New algorithms for relaxed multiplication

Joris van der Hoeven

Dépt. de Mathématiques (Bât. 425)
Université Paris-Sud
91405 Orsay Cedex
France
Email: joris@texmacs.org

August 11, 2003

In previous work, we have introduced the technique of relaxed power series computations. With this technique, it is possible to solve implicit equations almost as quickly as doing the operations which occur in the implicit equation. Here "almost as quickly" means that we need to pay a logarithmic overhead. In this paper, we will show how to reduce this logarithmic factor in the case when the constant ring has sufficiently many 2^p-th roots of unity.

Keywords: power series, multiplication, algorithm, FFT, computer algebra
A.M.S. subject classification: 68W25, 42-04, 68W30, 30B10, 33F05, 11Y55

1. Introduction

Let $\mathsf{C} \ni \{\frac{1}{2}\}$ be an effective ring and consider two power series $f = f_0 + f_1\, z + \cdots$ and $g = g_0 + g_1\, z + \cdots$ in $\mathsf{C}[[z]]$. In this paper we will be concerned with the efficient computation of the first n coefficients of the product $h = f\, g = h_0 + h_1\, z + \cdots$.

If the first n coefficients of f and g are known beforehand, then we may use any fast multiplication for polynomials in order to achieve this goal, such as divide and conquer multiplication [KO63, Knu97], which has a time complexity $K(n) = O(n^{\log 3/\log 2})$, or F.F.T. multiplication [CT65, SS71, CK91, vdH02], which has a time complexity $M(n) = O(n \log n \log\log n)$.

For certain computations, and most importantly the resolution of implicit equations, it is interesting to use so called "relaxed algorithms" which output the first i coefficients of h as soon as the first i coefficients of f and g are known for each $i \leqslant n$. This allows for instance the computation of the exponential $g = \exp f$ of a series f with $f_0 = 0$ using the formula

$$g = \int f'\, g. \tag{1}$$

In [vdH97, vdH02], we proved the following theorem:

Theorem 1. *There exists a relaxed multiplication algorithm of time complexity $O(M(n) \log n)$ and space complexity $O(n)$.*

In this paper, we will improve the time complexity bound in this theorem in the case when C admits 2^p-th roots of unity for any p. In section 2, we first reduce this problem to the case of "semi-relaxed multiplication", when one of the arguments is fixed and the other one relaxed. More precisely, let f and g be power series, such that g is known up to order n. Then a semi-relaxed multiplication algorithm computes the product $h = fg$ up to order n and outputs $(fg)_i$ as soon as $f_0, ..., f_i$ are known, for all $i < n$. In section 3, we show that the $\log n$ overhead in theorem 1 can be reduced to $O((\log n)^{\log 3/\log 2})$. In section 4, the technique of section 3 is further improved so as to yield an $O(\mathrm{e}^{2\sqrt{\log\log n}})$ overhead.

In the sequel, we will use the following notations from [vdH02]: we denote by $\mathsf{C}[[z]]_n \subseteq \mathsf{C}[z] \subseteq \mathsf{C}[[z]]$ the set of truncated power series of order n, like $f = f_0 + \cdots + f_{n-1} z^{n-1}$. Given $f \in \mathsf{C}[[z]]_n$ and $0 \leqslant i < j \leqslant n$, we will denote $f_{i...j} = f_i + \cdots + f_{j-1} z^{j-i-1} \in \mathsf{C}[[z]]_{j-i}$.

2. FULL AND SEMI-RELAXED MULTIPLICATION

In [vdH97, vdH02], we have stated a fast algorithm for relaxed multiplication. This algorithm is based on a subdivision of the $n \times n$ square which represents a full relaxed multiplication of truncated power series in $\mathsf{C}[z]_n$. We reproduced this subdivision in the left hand image of figure 1; each square corresponds to the computation of a zealous product.

Looking more carefully at this picture, we observe that the computation of a full $2\,n \times 2\,n$ product is roughly equivalent to a full relaxed $n \times n$ multiplication, two semi-relaxed $n \times n$ multiplications and one zealous $n \times n$ multiplication. At the right hand side, we also showed how a semi-relaxed $2\,n \times 2\,n$ multiplication reduces to two semi-relaxed $n \times n$ multiplications and two zealous $n \times n$ multiplications.

