Fast Multiplication of Polynomials over Arbitrary Rings*

David G. Cantor† *Erich Kaltofen*‡

† Math. Sci. Dept., UCLA, Los Angeles, CA 90024, dgc@cs.ucla.edu
‡ Dept. Comp. Sci., RPI, Troy, NY 12180-3590, kaltofen@csv.rpi.edu

Abstract

An algorithm is presented that allows to multiply two univariate polynomials of degree no more than n with coefficients from an arbitrary (possibly non-commutative) ring in $O(n \log(n) \log(\log n))$ additions and subtractions and $O(n \log(n))$ multiplications. The arithmetic depth of the algorithm is $O(\log(n))$. This algorithm is a modification of the Schönhage-Strassen procedure to arbitrary radix fast Fourier transforms, and division by the radix is circumvented. By a Kronecker homomorphism the method can be extended to multivariate polynomials.

* This material is based upon work supported by the National Science Foundation under Grant Nos. DCR-85-04391 and CCR-87-0563 and by an IBM Faculty Development Award (second author).

1. Introduction

The subject of this article is a generalization of the Schönhage-Strassen integer multiplication algorithm to multiplying polynomials with coefficients from an arbitrary, possibly non-commutative, ring. As our main result we construct an algorithm to multiply polynomials of degree not higher than n with coefficients from a ring in $O(n\log(n)\log(\log n))$ ring additions, subtractions, and multiplications. The non-scalar multiplicative complexity of our method is $O(n\log(n))$, and its parallel complexity as the depth of the corresponding arithmetic circuit is $O(\log(n))$.

Schönhage [14] first investigated the polynomial multiplication problem for arbitrary fields of characteristic 2, in which the standard 2^k-point Fourier transform inverts to the zero vector. If we have a division by 2, the Schönhage-Strassen integer multiplication algorithm can be easily recast as a polynomial multiplication procedure (cf [13]).. Observe that it is assumed that the fields do not automatically contain the primitive roots necessary to perform a fast Fourier transform and therefore such primitive roots must be synthetically adjoined. It is that what makes the total cost increase by a factor of $\log(\log n)$. Schönhage's characteristic 2 algorithm works with 3^k-point Fourier transforms on polynomial rings modulo cyclotomic polynomials of the same order. With a division by 3 he again gets an $O(n\log(n)\log(\log n))$ algorithm. For subtle reasons that approach seems not to generalize for even an order of 5^k. Here we offer an alternate method that will work for order s^k for any $s \geq 3$. The idea in our method is that we use two different polynomial rings which support an s^k-point Fourier transform and then obtain the $2n-1$ entries in the convolutions by Chinese remaindering.

We also eliminate the divisions by s by computing scalar multiples of the convolutions with a power of s for two relatively prime s's and then find a suitable integer linear combination that cancels those multipliers. Finally, we have removed the assumption of commutativity for the coefficient rings. This is more by accident, since the Schönhage-Strassen approach is already correct for skew-fields such as the quaternions. The reason is that only the primitive roots used in the Fourier transforms need to commute with the ring elements, which happens to be true for those synthetically generated roots. Our observation may prove useful, however, since we now have asymptotically fast algorithms for multiplying 'string polynomials' [10], §4.6.1, Exercise 17-18, or for multiplying matrix polynomials.

If one allows the total operation count to be asymptotically worse, a $O(n\log(n))$ non-scalar multiplicative complexity can be achieved differently [8], or over special rings that complexity is much smaller, for finite fields refer to [11]. The best known lower bounds for any complexity are no better than linear in the degree [3], [9]. A practically useful algorithm of worse asymptotic performance over finite fields is given in [4].

Our model of computation is that of a straight-line program for the coefficients of the product from the coefficients of the inputs [1] §1.5.I, although the asymptotic complexities remain true an algebraic random access machines as defined in [7]. Non-scalar multiplications are those

where both factors are dependent on the coefficients of the inputs.

2. Rings with a Fourier Transform

The main feature of our algorithm is that it works over an arbitrary ring R. We write $0 \in R$ for its additive zero element and $1 \in R$ for its multiplicative unit, $0 \neq 1$. Any ring is $\mathbf{Z}$ module by $(-1)a := -a$ and $na := \sum_{j=1}^{n} a$, $a \in R$, $n \in \mathbf{Z}$, $n \geq 0$. The integers $\mathbf{Z}$ are embedded into R by $\Pi(n) := n \cdot 1$, and we write n shortly for the element $n \cdot 1$. Notice that a 'rng' R without a unit can be embedded into the ring $\mathbf{Z}\times R$ with unit (1,0) and addition and multiplication defined as

$$(m, a) + (n, b) := (m + n, a + b),\ (m, a)(n, b) := (mn, mb + na + ab)$$

(cf [6],. §2.17), so all our results hold for such rngs as well. The subring $\Pi(\mathbf{Z})$ is either isomorphic to $\mathbf{Z}$ or to $\mathbf{Z}_m$, the integers modulo m for some m. We write R^n for the left R-module of n-dimensional vectors over R, and we write a_i, $0 \leq i \leq n-1$, for the i-th component of $\mathbf{a} \in R^n$.

In the following we introduce the theory for the fast discrete Fourier transform [5], here for the abstract module R^n.

Definition: Let R be a ring, $n \in \mathbf{Z}$, $n \geq 2$, $\omega \in R$. Then ω is a *principal n-th root of unity* in R if

(PR1) $\omega^n = 1$; note that then $\omega^{-1} = \omega^{n-1}$.

(PR2) For all $i \in \mathbf{Z}$: $i \not\equiv 0 \bmod n$ ☞ $\sum_{j=0}^{n-1} \omega^{ij} = 0$.

