MATHEMATICS OF COMPUTATION
Volume 77, Number 262, April 2008, Pages 1199–1221
S 0025-5718(07)01956-4
Article electronically published on November 15, 2007

Il y eut pourtant, dans l' vieux Paris
Un honnête homme sans malice
Brûlant d' contempler le nombril
D' la femme d'un agent de police...[1]

To François and Pierrick

FAST CONVOLUTIONS MEET MONTGOMERY

PREDA MIHĂILESCU

Abstract. Arithmetic in large ring and field extensions is an important problem of symbolic computation, and it consists essentially of the combination of one multiplication and one division in the underlying ring. Methods are known for replacing one division by two short multiplications in the underlying ring, which can be performed essentially by using convolutions.

However, while using *school-book multiplication*, modular multiplication may be grouped into $2\mathsf{M}(\mathbf{R})$ operations (where $\mathsf{M}(\mathbf{R})$ denotes the number of operations of one multiplication in the underlying ring), the *short multiplication problem* is an important obstruction to convolution. It raises the costs in that case to $3\mathsf{M}(\mathbf{R})$. In this paper we give a method for understanding and bypassing this problem, thus reducing the costs of ring arithmetic to roughly $2\mathsf{M}(\mathbf{R})$ when also using fast convolutions. The algorithms have been implemented with results which fit well the theoretical prediction and which shall be presented in a separate paper.

1. Introduction

Let $\mathbf{R}$ be a commutative ring and $(\nu) \subset \mathbf{R}$ an ideal. The problem we are interested in is the efficient reduction of arithmetic in $\mathbf{R}/(\nu)$ to arithmetic in $\mathbf{R}$. The base operation is assumed to be multiplication in $\mathbf{R}$, and we denote its runtime by $\mathsf{M}(\mathbf{R})$. For simplicity, we explicitly restrict our attention to the cases $\mathbf{R} = \mathbb{Z}$, $\nu = n$ and $\mathbf{R} = \mathbb{Z}/(n \cdot \mathbb{Z})[X]$, $\nu = p(X)$, with $n > 1$ an integer. However the analysis and solutions we provide are generic and allow for generalizations, e.g. in towers of extensions.

The multiplication in $\mathbb{Z}$ can be performed efficiently with the help of various procedures sharing the idea of *fast convolutions.* In $\mathbb{Z}/(n \cdot \mathbb{Z})$, multiplications are followed by a division, and no algorithms are known which allow one to accelerate a division to a similar performance as convolution does for multiplications. It is thus interesting to trade divisions for multiplications. A similar problem is posed

Received by the editor August 27, 2004 and, in revised form, April 18, 2006.
2000 *Mathematics Subject Classification.* Primary 11D61.
Part of this work was developed while the author was a guest of LiX, the *Laboratoire d'Informatique de l'École Polytechnique, Palaiseau*, in March 2004.

[1] "Le nombril de la femme d'un agent de police", by Georges Brassens. Free translation: *There was in old Paris / an honnest man without meanness / Eager to contemplate / A policeman's wife's navel.*

for the ground ring $\mathbf{R} = \mathbb{Z}/(n \cdot \mathbb{Z})[X]$, assuming the operations in $\mathbf{r} = \mathbb{Z}/(n \cdot \mathbb{Z})$ as given. Several algorithms are known which do exactly this: they trade one division for two *short (half-) multiplications*.

When using the *naive (or: school-book)* multiplication method, the short multiplications can be combined to the (run time-) equivalent of one single long multiplication. This is not possible with fast multiplication, and the total time required for a modular operation is in this case equivalent to *three* long multiplications. Recent research on the problem of *short multiplication* [Mu], [HZ] showed that it is encountered in various algorithmic contexts and it is hard to improve upon. This opens the question as to whether modular multiplication is dependent on this problem or not. In other words: *is it possible to obtain a fast modular multiplication which costs essentially two long multiplications, rather than three?*

The present paper answers this question affirmatively, by providing a modification of the setting in which multiplications are traded for divisions. The new algorithm is generic and can be combined with various fast convolutions; as examples, we treat the FFT and Toom-Cook convolutions in detail. We show in particular a range of applications in which the second is preferable. Plainly: asymptotically optimal algorithms have been known for decades, and improving upon constants is still an open problem in the field. This paper improves a constant (by a factor of roughly 1/3) which is correlated to the above mentioned problem of short multiplication.

The cases $\mathbf{R} = \mathbb{Z}$ and $\mathbf{R} = \mathbb{Z}/(n \cdot \mathbb{Z})[X]$ have been traditionally addressed independently, yet it shows that the respective solutions for one of them can be adapted to the other one. In the year 1973, Volker Strassen [St] indicated how to reduce polynomial division to power series division, and this was later shown independently by Sieveking [Si] and Kung [Ku] to admit an efficient solution based on formal Newton iteration: the Sieveking-Kung algorithm. The main ingredient consists in a precomputation (which can be done using Newton iteration) of $X^{-d} \bmod p(X)$, where $d = \deg(p(X))$ and $(\nu) = (p(X))$ as above. This trades divisions for multiplications and—not trivial, as we shall see—truncations modulo X^d.

More than ten years after Strassen's seminal paper, Montgomery [Mo] gave an algorithm for multiplication in rings $\mathbb{Z}/(n \cdot \mathbb{Z})$ with large $n \in \mathbb{N}$, in which modular division is replaced by two short multiplications, using an alternative representation of remainder classes, described below. This is currently the *state of the art* for modular reduction in long integer arithmetic. As shown in [GG], Sieveking-Kung may also be used for long integer arithmetic (Corollary 9.9) while Montgomery may be adapted for polynomial arithmetic (exercise 9.12, Algorithm 9.35).[2]

The long integer multiplication ($\mathbf{R} = \mathbb{Z}$) has complexity

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

In the case of polynomial arithmetic, when both $d, n \to \infty$, the ring multiplication can be performed for instance with FFT methods in

$$\mathsf{M}(\mathbf{R}) = O\left(d \log(n) \cdot \log(d \log(n))\right)$$

[2]Since both algorithms can be used in either case (integer and polynomial ring modular multiplication), the title might as well have used the names of Sieveking and Kung. Our choice is connected to the fact that Montgomery is more explicit in suggesting a modified representation of ring elements, an idea which we push in some contexts further than he did.

[GG]. In practice, with two parameters, various relations between n and d can lead to different optimal solutions. For instance, one will not use the same techniques in a field of characteristic two and high degree as in a field of very large characteristic and relatively small degree.[3]

Both algorithms perform a modular multiplication in $3 \cdot \mathsf{M}(\mathbf{R})$, where $\mathsf{M}(\mathbf{R})$ is the time for one multiplication of integers of the size of n–for long integer multiplication–and the time for multiplication of two polynomials of degree at most d, for extension rings. This is a general bound [GG], independent of the particular implementation of the base multiplications. It is however true for both algorithms that when combined with school-book multiplication, an adequate data treatment reduces the time to only $2 \cdot \mathsf{M}(\mathbf{R})$.

When using the very setting of Sieveking and Kung, resp. Montgomery, while fast convolution techniques are speeding up the base multiplication, it is hard to obtain the same factor 2. The core problem may be seen as connected with the truncation. This problem is also called the *short product* problem: given two polynomials $f(X), g(X)$ of degree $< d$, compute $f(X) \cdot g(X) \mod X^d$ using fast convolutions. Performing this task faster than the mere product $f(X) \cdot g(X)$ appears to be a hard problem [Mu], [HZ]. There is little hope to reduce the time of the short multiplication to half of the time of the full one. This obstruction is common for modular long integer and for polynomial multiplication, when using convolutions.

The central idea of this paper is to investigate methods for adapting the Montgomery representation for polynomial modular reduction, which are more efficient than the Sieveking-Kung algorithm. It turns out that replacing truncation modulo X^d by one which is adapted to the special type of convolution used for the polynomial multiplication is a the key idea for achieving the goal announced in the abstract. This leads to a *generic* algorithm for Montgomery reduction with convolutions: we call this the ConvREDC procedure, in analogy to Montgomery's initial REDC. We show that the algorithm is very useful for long integer arithmetic too, when this is done using FFT.

The idea was first proposed in the thesis [Mi], and the core of the algorithms we shall present is in that thesis. McLaughlin noticed in the context of long integer multiplication using Montgomery, that the choice $R = 2^N \pm 1$ is preferable when used together with small Winograd convolutions and derives some explicit *Frameworks*, corresponding to various convolution polynomials of this kind. The FFT method is also mentioned, in the context of polynomial multiplication.

Our generic procedure allows the use of any linear convolution, but here we investigate the FFT and Toom-Cook convolutions explicitly. The fact that the last algorithm is preferable to FFT in a quite large range of magnitudes for (d, n) is also in some respects a new result. The extensions occurring in *primality proving* typically belong to this range of magnitude: this was the initial motivation in [Mi].

The plan of the paper is as follows. In the second section we give a general treatment of fast polynomial multiplications using fast convolution techniques, following Winograd [Wn] and Blahut [Bl]. In the third section we describe Montgomery's procedure REDC in a generalized form (see also [GG], Exercise 9.12) and derive ConvREDC, our generic adaptation to fast convolutions. In the fourth section we

[3]The latter is frequent in *primality testing* and can be treated efficiently with the Toom-Cook algorithm [Mi].

consider the specialization of this generic procedure to the Fast Fourier transform and show that it improves upon Shoup's implementations [NTL], [Li] of modular polynomial arithmetic with FFT. In the fifth section we investigate long modular integer arithmetic with FFT and ConvREDC. In section six we consider the convolution of Toom-Cook, first with its classical spectral transforms [Kn], which are quadratic in d, but very simple. Asymptotically fast multi-point evaluation—which is the spectral transform for Toom-Cook—was first developed by Borodin and Moenck in [BM]. It was recently studied by Bostan, Lecerf and Schost [BLS] in a more general frame, which both improves upon the constants in the asymptotically best version and allows performing multi-point evaluation with *any* convolution. Consequently, this enables gradual transition from naive evaluation to fast convolution.

1.1. **The problem-setting.** Let $n \geq 2$ be an integer and $\mathbf{R}$ a commutative ring, which may be thought of as either $\mathbf{R} = \mathbb{Z}$ or as being a finite $\mathbb{Z}/(n \cdot \mathbb{Z})$-algebra, not necessarily different from $\mathbb{Z}/(n \cdot \mathbb{Z})$. In particular, if n is a prime, $\mathbf{R}$ may be the prime field $\mathbb{F}_n$ or a finite extension thereof: the algorithms which we develop in this paper apply to these rings as particular cases. As already mentioned, we shall restrict for simplicity our attention to the two cases $\mathbf{R} = \mathbb{Z}$ and $\mathbf{R} = \mathbb{Z}/(n \cdot \mathbb{Z})[X]$ and start with the focus on the second; thus let $\mathbf{r} = \mathbb{Z}/(n \cdot \mathbb{Z})$ and $\mathbf{R} = \mathbf{r}[X]$.

