Integer multiplication in time O(n log n)

David Harvey, Joris Van Der Hoeven

Integer multiplication in time $O(n \log n)$

David Harvey and Joris van der Hoeven

Abstract. We present an algorithm that computes the product of two n-bit integers in $O(n \log n)$ bit operations.

1. Introduction

Let $\mathsf{M}(n)$ denote the time required to multiply two n-bit integers. We work in the multitape Turing model, in which the time complexity of an algorithm refers to the number of steps performed by a deterministic Turing machine with a fixed, finite number of linear tapes [34]. The main results of this paper also hold in the Boolean circuit model [40, Sec. 9.3], with essentially the same proofs. We write $f(n) = O(g(n))$ (respectively $f(n) = \Omega(g(n))$) to indicate that there exist constants $C > 0$ and n_0 such that $f(n) \leqslant Cg(n)$ (respectively $f(n) \geqslant Cg(n)$) for all $n \geqslant n_0$, and $f(n) = \Theta(g(n))$ to mean that both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ hold.

Schönhage and Strassen conjectured in 1971 that the true complexity of integer multiplication lies in $\Theta(n \log n)$ [39], and in the same paper established their famous upper bound $\mathsf{M}(n) = O(n \log n \log\log n)$. In 2007 their result was sharpened by Fürer to $\mathsf{M}(n) = O(n \log n\, K^{\log^* n})$ [12, 13] for some unspecified constant $K > 1$, where $\log^* n$ denotes the iterated logarithm, i.e., $\log^* x := \min\{k \geqslant 0 : \log^{\circ k} x \leqslant 1\}$. Prior to the present work, the record stood at $\mathsf{M}(n) = O(n \log n\, 4^{\log^* n})$ [22].

The main result of this paper is a verification of the upper bound in Schönhage and Strassen's conjecture, thus completely closing the remaining $4^{\log^* n}$ gap:

Theorem 1.1. *There is an integer multiplication algorithm achieving*

$$\mathsf{M}(n) = O(n \log n).$$

If the Schönhage–Strassen conjecture is correct, then Theorem 1.1 is asymptotically optimal. Unfortunately, no super-linear lower bound for $\mathsf{M}(n)$ is known. Perhaps the best available evidence in favour of the conjecture is the $\Omega(n \log n)$ lower bound [6, 35] that has been proved for the "on-line" variant of the problem, in which the k-th bit of the product must be written before the $(k+1)$-th bits of the multiplicands are read. Again, the true complexity of on-line multiplication is not known: currently, the best known upper bound is $O(n \log n \exp(C\sqrt{\log\log n}))$ for $C = \sqrt{2 \log 2} + o(1)$ [29].

Theorem 1.1 has many immediate consequences, as many computational problems may be reduced to integer multiplication. For example, the theorem implies that quotients and k-th roots of real numbers may be computed to a precision of n significant bits in time $O(n \log n)$, and that transcendental functions and constants such as e^x and π may be computed to precision n in time $O(n \log^2 n)$ [5].

Harvey was supported by the Australian Research Council (grant FT160100219).

Another interesting application is to the problem of computing DFTs (discrete Fourier transforms) over $\mathbb{C}$. Given a transform length $m \geqslant 2$ and a target accuracy of $p = \Omega(\log m)$ bits, it was pointed out in [20, 25] that one may use Bluestein's trick [2] followed by Kronecker substitution [14, Corollary 8.27] to reduce a given DFT of length m to an integer multiplication problem of size $O(mp)$. Theorem 1.1 then implies that the DFT may be evaluated in time $O(mp \log(mp))$. This compares favourably with the traditional FFT (fast Fourier transform) approach, which requires $O(m \log m)$ operations in $\mathbb{C}$, and thus time $O(m \log m\ \mathsf{M}(p)) = O(mp \log m \log p)$ in the Turing model.

All of the algorithms presented in this paper can be made completely explicit, and all implied big-O constants are in principle effectively computable. On the other hand, we make no attempt to minimise these constants or to otherwise exhibit a practical multiplication algorithm. Our aim is to establish the theoretical $O(n \log n)$ bound as directly as possible.

We will actually describe two new multiplication algorithms. The first one depends on an unproved hypothesis concerning the least prime in an arithmetic progression. This hypothesis is much weaker than standard conjectures in this area, but somewhat stronger than the best unconditional results currently available. We give only a brief sketch of this algorithm (see Section 1.2.1). A detailed treatment is given in the companion paper [24], which also presents an analogue of this algorithm for multiplication in $\mathbb{F}_q[x]$. The bulk of the present paper (Sections 2–5) concentrates on working out the details of the second algorithm, which is technically more involved, but has the virtue of reaching the $O(n \log n)$ bound unconditionally.

In the remainder of Section 1, we review the literature on integer multiplication (Section 1.1), and give an overview of the new algorithms (Section 1.2).

1.1. **Survey of integer multiplication algorithms.** The first improvement on the classical $\mathsf{M}(n) = O(n^2)$ bound was found by Karatsuba in 1962. Significant progress was made during the 1960s by Toom, Cook, Schönhage and Knuth; see [25, Sec. 1.1] for further historical details and references for this period. FFTs were brought into the picture by Schönhage and Strassen [39] soon after the publication of the FFT by Cooley and Tukey [7]; see [28] for more on the history of the FFT. The multiplication algorithms published since [39] may be roughly classified into four families:

(1) *Schönhage–Strassen's first algorithm* [39] is, in hindsight, the most straightforward FFT-based integer multiplication algorithm imaginable. By splitting the n-bit multiplicands into chunks of size $\Theta(\log n)$, they reduce to the problem of multiplying polynomials in $\mathbb{Z}[x]$ of degree $\Theta(n/\log n)$ and coefficient size $\Theta(\log n)$. The product in $\mathbb{Z}[x]$ is handled by means of FFTs over $\mathbb{C}$, i.e., evaluating the polynomials at suitable roots of unity, multiplying their values pointwise in $\mathbb{C}$, and then interpolating to obtain the product polynomial. Elements of $\mathbb{C}$ are represented approximately, with a precision of $\Theta(\log n)$ bits. Arithmetic operations in $\mathbb{C}$ (such as multiplication) are reduced to arithmetic in $\mathbb{Z}$ by scaling by a suitable power of two. This leads to the recursive estimate

$$\mathsf{M}(n) = O(n\,\mathsf{M}(n')) + O(n \log n), \qquad n' = O(\log n),$$

whose explicit solution is

$$\mathsf{M}(n) = O(K^{\log^* n}\, n \log n \log\log n \cdots \log^{\circ((\log^* n)-1)} n)$$

for some constant $K > 0$. The algorithm achieves an exponential size reduction at each recursion level, from n to $O(\log n)$, and the number of levels is $\log^* n + O(1)$.

Pollard suggested a similar algorithm at around the same time [36], working over a finite field rather than $\mathbb{C}$. He did not analyse the bit complexity, but with some care one can prove essentially the same complexity bound as for the complex case (some technical difficulties arise due to the cost of finding suitable primes; these may be resolved by techniques similar to those discussed in [25, Sec. 8.2]).

(2) *Schönhage–Strassen's second algorithm* is the more famous and arguably the more ingenious of the two algorithms presented in [39]. It is probably the most widely used large-integer multiplication algorithm in the world today, due to the highly optimised implementation included in the free GNU Multiple Precision Arithmetic Library (GMP) [17, 15], which underlies the large-integer capabilities of all of the major contemporary computer algebra systems.

The basic recursive problem is taken to be multiplication in $\mathbb{Z}/(2^n+1)\mathbb{Z}$, where n is a power of two. Let $n' \coloneqq 2^{\lceil (\log_2 2n)/2 \rceil} = \Theta(n^{1/2})$ and $T \coloneqq 2n/n' = \Theta(n^{1/2})$, so that $(n')^2 \in \{2n, 4n\}$ and $T \mid n'$; then by splitting the inputs into chunks of size $n'/2$, the problem is reduced to multiplication in $R[x]/(x^T+1)$ where $R \coloneqq \mathbb{Z}/(2^{n'}+1)\mathbb{Z}$. The powers of 2 in R are sometimes called "synthetic" roots of unity, as they have been synthesised algebraically, or "fast" roots of unity, as one can multiply an element of R by an arbitrary power of 2 in linear time, i.e., in time $O(n')$. Consequently, for $\omega \coloneqq 2^{n'/T}$, one may evaluate a polynomial at $\omega, \omega^3, \ldots, \omega^{2T-1}$ (the roots of x^T+1) via the FFT in time $O((n' \log n')n') = O(n \log n)$. The original multiplication problem is thus reduced to T pointwise multiplications in R, which are handled recursively. Writing $\mathsf{M}_1(n)$ for the cost of a product in $\mathbb{Z}/(2^n+1)\mathbb{Z}$, one obtains the recurrence

$$\mathsf{M}_1(n) < \frac{2n}{n'}\,\mathsf{M}_1(n') + O(n \log n), \qquad n' = O(n^{1/2}). \tag{1.1}$$

Unlike the first Schönhage–Strassen algorithm, this algorithm performs only a *geometric* size reduction, from n to $O(n^{1/2})$, at each recursion level, and the number of recursion levels is $\log_2 \log n + O(1) = O(\log \log n)$.

The constant 2 in (1.1), which arises from zero-padding in the initial splitting stage, plays a crucial role in the complexity analysis: it ensures that at each recursion level, the total cost of the "fast" FFTs remains $O(n \log n)$, with the same implied constant at each level. The overall cost is thus $\mathsf{M}_1(n) = O(n \log n \log \log n)$.

(3) *Fürer's algorithm* [12, 13] combines the best features of the two Schönhage–Strassen algorithms: the exponential size reduction from the first algorithm, and the fast roots of unity from the second one. The overall strategy is similar to the first algorithm, but instead of working over $\mathbb{C}$, one uses a bivariate splitting to reduce to a polynomial multiplication problem over $R \coloneqq \mathbb{C}[y]/(y^r+1)$, where $r = \Theta(\log n)$ is a power of two. This ring contains a synthetic root of unity y of order $2r$, but also inherits higher-order roots of unity from $\mathbb{C}$. Elements of $\mathbb{C}$ are represented approximately, with a precision of $O(\log n)$ bits; thus an element of R occupies $O((\log n)^2)$ bits.

Fürer's key insight is to apply the Cooley–Tukey FFT decomposition in radix $2r$ instead of radix two. He decomposes each "long" transform of length $\Theta(n/(\log n)^2)$ into many "short" transforms of length $2r$, with one round of expensive "twiddle factor" multiplications interposed between each layer of short transforms. The short

transforms take advantage of the synthetic roots of unity, and the twiddle factor multiplications are handled recursively (via Kronecker substitution). This leads to the recurrence

$$\mathsf{M}(n) = O\left(\frac{n \log n}{n' \log n'}\, \mathsf{M}(n')\right) + O(n \log n), \qquad n' = O((\log n)^2),$$

and then to the explicit bound $\mathsf{M}(n) = O(n \log n\, K^{\log^* n})$ for some constant $K > 1$. Fürer did not give a specific value for K, but it is argued in [25, Sec. 7] that careful optimisation of his algorithm leads to the value $K = 16$.

Several authors have given variants of Fürer's algorithm that also achieve $\mathsf{M}(n) = O(n \log n\, K^{\log^* n})$, using essentially the same idea but working over different rings. De, Kurur, Saha and Saptharishi [10] replace $\mathbb{C}$ by a p-adic ring $\mathbb{Q}_p$; this has the benefit of avoiding numerical analysis over $\mathbb{C}$, but the value of K becomes somewhat larger. Covanov and Thomé give another variant that achieves $K = 4$, conditional on a conjecture on the distribution of generalised Fermat primes [8].

(4) *The Harvey–van der Hoeven–Lecerf algorithm* [25] follows Fürer in decomposing a "long" transform into many "short" transforms of exponentially smaller length. However, instead of working over a ring containing fast roots of unity, one works directly over $\mathbb{C}$ (as in the first Schönhage–Strassen algorithm), and converts the short transforms back to multiplication problems via Bluestein's trick [2]. These short products are then handled recursively.

The first version given in [25] achieved $\mathsf{M}(n) = O(n \log n\, K^{\log^* n})$ with $K = 8$. The value of K was improved gradually over a sequence of papers [18, 19, 21], reaching $K = 4$ in [22]. All of these algorithms perform exponential size reduction, and the number of recursion levels is $\log^* n + O(1)$.

An interesting feature of these algorithms — related to the fact that they dispense with the need for fast roots of unity — is that they can be adapted to prove bounds of the form $O(n \log n\, K^{\log^* n})$ for the cost of multiplying polynomials in $\mathbb{F}_q[x]$ of degree n (for fixed q). This was first established with the constant $K = 8$ in [26], and improved to $K = 4$ in [23]. As mentioned previously, the first of the two new algorithms presented in this paper may be adapted to obtain an $O(n \log n)$ bound for the $\mathbb{F}_q[x]$ case [24], but unfortunately this result is still conditional and so does not yet supersede the unconditional $O(n \log n\, 4^{\log^* n})$ bound given in [23].

1.2. **Overview of new algorithms.** Our new algorithms are motivated by the observation that certain *multivariate* polynomial rings admit particularly efficient multiplication algorithms. Let r be a power of two, and for $d \geqslant 2$ consider the ring

$$R[x_1, \ldots, x_{d-1}]/(x_1^{t_1} - 1, \ldots, x_{d-1}^{t_{d-1}} - 1), \qquad R \coloneqq \mathbb{C}[y]/(y^r + 1), \tag{1.2}$$

where $t_i \mid 2r$ for all i. One may multiply in this ring by first using FFTs to evaluate each x_i at the synthetic t_i-th roots of unity (the powers of y^{2r/t_i}), then multiplying pointwise in R, and finally performing inverse FFTs. Such transforms were studied extensively by Nussbaumer in the late 1970s (see for example [31]), and are sometimes known as *fast polynomial transforms*. They consist entirely of additions and subtractions in $\mathbb{C}$, and require no multiplications in $\mathbb{C}$ whatsoever.

In Sections 1.2.1 and 1.2.2 below, we outline two different ways of fashioning an integer multiplication algorithm from the polynomial multiplication algorithm just described. The key issue is to show how to transport an integer multiplication problem, which is intrinsically one-dimensional, to a ring of the type (1.2).

In both cases, we begin with the following setup. Suppose that we wish to multiply two n-bit integers. We choose a dimension parameter $d \geqslant 2$ and distinct primes $s_1, \ldots, s_d \approx (n/\log n)^{1/d}$, subject to certain conditions that will be explained in Sections 1.2.1 and 1.2.2. Just as in the first Schönhage–Strassen algorithm, we split the inputs into around $n/\log n$ chunks of roughly $\log n$ bits, thereby reducing the problem to multiplication in $\mathbb{Z}[x]/(x^{s_1 \cdots s_d} - 1)$. Now, following a technique described by Agarwal and Cooley [1] (which is closely related to the Good–Thomas FFT algorithm [16, 41]), we observe that the Chinese remainder theorem induces an isomorphism $\mathbb{Z}[x]/(x^{s_1 \cdots s_d} - 1) \cong \mathbb{Z}[x_1, \ldots, x_d]/(x_1^{s_1} - 1, \ldots, x_d^{s_d} - 1)$, so the problem amounts to computing a product in the latter ring. For this, it suffices to show how to efficiently compute a multidimensional complex DFT of size $s_1 \times \cdots \times s_d$, i.e., with respect to the complex s_i-th roots of unity, to an accuracy of $O(\log n)$ bits.

1.2.1. *A conditional algorithm — Rader's trick.* Suppose that we are able to choose the primes $s_1, \ldots, s_d$ so that $s_i = 1 \pmod r$, where r is a power of two, and where the s_i are not much larger than r. We may then deploy a multidimensional generalisation of *Rader's algorithm* [37] to reduce the given DFT of size $s_1 \times \cdots \times s_d$ to a multiplication problem in the ring $\mathbb{C}[x_1, \ldots, x_d]/(x_1^{s_1 - 1} - 1, \ldots, x_d^{s_d - 1} - 1)$ (together with some lower-dimensional multiplication problems of negligible cost). Crucially, the convolution lengths have been reduced from s_i to $s_i - 1$. Writing $s_i - 1 = q_i r$, where the q_i are "small", we may further reduce this product to a collection of complex DFTs of size $q_1 \times \cdots \times q_d$, plus a collection of multiplication problems in $\mathbb{C}[x_1, \ldots, x_d]/(x_1^r - 1, \ldots, x_d^r - 1)$. After replacing x_d with $e^{\pi \mathrm{i}/r} y$, we see that the latter products are exactly of the type (1.2). As discussed previously, we may use synthetic FFTs to reduce such a product to a collection of pointwise products in $R = \mathbb{C}[y]/(y^r + 1)$. These in turn are converted to integer multiplication problems via Kronecker substitution, and then handled recursively.

The main sticking point in the above algorithm is the cost of the auxiliary DFTs of size $q_1 \times \cdots \times q_d$. There are various options available for evaluating these DFTs, but to ensure that this step does not dominate the complexity, the key issue is to keep the size of the q_i under control. What we are able to prove is the following. For positive, relatively prime integers m and a, define

$$P(a, m) \coloneqq \min\{q > 0 : q \text{ prime and } q = a \bmod m\},$$

and put $P(m) \coloneqq \max_a P(a, m)$. *Linnik's theorem* states that there is an absolute constant $L > 1$ such that $P(m) = O(m^L)$. The best published value for L is currently $L = 5.18$ [42], and under the Generalised Riemann Hypothesis one may take $L = 2 + \varepsilon$ for any $\varepsilon > 0$ [27]. In the companion paper [24], we prove that if Linnik's theorem holds for some $L < 1 + \frac{1}{303}$, and if we take d near 10^6, then the cost of the auxiliary DFTs can be controlled and one does in fact obtain an overall $\mathsf{M}(n) = O(n \log n)$ bound. (Actually, in [24] we work over a finite field, but the same method should work over $\mathbb{C}$, possibly with a different threshold for L.)

On the other hand, it is widely expected that the bound $P(m) = O(m^L)$ should hold for *any* $L > 1$. For this reason, we strongly suspect that the algorithm sketched above does run in time $O(n \log n)$, despite us being unable to supply a proof. For further discussion, and examples of even stronger bounds for $P(m)$ that are expected to hold, see [24].

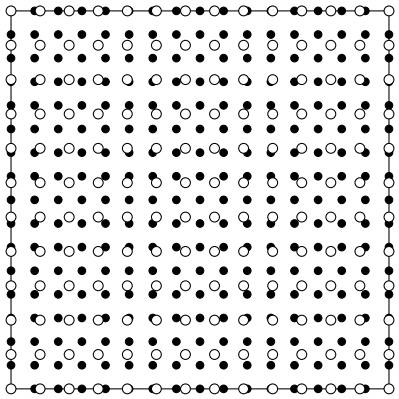

FIGURE 1. Torus $(\mathbb{R}/\mathbb{Z})^2$ with 13×11 source array (white circles) superimposed over 16×16 target array (black circles)

Remark 1.2. The idea of evaluating a multidimensional transform via a combination of Rader's algorithm and polynomial transforms was previously suggested in a different context by Nussbaumer and Quandalle [32, p. 141].

1.2.2. *An unconditional algorithm — Gaussian resampling.* The rest of the paper is devoted to the second method. Here we choose the primes $s_1, \ldots, s_d$ in such a way that each s_i is slightly smaller than a power of two t_i, and so that $t_1 \cdots t_d = O(s_1 \cdots s_d)$. Finding such primes is easily accomplished using the prime number theorem with a suitable error term (see Lemma 5.1).

Assume as before that we wish to compute a complex multidimensional DFT of size $s_1 \times \cdots \times s_d$, to an accuracy of $O(\log n)$ bits. Our key innovation is to show that *this problem may be reduced directly to the problem of computing a complex multidimensional DFT of size* $t_1 \times \cdots \times t_d$.

The idea of the reduction is as follows. Suppose that we are given as input an $s_1 \times \cdots \times s_d$ array of complex numbers $u = (u_{j_1,\ldots,j_d})_{0 \leqslant j_i < s_i}$. We may regard this array as lying inside the d-dimensional unit torus $(\mathbb{R}/\mathbb{Z})^d$: we imagine the coefficient $u_{j_1,\ldots,j_d}$ to be plotted at coordinates $(j_1/s_1, \ldots, j_d/s_d)$ in the torus (see Figure 1). We construct from u an intermediate $t_1 \times \cdots \times t_d$ array $v = (v_{k_1,\ldots,k_d})_{0 \leqslant k_i < t_i}$. Again, we think of $v_{k_1,\ldots,k_d}$ as being plotted at coordinates $(k_1/t_1, \ldots, k_d/t_d)$ in the torus. The coefficients of v are defined to be certain linear combinations of the coefficients of u. The weights are essentially d-dimensional Gaussians, so each coefficient of v depends mainly on the "nearby" coefficients of u within the torus.

This construction has two crucial properties. First, the rapid decay of the Gaussians allows us to compute (i.e., approximate) the coefficients of v very quickly from those of u; indeed, the cost of this step is asymptotically negligible compared to the cost of the DFTs themselves. Second, using the fact that the Fourier transform of a Gaussian is a Gaussian, we will show that $\hat{u}$ and $\hat{v}$ (the DFTs of u and v) are related by a fairly simple system of linear equations. In fact, the matrix of this system is of the same type as the matrix relating u and v. The system is somewhat overdetermined, because $t_1 \cdots t_d > s_1 \cdots s_d$. Provided that the ratios t_i/s_i are not too close to 1, we will show that this system may be solved in an efficient and numerically stable manner, and that we may therefore recover $\hat{u}$ from $\hat{v}$. This procedure forms the core of our "Gaussian resampling" method, and is developed in detail in

Section 4. It is closely related to the Dutt–Rokhlin algorithm for non-equispaced FFTs [11]; see Section 4.4.3 for a discussion of the similarities and differences.

We have therefore reduced to the problem of computing $\hat{v}$ from v, and we are free to do this by any convenient method. Note that this is a DFT of size $t_1 \times \cdots \times t_d$ rather than $s_1 \times \cdots \times s_d$. In Section 3 we will show how to use a multivariate generalisation of Bluestein's algorithm [2] to reduce this DFT to a multiplication problem in a ring of the form (1.2). As already pointed out, such a product may be handled efficiently via synthetic FFTs; the details of this step are also discussed in Section 3.

Analysis of this algorithm leads to a recurrence inequality of the form

$$\mathsf{M}(n) < \frac{Kn}{n'}\,\mathsf{M}(n') + O(n \log n), \qquad n' = n^{\frac{1}{d}+o(1)}, \tag{1.3}$$

where both K and the big-O constant are absolute, and in particular, do not depend on d. (In Section 5 we establish (1.3) with the explicit constant $K = 1728$, and in Section 5.4 we list some optimisations that improve it to $K = 8$.) The first term arises from pointwise multiplications in a ring of the type $R = \mathbb{C}[y]/(y^r + 1)$, and the second term from the fast FFTs and other auxiliary operations, including computing v from u and recovering $\hat{u}$ from $\hat{v}$.

We stress here the similarity with the corresponding bound (1.1) for the second Schönhage–Strassen algorithm; the difference is that we are now free to choose d. In Section 5, we will simply take $d \coloneqq 1729$ (any constant larger than K would do), and then it is easy to see that (1.3) implies that $\mathsf{M}(n) = O(n \log n)$. (A similar analysis holds for the conditional algorithm sketched in Section 1.2.1, for different values of K and d.)

It is striking that for fixed d, the new algorithm performs only a geometric size reduction at each recursion level, just like the second Schönhage–Strassen algorithm, and unlike the first Schönhage–Strassen algorithm or any of the post-Fürer algorithms. In the new algorithm, the total cost of the FFTs actually *decreases* by the constant factor $d/K > 1$ at each subsequent recursion level, unlike in the second Schönhage–Strassen algorithm, where it remains constant at each level, or any of the other algorithms mentioned, where it increases by a constant factor at each level.

Actually, with some care it is possible to allow d to grow with n, so as to achieve size reduction faster than geometric, and still reach the desired $O(n \log n)$ bound, but we will not carry out this analysis.

Finally, we mention that our reduction from a DFT of size $s_1 \times \cdots \times s_d$ to one of size $t_1 \times \cdots \times t_d$ is highly non-algebraic, and depends heavily on the archimedean property of $\mathbb{R}$. Consequently, we do not know how to give an analogue of this algorithm for multiplication in $\mathbb{F}_q[x]$.

2. DFTs, convolutions and fixed-point arithmetic

In the Turing model we cannot compute with elements of $\mathbb{C}$ exactly. In this section we introduce a framework for systematic discussion of DFTs and convolutions in the setting of fixed-point arithmetic. (This framework is loosely based on the presentation in [25, Sec. 3].)

2.1. **Integer arithmetic.** Integers are assumed to be stored in the standard binary representation. We briefly recall several well-known results concerning integer arithmetic; see [5, Ch. 1] for further details and literature references.

Let $p \geqslant 1$, and assume that we are given as input $x, y \in \mathbb{Z}$ such that $|x|, |y| \leqslant 2^p$. We may compute $x + y$ and $x - y$ in time $O(p)$. For multiplication, we will often use the crude estimate $\mathsf{M}(p) = O(p^{1+\delta})$, where for the rest of the paper δ denotes a small, fixed positive quantity; for definiteness, we assume that $\delta < \frac{1}{8}$. If $y > 0$, then we may compute the quotients $\lfloor x/y \rfloor$ and $\lceil x/y \rceil$ in time $O(p^{1+\delta})$. More generally, for a fixed positive rational number a/b, and assuming $x, y > 0$, we may compute $\lfloor (x/y)^{a/b} \rfloor$ and $\lceil (x/y)^{a/b} \rceil$ in time $O(p^{1+\delta})$.