(PR3) For all $w \in R$: $\omega w = w\omega$; this condition is needed when computing convolutions in non-commutative rings.

Example: Let F be a field, and let ω be a primitive n-th root of unity, that is

$$n = \min\{m \mid \omega^m = 1,\ m \in \mathbf{Z},\ m \geq 1\}.$$

Then ω is also a principal n-th root of unity. Clearly, (PR1) and (PR3) are satisfied. Since

$$0 = \omega^{in} - 1 = (\omega^i - 1) \sum_{j=0}^{n-1} \omega^{ij}$$

and $\omega^i = \omega^{i \bmod n} \neq 1$, the second factor of the RHS of the above equation must be 0. □

Definition: Let R be a ring, $n \geq 1$, ω a principal n-th root of unity in R, $\mathbf{a}, \hat{\mathbf{a}} \in R^n$. Then $\hat{\mathbf{a}}$ is the *discrete Fourier transformed* of $\mathbf{a}$ with respect to ω, $\hat{\mathbf{a}} = \mathrm{DFT}[[\omega]](\mathbf{a})$, respectively $\mathbf{a}$ is the *discrete Fourier inverse* of $\hat{\mathbf{a}}$ with respect to ω, $\mathbf{a} = \mathrm{DFI}[[\omega]](\hat{\mathbf{a}})$, if

$$\text{for all } i \text{ with } 0 \leq i \leq n-1\text{: } \hat{a}_i = \sum_{j=0}^{n-1} \omega^{ij} a_j.$$

For the purpose of fast computation, one chooses $n = s^k$. Following is the now-famous algorithm to compute the forward transform [5].

Algorithm *Fast DFT*

Input: $n = s^k$, $\omega \in R$ such that $\omega^n = 1$, $\mathbf{a} \in R^n$.

Output: $\hat{\mathbf{a}} = \mathrm{DFT}[[\omega]](\mathbf{a}) \in R^n$.

If $n = 1$ **Then Return** $\hat{\mathbf{a}} \leftarrow \mathbf{a} = [a_0]$.

Step S (Split-up): Since for $0 \le i < n/s$, $r \in \mathbf{Z}_s$ we have

$$\hat{a}_{si+r} = \sum_{j=0}^{n-1} \omega^{(si+r)j} a_j = \sum_{j=0}^{n/s-1} \sum_{l=0}^{s-1} \omega^{(si+r)(j+ln/s)} a_{j+ln/s} = \sum_{j=0}^{n/s-1} (\omega^s)^{ij} \left(\sum_{l=0}^{s-1} \omega^{rln/s+rj} a_{j+ln/s} \right)$$

we have

$$[\hat{a}_{si+r}]_{i=0,\ldots,n/s-1} = \mathrm{DFT}[[\omega^s]] \left(\left[\omega^{rj} \sum_{l=0}^{s-1} \omega^{rln/s} a_{j+ln/s} \right]_{j=0,\ldots,n/s-1} \right).$$

$\rho \leftarrow \omega^{n/s}$; $\rho^{(0)} \leftarrow 1$; $\omega^{(0)} \leftarrow 1$.

For $r \leftarrow 0, \ldots, s-1$ **Do**
  $\rho^{(r)} \leftarrow \rho^{(r-1)} \rho$; $\omega^{(r)} \leftarrow \omega^{(r-1)} \omega$.
  For $j \leftarrow 0, \ldots, n/s-1$ **Do**
    $\omega^{(rj)} \leftarrow \omega^{(rj-r)} \omega^{(r)}$.
    $u_j^{(r)} \leftarrow \omega^{(rj)} \sum_{l=0}^{s-1} (\rho^{(r)})^l a_{j+ln/s}$.

Step R (Recursion):
For $r \leftarrow 0, \ldots, s-1$ **Do**
  Call the algorithm recursively with $n' \leftarrow n/s$, $\omega' \leftarrow \omega^s$, and $\mathbf{u}^{(r)} \in R^{n/s}$, obtaining $\hat{\mathbf{u}}^{(r)} = \mathrm{DFT}[[\omega']](\mathbf{u}^{(r)})$.

Step I (Insertion):
For $r \leftarrow 0, \ldots, s-1$ **Do**
  For $i \leftarrow 0, \ldots, n/s$ **Do**
    $\hat{a}_{si+r} \leftarrow \hat{u}_i^{(r)}$.
Return $\hat{\mathbf{a}}$. □

Lemma 1: *The Fast DFT algorithm requires* $O(snk)$ *additions and multiplications with powers of* ω *in R. Its arithmetic circuit depth is* $O(\log(n))$. *If* $s = c^l$, *where c is a constant, the complexity can be reduced to* $O(lnk)$.

Proof: Step S costs $O(sn)$ arithmetic operations, step R $sT(n/s)$, where $T(m)$ is the operation count on m-dimensional input. Thus $T(n)$ satisfies the recursion

$$T(n) \le \gamma sn + s\,T(n/s), \quad \gamma \text{ a constant},$$

which by induction on k leads to $T(n) \le \gamma sk n$.
For $s = c^l$ we observe that for $j = 0, \ldots, n/s$

$$[u_j^{(r)}]_{r=0,\ldots,s-1} = \mathrm{DFT}[[\rho]]([a_{j+ln/s}]_{l=0,\ldots,s-1}) \times [\omega^{(rj)}]_{r=0,\ldots,s-1}^{\text{transposed}},$$

which thus can be computed by the original algorithm in $O(ls)$ operations. Hence the cost of step S reduces to $O(ln)$
Using a processor efficient parallel prefix computation one can get all ω^i, $0 \le i \le n-1$, in $O(\log(n))$ depth using no more than $O(n \log(n))$ operations. The parallel depth of the remaining computation of step S is $O(\log(s))$. Thus the total depth of $O(k \log(s))$, which is $O(\log(n))$. □

As is well known, condition (PR2) allows to compute Fourier inverses by forward transforms. We have the following lemma.