Let $p(X) \in \mathbf{R}$ be a polynomial of degree d and $\mathbf{A} = \mathbf{R}/(p(X))$. If $a \in \mathbf{R}$, we use the notation of von zur Gathen [GG] and denote by $a \text{ rem } p$ the unique polynomial of minimal degree in the residue class of a; likewise $a \div b$ denotes the Euclidean quotient of the polynomial a divided by b. We assume that arithmetic in $\mathbf{r}$ is optimal in some sense and give the design of efficient algorithms for the arithmetic in $\mathbf{A}$. Let us consider the elementary operations in $\mathbf{r}$ as complexity-units and denote by $\mathsf{M}(\mathbf{r}), \mathsf{A}(\mathbf{r})$ respectively the times required for a multiplication and an addition in $\mathbf{r}$; the multiplication of an element of $\mathbf{r}$ by a short integer (one machine word) will take $\mathsf{S}(\mathbf{r})$ operations. From a *certain distance*, one may consider that $\mathsf{A}(\mathbf{r})$ and $\mathsf{S}(\mathbf{r})$ can be neglected with respect to $\mathsf{M}(\mathbf{r})$. We shall take this perspective in section six, in order to give crude estimates for a new variant of the Toom-Cook algorithm.

Typical elements $a, b \in \mathbf{A}$ are represented in a minimal representant system by polynomials $a(X) = a \text{ rem } p(X), b(X) = b \text{ rem } p(X)$ with coefficients in $\mathbf{R}$ and degree $\leq d - 1$. The minimality of these representants is with respect to the degree, naturally; other representants of residue classes are possible and may be useful. The product $c = a \times b \in \mathbf{A}$ can be computed by building the product $c_0(X) = a(X) \times b(X)$ and then reducing it modulo $p(X)$; the question of how to best avoid the modular reduction by additional multiplications is the central topic of this paper.

2. Fast polynomial multiplication using convolutions

Let $\mathfrak{R}$ be a commutative, euclidean ring and $a, b \in \mathfrak{R}[X]$. For reasons of transparency we consider in this section the problem of efficiently computing the product $a(X) \times b(X)$ separately. We shall see however in the next section that important additional gains can be achieved, when considering the modular product $a(X) \times b(X) \text{ rem } p(X)$ as one single operation, rather than splitting it into a multiplication and a modular reduction step.

Let $a(X) = \sum_{i=0}^{n} a_i \cdot X^i$ and $b(X) = \sum_{b=0}^{m} b_j \cdot X^j$, with $a_i, b_j \in \mathbf{R}$ and $m, n < d$. Then

$$c(X) = \sum_{k=0}^{n+m} c_k \cdot X^k \quad \text{with} \quad c_k = \sum_{l=0}^{k} a_l \cdot b_{k-l}.$$

Methods allowing a fast *simultaneous* computation of all the coefficients c_k are called *fast convolution algorithms*. A large class of such algorithms share the following fundamental scheme, such as initially described by Winograd [Wn]; see also [Bl]. Let $C(X)$ be a polynomial of degree $2D > m+n$. Since the degree of $C(X)$ is larger than that of $c(X)$, it follows that $c(X) = a(X) \times b(X) = a(X) \times b(X) \text{ rem } C(X)$. One thus knows $c(X)$ if its image modulo $C(X)$ is known. The idea of fast convolutions is to split $C(X) = \prod_{s=1}^{l} C_s(X),\ l \leq 2D$, then compute

$$(1)\quad a_s(X) = a(X) \text{ rem } C_s(X) \quad \text{and} \quad b_s(X) = b(X) \text{ rem } C_s(X), \quad s = 1, 2, \ldots, l,$$

and multiply

$$(2)\qquad c_s(X) = a_s(X) \times b_s(X) \text{ rem } C_s(X).$$

Finally, $c(X)$ can be retrieved by using the Chinese Remainder Theorem:
(3)

$$c_s(X) = \sum_{s=1}^{l} e_s(X) \cdot c_s(X) \quad \text{where} \quad e_s(X) \equiv \delta_{s,t} \mod C_t(X), \quad t = 1, 2, \ldots, l.$$

We denote the operations (1) and (3) by *direct* and *and inverse* ***spectral*** *transforms*, in accordance with the roots of these ideas in the theory of Fourier and Laplace transforms. If the spectral transforms are *cheap*, then the l multiplications in (2) are the main step of the convolution algorithm—they trivially are so if $C(X)$ is irreducible. The aim is thus to choose highly composite $C(X)$ for which the cost of the spectral transforms is minimal to some respect. Certainly, good convolutions are expected to use linear polynomials $C(X)$. Some important convolution methods are the Fast Fourier Transform [GG], Winograd's *small convolutions* [Wn], and the Toom-Cook convolution [Co], [To]. We refer the reader wanting to know more about the large variety of convolution techniques designed for various contexts to the magnificent book of Blahut [Bl].

In this paper we concentrate on two particular convolution methods which both share the property that $C(X)$ splits into linear factors, albeit in different rings:

$$(4)\qquad C(X) = \prod_{s=1}^{2D} (X - \mu_s).$$

We designate convolutions for which $C(X)$ splits into linear factors by *linear convolutions*. As a consequence of (4), the direct spectral transform amounts to the simultaneous evaluation of the $2D$ (constant) values

$$(5)\qquad a_s(X) = a(\mu_s), \quad b_s(X) = b(\mu_s), \quad s = 1, 2, \ldots, 2D.$$

The values μ_s can thus be considered as interpolation points.

3. The modular reduction

Again we let $\mathbf{R} = \mathbf{r}[X] = \mathbb{Z}/(n \cdot \mathbb{Z})[X]$ and $\mathbf{A} = \mathbf{R}/(p(X))$. One multiplication in $\mathbf{A}$ consists of a polynomial multiplication followed by the reduction of the result modulo the polynomial $p(X)$. The second operation may well be the more time consuming: divisions are mostly so.

Here we present in a general frame the algorithm of Montgomery, which trades in multiplications for divisions. Based on this, we describe in the second part of this section a new, generic algorithm, which combines fast convolutions with Montgomery modular reduction.

3.1. **Montgomery's REDC.**

Let $\mathfrak{R}$ be a commutative Euclidean ring endowed with an order relation and let $\nu \in \mathfrak{R}$. The aim is to perform modular multiplications in $\mathfrak{N} = \mathfrak{R}/(\nu\mathfrak{R})$ avoiding divisions for the modular reduction. For this, let $R \in \mathfrak{R}$ with $R > \nu, (R, \nu) = 1$. The elements $a \in \mathfrak{N}$ will be represented by $\rho(a) = a \cdot R \text{ rem } \nu$. Thus

$$\rho(a \cdot b) = \rho(a) \cdot \rho(b) \cdot R^{-1} \text{ rem } \nu.$$

The core problem is now to compute $Z \cdot R^{-1} \text{ rem } \nu$ for inputs $Z < R \cdot \nu$. Assuming that $V \in \mathfrak{N}$ with $V \cdot \nu \equiv -1 \mod R$ is precomputed, this is done by the following procedure of Montgomery, which takes an input $0 < Z < R \cdot \nu$ and outputs $T = Z \cdot R^{-1} \text{ rem } \nu$:

```
**REDC(Z)**    *(* Montgomery[4] *)*

    0. $Z = A \cdot B$,
    1. $M = ((Z \text{ rem } R) \cdot V) \text{ rem } R$,
    2. $T = (Z + M \cdot \nu) \div R$,
    3. if $T \geq \nu$, then $T = T - \nu$. Return $T$.
```

Here is a brief review of the proof of integrity of this procedure. By definition of V, $Z \cdot V \cdot \nu \equiv -Z \mod R$. Thus $\nu \cdot M = (\nu \cdot (Z \text{ rem } R) \cdot V) \text{ rem } R \equiv Z \cdot V \cdot \nu \equiv -Z \mod R$, and $Z + M \cdot \nu \equiv 0 \mod R$. It follows that T is indeed an integer. The definition of T also yields: $T \cdot R = Z + M \cdot \nu$, i.e. $T \equiv Z \cdot R^{-1} \mod \nu$. Finally we have to show that T is indeed a *remainder*, so $T < \nu$. But $Z < R \cdot \nu$ while by definition $M < R$ and thus $Z + M \cdot \nu < 2 \cdot R \cdot \nu$. The value of T in point 2 is consequently $T < 2\nu$. Step 3 finally assures that the output is $T < \nu$. Note that this holds when $\mathfrak{R} = \mathbb{Z}$ and $\nu = n \in \mathbb{N}$, the purpose for which the algorithm was designed. We shall show below that when ν, R are polynomials, step 3 can be avoided, since the condition for the input is stronger.

The algorithm replaces the reduction modulo ν by two short multiplications and three divisions by R. Compared to an implementation of modular reduction using division, this is a gain in the case in which divisions by R are practically for free. In the initial setting for long integer arithmetic (thus $\mathfrak{R} = \mathbb{Z}$), one took $R = 2^{nw}$, where w is the machine word length, as an adequate choice.

[4]The original procedure REDC consists of steps 1–3; step 0 is added in order to make it more precise that, in most contexts, one has to think of Z as being the product $A \cdot B$. This will also allow a transparent operation count for various applications.

3.2. **An optimized REDC for fast convolutions.** The cost of a modular multiplication using Montgomery is in general $3\mathsf{M}(\mathfrak{R})$, i.e. three multiplications in $\mathfrak{R}$ [GG]. An appropriate goal would be however $2\mathsf{M}(\mathfrak{R})$, which is achieved in combination with school-book multiplication. One wishes to exploit the fact that the multiplications in steps 1 and 2 of REDC are partial and leave place for some savings—ideally, they should be performed at the cost of one single operation.

The choice of an adequate value for R plays an essential role. With naive polynomial multiplication, a good choice is $R = X^{d+1}$, used by Sieveking and Kung: division by the polynomial X^{d+1} is easy. It is also the analog of division by $R = 2^{nw}$ in Montgomery's choice for binary arithmetic. The choice is not very useful when the polynomial multiplication is done with convolutions. Indeed, in this case, it is in the *spectral space* that truncation has to be effective, if one wishes to reduce the cost for truncated multiplications. Thus by adapting truncation to the spectral space, one avoids the *short multiplication* obstruction mentioned in the introduction. We present a method for choosing R in combination with convolution techniques, which is adapted in the above sense. We define for this purpose a special representation of the data which is directly connected to the particular convolution polynomial $C(X)$ that one uses for long multiplication.

