Fast composition of numeric power series*

Joris van der Hoeven

CNRS, Département de Mathématiques
Bâtiment 425
Université Paris-Sud
91405 Orsay Cedex
France

Email: joris@texmacs.org
Web: http://www.math.u-psud.fr/~vdhoeven

December 9, 2008

Let f and g be two convergent power series in $\mathbb{R}[[z]]$ or $\mathbb{C}[[z]]$, whose first n terms are given numerically with a λn-bit precision for a fixed constant $\lambda > 0$. Assuming that $g_0 = 0$, we will show in this paper that the first n coefficients of $f \circ g$ can be computed with a λn-bit precision in time $\tilde{O}(n^2)$. Using Newton iteration, a similar complexity bound holds for power series reversion of g. Our method relies on fast multi-point evaluation, which will be recalled and further detailed for numeric polynomials. We also discuss relaxed variants of our algorithm.

Keywords: power series, composition, FFT, multi-point evaluation, algorithm
A.M.S. subject classification: 40-04, 42-04, 68W40

1. Introduction

Let $\mathcal{C}$ be an effective ring of coefficients (i.e. we have algorithms for performing the ring operations). Let $f, g \in \mathcal{C}[[z]]$ be two power series with $g_0 = 0$, so that the composition $h = f \circ g \in \mathcal{C}[[z]]$ is well-defined. We are interested in algorithms for fast composition: given $f_0, ..., f_{n-1}$ and $g_0, ..., g_{n-1}$, how much arithmetic operations in $\mathcal{C}$ are needed in order to compute $h_0, ..., h_{n-1}$?

A first efficient general purpose algorithm of time complexity $O(\mathsf{M}(n)\sqrt{n \log n})$ was given in [BK78, CT65, SS71]. Here $\mathsf{M}(n)$ denotes the complexity for the multiplication of two polynomials of degrees $< n$ and we have $\mathsf{M}(n) = O(n \log n \log\log n)$ [CK91]. In the case when g is polynomial [BK78] or algebraic [vdH02], then this complexity further reduces to $O(\mathsf{M}(n) \log n)$. For some very special series g, there even exist $O(\mathsf{M}(n))$ algorithms; see [BSS08] for an overview. In positive characteristic $q > 0$, right composition can also be performed in quasi-linear time $O((q/\log q)\,\mathsf{M}(n) \log n)$ [Ber98].

In this paper, we are interested in efficient algorithms when $\mathcal{C}$ is a ring of numbers, such as $\mathbb{Z}$, $\mathbb{Q}$ or a ring of floating point numbers. In that case, we are interested in the bit complexity of the composition, which means that we also have to take into account the bit precision p of the underlying integer arithmetic. In particular, we will denote by $\mathsf{I}(p)$ the time needed for multiplying two p-bit integers. We have $\mathsf{I}(p) = O(p \log p \log\log p)$ [SS71] and even $\mathsf{I}(p) = O(p \log p \log^* p)$ [Für07], where $\log^*$ satisfies $\log^*(\exp n) = \log^* n + 1$. If all coefficients $f_0, ..., f_{n-1}$, $g_0, ..., g_{n-1}$ and $h_0, ..., h_{n-1}$ correspond to p-bit numbers, then we will search for a composition algorithm which runs in quasi-linear time $\tilde{O}(n\,p) = O(n\,p\,(\log(n\,p))^{O(1)})$. Throughout the paper, we will assume that $\mathsf{I}(n)/n$ is increasing and $\mathsf{I}(O(n)) = O(\mathsf{I}(n))$.

*. This work was partially supported by the ANR Gecko project.

In section 2, we start by reviewing multiplication and division of polynomials with numeric coefficients. Multiplication of two polynomials of degrees $< n$ with p-bit coefficients can be done in time $O(\mathsf{I}(n\ p))$ using Kronecker's substitution [Kro82]. Division essentially requires a higher complexity $O(\mathsf{I}(n\ (p+n)))$ [Sch82], due to the fact that we may lose n bits of precision when dividing by a polynomial of degree n. Nevertheless, we will see that a best possible complexity can be achieved in terms of the output precision.