2.2. **Fixed-point coordinate vectors.** Fix a precision parameter $p \geqslant 100$. Let $\mathbb{C}_\circ \coloneqq \{u \in \mathbb{C} : |u| \leqslant 1\}$ denote the complex unit disc, and set

$$\tilde{\mathbb{C}}_\circ \coloneqq (2^{-p}\mathbb{Z}[\mathrm{i}]) \cap \mathbb{C}_\circ = \{2^{-p}(x + \mathrm{i}y) : x, y \in \mathbb{Z} \text{ and } x^2 + y^2 \leqslant 2^{2p}\}.$$

In the Turing model, we represent an element $z = 2^{-p}(x + \mathrm{i}y) \in \tilde{\mathbb{C}}_\circ$ by the pair of integers (x, y). It occupies $O(p)$ bits of storage, as $|x|, |y| \leqslant 2^p$. The precision p is always known from context and does not need to be stored alongside z.

We define a round-towards-zero function $\rho\colon \mathbb{C} \to \mathbb{C}$ as follows. First, define $\rho_0\colon \mathbb{R} \to \mathbb{Z}$ by $\rho_0(x) \coloneqq \lfloor x \rfloor$ for $x \geqslant 0$, and $\rho_0(x) \coloneqq \lceil x \rceil$ for $x < 0$. Then define $\rho_0\colon \mathbb{C} \to \mathbb{Z}[\mathrm{i}]$ by setting $\rho_0(x + \mathrm{i}y) \coloneqq \rho_0(x) + \mathrm{i}\rho_0(y)$ for $x, y \in \mathbb{R}$. Observe that $|\rho_0(u)| \leqslant |u|$ and $|\rho_0(u) - u| < \sqrt{2}$ for any $u \in \mathbb{C}$. Finally, set

$$\rho(u) \coloneqq 2^{-p}\rho_0(2^p u), \qquad u \in \mathbb{C}.$$

Thus $|\rho(u)| \leqslant |u|$ and $|\rho(u) - u| < \sqrt{2} \cdot 2^{-p}$ for any $u \in \mathbb{C}$. Clearly $\rho(\mathbb{C}_\circ) \subset \tilde{\mathbb{C}}_\circ$.

Now let V be a finite-dimensional vector space over $\mathbb{C}$. In this paper, every such V is understood to come equipped with a privileged choice of ordered basis $B_V = \{b_0, \ldots, b_{m-1}\}$, where $m = \dim_{\mathbb{C}} V$. For the special case $V = \mathbb{C}^m$, we always take the standard basis; in particular, for $V = \mathbb{C}$ the basis is simply $\{1\}$.

We define a norm $\|\cdot\| : V \to [0, \infty)$ in terms of the basis B_V by setting

$$\|\lambda_0 b_0 + \cdots + \lambda_{m-1} b_{m-1}\| \coloneqq \max_j |\lambda_j|, \qquad \lambda_j \in \mathbb{C}.$$

This norm satisfies $\|u + v\| \leqslant \|u\| + \|v\|$ and $\|\lambda u\| = |\lambda| \, \|u\|$ for any $u, v \in V$, $\lambda \in \mathbb{C}$. The unit ball in V is defined to be

$$V_\circ \coloneqq \{u \in V : \|u\| \leqslant 1\} = \{\lambda_0 b_0 + \cdots + \lambda_{m-1} b_{m-1} : \lambda_j \in \mathbb{C}_\circ\},$$

and we also define

$$\tilde{V}_\circ \coloneqq \{\lambda_0 b_0 + \cdots + \lambda_{m-1} b_{m-1} : \lambda_j \in \tilde{\mathbb{C}}_\circ\}.$$

We extend ρ to a function $\rho\colon V \to V$ by acting componentwise, i.e., we put

$$\rho(\lambda_0 b_0 + \cdots + \lambda_{m-1} b_{m-1}) \coloneqq \sum_j \rho(\lambda_j) b_j, \qquad \lambda_j \in \mathbb{C}.$$

Then $\|\rho(u)\| \leqslant \|u\|$ and $\|\rho(u) - u\| < \sqrt{2} \cdot 2^{-p}$ for any $u \in V$. Clearly $\rho(V_\circ) \subset \tilde{V}_\circ$.

In the special case $V = \mathbb{C}$ we have simply $\|u\| = |u|$ for any $u \in \mathbb{C}$, and the notations $\mathbb{C}_\circ$, $\tilde{\mathbb{C}}_\circ$ and $\rho\colon \mathbb{C} \to \mathbb{C}$ all agree with their previous definitions.

In the Turing model, an element $u \in \tilde{V}_\circ$ is represented by its coordinate vector with respect to B_V, i.e., as a list of m elements of $\tilde{\mathbb{C}}_\circ$, so u occupies $O(mp)$ bits of storage.

For $u \in V_\circ$, we systematically use the notation $\tilde{u} \in \tilde{V}_\circ$ to indicate a fixed-point approximation for u that has been computed by some algorithm. We write

$$\varepsilon(\tilde{u}) \coloneqq 2^p \left\| \tilde{u} - u \right\|$$

for the associated error, measured as a multiple of 2^{-p} (the "unit in the last place"). For example, we have the following result for addition and subtraction in V.

Lemma 2.1 (Addition/subtraction)**.** *Given as input $u, v \in \tilde{V}_\circ$, in time $O(mp)$ we may compute an approximation $\tilde{w} \in \tilde{V}_\circ$ for $w \coloneqq \frac{1}{2}(u \pm v) \in V_\circ$ such that $\varepsilon(\tilde{w}) < 1$.*

Proof. Consider first the case $m = 1$, i.e., assume that $V = \mathbb{C}$. Let $u = 2^{-p}a$ and $v = 2^{-p}b$ where $a, b \in \mathbb{Z}[\mathrm{i}]$ and $|a|, |b| \leqslant 2^p$. Since the denominators of the real and imaginary parts of $2^p w = \frac{1}{2}(a \pm b)$ are at most 2, we have $|\rho_0(2^p w) - 2^p w| \leqslant ((\frac{1}{2})^2 + (\frac{1}{2})^2)^{1/2} = \frac{1}{\sqrt{2}}$. Define $\tilde{w} \coloneqq \rho(w) = 2^{-p}\rho_0(2^p w)$. We may clearly compute $\tilde{w}$ in time $O(p)$, and $\varepsilon(\tilde{w}) = 2^p \left\| \rho(w) - w \right\| \leqslant \frac{1}{\sqrt{2}} < 1$. The general case ($m \geqslant 1$) follows by applying the same argument in each coordinate. □

Occasionally we will encounter a situation in which we have computed an approximation $\tilde{u} \in \tilde{V}_\circ$ for some $u \in V$, and we wish to compute an approximation for cu, where $c \geqslant 1$ is a fixed integer scaling factor for which it is known that $cu \in V_\circ$. A typical example is the final scaling step in an inverse FFT. Unfortunately, the obvious approximation $c\tilde{u}$ might lie just outside $V_\circ$. We deal with this minor technical nuisance as follows.

Lemma 2.2 (Scaling)**.** *Let $u \in V$ and let c be an integer such that $1 \leqslant c \leqslant 2^p$. Assume that $\|u\| \leqslant c^{-1}$, and let $v \coloneqq cu \in V_\circ$. Given as input c and an approximation $\tilde{u} \in \tilde{V}_\circ$, in time $O(mp^{1+\delta})$ we may compute an approximation $\tilde{v} \in \tilde{V}_\circ$ such that $\varepsilon(\tilde{v}) < 2c \cdot \varepsilon(\tilde{u}) + 3$.*

Proof. Again it suffices to handle the case $m = 1$, $V = \mathbb{C}$.

We first compute $2^{-p}(x + \mathrm{i}y) \coloneqq c\tilde{u}$ in time $O(p^{1+\delta})$. Note that $c\tilde{u}$ might not lie in $\tilde{\mathbb{C}}_\circ$, but x and y are certainly integers with $O(p)$ bits.

Next we compute $a \coloneqq x^2 + y^2$ in time $O(p^{1+\delta})$, so that $a^{1/2} = 2^p \left| c\tilde{u} \right|$. If $a \leqslant 2^{2p}$ then already $c\tilde{u} \in \tilde{\mathbb{C}}_\circ$, so we may simply take $\tilde{v} \coloneqq c\tilde{u}$, and then $\varepsilon(\tilde{v}) = 2^p \left| \tilde{v} - v \right| = 2^p \left| c\tilde{u} - cu \right| = c \cdot \varepsilon(\tilde{u}) < 2c \cdot \varepsilon(\tilde{u}) + 3$.

Suppose instead that $a > 2^{2p}$ (i.e., $c\tilde{u} \notin \tilde{\mathbb{C}}_\circ$). We then compute $b \coloneqq \lceil a^{1/2} \rceil > 2^p$, again in time $O(p^{1+\delta})$. Let $z \coloneqq 2^p c\tilde{u}/b = (x + \mathrm{i}y)/b$ and $\tilde{v} \coloneqq \rho(z)$. Note that $\tilde{v} = 2^{-p}(x' + \mathrm{i}y')$ where $x' = \rho_0(2^p x/b)$ and $y' = \rho_0(2^p y/b)$, so we may compute $\tilde{v}$ in time $O(p^{1+\delta})$. We have $|\tilde{v}| \leqslant |z| = 2^p \left| c\tilde{u} \right| / b \leqslant 2^p \left| c\tilde{u} \right| / a^{1/2} = 1$, so indeed $\tilde{v} \in \tilde{\mathbb{C}}_\circ$. Moreover,

$$|\tilde{v} - v| \leqslant |\tilde{v} - z| + |z - c\tilde{u}| + |c\tilde{u} - v| = |\rho(z) - z| + |z| \left| 1 - \tfrac{b}{2^p} \right| + c \left| \tilde{u} - u \right|,$$

so $\varepsilon(\tilde{v}) < \sqrt{2} + |2^p - b| + c \cdot \varepsilon(\tilde{u})$. We also have $2^p < b < a^{1/2} + 1 = 2^p \left| c\tilde{u} \right| + 1$, so

$$0 < b - 2^p < 2^p \left| c\tilde{u} \right| - 2^p + 1 \leqslant 2^p \left| cu \right| - 2^p + 2^p \left| c\tilde{u} - cu \right| + 1 \leqslant c \cdot \varepsilon(\tilde{u}) + 1.$$

We conclude that $|2^p - b| < c \cdot \varepsilon(\tilde{u}) + 1$, and therefore $\varepsilon(\tilde{v}) < 2c \cdot \varepsilon(\tilde{u}) + (1 + \sqrt{2})$. □

2.3. **Coefficient rings.** By a *coefficient ring* we mean a finite-dimensional commutative $\mathbb{C}$-algebra R with identity (together with a privileged basis B_R). We are chiefly interested in the following two examples:

(1) *Complex case:* $\mathbb{C}$ itself, with the basis $\{1\}$.
(2) *Synthetic case:* for any $r \geqslant 1$, the ring $\mathscr{R} \coloneqq \mathbb{C}[y]/(y^r + 1)$, with the basis $\{1, y, \ldots, y^{r-1}\}$.

Let R be a coefficient ring of dimension r with basis B_R, and let $n \geqslant 1$. Then R^n is a vector space of dimension nr over $\mathbb{C}$. We associate to R^n the "nested" basis formed by concatenating n copies of B_R. In particular, we have $\|u\| = \max_j \|u_j\|$ for $u = (u_0, \ldots, u_{n-1}) \in R^n$. In place of the awkward expressions $(R^n)_\circ$ and $(\tilde{R}^n)_\circ$, we write more compactly $R^n_\circ$ and $\tilde{R}^n_\circ$. In the Turing model, an element of $\tilde{R}^n_\circ$ occupies $O(nrp)$ bits of storage.

Now let $d \geqslant 1$ and $n_1, \ldots, n_d \geqslant 1$. We write $\otimes_{i=1}^d R^{n_i}$, or just $\otimes_i R^{n_i}$ when d is understood, for the tensor product $R^{n_1} \otimes_R \cdots \otimes_R R^{n_d}$. It is a free R-module of rank $n_1 \cdots n_d$, and also a vector space over $\mathbb{C}$ of dimension $n_1 \cdots n_d r$. An element $u \in \otimes_i R^{n_i}$ may be regarded as a d-dimensional array of elements of R of size $n_1 \times \cdots \times n_d$. For indices $j_1, \ldots, j_d$ where $0 \leqslant j_i < n_i$, we write $u_{j_1,\ldots,j_d} \in R$ for the $(j_1, \ldots, j_d)$-th component of u.

We associate to $\otimes_i R^{n_i}$ the nested basis consisting of $n_1 \cdots n_d$ copies of B_R arranged in lexicographical order, i.e., listing the coordinates of u in the order $(u_{0,\ldots,0}, u_{0,\ldots,1}, \ldots, u_{n_1-1,\ldots,n_d-1})$. Observe then that $\|u\| = \max_{j_1,\ldots,j_d} \|u_{j_1,\ldots,j_d}\|$. Instead of $(\otimes_i R^{n_i})_\circ$ and $(\otimes_i \tilde{R}^{n_i})_\circ$, we write $\otimes_i R^{n_i}_\circ$ and $\otimes_i \tilde{R}^{n_i}_\circ$. In the Turing model, an element of $\otimes_i \tilde{R}^{n_i}_\circ$ occupies $O(n_1 \cdots n_d rp)$ bits of storage.

Let $u \in \otimes_i R^{n_i}$. By an *i-slice* of u we mean a one-dimensional sub-array of u, consisting of the entries $u_{j_1,\ldots,j_d}$ where $j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_d$ are held fixed and j_i varies over $\{0, \ldots, n_i - 1\}$. We will occasionally wish to apply a given algorithm separately to each of the $n_1 \cdots n_{i-1} n_{i+1} \cdots n_d$ distinct i-slices of some $u \in \otimes_i \tilde{R}^{n_i}_\circ$. To accomplish this in the Turing model, we must first rearrange the data so that each i-slice is stored contiguously. In the lexicographical order specified above, this amounts to performing $n_1 \cdots n_{i-1}$ matrix transpositions of size $n_i \times (n_{i+1} \cdots n_d)$. This data rearrangement may be performed in time $O(n_1 \cdots n_d rp \log n_i)$ using a fast matrix transposition algorithm [4, Appendix].

2.4. **DFTs and convolutions.** Let R be a coefficient ring and let $n \geqslant 1$. Throughout the paper we adopt the convention that for a vector $u = (u_0, \ldots, u_{n-1}) \in R^n$ and an integer j, the expression u_j always means $u_{j \bmod n}$. For $u, v \in R^n$, we define the pointwise product $u \cdot v \in R^n$ and the convolution product $u * v \in R^n$ by

$$(u \cdot v)_j \coloneqq u_j v_j, \qquad (u * v)_j \coloneqq \sum_{k=0}^{n-1} u_k v_{j-k}, \qquad 0 \leqslant j < n.$$

Then $(R^n, \cdot)$ and $(R^n, *)$ are both commutative rings, isomorphic respectively to the direct sum of n copies of R, and the polynomial ring $R[x]/(x^n - 1)$.

A *principal n-th root of unity in R* is an element $\omega \in R$ such that $\omega^n = 1$, $\sum_{k=0}^{n-1} (\omega^j)^k = 0$ for every integer $j \neq 0 \pmod n$, and $\|\omega u\| = \|u\|$ for all $u \in R$. We define an associated R-linear DFT map $F_\omega \colon R^n \to R^n$ by the formula

$$(F_\omega u)_j \coloneqq \frac{1}{n} \sum_{k=0}^{n-1} \omega^{-jk} u_k, \qquad u \in R^n, \quad 0 \leqslant j < n.$$

It is immediate that ω^{-1} $(= \omega^{n-1})$ is also a principal n-th root of unity in R, and that $\|F_\omega u\| \leqslant \|u\|$ for all $u \in R^n$.

Lemma 2.3 (Convolution formula)**.** *For any $u, v \in R^n$ we have*

$$\frac{1}{n} u * v = n F_{\omega^{-1}}(F_\omega u \cdot F_\omega v).$$

Proof. For each j, the product $(F_\omega u)_j (F_\omega v)_j$ is equal to

$$\frac{1}{n^2} \sum_{s=0}^{n-1} \sum_{t=0}^{n-1} \omega^{-j(s+t)} u_s v_t = \frac{1}{n^2} \sum_{k=0}^{n-1} \omega^{-jk} \sum_{s+t=k \pmod n} u_s v_t = \frac{1}{n} F_\omega(u * v)_j,$$

so $F_\omega(u * v) = n F_\omega u \cdot F_\omega v$. On the other hand, for any $w \in R^n$ we have

$$(F_{\omega^{-1}}(F_\omega w))_j = \frac{1}{n^2} \sum_{s=0}^{n-1} \omega^{sj} \sum_{t=0}^{n-1} \omega^{-st} w_t = \frac{1}{n^2} \sum_{t=0}^{n-1} \left(\sum_{s=0}^{n-1} \omega^{s(j-t)} \right) w_t = \frac{1}{n} w_j,$$

so $F_{\omega^{-1}} F_\omega w = \frac{1}{n} w$. Taking $w \coloneqq u * v$, we obtain the desired result. □

For the two coefficient rings mentioned earlier, we choose ω as follows:

(1) *Complex case.* For $R = \mathbb{C}$, let $n \geqslant 1$ be any positive integer, and put $\omega \coloneqq e^{2\pi \mathrm{i}/n}$. We denote F_ω in this case by $\mathcal{F}_n \colon \mathbb{C}^n \to \mathbb{C}^n$. Explicitly,

$$(\mathcal{F}_n u)_j = \frac{1}{n} \sum_{k=0}^{n-1} e^{-2\pi \mathrm{i} jk/n} u_k, \qquad u \in \mathbb{C}^n, \quad 0 \leqslant j < n.$$

We also write $\mathcal{F}_n^* \colon \mathbb{C}^n \to \mathbb{C}^n$ for $F_{\omega^{-1}}$.

(2) *Synthetic case.* For $R = \mathscr{R} = \mathbb{C}[y]/(y^r + 1)$, let n be any positive divisor of $2r$. Then $\omega \coloneqq y^{2r/n}$ is a principal n-th root of unity in $\mathscr{R}$. We denote F_ω in this case by $\mathcal{G}_n \colon \mathscr{R}^n \to \mathscr{R}^n$. Explicitly,

$$(\mathcal{G}_n u)_j = \frac{1}{n} \sum_{k=0}^{n-1} y^{-2rjk/n} u_k, \qquad u \in \mathscr{R}^n, \quad 0 \leqslant j < n.$$

We also write $\mathcal{G}_n^* \colon \mathscr{R}^n \to \mathscr{R}^n$ for $F_{\omega^{-1}}$.

All of the concepts introduced above may be generalised to the multidimensional setting as follows. For $u, v \in \otimes_i R^{n_i}$, we define the pointwise product $u \cdot v \in \otimes_i R^{n_i}$ and the convolution product $u * v \in \otimes_i R^{n_i}$ by

$$\begin{aligned} (u \cdot v)_{j_1,\ldots,j_d} &\coloneqq u_{j_1,\ldots,j_d} v_{j_1,\ldots,j_d}, \\ (u * v)_{j_1,\ldots,j_d} &\coloneqq \sum_{k_1=0}^{n_1-1} \cdots \sum_{k_d=0}^{n_d-1} u_{k_1,\ldots,k_d} v_{j_1-k_1,\ldots,j_d-k_d}. \end{aligned}$$

Then $(\otimes_i R^{n_i}, \cdot)$ is isomorphic to the direct sum of $n_1 \cdots n_d$ copies of R, and $(\otimes_i R^{n_i}, *)$ is isomorphic to $R[x_1, \ldots, x_d]/(x^{n_1} - 1, \ldots, x^{n_d} - 1)$.

Let $\omega_1, \ldots, \omega_d \in R$ be principal roots of unity of orders $n_1, \ldots, n_d$. We define an associated R-linear d-dimensional DFT map by taking the tensor product (over R) of the corresponding one-dimensional DFTs, that is,

$$F_{\omega_1,\ldots,\omega_d} \coloneqq \otimes_i F_{\omega_i} \colon \otimes_i R^{n_i} \to \otimes_i R^{n_i}.$$

Explicitly, for $u \in \otimes_i R^{n_i}$ we have

$$(F_{\omega_1,\ldots,\omega_d} u)_{j_1,\ldots,j_d} = \frac{1}{n_1 \cdots n_d} \sum_{k_1=0}^{n_1-1} \cdots \sum_{k_d=0}^{n_d-1} \omega_1^{-j_1 k_1} \cdots \omega_d^{-j_d k_d} u_{k_1,\ldots,k_d}.$$

The multidimensional analogue of Lemma 2.3 is

$$\frac{1}{n_1 \cdots n_d} u * v = n_1 \cdots n_d F_{\omega_1^{-1},\ldots,\omega_d^{-1}}(F_{\omega_1,\ldots,\omega_d} u \cdot F_{\omega_1,\ldots,\omega_d} v), \tag{2.1}$$

and is proved in exactly the same way.

In particular, in the "complex case" we obtain the d-dimensional transform

$$\mathcal{F}_{n_1,\ldots,n_d} \coloneqq \otimes_i \mathcal{F}_{n_i} \colon \otimes_i \mathbb{C}^{n_i} \to \otimes_i \mathbb{C}^{n_i}$$

(take $\omega_i \coloneqq e^{2\pi \mathrm{i}/n_i}$), and in the "synthetic case" the d-dimensional transform

$$\mathcal{G}_{n_1,\ldots,n_d} \coloneqq \otimes_i \mathcal{G}_{n_i} \colon \otimes_i \mathscr{R}^{n_i} \to \otimes_i \mathscr{R}^{n_i}$$

(where each n_i is a divisor of $2r$, and $\omega_i \coloneqq y^{2r/n_i}$). We define similarly $\mathcal{F}^*_{n_1,\ldots,n_d} \coloneqq \otimes_i \mathcal{F}^*_{n_i}$ and $\mathcal{G}^*_{n_1,\ldots,n_d} \coloneqq \otimes_i \mathcal{G}^*_{n_i}$.

Any algorithm for computing $\mathcal{F}_n$ may easily be adapted to obtain an algorithm for computing $\mathcal{F}_n^*$, by adjusting signs appropriately. A similar remark applies to $\mathcal{G}_n$, and to the multidimensional generalisations of these maps. For the rest of the paper, we make use of these observations without further comment.

2.5. **Fixed-point multiplication.** We now consider the complexity of multiplication in the coefficient rings $R = \mathbb{C}$ and $R = \mathscr{R}$. In both cases we reduce the problem to integer multiplication. For the case $R = \mathscr{R}$ (Lemma 2.5) we will express the complexity in terms of $\mathsf{M}(\cdot)$ itself, as this eventually feeds into the main recurrence inequality for $\mathsf{M}(\cdot)$ that we prove in Section 5.3. For the case $R = \mathbb{C}$ (Lemma 2.4) we do not need the best possible bound; to simplify the subsequent complexity analysis, we prefer to use the crude estimate $\mathsf{M}(p) = O(p^{1+\delta})$.

Lemma 2.4 (Multiplication in $\mathbb{C}$)**.** *Given as input $u, v \in \tilde{\mathbb{C}}_\circ$, in time $O(p^{1+\delta})$ we may compute an approximation $\tilde{w} \in \tilde{\mathbb{C}}_\circ$ for $w \coloneqq uv \in \mathbb{C}_\circ$ such that $\varepsilon(\tilde{w}) < 2$.*

Proof. We take $\tilde{w} \coloneqq \rho(w)$, so that $\varepsilon(\tilde{w}) = 2^p \, \|\rho(w) - w\| < \sqrt{2} < 2$. Writing $u = 2^{-p}a$ and $v = 2^{-p}b$ where $a, b \in \mathbb{Z}[\mathrm{i}]$ and $|a|, |b| \leqslant 2^p$, we have $\tilde{w} = 2^{-p}\rho_0(2^{-p}ab)$. Thus $\tilde{w}$ may be computed in time $O(p^{1+\delta})$ by multiplying out the real and imaginary parts of a and b, and then summing and rounding appropriately. □

For the case $R = \mathscr{R}$, observe first that for any $u, v \in \mathscr{R}$ we have $\|uv\| \leqslant r \, \|u\| \, \|v\|$, as each coefficient of $uv = (u_0 + \cdots + u_{r-1}y^{r-1})(v_0 + \cdots + v_{r-1}y^{r-1}) \bmod y^r + 1$ is a sum of exactly r terms of the form $\pm u_i v_j$. In particular, if $u, v \in \mathscr{R}_\circ$, then $uv/r \in \mathscr{R}_\circ$.