Definition 1. Let $\mathbf{R}, p(X)$ and $\mathbf{A}$ be as before and let

$$C(X) = \prod_{s=1}^{2D} (X - \mu_s) \in \mathbf{X} \quad \text{with} \quad D > d, \quad (C(X), p(X)) = 1$$

be a polynomial defining a linear convolution.

Let $C(X) = R^{\perp}(X) \cdot R^{\top}(X)$ be a splitting of $C(X)$ in two coprime factors, such that $\deg(R^{\perp}(X)) = \deg(R^{\top}(X)) = D > d$ and let $\mathcal{R} = \{\mu_1, \ldots, \mu_{2D}\} = \mathcal{R}^{\perp} \cup \mathcal{R}^{\top}$ be the partition such that

$$R^{\perp}(X) = \prod_{\mu_s \in \mathcal{R}^{\perp}} (X - \mu_s) \quad \text{and} \quad R^{\top}(X) = \prod_{\mu_s \in \mathcal{R}^{\top}} (X - \mu_s).$$

For $g(X) \in \mathbf{r}[X]$ we define

$$\begin{array}{rcl} \sigma(g) & = & \{g(\mu_s) : \mu_s \in \mathcal{R}\} \quad \text{and} \\ \sigma^{\perp}(g) & = & \{g(\mu_s) : \mu_s \in \mathcal{R}^{\perp}\}, \\ \sigma^{\top}(g) & = & \{g(\mu_s) : \mu_s \in \mathcal{R}^{\top}\}. \end{array} \tag{6}$$

We may also write, for simplicity $\sigma(g)_s = g(\mu_s)$. If $\deg(g) \leq 2(d+1)$ this set of data uniquely determines $g(X)$ by interpolation.[5] Furthermore, we let $R = R^{\perp}(X)$ in REDC and define the Montgomery representation of the class $a(X) + (p(X)) \in \mathbf{A}$ by

$$\rho(a) = a(X) \cdot R^{\perp}(X) \text{ rem } p(X) \in \mathbf{A}, \quad \forall a(X) \in \mathbf{R}[X]. \tag{7}$$

Finally we define $\sigma' = \sigma \circ \rho$. We say that the elements of $\mathbf{A}$ are in σ' representation, if they are given by the set of data $\sigma'(a)$. One easily verifies that this set uniquely defines an element of $\mathbf{A}$. The elements $a \in \mathbf{A}$ thus have the combined representation

$$a \mapsto \rho(a)(X) \in \mathbf{R} \mapsto \sigma'(a) = \sigma\left(\rho(a)(X)\right).$$

[5] The choice of the degree of C is good for most applications; in a context in which sparing single multiplications may be relevant, it is possible with more care to reduce this degree by units. This would uselessly complicate our general presentation.

A final important *assumption* concerning the convolution method is that, given the values $\sigma^{\perp}(g)$ of a polynomial $g(X)$ with $\deg(g) \leq d$, the computation of $\sigma^{\top}(g)$ is easy, and vice versa. Formally, we define the following *shift operators*:

$$\mathfrak{S}^{+}\left(\sigma^{\perp}(g)\right) = \sigma^{\top}(g) \quad \forall g \in \mathbf{R} \quad \text{with} \quad \deg(g) < D, \tag{8}$$

$$\mathfrak{S}^{-}\left(\sigma^{\top}(g)\right) = \sigma^{\perp}(g) \quad \forall g \in \mathbf{R} \quad \text{with} \quad \deg(g) < D. \tag{9}$$

We now show that the choice of R is adequate for convolutions. Here is how the operations rem $R^{\perp}$ and $\div R^{\perp}$ act upon $\sigma(g)$. If

$$g(X) = q(X) \cdot R^{\perp}(X) + r(X) \tag{10}$$

is the Euclidean division of g by $R^{\perp}$, then $r(X) = g(X)$ rem $R^{\perp}(X)$ is a polynomial of degree $\leq d$ and is uniquely determined by its values at $d+1$ interpolation points. For $\mu_s \in \mathcal{R}^{\perp}$ we have $R^{\perp}(\mu_s) = 0$ so $g(\mu_s) = (g \text{ rem } R^{\perp})(\mu_s)$, for $\mu_s \in \mathcal{R}^{\perp}$ and thus

$$\sigma^{\perp}(g) = \sigma^{\perp}\left(g \text{ rem } R^{\perp}\right). \tag{11}$$

So the *truncation* g rem $R^{\perp}$ is *almost* for free. It is sometimes necessary to know the values $(g \text{ rem } R^{\perp})(\mu_s)$ for $\mu_s \in R^{\top}$. For this, one needs the shift operator $\mathfrak{S}^{+}$. Suppose now that $g(X)$ is divisible by $R^{\perp}(X)$; from (10),

$$\begin{aligned} \left(g(X) \div R^{\perp}(X)\right)(\mu_s) &= q(\mu_s) = g(\mu_s)/R^{\perp}(\mu_s) \quad \forall \mu_s \in \mathcal{R}^{\top} \quad \text{thus} \\ \sigma^{\top}(g) &= \sigma^{\top}(R^{\perp}) \cdot \sigma^{\top}\left(g \div R^{\perp}\right). \end{aligned} \tag{12}$$

Thus *division* by $R^{\perp}(X)$ is also *almost* for free too, since the values $\left(a(X) \div R^{\perp}(X)\right)(\mu_s)$ for $\mu_s \in \mathcal{R}^{\perp}$ can be gained using the $\mathfrak{S}^{-}$ operator.

Let $a, b \in \mathbf{A}$ be given by $\sigma'(a), \sigma'(b)$ and $v(X)$ above be precomputed and also given by $\sigma(v)$. We wish an algorithm which takes this input and returns $\sigma'(c) = \sigma'(a \cdot b)$. The first step is naturally to compute

$$z_s = \sigma'(a)_s \cdot \sigma'(b)_s \quad \text{for all} \quad s \in \mathcal{R}. \tag{13}$$

This is the core of a multiplication using fast convolution and corresponds to step 0 in REDC. The direct spectral transform is avoided by the σ-representation. Now z_s is the σ-representation of a polynomial $z(X) \in \mathbf{r}[X]$, of degree at most $2(d-1)$ and such that

$$z(X) \equiv (R^{\perp})^2(X) \cdot a(X) \cdot b(X) \equiv R^{\perp}(X) \cdot \rho(a \cdot b)(X) \mod p(X).$$

We need the equivalent of the procedure REDC to divide out the factor $R^{\perp}(X)$. The algorithm below accomplishes this task. Note that $z(X) = \rho(a) \cdot \rho(b)$ verifies $\deg(z) < 2d - 1$; this eliminates step 3 of that procedure.

```
Algorithm ConvREDC    (* REDC for Convolutions *)

Input: $\sigma'(a), \sigma'(b), a, b \in \mathbf{A}; \sigma(v), \sigma(p)$.
Output: $\sigma'(c) = \sigma'(a \cdot b)$

  1. $\sigma_s(z) = \sigma'_s(a) \cdot \sigma'_s(b)$, for $s \in \mathcal{R}$.
  2. $m(\mu_s) = z(\mu_s) \cdot v(\mu_s)$ for $\mu_s \in \mathcal{R}^{\perp}$. This yields $\sigma^{\perp}(m)$.
  3. $\sigma^{\top}(m) = \mathfrak{S}^{+}\left(\sigma^{\perp}(m)\right)$.
  4. $t(\mu_s) = \left(z(\mu_s) + m(\mu_s) \cdot p(\mu_s)\right)/R^{\perp}(\mu_s)$ for all $\mu_s \in \mathcal{R}^{\top}$.
     This yields $\sigma^{\top}(t)$.
```

5. $\sigma^{\perp}(t) = \mathfrak{S}^{-}\left(\sigma^{\top}(t)\right)$.
6. Set $\sigma'(c) = \{t(\mu_s) : \mu_s \in \mathcal{R}\}$ and return $\sigma'(c)$.

The divisions by $R^{\perp}(\mu_s)$ in step 4 can be optimized, and will in general take less than $\mathsf{M}(\mathbf{r})$, but certainly not more. The central step of the algorithm is the evaluation of the shift operators, which is comparable in both directions. We shall denote by $\theta(\mathfrak{S})$ an upper bound for the time for these evaluations—a time which depends on the convolution used. The run time for one modular multiplication in $\mathbf{A}$, in σ-representation, using (13) and ConvREDC is herewith

$$T \leq 5 \cdot d \cdot \mathsf{M}(\mathbf{r}) + 2\theta(\mathfrak{S}). \tag{14}$$

The first term in (14) is close to the complexity-theoretic lower bound for this task, while the last two terms are the times taken by the evaluations of $\mathfrak{S}^{\pm}$. These evaluations are thus crucial for the performance of the algorithm. Note that the σ-representation together with Montgomery's algorithm allow deferring spectral transformations to the end of a chain of computations. This is fundamental for the Toom-Cook method, but less relevant for FFT. If one has to compute an expression $(\beta_1, \beta_2, \ldots, \beta_j) = E(\alpha_1, \alpha_2, \ldots, \alpha_k)$, where the α's are inputs and the β's outputs, and if M is the number of $\mathbf{A}$-multiplications for the evaluation of E, then the k direct and j inverse spectral transforms for α, β are distributed as overhead over all the M ring operations. When M *is large compared* $j + k$, as typically in the case of exponentiations or multiplications on elliptic curves, this is an important gain.

The next theorem states the properties of the algorithm ConvREDC:

Theorem 1. *Let $n \in \mathbb{N}_{\geq 2}$, $\mathbf{r} = \mathbb{Z}/(n \cdot \mathbb{Z})$, $\mathbf{R} = \mathbf{r}[X]$ and $p(X) \in \mathbf{R}$ be a polynomial of degree $d > 1$. Let $\mathbf{A} = \mathbf{R}/(p(X))$ and a linear convolution polynomial*

$$C(X) = R^{\perp}(X) \cdot R^{\top}(X), \quad (C(X), p(X)) = 1$$

be given, so that $\deg(C) \geq 2(d+1)$, with $R^{\perp}(X), R^{\top}(X)$ a splitting of $C(X)$ in coprime factors of degree $> d$. Let the σ- and σ'-representations be defined by Definition 1 and suppose that a polynomial $v(X) \in \mathbf{R}[X]$ with

$$v(X) \cdot p(X) \equiv -1 \mod R^{\perp}(X)$$

is precomputed together with $\sigma(p), \sigma(v)$. Then, for any linear convolution $C(X)$ as above, the algorithm ConvREDC produces, on input $\sigma'(a), \sigma'(b)$, the σ'-representation of the modular product: $\sigma'(c) = \sigma'(ab)$, where the product is taken in $\mathbf{A}$. The number of operations is bounded by (14).