Lemma 2: *Let R be a ring and ω a principal n-th root of unity,* $\mathbf{a}, \hat{\mathbf{a}} \in R^n$, $\hat{\mathbf{a}} = \mathrm{DFT}[[\omega]](\mathbf{a})$. *Then ω^{n-1} is also a principal n-th root of unity and* $\mathrm{DFT}[[\omega^{n-1}]](\hat{\mathbf{a}}) = n\mathbf{a}$.

Proof: We first show the principality of ω^{n-1}. Conditions (PR1) and (PR3) are clear, for (PR2) we write

$$\sum_{j=0}^{n-1} (\omega^{(n-1)j})^i = \sum_{j=0}^{n-1} (\omega^{(n-1)j \bmod n})^i = \sum_{j=0}^{n-1} (\omega^j)^i = 0.$$

Let $\hat{\hat{\mathbf{a}}} = \mathrm{DFT}[[\omega^{n-1}]](\hat{\mathbf{a}})$. By definition

$$\hat{\hat{a}}_i = \sum_{l=0}^{n-1} \omega^{(n-1)il} \hat{a}_l = \sum_{l=0}^{n-1} \omega^{(n-1)il} \sum_{j=0}^{n-1} \omega^{lj} a_j = \sum_{j=0}^{n-1} a_j \sum_{l=0}^{n-1} (\omega^l)^{(n-1)i+j} = n a_i,$$

since by (PR2) we must have

$$\sum_{l=0}^{n-1} (\omega^{(n-1)i+j})^l = 0 \quad \text{for} \quad j \not\equiv i \bmod n. \quad \square$$

Observe that in general we cannot divide by n in R, which will be taken care off by the trick in theorem 2 below. Next we define convolutions of vectors.

Definition: Let $\mathbf{a}, \mathbf{b} \in R^n$. Then $\mathbf{c} \in R^n$ is the (*wrapped*) *convolution* of $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{c} = \mathbf{a} \circledcirc \mathbf{b}$, if

$$\text{for all } j \text{ with } 0 \le j \le n-1\text{: } c_j = \sum_{\substack{0 \le l,m < n \\ l+m \equiv j \bmod n}} a_l b_m.$$

Convolutions of vectors can be compute by two forward and one inverse Fourier transforms.

Lemma 3: *Let* $\mathbf{a}, \mathbf{b} \in R^n$, *$\omega$ a principal n-th root of unity,* $\mathbf{c} = \mathbf{a} \circledcirc \mathbf{b}$. *Let* $\hat{\mathbf{a}} = \mathrm{DFT}[[\omega]](\mathbf{a})$, $\hat{\mathbf{b}} = \mathrm{DFT}[[\omega]](\mathbf{b})$. *Then*

$$\mathrm{DFT}[[\omega^{n-1}]]([\hat{a}_i \hat{b}_i]_{i=0,\ldots,n-1}) = n\mathbf{c}.$$

Proof: For all $0 \le i \le n-1$ we have

$$\hat{a}_i \hat{b}_i = \left(\sum_{j=0}^{n-1} \omega^{ij} a_j \right) \left(\sum_{j=0}^{n-1} \omega^{ij} b_j \right) = \sum_{j=0}^{2n-2} \omega^{ij} \sum_{\substack{0 \le l,m < n \\ l+m=j}} a_l b_m = \sum_{j=0}^{n-1} c_j \omega^{ij} = \hat{c}_i .$$

It is here where we need (PR3). The lemma now follows from lemma 2. □

Our following lemma is the key to relating an $n = s^k$-th order root to an s-th order one and allows us to later synthesize such roots.

Lemma 4: *Let ρ be an s-th order principal root of unity in the ring R. Furthermore, for $n = s^k$, $k \ge 1$, let $\omega \in R$ be such that $\omega^{n/s} = \rho$ and $\omega w = w\omega$ for all $w \in R$. Then ω is an n-th order principal root of unity in R.*

Proof: The key condition is (PR2) for ω. We proceed by induction on k. Suppose the statement is true for ω^s. Assume now that $i \not\equiv 0 \bmod n$. If $i \not\equiv 0 \bmod n/s$ then

$$\sum_{j=0}^{n-1} \omega^{ij} = \sum_{j=0}^{n/s-1} \sum_{l=0}^{s-1} \omega^{i(sj+l)} = \sum_{l=0}^{s-1} \omega^{il} \left(\sum_{j=0}^{n/s-1} (\omega^s)^{ij} \right) = 0,$$

whereas if $i \equiv 0 \bmod n/s$ then $i = sm$ and $m \not\equiv 0 \bmod n/s$. Hence

$$\sum_{j=0}^{n-1} \omega^{ij} = \sum_{j=0}^{n-1} (\omega^s)^{mj} = \sum_{j=0}^{n-1} (\omega^s)^{(j \bmod n/s)m} = s \sum_{j=0}^{n/s-1} (\omega^s)^{jm} = 0. \quad \square$$

3. Polynomial Rings

Now let R be any ring. We will embed R into a ring $\overline{\bar{R}}$ with a high order principal root of unity as follows. First, we construct $\bar{R} := R[y]/(\Psi_s(y))$, where $\Psi_s(y)$ is the s-th order cyclotomic polynomial whose integral coefficients are projected into R by Π. The element $\bar{y} := y \bmod \Psi_s(y)$ in $\bar{R}$ is a principal s-th root of unity. Then we construct $\overline{\bar{R}} := \bar{R}[x]/(x^{s^k} - \bar{y})$. By lemma 4, x is a principal s^{k+1}-st root of unity in $\overline{\bar{R}}$.