Lemma 2.5 (Multiplication in $\mathscr{R}$)**.** *Assume that r is a power of two and that $r < 2^{p-1}$. Given as input $u, v \in \tilde{\mathscr{R}}_\circ$, in time $4\,\mathsf{M}(3rp) + O(rp)$ we may compute an approximation $\tilde{w} \in \tilde{\mathscr{R}}_\circ$ for $w \coloneqq uv/r \in \mathscr{R}_\circ$ such that $\varepsilon(\tilde{w}) < 2$.*

Proof. Write $2^p u = U_0(y) + \mathrm{i}\, U_1(y)$ and $2^p v = V_0(y) + \mathrm{i}\, V_1(y)$ where U_j and V_j are polynomials in $\mathbb{Z}[y]$ of degree less than r and whose coefficients lie in the interval $[-2^p, 2^p]$. Then $2^{2p} r w = W_0(y) + \mathrm{i}\, W_1(y)$ where

$$W_0 \coloneqq (U_0 V_0 - U_1 V_1) \bmod y^r + 1, \qquad W_1 \coloneqq (U_0 V_1 + U_1 V_0) \bmod y^r + 1.$$

We use the following algorithm, which is based on the well-known Kronecker substitution technique [14, Corollary 8.27].

(1) *Pack coefficients.* Evaluate $U_j(2^{3p}), V_j(2^{3p}) \in \mathbb{Z}$ for $j = 0, 1$. As the input coefficients have at most p bits, this amounts to concatenating the coefficients with

appropriate zero-padding (or one-padding in the case of negative coefficients), plus some carry and sign handling. The cost of this step is $O(rp)$.

(2) *Multiply in* $\mathbb{Z}$. Let $W_{j,k} \coloneqq U_j V_k \in \mathbb{Z}[y]$ for $j,k \in \{0,1\}$. Compute the four integer products $W_{j,k}(2^{3p}) = U_j(2^{3p})V_k(2^{3p})$. The cost of this step is $4\,\mathsf{M}(3rp)$.

(3) *Unpack coefficients.* For each pair (j,k), the coefficients of $W_{j,k} \in \mathbb{Z}[y]$ are bounded in absolute value by $r(2^p)^2 < 2^{3p-1}$, so $W_{j,k}$ may be recovered from the integer $W_{j,k}(2^{3p})$ in time $O(rp)$. (In more detail: the constant term of $W_{j,k}$ lies in the interval $(-2^{3p-1}, 2^{3p-1})$, so it is easily read off the last $3p$ bits of $W_{j,k}(2^{3p})$. After stripping off this term, one proceeds to the linear term, and so on.) We then deduce the polynomials $W_0 = (W_{0,0} - W_{1,1}) \bmod y^r + 1$ and $W_1 = (W_{0,1} + W_{1,0}) \bmod y^r + 1$ in time $O(rp)$.

(4) *Scale and round.* Let $c_\ell \coloneqq (W_0)_\ell + \mathrm{i}(W_1)_\ell \in \mathbb{Z}[\mathrm{i}]$ for $\ell \in \{0, \dots, r-1\}$. Then $w = (2^{2p}r)^{-1}(c_0 + \cdots + c_{r-1}y^{r-1})$, so $|c_\ell| \leqslant 2^{2p}r$ for each ℓ. In time $O(rp)$ we may compute $\tilde{w} \coloneqq \rho(w) = 2^{-p}\sum_{j=0}^{r-1} \rho_0((2^p r)^{-1}c_j)y^j \in \tilde{\mathscr{R}}_\circ$ (each division by $2^p r$ amounts to a bit-shift), and as usual $\varepsilon(\tilde{w}) = 2^p \left\|\rho(w) - w\right\| < \sqrt{2} < 2$. □

2.6. **Linear and bilinear maps.** Let $\mathcal{A}\colon V \to W$ be an $\mathbb{C}$-linear map between finite-dimensional vector spaces V and W. We define the operator norm of $\mathcal{A}$ to be

$$\left\|\mathcal{A}\right\| \coloneqq \sup_{v \in V_\circ} \left\|\mathcal{A}v\right\|.$$

For example, the normalised DFT maps F_ω defined in Section 2.4 all have norm exactly 1.

Assume now that $\left\|\mathcal{A}\right\| \leqslant 1$. By a *numerical approximation for* $\mathcal{A}$ we mean a function $\tilde{\mathcal{A}}\colon \tilde{V}_\circ \to \tilde{W}_\circ$ that is computed by some algorithm, typically via fixed-point arithmetic. The error of the approximation is defined to be

$$\varepsilon(\tilde{\mathcal{A}}) \coloneqq \max_{v \in \tilde{V}_\circ} 2^p \left\|\tilde{\mathcal{A}}v - \mathcal{A}v\right\|.$$

We write $\mathsf{C}(\tilde{\mathcal{A}})$ for the time required to compute $\tilde{\mathcal{A}}v$ from v (taking the maximum over all possible inputs $v \in \tilde{V}_\circ$).

Lemma 2.6 (Error propagation)**.** *Let $\mathcal{A}\colon V \to W$ be a $\mathbb{C}$-linear map such that $\left\|\mathcal{A}\right\| \leqslant 1$, and let $v \in V_\circ$. Let $\tilde{\mathcal{A}}\colon \tilde{V}_\circ \to \tilde{W}_\circ$ be a numerical approximation for $\mathcal{A}$, and let $\tilde{v} \in \tilde{V}_\circ$ be an approximation for v. Then $\tilde{w} \coloneqq \tilde{\mathcal{A}}\tilde{v} \in \tilde{W}_\circ$ is an approximation for $w \coloneqq \mathcal{A}v \in W_\circ$ such that $\varepsilon(\tilde{w}) \leqslant \varepsilon(\tilde{\mathcal{A}}) + \varepsilon(\tilde{v})$.*

Proof. We have

$$\begin{aligned}\varepsilon(\tilde{w}) = 2^p \left\|\tilde{\mathcal{A}}\tilde{v} - \mathcal{A}v\right\| &\leqslant 2^p \left\|\tilde{\mathcal{A}}\tilde{v} - \mathcal{A}\tilde{v}\right\| + 2^p \left\|\mathcal{A}\tilde{v} - \mathcal{A}v\right\| \\ &\leqslant \varepsilon(\tilde{\mathcal{A}}) + 2^p \left\|\mathcal{A}\right\| \left\|\tilde{v} - v\right\| \leqslant \varepsilon(\tilde{\mathcal{A}}) + \varepsilon(\tilde{v}).\end{aligned}$$ □

Lemma 2.6 yields the following estimate for compositions of linear maps.

Corollary 2.7 (Composition)**.** *Let $\mathcal{A}\colon U \to V$ and $\mathcal{B}\colon V \to W$ be $\mathbb{C}$-linear maps such that $\left\|\mathcal{A}\right\|, \left\|\mathcal{B}\right\| \leqslant 1$. Let $\tilde{\mathcal{A}}\colon \tilde{U}_\circ \to \tilde{V}_\circ$ and $\tilde{\mathcal{B}}\colon \tilde{V}_\circ \to \tilde{W}_\circ$ be numerical approximations. Then $\tilde{\mathcal{C}} \coloneqq \tilde{\mathcal{B}}\tilde{\mathcal{A}}\colon \tilde{U}_\circ \to \tilde{W}_\circ$ is a numerical approximation for $\mathcal{C} \coloneqq \mathcal{B}\mathcal{A}\colon U \to W$ such that $\varepsilon(\tilde{\mathcal{C}}) \leqslant \varepsilon(\tilde{\mathcal{B}}) + \varepsilon(\tilde{\mathcal{A}})$.*

Proof. For any $u \in \tilde{U}_\circ$, if we set $v \coloneqq \mathcal{A}u \in V_\circ$ and $\tilde{v} \coloneqq \tilde{\mathcal{A}}u \in \tilde{V}_\circ$, then

$$2^p \left\|\tilde{\mathcal{B}}\tilde{\mathcal{A}}u - \mathcal{B}\mathcal{A}u\right\| = 2^p \left\|\tilde{\mathcal{B}}\tilde{v} - \mathcal{B}v\right\| \leqslant \varepsilon(\tilde{\mathcal{B}}) + \varepsilon(\tilde{v}) \leqslant \varepsilon(\tilde{\mathcal{B}}) + \varepsilon(\tilde{\mathcal{A}}).$$ □

The above definitions and results may be adapted to the case of a $\mathbb{C}$-bilinear map $\mathcal{A}\colon U \times V \to W$ as follows. We define

$$\|\mathcal{A}\| \coloneqq \sup_{u \in U_\circ,\, v \in V_\circ} \|\mathcal{A}(u,v)\|.$$

If $\|\mathcal{A}\| \leqslant 1$, then a *numerical approximation for* $\mathcal{A}$ is a function $\tilde{\mathcal{A}}\colon \tilde{U}_\circ \times \tilde{V}_\circ \to \tilde{W}_\circ$ that is computed by some algorithm. The error of the approximation is

$$\varepsilon(\tilde{\mathcal{A}}) \coloneqq \max_{u \in \tilde{U}_\circ,\, v \in \tilde{V}_\circ} 2^p \, \|\tilde{\mathcal{A}}(u,v) - \mathcal{A}(u,v)\|,$$

and $\mathsf{C}(\tilde{\mathcal{A}})$ denotes the time required to compute $\tilde{\mathcal{A}}(u,v)$ from u and v. Lemma 2.6 has the following analogue in the bilinear case.

Lemma 2.8 (Bilinear error propagation)**.** *Let $\mathcal{A}\colon U \times V \to W$ be a $\mathbb{C}$-bilinear map with $\|\mathcal{A}\| \leqslant 1$, and let $u \in U_\circ$, $v \in V_\circ$. Let $\tilde{\mathcal{A}}\colon \tilde{U}_\circ \times \tilde{V}_\circ \to \tilde{W}_\circ$, $\tilde{u} \in \tilde{U}_\circ$, $\tilde{v} \in \tilde{V}_\circ$ be approximations. Then $\tilde{w} \coloneqq \tilde{\mathcal{A}}(\tilde{u},\tilde{v}) \in \tilde{W}_\circ$ is an approximation for $w \coloneqq \mathcal{A}(u,v) \in W_\circ$ such that $\varepsilon(\tilde{w}) \leqslant \varepsilon(\tilde{\mathcal{A}}) + \varepsilon(\tilde{u}) + \varepsilon(\tilde{v})$.*

Proof. We have

$$\begin{aligned}
\varepsilon(\tilde{w}) &\leqslant 2^p(\|\tilde{\mathcal{A}}(\tilde{u},\tilde{v}) - \mathcal{A}(\tilde{u},\tilde{v})\| + \|\mathcal{A}(\tilde{u},\tilde{v}) - \mathcal{A}(u,\tilde{v})\| + \|\mathcal{A}(u,\tilde{v}) - \mathcal{A}(u,v)\|) \\
&= 2^p(\|\tilde{\mathcal{A}}(\tilde{u},\tilde{v}) - \mathcal{A}(\tilde{u},\tilde{v})\| + \|\mathcal{A}(\tilde{u}-u,\tilde{v})\| + \|\mathcal{A}(u,\tilde{v}-v)\|) \\
&\leqslant \varepsilon(\tilde{\mathcal{A}}) + 2^p \, \|\mathcal{A}\| \, \|\tilde{u}-u\| \, \|\tilde{v}\| + 2^p \, \|\mathcal{A}\| \, \|u\| \, \|\tilde{v}-v\| \\
&\leqslant \varepsilon(\tilde{\mathcal{A}}) + \varepsilon(\tilde{u}) + \varepsilon(\tilde{v}).
\end{aligned}$$

□

The following application of Lemma 2.8 will frequently be useful.

Corollary 2.9. *Let $u, v \in \mathbb{C}_\circ$, and let $w \coloneqq uv \in \mathbb{C}_\circ$. Given as input approximations $\tilde{u}, \tilde{v} \in \tilde{\mathbb{C}}_\circ$, in time $O(p^{1+\delta})$ we may compute an approximation $\tilde{w} \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{w}) < \varepsilon(\tilde{u}) + \varepsilon(\tilde{v}) + 2$.*

Proof. Define a bilinear map $\mathcal{A}\colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ by $\mathcal{A}(u,v) \coloneqq uv$. Then $\|\mathcal{A}\| \leqslant 1$, and Lemma 2.4 yields an approximation $\tilde{\mathcal{A}}\colon \tilde{\mathbb{C}}_\circ \times \tilde{\mathbb{C}}_\circ \to \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{\mathcal{A}}) < 2$ and $\mathsf{C}(\tilde{\mathcal{A}}) = O(p^{1+\delta})$. Applying Lemma 2.8 to $\mathcal{A}$ and $\tilde{\mathcal{A}}$ yields the desired result. □

2.7. **Tensor products.** The following result is used to construct numerical approximations for tensor products of linear maps over a coefficient ring.

Lemma 2.10 (Tensor products)**.** *Let R be a coefficient ring of dimension r, and let $m_1, \ldots, m_d, n_1, \ldots, n_d \geqslant 1$. Put $M \coloneqq \prod_i \max(m_i, n_i)$, and assume that $M \geqslant 2$. For $i \in \{1, \ldots, d\}$, let $\mathcal{A}_i\colon R^{m_i} \to R^{n_i}$ be an R-linear map with $\|\mathcal{A}_i\| \leqslant 1$, and let $\tilde{\mathcal{A}}_i\colon \tilde{R}_\circ^{m_i} \to \tilde{R}_\circ^{n_i}$ be a numerical approximation. Let $\mathcal{A} \coloneqq \otimes_i \mathcal{A}_i\colon \otimes_i R^{m_i} \to \otimes_i R^{n_i}$ (note that automatically $\|\mathcal{A}\| \leqslant 1$).*

Then we may construct a numerical approximation $\tilde{\mathcal{A}}\colon \otimes_i \tilde{R}_\circ^{m_i} \to \otimes_i \tilde{R}_\circ^{n_i}$ such that $\varepsilon(\tilde{\mathcal{A}}) \leqslant \sum_i \varepsilon(\tilde{\mathcal{A}}_i)$ and

$$\mathsf{C}(\tilde{\mathcal{A}}) \leqslant M \sum_i \frac{\mathsf{C}(\tilde{\mathcal{A}}_i)}{m_i} + O(Mrp \log M).$$

Proof. For $i \in \{0, 1, \ldots, d\}$, let

$$U^i \coloneqq R^{n_1} \otimes \cdots \otimes R^{n_{i-1}} \otimes R^{n_i} \otimes R^{m_{i+1}} \otimes \cdots \otimes R^{m_d}.$$

In particular, $U^0 = \otimes_i R^{m_i}$ and $U^d = \otimes_i R^{n_i}$. The map $\mathcal{A}\colon U^0 \to U^d$ admits a decomposition $\mathcal{A} = \mathcal{B}_d \cdots \mathcal{B}_1$ where $\mathcal{B}_i\colon U^{i-1} \to U^i$ is given by

$$\mathcal{B}_i \coloneqq \mathcal{I}_{n_1} \otimes \cdots \otimes \mathcal{I}_{n_{i-1}} \otimes \mathcal{A}_i \otimes \mathcal{I}_{m_{i+1}} \otimes \cdots \otimes \mathcal{I}_{m_d}$$

(here $\mathcal{I}_k$ denotes the identity map on R^k). In other words, $\mathcal{B}_i$ acts by applying $\mathcal{A}_i$ separately on each i-slice. Explicitly, for any $u \in U^{i-1}$ we have $(\mathcal{B}_i u)_{j_1,\ldots,j_d} = (\mathcal{A}_i v)_{j_i}$ where $v \in R^{m_i}$ is the vector defined by $v_k \coloneqq u_{j_1,\ldots,j_{i-1},k,j_{i+1},\ldots,j_d}$.

We may define an approximation $\tilde{\mathcal{B}}_i\colon \tilde{U}_\circ^{i-1} \to \tilde{U}_\circ^i$ by mimicking the above formula for $\mathcal{B}_i$; i.e., for $u \in \tilde{U}_\circ^{i-1}$ we define $(\tilde{\mathcal{B}}_i u)_{j_1,\ldots,j_d} \coloneqq (\tilde{\mathcal{A}}_i v)_{j_i}$, where $v \in \tilde{R}_\circ^{m_i}$ is given by $v_k \coloneqq u_{j_1,\ldots,j_{i-1},k,j_{i+1},\ldots,j_d}$. We may evaluate $\tilde{\mathcal{B}}_i$ by first rearranging the data so that each i-slice is stored contiguously (see Section 2.3), then applying $\tilde{\mathcal{A}}_i$ to each i-slice, and finally rearranging the data back into the correct order. We then define $\tilde{\mathcal{A}} \coloneqq \tilde{\mathcal{B}}_d \cdots \tilde{\mathcal{B}}_1$.

We clearly have $\varepsilon(\tilde{\mathcal{B}}_i) = \varepsilon(\tilde{\mathcal{A}}_i)$ for all i, so by Corollary 2.7 we obtain $\varepsilon(\tilde{\mathcal{A}}) \leqslant \sum_i \varepsilon(\tilde{\mathcal{B}}_i) = \sum_i \varepsilon(\tilde{\mathcal{A}}_i)$. The cost of the data rearrangement at stage i is

$$\begin{aligned} O(n_1 \cdots n_{i-1} n_i m_{i+1} \cdots m_d rp \log n_i) + O(n_1 \cdots n_{i-1} m_i m_{i+1} \cdots m_d rp \log m_i) \\ = O(Mrp\,(\log n_i + \log m_i)), \end{aligned}$$

so the total over all i is $O(Mrp \log M)$ (here we have used the hypothesis that $M \geqslant 2$). The total cost of the invocations of $\tilde{\mathcal{A}}_i$ is

$$\sum_i n_1 \cdots n_{i-1} m_{i+1} \cdots m_d\, \mathsf{C}(\tilde{\mathcal{A}}_i) \leqslant \sum_i (M/m_i)\, \mathsf{C}(\tilde{\mathcal{A}}_i). \qquad \square$$

2.8. **Exponential functions.** The next three results concern the approximation of real and complex exponentials. We give only a sketch of the proofs, omitting routine details of the error analysis. We use the fact that the constants π and $\log 2$ may be approximated to within an error of 2^{-p} in time $O(p^{1+\delta})$, and that for z lying in any fixed bounded subset of $\mathbb{C}$, we may approximate e^z to within an error of 2^{-p} in time $O(p^{1+\delta})$ (in fact, in time $O(\mathsf{M}(p) \log p)$; see [3, Ch. 6–7] or [5, Ch. 4]).

Lemma 2.11 (Complex exponentials)**.** *Let $k \geqslant 1$ and j be integers such that $0 \leqslant j < k$, and let $w \coloneqq e^{2\pi \mathrm{i} j/k} \in \mathbb{C}_\circ$. Given j and k as input, we may compute an approximation $\tilde{w} \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{w}) < 2$ in time $O(\max(p, \log k)^{1+\delta})$.*

Proof. Temporarily increasing the working precision to $p' \coloneqq p + C$ for a suitable constant $C > 0$, we first compute a p'-bit approximation for $2\pi j/k \in \mathbb{R}$ in time $O(\max(p, \log k)^{1+\delta})$, then approximate $\exp(-2^{-p'} + 2\pi \mathrm{i} j/k) \in \mathbb{C}_\circ$ in time $O(p^{1+\delta})$ (the $2^{-p'}$ term ensures that the approximation lies within in the unit circle). We finally round towards zero to obtain an element of $\tilde{\mathbb{C}}_\circ$ at the original precision p. $\square$

Lemma 2.12 (Real exponentials, negative case)**.** *Let $k \geqslant 1$ and $j \geqslant 0$ be integers, and let $w \coloneqq e^{-\pi j/k} \in \mathbb{C}_\circ$. Given j and k as input, we may compute an approximation $\tilde{w} \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{w}) < 2$ in time $O(\max(p, \log(|j|+1), \log k)^{1+\delta})$.*

Proof. We first check whether $j > kp$ in time $O(\max(\log p, \log(|j|+1), \log k)^{1+\delta})$. If so, then $e^{-\pi j/k} < e^{-\pi p} < 2^{-p}$, so we may simply take $\tilde{w} \coloneqq 0$.

Otherwise, we may assume that $0 \leqslant j/k \leqslant p$. In this case, we first compute an integer $\tau \geqslant 0$ such that $\tau \leqslant \frac{\pi}{\log 2} j/k \leqslant \tau + 2$ in time $O(\max(\log p, \log k)^{1+\delta})$ (note that $\tau = O(p)$). Temporarily increasing the precision to $p' \coloneqq p + \lceil \log_2 p \rceil + C$, we now compute an approximation for $z \coloneqq \tau \log 2 - \pi j/k \in [-2 \log 2, 0]$ in time

$O(\max(p, \log k)^{1+\delta})$ and then for $e^z = 2^\tau e^{-\pi j/k} \leqslant 1$ in time $O(p^{1+\delta})$. We finally divide by 2^τ and round towards zero to obtain an approximation for $e^{-\pi j/k}$ in $\tilde{\mathbb{C}}_\circ$ at the original precision p. □

Lemma 2.13 (Real exponentials, positive case)**.** *Let $k \geqslant 1$, $j \geqslant 0$ and $\sigma \geqslant 0$ be integers, and assume that $e^{\pi j/k} \leqslant 2^\sigma$ and $\sigma \leqslant 2p$. Let $w \coloneqq 2^{-\sigma} e^{\pi j/k} \in \mathbb{C}_\circ$. Given j, k and σ as input, we may compute an approximation $\tilde{w} \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{w}) < 2$ in time $O(\max(p, \log k)^{1+\delta})$.*

Proof. The hypotheses automatically ensure that $j < kp$. We now proceed along similar lines to the proof of Lemma 2.12: we first compute an integer $\tau \geqslant 0$ near $\sigma - \frac{\pi}{\log 2} j/k$, and then at suitably increased precision we approximate successively $z \coloneqq (\tau - \sigma)\log 2 + \pi j/k$ and $e^z = 2^{\tau-\sigma} e^{\pi j/k}$, and finally divide by 2^τ and round towards zero to obtain an approximation for $2^{-\sigma} e^{\pi j/k}$ at precision p. □

3. Complex transforms for power-of-two sizes

Let $p \geqslant 100$ be the working precision as defined in Section 2. The goal of this section is to construct an efficiently computable approximation for the d-dimensional complex transform $\mathcal{F}_{t_1,\ldots,t_d} \colon \otimes_i \mathbb{C}^{t_i} \to \otimes_i \mathbb{C}^{t_i}$ in the special case that the t_i are powers of two. The following theorem is proved at the end of the section.

Theorem 3.1 (Power-of-two complex transforms)**.** *Let $d \geqslant 2$ and let $t_1, \ldots, t_d$ be powers of two such that $t_d \geqslant \cdots \geqslant t_1 \geqslant 2$. Let $T \coloneqq t_1 \cdots t_d$ and assume that $T < 2^p$. Then we may construct a numerical approximation $\tilde{\mathcal{F}}_{t_1,\ldots,t_d} \colon \otimes_i \tilde{\mathbb{C}}_\circ^{t_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ for $\mathcal{F}_{t_1,\ldots,t_d}$ such that $\varepsilon(\tilde{\mathcal{F}}_{t_1,\ldots,t_d}) < 8T \log_2 T$ and*

$$\mathsf{C}(\tilde{\mathcal{F}}_{t_1,\ldots,t_d}) < \frac{4T}{t_d}\, \mathsf{M}(3t_d p) + O(Tp \log T + Tp^{1+\delta}).$$

Throughout this section we set

$$r \coloneqq t_d, \qquad \mathscr{R} \coloneqq \mathbb{C}[y]/(y^r + 1).$$

The basic idea of the proof of Theorem 3.1 is to use Bluestein's method [2] to reduce the DFT to the problem of computing a $(d-1)$-dimensional cyclic convolution of size $t_1 \times \cdots \times t_{d-1}$ over $\mathscr{R}$, and then to perform that convolution by taking advantage of the synthetic roots of unity in $\mathscr{R}$. The $\mathsf{M}(\cdot)$ term in the complexity bound arises from the pointwise multiplications in $\mathscr{R}$. The $O(Tp \log T)$ terms covers the cost of the synthetic FFTs over $\mathscr{R}$, and the $O(Tp^{1+\delta})$ term covers various auxiliary operations.

For the rest of this section, $\otimes_i \mathscr{R}^{t_i}$ always means $\otimes_{i=1}^{d-1} \mathscr{R}^{t_i}$, and $\otimes_i \mathbb{C}^{t_i}$ always means $\otimes_{i=1}^{d} \mathbb{C}^{t_i}$.

3.1. Transforms and convolutions over $\mathscr{R}$.

We begin with the one-dimensional case. Recall that we have defined a synthetic transform $\mathcal{G}_t \colon \mathscr{R}^t \to \mathscr{R}^t$ for each positive divisor t of $2r$, i.e., for $t \in \{1, 2, 4, \ldots, 2r\}$.

Lemma 3.2 (FFT over $\mathscr{R}$)**.** *For $t \in \{1, 2, 4, \ldots, 2r\}$, we may construct a numerical approximation $\tilde{\mathcal{G}}_t \colon \tilde{\mathscr{R}}_\circ^t \to \tilde{\mathscr{R}}_\circ^t$ for $\mathcal{G}_t$ such that $\varepsilon(\tilde{\mathcal{G}}_t) \leqslant \log_2 t$ and $\mathsf{C}(\tilde{\mathcal{G}}_t) = O(trp \log 2t)$.*

Proof. For the case $t = 1$, observe that $\mathcal{G}_1 \colon \mathscr{R} \to \mathscr{R}$ is the identity map, and admits the trivial approximation $\tilde{\mathcal{G}}_1 \colon \tilde{\mathscr{R}}_\circ \to \tilde{\mathscr{R}}_\circ$ given by $\tilde{\mathcal{G}}_1 u \coloneqq u$. This satisfies $\varepsilon(\tilde{\mathcal{G}}_1) = 0$ and $\mathsf{C}(\tilde{\mathcal{G}}_1) = O(rp)$.