Proof. The integrity of this procedure is easily deduced from the one of the general procedure REDC. In the first step one performs the multiplication which will be the input for the actual reduction algorithm. In the next two steps one computes a complete set $\sigma(m)$ of interpolation values of m, using (11). This corresponds to step 1 of REDC. The following two steps produce $\sigma(t)$, using (12); they correspond to step 2 of REDC. Finally, we show that $\deg(t) < d$, which makes step 3 superfluous. The polynomial m is a remainder of $R^{\perp}$ and has degree $\deg(m) < D$. Then $\deg(m(X) \cdot p(X)) < D + d$ in step 2 of REDC, and since $\deg(z) < 2d - 1$ it follows that $\deg(t \cdot R^{\perp}) < D + d$. But $R^{\perp}(X)$ has degree D, so the last inequality implies $\deg(t) < d = \deg(p(X))$, which completes the proof. □

The algorithm ConvREDC is *generic*, since it can be combined with any linear convolution; the generalization to non-linear convolutions is at hand, but of little present interest. We shall investigate in the next sections in more detail the way ConvREDC works together with the FFT and Toom-Cook convolutions.

4. ConvREDC with FFT

For FFT one will reasonably choose $C(X) = X^{2D} - 1$ and $R^{\perp}(X) = X^D - 1$, $R^{\top}(X) = X^D + 1$, where $d + 1 \leq D < 2(d + 1)$.

Remark 1. The choice of D is due to the fact that traditionally one wishes D to be a power of 2. This leads to the unpleasant feature that the FFT complexity becomes roughly a step function of d, rather than a continuously growing one. Recently van der Hoeven [vH] has shown that this effect can be avoided. For this he uses the non-recursive version of FFT and chooses as interpolation points for a polynomial of degree d the first d values computed by the non-recursive version of FFT(D), where D is the least power of 2 which is larger than d.

In FFT mode, $\mathfrak{S}^{\pm}$ can be evaluated by an inverse spectral transform followed by a direct transform, both of dimension D, rather than $2D$. The major advantage of FFT is that the operators $\mathfrak{S}^{\pm}$ can be performed in $O(d \cdot \log(d) \cdot \mathsf{M}(\mathbf{r}))$ operations; this is important when d is large.

4.1. **FFT representation.**

Here we shall follow the guidelines set by Shoup in [Sh] for implementing polynomial modular arithmetic over finite fields, since his implementation [NTL] is considered *the* state of the art. The roots of unity $\mu_s = \omega^s$ are in general not in $\mathbf{r}$, so the FFT operations will be multiplications on numbers in a ring $\mathbf{F} = \mathbb{Z}/(N \cdot \mathbb{Z})$ which contains the required roots of unity (which *affords* $2D$*-point FFT*, in [GG] terminology): we shall write for simplicity, in this section, $\mathsf{M}(n) = \mathsf{M}(\mathbf{r})$ and $\mathsf{M}(N) = \mathsf{M}(\mathbf{F})$. Working over $\mathbf{F}$ corresponds to embedding $g(X) \in \mathbf{r}[X]$ in $\mathbb{Z}[X]$ and then the result in $\mathbf{F}[X]$. In order to retrieve meaningful results, N must be larger than the coefficients of the polynomial product in $\mathbb{Z}[X]$.

Following Shoup [Sh], one lets N be a product of *small primes* p_i (machine word length, typically), all of which are $p_i \equiv 1 \mod 2D$: so all rings $\mathbb{Z}/(p_j \cdot \mathbb{Z})$ contain $2D$ - th roots of unity and thus so does $\mathbf{F} = \mathbb{Z}/(N \cdot \mathbb{Z})$. In order to have sign information, one uses the interval $[-\lfloor N/2 \rfloor, -\lfloor N/2 \rfloor + N - 1]$ as the set of representants for $\mathbb{Z}/(N \cdot \mathbb{Z})$.

Definition 2. We shall say that $g(X) \in \mathbb{Z}[X]$ is $\mathbb{Z}/(N \cdot \mathbb{Z})$-*normal*, if

$$g(X) = \sum_i g_i \cdot X^i \quad \text{and} \quad |g_i| < N/2.$$

In the chosen representation for $\mathbb{Z}/(N \cdot \mathbb{Z})$, the normal polynomials are characterized by:

$$g(X) = g(X) \text{ rem } N. \tag{15}$$

Conversely,

Lemma 1. *If the polynomials $G_1(X), G_2(X), \ldots, G_r(X)$ have the product*

$$G(X) = G_1(X) \cdot G_2(X) \cdot \ldots \cdot G_r(X)$$

and there is a chain of intermediate products $F_1(X), F_2(X), \ldots, F_{r'}(X) = G_r(X)$ which are all $\mathbb{Z}/(N \cdot \mathbb{Z})$-normal, then the product can be evaluated using FFT in $\mathbb{Z}/(N \cdot \mathbb{Z})$ without loss of information.

Proof. The polynomials

$$F_1(X) \text{ rem } N, F_2(X) \text{ rem } N, \ldots, F_{r'}(X) \text{ rem } N = G_r(X) \text{ rem } N$$

can be computed using FFT-convolutions in $\mathbb{Z}/(N \cdot \mathbb{Z})$. Since they all are normal, we have $F_j(X) = F_j(X)$ rem N and finally $G(X) = G(X)$ rem N, which completes the proof. ☐

A polynomial $g(X) \in \mathbb{Z}/(n \cdot \mathbb{Z})[X]$ will be embedded in $\mathbb{Z}[X]$, the image being a polynomial $G(X) \in \mathbb{Z}[X]$ with coefficients $|g_i| < n/2$. In order to compute products in $\mathbb{Z}/(n{\cdot}\mathbb{Z})[X]$ with FFT in $\mathbb{Z}/(N{\cdot}\mathbb{Z})$ using Lemma 1, one must choose N sufficiently large, so that normality of the occurring products is preserved. If the chain has length r and the degrees of all factors $\deg(G_j(X)) \leq d$, it suffices to choose

$$N > T(r) = 2(d+1)^{r-1}(n/2)^r : \tag{16}$$

one easily verifies by induction on r that $T(r)$ is an universal upper bound for the coefficients of products of r polynomials in $\mathbb{Z}/(n \cdot \mathbb{Z})[X]$, when the coefficients are in signed minimal representation, and thus have absolute value $\leq n/2$. An often better alternative is to reduce modulo n the coefficients of each product. This way $T(2)$ is sufficient, but one has to perform more FFT steps for the renormalization.

4.2. **Modular reduction in FFT mode.**

We let $\mathbf{r} = \mathbb{Z}/(n \cdot \mathbb{Z})$ and $p(X) \in \mathbf{r}[X]$ be as before, a polynomial of degree d. The convolution polynomial is $C(X) = X^{2D} - 1, R^{\perp}(X) = X^D - 1$ and $R^{\top}(X) = X^D + 1$, with $d < D \leq 2d$. With Montgomery reduction, the factor $z(X)$ in step 1 may have degree $2d > \deg(z) > d$. In view of (16) we have two alternatives:

A. Compute $m(X)$ without previous reduction of $z(X)$ modulo n; in this case $m(X)$ is the product of three polynomials, and we must choose $r = 3$ in (16).

B. First reduce $z(X)$ modulo n and then compute $m(X)$. This costs two additional FFT operations, but reduces r to 2.

Note that although $N > dn^2$, the arithmetic in $\mathbb{Z}/(N \cdot \mathbb{Z})$ is particularly appealing and it may be that $\mathsf{M}(N) < \mathsf{M}(n)$ for large n. Indeed, multiplications in $\mathbb{Z}/(N{\cdot}\mathbb{Z})$ require only $\log(N)/w$ word multiplications, where w is the machine word length (the log will always be in base 2, when concrete constants are implied). One FFT operation requires a chain of $\frac{1}{2} \cdot D \log(D) \log(N)/w$ word multiplications and two Chinese Remainder operations. The price of CRT decomposition and recomposition modulo N is paid only once per FFT operation, so for D sufficiently large (e.g. $O(\log(N))$), we have indeed $\mathsf{M}(N) < \mathsf{M}(n)$, when the cost of one $\mathbb{Z}/(N \cdot \mathbb{Z})$ multiplication is averaged over an FFT(D) operation. Briefly: the arithmetic in $\mathbb{Z}/(N \cdot \mathbb{Z})$ is *linear* in $\log N$, when CRT transforms are negligible. Increasing the size of N by a factor c essentially amounts to increasing the number of factors p_j by that constant, and, finally increasing the FFT operation time by the same constant factor. The increase of N is from dn^2 to d^2n^3. The time for one FFT operation is thus increased by at most

$$\lambda = \frac{3}{2} + \frac{1}{2} \cdot \frac{1}{1 + 2\frac{\log(n)}{\log(d)}} \sim \frac{3}{2}, \tag{17}$$

where the larger n/d, the closer λ approaches $3/2$. The degrees of the polynomials are directly controlled by the procedure ConvREDC, so it is only the size of the coefficients that need particular attention in FFT mode.

Remark 2. Switching from $N_3 \sim d^2n^3$ to $N_2 \sim dn^2$ is particularly easy: one disregards part of the small primes used for N_3. This is practical, since after reducing $m(X) \mod n$ we only have products of two polynomials, so such a switch is allowed.