In section 3, we will study multi-point evaluation of a polynomial $P = P_0 + \cdots + P_{n-1}\, z^{n-1}$ at n points $z_0, \ldots, z_{n-1}$ in the case of numeric coefficients. Adapting the classical binary splitting algorithm [AHU74] of time complexity $O(\mathsf{M}(n)\ \log\ n)$ to the numeric context, we will prove the complexity bound $O(\mathsf{I}(n\,(p+n))\log n)$. In the case when $p=o(n)$, this complexity bound is again non-optimal in many cases. It would be nice if a bound of the form $O(\mathsf{I}(n\,p)\log n)$ could be achieved and we will present some ideas in this direction in section 3.2.

In section 4, we turn to the problem of numeric power series composition. Under the assumptions $p = O(n)$ and $n = O(p)$, we first show that the computation of the first n coefficients of a convergent power series is essentially equivalent to its evaluation at n equally spaced points on a given circle (this fact was already used in [Sch82], although not stated as explicitly). The composition problem can therefore be reduced to one fast Fourier transform, one multi-point evaluation and one inverse Fourier transform, leading to a complexity $O(\mathsf{I}((n+p)^2)\log(n+p))$. We first prove this result under certain normalization conditions for floating point coefficients. We next extend the result to different types of numeric coefficients (including integers and rationals) and also consider non-convergent series. The good complexity of these algorithms should not come as a full surprise: under the analogy numbers $\leftrightarrow$ series, it was already known [BK77] how to compose *multivariate* power series fast.

In the last section, we conclude by giving relaxed versions [vdH02] of our composition algorithm. This allows for the resolution of functional equations for numeric power series, involving composition. Unfortunately, we did not yet achieve a quasi-linear bound, even though some progress has been made on the exponent. The special case of power series reversion may be reduced more directly to composition, using Newton's method [BK75].

2. FUNDAMENTAL OPERATIONS

Let V be a normed vector space. Given $x, \tilde{x} \in V$ and $\varepsilon \in \mathbb{Q}^> = \{c \in \mathbb{Q}: c > 0\}$, we say that $\tilde{x}$ is an *ε-approximation* of x if $|\tilde{x} - x| \leqslant \varepsilon$. We will also say that $\tilde{x}$ is an approximation of x with *absolute error* ε or an absolute precision of $-\lceil \log_2 \varepsilon \rceil$ bits. For polynomials $P = P_d\, z^d + \cdots + P_0 \in \mathbb{C}[z]$, we will use the norm $|P| = |P_d| + \cdots + |P_0|$, and notice that $|PQ| \leqslant |P|\,|Q|$. For vectors $v = (v_0, \ldots, v_{n-1}) \in \mathbb{C}^n$, we will use the norm $|v| = \max(|v_0|, \ldots, |v_{n-1}|)$.

2.1. Multiplication

Given two polynomials $P, Q \in \mathbb{Z}[z]$ with $\deg P < n$, $\deg Q < n$, $|P| \leqslant 2^p$ and $|Q| \leqslant 2^p$, take $k = 2\ p + \lceil \log_2 n \rceil + 2$. Then the product PQ can be read off from the integer product $(PQ)(2^k) = P(2^k)\, Q(2^k)$, since the coefficients of the result all fit into k bits. This method is called Kronecker multiplication [Kro82] and its cost is bounded by $O(\mathsf{I}(n\,p))$. Clearly, the method generalizes to the case when $P, Q \in \mathbb{Z}[\mathrm{i}][z]$ or $P, Q \in \mathbb{Z}[\mathrm{i}]\,[z]\,2^{\mathbb{Z}}$. A direct consequence for the computation with floating point polynomials is the following:

LEMMA 1. *Let $A, B \in \mathbb{C}[z]$ be two polynomials with $n = \max(\deg A, \deg B) + 1$. Given a, $b, p \in \mathbb{N}$ with $|A| \leqslant 2^a$ and $|B| \leqslant 2^b$, we may compute a 2^{-p}-approximation of $A\,B$ in time $O(\mathsf{I}((p+a+b)\,n))$.*