Now let $t \in \{2, 4, \dots, 2r\}$, and assume that $\tilde{\mathcal{G}}_{t/2} \colon \tilde{\mathscr{R}}_\circ^{t/2} \to \tilde{\mathscr{R}}_\circ^{t/2}$ has already been constructed. Given as input $u \in \tilde{\mathscr{R}}_\circ^t$, we will use the well-known Cooley–Tukey algorithm [7] to approximate $\mathcal{G}_t u \in \mathscr{R}_\circ^t$.

For any $j \in \{0, \dots, t-1\}$, observe that

$$\begin{aligned}(\mathcal{G}_t u)_j = \frac{1}{t} \sum_{k=0}^{t-1} y^{-2rjk/t} u_k &= \frac{1}{t} \sum_{k=0}^{\frac{t}{2}-1} y^{-2rjk/t} u_k + \frac{1}{t} \sum_{k=0}^{\frac{t}{2}-1} y^{-2rj(k+\frac{t}{2})/t} u_{k+\frac{t}{2}} \\ &= \frac{1}{t/2} \sum_{k=0}^{\frac{t}{2}-1} y^{-2rjk/t} \frac{u_k + (-1)^j u_{k+\frac{t}{2}}}{2},\end{aligned}$$

where we have used the fact that $y^r = -1$. For $\ell \in \{0, \dots, \frac{t}{2} - 1\}$ this implies that $(\mathcal{G}_t u)_{2\ell} = (\mathcal{G}_{t/2} v)_\ell$ and $(\mathcal{G}_t u)_{2\ell+1} = (\mathcal{G}_{t/2} w)_\ell$, where $v, w \in \mathscr{R}_\circ^{t/2}$ are given by

$$v_k \coloneqq \tfrac{1}{2}(u_k + u_{k+\frac{t}{2}}), \qquad w_k \coloneqq \tfrac{1}{2} y^{-2rk/t}(u_k - u_{k+\frac{t}{2}}), \qquad 0 \leqslant k < t/2.$$

We may therefore use the following algorithm.

(1) *Butterflies.* For $k \in \{0, \dots, \frac{t}{2} - 1\}$, we use Lemma 2.1 to compute approximations $\tilde{v}_k, \tilde{w}'_k \in \tilde{\mathscr{R}}_\circ$ for v_k and $w'_k \coloneqq \frac{1}{2}(u_k - u_{k+\frac{t}{2}})$ such that $\varepsilon(\tilde{v}_k), \varepsilon(\tilde{w}'_k) < 1$. We then compute an approximation $\tilde{w}_k \in \tilde{\mathscr{R}}_\circ$ for $w_k = y^{-2rk/t} w'_k$; as $y^r = -1$, this amounts to cyclically permuting the coefficients of $\tilde{w}'_k$ (and adjusting signs), and clearly $\varepsilon(\tilde{w}_k) = \varepsilon(\tilde{w}'_k) < 1$. The cost of this step is $O(trp)$.

(2) *Recurse.* We compute $\tilde{\mathcal{G}}_{t/2}\tilde{v}$ and $\tilde{\mathcal{G}}_{t/2}\tilde{w}$ using the previously constructed map $\tilde{\mathcal{G}}_{t/2}$, and interleave the results (at a further cost of $O(trp)$) to obtain the output vector $\tilde{\mathcal{G}}_t u \in \tilde{\mathscr{R}}_\circ^t$ defined by $(\tilde{\mathcal{G}}_t u)_{2\ell} \coloneqq (\tilde{\mathcal{G}}_{t/2}\tilde{v})_\ell$ and $(\tilde{\mathcal{G}}_t u)_{2\ell+1} \coloneqq (\tilde{\mathcal{G}}_{t/2}\tilde{w})_\ell$ for $\ell \in \{0, \dots, \frac{t}{2} - 1\}$.

By Lemma 2.6 and induction we have

$$\begin{aligned}2^p \, \|(\tilde{\mathcal{G}}_t u)_{2\ell} - (\mathcal{G}_t u)_{2\ell}\| &= 2^p \, \|(\tilde{\mathcal{G}}_{t/2}\tilde{v})_\ell - (\mathcal{G}_{t/2} v)_\ell\| \\ &\leqslant \varepsilon(\tilde{\mathcal{G}}_{t/2}) + \varepsilon(\tilde{v}) \leqslant \log_2(t/2) + 1 = \log_2 t.\end{aligned}$$

A similar argument applies for $(\tilde{\mathcal{G}}_t u)_{2\ell+1}$. Therefore $\varepsilon(\tilde{\mathcal{G}}_t) \leqslant \log_2 t$. As for the complexity, the above discussion shows that $\mathsf{C}(\tilde{\mathcal{G}}_t) < 2\,\mathsf{C}(\tilde{\mathcal{G}}_{t/2}) + O(trp)$. This immediately yields the bound $\mathsf{C}(\tilde{\mathcal{G}}_t) = O(trp \log t)$ for $t \geqslant 2$. □

Combining Lemmas 3.2 and 2.10, we obtain the following approximation for the multidimensional transform $\mathcal{G}_{t_1,\dots,t_{d-1}} \colon \otimes_i \mathscr{R}^{t_i} \to \otimes_i \mathscr{R}^{t_i}$.

Proposition 3.3 (Multivariate FFT over $\mathscr{R}$)**.** *Let $t_1, \dots, t_d$ and T be as in Theorem 3.1. We may construct a numerical approximation $\tilde{\mathcal{G}}_{t_1,\dots,t_{d-1}} \colon \otimes_i \tilde{\mathscr{R}}_\circ^{t_i} \to \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$ for $\mathcal{G}_{t_1,\dots,t_{d-1}}$ such that $\varepsilon(\tilde{\mathcal{G}}_{t_1,\dots,t_{d-1}}) < \log_2 T$ and $\mathsf{C}(\tilde{\mathcal{G}}_{t_1,\dots,t_{d-1}}) = O(Tp \log T)$.*

Proof. We apply Lemma 2.10 (with d replaced by $d-1$), taking $R \coloneqq \mathscr{R}$, $m_i \coloneqq t_i$, $n_i \coloneqq t_i$, $\mathcal{A}_i \coloneqq \mathcal{G}_{t_i}$ for $i \in \{1, \dots, d-1\}$. The quantity M defined in Lemma 2.10 is given by $M = t_1 \cdots t_{d-1} = T/r$. Using the approximations $\tilde{\mathcal{G}}_{t_i}$ constructed in

Lemma 3.2, we obtain

$$\varepsilon(\tilde{\mathcal{G}}_{t_1,\dots,t_{d-1}}) \leqslant \sum_{i=1}^{d-1} \varepsilon(\tilde{\mathcal{G}}_{t_i}) \leqslant \sum_{i=1}^{d-1} \log_2 t_i = \log_2(T/r) < \log_2 T$$

and

$$\begin{aligned}\mathsf{C}(\tilde{\mathcal{G}}_{t_1,\dots,t_{d-1}}) &\leqslant \frac{T}{r} \sum_{i=1}^{d-1} \frac{\mathsf{C}(\tilde{\mathcal{G}}_{t_i})}{t_i} + O((T/r)rp\log(T/r)) \\ &= \frac{T}{r} \sum_{i=1}^{d-1} O(rp \log 2t_i) + O(Tp\log T) = O(Tp\log T). \qquad \square\end{aligned}$$

Next we will use the above result to approximate the normalised $(d-1)$-dimensional convolution map $\mathcal{M}_{\mathscr{R}} \colon \otimes_i \mathscr{R}^{t_i} \times \otimes_i \mathscr{R}^{t_i} \to \otimes_i \mathscr{R}^{t_i}$ defined by

$$\mathcal{M}_{\mathscr{R}}(u, v) \coloneqq \frac{1}{T} u * v, \qquad u, v \in \otimes_i \mathscr{R}^{t_i}.$$

Note that $\|\mathcal{M}_{\mathscr{R}}\| \leqslant 1$; indeed, each component of $u * v$ is a sum of $t_1 \cdots t_{d-1} = T/r$ terms of the form $u_{j_1,\dots,j_{d-1}} v_{k_1,\dots,k_{d-1}}$, and we saw earlier that $\|ab\| \leqslant r \|a\| \|b\|$ for all $a, b \in \mathscr{R}$.

Proposition 3.4 (Convolution over $\mathscr{R}$)**.** *Let $t_1, \dots, t_d$ and T be as in Theorem 3.1. We may construct a numerical approximation $\tilde{\mathcal{M}}_{\mathscr{R}} \colon \otimes_i \tilde{\mathscr{R}}_\circ^{t_i} \times \otimes_i \tilde{\mathscr{R}}_\circ^{t_i} \to \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$ for $\mathcal{M}_{\mathscr{R}}$ such that $\varepsilon(\tilde{\mathcal{M}}_{\mathscr{R}}) < 3T\log_2 T + 2T + 3$ and*

$$\mathsf{C}(\tilde{\mathcal{M}}_{\mathscr{R}}) < \frac{4T}{r} \mathsf{M}(3rp) + O(Tp\log T + Tp^{1+\delta}).$$

Proof. We are given as input $u, v \in \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$. Let $w \coloneqq \mathcal{M}_{\mathscr{R}}(u, v) = \frac{1}{T} u * v \in \otimes_i \mathscr{R}_\circ^{t_i}$ be the exact (normalised) convolution. According to (2.1) we have

$$\frac{1}{t_1 \cdots t_{d-1}} u * v = (t_1 \cdots t_{d-1}) \mathcal{G}^*_{t_1,\dots,t_{d-1}}((\mathcal{G}_{t_1,\dots,t_{d-1}} u) \cdot (\mathcal{G}_{t_1,\dots,t_{d-1}} v)).$$

Dividing both sides by $r = t_d$, we obtain $w = (T/r) w'$ where

$$w' \coloneqq \mathcal{G}^*_{t_1,\dots,t_{d-1}} \left(\tfrac{1}{r}(\mathcal{G}_{t_1,\dots,t_{d-1}} u) \cdot (\mathcal{G}_{t_1,\dots,t_{d-1}} v)\right) \in \otimes_i \mathscr{R}_\circ^{t_i}.$$

We use the following algorithm.

(1) *Forward transforms.* We invoke Proposition 3.3 to compute approximations $\tilde{u}', \tilde{v}' \in \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$ for $u' \coloneqq \mathcal{G}_{t_1,\dots,t_{d-1}} u \in \otimes_i \mathscr{R}_\circ^{t_i}$ and $v' \coloneqq \mathcal{G}_{t_1,\dots,t_{d-1}} v \in \otimes_i \mathscr{R}_\circ^{t_i}$, with $\varepsilon(\tilde{u}'), \varepsilon(\tilde{v}') < \log_2 T$. The cost of this step (and step (3) below) is $O(Tp\log T)$.

(2) *Pointwise multiplications.* Let $\mathcal{A} \colon \mathscr{R} \times \mathscr{R} \to \mathscr{R}$ be the normalised multiplication map defined by $\mathcal{A}(a, b) \coloneqq ab/r$; then Lemma 2.5 yields an approximation $\tilde{\mathcal{A}} \colon \tilde{\mathscr{R}}_\circ \times \tilde{\mathscr{R}}_\circ \to \tilde{\mathscr{R}}_\circ$ such that $\varepsilon(\tilde{\mathcal{A}}) < 2$ (note that $r = t_d \leqslant T/2 < 2^{p-1}$). Applying $\tilde{\mathcal{A}}$ to each component of $\tilde{u}'$ and $\tilde{v}'$, we obtain an approximation $\tilde{z} \in \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$ for $z \coloneqq \frac{1}{r} u' \cdot v' \in \otimes_i \mathscr{R}_\circ^{t_i}$. This step requires time

$$\frac{T}{r}(4\,\mathsf{M}(3rp) + O(rp)) = \frac{4T}{r} \mathsf{M}(3rp) + O(Tp),$$

and by Lemma 2.8 we have

$$\varepsilon(\tilde{z}) \leqslant \varepsilon(\tilde{\mathcal{A}}) + \varepsilon(\tilde{u}') + \varepsilon(\tilde{v}') < 2 + \log_2 T + \log_2 T = 2\log_2 T + 2.$$

(3) *Inverse transform.* We use Proposition 3.3 again to compute an approximation $\tilde{w}' \in \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$ for $w' = \mathcal{G}^*_{t_1,\dots,t_{d-1}} z \in \otimes_i \mathscr{R}_\circ^{t_i}$; by Lemma 2.6 we obtain

$$\varepsilon(\tilde{w}') \leqslant \varepsilon(\tilde{\mathcal{G}}^*_{t_1,\dots,t_{d-1}}) + \varepsilon(\tilde{z}) < \log_2 T + (2\log_2 T + 2) = 3\log_2 T + 2.$$

(4) *Scaling.* Recall that $w = (T/r)w'$ and that $\|w\| \leqslant 1$. We may therefore apply Lemma 2.2 (with $c \coloneqq T/r \leqslant 2^p$) to compute an approximation $\tilde{w} \in \otimes_i \tilde{\mathscr{R}}_\circ^{t_i}$ such that

$$\varepsilon(\tilde{w}) < 2(T/r)\varepsilon(\tilde{w}') + 3 \leqslant T(3\log_2 T + 2) + 3$$

(here we have used the hypothesis that $r = t_d \geqslant 2$). The cost of this scaling step is $O(Tp^{1+\delta})$. Finally we take $\tilde{\mathcal{M}}_{\mathscr{R}}(u, v) \coloneqq \tilde{w}$. □

3.2. **Transforms and convolutions over $\mathbb{C}$.** We now transfer the results of the previous section from $\mathscr{R}$ to $\mathbb{C}$. Consider the normalised d-dimensional convolution map $\mathcal{M}_{\mathbb{C}} \colon \otimes_i \mathbb{C}^{t_i} \times \otimes_i \mathbb{C}^{t_i} \to \otimes_i \mathbb{C}^{t_i}$ defined by

$$\mathcal{M}_{\mathbb{C}}(u, v) \coloneqq \frac{1}{T} u * v, \qquad u, v \in \otimes_i \mathbb{C}^{t_i}.$$

As before we have $\|\mathcal{M}_{\mathbb{C}}\| \leqslant 1$.

Proposition 3.5 (Convolution over $\mathbb{C}$)**.** *Let $t_1, \dots, t_d$ and T be as in Theorem 3.1. We may construct a numerical approximation $\tilde{\mathcal{M}}_{\mathbb{C}} \colon \otimes_i \tilde{\mathbb{C}}_\circ^{t_i} \times \otimes_i \tilde{\mathbb{C}}_\circ^{t_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ for $\mathcal{M}_{\mathbb{C}}$ such that $\varepsilon(\tilde{\mathcal{M}}_{\mathbb{C}}) < 3T\log_2 T + 2T + 15$ and*

$$\mathsf{C}(\tilde{\mathcal{M}}_{\mathbb{C}}) < \frac{4T}{r}\,\mathsf{M}(3rp) + O(Tp\log T + Tp^{1+\delta}).$$

Proof. Let $\zeta \coloneqq e^{\pi \mathrm{i}/r}$ and consider the $\mathbb{C}$-linear map $\mathcal{S} \colon \mathbb{C}^r \to \mathscr{R}$ defined by

$$\mathcal{S}(u_0, \dots, u_{r-1}) \coloneqq u_0 + \zeta u_1 y + \cdots + \zeta^{r-1} u_{r-1} y^{r-1}.$$

Then $\mathcal{S}$ is a isomorphism of rings between $(\mathbb{C}^r, *)$ and $\mathscr{R}$; in fact, recalling that $(\mathbb{C}^r, *) \cong \mathbb{C}[x]/(x^r - 1)$, we may regard $\mathcal{S}$ as the map sending x to ζy. Moreover, $\mathcal{S}$ induces an isomorphism of rings

$$\mathcal{T} \colon (\otimes_{i=1}^d \mathbb{C}^{t_i}, *) \to (\otimes_{i=1}^{d-1} \mathscr{R}^{t_i}, *).$$

Indeed, identifying these rings respectively as $\mathbb{C}[x_1, \dots, x_d]/(x_1^{t_1} - 1, \dots, x_d^{t_d} - 1)$ and $\mathbb{C}[x_1, \dots, x_{d-1}, y]/(x_1^{t_1} - 1, \dots, x_{d-1}^{t_{d-1}} - 1, y^r + 1)$, the isomorphism $\mathcal{T}$ sends $u(x_1, \dots, x_{d-1}, x_d)$ to $u(x_1, \dots, x_{d-1}, \zeta y)$. Writing $\mathcal{U} \coloneqq \mathcal{T}^{-1}$ for the inverse isomorphism, we obtain

$$\mathcal{M}_{\mathbb{C}}(u, v) = \mathcal{U}(\mathcal{M}_{\mathscr{R}}(\mathcal{T}u, \mathcal{T}v)), \qquad u, v \in \otimes_i \mathbb{C}^{t_i}.$$

Now we construct numerical approximations for the maps just introduced. We may construct an approximation $\tilde{\mathcal{S}} \colon \tilde{\mathbb{C}}_\circ^r \to \tilde{\mathscr{R}}_\circ$ by first using Lemma 2.11 to compute an approximation $\tilde{\zeta}_j \in \tilde{\mathbb{C}}_\circ$ for each $\zeta_j \coloneqq \zeta^j = e^{\pi \mathrm{i} j/r} \in \mathbb{C}_\circ$, and then using Corollary 2.9 to compute an approximation $\tilde{v}_j \in \tilde{\mathbb{C}}_\circ$ for each product $v_j \coloneqq \zeta_j u_j \in \mathbb{C}_\circ$. We obtain $\varepsilon(\tilde{\zeta}_j) < 2$ and then $\varepsilon(\tilde{v}_j) < \varepsilon(\tilde{\zeta}_j) + 2 < 4$. Hence $\varepsilon(\tilde{\mathcal{S}}) < 4$ and $\mathsf{C}(\tilde{\mathcal{S}}) = O(rp^{1+\delta})$. Then, applying $\tilde{\mathcal{S}}$ separately to the coefficient of each $x_1^{j_1} \cdots x_{d-1}^{j_{d-1}}$, we obtain an approximation $\tilde{\mathcal{T}} \colon \otimes_{i=1}^d \tilde{\mathbb{C}}_\circ^{t_i} \to \otimes_{i=1}^{d-1} \tilde{\mathscr{R}}_\circ^{t_i}$ such that $\varepsilon(\tilde{\mathcal{T}}) < 4$ and $\mathsf{C}(\tilde{\mathcal{T}}) = O((T/r)rp^{1+\delta}) = O(Tp^{1+\delta})$. The inverse is handled similarly; we obtain an approximation $\tilde{\mathcal{U}} \colon \otimes_{i=1}^{d-1} \tilde{\mathscr{R}}_\circ^{t_i} \to \otimes_{i=1}^d \tilde{\mathbb{C}}_\circ^{t_i}$ such that $\varepsilon(\tilde{\mathcal{U}}) < 4$ and $\mathsf{C}(\tilde{\mathcal{U}}) = O(Tp^{1+\delta})$.

Finally, given as input $u, v \in \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$, we define $\tilde{\mathcal{M}}_{\mathbb{C}}(u,v) \coloneqq \tilde{\mathcal{U}}(\tilde{\mathcal{M}}_{\mathscr{R}}(\tilde{\mathcal{T}}u, \tilde{\mathcal{T}}v))$. Then Lemma 2.6, Lemma 2.8 and Proposition 3.4 together imply that

$$\varepsilon(\tilde{\mathcal{M}}_{\mathbb{C}}) \leqslant \varepsilon(\tilde{\mathcal{U}}) + \varepsilon(\tilde{\mathcal{M}}_{\mathscr{R}}) + \varepsilon(\tilde{\mathcal{T}}) + \varepsilon(\tilde{\mathcal{T}}) < (3T\log_2 T + 2T + 3) + 4 + 4 + 4,$$

and the estimate for $\mathsf{C}(\tilde{\mathcal{M}}_{\mathbb{C}})$ follows immediately from Proposition 3.4. □

Finally we use Bluestein's trick [2] to prove the main result of this section.

Proof of Theorem 3.1. We are given as input $u \in \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$. We wish to approximate its transform $v \coloneqq \mathcal{F}_{t_1,\dots,t_d} u \in \otimes_i \mathbb{C}_\circ^{t_i}$, which is given explicitly by

$$v_{j_1,\dots,j_d} = \frac{1}{T} \sum_{k_1=0}^{t_1-1} \cdots \sum_{k_d=0}^{t_d-1} e^{-2\pi \mathrm{i}(j_1k_1/t_1 + \cdots + j_dk_d/t_d)} u_{k_1,\dots,k_d}, \qquad 0 \leqslant j_i < t_i.$$

For any $j_1, \dots, j_d \in \mathbb{Z}$ set

$$a_{j_1,\dots,j_d} \coloneqq e^{\pi \mathrm{i}(j_1^2/t_1 + \cdots + j_d^2/t_d)} \in \mathbb{C}_\circ. \tag{3.1}$$

The identity $-2jk = (j-k)^2 - j^2 - k^2$ implies that

$$v_{j_1,\dots,j_d} = \bar{a}_{j_1,\dots,j_d} \frac{1}{T} \sum_{k_1=0}^{t_1-1} \cdots \sum_{k_d=0}^{t_d-1} a_{j_1-k_1,\dots,j_d-k_d} (\bar{a}_{k_1,\dots,k_d} u_{k_1,\dots,k_d}), \tag{3.2}$$

where $\bar{\cdot}$ denotes complex conjugation. Moreover, we observe that $a_{j_1,\dots,j_d}$ is periodic in each j_i with period t_i, as $e^{\pi \mathrm{i}(j_i+t_i)^2/t_i} = e^{\pi \mathrm{i} j_i^2/t_i} (e^{\pi \mathrm{i}})^{2j_i+t_i} = e^{\pi \mathrm{i} j_i^2/t_i}$ (using the fact that t_i is even). Therefore, regarding (3.1) as defining a vector $a \in \otimes_i \mathbb{C}_\circ^{t_i}$, we may rewrite (3.2) in the form

$$v = \bar{a} \cdot (\tfrac{1}{T} a * (\bar{a} \cdot u)).$$

We now use the following algorithm.

(1) *Compute a.* Recalling that each t_i divides r, we may write

$$a_{j_1,\dots,j_d} = e^{2\pi \mathrm{i} \eta_{j_1,\dots,j_d}/2r}, \qquad \eta_{j_1,\dots,j_d} \coloneqq \frac{r}{t_1} j_1^2 + \cdots + \frac{r}{t_d} j_d^2 \pmod{2r}.$$

Iterating over the tuples $(j_1, \dots, j_d)$ in lexicographical order, we may compute $\eta_{j_1,\dots,j_d}$ in amortised time $O(\log r) = O(p)$ per tuple (for example by precomputing a table of squares modulo $2r$), and then use Lemma 2.11 to compute an approximation $\tilde{a}_{j_1,\dots,j_d} \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{a}_{j_1,\dots,j_d}) < 2$ in time $O(p^{1+\delta})$. We thus obtain $\tilde{a} \in \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ with $\varepsilon(\tilde{a}) < 2$ in time $O(Tp^{1+\delta})$.

(2) *Pre-multiply.* We use Corollary 2.9 to compute an approximation $\tilde{b} \in \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ for $b \coloneqq \bar{a} \cdot u$ with $\varepsilon(\tilde{b}) < \varepsilon(\tilde{a}) + 2 < 4$ in time $O(Tp^{1+\delta})$.

(3) *Convolution.* We use Proposition 3.5 to compute an approximation $\tilde{c} \in \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ for $c \coloneqq \frac{1}{T} a * b$. This requires time $(4T/r)\,\mathsf{M}(3rp) + O(Tp\log T + Tp^{1+\delta})$, and by Lemma 2.8 we have

$$\varepsilon(\tilde{c}) \leqslant \varepsilon(\tilde{\mathcal{M}}_{\mathbb{C}}) + \varepsilon(\tilde{a}) + \varepsilon(\tilde{b}) < 3T\log_2 T + 2T + 21.$$

(4) *Post-multiply.* We invoke Corollary 2.9 again to compute an approximation $\tilde{v} \in \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ for $v = \bar{a} \cdot c$ such that

$$\varepsilon(\tilde{v}) \leqslant \varepsilon(\tilde{a}) + \varepsilon(\tilde{c}) + 2 < 3T\log_2 T + 2T + 25$$

in time $O(Tp^{1+\delta})$. We have $2T + 25 < 5T\log_2 T$ (because $T \geqslant t_1 t_2 \geqslant 4$), so $\varepsilon(\tilde{v}) < 8T\log_2 T$. Finally we take $\tilde{\mathcal{F}}_{t_1,\dots,t_d} u \coloneqq \tilde{v}$. □

Remark 3.6. In the algorithm developed in this section, it is essential that the multidimensional complex transform be performed "all at once". If instead we decompose it into one-dimensional transforms in the usual way, and then use Bluestein's method to convert each of these to a one-dimensional convolution over $\mathscr{R}$, we would increase the number of multiplications in $\mathscr{R}$ by a factor of $O(d)$, leading to an extraneous factor of $O(d)$ on the right hand side of (1.3).