In either case, let $\omega \in \mathbb{Z}/(N \cdot \mathbb{Z})$ be a primitive $2D$-th root of unity. Then $X^D + 1 = -\left((\omega X)^D - 1\right)$. This allows one to perform operations modulo $X^D + 1$ using FFT(D)-operations. We give the applications in the following

Lemma 2. *Let ω be a primitive $2D$ - th root of unity and*

$$\mathbf{M}_D = \left(\omega^{2ij}\right)_{i,j=0}^{D-1}.$$

Let $G(X) \in \mathbf{R}[X]$ have degree $\deg(G) \leq D$ and $\vec{g}$ be its padded coefficient vector of length D. Consider the diagonal matrix $\mathbf{\Delta} = \operatorname{diag}\left(1, \omega, \ldots, \omega^{D-1}\right)$. Finally, let $\vec{\gamma} = \left(G(\omega^{1+2j})\right)_{j=0}^{D-1}$ be the spectrum of $G(X) \mod X^D + 1$. Then

$$\begin{aligned} \vec{\gamma} &= \mathbf{M}_D \times \mathbf{\Delta} \times \vec{g}, \\ \vec{g} &= \mathbf{\Delta}^{-1} \times \mathbf{M}_D^{-1} \times \vec{\gamma}. \end{aligned} \tag{18}$$

Proof. One verifies from the definition that

$$\mathbf{M}_D \times \mathbf{\Delta} = \left(\omega^{i\cdot(2j+1)}\right)_{i,j=0}^{D-1}.$$

This together with the definition of $\vec{\gamma}$ proves the first line in (18). The second line is solving the linear system in the first line with respect to $\vec{g}$. □

In terms of σ-representation, this means, for $a(X) \in \mathbf{A}$:

$$\begin{aligned} \mathbf{M}_D \times \vec{a} &= \sigma^{\perp}(a), \\ \mathbf{M}_D \times \mathbf{\Delta} \times \vec{a} &= \sigma^{\top}(a). \end{aligned} \tag{19}$$

We now consider the implementation of the *shift operators*. The operator $\mathfrak{S}^+$ is given by

$$\mathfrak{S}^+ = (\mathbf{M}_D \times \mathbf{\Delta}) \circ (\operatorname{mod} n) \circ \mathbf{M}_D^{-1}, \tag{20}$$

in the natural execution sequence of operations from right to left. The modular reduction in the center of expression (20)—denoted by (mod n)—is used for renormalization. However, since it produces the output, along with computing $\mathfrak{S}^-(t)$, after renormalization one should also recompute the top part which served as input for $\mathfrak{S}^-$. Thus:

$$\mathfrak{S}_t^- = (\mathbf{M}_D | \mathbf{\Delta} \times \mathbf{M}_D) \circ (\operatorname{mod} n) \circ (\mathbf{M}_D \times \mathbf{\Delta})^{-1}. \tag{21}$$

Here the symbol $''|''$ stands for concatenation of two matrices. The first two steps in (21) already give the coefficients of $\rho(c)$, but the integrity of the procedure ConvREDC requires that the output be the full σ'-representation of $a \cdot b$, so one needs to apply $\mathbf{M}_D$ at the end.

Note that $R^{\perp}(\mu_s) = \mu_s^{d+1} - 1 = -2$ for all $\mu_s \in R^{\top}$ and thus the division by $R^{\perp}(\mu_s)$ in step 4 is trivial; due to linearity it may even be performed after the inverse spectral transform. We let $v(X) \in \mathbb{Z}/(n \cdot \mathbb{Z})[X]$ be defined by $v(X) \cdot p(X) \equiv -1 \bmod R^{\perp}(X)$ and suppose that both $v(X), p(X)$ are given by their precomputed σ-representation, like in ConvREDC. We write for simplicity $\mathbf{M} = \mathbf{M}_D$ and $\mathbf{M}_+ = \mathbf{\Delta} \times \mathbf{M}_D$, when $\omega \in \mathbb{Z}/(N \cdot \mathbb{Z})$ with $N = N_2 \sim T(2)$ and replace $\mathbf{M}$ by $\mathbf{M}'$, when $N = N_3 \sim T(3)$. We also assume that N_2 divides N_3.

In the first variant we shall require that the σ-representation that the algorithm maintains is with respect to $N_3 \sim T(3)$. In order to save one FFT operation, we note that after step 3 of ConvREDC, simple precision in FFT is sufficient, so we introduce a switch between FFT modulo N_3 and FFT modulo N_2. With this, ConvREDC translates into:

```
Algorithm FFTConvREDC   (* ConvREDC for FFT, $\sigma$ *)

Input: $\sigma'(a)(X), \sigma'(b)(X); \sigma(p(X)), \sigma(v(X))$. The input is modulo
$N_3$.
Output: $\sigma'(a \cdot b)$.

  1. $\sigma_s(z) = \sigma'_s(a) \cdot \sigma'_s(b)$ for all $\mu_s \in \mathcal{R}$.
  2. $m(\mu_s) = z(\mu_s) \cdot v(\mu_s)$ for $\mu_s \in \mathcal{R}^{\perp}$. This yields $\sigma^{\perp}(m)$
     mod $N_3$.
  3. Let $\vec{m}^{\perp} = \mathbf{M}'_D \times (\sigma^{\perp}(m(X)))$ and reduce the vector modulo
     $n$.
     Switch to $N_2$-representation (i.e. start using matrices mod-
     ulo $N_2$) and let $\sigma^{\top}(m(X)) = \mathbf{M}_+ \times \vec{m}^{\perp}$.
  4. $t(\mu_s) = (z(\mu_s) + m(\mu_s) \cdot p(\mu_s)) / (-2)$ for all $\mu_s \in \mathcal{R}^{\top}$. This
     yields $\sigma^{\top}(t)$.
  5. Let $\vec{t}^{\top} = \mathbf{M}_+ \times (\sigma^{\top}(t(X)))$ and reduce the vector modulo $n$.
     Switch back to $N_3$-representation for the output. Let $\sigma'^{\perp}(c) =$
     $\mathbf{M}'_D \times \vec{t}$ and $\sigma'^{\top}(c) = \mathbf{M}'_+ \times \vec{t}$. Return $\sigma'(c)$.
```

For the time analysis, we shall write FFT(D) for the time for multiplication of a vector with entries of the size of $N_2 = N \sim dn^2$ by the $D \times D$ matrix $\mathbf{M}_D$: this is known to be $O(D \cdot \log(D) \cdot \mathsf{M}(N))$ operations, and more precisely $\frac{1}{2} D \log(D) \cdot \mathsf{M}(N)$, if D is a power of 2. According to the discussion above, the FFT modulo N_3 for the same size matrix will then cost $\lambda \times FFT(D)$ operations. Using switches, the evaluation of $\mathfrak{S}^+$ costs $(\lambda + 1) \times FFT(D) + 2\mathsf{M}(n)$, where $2\mathsf{M}(n)$ accounts for the reduction modulo n and the one of $\mathfrak{S}^-$ takes $(2\lambda+1) \times FFT(D) + 2\mathsf{M}(n)$ operations. Steps 1, 2 and 4 require $4 \times D \times \mathsf{M}(N)$ operations. Steps 3 and 5 are $\mathfrak{S}^{\pm}$ evaluations requiring a total of $(3\lambda + 2) \times FFT(D) + 2d \times \mathsf{M}(n)$ operations. The total operation count is thus:

$$T_f \sim (3\lambda + 2) \times FFT\,(D) + 4D \times \mathsf{M}(N) + 4d \times \mathsf{M}(n). \tag{22}$$

In order to reduce the size of the operands to $N \sim dn^2$, we now focus on strategy B and add a *partial* renormalization of $z(X)$. Since only the values $\sigma^{\perp}(z)$ are involved in a further multiplication, it is their size that needs to be controlled. For this, we retrieve the coefficients of $z(X)$ rem $R^{\perp}(X)$, reduce them modulo n and then recompute the values $\sigma^{\perp}(z)$ with the renormalized coefficients. Writing $\mathfrak{N}(z)$ for this operation, we have:

$$\mathfrak{N} = \mathbf{M} \circ (\operatorname{mod} n\ ) \circ \mathbf{M}^{-1}. \tag{23}$$

One notices the structural analogy with the shift operators; the time required is, accordingly, $2 \times FFT(D) + 2d \times \mathsf{M}(n)$. The ring $\mathbb{Z}/(N \cdot \mathbb{Z})$ is now smaller, the factor $\lambda = 1$ for the evaluation of $\mathfrak{S}^{\pm}$ and no switches are necessary. Certainly, we assume that input and output are in the σ-representation for *small* $N = N_2$. This leads to

```
**Algorithm SFFTConvREDC** (* ConvREDC for FFT, $\sigma$, small $N$ *)

Input: $\sigma'(a)(X), \sigma'(b)(X); \sigma(p(X)), \sigma(v(X))$.
Output: $\sigma'(a \cdot b)$.

  1. $\sigma_s(z) = \sigma'_s(a) \cdot \sigma'_s(b)$ for all $\mu_s \in \mathcal{R}$.
  2. Renormalize: $\sigma^{\perp}(z) = \mathfrak{N}^{\perp}(\sigma^{\perp}(z))$.
  3. $m(\mu_s) = z(\mu_s) \cdot v(\mu_s)$ for $\mu_s \in \mathcal{R}^{\perp}$. This yields $\sigma^{\perp}(m)$.
  4. Let $\sigma^{\top}(m(X)) = \mathfrak{S}^{+}\left(\sigma^{\perp}(m(X))\right)$.
  5. $t(\mu_s) = \left(z(\mu_s) + m(\mu_s) \cdot p(\mu_s)\right)/(-2)$ for all $\mu_s \in \mathcal{R}^{\top}$. This
     yields $\sigma^{\top}(t)$.
  6. Let ${\sigma'}^{\top}(c) = \sigma^{\top}(t)$ and ${\sigma'}^{\perp}(c) = \mathfrak{S}^{-}\left(\sigma^{\top}(t)\right)$. Return $\sigma'(c)$.
```

The cost of the short variant of SFFTConvREDC is

$$T_s \sim 7 \times FFT(D) + 4D \times \mathsf{M}(N) + 4d \times \mathsf{M}(n). \tag{24}$$

4.3. **Special cases and preconditioning.** In the context of exponentiations, one encounters squarings and repeated multiplications with a fixed factor. It is important to use the specificity of these operations in order to save some FFT operations. For squaring, the σ-representation allows no savings, but these are implicitly provided by the possibility to avoid the computation of the spectra of the input data.

For multiplication by a fixed factor we can save on the computation of $m(X)$ by using the following identity, which holds for any triple of polynomials $f(X), g(X), h(X) \in \mathbf{R}[X]$ (here $R(X) = R^{\perp}(X)$):

$$\begin{aligned}((f(X) \cdot g(X) \text{ rem } R(X)) \cdot h(X) \text{) rem } R(X) \\ = (f(X) \cdot (g(X) \cdot h(X) \text{ rem } R(X)) \text{) rem } R(X).\end{aligned} \tag{25}$$

The proof is evident: both terms are remainders modulo $R(X)$ by definition and they are congruent to $f(X) \cdot g(X) \cdot h(X) \mod R(X)$. They must be equal by the uniqueness of the remainder. Applying (25) to the definition of $m(X)$ in step 2 of ConvREDC, we see that this can be replaced by:

$$m(X) = a(X) \cdot \left(b(X) \cdot v(X) \text{ rem } R^{\perp}(X)\right) \text{ rem } R^{\perp}(X).$$

If $b(X)$ is frequently used, one can store the spectral representation of $\phi(X) = b(X) \cdot v(X)$ rem $R^{\perp}(X)$. More precisely, we assume that the product polynomial $\phi(X)$ of degree $\deg(\phi) \leq d$ has been previously reduced modulo n, so its coefficients are all $< n$. In this case all polynomials involved have coefficients dominated by dn^2 and one can save the partial renormalization in SFFTConvREDC, thus reducing the costs in this case to $\sim 5FFT(D)$. There are less savings in FFTConvREDC, although all computations can be made modulo N_2. On output, the σ'-representation is required modulo N_3, and computing it will require one more FFT operation than SFFTConvREDC.