Several lemmas follow now.

Lemma 5: *Let R be a ring, $\bar{R} := R[y]/(\Psi_s(y))$. Then $y^t \bmod \Psi_s(y) \in \bar{R}$, $t \in \mathbf{Z}_s^*$, the multiplicative units in $\mathbf{Z}_s$, is a principal s-th root of unity in $\bar{R}$.*

Proof: (PR1) is clearly satisfied since $\Psi_s(y)$ divides $y^{st} - 1$. Similarly,

$$(y^t)^{is} - 1 = (y^{ti} - 1) \sum_{j=0}^{s-1} (y^{ti})^j \equiv 0 \bmod \Psi_s(y),$$

so for $i \not\equiv 0 \bmod s$, $\Psi_s(y)$ divides $\sum_{j=0}^{s-1} (y^t)^{ij}$ as integer polynomials, which applying Π gives (PR2). (PR3) follows from the fact that for any integer n, $n \cdot 1$ commutes with any ring element in R, so any element in $\Pi(\mathbf{Z})[y]$ with any element in $R[y]$. □

Lemma 6: *Let R be a ring, $\bar{R} = R[y]/(\Psi_s(y))$, $\bar{y} := y \bmod \Psi_s(y)$, $s \geq 2$ a fixed integer, $n = s^k$, $k \geq 1$, $\overline{\bar{R}} := \bar{R}[x]/(x^n - \bar{y}^t)$, $t \in \mathbf{Z}_s^*$, $0 \leq \kappa \leq k$, $l := n/s^{\kappa-1}$, $\omega_l := x^{s^\kappa} \in \overline{\bar{R}}$. Then we can compute the* DFT[[ω_l]]($\mathbf{a}$) *and* DFT[[ω_l^{l-1}]]($\mathbf{a}$), $\mathbf{a} \in \overline{\bar{R}}^l$, *in $O(n\, l \log(l))$ arithmetic operations in R. In particular, we can compute $\omega_l^j a$, $a \in \overline{\bar{R}}$, $j \in \mathbf{Z}$, in $O(n)$ operations in R.*

Proof: We compute DFT[[ω]]($\mathbf{a}$) by the Fast DFT algorithm. By lemma 1 that costs, since s is fixed, $O(l \log(l))$ additions (or subtractions) and multiplications by ω_l^j, $1 \leq j < l$, in $\overline{\bar{R}}$. Each addition in $\overline{\bar{R}}$ costs $O(n)$ additions in $\bar{R}$, and, since s is fixed, also $O(n)$ additions in R. Multiplication by ω_l^j, $j \in \mathbf{Z}$, is done by observing that in $\overline{\bar{R}}$, $x^n = \bar{y}^t \in \bar{R}$. Let j' and j'' be such that $s^\kappa j \bmod sn = j'n + j''$, $0 \leq j' < s$, $0 \leq j'' < n$. Then for $a = \sum_{\nu=0}^{n-1} \alpha_\nu x^\nu \in \overline{\bar{R}}$, $\alpha_\nu \in \bar{R}$, we have

$$\omega_l^j a = (x^{s^\kappa})^j \left(\sum_{\nu=0}^{n-1} \alpha_\nu x^\nu \right) = (\bar{y}^t)^{j'} \left(\sum_{\nu=0}^{n-1} \alpha_\nu x^{\nu + j''} \right) = (\bar{y}^t)^{j'} \left(\sum_{\mu=0}^{j''-1} \bar{y}^t \alpha_{n-j''+\mu} x^\mu + \sum_{\mu=j''}^{n-1} \alpha_{\mu-j''} x^\mu \right).$$

Thus, the multiplication problem of a is essentially multiplying its coefficients by powers of $\bar{y}$ and rearranging indices. Each such coefficient multiplication has constant cost, totaling $O(n)$ arithmetic steps in R. □

Lemma 7: *Let $s \in \mathbf{Z}$, $s \geq 3$, $t \in \mathbf{Z}_s^*$, $t \geq 2$. Then there exist $h_1(y)$, $h_2(y) \in \mathbf{Z}[y]$ and $e \in \mathbf{Z}$, $e \geq 1$, such that over $\mathbf{Z}[y]$,*

$$h_1(y)(y^t - y) + h_2(y)\Psi_s(y) = s^e.$$

Moreover, if $t = s - 1$, a solution is possible with $e = 1$.