In fact, this observation corresponds to the fact that the fast relaxed multiplication algorithm actually applies Newton's method in a hidden way. Indeed, since Brent and Kung [BK78], it is well known that Newton's method can also be used in the context of formal power series in order to solve differential or functional equations. One step of Newton's method at order n involves the recursive application of the method at order $\lceil n/2 \rceil$ and the resolution of a linear equation at order $\lfloor n/2 \rfloor$. The resolution of the linear equation corresponds to the computation of the two semi-relaxed products.

From a complexity point of view, it follows that any semi-relaxed multiplication algorithm of time complexity $Q(n)$ also yields a full relaxed multiplication algorithm of time complexity $O(Q(n))$, under the assumption that $\frac{Q(n)}{n}$ increases towards infinity.

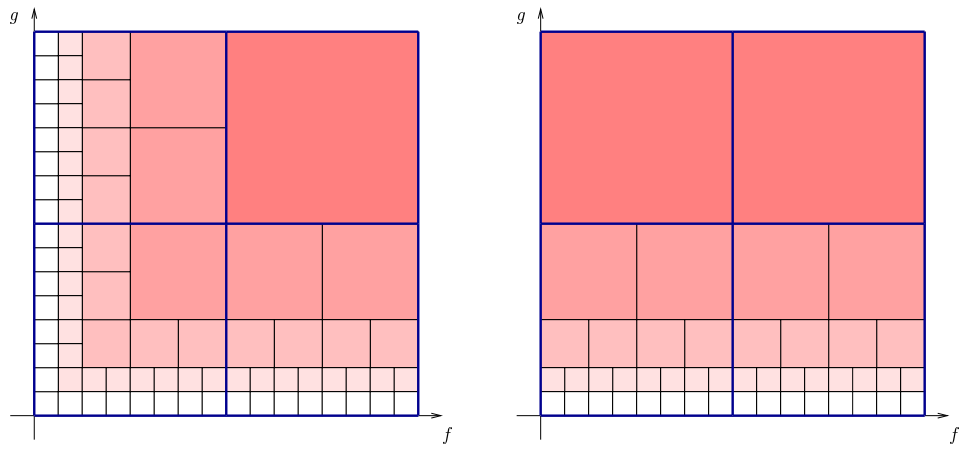

Figure 1. Illustration of the facts that (1) a full relaxed $2\,n \times 2\,n$ multiplication reduces to one full relaxed $n \times n$ multiplication, two semi-relaxed $n \times n$ multiplication and one zealous $n \times n$ multiplication (2) a semi-relaxed $2\,n \times 2\,n$ multiplication reduces to two semi-relaxed $n \times n$ multiplications and two zealous $n \times n$ multiplications.

3. A new algorithm for fast relaxed multiplication

Assume from now on that $\mathbf{C}$ admits an n-th root of unity ω_n for every power of two $n \in 2^{\mathbb{N}}$. Given an element $f \in \mathbf{C}[[z]]_n$, let $\mathrm{FFT}_n(f) \in \mathbf{C}^n$ denote its Fourier transform

$$\mathrm{FFT}_n(f) = (f(1), f(\omega_n), ..., f(\omega_n^{n-1}))$$

and let $\mathrm{FFT}_n^{-1}: \mathbf{C}^n \to \mathbf{TPS}(n)$ be the inverse mapping of FFT_n. It is well known that both FFT_n and FFT_n^{-1} can be computed in type $O(n \log n)$. Furthermore, if $f, g \in \mathbf{C}[[z]]_n$ are such that $fg \in \mathbf{C}[[z]]_n$, then

$$fg = \mathrm{FFT}_n^{-1}(\mathrm{FFT}_n(f)\,\mathrm{FFT}_n(g)),$$

where the product in $\mathbf{C}^n$ is scalar multiplication $(a_0, ..., a_{n-1})\,(b_0, ..., b_{n-1}) = (a_0\,b_0, ..., a_{n-1}\,b_{n-1})$.

Now consider a decomposition $n = n_1\,n_2$ with $n_1 = 2^{p_1}$ and $n_2 = 2^{p_2}$. Then a truncated power series $f \in \mathbf{C}[z]_n$ can be rewritten as a series

$$f_{0...n_1} + f_{n_1...2n_1}\,y + \cdots + f_{(n_2-1)n_1...n_2n_1}\,y^{n_2-1}$$

in $\mathbf{C}[z]_{n_1}[y]_{n_2}$, where $y = z^{n_1}$. This series may again be reinterpreted as a series $\mathrm{N}(f) \in \mathbf{C}[z]_{2n_1}[y]_{n_2}$, and we have