Remark 3.7. An alternative method for reducing multidimensional complex transforms to synthetic transforms was described in [33]. Briefly, given as input $u \in \mathbb{C}[x_1, \ldots, x_{d-1}, y]/(x_1^{t_1} - 1, \ldots, x_{d-1}^{t_{d-1}} - 1, y^r + 1)$, assume that we wish to evaluate each x_i at the complex t_i-th roots of unity, and y at the "odd" complex $2r$-th roots of unity (i.e., roots of $y^r + 1$). We first use $(d-1)$-dimensional synthetic transforms to compute the polynomials $u_{i_1,\ldots,i_{d-1}}(y) \coloneqq u(y^{2ri_1/t_1}, \ldots, y^{2ri_{d-1}/t_{d-1}}, y) \in \mathbb{C}[y]/(y^r + 1)$, for all $i_k \in \{0, \ldots, t_k - 1\}$. It then suffices to compute the one-dimensional complex DFT of each $u_{i_1,\ldots,i_{d-1}}(y)$, which could be done for example by Bluestein's method.

4. Gaussian resampling

Let $p \geqslant 100$ be the working precision as defined in Section 2. The aim of this section is to show how to reduce the problem of approximating a multidimensional complex transform $\mathcal{F}_{s_1,\ldots,s_d} \colon \otimes_i \mathbb{C}^{s_i} \to \otimes_i \mathbb{C}^{s_i}$, for a given "source" size $s_1 \times \cdots \times s_d$, to the problem of approximating another transform $\mathcal{F}_{t_1,\ldots,t_d} \colon \otimes_i \mathbb{C}^{t_i} \to \otimes_i \mathbb{C}^{t_i}$ for a somewhat larger "target" size $t_1 \times \cdots \times t_d$. (In Section 5 we will specialise to the case that the s_i are primes and the t_i are powers of two.) The following theorem is proved at the end of Section 4.3. It may be strengthened in various ways; see the discussion in Section 4.4.

Theorem 4.1 (Gaussian resampling)**.** *Let $d \geqslant 1$, let $s_1, \ldots, s_d$ and $t_1, \ldots, t_d$ be integers such that $2 \leqslant s_i < t_i < 2^p$ and $\gcd(s_i, t_i) = 1$, and let $T \coloneqq t_1 \cdots t_d$. Let α be an integer in the interval $2 \leqslant \alpha < p^{1/2}$. For each i, let $\theta_i \coloneqq t_i/s_i - 1$, and assume that $\theta_i \geqslant p/\alpha^4$.*

Then there exist linear maps $\mathcal{A} \colon \otimes_i \mathbb{C}^{s_i} \to \otimes_i \mathbb{C}^{t_i}$ and $\mathcal{B} \colon \otimes_i \mathbb{C}^{t_i} \to \otimes_i \mathbb{C}^{s_i}$, with $\|\mathcal{A}\|, \|\mathcal{B}\| \leqslant 1$, such that

$$\mathcal{F}_{s_1,\ldots,s_d} = 2^\gamma \mathcal{B} \mathcal{F}_{t_1,\ldots,t_d} \mathcal{A}, \qquad \gamma \coloneqq 2d\alpha^2.$$

Moreover, we may construct numerical approximations $\tilde{\mathcal{A}} \colon \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ and $\tilde{\mathcal{B}} \colon \otimes_i \tilde{\mathbb{C}}_\circ^{t_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$ such that $\varepsilon(\tilde{\mathcal{A}}), \varepsilon(\tilde{\mathcal{B}}) < dp^2$ and

$$\mathsf{C}(\tilde{\mathcal{A}}), \mathsf{C}(\tilde{\mathcal{B}}) = O(dTp^{3/2+\delta}\alpha + Tp \log T).$$

This theorem shows that to approximate a transform of size $s_1 \times \cdots \times s_d$, one may first apply $\tilde{\mathcal{A}}$, then compute a transform of size $t_1 \times \cdots \times t_d$, then apply $\tilde{\mathcal{B}}$, and finally multiply by 2^γ. The $dTp^{3/2+\delta}\alpha$ term in the complexity bound arises from the numerical computations at the heart of the "Gaussian resampling" method. The $Tp \log T$ term covers the cost of various data rearrangements in the Turing model (this term would not appear if we worked in the Boolean circuit model). In the application in Section 5, the parameters will be chosen so that the first term is negligible compared to the $O(Tp \log T)$ cost of evaluating $\tilde{\mathcal{F}}_{t_1,\ldots,t_d}$.

0.5000	0.2280	0.0216	4.2e-4	1.7e-6	2.9e-9	1.7e-6	4.2e-4	0.0216	0.2280
0.3142	**0.4795**	0.1522	0.0100	1.3e-4	3.9e-7	8.9e-9	7.1e-6	0.0012	0.0428
0.0779	0.3982	0.4230	0.0934	0.0043	4.0e-5	8.1e-8	4.7e-8	2.6e-5	0.0032
0.0076	0.1305	**0.4642**	0.3432	0.0527	0.0017	1.1e-5	1.5e-8	2.3e-7	9.2e-5
2.9e-4	0.0169	0.2011	**0.4977**	0.2561	0.0274	6.0e-4	2.8e-6	3.5e-9	1.0e-6
4.4e-6	8.6e-4	0.0344	0.2849	**0.4908**	0.1757	0.0131	2.0e-4	6.5e-7	5.3e-9
2.7e-8	1.7e-5	0.0023	0.0644	0.3714	**0.4452**	0.1109	0.0057	6.1e-5	1.3e-7
2.7e-8	1.3e-7	6.1e-5	0.0057	0.1109	**0.4452**	0.3714	0.0644	0.0023	1.7e-5
4.4e-6	5.3e-9	6.5e-7	2.0e-4	0.0131	0.1757	**0.4908**	0.2849	0.0344	8.6e-4
2.9e-4	1.0e-6	3.5e-9	2.8e-6	6.0e-4	0.0274	0.2561	**0.4977**	0.2011	0.0169
0.0076	9.2e-5	2.3e-7	1.5e-8	1.1e-5	0.0017	0.0527	0.3432	**0.4642**	0.1305
0.0779	0.0032	2.6e-5	4.7e-8	8.1e-8	4.0e-5	0.0043	0.0934	0.4230	0.3982
0.3142	0.0428	0.0012	7.1e-6	8.9e-9	3.9e-7	1.3e-4	0.0100	0.1522	**0.4795**

1.0000	5.9e-10	1.2e-37	9.8e-84	2.6e-148	5.2e-231	2.6e-148	9.8e-84	1.2e-37	5.9e-10
3.4e-6	**0.3227**	1.0e-14	1.2e-46	5.3e-97	8.1e-166	1.6e-210	9.2e-132	1.8e-71	1.3e-29
1.4e-22	0.0021	0.0108	1.9e-20	1.3e-56	3.0e-111	2.5e-184	1.0e-190	3.3e-116	3.6e-60
7.6e-50	1.6e-16	**0.1339**	3.7e-5	3.8e-27	1.3e-67	1.8e-126	8.4e-204	7.1e-172	1.2e-101
4.7e-88	1.6e-40	2.0e-11	**0.8819**	1.3e-8	7.7e-35	1.5e-79	1.1e-142	2.8e-224	4.9e-154
3.6e-137	1.9e-75	3.6e-32	2.4e-7	**0.6049**	5.2e-13	1.6e-43	1.8e-92	7.2e-160	2.4e-217
3.3e-197	2.7e-121	8.1e-64	8.5e-25	3.2e-4	**0.0432**	2.0e-18	3.5e-53	2.2e-106	4.8e-178
3.3e-197	4.8e-178	2.2e-106	3.5e-53	2.0e-18	**0.0432**	3.2e-4	8.5e-25	8.1e-64	2.7e-121
3.6e-137	2.4e-217	7.2e-160	1.8e-92	1.6e-43	5.2e-13	**0.6049**	2.4e-7	3.6e-32	1.9e-75
4.7e-88	4.9e-154	2.8e-224	1.1e-142	1.5e-79	7.7e-35	1.3e-8	**0.8819**	2.0e-11	1.6e-40
7.6e-50	1.2e-101	7.1e-172	8.4e-204	1.8e-126	1.3e-67	3.8e-27	3.7e-5	**0.1339**	1.6e-16
1.4e-22	3.6e-60	3.3e-116	1.0e-190	2.5e-184	3.0e-111	1.3e-56	1.9e-20	0.0108	0.0021
3.4e-6	1.3e-29	1.8e-71	9.2e-132	1.6e-210	8.1e-166	5.3e-97	1.2e-46	1.0e-14	**0.3227**

FIGURE 2. Matrices of $\mathcal{S}$ and $\mathcal{T}$ for $s = 10$, $t = 13$, $\alpha = 2$. The maximal entries in each column are shown in bold.

Throughout this section we use the notation

$$[x] \coloneqq \lfloor x + \tfrac{1}{2} \rfloor, \qquad \langle x \rangle \coloneqq x - [x], \qquad x \in \mathbb{R}.$$

Thus $[x]$ is the nearest integer to x, rounding upwards in case of a tie, and $\langle x \rangle$ is the corresponding fractional part with $-\frac{1}{2} \leqslant \langle x \rangle < \frac{1}{2}$.

4.1. **The resampling identity.** Throughout Sections 4.1 and 4.2, let s and $t > s$ be positive integers such that $\gcd(s, t) = 1$, and let $\alpha \in (0, \infty)$.

Define "resampling maps" $\mathcal{S} \colon \mathbb{C}^s \to \mathbb{C}^t$ and $\mathcal{T} \colon \mathbb{C}^s \to \mathbb{C}^t$ by

$$(\mathcal{S}u)_k \coloneqq \alpha^{-1} \sum_{j \in \mathbb{Z}} e^{-\pi \alpha^{-2} s^2 (\frac{k}{t} - \frac{j}{s})^2} u_j, \qquad u \in \mathbb{C}^s, \quad 0 \leqslant k < t,$$

$$(\mathcal{T}u)_k \coloneqq \sum_{j \in \mathbb{Z}} e^{-\pi \alpha^2 t^2 (\frac{k}{t} - \frac{j}{s})^2} u_j, \qquad u \in \mathbb{C}^s, \quad 0 \leqslant k < t.$$

These sums certainly converge due to the rapid decay of the function e^{-x^2}. Each entry $(\mathcal{S}u)_k$ and $(\mathcal{T}u)_k$ is a weighted linear combination of $u_0, \ldots, u_{s-1}$, with the largest weightings given to those u_j for which j/s is closest to k/t modulo 1. Figure 2 shows examples of the matrices of $\mathcal{S}$ and $\mathcal{T}$. They have relatively large entries near the "diagonal" of slope t/s, and the entries decay rapidly away from the diagonal according to a Gaussian law. The parameter α controls the rate of decay.

We also define permutation maps $\mathcal{P}_s \colon \mathbb{C}^s \to \mathbb{C}^s$ and $\mathcal{P}_t \colon \mathbb{C}^t \to \mathbb{C}^t$ by

$$(\mathcal{P}_s u)_j \coloneqq u_{tj}, \qquad u \in \mathbb{C}^s, \quad 0 \leqslant j < s,$$

$$(\mathcal{P}_t u)_k \coloneqq u_{-sk}, \qquad u \in \mathbb{C}^t, \quad 0 \leqslant k < t.$$

Then we have the following fundamental identity, which uses $\mathcal{S}$ and $\mathcal{T}$ to transform $\mathcal{F}_s$ into $\mathcal{F}_t$.

Theorem 4.2 (Resampling identity)**.** *We have $\mathcal{T}\mathcal{P}_s\mathcal{F}_s = \mathcal{P}_t\mathcal{F}_t\mathcal{S}$. In other words, the following diagram commutes:*

$$\begin{array}{ccccc} \mathbb{C}^s & \xrightarrow{\mathcal{F}_s} & \mathbb{C}^s & \xrightarrow{\mathcal{P}_s} & \mathbb{C}^s \\ \Big\downarrow{\scriptstyle \mathcal{S}} & & & & \Big\downarrow{\scriptstyle \mathcal{T}} \\ \mathbb{C}^t & \xrightarrow{\mathcal{F}_t} & \mathbb{C}^t & \xrightarrow{\mathcal{P}_t} & \mathbb{C}^t \end{array}$$

Proof. Given $u \in \mathbb{C}^s$, define a smooth, 1-periodic function $f_u \colon \mathbb{R} \to \mathbb{C}$ by

$$f_u(x) \coloneqq \sum_{j \in \mathbb{Z}} u_j g(x - \tfrac{j}{s}), \qquad g(x) \coloneqq e^{-\pi \alpha^{-2} s^2 x^2}.$$

It has an absolutely and uniformly convergent Fourier expansion

$$f_u(x) = \sum_{r \in \mathbb{Z}} \hat{f}_u(r) e^{2\pi \mathrm{i} r x},$$

where the Fourier coefficients are given by

$$\begin{aligned} \hat{f}_u(r) &= \int_0^1 e^{-2\pi \mathrm{i} r x} f_u(x)\, dx \\ &= \int_0^1 \sum_{j \in \mathbb{Z}} u_j e^{-2\pi \mathrm{i} r x} g(x - \tfrac{j}{s})\, dx \\ &= \int_0^1 \sum_{j=0}^{s-1} \sum_{q \in \mathbb{Z}} u_j e^{-2\pi \mathrm{i} r x} g(x - q - \tfrac{j}{s})\, dx \\ &= \sum_{j=0}^{s-1} u_j \int_{-\infty}^{\infty} e^{-2\pi \mathrm{i} r x} g(x - \tfrac{j}{s})\, dx \\ &= \sum_{j=0}^{s-1} u_j e^{-2\pi \mathrm{i} r j/s} \int_{-\infty}^{\infty} e^{-2\pi \mathrm{i} r x} g(x)\, dx. \end{aligned}$$

Using the well-known fact that the Fourier transform of $g(x)$ on $\mathbb{R}$ is given by

$$\int_{-\infty}^{\infty} e^{-2\pi \mathrm{i} y x} g(x)\, dx = \alpha s^{-1} e^{-\pi \alpha^2 s^{-2} y^2}, \qquad y \in \mathbb{R},$$

we obtain

$$\hat{f}_u(r) = \alpha e^{-\pi \alpha^2 s^{-2} r^2} (\mathcal{F}_s u)_r, \qquad r \in \mathbb{Z}.$$

By definition $(\mathcal{S}u)_\ell = \alpha^{-1} f_u(\ell/t)$ for any $\ell \in \mathbb{Z}$, so for any $k \in \{0, \dots, t-1\}$ we have

$$\begin{aligned} (\mathcal{P}_t \mathcal{F}_t \mathcal{S} u)_k &= (\mathcal{F}_t \mathcal{S} u)_{-sk} \\ &= \alpha^{-1} t^{-1} \sum_{\ell=0}^{t-1} e^{2\pi \mathrm{i} s k \ell/t} f_u(\ell/t) \\ &= \alpha^{-1} t^{-1} \sum_{\ell=0}^{t-1} e^{2\pi \mathrm{i} s k \ell/t} \sum_{r \in \mathbb{Z}} \hat{f}_u(r) e^{2\pi \mathrm{i} r \ell/t} \end{aligned}$$

$$\begin{aligned}
&= \alpha^{-1} \sum_{r=-sk \bmod t} \hat{f}_u(r) \\
&= \sum_{r=-sk \bmod t} e^{-\pi\alpha^2 s^{-2} r^2} (\mathcal{F}_s u)_r \\
&= \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^2 s^{-2}(tj-sk)^2} (\mathcal{F}_s u)_{tj-sk} \\
&= \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^2 t^2 (\frac{j}{s}-\frac{k}{t})^2} (\mathcal{P}_s \mathcal{F}_s u)_j \\
&= (\mathcal{T}\mathcal{P}_s \mathcal{F}_s u)_k. \qquad \square
\end{aligned}$$

Remark 4.3. Another interpretation of the above proof is that the measure $f_u(x)\,dx$ is the convolution of the measures $\sum_{j=0}^{s-1} u_j \delta_{j/s}$ and $\sum_{j\in\mathbb{Z}} g(x-j)\,dx$ on $\mathbb{R}/\mathbb{Z}$, where δ_x means a unit mass concentrated at x. The key point is that the Fourier transform maps convolution of measures to pointwise multiplication of their Fourier coefficients.

We conclude this section with a straightforward bound for $\|\mathcal{S}\|$.

Lemma 4.4. *We have* $\|\mathcal{S}\| < 1 + \alpha^{-1}$.

Proof. For any $u \in \mathbb{C}_\circ^s$ and $k \in \{0, \dots, t-1\}$, we have

$$|(\mathcal{S}u)_k| \leqslant \alpha^{-1} \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^{-2} s^2 (\frac{k}{t}-\frac{j}{s})^2} = \alpha^{-1} \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^{-2}(j-\frac{sk}{t})^2} = \alpha^{-1} \sum_{j\in\mathbb{Z}} G(\eta + j),$$

where $\eta \coloneqq \langle \frac{-sk}{t} \rangle \in [-\frac{1}{2}, \frac{1}{2}]$ and $G(x) \coloneqq e^{-\pi\alpha^{-2}x^2}$.

First suppose that $\eta \in [-\frac{1}{2}, 0]$. Then $G(x)$ is increasing on $(-\infty, \eta)$ and decreasing on $(\eta+1, \infty)$, so

$$\int_{-\infty}^{\eta} G(x)\,dx > \sum_{j=-\infty}^{-1} G(\eta + j), \qquad \int_{\eta+1}^{\infty} G(x)\,dx > \sum_{j=2}^{\infty} G(\eta + j).$$

For the remaining interval $(\eta, \eta+1)$, we observe that $G(x) \geqslant G(\eta+1)$ for $x \in (0, \eta+1)$ and $G(x) \geqslant G(\eta)$ for $x \in (\eta, 0)$; but we have additionally $G(\eta) \geqslant G(\eta+1)$ because $|\eta| \leqslant \frac{1}{2} \leqslant |\eta+1|$, so in fact $G(x) \geqslant G(\eta+1)$ on the whole interval $(\eta, \eta+1)$. This implies that $\int_{\eta}^{\eta+1} G(x)\,dx \geqslant G(\eta+1)$, and adding the three integrals yields

$$\sum_{j\in\mathbb{Z}} G(\eta + j) = G(\eta) + \sum_{j\neq 0} G(\eta + j) < 1 + \int_{-\infty}^{\infty} G(x)\,dx = 1 + \alpha.$$

A symmetrical argument yields the same bound for the case $\eta \in [0, \frac{1}{2}]$. We conclude that $|(\mathcal{S}u)_k| < \alpha^{-1}(1+\alpha) = 1 + \alpha^{-1}$, and hence $\|\mathcal{S}\| < 1 + \alpha^{-1}$. $\square$

4.2. **Solving the system.** We wish to use Theorem 4.2 to express $\mathcal{F}_s$ in terms of $\mathcal{F}_t$. To do this, we must show how to solve a system of the form $\mathcal{T}x = y$. This system is overdetermined, as $t > s$. For fixed α, it turns out that the system is numerically unstable if t/s is too close to 1, or in other words, if the quantity $\theta \coloneqq t/s - 1$ is too close to zero. On the other hand, we will show that imposing the condition $\theta \geqslant 1/\alpha^2$ is enough to ensure that the system becomes numerically tractable, and in this case we may even construct an explicit left inverse for $\mathcal{T}$.

$$\begin{bmatrix}
9.6\text{e-}923 & 1.8\text{e-}9 & 9.5\text{e-}37 & 1.1\text{e-}83 & 4.4\text{e-}148 & 1.2\text{e-}229 & 4.4\text{e-}148 & 1.1\text{e-}83 & 9.5\text{e-}37 & 1.8\text{e-}9 \\
3.4\text{e-}6 & 1.7\text{e-}880 & 7.9\text{e-}14 & 1.4\text{e-}46 & 8.8\text{e-}97 & 1.8\text{e-}164 & 2.7\text{e-}210 & 1.0\text{e-}131 & 1.3\text{e-}70 & 4.1\text{e-}29 \\
7.6\text{e-}50 & 5.2\text{e-}16 & 2.7\text{e-}866 & 4.3\text{e-}5 & 6.3\text{e-}27 & 3.2\text{e-}66 & 3.0\text{e-}126 & 9.5\text{e-}204 & 5.3\text{e-}171 & 3.8\text{e-}101 \\
4.7\text{e-}88 & 5.0\text{e-}40 & 1.5\text{e-}10 & 7.4\text{e-}909 & 2.2\text{e-}8 & 1.7\text{e-}33 & 2.5\text{e-}79 & 1.2\text{e-}142 & 2.1\text{e-}223 & 1.5\text{e-}153 \\
3.6\text{e-}137 & 5.9\text{e-}75 & 2.7\text{e-}31 & 2.8\text{e-}7 & 1.1\text{e-}894 & 1.2\text{e-}11 & 2.7\text{e-}43 & 2.0\text{e-}92 & 5.3\text{e-}159 & 7.6\text{e-}217 \\
3.3\text{e-}197 & 1.5\text{e-}177 & 1.6\text{e-}105 & 4.0\text{e-}53 & 3.4\text{e-}18 & 4.2\text{e-}852 & 5.3\text{e-}4 & 9.7\text{e-}25 & 6.0\text{e-}63 & 8.6\text{e-}121 \\
3.6\text{e-}137 & 7.6\text{e-}217 & 5.3\text{e-}159 & 2.0\text{e-}92 & 2.7\text{e-}43 & 1.2\text{e-}11 & 1.1\text{e-}894 & 2.8\text{e-}7 & 2.7\text{e-}31 & 5.9\text{e-}75 \\
4.7\text{e-}88 & 1.5\text{e-}153 & 2.1\text{e-}223 & 1.2\text{e-}142 & 2.5\text{e-}79 & 1.7\text{e-}33 & 2.2\text{e-}8 & 7.4\text{e-}909 & 1.5\text{e-}10 & 5.0\text{e-}40 \\
7.6\text{e-}50 & 3.8\text{e-}101 & 5.3\text{e-}171 & 9.5\text{e-}204 & 3.0\text{e-}126 & 3.2\text{e-}66 & 6.3\text{e-}27 & 4.3\text{e-}5 & 2.7\text{e-}866 & 5.2\text{e-}16 \\
3.4\text{e-}6 & 4.1\text{e-}29 & 1.3\text{e-}70 & 1.0\text{e-}131 & 2.7\text{e-}210 & 1.8\text{e-}164 & 8.8\text{e-}97 & 1.4\text{e-}46 & 7.9\text{e-}14 & 1.7\text{e-}880
\end{bmatrix}$$

FIGURE 3. Matrix of $\mathcal{E}$ for $s = 10$, $t = 13$, $\alpha = 2$.

The function $\ell \mapsto [t\ell/s]$ maps $\{0, \dots, s-1\}$ (injectively) into $\{0, \dots, t-1\}$, so we may define a map $\mathcal{C}\colon \mathbb{C}^t \to \mathbb{C}^s$ by the formula

$$(\mathcal{C}u)_\ell \coloneqq u_{[t\ell/s]}, \qquad u \in \mathbb{C}^t, \quad 0 \leqslant \ell < s.$$

We then set

$$\mathcal{T}' \coloneqq \mathcal{C}\mathcal{T}\colon \mathbb{C}^s \to \mathbb{C}^s.$$

Note that the matrix of $\mathcal{T}'$ is obtained by deleting $t - s$ rows from the matrix of $\mathcal{T}$. If we can show that $\mathcal{T}'$ is invertible, then a left inverse for $\mathcal{T}$ is given by $(\mathcal{T}')^{-1}\mathcal{C}$.

The entries of $\mathcal{T}'u$ are given explicitly by

$$\begin{aligned}(\mathcal{T}'u)_\ell = (\mathcal{T}u)_{[t\ell/s]} = \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^2 t^2 (\frac{1}{t}\left[\frac{t\ell}{s}\right] - \frac{j}{s})^2} u_j &= \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^2 (\frac{tj}{s} - \left[\frac{t\ell}{s}\right])^2} u_j \\ &= \sum_{h\in\mathbb{Z}} e^{-\pi\alpha^2 (\frac{th}{s} + \beta_\ell)^2} u_{\ell+h},\end{aligned}$$

where

$$\beta_\ell \coloneqq \tfrac{t\ell}{s} - \left[\tfrac{t\ell}{s}\right] = \left\langle \tfrac{t\ell}{s} \right\rangle, \qquad \ell \in \mathbb{Z}.$$

Observe that β_ℓ depends only on ℓ modulo s, and that $|\beta_\ell| \leqslant \frac{1}{2}$ for all ℓ.

We normalise $\mathcal{T}'$ as follows. Let $\mathcal{D}\colon \mathbb{C}^s \to \mathbb{C}^s$ be the diagonal map defined by $(\mathcal{D}u)_\ell \coloneqq d_\ell u_\ell$, where $d_\ell \coloneqq e^{\pi\alpha^2\beta_\ell^2}$ for $\ell \in \mathbb{Z}$. Since $\beta_\ell^2 \leqslant \frac{1}{4}$ we have $1 \leqslant d_\ell \leqslant e^{\pi\alpha^2/4}$, and in particular

$$\|\mathcal{D}\| \leqslant e^{\pi\alpha^2/4}. \tag{4.1}$$

Define

$$\mathcal{N} \coloneqq \mathcal{T}'\mathcal{D}\colon \mathbb{C}^s \to \mathbb{C}^s.$$

In other words, the matrix of $\mathcal{N}$ is obtained by multiplying the ℓ-th column of the matrix of $\mathcal{T}'$ by d_ℓ. Explicitly,

$$(\mathcal{N}u)_\ell = \sum_{h\in\mathbb{Z}} e^{-\pi\alpha^2(\frac{th}{s}+\beta_\ell)^2} d_{\ell+h} u_{\ell+h} = \sum_{h\in\mathbb{Z}} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2)} u_{\ell+h}.$$

In this last expression, the $h = 0$ term is simply u_ℓ. Therefore, setting $\mathcal{E} \coloneqq \mathcal{N} - \mathcal{I}$, where $\mathcal{I}\colon \mathbb{C}^s \to \mathbb{C}^s$ is the identity map, we have

$$(\mathcal{E}u)_\ell = \sum_{h\neq 0} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2)} u_{\ell+h}, \qquad u \in \mathbb{C}^s, \quad 0 \leqslant \ell < s.$$

An example of the matrix of $\mathcal{E}$ is shown in Figure 3.