Proof: The main part of the proof centers on the resultants

$$\delta_{s,t} := \mathrm{res}_y(y^{t-1} - 1, \Psi_s(y)) \in \mathbf{Z}.$$

First note that since

$$\mathrm{res}(f_1 f_2, g) = \mathrm{res}(f_1, g)\mathrm{res}(f_2, g), \text{ and } y^{t-1} - 1 = \prod_{d \mid t-1} \Psi_d(y)$$

(cf. van der Waerden [16], §28 and §36) we have

$$\delta_{s,t} = \prod_{d \mid t-1} \mathrm{res}_y(\Psi_d(y), \Psi_s(y)).$$

Now by (1.3) and theorem 4 in [2] we have for $s > d \geq 1$,

$$\mathrm{res}(\Psi_d, \Psi_s) = \begin{cases} p^{\phi(d)} & \text{if } \dfrac{s}{d} \text{ is a power of a prime } p, \\ 1 & \text{else.} \end{cases} \qquad (\dagger)$$

Therefore there exists an integer $e_d \geq 0$ such that $\mathrm{res}(\Psi_d, \Psi_s) \mid s^{e_d}$. Hence $\delta_{s,t} \mid s^e$, $e := \sum_{d \mid t-1} e_d$. Furthermore, $\mathrm{res}_y(y, \Psi_s(y)) = (-1)^{\phi(s)}$ and therefore

$$\mathrm{res}_y(y^t - y, \Psi_s(y)) = \pm \delta_{s,t} \mid s^e.$$

Now there exist polynomials $g_1(y), g_2(y) \in \mathbf{Z}[y]$ with

$$g_1(y)(y^t - y) + g_2(y)\Psi_s(y) = \mathrm{res}_y(y^t - y, \Psi_s(y)).$$

Hence the polynomials h_1, h_2 can be chosen integral multiples if g_1 and g_2.

If $t = s - 1$ then the only possible d's with $d \mid s - 2$ and s/d being a power of a prime are $d = 1$ and $d = 2$. From (†) we then get $\delta_{s,s-1} = s$ for $s = p$ or $2p$, p a prime. In all other cases $\delta_{s,s-1}$ is a proper divisor of s. □

Lemma 8: *Let R be a ring, $s \in \mathbf{Z}$, $s \geq 3$. Furthermore, let $t \in \mathbf{Z}_s^*$, $t \geq 2$, and let $n \geq 1$,*

$$\overline{\overline{R}}_0 = R[x, y]/((x^n - y)(x^n - y^t), \Psi_s(y)),$$

$$\overline{\overline{R}}_1 = R[x, y]/(x^n - y, \Psi_s(y)), \quad \overline{\overline{R}}_2 = R[x, y]/(x^n - y^t, \Psi_s(y)).$$

Moreover, let e and $h_1(y)$ be as in lemma 7, with its coefficients projected into R. Consider the projection

$$\begin{aligned} P\colon \overline{\overline{R}}_0 &\to \overline{\overline{R}}_1 \times \overline{\overline{R}}_2 \\ a_0 &\to [a_0 \bmod (x^n - y, \Psi_s(y)),\ a_0 \bmod (x^n - y^t, \Psi_s(y))] \end{aligned}$$

and the Chinese remainder map

$$\begin{aligned} C\colon \overline{\overline{R}}_1 \times \overline{\overline{R}}_2 &\rightarrow \overline{\overline{R}}_0 \\ [a_1, a_2] &\rightarrow s^e a_1 + (x^n - y)h_1(y)(a_2 - a_1). \end{aligned}$$

Then $C(P(a_0)) = s^e a_0$ *for all* $a_0 \in \overline{\overline{R}}_0$.

Proof: We provide the proof since the rings are somewhat unusual. First we show that P is injective. Assume $P(a_0) = P(b_0)$ for $a_0, b_0 \in \overline{\overline{R}}_0$. Now $a_0 \equiv b_0 \bmod (x^n - y, \Psi_s(y))$ which means that

$$a_0(x, y) - b_0(x, y) = (x^n - y)c_0(x, y) \quad \text{for some } c_0(x, y) \in R[x, y].$$

Also $0 \equiv a_0 - b_0 \equiv (x^n - y)c_0 \bmod x^n - y^t$, which means that $x^n - y^t$ must divide $c_0(x, y)$. Thus $a_0 \equiv b_0 \bmod (x^n - y)(x^n - y^t)$, which means that $a_0 = b_0$ in $\overline{\overline{R}}_0$. Next we establish that $P(C([a_1, a_2])) = [s^e a_1, s^e a_2]$. This is clear for the first component, the second component follows from lemma 7 as

$$\begin{aligned} s^e a_1 + (x^n - y)h_1(y)(a_1 - a_2) \bmod x^n - y^t &= s^e a_1 + (y^t - y)h_1(y)(a_2 - a_1) \\ &= s^e a_1 + s^e(a_2 - a_1) = s^e a_2. \end{aligned}$$

Finally, we have $P(C(P(a_0))) = P(s^e a_0)$, which by the injectivity of P yields $C(P(a_0)) = s^e a_0$. □

4. Polynomial Multiplication

Let us briefly describe the idea of the algorithm. We have a ring R and we want to multiply polynomials in $R[x]$. For a moment let us assume that s is invertible in R. In fact, will multiply elements in

$$\bar{R}[x]/(x^n - \bar{y}^{t_1}),\ \ n = s^k,\ \ t_1 \in \mathbf{Z}_s^*,\ \ \bar{R} := R[y]/(\Psi_s(y)),\ \bar{y} := y \bmod \Psi_s(y).$$

For simplicity, let us assume that k is even. Following Schönhage and Strassen [15] we write two elements in $\overline{\bar{R}}$ as

$$f(x) = \sum_{i=0}^{\sqrt{n}-1} a_i(x)(x^{\sqrt{n}})^i,\ \ a_i \in \bar{R}[x],\ \deg(a_i) < \sqrt{n},$$

$$g(x) = \sum_{i=0}^{\sqrt{n}-1} b_i(x)(x^{\sqrt{n}})^i,\ \ b_i \in \bar{R}[x],\ \deg(b_i) < \sqrt{n}.$$

Now

$$f(x)\, g(x) \bmod x^n - \bar{y}^{t_1} = \sum_{i=0}^{\sqrt{n}-1} c_i(x)(x^{\sqrt{n}})^i$$

with

$$c_i(x) = \sum_{j+l=i} a_j b_l + \bar{y}^{t_1} \sum_{j+l=\sqrt{n}+i} a_j b_l,\ \ 0 \le i < \sqrt{n}.$$