This appears to compare quite well with the state of the art implementation [NTL] of this method and even brings some gains for arbitrary d. The gains were confirmed by an implementation on top of [NTL] and agree with the theoretical prediction; this will be described in a separate paper. Here is a crude comparison, based only on the number of FFT steps and using the data in [Sh]. For FFTConvREDC, we use the scaling factor $\lambda = 3/2$; as indicated by (17), this is a very good

Table 1. Number of FFT operations at a glance

Operation (Number of $FFT(D)$)	**NTL-Shoup**	**SFFTConvREDC**	**FFTConvREDC**
General modular multiplication	12	-	-
Modular squaring	10	7	6.5
Preconditioned modular multiplication	6	5	6

(lower) approximation, for d up to some fractional power of n. For small n and in characteristic 2, λ approaches 2, and the data for FFTConvREDC in Table 1 are false. Note also that we write no value for general modular multiplications, since the σ-representation reduces in some respect the generality.

The results of this section are summed up in the following.

Theorem 2. *Let $n \geq 2$ be an integer and $p(X) \in \mathbb{Z}/(n\cdot\mathbb{Z})[X]$, $\mathbf{A} = (\mathbb{Z}/(n \cdot \mathbb{Z})[X]) / (p(X))$ with $\deg(p) = d < D < 2d$ and $(p(X), X^{2D}-1) = 1$, with D suited for FFT. Suppose that $v(X) \in \mathbf{A}$ such that*

$$v(X) \cdot p(X) \equiv -1 \mod X^D - 1$$

is precomputed and belongs to the description of $\mathbf{A}$. Furthermore, if multiplication by a polynomial $b(X) \in \mathbf{R}[X]$ occurs frequently, one also precomputes the spectrum of $b(X)$ and of $b(X) \cdot v(X)$ rem $p(X)$. Then multiplication in $\mathbf{A}$ can be performed in Montgomery representation using FFT transforms. This task is fulfilled by one of the procedures FFTConvREDC or SFFTConvREDC. The required number of FFT(D) transforms is indicated by Table 1.

Remark 3. We have presented our algorithms in σ-representation, for the sake of consistency with the generic definition of ConvREDC. It should be mentioned however, that in the case of FFT which requires renormalizations, this representation has no particular advantage. One may as well use the *natural* representation, i.e. polynomials are given by their coefficients. The times are essentially the same, one even saves one FFT operation in FFTConvREDC with preconditioned modular multiplication. In this case, one can also give times for the general multiplication, and they improve upon NTL too. The changes being quite straightforward, we do not include them here.

Remark 4. For small d, the CRT operations modulo the small primes p_j cannot be neglected. This will increase the factor λ. Thus FFTConvREDC saves 1/2 FFT only for large d. Naturally, the impact of CRT is more important for small d, also in [NTL]. We shall see that the Toom-Cook is in many respects preferable for such values of d.

5. Long arithmetic with FFT and ConvREDC

Let n be a *very large* integer and $B = 2^w$ a block. This can be the size of a machine word, in which case FFT is implemented directly on top of machine operations, or several machine words, assuming that multiplication of integers up to the size of a block is done using some fast method with less overhead.

The question of whether this choice of $R = B^k$ in REDC is still the best when using linear convolutions for the long multiplication was addressed by Crandall and Fagin in [CF] and, recently, by McLaughlin. In [Ml], he uses some interesting ad-hoc small Winograd convolutions in several detailed *frameworks* which improve long integer arithmetic. He also proposes an improvement of polynomial arithmetic using FFT, but does not analyse it or use it for long integer arithmetic.

Here we consider the way SFFTConvREDC adapts to long integer arithmetic. Assume that FFT is carried out in one of the ways described in the previous section, probably using the *three primes* FFT [GG], or similar. In the SFFT variant, it is sufficient to let $N > N_0$ with

$$N_0 = 2 \cdot \left\lceil \frac{\log(n)}{\log(B)} \right\rceil \cdot B^2 = K \cdot B^2. \tag{26}$$

The factor 2 allows us to take the sum of two products of polynomials in spectral representation: this is required in step 5 of SFFTConvREDC. We assume that $K < B$, so $n^2 < B^B$. We assume the matrices $\mathbf{M}_D, \mathbf{M}_+$ are computed with respect to $\mathbb{Z}/(N \cdot \mathbb{Z})$, a ring *affording* $2D$-point FFT.

5.1. **Representation of integers, normalization and carry.** In order to apply SFFTConvREDC, we choose $R^{\perp}(X) = X^D - 1, R^{\top}(X) = X^D + 1$ and let $R = R^{\perp}(B)$. We impose the condition $R > 4 \cdot n$.

If $a = \sum_{i=0}^{h} a_i B^i,\ 0 \leq a_i < B$, is an integer, then it will be convenient to use a notation, say $\phi(a)(X) = \sum_{i=0}^{h} a_i X^i \in \mathbb{Z}[X]$ for the associated polynomial. In the case $0 \leq a_i < B$ we say that $\phi(a)(X)$ is a polynomial in *normal form.* If $f(X) \in \mathbb{Z}[X]$, then the inverse operation is an evaluation at B, which we denote by $\widehat{B} : f(X) \in \mathbb{Z}[X] \mapsto f(B) \in \mathbb{Z}$. We also denote by *normalization* the operation

$$\Phi(f) = \phi\left(f(B)\right) = \phi \circ \widehat{B}. \tag{27}$$

Naturally, here $f(B)$ is rewritten as a polynomial in B with block-coefficients $0 \leq f_i < B$.

We let $0 < V < R$ be such that $V \cdot n \equiv -1 \mod R$, set $p(X) = \phi(n),\ v(X) = \phi(V)$ and assume that these polynomials are precomputed together with their spectral representations $\sigma(v(X)), \sigma(p(X))$. Note that we do not necessarily have $p(X) \cdot v(X) \equiv -1 \mod R^{\perp}(X)$, as polynomials, but $p(B) \cdot v(B) \equiv -1 \mod R^{\perp}(B)$. An integer $a \in \mathbb{Z}$ will have the Montgomery representation $\rho(a) = a \cdot R = a \cdot (B^D - 1) \text{ rem } n$.

The σ'-representation will be the composition of three opertaions $\sigma' = \sigma \circ \phi \circ \rho$. First, $\rho(a) \in \mathbb{Z}/(n \cdot \mathbb{Z})$ is the Montgomery representation of a and ϕ transforms it into a polynomial of degree $\deg(\phi(\rho(a)(X))) < D$. Finally, this polynomial is given by its (Fourier) spectral representation σ.

We saw in the previous section that normalization plays an important role when using FFT for arithmetic in $\left(\mathbb{Z}/(n \cdot \mathbb{Z})[X]\right)/(p(X))$. Now we investigate how normalization works in the context of long integer arithmetic. If $z(X)$ is a polynomial with coefficients which exceed B, then normalization can modify its degree; yet, it leaves the value at B unchanged. This happens in particular if $z(X)$ is computed as the product of two polynomial representations of integers. So there is a problem of *carry propagation* that we have to control. The problem is addressed in the following

Lemma 3. *With the notations above, let $0 < a, b < R$ be two integers and $a \cdot b = Z$, $\phi(a)(X) \cdot \phi(b)(X) = z(X)$. Then $z(B) = Z$ and there is a linear polynomial $G(X)$*

in normal form, such that

$$\begin{aligned} z^{\perp}(B) &= (Z \text{ rem } R^{\perp}(B)) + G(B) \cdot R^{\perp}(B) \quad \textit{and} \\ z^{\top}(B) &= (Z \text{ rem } R^{\top}(B)) + G(B) \cdot R^{\top}(B). \end{aligned} \tag{28}$$

Proof. By construction,

$$z(B) = (\phi(a))(B) \times (\phi(b))(B) = a \cdot b = Z,$$

which is the first statement. For the rest, it will be convenient to use some related Chinese Remainder base for polynomials and numbers; notice that $R^{\top}(X) = R^{\perp}(X) + 2$. This relation yields, up to a factor 2, the required base. With the usual meaning for $z^{\perp}(X), z^{\top}(X)$, etc., we have

$$\begin{aligned} 2 \cdot z(X) &= z^{\perp}(X) \cdot R^{\top}(X) - z^{\top}(X) \cdot R^{\perp}(X), \\ 2 \cdot Z &= Z^{\perp} \cdot R^{\top} - Z^{\top} \cdot R^{\perp}, \end{aligned} \tag{29}$$

the second line being the result of the evaluation $\widehat{B}$ on the equality in the first line. Since $z(B) = Z$, we have by definition of $z^{\perp}(X), z^{\top}(X)$ also $z^{\perp}(B) \equiv Z^{\perp} \mod R^{\perp}(B)$, $z^{\top}(B) \equiv Z^{\top} \mod R^{\top}(B)$, although in general $Z^{\perp} \neq z^{\perp}(B)$. Thus we let

$$\begin{aligned} z^{\perp}(B) &= Z^{\perp} + g(B) \cdot R^{\perp}(B), \\ z^{\top}(B) &= Z^{\top} + h(B) \cdot R^{\top}(B), \end{aligned}$$

and insert these relations in (29), in which we set $X = B$ and use the fact that $z(B) = Z$:

$$\begin{aligned} 2 \cdot z(B) &= \left(Z^{\perp} + g(B) \cdot R^{\perp}(B)\right) \cdot R^{\top}(B) - \left(Z^{\top} + h(B) \cdot R^{\top}(B)\right) \cdot R^{\perp}(B) \\ &= \left(Z^{\perp} \cdot R^{\top}(B) - Z^{\top} \cdot R^{\perp}(B)\right) + R^{\perp}(B) \cdot R^{\top}(B) \cdot (g(B) - h(B)) \\ &= \left(Z^{\perp} \cdot R^{\top}(B) - Z^{\top} \cdot R^{\perp}(B)\right) = 2 \cdot Z. \end{aligned}$$

By comparing the last two lines, it follows that $g(B) = h(B)$. The relations (28) follow with $G(B) = g(B) = h(B)$. The coefficients of $z^{\perp}(X), z^{\top}(X)$ are all $< T$ and the degree of the polynomials is $< D$. Consequently $|z^{\perp}(B)| < T \cdot B^{D-1} < K \cdot B^{D+1} < (1/2)B^{D+2}$, by (26) and the assumption on K. Since $G(B) = z^{\perp}(B) \div R$, in normal form, $G(X)$ must be at most linear. □