PROOF. Let $k = p + a + b + 2$ and consider polynomials $A', B' \in \mathbb{Z}[i][z]$ with

$$\begin{aligned} |A' - 2^{k-a} A| &\leqslant 1 \\ |B' - 2^{k-b} B| &\leqslant 1 \end{aligned}$$

By assumption, the bit-sizes of the coefficients of A' and B' are bounded by k. Therefore, we may compute the exact product $A'B'$ in time $O(\mathsf{I}(kn))$, using Kronecker multiplication. We have

$$\begin{aligned} |A\,B - 2^{a+b-2k} A'\,B'| &\leqslant |A\,(B - 2^{b-k} B')| + |B\,(A - 2^{a-k} A')| + \\ &\quad |(A - 2^{a-k} A')\,(B - 2^{b-k} B')| \\ &\leqslant |A|\,|B - 2^{b-k} B'| + |B|\,|A - 2^{a-k} A'| + \\ &\quad |A - 2^{a-k} A'|\,|B - 2^{b-k} B'| \\ &\leqslant 2^{a+b-k+1} + 2^{a+b-2k} \leqslant 2^{-p}. \end{aligned}$$

This proves that $2^{a+b-2k} A'B'$ is a 2^{-p}-approximation of $A\,B$. One may optionally truncate the mantissas of the coefficients of $2^{a+b-2k} A'\,B'$ so as to fit into $p + a + b$ bits. □

REMARK 2. In order to increase readability, we will loosely use real and complex numbers as inputs of our algorithms. In practice, such inputs are really floating point numbers whose precisions should be clear from the context.

2.2. Inversion

LEMMA 3. *Let $z_0, \ldots, z_{n-1} \in \mathbb{C}$ be such that $|z_i| \leqslant 1$ for all i and*

$$P = (1 - z_0\,z) \cdots (1 - z_{n-1}\,z).$$

Let $\varphi = P^{-1} \in \mathbb{C}[[z]]$ and $\Phi = \varphi_0 + \cdots + \varphi_n\,z^n$. Let $a, b \in \mathbb{N}$ be such that $|P| \leqslant 2^a$ and $|\Phi| \leqslant 2^b$. Given $p \in \mathbb{N}$, we may compute a 2^{-p}-approximation of Φ in time $O(\mathsf{I}((p + a + b)\,n))$.

PROOF. Assume that we are given an approximation Ψ of Φ. Then we may compute a better approximation using the classical Newton iteration

$$\tilde{\Psi} = \Psi - (\Psi\,P - 1)\,\Psi. \tag{1}$$

If $\Psi\,P - 1$ is "small", then

$$\tilde{\Psi}\,P - 1 = (\Psi\,P - 1) - (\Psi\,P - 1)\,\Psi\,P = -(\Psi\,P - 1)^2 \tag{2}$$

is about twice as "small". We will apply the Newton iteration for a polynomial Ψ whose first $n/2$ coefficients are good approximations of Φ and the remaining coefficients less good approximations.

Given a polynomial $A \in \mathbb{C}[z]$, we write

$$\begin{aligned} A_{\mathrm{lo}} &= A_0 + \cdots + A_{\lceil n/2 \rceil}\,z^{\lceil n/2 \rceil} \\ A_{\mathrm{hi}} &= A_{\lceil n/2 \rceil + 1}\,z^{\lceil n/2 \rceil + 1} + \cdots + A_n\,z^n \\ A_{\mathrm{lo+hi}} &= A_{\mathrm{lo}} + A_{\mathrm{hi}}. \end{aligned}$$

Assume that $|(\Psi - \Phi)_{\mathrm{lo}}| \leqslant 2^{-p}$ and $|(\Psi - \Phi)_{\mathrm{hi}}| \leqslant 2^{-q}$, with $p \geqslant q$. Setting $E = (\Psi\,P - 1)_{\mathrm{lo}}$ and $R = (\Psi\,P - 1)_{\mathrm{hi}}$, we have

$$\begin{aligned} |E| &= |(\Psi\,P - 1)_{\mathrm{lo}}| = |((\Psi - \Phi)_{\mathrm{lo}}\,P)_{\mathrm{lo}}| \leqslant 2^{a-p} \\ |R| &= |(\Psi\,P - 1)_{\mathrm{hi}}| = |((\Psi - \Phi)_{\mathrm{lo+hi}}\,P)_{\mathrm{hi}}| \leqslant 2^{a-q+1}. \end{aligned}$$