The following estimate is crucial for establishing left-invertibility of $\mathcal{T}$ and for obtaining a fast algorithm for solving the system $\mathcal{T}x = y$.

Lemma 4.5. *Assume that $\alpha^2\theta \geqslant 1$. Then*

$$\|\mathcal{E}\| < 2.01 \cdot e^{-\pi\alpha^2\theta/2} < 2^{-\alpha^2\theta}.$$

Proof. For any $u \in \mathbb{C}_\circ^s$, the above formula for $(\mathcal{E}u)_\ell$ implies that

$$|(\mathcal{E}u)_\ell| \leqslant \sum_{h\neq 0} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2-\beta_{\ell+h}^2)}, \qquad 0 \leqslant \ell < s.$$

Since $|\beta_\ell| \leqslant \frac{1}{2} < \frac{t}{2s}$, we have $|\frac{th}{s}+\beta_\ell| \geqslant |\frac{th}{s}| - |\beta_\ell| > \frac{t}{s}(|h|-\frac{1}{2})$. For $h \neq 0$ we have $|h|-\frac{1}{2} \geqslant \frac{1}{2} > 0$, so

$$\begin{aligned}(\tfrac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2 &> (t/s)^2(|h|-\tfrac{1}{2})^2 - \tfrac{1}{4} \\ &= (1+\theta)^2(|h|-\tfrac{1}{2})^2 - \tfrac{1}{4} \\ &\geqslant (1+2\theta)(|h|-\tfrac{1}{2})^2 - \tfrac{1}{4} \\ &= (|h|-\tfrac{1}{2})^2 - \tfrac{1}{4} + 2\theta(|h|-\tfrac{1}{2})^2 \\ &\geqslant 2\theta(|h|-\tfrac{1}{2})^2.\end{aligned}$$

Therefore

$$|(\mathcal{E}u)_\ell| < \sum_{h\neq 0} e^{-2\pi\alpha^2\theta(|h|-\frac{1}{2})^2} = 2(w^{1/4} + w^{9/4} + w^{25/4} + \cdots)$$

where $w \coloneqq e^{-2\pi\alpha^2\theta}$. Since $\alpha^2\theta \geqslant 1$ we have $w \leqslant e^{-2\pi} < 0.002$, so

$$|(\mathcal{E}u)_\ell| < 2w^{1/4}(1 + w^2 + w^6 + \cdots) < 2.01 \cdot w^{1/4} = 2.01 \cdot e^{-\pi\alpha^2\theta/2} < 2^{-\alpha^2\theta},$$

where we have used the fact that $2.01 \cdot e^{-\pi x/2} < 2^{-x}$ for all $x \geqslant 1$. □

Under the hypothesis of Lemma 4.5, we see that $\|\mathcal{E}\| < \frac{1}{2}$, so $\mathcal{N}$ is invertible, with inverse given by $\mathcal{N}^{-1} = \mathcal{I} - \mathcal{E} + \mathcal{E}^2 - \cdots$. Moreover, $\mathcal{D}\mathcal{N}^{-1}\mathcal{C}$ is the promised left inverse for $\mathcal{T}$, as

$$(\mathcal{D}\mathcal{N}^{-1}\mathcal{C})\mathcal{T} = \mathcal{D}(\mathcal{T}'\mathcal{D})^{-1}\mathcal{T}' = \mathcal{I}. \tag{4.2}$$

4.3. **Proof of the theorem.** We may now prove the following special case of Theorem 4.1.

Proposition 4.6 (Gaussian resampling in one dimension)**.** *Let s and t be integers such that $2 \leqslant s < t < 2^p$ and $\gcd(s,t) = 1$. Let α be an integer in the interval $2 \leqslant \alpha < p^{1/2}$. Let $\theta \coloneqq t/s - 1 > 0$, and assume that $\theta \geqslant p/\alpha^4$. Then:*

(i) *There exist linear maps $\mathcal{A}\colon \mathbb{C}^s \to \mathbb{C}^t$ and $\mathcal{B}\colon \mathbb{C}^t \to \mathbb{C}^s$ with $\|\mathcal{A}\|, \|\mathcal{B}\| \leqslant 1$ such that $\mathcal{F}_s = 2^{2\alpha^2}\mathcal{B}\mathcal{F}_t\mathcal{A}$.*

(ii) *We may construct numerical approximations $\tilde{\mathcal{A}}\colon \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^t$ and $\tilde{\mathcal{B}}\colon \tilde{\mathbb{C}}_\circ^t \to \tilde{\mathbb{C}}_\circ^s$ such that $\varepsilon(\tilde{\mathcal{A}}), \varepsilon(\tilde{\mathcal{B}}) < p^2$ and $\mathsf{C}(\tilde{\mathcal{A}}), \mathsf{C}(\tilde{\mathcal{B}}) = O(tp^{3/2+\delta}\alpha + tp\log t)$.*

Proof of (i). We apply the results of Sections 4.1–4.2 with the given s, t and α. Lemma 4.4 implies that $\|\mathcal{S}\| < \frac{3}{2}$. The hypotheses $\alpha < p^{1/2}$ and $\theta \geqslant p/\alpha^4$ imply that $\alpha^2\theta \geqslant 1$, so Lemma 4.5 yields $\|\mathcal{E}\| < 2.01 \cdot e^{-\pi/2} < 0.42$. In particular, $\mathcal{N} = \mathcal{I} + \mathcal{E}$ is invertible and $\|\mathcal{N}^{-1}\| < 1 + 0.42 + (0.42)^2 + \cdots < \frac{7}{4}$.

Let $\mathcal{J} \coloneqq \mathcal{N}^{-1}$, and define normalised maps

$$\mathcal{S}' \coloneqq \mathcal{S}/2, \qquad \mathcal{J}' \coloneqq \mathcal{J}/2, \qquad \mathcal{D}' \coloneqq \mathcal{D}/2^{2\alpha^2-2}.$$

Then $\|\mathcal{S}'\| < \frac{3}{4} < 1$ and $\|\mathcal{J}'\| < \frac{7}{8} < 1$. By (4.1) we have $\|\mathcal{D}\| \leqslant e^{\pi\alpha^2/4} < 2^{1.14\alpha^2} < 2^{2\alpha^2-2}$, as $1.14x < 2x - 2$ for all $x \geqslant 4$; hence also $\|\mathcal{D}'\| < 1$.

Now define

$$\mathcal{A} \coloneqq \mathcal{S}', \qquad \mathcal{B} \coloneqq \mathcal{P}_s^{-1}\mathcal{D}'\mathcal{J}'\mathcal{C}\mathcal{P}_t.$$

It is clear that $\|\mathcal{P}_t\| = \|\mathcal{P}_s^{-1}\| = \|\mathcal{C}\| = 1$, so $\|\mathcal{A}\| < 1$ and $\|\mathcal{B}\| < 1$. Moreover, by (4.2) and Theorem 4.2 we have

$$\begin{aligned} 2^{2\alpha^2}\mathcal{B}\mathcal{F}_t\mathcal{A} &= \mathcal{P}_s^{-1}(2^{2\alpha^2-2}\mathcal{D}')(2\mathcal{J}')\mathcal{C}\mathcal{P}_t\mathcal{F}_t(2\mathcal{S}') \\ &= \mathcal{P}_s^{-1}(\mathcal{D}\mathcal{N}^{-1}\mathcal{C})(\mathcal{P}_t\mathcal{F}_t\mathcal{S}) \\ &= \mathcal{P}_s^{-1}(\mathcal{D}\mathcal{N}^{-1}\mathcal{C})(\mathcal{T}\mathcal{P}_s\mathcal{F}_s) = \mathcal{P}_s^{-1}\mathcal{P}_s\mathcal{F}_s = \mathcal{F}_s. \end{aligned}$$

□

We break up the proof of (ii) into several lemmas. We begin with a straightforward algorithm for approximating $\mathcal{D}'$ (Lemma 4.7). Next we give algorithms for approximating $\mathcal{S}' = \mathcal{S}/2$ and $\mathcal{E}$ (Lemmas 4.8 and 4.10); these amount to merely evaluating sufficiently many terms of the defining series, which converge quickly thanks to the rapid decay of the Gaussian weights. We then give an algorithm for approximating $\mathcal{J}' = \mathcal{N}^{-1}/2$, using the series $\mathcal{N}^{-1} = \mathcal{I} - \mathcal{E} + \mathcal{E}^2 - \cdots$ (Lemma 4.11); here the fast convergence is guaranteed by the bound on $\|\mathcal{E}\|$ given in Lemma 4.5.

Lemma 4.7. *We may construct a numerical approximation $\tilde{\mathcal{D}}' \colon \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^s$ for $\mathcal{D}'$ such that $\varepsilon(\tilde{\mathcal{D}}') < 4$ and $\mathsf{C}(\tilde{\mathcal{D}}') = O(tp^{1+\delta})$.*

Proof. We are given as input $u \in \tilde{\mathbb{C}}_\circ^s$. For each $\ell \in \{0, \dots, s-1\}$, by definition $(\mathcal{D}'u)_\ell = d'_\ell u_\ell$ where

$$d'_\ell \coloneqq e^{\pi\alpha^2\beta_\ell^2}/2^{2\alpha^2-2} = d_\ell/2^{2\alpha^2-2} \leqslant 1.$$

We may rewrite the rational part of the exponent of d'_ℓ as $\alpha^2\beta_\ell^2 = \alpha^2\langle\frac{t\ell}{s}\rangle^2 = \alpha^2 k_\ell/s^2$ for some integer k_ℓ. As α, s, t and ℓ are all integers with $O(p)$ bits (here we have used the hypotheses $s, t < 2^p$ and $\alpha < p^{1/2}$), we may compute $\alpha^2 k_\ell$ and s^2 in $O(p^{1+\delta})$ bit operations. Feeding this as input to Lemma 2.13 (with $\sigma \coloneqq 2\alpha^2 - 2 < 2p$), we obtain an approximation $\tilde{d}'_\ell \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{d}'_\ell) < 2$ in time $O(p^{1+\delta})$. We then use Corollary 2.9 to compute an approximation $\tilde{z}_\ell \in \tilde{\mathbb{C}}_\circ$ for $z_\ell \coloneqq d'_\ell u_\ell$ such that $\varepsilon(\tilde{z}_\ell) < \varepsilon(\tilde{d}'_\ell) + 2 < 4$ in time $O(p^{1+\delta})$. Finally we set $(\tilde{\mathcal{D}}'u)_\ell \coloneqq \tilde{z}_\ell$. The total cost over all ℓ is $O(sp^{1+\delta}) = O(tp^{1+\delta})$. □

Lemma 4.8. *We may construct a numerical approximation $\tilde{\mathcal{S}}' \colon \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^t$ for $\mathcal{S}'$ such that $\varepsilon(\tilde{\mathcal{S}}') < 18p$ and $\mathsf{C}(\tilde{\mathcal{S}}') = O(tp^{3/2+\delta}\alpha)$.*

Proof. We are given as input $u \in \tilde{\mathbb{C}}_\circ^s$. For each $k = 0, \dots, t-1$ in turn, we approximate $(\mathcal{S}'u)_k$ as follows. By definition

$$(\mathcal{S}'u)_k = (\tfrac{1}{2}\mathcal{S}u)_k = \sum_{j\in\mathbb{Z}} \frac{1}{2}\alpha^{-1}e^{-\pi\alpha^{-2}(j-\frac{sk}{t})^2}u_j.$$

Let $m \coloneqq \lceil p^{1/2}\rceil\alpha$ and consider the truncated sum

$$T_k \coloneqq \sum_{|j-\frac{sk}{t}|<m} \frac{1}{2}\alpha^{-1}e^{-\pi\alpha^{-2}(j-\frac{sk}{t})^2}u_j. \tag{4.3}$$

There are at most $2m$ terms in this sum. Since $\|u\| \leqslant 1$ and $\alpha \geqslant 2$ we have

$$|(\mathcal{S}'u)_k - T_k| \leqslant \sum_{|j-\frac{sk}{t}|\geqslant m} \frac{1}{2}\alpha^{-1}e^{-\pi\alpha^{-2}(j-\frac{sk}{t})^2}|u_j| \leqslant \frac{1}{4}\sum_{|j-\frac{sk}{t}|\geqslant m} e^{-\pi\alpha^{-2}(j-\frac{sk}{t})^2}.$$

Let $w \coloneqq e^{-\pi\alpha^{-2}} < 1$; then

$$\begin{aligned}|(\mathcal{S}'u)_k - T_k| &\leqslant \tfrac{1}{2}(w^{m^2} + w^{(m+1)^2} + w^{(m+2)^2} + \cdots)\\ &= \tfrac{1}{2}w^{m^2}(1 + w^{2m+1} + w^{4m+4} + \cdots).\end{aligned}$$

Since $\alpha < p^{1/2}$ we have

$$w^m = e^{-\pi\lceil p^{1/2}\rceil/\alpha} \leqslant e^{-\pi p^{1/2}/\alpha} < e^{-\pi} < 0.05,$$

so certainly $1 + w^{2m+1} + w^{4m+4} + \cdots < 2$. We conclude that

$$|(\mathcal{S}'u)_k - T_k| < w^{m^2} \leqslant e^{-\pi\lceil p^{1/2}\rceil^2} \leqslant e^{-\pi p} < 2^{-p}.$$

Now we explain how to compute a suitable fixed-point approximation for T_k. Let $\beta \coloneqq \frac{1}{2}\alpha^{-1}$, and for each j appearing in (4.3), let $x_j \coloneqq e^{-\pi\alpha^{-2}(j-sk/t)^2}$, $y_j \coloneqq \beta x_j$, $z_j \coloneqq y_j u_j$, so that $T_k = \sum_j z_j$. We first compute $\tilde{\beta} \coloneqq \rho(\beta) = 2^{-p}\rho_0(2^{p-1}/\alpha) \in \tilde{\mathbb{C}}_\circ$ in time $O(p^{1+\delta})$; certainly $\varepsilon(\tilde{\beta}) < 2$. Then for each j we perform the following steps. As s, t, j, k and α are all integers with $O(p)$ bits, we may use Lemma 2.12 to compute an approximation $\tilde{x}_j \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{x}_j) < 2$ in time $O(p^{1+\delta})$. We then use Corollary 2.9 to compute an approximation $\tilde{y}_j \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{y}_j) < \varepsilon(\tilde{\beta}) + \varepsilon(\tilde{x}_j) + 2 < 6$, and again to obtain $\tilde{z}_j \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{z}_j) < \varepsilon(\tilde{y}_j) + 2 < 8$, in time $O(p^{1+\delta})$. Finally, we form the sum $\tilde{T}_k \coloneqq \sum_j \tilde{z}_j$; that is, writing $\tilde{z}_j = 2^{-p}a_j$ for integers a_j, we compute $\sum_j a_j$ and set $\tilde{T}_k \coloneqq 2^{-p}\sum_j a_j$. Observe that

$$\begin{aligned}2^p\,|\tilde{T}_k - (\mathcal{S}'u)_k| &\leqslant 2^p\,|\tilde{T}_k - T_k| + 2^p\,|T_k - (\mathcal{S}'u)_k|\\ &< (\textstyle\sum_j 2^p\,|\tilde{z}_j - z_j|) + 1 < (2m)\cdot 8 + 1 = 16m + 1.\end{aligned}$$

As $m < \lceil p^{1/2}\rceil p^{1/2} \leqslant p + p^{1/2}$ and $p \geqslant 100$, we find that

$$2^p\,|\tilde{T}_k - (\mathcal{S}'u)_k| \leqslant 16p + 16p^{1/2} + 1 < 18p.$$

Therefore

$$|\tilde{T}_k| \leqslant |\tilde{T}_k - (\mathcal{S}'u)_k| + |(\mathcal{S}'u)_k| < 18p\cdot 2^{-p} + \|\mathcal{S}'\|\,\|u\| \leqslant 10^{-26} + \tfrac{3}{4}\cdot 1 < 1,$$

confirming that $\tilde{T}_k \in \tilde{\mathbb{C}}_\circ$. Defining $(\tilde{\mathcal{S}}'u)_k \coloneqq \tilde{T}_k$, we have $2^p\,|(\tilde{\mathcal{S}}'u)_k - (\mathcal{S}'u)_k| < 18p$ as desired. The cost of the above procedure for each k is $O(mp^{1+\delta}) = O(p^{3/2+\delta}\alpha)$, so the total over all k is $O(tp^{3/2+\delta}\alpha)$. □

Remark 4.9. The algorithm in the proof of Lemma 4.8 amounts to multiplying a vector by a matrix of the type shown in Figure 2, including only those entries that are numerically significant, i.e., a strip of width roughly $2m$ around the diagonal. To avoid excessive tape movements in the Turing model, one must arrange the data so that it is not necessary to "jump" between u_0 and u_{s-1} when the index j wraps around modulo s. One simple way to achieve this is to perform a preprocessing step that generates the extended array $u_{-m}, u_{-m+1}, \ldots, u_{s+m}$; then the main algorithm operates on this array, working from left to right, with a sliding window of width $2m$. Similar remarks apply to the proof of Lemma 4.10 below.

Lemma 4.10. *We may construct a numerical approximation $\tilde{\mathcal{E}} \colon \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^s$ for $\mathcal{E}$ such that $\varepsilon(\tilde{\mathcal{E}}) < \frac{1}{3}p$ and $\mathsf{C}(\tilde{\mathcal{E}}) = O(tp^{3/2+\delta}\alpha^{-1})$.*

Proof. The argument is similar to the proof of Lemma 4.8. Given as input $u \in \tilde{\mathbb{C}}_\circ^s$, for each $\ell = 0, \dots, s-1$ we approximate $(\mathcal{E}u)_\ell$ as follows. By definition

$$(\mathcal{E}u)_\ell = \sum_{h \neq 0} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2)} u_{\ell+h}.$$

As in the proof of Lemma 4.5, for $h \neq 0$ we have

$$\begin{aligned}(\tfrac{th}{s} + \beta_\ell)^2 - \beta_{\ell+h}^2 &> (t/s)^2(|h| - \tfrac{1}{2})^2 - \tfrac{1}{4} \\ &> (|h| - \tfrac{1}{2})^2 - \tfrac{1}{4} = |h|\,(|h| - 1) \geqslant (|h| - 1)^2.\end{aligned}$$

Let $m \coloneqq \lceil (p/4\alpha^2)^{1/2} \rceil = \lceil p^{1/2}/2\alpha \rceil \geqslant 1$, and consider the truncated sum

$$T_\ell \coloneqq \sum_{h \neq 0,\, |h| \leqslant m} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2)} u_{\ell+h}. \tag{4.4}$$

This sum has exactly $2m$ terms. As $\|u\| \leqslant 1$ we have

$$|(\mathcal{E}u)_\ell - T_\ell| \leqslant \sum_{|h|>m} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2)}\, |u_{\ell+h}| < \sum_{|h|>m} e^{-\pi\alpha^2(|h|-1)^2}.$$

Let $w \coloneqq e^{-\pi\alpha^2} \leqslant e^{-4\pi} < 10^{-5}$; then $w^m < 10^{-5}$ and

$$\begin{aligned}|(\mathcal{E}u)_\ell - T_\ell| &\leqslant 2(w^{m^2} + w^{(m+1)^2} + w^{(m+2)^2} + \cdots) \\ &= 2w^{m^2}(1 + w^{2m+1} + w^{4m+4} + \cdots) \\ &< 3w^{m^2} \leqslant 3e^{-\pi\alpha^2(p/4\alpha^2)} = 3e^{-\pi p/4} < 3 \cdot 2^{-p}.\end{aligned}$$

Now we explain how to approximate T_ℓ. For each h appearing in (4.4), let $x_h \coloneqq e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell)^2 - \beta_{\ell+h}^2)}$ and $z_h \coloneqq x_h u_{\ell+h}$, so that $T_\ell = \sum_h z_h$. As in the proof of Lemma 4.8, we may use Lemma 2.12 to compute an approximation $\tilde{x}_h \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{x}_h) < 2$ in time $O(p^{1+\delta})$. Using Corollary 2.9, we then compute $\tilde{z}_h \in \tilde{\mathbb{C}}_\circ$ such that $\varepsilon(\tilde{z}_h) < \varepsilon(\tilde{x}_h) + 2 < 4$. Finally we set $\tilde{T}_\ell \coloneqq \sum_h \tilde{z}_h$. We have

$$\begin{aligned}2^p\,|\tilde{T}_\ell - (\mathcal{E}u)_\ell| &\leqslant 2^p\,|\tilde{T}_\ell - T_\ell| + 2^p\,|T_\ell - (\mathcal{E}u)_\ell| \\ &< (\textstyle\sum_h 2^p\,|\tilde{z}_h - z_h|) + 3 < (2m) \cdot 4 + 3 = 8m + 3.\end{aligned}$$

As $m \leqslant \frac{1}{4}p^{1/2} + 1$ and $p \geqslant 100$, we find that

$$2^p\,|\tilde{T}_\ell - (\mathcal{E}u)_\ell| < 2p^{1/2} + 11 < \tfrac{1}{3}p.$$

Therefore

$$|\tilde{T}_\ell| \leqslant |\tilde{T}_\ell - (\mathcal{E}u)_\ell| + |(\mathcal{E}u)_\ell| < \tfrac{1}{3}p \cdot 2^{-p} + \|\mathcal{E}\|\,\|u\| < 10^{-28} + \tfrac{1}{2} \cdot 1 < 1,$$

so we may define $(\tilde{\mathcal{E}}u)_\ell \coloneqq \tilde{T}_\ell \in \tilde{\mathbb{C}}_\circ$. The cost of the above procedure for each ℓ is $O(mp^{1+\delta})$. The hypothesis $\alpha < p^{1/2}$ implies that $m \leqslant \frac{1}{2}p^{1/2}\alpha^{-1} + 1 = O(p^{1/2}\alpha^{-1})$, so the total cost over all ℓ is $O(tp^{3/2+\delta}\alpha^{-1})$. □

Lemma 4.11. *We may construct a numerical approximation $\tilde{\mathcal{J}}' \colon \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^s$ for $\mathcal{J}'$ such that $\varepsilon(\tilde{\mathcal{J}}') < \frac{3}{4}p^2$ and $\mathsf{C}(\tilde{\mathcal{J}}') = O(tp^{3/2+\delta}\alpha)$.*

Proof. We are given as input $u \in \tilde{\mathbb{C}}_\circ^s$. Let $v \coloneqq u/2 \in \mathbb{C}_\circ^s$, and define $v^{(j)} \coloneqq \mathcal{E}^j v \in \mathbb{C}_\circ^s$ for $j \geqslant 0$ (recall that $\|\mathcal{E}\| < 2^{-\alpha^2\theta} \leqslant \frac{1}{2}$ by Lemma 4.5). We wish to approximate

$$\mathcal{J}'u = (\mathcal{N}^{-1}/2)u = \mathcal{N}^{-1}v = v - \mathcal{E}v + \mathcal{E}^2 v - \cdots = v^{(0)} - v^{(1)} + v^{(2)} - \cdots .$$

Let $n \coloneqq \lceil p/\alpha^2\theta \rceil = \lceil ps/\alpha^2(t-s) \rceil \geqslant 1$. We compute a sequence of approximations $\tilde{v}^{(0)}, \ldots, \tilde{v}^{(n-1)} \in \tilde{\mathbb{C}}_\circ^s$ as follows. First set $\tilde{v}^{(0)} \coloneqq \rho(v^{(0)}) = 2^{-p}\rho_0(2^{p-1}u) \in \tilde{\mathbb{C}}_\circ^s$, so that $\varepsilon(\tilde{v}^{(0)}) < 2$. Then compute in sequence $\tilde{v}^{(j)} \coloneqq \tilde{\mathcal{E}}\tilde{v}^{(j-1)} \in \tilde{\mathbb{C}}_\circ^s$ for $j = 1, \ldots, n-1$, using Lemma 4.10. We claim that $\varepsilon(\tilde{v}^{(j)}) < \frac{2}{3}p$ for each j. This is clear for $j = 0$, and for $j \geqslant 1$ it follows by induction as

$$\begin{aligned}\varepsilon(\tilde{v}^{(j)}) = 2^p \, \|\tilde{v}^{(j)} - v^{(j)}\| &\leqslant 2^p \, \|\tilde{\mathcal{E}}\tilde{v}^{(j-1)} - \mathcal{E}\tilde{v}^{(j-1)}\| + 2^p \, \|\mathcal{E}\tilde{v}^{(j-1)} - \mathcal{E}^j v\| \\ &\leqslant \varepsilon(\tilde{\mathcal{E}}) + \|\mathcal{E}\| \, \varepsilon(\tilde{v}^{(j-1)}) < \tfrac{1}{3}p + \tfrac{1}{2} \cdot \tfrac{2}{3}p = \tfrac{2}{3}p.\end{aligned}$$

Finally, define $\tilde{\mathcal{J}}'u \coloneqq \tilde{v}^{(0)} - \tilde{v}^{(1)} + \cdots \pm \tilde{v}^{(n-1)}$. Then we have

$$2^p \, \|\tilde{\mathcal{J}}'u - \mathcal{J}'u\| \leqslant \sum_{j=0}^{n-1} 2^p \, \|\tilde{v}^{(j)} - v^{(j)}\| + 2^p \sum_{j=n}^{\infty} \|v^{(j)}\| < \tfrac{2}{3}np + \frac{2^p \, \|\mathcal{E}\|^n}{2(1 - \|\mathcal{E}\|)}.$$

Observe that $2(1 - \|\mathcal{E}\|) > 1$ and $\|\mathcal{E}\|^n < 2^{-\alpha^2\theta n} \leqslant 2^{-p}$; also, as $\alpha^2\theta \geqslant 1$, we have $n \leqslant p$. Thus

$$2^p \, \|\tilde{\mathcal{J}}'u - \mathcal{J}'u\| < \tfrac{2}{3}p^2 + 1 < \tfrac{3}{4}p^2.$$

This also shows that $\tilde{\mathcal{J}}'u \in \tilde{\mathbb{C}}_\circ^s$, as

$$\|\tilde{\mathcal{J}}'u\| \leqslant \|\mathcal{J}'u\| + \|\tilde{\mathcal{J}}'u - \mathcal{J}'u\| < \|\mathcal{J}'\| \, \|u\| + \tfrac{3}{4}p^2 \cdot 2^{-p} < \tfrac{7}{8} \cdot 1 + 10^{-26} < 1.$$

In the algorithm above, the bulk of the work consists of n invocations of $\tilde{\mathcal{E}}$. The hypothesis $\theta \geqslant p/\alpha^4$ implies that

$$n \leqslant \frac{p}{\alpha^2\theta} + 1 \leqslant \alpha^2 + 1,$$

so the total cost is $O(tnp^{3/2+\delta}\alpha^{-1}) = O(tp^{3/2+\delta}\alpha)$. □

Now we may complete the proof of Proposition 4.6.