$$fg = \mathrm{N}^{-1}(\mathrm{N}(f)\,\mathrm{N}(g)),$$

where $\mathrm{N}^{-1}: \mathbf{C}[z]_{2n_1}[y] \to \mathbf{C}[z]$ is the mapping which substitutes z^{n_1} for y. Also, the FFT-transform $\mathrm{FFT}_{2n_1}: \mathbf{C}[z]_{2n_1} \to \mathbf{C}^{2n_1}$ may be extended to a mapping

$$\begin{aligned} \mathbf{C}[z]_{2n_1}[y]_l &\longrightarrow \mathbf{C}^{2n_1}[y]_l \\ c_0 + \cdots + c_{l-1}\,y^{l-1} &\longmapsto \mathrm{FFT}_d(c_0) + \cdots + \mathrm{FFT}_d(c_{l-1})\,y^{l-1} \end{aligned}$$

for each l, and similarly for its inverse $\mathrm{FFT}_{2n_1}^{-1}$. Now the formula

$$fg = \mathrm{N}^{-1}(\mathrm{FFT}_{2n_1}^{-1}(\mathrm{FFT}_{2n_1}(\mathrm{N}(f))\,\mathrm{FFT}_{2n_1}(\mathrm{N}(g))))$$

yields a way to compute fg by reusing the Fourier transforms of the "bunches of coefficients" $f_{kn_1...(k+1)n_1}$ and $g_{ln_1...(l+1)n_1}$ many times.

In the context of a semi-relaxed multiplication fg with fixed argument g, the above scheme almost reduces the computation of an $n \times n$ product with coefficients in $\mathbf{C}$ to the computation of an $n_2 \times n_2$ product with coefficients in $\mathbf{C}^{2n_1}$. The only problem which remains is that we can only compute $\mathrm{FFT}_{2n_1}(f_{kn_1...(k+1)n_1})$ when $f_{kn_1}, ..., f_{(k+1)n_1-1}$ are all known. Consequently, the products $f_{kn_1...(k+1)n_1}\,g_{0...n_1}$ should be computed apart, using a traditional semi-relaxed multiplication. In other words, we have reduced the computation of a semi-relaxed $n \times n$ product with coefficients in $\mathbf{C}$ to the computation of n_2 semi-relaxed $n_1 \times n_1$ products with coefficients in $\mathbf{C}$, one semi-relaxed $n_2 \times (n_2 - 1)$ product with coefficients in $\mathbf{C}^{2n_1}$ and $4\,n_2 - 3$ FFT-transforms of length $2\,n_1$. This has been illustrated in figure 2.

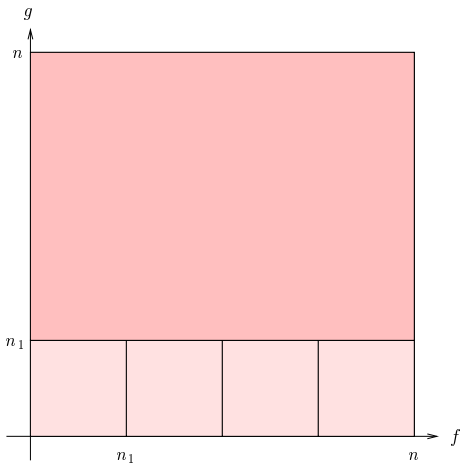

Figure 2. New decomposition of a semi-relaxed $n \times n$ multiplication into n/n_1 semi-relaxed $n_1 \times n_1$ multiplications (the light regions) and one semi-relaxed $n_2 \times (n_2 - 1)$ multiplication (the dark region) with FFT-ed coefficients in C^{2n_1}.

In order to obtain an efficient algorithm, we may choose $p_1 = \lceil p/2 \rceil$ and $p_2 = \lfloor p/2 \rfloor$:

THEOREM 2. *Assume that* C *admits an* n*-th root of unity for each* $n \in 2^{\mathbb{N}}$. *Then there exists a relaxed multiplication algorithm of time complexity* $O(n\,(\log n)^{\log 3/\log 2})$ *and space complexity* $O(n \log n)$.