We will compute the c_i's exactly by computing certain convolutions of the $\sqrt{n}$-dimensional vectors $[a_i]_{0\le i<\sqrt{n}}$ over the rings $R_\tau := \bar{R}[x]/(x^{\sqrt{n}} - \bar{y}^{t_\tau})$, $\tau = 1, 2$, $t_2 \in \mathbf{Z}_s^*$, $t_1 \ne t_2$. Once we have

$$c_i(x) \bmod x^{\sqrt{n}} - \bar{y}^{t_\tau},\ \ \tau = 1, 2,$$

we obtain the true c_i's by Chinese remaindering. It is here where we have departed from previous methods [15], [14], [13], which obtain the c_i's correctly by choosing a modulus of degree at least $2\sqrt{n}$. We must account for the particular wrap-around in the c_i's. Here we follow the approach in [1], theorem 7.2. First we observe that in $\overline{\bar{R}}_1$, $x^{\sqrt{n}} = \bar{y}^{t_1}$, so

$$[x^i a_i(x)]_{0\le i<\sqrt{n}} \circledcirc [x^i b_i(x)]_{0\le i<\sqrt{n}} = [x^i c_i(x)]_{0\le i<\sqrt{n}}.$$

Therefore $c_i \bmod x^{\sqrt{n}} - \bar{y}^{t_1}$ can be found by premultiplication by x^i and postmultiplication by x^{-i}. In $\overline{\bar{R}}_2$ the proper multipliers are $x^{t_3 i}$ with $t_3 \equiv t_2^{-1} t_1 \bmod s$. Clearly, $(x^{t_3})^{\sqrt{n}} = \bar{y}^{t_1} \bmod x^{\sqrt{n}} - \bar{y}^{t_2}$. The algorithm follows.

Algorithm *Polynomial Multiplication*

Input: $n = s^k$, $s \ge 3$, $k \ge 1$, $t \in \{1, s-1\}$, $f, g \in R[x, y]$, $\deg_x(f) < n$, $\deg_x(g) < n$, $\deg_y(f) < \phi(s)$, $\deg_y(g) < \phi(s)$.

Output: $(s^{\eta_s(n)} fg) \bmod (x^n - y^t, \Psi_s(y))$, where $\eta_s(n)$ is a non-negative integer depending on n.

Step I (Initialize): Determine the block number $l = s^{\lfloor k/2 \rfloor}$, and the block size $m = s^{\lceil k/2 \rceil}$. Notice that $lm = n$. Split f and g into

$$f(x) = \sum_{i=0}^{l-1} a_i(x)(x^m)^i, \quad g(x) = \sum_{i=0}^{l-1} b_i(x)(x^m)^i,$$

$a_i, b_i \in R[x, y]$, $\deg_x(a_i) < m$, $\deg_x(b_i) < m$. We will compute

$$c_i(x) = s^{\eta_s(n)} \left(\sum_{j_1+j_2=i} a_{j_1}(x) b_{j_2}(x) + y^t \sum_{j_1+j_2=i+l} a_{j_1}(x) b_{j_2}(x) \right), \quad 0 \le i < l.$$

Let $t_1 = t$, $t_2 = s - t$,

$$\overline{\overline{R}}_1 = R[x, y]/(x^m - y^{t_1}, \Psi_s(y)), \quad \overline{\overline{R}}_2 = R[x, y]/(x^m - y^{t_2}, \Psi_s(y)).$$

Now

$$\omega \leftarrow \begin{cases} x & \text{if } l = m, \\ x^s & \text{if } sl = m, \end{cases}$$

is a principal sl-th root of unity in both $\overline{\overline{R}}_1$ and $\overline{\overline{R}}_2$. We compute

$$c_i^{(\tau)}(x) = c_i(x) \bmod (x^m - y^{t_\tau}, \Psi_s(y)), \quad 0 \le i < l, \ \tau \in \{0, 1\},$$

as follows.

Step P (Premultiplication): Compute

$$\mathbf{a}^{(1)} \leftarrow [a_i \omega^i]_{0 \le i < l}, \quad \mathbf{b}^{(1)} \leftarrow [b_i \omega^i]_{0 \le i < l},$$

over $\overline{\overline{R}}_1$ and

$$\mathbf{a}^{(2)} \leftarrow [a_i (\omega^{s-1})^i]_{0 \le i < l}, \quad \mathbf{b}^{(2)} \leftarrow [b_i (\omega^{s-1})^i]_{0 \le i < l},$$

over $\overline{\overline{R}}_2$ by special multiplication with powers of x. Notice that always $(s - 1)t_2 \equiv t_1 \bmod s$.

Step F (Forward Transform): For $\tau = 1$ and $\tau = 2$ compute

$$\hat{\mathbf{a}}^{(\tau)} \leftarrow \mathrm{DFT}[[\omega^s]](\mathbf{a}^{(\tau)}), \quad \hat{\mathbf{b}}^{(\tau)} \leftarrow \mathrm{DFT}[[\omega^s]](\mathbf{b}^{(\tau)})$$

over the rings $\overline{\overline{R}}_\tau$. Notice that ω^s is a principal l-th root of unity.

Step M (Componentwise Multiplication): For $\tau = 1$ and $\tau = 2$ compute

$$\hat{\mathbf{c}}^{(\tau)} \leftarrow [s^{\eta_s(m)} \hat{a}_i^{(\tau)} \hat{b}_i^{(\tau)}]_{0 \le i < l}$$

over $\overline{\overline{R}}_\tau$ by recursive application of the procedure.