The relation (2) yields

$$\tilde{\Psi}\, P - 1 = -E^2 - 2\, E\, R + O(z^{n+1}),$$

whence

$$\begin{aligned} |(\tilde{\Psi}\, P - 1)_{\mathrm{lo+hi}}| &\leqslant |E|\,(|E| + 2\,|R|) \leqslant 2^{a-p}\,(2^{a-p} + 2^{a-q+2}) \leqslant 2^{2a-p-q+3} \\ |(\tilde{\Psi} - \Phi)_{\mathrm{lo+hi}}| &\leqslant |(\tilde{\Psi}\, P - 1)_{\mathrm{lo+hi}}|\,|\Phi| \leqslant 2^{2a+b-p-q+3}. \end{aligned}$$

When starting with $\Psi = \Psi_{\mathrm{lo}}$, we may take $q = -b$. If $p \geqslant 4\,a + 3\,b + 6$ is sufficiently large, then one Newton iteration yields $|(\tilde{\Psi} - \Phi)_{\mathrm{lo+hi}}| \leqslant 2^{-2a-b-3}$. Applying the Newton iteration once more for $q = 2\,a + b + 3$, we obtain $|(\tilde{\Psi} - \Phi)_{\mathrm{lo+hi}}| \leqslant 2^{-p}$. In view of lemma 1 and the assumption $\mathsf{I}(O(n)) = O(\mathsf{I}(n))$, the cost of two such Newton iterations is bounded by $C\,\mathsf{I}((p+a+b)\,n)$ for a suitable constant C.

We are now in a position to prove the lemma. Since an increase of p by $O(a+b)$ leaves the desired complexity bound unaltered, we may assume without loss of generality that $p \geqslant 4\,a + 3\,b + 6$. Then the cost $T(n)$ of the inversion at order n satisfies $T(n) \leqslant T(\lceil n/2 \rceil) + C\,\mathsf{I}((p+a+b)\,n)$. We conclude that $T(n) \leqslant C\,\mathsf{I}((p+a+b)\,n) + C\,\mathsf{I}((p+a+b)\,\lceil n/2 \rceil) + \cdots \leqslant (2 + o(1))\,C\,\mathsf{I}((p+a+b)\,n)$. □

REMARK 4. If only P is given, then a constant b for the bound $|\Phi| \leqslant 2^b$ is not known *a priori*. Assume that $|\Psi - \Phi_{\mathrm{lo}}| \leqslant 2^{-p}$ and write $\Psi\, P = 1 + \Delta$. Then $|\Delta_{\mathrm{lo}}| \leqslant 2^{a-p}$ and

$$\frac{1}{P} = \frac{\Psi}{P\,\Psi} = \frac{\Psi}{1+\Delta} = \frac{\Psi}{1 + \Delta_{\mathrm{lo}} + \Delta_{\mathrm{hi}}} + O(z^{n+1}) = \frac{\Psi}{1+\Delta_{\mathrm{lo}}}\left(1 - \frac{\Delta_{\mathrm{hi}}}{1+\Delta_{\mathrm{lo}}}\right) + O(z^{n+1}).$$

Assuming $p \geqslant a + 3$, it follows that

$$|\Phi| \leqslant 2\,|\Psi|\,(1 + |\Delta_{\mathrm{hi}}|).$$

We may thus take $b = \lceil \log_2 (2\,|\Psi|\,(1 + |\Delta_{\mathrm{hi}}|)) \rceil$. Inversely, we have

$$2|\Psi|\,(1 + |\Delta_{\mathrm{hi}}|) \leqslant 2\,|\Psi|\,(2 + |\Psi|\,|P|) \leqslant 2\,|\Phi|^2\,|P|\,(1 + o(1)).$$

In other words, our choice of b is at most a factor two worse than the optimal value.