PROOF. In view of section 2, it suffices to consider the case of a semi-relaxed product. Let $T(n)$ denote the time complexity of the above method. Then we observe that

$$\begin{aligned} T(n) &\leqslant n_2\, T(n_1) + 2\, n_1\, T(n_2) + O(n_2\, n_1 \log n_1) \\ &\leqslant n_2\, T(n_1) + 2\, n_1\, T(n_2) + O(n \log n). \end{aligned}$$

Taking $p_1 = \lfloor p/2 \rfloor$, $p_2 = \lceil p/2 \rceil$ and $U(p) = T(2^p)/2^p$, we obtain

$$U(p) \leqslant U(\lceil p/2 \rceil) + 2\, U(\lfloor p/2 \rfloor) + O(p),$$

from which we deduce that $U(p) = O(p^{\log 3/\log 2})$ and $T(n) = O(n\,(\log n)^{\log 3/\log 2})$. Similarly, the space complexity $S(n)$ satisfies the bound

$$S(n) \leqslant S(n_1) + 2\, n_1\, S(n_2) + O(n) \leqslant (2\, n_1 + 1)\, S(n_2) + O(n).$$

Setting $R(p) = S(2^p)/2^p$, it follows that

$$R(p) \leqslant (2 + \tfrac{1}{2^{\lfloor p/2 \rfloor}})\, R(\lceil p/2 \rceil) + O(1)$$

Consequently, $R(p) = O(p)$ and $S(n) = O(n\, p) = O(n \log n)$. □

4. FURTHER IMPROVEMENTS OF THE ALGORITHM

More generally, if $n = n_1 \cdots n_l$ with $n_1 = 2^{p_1}, \ldots, n_l = 2^{p_l}$, then we may reduce the computation of a semi-relaxed $n \times n$ product with coefficients in C into the computation of

- $\frac{n}{n_1}$ semi-relaxed $n_1 \times n_1$ products over C of the form $f_{k n_1 \ldots (k+1) n_1}\, g_{0 \ldots n_1}$;
- $2\,(\frac{n}{n_1} + n_2 - 1) - 1$ FFT-transforms of length $2\, n_1$;

- $\frac{n}{n_1\,n_2}$ semi-relaxed $n_2 \times (n_2 - 1)$ products over $\mathsf{C}^{2\,n_1}$;
- $2\,(\frac{n}{n_1\,n_2} + n_3 - 1) - 1$ FFT-transforms of length $2\,n_1\,n_2$;
- $\frac{n}{n_1\,n_2\,n_3}$ semi-relaxed $n_3 \times (n_3 - 1)$ products over $\mathsf{C}^{2\,n_1\,n_2}$;
- $\vdots$
- $4\,n_l - 3$ FFT-transforms of length $2\,\frac{n}{n_l}$;
- one semi-relaxed $n_l \times (n_l - 1)$ product over $\mathsf{C}^{2\,n_1 \cdots n_{l-1}}$.

This computation is illustrated in 3. From the complexity point of view, it leads to the following theorem:

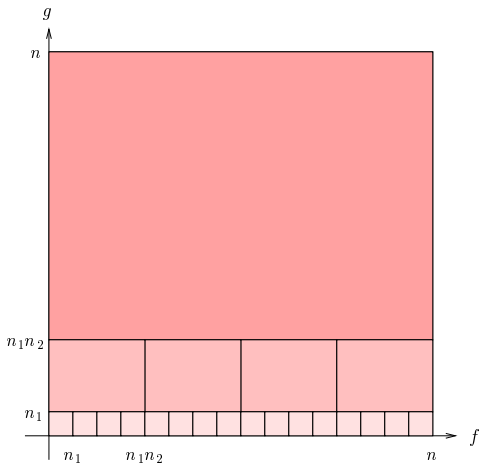

Figure 3. Generalized decomposition of a semi-relaxed $n \times n$ multiplication into $l = 3$ layers.

THEOREM 3. *Assume that* C *admits an* n*-th root of unity for each* $n \in 2^{\mathbb{N}}$. *Then there exists a relaxed multiplication algorithm of time complexity* $O(n \log n\, \mathrm{e}^{2\sqrt{\log\log n}})$ *and space complexity* $O(n\, 4^{\sqrt{\log\log n}})$.

