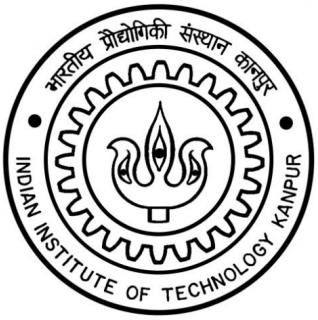
Fast Integer Multiplication

using the Harvey - van der Hoeven algorithm



Bhaskar Goyal
School of Mathematical Sciences, NISER Bhubaneswar

Under the guidance of:
Prof. Nitin Saxena
Department of Computer Science and Engineering, IIT Kanpur

1 Introduction

Integer multiplication is one of the most fundamental problems in computer algebra. Schönhage and Strassen conjectured in 1971 that the product of two n-bit integers may be computed in $O(n \log n)$ bit operations. The problem remained open for decades, until the conjecture was proven to be true by Harvey and van der Hoeven in [2] . In this report, we try to understand the *Harvey-van der Hoeven-Lecerf algorithm* and list down the possible open problems.

We will assume that the reader is familiar with the Schönhage-Strassen integer multiplication algorithm using Fast Fourier Transform [4].

2 Brief Overview

In this section, we will try to explain the the idea behind the algorithm with as few technical details as possible. This is to ensure that the reader has a clear understanding of what the algorithm is trying to achieve. The technical details will be covered in sections 4 and 5.

The main motivation behind this algorithm is that certain polynomial rings inherit highly efficient multiplication algorithms. In particular, rings of the form:

$$\frac{R[x_1, x_2, ..., x_{d-1}]}{(x_1^{t_1} - 1, x_2^{t_2} - 1, ..., x_{d-1}^{t_{d-1}} - 1)}, R := \frac{\mathbb{C}[y]}{(y^r + 1)} \tag{1}$$

where $t_i | r$ for all i, and r is a power if two. We may multiply in this ring by first evaluating each x_i at powers of y^{2r/t_i} to compute the Discrete Fourier Transform, multiplying point-wise in R, and computing the inverse DFT.

The goal is to reduce the problem of integer multiplication to a problem of multiplication in a ring of the form (1). To do so, let $s_1, s_2, ..., s_d$ be distinct primes close to $(n/\log n)^{1/d}$. $d >= 2$ is a dimension parameter which will be fixed later. We start by breaking the integer into roughly $n/\log n$ pieces, each of size roughly $\log n$. Thus, the problem is reduced to that of multiplication in $\frac{Z[x]}{x^{s_1 \cdots s_d} - 1}$. We may use Chinese Remainder Theorem to show that $\frac{Z[x]}{x^{s_1 \cdots s_d} - 1} \cong \frac{Z[x_1, ..., x_d]}{(x_1^{s_1} - 1, ..., x_d^{s_d - 1})}$ with the isomorphism being x goes to $x_1 ... x_d$.

Thus the problem is now reduced to that of computing a multidimensional DFT of size $s_1 \times s_2 \times ... \times s_d$. We want to reduce this to the problem of computing a multidimensional DFT of size $t_1 \times t_2 \times ... \times t_d$. This is done through a technique called **Gaussian Resampling**. This is the key innovation of this paper, and we dedicate the entirety of the next section to this technique.

The idea is as follows: we start with a tuple of size s_i and think of it as lying inside the torus $\mathbb{R}/\mathbb{Z}$. Suppose the tuple is $(j_1, j_2, ..., j_{s_i})$. We plot the points $((1/s_i, j_1), (2/s_i, j_2), ..., (s_i/s_i, j_{s_i}))$. We compute the Gaussians around each of these points (where $s_i/s_i \sim 0$, since we are working in $\mathbb{R}/\mathbb{Z}$), and add them up. Then we evaluate the resulting function at t_i equally spaced points to obtain a tuple of the required size. This analytic nature of the Gaussian Resampling technique is the reason that so far no such algorithm for finite fields is known.

