynomial evaluation and interpolation on special sets of points*, J. Complexity **21** (2005), no. 4, 420–446. MR 2152715 (2006g:12016)

[CC90] D.V. Chudnovsky and G.V. Chudnovsky, *Computer algebra in the service of mathematical physics and number theory (computers in mathematics, stanford, ca, 1986)*, Lect. Notes in Pure and Applied Math. (New-York), vol. 125, Dekker, 1990, pp. 109–232.

[CK91] D.G. Cantor and E. Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Informatica **28** (1991), 693–701.

[CW87] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, Proc. of the 19^{th} Annual Symposium on Theory of Computing (New York City), may 25–27 1987, pp. 1–6.

[Für09] Martin Fürer, *Faster integer multiplication*, SIAM J. Comput. **39** (2009), no. 3, 979–1005. MR 2538847 (2011b:68296)

[Gal14] François Le Gall, *Powers of tensors and fast matrix multiplication*, Proc. ISSAC 2014 (Kobe, Japan), July 23–25 2014, pp. 296–303.

[Har14] David Harvey, *Counting points on hyperelliptic curves in average polynomial time*, Ann. of Math. (2) **179** (2014), no. 2, 783–803.

[HHL14a] David Harvey, Joris van der Hoeven, and Grégoire Lecerf, *Even faster integer multiplication*, preprint http://arxiv.org/abs/1407.3360, 2014.

[HHL14b] —, *Faster polynomial multiplication over finite fields*, preprint http://arxiv.org/abs/1407.3361, 2014.

[HLM+02] J. van der Hoeven, G. Lecerf, B. Mourrain, et al., *Mathemagix*, 2002, http://www.mathemagix.org.

[HLQ14] J. van der Hoeven, G. Lecerf, and G. Quintin, *Modular SIMD arithmetic in Mathemagix*, Tech. report, ArXiv, 2014, http://arxiv.org/abs/1407.3383.

[Hoe99] J. van der Hoeven, *Fast evaluation of holonomic functions*, TCS **210** (1999), 199–215.

[Hoe01] —, *Fast evaluation of holonomic functions near and in singularities*, JSC **31** (2001), 717–743.

[Hoe04] —, *The truncated Fourier transform and applications*, Proc. ISSAC 2004 (Univ. of Cantabria, Santander, Spain) (J. Gutierrez, ed.), July 4–7 2004, pp. 290–296.

[Hoe07] —, *Efficient accelero-summation of holonomic functions*, JSC **42** (2007), no. 4, 389–428.

[Hoe10] —, *Newton's method and FFT trading*, JSC **45** (2010), no. 8, 857–878.

[HP97] B. Haible and T. Papanikolaou, *Fast multiple-precision evaluation of elementary functions*, Tech. Report TI-7/97, Universität Darmstadt, 1997.

[HS14] David Harvey and Andrew V. Sutherland, *Computing Hasse–Witt matrices of hyperelliptic curves in average polynomial time*, Algorithmic Number Theory Eleventh International Symposium (ANTS XI), vol. 17, London Mathematical Society Journal of Computation and Mathematics, 2014, pp. 257–273.

[Pap94] Christos H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, Reading, MA, 1994. MR 1251285 (95f:68082)

[Shp96] Igor Shparlinski, *On finding primitive roots in finite fields*, Theoret. Comput. Sci. **157** (1996), no. 2, 273–275. MR 1389773 (97a:11203)

[SS71] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR 0292344 (45 #1431)

[Sto00] Arne Storjohann, *Algorithms for matrix canonical forms*, Ph.D. thesis, ETH Zürich, 2000, http://dx.doi.org/10.3929/ethz-a-004141007.

[Str69] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. **13** (1969), 352–356.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, second ed., Cambridge University Press, Cambridge, 2003. MR 2001757 (2004g:68202)