Proof of Proposition 4.6(ii). For $\tilde{\mathcal{A}}$ we simply take $\tilde{\mathcal{A}} \coloneqq \tilde{\mathcal{S}}'$ where $\tilde{\mathcal{S}}'$ is as described in Lemma 4.8; then $\varepsilon(\tilde{\mathcal{A}}) < 18p < p^2$, and $\mathsf{C}(\tilde{\mathcal{A}}) = O(tp^{3/2+\delta}\alpha)$.

For $\tilde{\mathcal{B}}$ we take $\tilde{\mathcal{B}} \coloneqq \tilde{\mathcal{P}}_s^{-1}\tilde{\mathcal{D}}'\tilde{\mathcal{J}}'\tilde{\mathcal{C}}\tilde{\mathcal{P}}_t$, where $\tilde{\mathcal{D}}'$ and $\tilde{\mathcal{J}}'$ are as described in Lemmas 4.7 and 4.11, and where $\tilde{\mathcal{C}} \colon \tilde{\mathbb{C}}_\circ^t \to \tilde{\mathbb{C}}_\circ^s$, $\tilde{\mathcal{P}}_s^{-1} \colon \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^s$ and $\tilde{\mathcal{P}}_t \colon \tilde{\mathbb{C}}_\circ^t \to \tilde{\mathbb{C}}_\circ^t$ are the maps performing the obvious data rearrangements corresponding to $\mathcal{C}$, $\mathcal{P}_s^{-1}$ and $\mathcal{P}_t$, namely

$$\begin{aligned}(\tilde{\mathcal{C}}u)_\ell &\coloneqq u_{[t\ell/s]}, && u \in \tilde{\mathbb{C}}_\circ^t, \quad 0 \leqslant \ell < s, \\ (\tilde{\mathcal{P}}_s^{-1}u)_j &\coloneqq u_{(t^{-1} \bmod s)j}, && u \in \tilde{\mathbb{C}}_\circ^s, \quad 0 \leqslant j < s, \\ (\tilde{\mathcal{P}}_t u)_k &\coloneqq u_{-sk}, && u \in \tilde{\mathbb{C}}_\circ^t, \quad 0 \leqslant k < t.\end{aligned}$$

These do not perform any arithmetic in $\tilde{\mathbb{C}}_\circ$ so $\varepsilon(\tilde{\mathcal{C}}) = \varepsilon(\tilde{\mathcal{P}}_s^{-1}) = \varepsilon(\tilde{\mathcal{P}}_t) = 0$. By Corollary 2.7 we obtain $\varepsilon(\tilde{\mathcal{B}}) \leqslant \varepsilon(\tilde{\mathcal{D}}') + \varepsilon(\tilde{\mathcal{J}}') < 4 + \frac{3}{4}p^2 < p^2$.

As for the complexity, first observe that $\tilde{\mathcal{C}}$ simply copies its input in order, skipping $t - s$ unwanted entries, so $\mathsf{C}(\tilde{\mathcal{C}}) = O(tp)$. To compute $\tilde{\mathcal{P}}_s^{-1}u$ for $u \in \tilde{\mathbb{C}}_\circ^s$,

we use a "label-and-sort" algorithm: we first construct the list of ordered pairs $(tj \bmod s, u_j)$ for $j = 0, \dots, s-1$ in time $O(tp)$ (each label occupies $O(p)$ bits as $s < 2^p$), then sort the list by the first entry using merge sort in time $O(tp \log t)$ [30], and finally extract the second entries to obtain the desired output in time $O(tp)$. Thus $\mathsf{C}(\tilde{\mathcal{P}}_s^{-1}) = O(tp \log t)$, and similarly $\mathsf{C}(\tilde{\mathcal{P}}_t) = O(tp \log t)$. Altogether we have

$$\mathsf{C}(\tilde{\mathcal{B}}) = \mathsf{C}(\tilde{\mathcal{D}}') + \mathsf{C}(\tilde{\mathcal{J}}') + O(tp \log t) = O(tp^{1+\delta} + tp^{3/2+\delta}\alpha + tp \log t). \qquad \square$$

Finally we show how to deduce the general case from the one-dimensional case.

Proof of Theorem 4.1. Let $s_1, \dots, s_d$ and $t_1, \dots, t_d$ be as in the statement of the theorem. Applying Proposition 4.6 for each i, we obtain maps $\mathcal{A}_i \colon \mathbb{C}^{s_i} \to \mathbb{C}^{t_i}$ and $\mathcal{B}_i \colon \mathbb{C}^{t_i} \to \mathbb{C}^{s_i}$ with $\|\mathcal{A}_i\|, \|\mathcal{B}_i\| \leqslant 1$ such that $\mathcal{F}_{s_i} = 2^{2\alpha^2} \mathcal{B}_i \mathcal{F}_{t_i} \mathcal{A}_i$, and approximations $\tilde{\mathcal{A}}_i \colon \tilde{\mathbb{C}}_\circ^{s_i} \to \tilde{\mathbb{C}}_\circ^{t_i}$ and $\tilde{\mathcal{B}}_i \colon \tilde{\mathbb{C}}_\circ^{t_i} \to \tilde{\mathbb{C}}_\circ^{s_i}$ such that $\varepsilon(\tilde{\mathcal{A}}_i), \varepsilon(\tilde{\mathcal{B}}_i) < p^2$ and $\mathsf{C}(\tilde{\mathcal{A}}_i), \mathsf{C}(\tilde{\mathcal{B}}_i) = O(t_i p^{3/2+\delta}\alpha + t_i p \log t_i)$.

Now observe that

$$\mathcal{F}_{s_1,\dots,s_d} = \otimes_i \mathcal{F}_{s_i} = 2^{2d\alpha^2} (\otimes_i \mathcal{B}_i)(\otimes_i \mathcal{F}_{t_i})(\otimes_i \mathcal{A}_i) = 2^{2d\alpha^2} \mathcal{B} \mathcal{F}_{t_1,\dots,t_d} \mathcal{A},$$

where $\mathcal{A} \coloneqq \otimes_i \mathcal{A}_i \colon \otimes_i \mathbb{C}^{s_i} \to \otimes_i \mathbb{C}^{t_i}$ and $\mathcal{B} \coloneqq \otimes_i \mathcal{B}_i \colon \otimes_i \mathbb{C}^{t_i} \to \otimes_i \mathbb{C}^{s_i}$. Applying Lemma 2.10 (with $R \coloneqq \mathbb{C}$, $r \coloneqq 1$, $m_i \coloneqq s_i$, $n_i \coloneqq t_i$), we may construct an approximation $\tilde{\mathcal{A}} \colon \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ such that $\varepsilon(\tilde{\mathcal{A}}) \leqslant \sum_i \varepsilon(\tilde{\mathcal{A}}_i) < dp^2$. Moreover, let M be as in Lemma 2.10; then $M = \prod_i \max(s_i, t_i) = t_1 \cdots t_d = T$, so

$$\begin{aligned} \mathsf{C}(\tilde{\mathcal{A}}) &< T \sum_i \frac{\mathsf{C}(\tilde{\mathcal{A}}_i)}{t_i} + O(Tp \log T) \\ &< T \sum_i O(p^{3/2+\delta}\alpha + p \log t_i) + O(Tp \log T) \\ &= O(dTp^{3/2+\delta}\alpha + Tp \log T). \end{aligned}$$

We may similarly construct an approximation $\tilde{\mathcal{B}}$ satisfying exactly the same error and cost bounds, and this completes the proof. $\square$

4.4. **Further remarks.** Our presentation of the Gaussian resampling technique has been optimised in favour of giving the simplest possible proof of the main $\mathsf{M}(n) = O(n \log n)$ bound. In this section we outline several ways in which these results may be improved and generalised, with an eye towards practical applications.

4.4.1. *Minor technical issues.* In our presentation we insisted that α be an integer and that $\alpha \geqslant 2$. Neither of these restrictions are essential; they were made for technical reasons to simplify certain proofs.

Similarly, the assumption $\gcd(s, t) = 1$ is not necessary. If $g \coloneqq \gcd(s, t) > 1$, then one can prove a variant of Theorem 4.1 in which the system corresponding to $\mathcal{T}x = y$ breaks up into g pieces that may be solved independently. In fact, the problem becomes slightly easier from a computational point of view.

4.4.2. *Faster system solving.* The iterative method used in Lemma 4.11 to approximate $\mathcal{J} = \mathcal{N}^{-1}$ (i.e., to solve the system $\mathcal{T}x = y$) has complexity $O(tp^{5/2+\delta}/\alpha^3\theta)$. To ensure that this step does not dominate the $O(tp^{3/2+\delta}\alpha)$ complexity of approximating $\mathcal{S}$ (Lemma 4.8), we were compelled to introduce the hypothesis $\theta \geqslant p/\alpha^4$. On the other hand, to make the target DFT of length t as cheap as possible, it is desirable for θ to be as close to zero as possible. Together, these considerations

imply that we cannot take α smaller than about $p^{1/4}$. (Indeed, in Section 5, for fixed d, we do take $\alpha = \Theta(p^{1/4})$ for this very reason.) For the choice $\alpha = \Theta(p^{1/4})$, the overall complexity in Proposition 4.6 is $O(tp^{7/4+\delta})$.

A better complexity bound may be obtained by precomputing an LU (alternatively, Cholesky) decomposition for $\mathcal{N}$, and then solving the system directly. The cost of the precomputation is $O(tp^{2+\delta}/\alpha^2)$ (assuming classical matrix arithmetic), and then the cost of each application of $\mathcal{J}$ becomes $O(tp^{3/2+\delta}/\alpha)$. This allows us to relax the condition $\theta \geqslant p/\alpha^4$ to merely $\theta \geqslant 1/\alpha^2$. Taking $\alpha = \Theta(1)$, the overall complexity in Proposition 4.6 (discounting the precomputation) falls to $O(tp^{3/2+\delta})$. We did not use this method in our presentation because the error analysis is considerably more intricate than for the iterative method.

After making this modification, it would be interesting to investigate whether this method is competitive for practical computations of complex DFTs of length s when s is a large prime. One would choose a smooth transform length t somewhat larger than s, say $1.25s < t < 1.5s$, and use the algorithm to reduce the desired DFT of length s to a DFT of length t; the latter could be handled via existing software libraries implementing the Cooley–Tukey algorithm. For large enough s, perhaps around 2^{20} or 2^{30}, we expect that the invocations of $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{J}}$ would be quite cheap compared to the FFT of length t. Indeed, $\tilde{\mathcal{S}}$ can be computed in a single pass over the input vector, and $\tilde{\mathcal{J}}$ in two passes (one for each of the L and U matrices), so they have excellent locality. It is conceivable that a highly optimised implementation could outperform existing software libraries, which handle transforms of prime length by techniques such as Rader's algorithm [37]. Such techniques introduce a large constant factor overhead that does not arise in the method just sketched.

We also mention that it may be of interest to replace the Gaussian in the definition of $\mathcal{S}$ by some other smooth function $g(x)$, and accordingly replace the Gaussian in the definition of $\mathcal{T}$ by its Fourier transform $\hat{g}(y)$. The main requirement is that both of these functions should be strongly localised around zero. Different choices for $g(x)$ may lead to various tradeoffs between numerical accuracy and computational complexity.

4.4.3. *Comparison with the Dutt–Rokhlin method.* There is an enormous literature on "non-uniform FFTs" (sometimes called "non-equispaced FFTs"), going back to the seminal paper of Dutt and Rokhlin [11]. They consider transforms of the type

$$v_j \coloneqq \sum_{k=0}^{t-1} e^{2\pi \mathrm{i} \omega_k y_j} u_k, \qquad 0 \leqslant j < t. \tag{4.5}$$

The ordinary "uniform" DFT may be regarded as the special case where $\omega_k \coloneqq k$ and $y_j \coloneqq j/t$, but the Dutt–Rokhlin algorithms may be applied in cases where the frequencies ω_k are not necessarily integers, and/or the sample points y_j are not necessarily integer multiples of $1/t$. In these cases the algorithms reduce the problem to an ordinary FFT of length t (or in some variants, a small multiple of t). The complexity, counting floating-point operations, is $O(t \log t + tp)$, where p is the desired precision in bits.

If we now take instead $\omega_k \coloneqq k$ and $y_j \coloneqq j/s$, where s is the "source" transform length, we see that (4.5) is exactly a DFT of length s (apart from some inconsequential zero-padding), so the Dutt–Rokhlin algorithms may be used to compute a

DFT of length s by means of an FFT of length t. Inspection of their algorithms in this case reveals them to be *essentially equivalent to our method in the special case that* $\alpha = \Theta(p^{1/2})$.

For example, consider [11, Algorithm 2], which corresponds roughly to a "transposed" version of our algorithm. Step 3 of that algorithm is analogous to approximating $\mathcal{S}$ (see Lemma 4.8). For the choice $\alpha = \Theta(p^{1/2})$, the complexity for this step is $O(tp^{2+\delta})$ bit operations, corresponding to the $O(tp)$ term in their complexity bound. Step 2 corresponds to our $\mathcal{F}_t$, and yields the $O(t \log t)$ term. The most interesting point of comparison is Step 1, which corresponds roughly to solving the system $\mathcal{T}x = y$. The choice $\alpha = \Theta(p^{1/2})$ implies that this system is essentially *diagonal*, i.e., the off-diagonal entries of $\mathcal{T}$ decay so rapidly that for numerical purposes they may be discarded. Solving the system is therefore trivial: their Step 1 consists of simply dividing each coefficient by the corresponding diagonal entry of $\mathcal{T}$ (in the literature these are often called "scale factors"). This step contributes only $O(t)$ floating-point operations.

The reason that Dutt and Rokhlin are (in effect) unable to take α smaller than about $p^{1/2}$ is essentially due to the approximation error committed when they truncate the Gaussian, for example in [11, Theorem 2.7]. Our Theorem 4.1 may be viewed as an "exact" replacement for that theorem. Rather than truncate the Gaussian, we take into account the effect of the Gaussian tail, which manifests as the off-diagonal entries of our $\mathcal{T}$ matrix. For α considerably smaller than $p^{1/2}$, these entries are numerically significant and cannot be ignored.

In our algorithm, assuming that we use the LU decomposition method mentioned in Section 4.4.2, as α decreases from $\Theta(p^{1/2})$ to $\Theta(1)$ we see that the complexity of approximating $\mathcal{S}$ decreases from $O(tp)$ to $O(tp^{1/2})$ floating-point operations, and the complexity of approximating $\mathcal{J}$ increases from $O(t)$ to $O(tp^{1/2})$. When $\alpha = \Theta(1)$ they are balanced, and the overall complexity drops to $O(t \log t + tp^{1/2})$; the last term improves on the Dutt–Rokhlin bound by a factor of $p^{1/2}$. Note that the Dutt–Rokhlin bound is not strong enough for our application to integer multiplication; using their bound, the error term in Proposition 5.2 would grow to $O(n(\log n)^{1+\delta})$ which is unacceptably large.

Of course, our discussion has only considered the case corresponding to the DFT of length s, i.e., the choice $y_j \coloneqq j/s$. An interesting question is whether the bound $O(t \log t + tp^{1/2})$ can be proved for the general non-equispaced case, and if so, whether this method outperforms the Dutt–Rokhlin algorithm in practice.

5. The main algorithm

In this section we present the main integer multiplication algorithm. We actually give a family of algorithms, parameterised by a dimension parameter $d \geqslant 2$. Let

$$n_0 \coloneqq 2^{d^{12}} \geqslant 2^{4096},$$

and suppose that we wish to multiply integers with n bits. For $n < n_0$, we may use any convenient base-case multiplication algorithm, such as the classical $O(n^2)$ algorithm. For $n \geqslant n_0$ we will describe a recursive algorithm that reduces the problem to a collection of multiplication problems of size roughly $n^{1/d}$. We will show that this algorithm achieves $\mathsf{M}(n) = O(n \log n)$, provided that $d \geqslant 1729$.

5.1. Parameter selection.

Henceforth we assume that $n \geqslant n_0$. We first discuss the computation of several parameters depending on n that will be needed later.

Let

$$b \coloneqq \lceil \log_2 n \rceil \geqslant d^{12} \geqslant 4096$$

be the "chunk size", and let the working precision be

$$p \coloneqq 6b = 6\lceil \log_2 n \rceil \geqslant 6d^{12} \geqslant 24576 > 100.$$

Define

$$\alpha \coloneqq \lceil (12d^2 b)^{1/4} \rceil.$$

Clearly $\alpha \geqslant 2$, and as $d \leqslant b^{1/12}$ and $b \geqslant 4096$, we also have

$$\alpha \leqslant \lceil 12^{1/4} b^{7/24} \rceil \leqslant 1.87 \cdot b^{7/24} + 1 < 2b^{7/24} < p^{1/2}. \tag{5.1}$$

As in Theorem 4.1, set

$$\gamma \coloneqq 2d\alpha^2 < 2b^{1/12} \cdot 4b^{7/12} = 8b^{2/3}. \tag{5.2}$$

Let T be the unique power of two lying in the interval

$$4n/b \leqslant T < 8n/b, \tag{5.3}$$

and let r be the unique power of two in the interval

$$T^{1/d} \leqslant r < 2T^{1/d}. \tag{5.4}$$

We certainly have $b \leqslant 4n^{1/2}$, so

$$r \geqslant (4n/b)^{1/d} \geqslant n^{1/2d} \geqslant 2^{d^{10}}. \tag{5.5}$$

We now construct a factorisation $T = t_1 \cdots t_d$ satisfying the hypotheses of Theorem 3.1. Let $d' \coloneqq \log_2(r^d/T)$. As $T \leqslant r^d < 2^d T$ we have $1 \leqslant r^d/T < 2^d$ and hence $0 \leqslant d' < d$. Define

$$t_1, \ldots, t_{d'} \coloneqq \frac{r}{2}, \qquad t_{d'+1}, \ldots, t_d \coloneqq r.$$

Then $t_d \geqslant \cdots \geqslant t_1 \geqslant 2$ and

$$t_1 \cdots t_d = (r/2)^{d'} r^{d-d'} = r^d / 2^{d'} = T.$$

Also

$$T < 8n/b < n \leqslant 2^b < 2^p,$$

so the hypotheses of Theorem 3.1 are indeed satisfied. The parameters b, p, α, γ, T, r and $t_1, \ldots, t_d$ may all be computed in time $(\log n)^{O(1)}$.

Our next task is to choose distinct primes $s_1, \ldots, s_d$ that are slightly smaller than the corresponding $t_1, \ldots, t_d$. In a moment we will use Theorem 4.1 to reduce a transform of size $s_1 \times \cdots \times s_d$ to a transform of size $t_1 \times \cdots \times t_d$; to avoid excessive data expansion in this reduction, we must ensure that the ratio $t_1 \cdots t_d / s_1 \cdots s_d$ is not too large. On the other hand, to satisfy the requirements of the theorem, we must also ensure that the individual ratios t_i/s_i are not too close to 1. We will achieve this by means of the following result.

Lemma 5.1. *Let $\eta \in (0, \frac{1}{4})$ and let $x \geqslant e^{2/\eta}$. Then there are at least $\frac{1}{2}\eta x / \log x$ primes q in the interval $(1-2\eta)x < q \leqslant (1-\eta)x$.*

Proof. Let $\vartheta(y) \coloneqq \sum_{q \leqslant y} \log q$ (sum taken over primes) denote the usual Chebyshev function. According to [38, Thm. 4], for all $y \geqslant 563$ we have

$$y - \frac{y}{2\log y} < \vartheta(y) < y + \frac{y}{2\log y}.$$

As the function $y/\log y$ is increasing for $y \geqslant 563$, we see that

$$y - \frac{x}{2\log x} < \vartheta(y) < y + \frac{x}{2\log x}, \qquad 563 \leqslant y \leqslant x.$$

Applying this result for $y_0 \coloneqq (1-2\eta)x > x/2 > e^8/2 > 563$, and then again for $y_1 \coloneqq (1-\eta)x > 3x/4 > 563$, we obtain

$$\vartheta(y_1) - \vartheta(y_0) > y_1 - \frac{x}{2\log x} - y_0 - \frac{x}{2\log x} = \eta x - \frac{x}{\log x} \geqslant \eta x - \frac{x}{2/\eta} = \frac{\eta x}{2}$$

and hence

$$\sum_{y_0 < q \leqslant y_1} 1 \geqslant \frac{1}{\log x} \sum_{y_0 < q \leqslant y_1} \log q = \frac{\vartheta(y_1) - \vartheta(y_0)}{\log x} > \frac{\eta x}{2\log x}. \qquad \square$$

Let us apply Lemma 5.1 with

$$\eta \coloneqq \frac{1}{4d} \leqslant \frac{1}{8}$$

and $x \coloneqq r/2$, noting that (5.5) implies that $r/2 \geqslant 2^{d^{10}-1} \geqslant e^{8d} = e^{2/\eta}$. We find that there are at least

$$\frac{1}{8d} \cdot \frac{r/2}{\log(r/2)} \geqslant \frac{1}{16d} \cdot \frac{r}{\log r} \geqslant \frac{1}{16d} \cdot \frac{2^{d^{10}}}{\log(2^{d^{10}})} = \frac{2^{d^{10}}}{(16\log 2)d^{11}} \geqslant d \geqslant d'$$

primes q in the interval

$$(1-2\eta)\frac{r}{2} < q \leqslant (1-\eta)\frac{r}{2}.$$

We may find d' such primes $s_1, \dots, s_{d'}$ in time $r^{1+o(1)} = o(n)$. Applying Lemma 5.1 again, with the same η but now with $x \coloneqq r \geqslant e^{2/\eta}$, we find that there are at least $d \geqslant d - d'$ primes q in the interval

$$(1-2\eta)r < q \leqslant (1-\eta)r.$$

Again, we may find $d - d'$ such primes $s_{d'+1}, \dots, s_d$ in time $o(n)$. These two collections of primes are disjoint, as

$$(1-\eta)\frac{r}{2} < \frac{r}{2} < \frac{3r}{4} \leqslant (1-2\eta)r.$$

In summary, we have found d distinct primes $s_1, \dots, s_d$ such that

$$(1-2\eta)t_i < s_i \leqslant (1-\eta)t_i, \qquad i \in \{1, \dots, d\}.$$

Setting $S \coloneqq s_1 \cdots s_d < T$, we see that

$$\frac{S}{T} > (1-2\eta)^d = \left(1 - \frac{1}{2d}\right)^d > \frac{1}{2}, \tag{5.6}$$

as $(1-x)^d > 1 - dx$ for all $x \in (0,1)$.

5.2. **DFTs and convolutions for coprime sizes.** The following result combines Theorems 3.1 and 4.1 to obtain an approximation for the complex transform $\mathcal{F}_{s_1,\dots,s_d} \colon \otimes_i \mathbb{C}^{s_i} \to \otimes_i \mathbb{C}^{s_i}$. Recall that the working precision was chosen to be $p \coloneqq 6b = 6\lceil \log_2 n \rceil$.

Proposition 5.2. *We may construct a numerical approximation* $\tilde{\mathcal{F}}_{s_1,\dots,s_d}\colon \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$ *for* $\mathcal{F}_{s_1,\dots,s_d}$ *such that* $\varepsilon(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) < 2^{\gamma+5} T \log_2 T$ *and*

$$\mathsf{C}(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) < \frac{4T}{r}\,\mathsf{M}(3rp) + O(n \log n).$$

Proof. Let us verify the hypotheses of Theorem 4.1. We have already seen that $2 \leqslant s_i < t_i < 2^p$ for all i, and that $2 \leqslant \alpha < p^{1/2}$. We certainly have $\gcd(s_i, t_i) = 1$, as s_i is an odd prime and t_i is a power of two. Let

$$\theta_i \coloneqq \frac{t_i}{s_i} - 1 \geqslant \frac{1}{1-\eta} - 1 = \frac{\eta}{1-\eta} > \eta = \frac{1}{4d}.$$

Then

$$\alpha^4 \theta_i \geqslant \frac{12d^2 b}{4d} = 3db = \frac{d}{2} \cdot p \geqslant p,$$

so $\theta_i \geqslant p/\alpha^4$ as required.