We will show that this process can be done efficiently. On the application of this technique, we end up with a ring of the form (1) and can use synthetic roots of unity to compute the DFT quickly. Now we are ready to dive into the technical details. In section 4 we describe Gaussian Resampling, and in section 5 we explain the algorithm for fast multiplication in (1). Section 6 deals with the selection of parameters, including choosing the primes $s_1, ..., s_d$.

3 Preliminaries

In this section we give some definitions which we have used throughout the paper. We define

$$[x] := \lceil x - 1/2 \rceil$$

i.e. the closest integer function and

$$< x >:= x - [x].$$

The norm $||.||$ is the standard sup norm in the case of vectors, and the operator norm defined by

$$||\mathcal{A}|| = sup_{v \in V} ||\mathcal{A}v||$$

in the case of operators.

We will be working over $\mathbb{C}$ and hence we need to define a precision with which we are working. Our precision will be of p bits i.e. the error will be less than 2^{-p}. Throughout the report, by a "good enough approximation" we mean an approximation $\tilde{v}$ of v such that $|v - \tilde{v}| < 2^{-p}$ in the case of vectors, and an approximation $\tilde{\mathcal{A}}$ of $\mathcal{A}$ such that $||\mathcal{A} - \tilde{\mathcal{A}}|| < 2^{-p}$.

We make some assumptions while proving the results. We list them below:

- Multiplication of two p-bit integers can be done in $O(p^{1+\delta})$ bit operations, where $\delta \in (0, 1/8)$.
- (Real Exponentials) $\omega := e^{\frac{-\pi j}{k}}$ may be approximated in time $O(max(p, log(j+1), logk)^{1+\delta}$
- Some lemmas on efficient approximations of linear maps, bilinear maps and Tensor products. We will quote them when required.

Refer to section 2 of [2] for proofs of these lemmas. We will give the complete statements of these lemmas when they are used in sections 4 and 5.

4 Gaussian Resampling

At the heart of the Gaussian Resampling technique is the Resampling Identity, which states that in some sense, the Resampling Map and the DFT map commute. Before we prove this identity, let us clearly define the DFT and resampling maps:

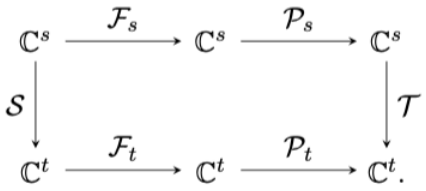


Figure 2

Definition 1 *(Discrete Fourier Transform) The map* $\mathcal{F}_n := \mathbb{C}^n \to \mathbb{C}^n$ *defined by:*

$$(\mathcal{F}_n u)_j := \frac{1}{n}\Sigma_{k=0}^{n-1} e^{-2\pi ijk/n} u_k, u \in \mathbb{C}^n, 0 <= j < n$$

Definition 2 *(Resampling Maps) The maps* $\mathcal{S}\colon \mathbb{C}^s \to \mathbb{C}^t$ *and* $\mathcal{T}\colon \mathbb{C}^s \to \mathbb{C}^t$ *defined by:*

$$(\mathcal{S}u)_k := \alpha^{-1}\Sigma_{j\in\mathbb{Z}} e^{-\pi\alpha^{-2}s^2(\frac{k}{t}-\frac{j}{s})^2} u_j u \in \mathbb{C}^s, 0 <= k < t$$

$$(\mathcal{T}u)_k := \Sigma_{j\in\mathbb{Z}} e^{-\pi\alpha^{2}t^2(\frac{k}{t}-\frac{j}{s})^2} u_j u \in \mathbb{C}^s, 0 <= k < t$$

We also need the following permutation maps:
$\mathcal{P}_s\colon \mathbb{C}^s \to \mathbb{C}^s$ and $\mathcal{P}_t\colon \mathbb{C}^t \to \mathbb{C}^t$ defined by:

$$(\mathcal{P}_s u)_j := u_{tj}$$

$$(\mathcal{P}_t u)_k := u_{-sk}$$

The key idea used in the proof is that the Fourier transform of a Gaussian is again a Gaussian.