REMARK 5. The following alternative algorithm for the approximation of Φ is a variant of the method described in [Sch82, Section 4]:

- Choose $r > 0$ sufficiently small and N sufficiently large, such that

$$|\varphi_N\, z^N + \varphi_{N+1}\, z^{N+1} + \cdots| \leqslant \frac{2^{-p}\, r^n}{2\,N}, \qquad \text{for } |z| \leqslant r. \tag{3}$$

- Evaluate $\varphi = P^{-1}$ at $r, \ldots, r\,\omega^{N-1}$ where $\omega = \mathrm{e}^{2\pi \mathrm{i}/N}$, using one direct FFT for the polynomial $P(r\,z)$ and N scalar inversions.
- Let $\Psi(r\,z)$ be the polynomial of degree $< N$ we recover when applying the inverse FFT on $\varphi(r)$, ..., $\varphi(r\,\omega^{N-1})$. Then (3) implies $|\varphi(r\,z) - \Psi(r\,z)| \leqslant 2^{-p-1}\, r^n$. Consequently, $|\Phi - \Psi_n\, z^n - \cdots - \Psi_0| \leqslant 2^{-p}$.

Because of (5) below, we may always take $r = 1/8$ and $N = O(n)$, which gives a complexity bound of the form $O(\mathsf{I}(p+n)\,n \log n)$. In fact the FFT of a numeric p-bit polynomial of degree n can be computed in time $O(\mathsf{I}(p\,n))$ [Sch82, Section 3], which drops the bound further down to $O(\mathsf{I}(n\,(p+n)))$.

In practice, we may also start with $r \approx 1$ and double $1 - r$ until the computed approximation Ψ of Φ satisfies the equation $P\,\Psi = 1$ up to a sufficient precision. This leads to the same complexity bound $O(\mathsf{I}((p+a+b)\,n))$ as in the lemma.

It is not clear which of the two methods is most efficient: Newton's method performs a certain amount of recomputations, whereas the alternative method requires us to work at a sufficiently large degree $N > n$ for which (3) holds.

Given power series $a \in \mathbb{C}[[z]]$ and $b \in \mathbb{R}^{\geqslant}[[z]]$, where $\mathbb{R}^{\geqslant} = \{x \in \mathbb{R}: x \geqslant 0\}$, we will say that b *majorates* a and write $a \trianglelefteqslant b$ if $|a_i| \leqslant b_i$ for all coefficients of a. This notation applies in particular to polynomials.

LEMMA 6. *For Ψ and p as in lemma 3, we may compute a 2^{-p}-approximation of Ψ in time $O(\mathsf{I}(n\,(p+n)))$.*

PROOF. Then $P \trianglelefteqslant (1+z)^n$ and $\Phi \trianglelefteqslant \frac{1}{(1-z)^n}$, whence

$$|P| \leqslant 2^n \tag{4}$$

$$|\Phi| \leqslant \left(\frac{1}{(1-z)^{n+1}}\right)_n = \binom{2\,n}{n} \leqslant 4^n. \tag{5}$$

We conclude by applying lemma 3 for $a = n$ and $b = 2\,n$. □

2.3. Euclidean division

The following result was first proved in [Sch82, Section 4].

LEMMA 7. *Let $A, B \in \mathbb{C}[z]$ be polynomials with $|A| \leqslant 1$, $\deg A < 2\,n$ and*

$$B = (z - z_0) \cdots (z - z_{n-1})$$

for $z_0, \ldots, z_{n-1} \in \mathbb{C}$ with $|z_i| \leqslant 1$ for all i. Consider the Euclidean division

$$A = Q\,B + R,$$

with $\deg R < n$. Given $p \in \mathbb{N}$, we may compute a 2^{-p}-approximations of Q and R in time $O(\mathsf{I}(n\,(p+n)))$.