PROOF. In view of section 2, it suffices to consider the case of a semi-relaxed product. Denoting by $T(n)$ the time complexity of the above method, we have

$$T(n) \leqslant \frac{n}{n_1}\,T(n_1) + \frac{2\,n}{n_2}\,T(n_2) + \cdots + \frac{2\,n}{n_l}\,T(n_l) + O(l\,n \log n).$$

Setting $T(n) = U(2^p)\,2^{-p}$, we obtain

$$U(p) \leqslant 2\,(U(p_1) + \cdots + U(p_l)) + O(l\,p).$$

Assuming that $p > 1$, choose $l = \lfloor \mathrm{e}^{\sqrt{\log p}} \rfloor$ and $p_i = \lfloor \frac{p + i - 1}{l} \rfloor$ for $i = 1, \ldots, l$. Then we claim that

$$U(p) = O(p\,\mathrm{e}^{2\sqrt{\log p}}). \tag{2}$$

Indeed, denoting

$$M(p) = \max_{q \leqslant p} \frac{U(q)}{q\,\mathrm{e}^{2\sqrt{\log q}}},$$

we have

$$\begin{aligned}
U(p) &\leqslant 2\,l\,U(\lceil p/l\rceil) + O(l\,p)\\
&\leqslant 2\,l\,\lceil p/l\rceil\,\mathrm{e}^{2\sqrt{\log\lceil p/l\rceil}}\,M(\lceil p/l\rceil) + O(l\,p)\\
&= 2\,(p+O(l))\,\mathrm{e}^{2\sqrt{\log p-\log l+O(1))}}\,M(\lceil p/l\rceil) + O(l\,p)\\
&= 2\,p\,\mathrm{e}^{2\sqrt{\log p-\sqrt{\log p}+O(1)}}\,(1+o(1))\,M(\lceil p/l\rceil) + O(l\,p)\\
&= 2\,p\,\mathrm{e}^{2\sqrt{\log p}-1+O(1/\sqrt{\log p})}\,(1+o(1))\,M(\lceil p/l\rceil) + O(l\,p)\\
&= \tfrac{2}{\mathrm{e}}\,p\,\mathrm{e}^{2\sqrt{\log p}}\,(1+o(1))\,M(\lceil p/l\rceil) + O(p\,\mathrm{e}^{\sqrt{\log p}})\\
&\sim \tfrac{2}{\mathrm{e}}\,p\,\mathrm{e}^{2\sqrt{\log p}}\,M(\lceil p/l\rceil)
\end{aligned}$$

In other words, there exists a p_0, such that for all $p \geqslant p_0$, we have

$$M(p) = \max_{q\leqslant p}\frac{U(q)}{q\,\mathrm{e}^{2\sqrt{\log q}}} \leqslant \max_{q\leqslant p} M(\lceil q/l\rceil).$$

We conclude that $M(p)$ is ultimately constant, whence (2) and $T(n) = O(n\log n\,\mathrm{e}^{2\sqrt{\log\log n}})$.

As to the space complexity $S(n)$, we have

$$S(n) \leqslant S(n_1) + 2\,n_1\,S(n_2) + \cdots + 2\,n_1\cdots n_{l-1}\,S(n_l) + O(n).$$

For l and $n_1, \ldots, n_l$ as above, this yields

$$S(n) \leqslant (2 + O(\tfrac{1}{n_1}))\,\tfrac{n}{n_l}\,S(n_l) + O(n)$$

for all sufficienty large n. Setting $R(p) = S(2^p)/2^p$, we obtain

$$R(p) \leqslant \left(2 + O\left(\frac{1}{\mathrm{e}^{\sqrt{\log p}}}\right)\right) R\left\lceil\frac{p}{\mathrm{e}^{\sqrt{\log p}}}\right\rceil + O(1)$$

After a similar computation as above, this implies

$$R(p) = O(4^{\sqrt{\log p}}).$$

Hence, $S(n) = O(n\,4^{\sqrt{\log\log n}})$. □

5. Conclusion

We have shown how to improve the complexity of relaxed multiplication in the case when the coefficient ring admits sufficiently many 2^p-th roots of unity. The improvement is based on reusing FFT-transforms of pieces of the multiplicands at different levels of the underlying binary splitting algorithm. For further studies, it would be interesting to study the price of artificially adding 2^p-th roots of unity, like in the Schönhage-Strassen algorithm. It might also be worth it to investigate whether a similar idea might be used for other binary splitting algorithms which involve FFT-multiplication, like multipoint evaluation or fast evaluation of holonomic functions.

Bibliography

[BK78] R.P. Brent and H.T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.

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

[CT65] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

[Knu97] D.E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 3-rd edition, 1997.

[KO63] A. Karatsuba and J. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing 7*, 7:281–292, 1971.

[vdH97] J. van der Hoeven. Lazy multiplication of formal power series. In W. W. Küchlin, editor, *Proc. ISSAC '97*, pages 17–20, Maui, Hawaii, July 1997.

[vdH02] Joris van der Hoeven. Relax, but don't be too lazy. *JSC*, 34:479–542, 2002.