Theorem 1 *(Resampling Identity)* $\mathcal{T}\mathcal{P}_s\mathcal{F}_s = \mathcal{P}_t\mathcal{F}_t\mathcal{S}$ *i.e, the following diagram commutes:*

Proof : Given $u \in \mathbb{C}^s$, define the function $f_u : \mathbb{R} \to \mathbb{C}$ by:

$$f_u := \sum_{m\in\mathbb{Z}} u_m g(x - \frac{m}{s}), g(x) = e^{-\pi\alpha^{-2}s^2x^2}$$

Its Fourier series expansion is:

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

where the Fourier coefficients are given by:

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

It is known that the Fourier transform of g(x) on $\mathbb{R}$ is given by:

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

We obtain:

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

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

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

Our goal is to use the resampling identity to compute the s-dimensional DFT via an FFT algorithm involving t-dimensions. For doing this there are 3 broad steps involved:

- Compute $v = \mathcal{S}u$ i.e a t-dimensional vector from an s-dimensional vector
- Compute $\hat{v} = \mathcal{F}_t \mathcal{P}_t v$ i.e the DFT of v
- Solve for $\hat{u}$ in $\mathcal{T}\hat{u} = \hat{v}$

We will show that all of these steps are efficient.

The second step is the computation of DFT in a ring of the form (1) and will be dealt with in section 5. We will focus on the third step which will cover the explanation for the cost of computing of the first step as well.

4.1 Solving for $\hat{u}$ in $\mathcal{T}\hat{u} = \hat{v}$

The difficulty lies in the fact that the matrix for $\mathcal{T}$ does not have an inverse. In fact, it is rectangular. We will start by removing the extra $t - s$ rows to make it a square system. We do this by defining a map $\mathcal{C}\colon \mathbb{C}^t \to \mathbb{C}^s$ defined by:

$$(Cu)_\ell := u_{[t\ell/s]}$$

We write $\mathcal{T}' := \mathcal{C}\mathcal{T}\colon \mathbb{C}^s \to \mathbb{C}^s$. If we can show that $\mathcal{T}'$ is invertible, then $(\mathcal{T}')^{-1}\mathcal{C}$ is the left inverse required to solve the system. After some simplification we can see:

$$(\mathcal{T}'u)_\ell = \Sigma_{h\in\mathbb{Z}} e^{-\pi\alpha^2(\frac{th}{s}+\beta_\ell^2)} u_{\ell+h}$$

where $\beta_\ell :=< \frac{t\ell}{s} >$, $\ell \in \mathbb{Z}$. Note that $|\beta| <= \frac{1}{2}$.

To approximate the inverse of $\mathcal{T}'$, we will try to write it as a sum of a diagonal matrix and another matrix with only a small number of relevant terms. To do this, define $\mathcal{D}\colon \mathbb{C}^s \to \mathbb{C}^s$ as:

$$(\mathcal{D}u)_\ell := e^{\pi\alpha^2\beta_\ell^2}$$

and $\mathcal{N}' := \mathcal{T}'\mathcal{D}\colon \mathbb{C}^s \to \mathbb{C}^s$ as:

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

Setting $\mathcal{E}\colon = \mathcal{N} - \mathcal{I}$, we get:

$$(\mathcal{N}u)_\ell = \Sigma_{h\in\mathbb{Z}-\{0\}} e^{-\pi\alpha^2((\frac{th}{s}+\beta_\ell^2)-\beta_{\ell+h}^2)} u_{\ell+h}$$

where $u \in \mathbb{C}^s, 0 <= \ell < s$ Since the matrices $\mathcal{I}$ and $\mathcal{E}$ commute, we may write

$$\mathcal{N}^{-1} = \mathcal{I} - \mathcal{E} + \mathcal{E}^2 - ...$$

and $(\mathcal{D}\mathcal{N}^{-1}\mathcal{C})$ is the required left inverse for $\mathcal{T}$.

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

Our goal is to show that all these matrices may be computed quickly. This is done using the earlier assumptions/lemmas. We state them here for clarity: For

all these lemmas, let p be the working precision in bits, $\mathcal{E}(\tilde{A})$ be the worst case error in approximation of A by $\tilde{A}$, and $\mathcal{C}(\tilde{\mathcal{A}})$ be the worst case cost of evaluating $\tilde{\mathcal{A}}$ on a single vector.