5.2. **An algorithm for long integer arithmetic.** The shifts $\mathfrak{S}^{\pm}$ are connected in this case to some additional operations, so they will be split in several steps. Here is the long integer arithmetic application of SFFTConvREDC:

```
**Algorithm IntConvREDC** *(* REDC for integer arithmetic with FFT *)*
Input: $\sigma'(a)(X), \sigma'(b)(X); \sigma(p(X)), \sigma(v(X))$.
Output: $\sigma'(a \cdot b)$.
  1. $\sigma_s(z) = \sigma'_s(a) \cdot \sigma'_s(b)$ for all $\mu_s \in \mathcal{R}$.
  2. Renormalize $z^{\perp}$:
     2.1 Retrieve $z^{\perp}(X)$ from its spectrum $\sigma^{\perp}(z)$ applying $\mathbf{M}_D^{-1}$.
     2.2 Let $z_1(X) = \phi(z^{\perp}(B) \text{ rem } R) = \phi(Z^{\perp})$ and compute
         $\sigma^{\perp}(z_1)$ by FFT ($\mathbf{M}_D$).
  3. Compute $\sigma^{\perp}(m)$ as follows: $m(\mu_s) = z_1(\mu_s) \cdot v(\mu_s)$ for $\mu_s \in \mathcal{R}^{\perp}$.
```

```
4. Compute $\mathfrak{S}^+(\sigma^\perp(m))$:
    4.1 Retrieve $m^\perp(X)$ from its spectrum, by FFT with $\mathbf{M}_D^{-1}$.
    4.2 Let $M = m^\perp(B)$ rem $R$ and set $m_1(X) = \phi(M)$.
    4.3 Let $\sigma^\top(m) = \sigma^\top(m_1)$ $(\mathbf{M}_+)$.
5. $t(\mu_s) = (z(\mu_s) + m(\mu_s) \cdot p(\mu_s))$ for all $\mu_s \in \mathcal{R}^\top$ and compute
   $t^\top(X)$ (using $\mathbf{M}_+^{-1}$).
6. Set $T = \left(-R^\top(B) + \left(t^\top(B) \text{ rem } R^\top(B)\right)\right)/(-2)$.
7. If $T > n$, then $T = T - n$.
8. Compute $\sigma'(c) = \sigma(\phi(T))$ (using $\mathbf{M}_D|\mathbf{M}_+$) and return $\sigma'(c)$.
```

The correctness of this procedure is asserted by the following

Theorem 3. *Let $n \in \mathbb{N}$ be a large integer, $R = B^D - 1 > n$ with $B = 2^w$ a power of two and suppose that $V \cdot R \equiv -1 \mod n$. Then procedure IntConvREDC correctly computes the product of two numbers $a, b \in \mathbb{Z}/(n \cdot \mathbb{Z})$, in their spectral Montgomery representation with respect to R, using $\sim 7 \cdot FFT(D)$ operations. The reductions for squaring, resp. repeated multiplication by a fixed factor, are of 0, resp. 2 FFT(D) operations.*

Proof. We verify the integrity of the algorithm by comparing with Montgomery's REDC. Step 2.2 implies that $\sigma^\perp(z)$ is the spectrum of a polynomial with $z_1(B) = Z^\perp$. REDC requires $M = (Z^\perp \cdot V)$, so step 3 correctly computes $\sigma^\perp(\phi(Z^\perp \cdot V))$, and step 4.2 provides the correct value of M.

Retrieving T is delicate. Note that $z(B) = Z$, $m(B) = M$ and $v(B) = V$; thus $R^\perp(B) \cdot T(B) = R \cdot T$. We can apply Lemma 3 by additivity to the two products $a(X) \cdot b(X) = z(X)$ and $m(X) \cdot v(X)$ and deduce that

$$T^\top(B) = \left(R \cdot T \text{ rem } R^\top(B)\right) + G(B) \cdot R^\top(B) = -2 \cdot T + G(B) \cdot R^\top(B).$$

By the initial condition imposed on R we have $2T < 4n < R < R^\top(B)$. Consequently

$$T^\top(B) \text{ rem } R^\top(B) = R^\top(B) - 2 \cdot T,$$

and step 6 correctly computes the value of T. Step 7 is the correction in step 3 of REDC and step 8 returns the value in σ'-representation. □

6. ConvREDC with Toom-Cook

The Toom-Cook algorithm uses (iterated) Lagrange interpolation, with multiple and not necessarily fixed number of interpolation points per iteration; as an extreme case, Karatsuba-Ofman uses a constant of three interpolation points per iteration. We consider here the other extreme degenerated case with only one iteration. In this case, the interpolation points $\mathcal{R} = \{\mu_s = s - 1, s = 1, 2, \ldots, 2d + 2\}$ are used, and we set $\mathcal{R}^\perp = \{0, 1, \ldots, d\}$. The choices $\mu_s = s - d$ or $\mu_s = 2^{s-1}$ are also interesting, the differences being not substantial. The procedure ConvREDC can now be followed step by step:

```
Algorithm TCConvREDC    (* ConvREDC with Tom–Cook *)

Input: $\sigma'(a), \sigma'(b), a, b \in \mathbf{A}; \sigma(v), \sigma(p)$.
Output: $\sigma'(c) = \sigma'(a \cdot b)$

1. $\sigma_s(z) = \sigma'_s(a) \cdot \sigma'_s(b)$, for $s \in \mathcal{R}$.
```

2. $m(\mu_s) = z(\mu_s) \cdot v(\mu_s)$ for $\mu_s \in \mathcal{R}^{\perp}$. This yields $\sigma^{\perp}(m)$.
3. $\sigma^{\top}(m) = \mathfrak{S}^{+}\left(\sigma^{\perp}(m)\right)$.
4. $t(\mu_s) = (z(\mu_s) + m(\mu_s) \cdot p(\mu_s)) / R^{\perp}(\mu_s)$ for all $\mu_s \in \mathcal{R}^{\top}$. This yields $\sigma^{\top}(t)$.
5. $\sigma^{\perp}(t) = \mathfrak{S}^{-}\left(\sigma^{\top}(t)\right)$. Return $\sigma'(c) = \sigma(t)$.

6.1. **Run time discussion for Toom-Cook.** The complexity of the steps 1, 2 and 4 of this algorithm depends only on the underlying multiplication in $\mathbf{r}$ and adds up to $T_1 = 5d\mathsf{M}(\mathbf{r})$. We shall focus here on the evaluation of $\mathfrak{S}^{\pm}$. Asymptotically optimal methods for evaluating these operators in $c \cdot M(\mathbf{A})$ have been known since [BM]. The constant c has been improved by Bostan, Gaudry and Schost [BGS] and Bostan and Schost [BS], starting from the Tellegen principle. One may find in Bostan's thesis [Bo] a detailed account both of historic aspects of the problem and of the best available algorithms (in part developed by that author) for various cases. It is made explicite in [Bo], [BS], that the evaluation can be made in fact by using *any* (including school-book) convolution. This allows a gradual balance between overhead and asymptotic behavior, which is practically very interesting.

We shall give below an ad hoc presentation of the algorithm for transforming the evaluation of the shift operators $\mathfrak{S}^{\pm}$ in convolution problems, which may be attacked with any method, including Karatsuba or Toom-Cook itself, but also FFT; the idea is the same as in [BS], derived in more of a numerical analyst's way. The approach has a large overhead but is asymptotically, for d not very small with respect to n, quite fast.

On the other extreme, without overhead, but slower for increasing values of $d/\log(n)$, one has a classical result, described in [Kn] and based on Newton polynomials. The operators $\mathfrak{S}^{\pm}$ can be evaluated using finite differences, the operations are then very simple, but their number is quadratic in d. The operation count is

$$T_2 = \theta(\mathfrak{S}^{+}) + \theta(\mathfrak{S})^{-} \sim d^2 \cdot (\mathsf{A}(\mathbf{r}) + 2\mathsf{S}(\mathbf{r})) . \tag{30}$$

This approach is preferable for its simplicity, as long as the quadratic number of additions is not dominant. The total run time per operation is in general bounded by (14). Using finite differences, this becomes:

$$T = 5d\mathsf{M}(\mathbf{r}) + d^2 \cdot (\mathsf{A}(\mathbf{r}) + 2\mathsf{S}(\mathbf{r})) . \tag{31}$$

The estimates (14), (31) should be considered with the following caution: the natural input and output of a chain of operations is *not* the σ-representation, and thus the cost of the input and output conversion should be averaged over the chain of operations that one intends to implement. Typical candidates are exponentiation, addition in abelian varieties, etc.

Note that if d is so small with respect to n that one may assume that

$$d \cdot \mathsf{S}(\mathbf{r}) < k\mathsf{M}(\mathbf{r}) \tag{32}$$

for some fixed constant, then (31) becomes

$$T < (5 + 3k)d \cdot \mathsf{M}(\mathbf{r}).$$

The time is in these cases linear in d, a performance which cannot be achieved by FFT, e.g. due to the factor $d \cdot \log(d)$ in the run time of a d-point FFT transform. We have thus found a very defensive estimate range of magnitudes (32) in which Toom-Cook is the preferable method for extension ring arithmetic. Note that in the range of small of values of d defined by (32), even the $\mathbb{Z}/(N \cdot \mathbb{Z})$-multiplications in the FFT method are comparable to $\mathbb{Z}/(n \cdot \mathbb{Z})$-multiplications, since the CRT steps dominate for small d.

When $n \to \infty$, then $\mathsf{S}(\mathbf{r}) \sim \log(n)$ and $\mathsf{M}(\mathbf{r}) \sim \log(n) \log\log(n)$, so asymptotically the estimate (32) merely yields $d = O(\log\log(n))$. It is useful to have even for this rather narrow range an algorithm which is close to the complexity-theoretic lower bound. However, this estimate reflects reality only poorly, and (32) is preferable for practical use. For sizes of n allowing subquadratic—but no linear—multiplication, a bound reflecting this situation is

$$d \cdot \mathsf{S}(\mathbf{r}) < d^{\varepsilon} \mathsf{M}(\mathbf{r}), \tag{33}$$

where the constant ε may also be estimated empirically. The bounds (32), (33) are useful when it comes to deciding which algorithm to adopt for a certain range in the (d, n)-plane.