Step $\mathbf{F^{-1}}$ (Inverse Transformation)
For $\tau = 1$ and $\tau = 2$ compute

$$\tilde{\mathbf{c}}^{(\tau)} \leftarrow \mathrm{DFT}[[(\omega^s)^{l-1}]](\hat{\mathbf{c}}^{(\tau)}) = l\mathrm{DFI}[[\omega^s]](\hat{\mathbf{c}}^{(\tau)}).$$

Step $\mathbf{P^{-1}}$ (Componentwise postmultiplication):
Compute

$$\mathbf{c}^{(1)} \leftarrow [\omega^{-i}\tilde{c}_i^{(1)}]_{0\le i<l},\ \ \mathbf{c}^{(2)} \leftarrow [\omega^{-i(s-1)}\tilde{c}_i^{(2)}]_{0\le i<l},$$

again by special multiplication with powers of x.

Step C (Chinese remaindering): Let $h_1(y)$, $h_2(y) \in \mathbf{Z}[y]$ be such that

$$h_1(y)(y^{s-1} - y) + h_2(y)\Psi_s(y) = s.$$

By lemma 7 such polynomials exist. Let

$$\lambda := \begin{cases} 1 \text{ if } t = 1 \\ 2 \text{ if } t = s-1 \end{cases}, \quad \mu := 3 - \lambda,$$

This makes $\overline{\overline{R}}_\lambda = R[x, y]/(x^m - y, \Psi_s(y))$ and $\overline{\overline{R}}_\mu = R[x, y]/(x^m - y^{s-1}, \Psi_s(y))$.
For $i \leftarrow 0, \ldots, l-1$ **Do**

$$c_i(x) \leftarrow sc_i^{(\lambda)} + (x^m - y)h_1(y)(c_i^{(\mu)} - c_i^{(\lambda)}) \bmod \Psi_s(y)$$

(cf. Lemma 8). Notice that we now have $\eta_s(n) = \eta_s(m) + \lfloor k/2 \rfloor + 1$, thus $\eta_s(n) = O(\log(n))$.

Step R (Final return): Compute

$$h(x) \leftarrow \sum_{i=0}^{l-1} c_i(x)x^{mi} \bmod (x^n - y^t, \Psi_s(y))$$

by polynomial additions and proper wrap-around at $x^n = y^t$. **Return** $h(x)$. □

Theorem 1: *Algorithm Polynomial Multiplication requires $O(n\log(n)\log(\log n))$ arithmetic steps in the ring R. Its multiplicative complexity is $O(n\log(n))$. Its parallel complexity is $O(\log(n))$.*

Proof: Let $T(n)$, $M(n)$, and $D(n)$ be the total, multiplicative, and parallel complexity, respectively. From lemma 6 it easily follows that

$$T(n) \le \gamma_1 ml\log(l) + 2l\,T(m),\ \ \gamma_1 \text{ a constant},$$

whose solution is $T(n) = O(n\log(n)\log(\log n))$ [1], theorem 7.8. The only multiplications occur in step M, if we carry out the scalar multiplications in step C by additions. Thus

$$M(n) = 2l\,M(m),\ \ M(s) \text{ constant},$$

whose solution is $M(n) \le \gamma_2 s^k(k-1)$, γ_2 a constant. Finally, by lemma 1 the depth satisfies

$$D(n) \le \gamma_3 \log(l) + D(m), \ \ \gamma_3 \text{ a constant,}$$

which clearly satisfies $D(n) \le \gamma_3 \log(n)$. Here we refer also to Wüthrich's master's thesis [17] for a detailed analysis of the depth of the Schönhage-Strassen method. □

We now come to the main theorem, where we remove the factor $s^{\eta_s(n)}$ without introducing divisions. We also generalize the algorithm to multivariate polynomials.

Theorem 2: *Let*

$$f(x_1,\ldots,x_v) = \sum_{\substack{0\le i_j<n_j \\ j=1,\ldots,v}} a_{i_1,\ldots,i_v} x_1^{i_1} \cdot \ \cdot \ \cdot x_v^{i_v}, \ \ g(x_1,\ldots,x_v) = \sum_{\substack{0\le i_j<n_j \\ j=1,\ldots,v}} b_{i_1,\ldots,i_v} x_1^{i_1} \cdot \ \cdot \ \cdot x_v^{i_v} \in R[x_1,\ldots,x_v],$$

R an arbitrary ring, and let $N = \prod_{j=1}^{v}(2n_j - 1)$. *Then* $c_{k_1,\ldots,k_v} \in R$, $0 \le k_j < 2n_j - 1$, *with*

$$\sum_{\substack{0\le k_j<2n_j-1 \\ j=1,\ldots,v}} c_{k_1,\ldots,k_v} x_1^{k_1} \cdot \ \cdot \ \cdot x_v^{k_v} = f(x_1,\ldots,x_v)g(x_1,\ldots,x_v)$$

can be computed simultaneously in $O(N\log(N)\log(\log N))$ *additions and subtractions,* $O(N\log(N))$ *multiplications, and in* $O(\log(N))$ *parallel depth.*

Proof: The multivariate case is reduced to the univariate one by performing the well-known Kronecker substitution

$$x_j \leftarrow y^{(2n_1-1)\cdots(2n_{j-1}-1)}$$

(cf [12]).. For the univariate case we apply the Polynomial multiplication algorithm twice for $n_1 = 3^{k_1} \ge N$, $n_2 = 4^{k_2} \ge N$. We get

$$3^{\eta_3(n_1)} c_{k_1,\ldots,k_v}, \ \ 4^{\eta_4(n_2)} c_{k_1,\ldots,k_v}$$

within the stated complexity. Now there exist integers K and L with

$$3^{\eta_3(n_1)} K + 4^{\eta_4(n_2)} L = 1,$$

so computing the corresponding linear combination of the obtained multiples of the convolutions we get in $O(N)$ additional additions and multiplications the true coefficients of the product. □

The above theorem would already have followed using the 2-fold split-up method by Schönhage and Strassen [15] adopted to polynomial multiplication (cf [13]). and the 3-fold split-up method by Schönhage. However, if we have a division by s in R our algorithm is superior. For $R \supset \mathbf{Z}_6$, for example our method can be carried out with $s = 5$. Moreover, our method is based on one generic algorithm.