Lemma 1: (Multiplication in $\mathbb{C}$). Given as input $u, v \in \tilde{\mathbb{C}}_0$, in time $O\left(p^{1+\delta}\right)$ we may compute an approximation $\tilde{w} \in \tilde{\mathbb{C}}_\circ$ for $w := uv \in \mathbb{C}_\circ$ such that $\varepsilon(\tilde{w}) < 2$.

Lemma 2: (Real exponentials, negative case). Let $k \geqslant 1$ and $j \geqslant 0$ be integers, and let $w := e^{-\pi j/k} \in \mathbb{C}_\circ$. 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 $O\left(\max(p, \log(j+1), \log k)^{1+\delta}\right)$.

Lemma 3: (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 $w := 2^{-\sigma} e^{\pi j/k} \in \mathbb{C}_0$. 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 $O\left(\max(p, \log k)^{1+\delta}\right)$.

Lemma 4: (Tensor products). Let R be a coefficient ring of dimension $r \geqslant 1$. Let $d \geqslant 1$, let $m_1, \ldots, m_d, n_1, \ldots, n_d \geqslant 2$, and put $M := \prod_i \max(m_i, n_i)$. For $i \in \{1, \ldots, d\}$, let $\mathcal{A}_i : R^{m_i} \to R^{n_i}$ be an R-linear map with $\|\mathcal{A}_i\| \leqslant 1$, and let $\tilde{\mathcal{A}}_i : \tilde{R}_\circ^{m_i} \to \tilde{R}_\circ^{n_i}$ be a numerical approximation. Let $\mathcal{A} := \otimes_i \mathcal{A}_i : \otimes_i R^{m_i} \to \otimes_i R^{n_i}$. (Note that automatically $\|\mathcal{A}\| \leqslant 1$.)

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

$$\mathrm{C}(\tilde{\mathcal{A}}) \leqslant M \sum_i \frac{\mathrm{C}\left(\tilde{\mathcal{A}}_i\right)}{\max(m_i, n_i)} + O(Mrp \log M)$$

The first step is to give an algorithm to efficiently approximate $\mathcal{D}' = \frac{\mathcal{D}}{2^{2\alpha^{2^{-2}}}}$. This algorithm is pretty straightforward, with the analysis being done using the above lemmas. Each element of the diagonal of $\mathcal{D}'$ is of the form $e^{\pi\alpha^2\beta_\ell^2}/2^{2\alpha^{2\alpha^{2^{-2}}}}$. The rational part of the exponent may be written as $\alpha^2\beta_\ell^2 = \alpha^2 < tl/s >^2 = \alpha^2 k_l/s^2$ for some integer $k_l < s^2/4$. We can now use lemma 3 to say that $e^{\pi\alpha^2\beta_\ell^2}/2^{2\alpha^{2\alpha^{2^{-2}}}}$ may be approximated in $O(p^{1+\delta})$. Since there are s terms and using Lemma 1, we can conclude that $\mathcal{D}'u$ may be computed in time $O(sp^{1+\delta}) = O(tp^{1+\delta})$.

The next step is to show that $\mathcal{S}'u = \frac{S}{2}u$ may be approximated efficiently. By definition:

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

Let $m := \left\lceil p^{1/2} \right\rceil \alpha$, and consider the truncated sum

$$T_k := \sum_{\left|j-\frac{sk}{t}\right|<m} \frac{1}{2}\alpha^{-1} e^{-\pi\alpha^{-2}\left(j-\frac{sk}{t}\right)^2} u_j.$$

We can actually show that T_k is a good enough approximation for $(\mathcal{S}'u)_k$, essentially implying that there are only about $2m$ relevant terms in the series

corresponding to each of the s elements in the vector $(\mathcal{S}'u)$. Hence proceeding as in the case of $\mathcal{D}'$, we get that $\mathcal{S}'$ may be approximated in $O(sp^{1+\delta}m) = O(tp^{3/2+\delta}\alpha)$ operations.