7. Fast evaluation of the Toom-Cook spectral transforms

We present here, for reasons of completeness, the background for fast convolution evaluation of the Toom-Cook shift operators. As already mentioned, the ideas and essential results are already in [Bo], and in part, in [BS]. Let $\mathcal{R}^{\perp} = \{0, 1, \dots, d\}$ and $\mathcal{R}^{\top} = \{d+1, d+2, \dots, 2d+1\}$, making $C(X)$ into a polynomial of degree $2(d+1)$; so $R^{\top} = R^{\perp} + (d+1)$, with the obvious meaning for the addition. Let $e_s^{\perp}(X), s \in \mathcal{R}^{\perp}$ and $e_s^{\top}(X), s \in \mathcal{R}^{\top}$ be Chinese Remainder bases for $R^{\perp}$ and $R^{\top}$, respectively. Thus

$$\begin{array}{rclll} e_s^{\perp}(X) & = & \delta_{s,u} & \bmod e_u^{\perp}(X) & \text{for} \quad s, u \in \mathcal{R}^{\perp}, \\ e_s^{\top}(X) & = & \delta_{s,u} & \bmod e_u^{\top}(X) & \text{for} \quad s, u \in \mathcal{R}^{\top}. \end{array} \tag{34}$$

The polynomials above can be given explicitly [GG], [Kn]:

$$e_s^{\perp}(X) = \frac{R^{\perp}(X)}{(X - s) \cdot (R^{\perp})'(s)}, \quad s \in \mathcal{R}^{\perp}. \tag{35}$$

A similar formula applies for $e_s^{\top}(X)$, but it is more convenient to relate these polynomials to $e_s^{\perp}(X)$. One verifies from the definitions, that

$$e_{s+d+1}^{\top}(X) = e_s^{\perp}(X - (d+1)),$$

for $s + d + 1 \in \mathcal{R}^{\top}$. If $g(X) \in \mathbf{R}[X]$ has $\deg(g) \leq d$ and is given by the interpolation values $g(s)$, $s \in \mathcal{R}^{\perp}$, the shift operator $\mathfrak{S}^{+}$ produces the values at $d + 1 + \mathcal{R}^{\perp}$. Since the number of interpolation points in both sets exceeds the degree of the polynomial by at least 1, both sets of interpolation values entirely determine $g(X)$. We write this fact using the respective CRT bases (34), (35):

$$g(X) = \sum_{s=0}^{d} g(s) \cdot e_s(X) = \sum_{s=0}^{d+1} g(s + d + 1) \cdot e_s(X - (d+1)). \tag{36}$$

Now insert $X = u + d + 1$ in the above equation and apply the CRT base properties (34). The right hand sum becomes $\sum_{s=0}^{d+1} g(s + d + 1) \cdot e_s(u) = g(u + d + 1)$, while the left hand side is $\sum_{s=0}^{d} g(s) \cdot e_s(u + d + 1)$. We have obtained the following useful expression for the interpolation values:

$$g(u+d+1) = \sum_{s=0}^{d} g(s) \cdot e_s(u+d+1) \quad \text{for} \quad u \in R^{\perp}.$$

The cofactors of $g(s)$ above are $e_s(u+d+1) = \frac{R(s+d+1)}{R'(s)\cdot(u+d+1-s)}$. The only expression here which depends on u is $1/(u+d+1-s) = (1/X)\big|_{X=d+1+(u-s)}$; it does in fact only depend on $u-s$. Note also that the denominator never vanishes for $s \in \mathcal{R}^{\perp}$. Let $\gamma(s) = \frac{R(s+d+1)}{R'(s)}$; these values can be precomputed and $\Gamma_0(s) = g(s)\cdot\gamma(s),\ s \in \mathcal{R}^{\perp}$, do not depend on u. They can be computed in $d+1$ multiplications in $\mathbf{R}$. Finally let $R(X) = \sum_{i=0}^{2d+1} r_i X^i = \sum_{i=0}^{2d+1} \frac{X^i}{i}$. The expression for the shifted values becomes:

$$g(u+d+1) = \sum_{s=0}^{d} r_{d+1+u-s} \cdot \Gamma_0(s). \tag{37}$$

The equation (37) looks *almost like a convolution.* Let us define the sequence:

$$\Gamma(s) \quad = \quad \begin{cases} \Gamma_0(s) & \text{if} \quad 0 \le s \le d, \\ 0 & \text{if} \quad d < s < 2(d+1). \end{cases} \tag{38}$$

One can now regard (37) either as the middle part of the coefficients of the polynomial product

$$\begin{aligned} P(X) = R(X)\cdot G(X) &= \left(\sum_{i=0}^{2d+1} r_i \cdot X^i\right) \cdot \left(\sum_{i=0}^{d} \Gamma(i)\cdot X^i\right) \\ &= \left(\sum_{i=0}^{2d+1} X^i/i\right) \cdot \left(\sum_{i=0}^{d} \Gamma(i)\cdot X^i\right), \end{aligned}$$

i.e. the coefficients of $X^{d+1}, X^{d+2}, \ldots, X^{2d+1}$ in $P(X)$, or as the upper half of the $2(d+1)$)-*cyclic convolution* of the sequences $(\Gamma_s), (r_s)$, i.e. the modular product

$$P_{cyc}(X) = R(X)\cdot G(X) \mod X^{2(d+1)} - 1.$$

Considering (37) as a convolution has the appealing consequence that it allows for a *transition zone* between various areas of the (d, n) plane [BS]. The evaluation for $\mathfrak{S}^-$ is absolutely analog, with the needed sign changes in the right places. We leave this to the reader. Note that in FFT-mode, the cyclic convolution of $G(X)$ by the fixed polynomial $R(X)$-with precomputed spectral values-requires 4 FFT operations:

Theorem 4. *The operators $\mathfrak{S}^{\pm}$ for the Toom-Cook method can be evaluated in essentially $4FFT(d+1)$ operations.*

With the fast evaluation of the shift operators, the overview of computation times is given by Table 2.

TABLE 2. Number of FFT operations at a second glance

Operation (Number of $FFT(D)$)	**NTL - Shoup**	**SFFTConv REDC**	**FFTConv REDC**	**IntFFT ConvREDC**	**TC + FFT + ConvREDC**
Squaring	10	7	6.5	7	8
Precond. mult.	6	5	6	5	8

Remark 5. Following [Bo], a useful interpretation of the shift operators consists in regarding them as evaluation of a d-recurrent sequence with fixed initial values. This is of particular interest in small characteristic, when the reciprocal values r_i are not defined.

Acknowledgments

I am very grateful to François Morain, who insisted that I develop upon the ideas in [Mi] and provided the conditions for the development of this paper at the École Polytechnique, and to him, Alin Bostan and Pierrick Gaudry for lively and stimulating discussions during the work on the paper. I thank the referee and editor for their attentive revision and useful remarks.

References

[Bl] R. Blahut: *Fast Algorithm for Digital Signal Processing*, Addison-Wesley (1985). MR0777867 (86h:94005)

[Bo] A. Bostan: *Algorithmique efficace pour des opérations de base en Calcul formel*, Thèse de doctorat à l'École polytechnique de Paris (2003).

[BGS] A. Bostan, P. Gaudry and É. Schost: *Linear Recurrences with Polynomial Coefficients and Computation of the Cartier-Manin Operator on Hyperelliptic Curves*, in Proceedings Fq7, LNCS **2948**, pp. 40–58. MR2092621 (2006f:11070)

[BLS] A. Bostan, G. Lecerf and É. Schost: *Tellegen's Principle into Practice*, Proceedings of ISSAC'03 (2003), ACM Press, pp. 37–44.

[BM] A. Borodin and R.T. Moenck: *Fast modular transforms*, Computer System Sci. **8**, Nr. 3 (1974), pp. 366–386. MR0371144 (51:7365)

[BS] A. Bostan and É. Schost: *Polynomial evaluation and interpolation on special sets of points*, Journal of Complexity, vol. **21**, no. 4, pp. 420–446, Aug. 2005. MR2152715 (2006g:12016)

[CF] R. Crandall and B. Fagin: *Discrete weighted transforms and large-integer arithmetic*, Math. Comp. **62** (1994), pp. 305-324. MR1185244 (94c:11123)

[Co] S. A. Cook: *On the Minimum Computation Time of Function*, Ph.D. Thesis, Harvard (1966).

[CT] J. Cooley and W. Tukey: *An algorithm for the machine computation of complex Fourier series*, Mathematics of Computation **19** (1965), pp. 297–301. MR0178586 (31:2843)

[gmp] GNU Multiple Precision Library **gmp**, Version 4.12 http://www.swox.com/gmp/

[GG] J. von zur Gathen and J. Gerhard: *Modern Computer Algebra*, Cambridge University Press (2000). MR1689167 (2000j:68205)

[GKP] R. Graham, D. Knuth and O. Patashnik: *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley 1990 MR1397498 (97d:68003)

[HZ] G. Hanrot and P. Zimmerman: *A Long Note on Mulders' Short Product*, Journal of Symbolic Computation, **37** (2004), pp. 391–401. MR2092652 (2005g:65011)

[Kn] D. Knuth: *The Art of Computer Programming, Vol II: Seminumerical Algorithms*, Addison and Wesley (1998), 3rd Edition. MR633878 (83i:68003)

[Ku] H. T. Kung: *On computing reciprocals of power series*, Numerische Mathematik **92** (1974), pp. 341–348. MR0351045 (50:3536)

[Li] LiDIA: *A C++ Library For Computational Number Theory,* http://www.informatik.tu-darmstadt.de/TI/LiDIA

[Mi] P. Mihăilescu: *Cyclotomy of Rings & Primality Testing*, dissertation 12278, ETH Zürich, 1997.

[Ml] P. McLaughlin, Jr.: *New Frameworks for Montgomery's Modular Multiplication Method*, Math. Comp **73**, Nr. 246 (2003) pp. 899–906. MR2031414 (2004j:11149)

[Mo] P. Montgomery: *Modular Multiplication without Trial Division*, Math. Comp. **44** (1985), 519–521. MR777282 (86e:11121)

[Mu] T. Mulders: *On Short Multiplications and Divisions*, Proceedings AAECC **11** (2000), pp. 69–88. MR1817699 (2001m:68187)

[NTL] V. Shoup: ***NTL**: A Library for doing Number Theory*, Version 5.3.1, http://www.shoup.net/ntl/

[ScSt] A. Schönhage and V. Strassen: *Schnelle Multiplikation großer Zahlen*, Computing **7**, (1971) pp. 281–292. MR0292344 (45:1431)

[Sh] V. Shoup: *A New Polynomial Factorization Algorithm and its Implementation*, Journal of Symbolic Computation, **20** (1996) pp. 363–397. MR1384454 (97d:12011)

[Si] M. Sieveking: *An Algorithm for Division of Power Series*, Computing **10** (1972), pp. 153–156. MR0312701 (47:1257)

[St] V. Strassen: *Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten*, Numerische Mathematik **20** (1972/73), pp. 238–251. MR0324947 (48:3296)

[To] A. L. Toom: *The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers*, Soviet Mathematics Doklady **4** (1963), pp. 714–716.

[vH] J. van der Hoeven: *The Truncated Fourier Trnsform and Applications*, Proceedings ISSAC 2004 MR2126956

[Wn] S. Winograd: *On Computing the Discrete Fourier Transform*, Math. Comp. **32** (1978), pp. 175–199. MR0468306 (57:8142)

Mathematisches Institut der Universität Göttingen, Bunsenstrasse 3–5, D-37073 Göttingen, Germany

E-mail address: `preda@upb.de`