There is a nice trick due to L. I. Bluestein to reduce the computation of an n-point transform, n arbitrary, to an s^k-dimensional convolution problem [10], §4.3.3, Exercise 8. We thus have the following corollary.

Corollary: DFT[[ω]]($\mathbf{a}$), $\omega \in R$ *invertible,* $\mathbf{a} \in R^n$, *R an arbitrary ring, n an arbitrary positive integer, can be computed in* $O(n \log(n) \log(\log n))$ *arithmetic operations in R.* □

5. Concluding Remark

Our investigation to generalize the Schönhage method to s-fold split-ups was undertaken in an attempt to speed the Schönhage-Strassen method itself. Our hope was that in choosing the s depending on n, say of order $\log(n)$, the decrease in the depth of the recursion would lead to an asymptotic improvement overall. Unfortunately, this appears not to help.

The main open question is, of course, to improve the algorithm by a $\log\log n$ factor. But other questions are interesting as well. Theorem 2 is formulated for polynomials that achieve maximum individual degrees. One could consider inputs of maximum total degree

$$f(x_1,\ldots,x_v) = \sum_{\substack{0\le i_j<n\\ j=1,\ldots,v\\ i_1+\cdots+i_v\le n-1}} a_{i_1,\ldots,i_v} x_1^{i_1}\cdot\ \cdot\ \cdot x_v^{i_v},\quad g(x_1,\ldots,x_v) = \sum_{\substack{0\le i_j<n\\ j=1,\ldots,v\\ i_1+\cdots+i_v\le n-1}} b_{i_1,\ldots,i_v} x_1^{i_1}\cdot\ \cdot\ \cdot x_v^{i_v}$$

and ask for the complexity on terms of the number of monomials in the product, $N := \binom{2n-2+v}{v}$. It is not clear how to compute that product in $O(N\log(N)\log(\log N))$ arithmetic steps.*

Acknowledgement: Our renewed interest in this subject was stimulated by Professor Arnold Schönhage during the complexity theory meeting in Oberwolfach in November 1986.

References

1. Aho, A., Hopcroft, J., and Ullman, J., *The Design and Analysis of Algorithms,* Addison and Wesley, Reading, MA (1974).

2. Apostol, T. M., "Resultants of cyclotomic polynomials," *Proc. Amer. Math. Soc.* **24**, pp. 457-462 (1970).

3. Brown, M. R. and Dobkin, D. P., "An improved lower bound on polynomial multiplication," *IEEE Trans. Comp.* **C-29**, pp. 337-340 (1980).

4. Cantor, D. G., "On a substitute for the fast Fourier transform for finite fields," Manuscript (October 1986).

5. Cooley, J. W. and Tuckey, J. W., "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.* **19**, pp. 297-301 (1965).

6. Jacobson, N., *Basic Algebra I,* W. H. Freeman Co., San Francisco (1974).

7. Kaltofen, E., "Greatest common divisors of polynomials given by straight-line programs," *J. ACM* **35**(1), pp. 231-264 (1988).

8. Kaminski, M., "Multiplication of polynomials over the ring of integers," *Proc. 25th IEEE Symp. Foundation Comp. Sci.*, pp. 251-254 (1984).

---

* Note added on April 26, 1989: This problem has been solved by the second author and Lakshman Yagati using sparse polynomial transforms [Proc. ISSAC '89, ACM Press, to appear].

9. Kaminski, M. and Bshouty, N. H., “Multiplicative complexity of polynomial multiplication over finite fields,” *Proc. 28th Annual Symp. Foundations Comp. Sci.*, pp. 138-140 (1987).

10. Knuth, D. E., *The Art of Programming, vol. 2, Semi-Numerical Algorithms, ed. 2,* Addison Wesley, Reading, MA (1981).

11. Lempel, A., Seroussi, G., and Winograd, S., “On the complexity of multiplication in finite fields,” *Theoret. Comp. Sci.* **22**, pp. 285-296 (1983).

12. Moenck, R. T., “Another polynomial homomorphism,” *Acta Inf.* **6**, pp. 153-169 (1976).

13. Nussbaumer, H. J., “Fast polynomial transform algorithms for digital convolutions,” *IEEE Trans. ASSP* **28**, pp. 205-215 (1980).

14. Schönhage, A., “Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2,” *Acta Inf.* **7**, pp. 395-398 (1977). (In German).

15. Schönhage, A. and Strassen, V., “Schnelle Multiplikation grosser Zahlen,” *Computing* **7**, pp. 281-292 (1971). (In German).

16. Waerden, B. L. van der, *Modern Algebra,* F. Ungar Publ. Co., New York (1953).

17. Wüthrich, R., “Schnelle Multiplikation grosser Zahlen,” Diplomarbeit, Seminar f. angew. Math., Univ. Zürich. (In German.).