We will now show that $\mathcal{E}$ may be approximated efficiently. This is very similar to the previous case.

$$(\mathcal{E}u)_\ell = \sum_{h\in\mathbb{Z}\setminus\{0\}} e^{-\pi\alpha^2\left(\left(\frac{th}{s}+\beta_\ell\right)^2-\beta_{\ell+h}^2\right)}u_{\ell+h}$$

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

$$T_\ell := \sum_{\substack{h\in\mathbb{Z}\setminus\{0\}\\ |h|\leqslant m}} e^{-\pi\alpha^2\left(\left(\frac{th}{s}+\beta_\ell\right)^2-\beta_{\ell+h}^2\right)}u_{\ell+h}$$

Again it is possible to show that T_ℓ is a good enough approximation for $(\mathcal{E}u)_\ell$. Hence there are only about $2m$ relevant terms in the series corresponding to each of the s elements in the vector $(\mathcal{E}u)$. Proceeding as before we get that $\mathcal{E}'$ may be approximated in $O(sp^{1+\delta}m) = O(tp^{3/2+\delta}\alpha^{-1})$ operations.

The final step is to approximate $\mathcal{J}' = \mathcal{N}^{-1}/2$. To do this, define $v = \frac{u}{2}$. Then

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

Let $n := \lceil p/(\alpha^2\theta) \rceil$ where $\theta := t/s - 1$ It can be shown that only about n terms of the above series are required for a good enough approximation of $\mathcal{J}'u$. So we need only about n invocations of the previous algorithm approximating $\mathcal{E}$. Assuming $\theta >= p/\alpha^4$, we get $n <= \alpha^2 + 1$. So the total cost is $O(tnp^{3/2+\delta}\alpha^{-1}) = O(tp^{3/2+\delta}\alpha)$.

Theorem 2 *(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 $\theta := t/s - 1 > 0$, and assume that $\theta \geqslant p/\alpha^4$. Then (i) There exist linear maps $\mathcal{A} : \mathbb{C}^s \to \mathbb{C}^t$ and $\mathcal{B} : \mathbb{C}^t \to \mathbb{C}^s$ 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 $\tilde{\mathcal{A}} : \tilde{\mathbb{C}}_\circ^s \to \tilde{\mathbb{C}}_\circ^t$ and $\tilde{\mathcal{B}} : \tilde{\mathbb{C}}_\circ^t \to \tilde{\mathbb{C}}_\circ^s$ such that $\varepsilon(\tilde{\mathcal{A}}), \varepsilon(\tilde{\mathcal{B}}) < p^2$ and $\mathrm{C}(\tilde{\mathcal{A}}), \mathrm{C}(\tilde{\mathcal{B}}) - O\left(tp^{3/2+\delta}\alpha + tp\log t\right)$.*

Simply take $\mathcal{A} = \mathcal{S}'$, which is approximated in $O(tp^{3/2+\delta}\alpha)$ For $\mathcal{B}$, take $\mathcal{B} = \mathcal{P}_s^{-1}\mathcal{D}'\mathcal{J}'\mathcal{C}\mathcal{P}_t$ Time taken for approximating $\mathcal{B}$ is $O(tp\log t) + C(\mathcal{D}') + C(\mathcal{J}')$.

We can now use Lemma 4 to show the main theorem of this section:

Theorem 3 *(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 $T := t_1 \cdots t_d$.*

Let α be an integer in the interval $2 \leqslant \alpha < p^{1/2}$. For each i, let $\theta_i := t_i/s_i - 1$, and assume that $\theta_i \geqslant p/\alpha^4$.