Theorem 4.1 thus produces maps $\mathcal{A}\colon \otimes_i \mathbb{C}^{s_i} \to \otimes_i \mathbb{C}^{t_i}$ and $\mathcal{B}\colon \otimes_i \mathbb{C}^{t_i} \to \otimes_i \mathbb{C}^{s_i}$ such that $\mathcal{F}_{s_1,\dots,s_d} = 2^\gamma \mathcal{A} \mathcal{F}_{t_1,\dots,t_d} \mathcal{B}$, and approximations $\tilde{\mathcal{A}}\colon \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ and $\tilde{\mathcal{B}}\colon \otimes_i \tilde{\mathbb{C}}_\circ^{t_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$. Applying Theorem 3.1 (whose hypotheses were verified earlier), we obtain furthermore an approximation $\tilde{\mathcal{F}}_{t_1,\dots,t_d}$ for $\mathcal{F}_{t_1,\dots,t_d}$. Now consider the scaled transform

$$\mathcal{F}'_{s_1,\dots,s_d} \coloneqq 2^{-\gamma} \mathcal{F}_{s_1,\dots,s_d} = \mathcal{A} \mathcal{F}_{t_1,\dots,t_d} \mathcal{B},$$

and the approximation $\tilde{\mathcal{F}}'_{s_1,\dots,s_d} \coloneqq \tilde{\mathcal{A}} \tilde{\mathcal{F}}_{t_1,\dots,t_d} \tilde{\mathcal{B}}$. By Corollary 2.7 we have

$$\varepsilon(\tilde{\mathcal{F}}'_{s_1,\dots,s_d}) \leqslant \varepsilon(\tilde{\mathcal{A}}) + \varepsilon(\tilde{\mathcal{F}}_{t_1,\dots,t_d}) + \varepsilon(\tilde{\mathcal{B}}) < 2dp^2 + 8T \log_2 T.$$

As it is known that $\|\mathcal{F}_{s_1,\dots,s_d}\| \leqslant 1$, we obtain the desired approximation $\tilde{\mathcal{F}}_{s_1,\dots,s_d}$ by applying Lemma 2.2 with $c \coloneqq 2^\gamma$ to the output of $\tilde{\mathcal{F}}'_{s_1,\dots,s_d}$ (the condition $c \leqslant 2^p$ holds as $\gamma < 8b^{2/3} < p$; see (5.2)). We therefore obtain

$$\begin{aligned}
\varepsilon(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) &< 2^{\gamma+1} \varepsilon(\tilde{\mathcal{F}}'_{s_1,\dots,s_d}) + 3 \\
&< 2^{\gamma+1}(2dp^2 + 8T \log_2 T) + 3 \\
&< 2^{\gamma+1}(3dp^2 + 8T \log_2 T).
\end{aligned}$$

Since

$$3dp^2 \leqslant 108 b^{1/12} b^2 = 108 \lceil \log_2 n \rceil^{25/12} < n^{1/2} < T < 8T \log_2 T,$$

we conclude that $\varepsilon(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) < 2^{\gamma+5} T \log_2 T$.

The cost of the scaling is $O(Sp^{1+\delta}) = O(Tp^{1+\delta})$, so the overall complexity is

$$\begin{aligned}
\mathsf{C}(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) &= \mathsf{C}(\tilde{\mathcal{A}}) + \mathsf{C}(\tilde{\mathcal{F}}_{t_1,\dots,t_d}) + \mathsf{C}(\tilde{\mathcal{B}}) + O(Tp^{1+\delta}) \\
&= \frac{4T}{r}\,\mathsf{M}(3rp) + O(dTp^{3/2+\delta}\alpha + Tp \log T + Tp^{1+\delta}).
\end{aligned}$$

Recalling (5.1) and the assumption $\delta < \frac{1}{8}$, we see that

$$dp^{3/2+\delta}\alpha = O(p^{1/12} p^{3/2+\delta} p^{7/24}) = O(p^{15/8+\delta}) = O(p^2).$$

By definition $p = O(\log n)$, and (5.3) yields $T = O(n/\log n)$. The bound for $\mathsf{C}(\tilde{\mathcal{F}}_{s_1,\dots,s_d})$ thus simplifies to $(4T/r)\,\mathsf{M}(3rp) + O(n \log n)$. □

Next we construct an approximation for the scaled convolution map $\mathcal{M}\colon \otimes_i \mathbb{C}^{s_i} \times \otimes_i \mathbb{C}^{s_i} \to \otimes_i \mathbb{C}^{s_i}$ given by $\mathcal{M}(u,v) \coloneqq \frac{1}{S} u * v$. Note that $\|\mathcal{M}\| \leqslant 1$.

Proposition 5.3. *We may construct a numerical approximation $\tilde{\mathcal{M}}\colon \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \times \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$ for $\mathcal{M}$ such that $\varepsilon(\tilde{\mathcal{M}}) < 2^{\gamma+8} T^2 \log_2 T$ and*

$$\mathsf{C}(\tilde{\mathcal{M}}) < \frac{12T}{r}\,\mathsf{M}(3rp) + O(n \log n). \tag{5.7}$$

Proof. We are given as input $u, v \in \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$. Let $w \coloneqq \mathcal{M}(u,v) = \frac{1}{S} u * v \in \otimes_i \mathbb{C}^{s_i}$. According to (2.1) we have $w = Sw'$ where $w' \coloneqq \mathcal{F}^*_{s_1,\dots,s_d}(\mathcal{F}_{s_1,\dots,s_d} u \cdot \mathcal{F}_{s_1,\dots,s_d} v)$. We use the following algorithm (essentially the same as in the proof of Proposition 3.4). We first compute an approximation $\tilde{w}' \in \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$ by using Proposition 5.2 to handle the forward and inverse transforms, and Corollary 2.9 to handle the pointwise multiplications. Applying Lemma 2.6 in the usual way, we obtain

$$\begin{aligned}\varepsilon(\tilde{w}') &\leqslant \varepsilon(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) + \varepsilon(\tilde{\mathcal{F}}_{s_1,\dots,s_d}) + \varepsilon(\tilde{\mathcal{F}}^*_{s_1,\dots,s_d}) + 2 \\ &< 3 \cdot 2^{\gamma+5} T \log_2 T + 2 < \tfrac{7}{2} \cdot 2^{\gamma+5} T \log_2 T.\end{aligned}$$

Then we apply Lemma 2.2 (with $c \coloneqq S \leqslant T \leqslant 2^p$) to obtain an approximation $\tilde{w} \in \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$ such that

$$\varepsilon(\tilde{w}) < 2S\varepsilon(\tilde{w}') + 3 < 7S \cdot 2^{\gamma+5} T \log_2 T + 3 < 2^{\gamma+8} T^2 \log_2 T.$$

The cost of the pointwise multiplications and scalings is $O(Sp^{1+\delta}) = O(n \log n)$, and the constant 12 accounts for the three invocations of Proposition 5.2. □

5.3. **Integer multiplication.** We are now in a position to describe the recursive step of the main integer multiplication algorithm.

Proposition 5.4. *For $n \geqslant n_0$ we have*

$$\mathsf{M}(n) < \frac{12T}{r}\,\mathsf{M}(3rp) + O(n \log n). \tag{5.8}$$

Proof. We are given as input positive integers $f, g < 2^n$. The algorithm consists of the following series of reductions.

(1) *Reduce to one-dimensional convolution over $\mathbb{Z}$.* Let $N \coloneqq \lceil n/b \rceil$ where $b \coloneqq \lceil \log_2 n \rceil$ as above. We split f and g into N chunks of b bits, i.e., we write $f = F(2^b)$ and $g = G(2^b)$ where

$$F(x) = \sum_{j=0}^{N-1} F_j x^j \in \mathbb{Z}[x], \qquad G(x) = \sum_{j=0}^{N-1} G_j x^j \in \mathbb{Z}[x], \qquad 0 \leqslant F_j, G_j < 2^b.$$

We have $fg = (FG)(2^b)$, so it suffices to compute the polynomial product $H(x) \coloneqq F(x)G(x)$ and then evaluate at $x = 2^b$. The coefficients of $H(x)$ lie in the interval $0 \leqslant H_j < 2^{2b} N < 2^{3b}$; in particular, they have at most $3b = O(\log n)$ bits, so the evaluation may be achieved via a straightforward overlap-add algorithm in time $O(N \log n) = O(n)$. By (5.3) and (5.6) we have

$$\deg H \leqslant 2N - 2 \leqslant \frac{2n}{b} \leqslant \frac{T}{2} < S,$$

so it suffices to compute $F(x)G(x) \pmod{x^S - 1}$.

(2) *Reduce to d-dimensional convolution over $\mathbb{Z}$.* We now use the Agarwal–Cooley method [1] to reduce to a multidimensional convolution. Consider the ring

$$\mathscr{A} \coloneqq \mathbb{Z}[x_1, \dots, x_d] / (x_1^{s_1} - 1, \dots, x_d^{s_d} - 1).$$

As the integers $s_1, \dots, s_d$ are pairwise relatively prime, there is an isomorphism of rings $\mathbb{Z}[x]/(x^S - 1) \cong \mathscr{A}$ induced by the Chinese remainder theorem (for instance, by sending x to $x_1 \cdots x_d$). Let $F', G', H' \in \mathscr{A}$ be the images of F, G and H, so that $H' = F'G'$. In the Turing model, F', G' and H' are represented as d-dimensional arrays of integers of $3b$ bits, using a similar layout to the tensor products in Section 2.3. The isomorphism amounts to a data rearrangement, and may be computed in either direction in time

$$O(bS \log S) = O(bT \log T) = O(n \log n)$$

by attaching labels and sorting, as in the proof of Proposition 4.6(ii). (For an alternative algorithm that does not rely on sorting, see [23, Sec. 2.3].) We have thus reduced to the problem of computing $H' = F'G'$ in $\mathscr{A}$.

(3) *Reduce to approximate d-dimensional convolution over $\mathbb{C}$.* Regarding $\mathscr{A}$ as a subring of

$$\mathbb{C}[x_1, \dots, x_d]/(x_1^{s_1} - 1, \dots, x_d^{s_d} - 1) \cong (\otimes_i \mathbb{C}^{s_i}, *),$$

let $F'', G'', H'' \in \otimes_i \mathbb{C}^{s_i}$ be the elements corresponding to F', G' and H', so that $H'' = F'' * G''$. Let $u \coloneqq 2^{-b} F''$, $v \coloneqq 2^{-b} G''$ and $w \coloneqq \mathcal{M}(u, v) = \frac{1}{S} u * v$. Then $\|u\|, \|v\|, \|w\| \leqslant 1$, and $H'' = 2^{2b} S w$. Recalling our choice of working precision $p \coloneqq 6\lceil \log_2 n \rceil$, we may use Proposition 5.3 to compute an approximation $\tilde{w} \coloneqq \tilde{\mathcal{M}}(u, v) \in \otimes_i \tilde{\mathbb{C}}_\circ^{s_i}$ such that $\varepsilon(\tilde{w}) < 2^{\gamma+8} T^2 \log_2 T$ in time $(12T/r)\,\mathsf{M}(3rp) + O(n \log n)$.

Now observe that

$$\|H'' - 2^{2b} S \tilde{w}\| = 2^{2b} S \, \|w - \tilde{w}\| < 2^{2b+\gamma+8-p} T^3 \log_2 T.$$

Since $T < n \leqslant 2^b$ and $T \log_2 T \leqslant T \log_2 n \leqslant Tb < 8n \leqslant 2^{b+3}$, this yields

$$\|H'' - 2^{2b} S \tilde{w}\| < 2^{5b+\gamma+11-p} = 2^{-b+\gamma+11}.$$

But $\gamma < 8b^{2/3} < b - 13$ (as $b \geqslant 4096$), so

$$\|H'' - 2^{2b} S \tilde{w}\| < \tfrac{1}{4}.$$

In particular, we may recover H'' in time $O(Sp^{1+\delta}) = O(n \log n)$ by multiplying each coefficient of $\tilde{w}$ by $2^{2b} S$ and then rounding to the nearest integer. □

Corollary 5.5. *Define $\mathsf{T}(n) \coloneqq \mathsf{M}(n)/(n \log n)$ for $n \geqslant 2$. For $n \geqslant n_0$ we have*

$$\mathsf{T}(n) < \frac{1728}{d - \frac{1}{2}} \, \mathsf{T}(3rp) + O(1).$$

Proof. Dividing (5.8) by $n \log n$ yields

$$\mathsf{T}(n) < \frac{36Tp}{n} \cdot \frac{\log(3rp)}{\log n} \cdot \mathsf{T}(3rp) + O(1).$$

By (5.3) we have $36Tp/n = 216Tb/n < 1728$. By (5.4) we have $r < 2T^{1/d} < 2n^{1/d}$, so

$$\frac{\log(3rp)}{\log n} < \frac{1}{d} + \frac{\log(36b)}{\log n} = \frac{1}{d} + \frac{\log_2(36b)}{\log_2 n} \leqslant \frac{1}{d} + \frac{\log_2(36b)}{b - 1}.$$

Since $b \geqslant 4096$ we have

$$\frac{\log_2(36b)}{b - 1} < \frac{1}{2b^{1/6}} \leqslant \frac{1}{2d^2},$$

and the result follows from the observation that

$$\frac{1}{d} + \frac{1}{2d^2} = \frac{1}{d}\left(1 + \frac{1}{2d}\right) < \frac{1}{d}\left(1 - \frac{1}{2d}\right)^{-1} = \frac{1}{d - \frac{1}{2}}.$$

□

Finally we may prove the main result of the paper.

Proof of Theorem 1.1. We take $d \coloneqq 1729$. According to Corollary 5.5, there is an absolute constant $A > 0$ such that

$$\mathsf{T}(n) < 0.9998 \cdot \mathsf{T}(3rp) + A$$

for all $n \geqslant n_0 = 2^{1729^{12}}$. Define

$$B \coloneqq \max_{2 \leqslant n < n_0} \mathsf{T}(n), \qquad C \coloneqq \max(B, 5000A)$$

(Recall that for $n < n_0$, we use any convenient base-case multiplication algorithm to define $\mathsf{M}(n)$, and hence $\mathsf{T}(n)$.)

We prove by induction that $\mathsf{T}(n) \leqslant C$ for all $n \geqslant 2$. The choice of B ensures that the statement holds for $n < n_0$. Now assume that $n \geqslant n_0$. By (5.4) we have

$$3rp < 6T^{1/d}p < 36n^{1/d}b = 36n^{1/1729}\lceil \log_2 n \rceil < n.$$

By induction,

$$\mathsf{T}(n) < 0.9998C + A \leqslant 0.9998C + 0.0002C = C.$$

Hence $\mathsf{T}(n) = O(1)$ and $\mathsf{M}(n) = O(n \log n)$. □

5.4. **Optimising the dimension threshold.** It is possible to improve the factor $K = 1728$ appearing in Corollary 5.5, at the expense of introducing various technical complications into the algorithm. In this section we outline a number of such modifications that together reduce the constant to $K = 8 + \epsilon$, so that the modified algorithm achieves $\mathsf{M}(n) = O(n \log n)$ for any $d \geqslant 9$ (rather than $d \geqslant 1729$). The techniques described here are similar to those used in [25] to optimise the value of K in the Fürer-type bound $\mathsf{M}(n) = O(n \log n\, K^{\log^* n})$.

(1) We may increase the chunk size from $b = \Theta(\log n)$ to $b = \Theta(\log n \log\log n)$, and then take the working precision to be $p = 2b + O(\log n) = (2 + o(1))b$ rather than $6b$. This improves K by a factor of 3. Note that the $O(\log n)$ term must be chosen large enough to ensure correct rounding at the end of the proof of Proposition 5.4. (We cannot take b as large as $(\log n)^2$, as is done in [25], because then the Gaussian resampling becomes too expensive.)

(2) The choice of b in the previous item allows us to improve the term $3rp$ in Lemma 2.5 to $(2+o(1))rp$, by packing the coefficients together more tightly in the Kronecker substitution step (the hypothesis $r < 2^{p-1}$ must also be tightened somewhat). This improves K by a factor of $3/2$.

(3) In Bluestein's algorithm (see proof of Proposition 3.1), the multiplicand a is invariant, i.e., does not depend on the input vector u. To take advantage of this, we change the basic problem from multiplication of two arbitrary integers to multiplication of an arbitrary integer by a fixed integer. Consequently we save one forward transform in the proof of Proposition 5.3, reducing the factor 12 in (5.7) to 8. This improves K by a factor of $3/2$.

(4) We may choose the primes s_i closer to t_i, so that instead of $T < 2S$ (see (5.6)) we have $T < (1 + o(1))S$. This improves K by a factor of 2. Some care is needed to avoid excessive precision loss in the Gaussian resampling step, due to the larger values of α and γ.

(5) By allowing T to contain small odd prime factors, or alternatively by introducing flexibility into the choice of b, we may improve the choice of T to $(4 + o(1))n/b$ (see (5.3)). This improves K by a factor of 2.

(6) We may change the basic problem from multiplication in $\mathbb{Z}$ to multiplication in $\mathbb{Z}[\mathrm{i}]$. In step (1) of the proof of Proposition 5.4, the chunks F_j and G_j are taken in $\mathbb{Z}[\mathrm{i}]$ instead of $\mathbb{Z}$, and in step (1) of the proof of Lemma 2.5, the evaluations lie in $\mathbb{Z}[\mathrm{i}]$ instead of $\mathbb{Z}$. This improves K by a factor of 4, essentially by eliminating the factor 4 appearing in Lemma 2.5.

(7) We may change the basic problem from multiplication in $\mathbb{Z}[\mathrm{i}]$ to multiplication in $\mathbb{Z}[\mathrm{i}]/(2^n+1)\mathbb{Z}[\mathrm{i}]$. Note that the Kronecker substitution in Lemma 2.5 maps the multiplication modulo y^r+1 naturally onto this problem. This improves K by a factor of 2, because it avoids the degree growth in step (1) in the proof of Proposition 5.4. It also introduces a technical difficulty into that step: to reduce a multiplication modulo 2^n+1 to a polynomial multiplication modulo x^S-1 (or x^S+1), we must split an n-bit integer into S chunks, even though n will not in general be divisible by S. This may be addressed by means of the Crandall–Fagin algorithm [9].

References

1. R. Agarwal and J. Cooley, *New algorithms for digital convolution*, IEEE Transactions on Acoustics, Speech, and Signal Processing **25** (1977), no. 5, 392–410.
2. L. I. Bluestein, *A linear filtering approach to the computation of discrete Fourier transform*, IEEE Transactions on Audio and Electroacoustics **18** (1970), no. 4, 451–455.
3. J. M. Borwein and P. B. Borwein, *Pi and the AGM*, Canadian Mathematical Society Series of Monographs and Advanced Texts, John Wiley & Sons, Inc., New York, 1987, A study in analytic number theory and computational complexity, A Wiley-Interscience Publication. MR 877728
4. A. Bostan, P. Gaudry, and É. 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)
5. R. P. Brent and P. Zimmermann, *Modern computer arithmetic*, Cambridge Monographs on Applied and Computational Mathematics, vol. 18, Cambridge University Press, Cambridge, 2011. MR 2760886
6. S. A. Cook and S. O. Aanderaa, *On the minimum computation time of functions*, Trans. Amer. Math. Soc. **142** (1969), 291–314. MR 0249212
7. J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), 297–301. MR 0178586
8. S. Covanov and E. Thomé, *Fast integer multiplication using generalized Fermat primes*, Math. Comp. **88** (2019), no. 317, 1449–1477. MR 3904152
9. R. Crandall and B. Fagin, *Discrete weighted transforms and large-integer arithmetic*, Math. Comp. **62** (1994), no. 205, 305–324. MR 1185244
10. A. De, P. Kurur, C. Saha, and R. Saptharishi, *Fast integer multiplication using modular arithmetic*, SIAM Journal on Computing **42** (2013), no. 2, 685–699.
11. A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput. **14** (1993), no. 6, 1368–1393. MR 1241591
12. M. Fürer, *Faster integer multiplication*, STOC'07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing, ACM, New York, 2007, pp. 57–66. MR 2402428 (2009e:68124)
13. ———, *Faster integer multiplication*, SIAM J. Comput. **39** (2009), no. 3, 979–1005. MR 2538847 (2011b:68296)
14. J. von zur Gathen and J. Gerhard, *Modern computer algebra*, 3rd ed., Cambridge University Press, Cambridge, 2013. MR 3087522
15. P. Gaudry, A. Kruppa, and P. Zimmermann, *A GMP-based implementation of Schönhage-Strassen's large integer multiplication algorithm*, ISSAC 2007, ACM, New York, 2007, pp. 167–174. MR 2396199
16. I. J. Good, *The interaction algorithm and practical Fourier analysis*, J. Roy. Statist. Soc. Ser. B **20** (1958), 361–372. MR 0102888 (21 #1674)

17. T. Granlund, *The GNU Multiple Precision Arithmetic Library (Version 6.1.2)*, `http://gmplib.org/`.
18. D. Harvey, *Faster truncated integer multiplication*, `https://arxiv.org/abs/1703.00640`, 2017.
19. D. Harvey and J. van der Hoeven, *Faster integer and polynomial multiplication using cyclotomic coefficient rings*, `https://arxiv.org/abs/1712.03693`, 2017.
20. ———, *On the complexity of integer matrix multiplication*, J. Symbolic Comput. **89** (2018), 1–8. MR 3804803
21. ———, *Faster integer multiplication using plain vanilla FFT primes*, Math. Comp. **88** (2019), no. 315, 501–514. MR 3854069
22. ———, *Faster integer multiplication using short lattice vectors*, Proceedings of the Thirteenth Algorithmic Number Theory Symposium, Open Book Series 2 (R. Scheidler and J. Sorenson, eds.), Mathematical Sciences Publishers, Berkeley, 2019, pp. 293–310.
23. ———, *Faster polynomial multiplication over finite fields using cyclotomic coefficient rings*, to appear in Journal of Complexity, 2019.
24. ———, *Polynomial multiplication over finite fields in time* $O(n \log n)$, preprint, 2019.
25. D. Harvey, J. van der Hoeven, and G. Lecerf, *Even faster integer multiplication*, J. Complexity **36** (2016), 1–30. MR 3530637
26. ———, *Faster polynomial multiplication over finite fields*, J. ACM **63** (2017), no. 6, 52:1–52:23.
27. D. R. Heath-Brown, *Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression*, Proc. London Math. Soc. (3) **64** (1992), no. 2, 265–338. MR 1143227 (93a:11075)
28. M. T. Heideman, D. H. Johnson, and C. S. Burrus, *Gauss and the history of the fast Fourier transform*, Arch. Hist. Exact Sci. **34** (1985), no. 3, 265–277. MR 815154 (87f:01018)
29. J. van der Hoeven, *Faster relaxed multiplication*, ISSAC 2014—Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2014, pp. 405–412. MR 3239953
30. D. E. Knuth, *The art of computer programming. Vol. 3*, Addison-Wesley, Reading, MA, 1998, Sorting and searching, Second edition [of MR0445948]. MR 3077154
31. H. J. Nussbaumer, *Fast polynomial transform algorithms for digital convolution*, IEEE Trans. Acoust. Speech Signal Process. **28** (1980), no. 2, 205–215. MR MR563380 (80m:94004)
32. H. J. Nussbaumer and P. Quandalle, *Computation of convolutions and discrete Fourier transforms by polynomial transforms*, IBM J. Res. Develop. **22** (1978), no. 2, 134–144. MR 0471260
33. ———, *Fast computation of discrete Fourier transforms using polynomial transforms*, IEEE Trans. Acoust. Speech Signal Process. **27** (1979), no. 2, 169–181. MR 523618
34. C. H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, Reading, MA, 1994. MR 1251285 (95f:68082)
35. M. S. Paterson, M. J. Fischer, and A. R. Meyer, *An improved overlap argument for on-line multiplication*, (1974), 97–111. SIAM–AMS Proc., Vol. VII. MR 0423875
36. J. M. Pollard, *The fast Fourier transform in a finite field*, Math. Comp. **25** (1971), 365–374. MR 0301966
37. C. M. Rader, *Discrete Fourier transforms when the number of data samples is prime*, Proc. IEEE **56** (1968), no. 6, 1107–1108.
38. J. B. Rosser and L. Schoenfeld, *Approximate formulas for some functions of prime numbers*, Illinois J. Math. **6** (1962), 64–94. MR 0137689
39. A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR 0292344 (45 #1431)
40. M. Sipser, *Introduction to the Theory of Computation*, 2nd ed., Thomson Course Technology, 2006.
41. L. H. Thomas, *Using computers to solve problems in physics*, Applications of digital computers **458** (1963), 42–57.
42. T. Xylouris, *On the least prime in an arithmetic progression and estimates for the zeros of Dirichlet L-functions*, Acta Arith. **150** (2011), no. 1, 65–91. MR 2825574 (2012m:11129)

School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia
E-mail address: d.harvey@unsw.edu.au

CNRS, Laboratoire d'informatique, École polytechnique, 91128 Palaiseau, France
E-mail address: vdhoeven@lix.polytechnique.fr