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

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

Moreover, we may construct numerical approximations $\tilde{\mathcal{A}} : \otimes_i \tilde{\mathbb{C}}_\circ^{s_i} \to \otimes_i \tilde{\mathbb{C}}_\circ^{t_i}$ and $\tilde{\mathcal{B}} : \otimes_i \tilde{\mathbb{C}}_0^{t_i} \to \otimes_i \tilde{\mathbb{C}}_0^{s_i}$ such that $\varepsilon(\tilde{\mathcal{A}}), \varepsilon(\tilde{\mathcal{B}}) < dp^2$ and

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

5 Multidimensional DFTs of powers of two sizes

The goal of Gaussian Resampling was to reduce the problem of computing multidimensional DFT of $s_1 \times s_2 \times ... \times s_d$ to a problem of computing a multidimensional DFT of size $t_1 \times t_2 \times ... \times t_d$, i.e. a multiplication problem in a ring of the form (1). In this section we will give an algorithm for fast computation of this DFT. We have:

$$u \in \frac{\mathbb{C}[x_1, x_2, ..., x_{d-1}, y]}{(x_1^{t_1} - 1, x_2^{t_2} - 1, ..., x_{d-1}^{t_{d-1}} - 1, y^r + 1)}$$

where $t_i | r$ for all i. Note that in this ring y^{2r/t_i} is a t_ith root of unity, and we may use this to compute a DFT of size t_i by evaluating x_i at the powers of y^{2r/t_i}. Doing this for all $i \in \{1, 2, ..., d-1\}$, we get $t_1 t_2 ... t_{d-1}$ many polynomials

$$u_{i_1,...,i_{d-1}} := u(y^{2ri_1/t_1}, ..., y^{2ri_{d-1}/t_{d-1}}) \in \frac{\mathbb{C}[y]}{(y^r + 1)}$$

for all $i_k \in \{1, 2, ..., t_{d-1}\}$. Hence this problem is reduced to a one dimensional DFT of size r which we may compute using Bluestein's FFT algorithm [1] in time

$$O(r \log r) = O(t_d \log t_d)$$

. Since multiplication with y is simply a bit shift with a cyclic wraparound after $2r$ terms, computing all the polynomials takes time $O(pt_1 t_2 ... t_{d-1})$. So the total time taken is:

$$O((t_1 t_2 ... t_{d-1})(t_d \log t_d)) + O(pt_1 t_2 ... t_{d-1}) = O(Tp \log t_d)$$

Since t_d was chosen of $(O(T)^{1/d})$, we get the time complexity of the multidimensional DFT as $O(\frac{1}{d} pT \log T)$.

Now we will use this DFT algorithm to compute the product in ring (1). The forward and inverse DFT take time $O(\frac{1}{d} pT \log T)$. Now we need to compute the pointwise product, which takes time $\frac{4T}{r} M(rp)$ where $M(.)$ denotes the time complexity of integer multiplication. (This is true as each pointwise

product is between elements from the ring $\frac{\mathbb{C}[y]}{(y^r+1)}$, with there being T/r such multiplications. Hence we get the time complexity of multiplication in 1 as:

$$\frac{4T}{r}M(rp) + O(\frac{1}{d}pT\log T) \tag{3}$$

6 Parameters and Complexity Analysis

Let

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

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

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

Define

$$\alpha := \left\lceil \left(12d^2 b\right)^{1/4} \right\rceil.$$

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

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

Let T be the unique power of 2 in the interval:

$$4n/b \leqslant T < 8n/b$$

then $T < n \leqslant 2^b < 2^p$. We may compute a factorisation $T = t_1 t_2 ... t_d$ in time $(\log n)^{O(1)}$ (as T is a power of 2).

We also need primes $s_1, s_2, ..., s_d$. This can be done through Eratosthenes' sieve. The existence of such primes is guarantied by the following lemma (refer to section 5 of [2] for proof):

Lemma 5: Let $\eta \in \left(0, \frac{1}{4}\right)$ and let $x \geqslant e^{2/\eta}$. Then there are at least $\frac{1}{2}\eta x/\log x$ primes q in the interval $(1-2\eta)x < q \leqslant (1-\eta)x$.

The overall complexity of the Algorithm is:

$$\frac{4T}{r}M(rp) + O(\frac{1}{d}pT\log T) + O\left(dTp^{3/2+\delta}\alpha + Tp\log T\right) + O(Tp^{1+\delta})$$

which we get from (2) and (3). The $O(Tp^{1+\delta})$ factor is due to the cost of scaling. Note that:

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

and $T = O(n/\log n)$, $p = O(\log n)$which gives us the final complexity of integer multiplication:

$$M(n) < \frac{4T}{r}M(rp) + O(n\log n)$$

Define $T(n) = M(n)/(n\log n)$, then dividing the previous equation by $(n\log n)$ gives:

$$T(n) = \frac{4Tp}{n}.\frac{\log(rp)}{\log n}.T(rp) + O(1)$$

Then $4Tp/n = 24Tb/n < 1728$. Also $r < 2n^{1/d}$, so

$$\frac{\log(rp)}{n} = \frac{\log r/2 + \log(12b)}{\log n} < 1/d + \frac{\log 12b}{b-1}$$

Since $b \geqslant 4096$ and $d \leqslant b^{1/12}$,

$$\frac{\log 12b}{b-1} < \frac{1}{2d^2}$$

Then

$$\frac{1}{d} + \frac{1}{2d^2} = \frac{1}{d}(1 - \frac{1}{2d})^{-1} = \frac{1}{d - 1/2}$$

and hence

$$T(n) < \frac{1728}{d-1/2}T(rp) + O(1)$$

Now we are in a position to prove the main theorem of this paper:

Theorem 4 *There is an integer multiplication algorithm achieving*

$$M(n) = O(n \log n)$$

Proof: According to the previous equation, there exists a constant $A > 0$ such that

$$T(n) < \frac{1728}{d-1/2}T(rp) + A$$

for all $n \geqslant n_0$, where $n_0 := 2^{d^{12}}$. We now take d:= 1729, then for all $n \geqslant n_0$ we have

$$T(n) < 0.9998T(rp) + A.$$

Define:

$$B := \max_{2 \leqslant n < n_0} T(n), C := \max(B, 5000A)$$

(for $n < n_0$, we use a base case multiplication algorithm which defines $M(n)$ and hence $T(n)$) .

We will use induction on n to show that $T(n) \leqslant C$ for all $n \geqslant 2$. The choice of B ensures that the theorem holds for $n < n_0$. For $n \geqslant n_o$ we have

$$rp < 12T^{1/d}b = 12n^{1/1729}\lceil \log_2 n \rceil < n$$

By induction we have

$$T(n) < 0.9998C + A \leqslant C.$$

Hence, $T(n) = O(1)$ and $M(n) = O(n \log n)$.

7 Polynomial Multiplication over Finite Fields

The algorithm we presented in the report is an unconditional $O(n \log n)$ algorithm for integer multiplication. In [3], Harvey and van der Hoeven describe another algorithm for integer multiplication in time $O(n \log n)$, but it is conditional on some open conjectures in number theory. The advantage with this algorithm is that it may be adapted to get $O(n \log n)$ algorithm for polynomial multiplication over finite fields $\mathbb{F}_q[x]$. The unconditional bound for this remains at $O(n \log n 4^{\log * n})$.

8 Conclusion and Open Problems

The algorithm is able to achieve an unconditional upper bound of $O(n \log n)$ on the cost of integer multiplication. However, the Gaussian Resampling technique is highly complicated and due to its analytic nature, so far it has not been adapted for fast polynomial multiplication in $\mathbb{F}_q[x]$. This is the most fundamental open problem related to the paper.

Another interesting problem lies in Gaussian Resampling, particularly in solving for $\hat{u}$ in $\mathcal{T}\hat{u} = \hat{v}$. We have approximated $\mathcal{N}^{-1}$ efficiently by making the assumption $\theta \geqslant p/\alpha^4$. On the other hand, we need to have θ as close to 0 as possible to keep the size of the DFT small. An alternative method is LU factorisation of $\mathcal{N}$ which may be more efficient, allowing us to remove the assumption that $\theta \geqslant p/\alpha^4$. However, the error analysis becomes much more complicated in this case.

References

[1] L. Bluestein. A linear filtering approach to the computation of discrete fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, 1970.

[2] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193(2):563 – 617, 2021.

[3] David Harvey and Joris van der Hoeven. Polynomial multiplication over finite fields in time O(n log n). *Journal of the ACM (JACM)*, April 2022.

[4] A. Schönhage and V. Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7(3):281–292, 1971.