PROOF. Setting $z = 1/t$ and $m = \deg A$, we may write

$$\begin{aligned} A(z) &= z^m\,\hat{A}(t) = z^m\,(A_m + A_{m-1}\,t + \cdots + A_0\,t^m) \\ B(z) &= z^n\,\hat{B}(t) = z^n\,(B_n + B_{n-1}\,t + \cdots + B_0\,t^n). \end{aligned}$$

Setting $\hat{Q} = \hat{A}/\hat{B} \in \mathbb{C}[[t]]$, we then have

$$\begin{aligned} Q(z) &= z^{m-n}\,(\hat{Q}_0 + \hat{Q}_1\,t + \cdots + \hat{Q}_{m-n}\,t^{m-n}) \\ R &= A - Q\,B. \end{aligned}$$

Our lemma now follows from lemmas 6 and 1. □

REMARK 8. Assuming that $|A| \leqslant 2^a$, $|B| \leqslant 2^b$, $|Q| \leqslant 2^q$ and $|R| \leqslant 2^r$, it is again possible to prove the improved estimate $O(\mathsf{I}((p+a+b+q+r)\,n))$. However, the proof relies on Schönhage's original method, using the same doubling strategy as in remark 5. Indeed, when using Newton's method for inverting $\hat{B}$, nice bounds $|Q| \leqslant 2^q$ and $|R| \leqslant 2^r$ do not necessarily imply a nice bound for $|\Phi|$, where Φ is the truncated inverse of $\hat{B}$. Using a relaxed division algorithm [vdH02] for $\hat{A}/\hat{B}$, one may still achieve the improved bound, up to an additional $O(\log n)$ overhead.

For applications where Euclidean division is used as a subalgorithm, it is necessary to investigate the effect of small perturbations of the inputs A and B on the outputs Q and R.

LEMMA 9. *Consider two Euclidean divisions*

$$\begin{aligned} A &= Q\,B + R \\ \tilde{A} &= \tilde{Q}\,\tilde{B} + \tilde{R}, \end{aligned}$$

with similar hypotheses as in lemma 7. Then

$$|\tilde{R} - R| \leqslant 2^{3n+1}\,|\tilde{A} - A| + 2^{5n+1}\,|A|\,|\tilde{B} - B|.$$

PROOF. With the notations of the proof of lemma 7, let Φ and $\tilde{\Phi}$ be the truncations of the power series $\hat{B}^{-1}$ and $\hat{\tilde{B}}^{-1}$ at order n. Let us first consider the case when $\tilde{B} = B$, so that

$$\tilde{A} - A = (\tilde{Q} - Q)\,B + \tilde{R} - R.$$

Then $|\tilde{Q} - Q| \leqslant |\tilde{A} - A|\,|\Phi|$ and

$$|\tilde{R} - R| \leqslant |\tilde{A} - A|\,(1 + |B|\,|\Phi|).$$

Let us next consider the case when $\tilde{A} = A$. Then

$$\frac{1}{\hat{\tilde{B}}} - \frac{1}{\hat{B}} = \frac{\hat{\tilde{B}} - \hat{B}}{\hat{B}\,\hat{\tilde{B}}},$$

whence $|\tilde{Q} - Q| \leqslant |A|\,|\tilde{B} - B|\,|\Phi|\,|\tilde{\Phi}|$ and

$$|\tilde{R} - R| = |Q\,(B - \tilde{B}) + \tilde{B}\,(\tilde{Q} - Q)| \leqslant |A|\,|\Phi|\,|\tilde{B} - B|\,(1 + |\tilde{B}|\,|\tilde{\Phi}|).$$

In general, the successive application of the second and the first case yield

$$|\tilde{R} - R| \leqslant (|\tilde{A} - A| + |A|\,|\Phi|\,|\tilde{B} - B|)\,(1 + |\tilde{B}|\,|\tilde{\Phi}|)$$

We have also seen in the proof of lemma 6 that $|\tilde{B}| \leqslant 2^n$, $|\Phi| \leqslant 4^n$ and $|\tilde{\Phi}| \leqslant 4^n$. □

LEMMA 10. *Let A and B be as in lemma 7, while allowing for the case when $m = \deg A \geqslant 2\,n$. Then*

$$|R| \leqslant \left(\frac{4\,m}{n}\right)^n |A|.$$

PROOF. With the notations from the proof of lemma 7, we have

$$\begin{aligned} \hat{A} &\leqslant |A|\,(1-t)^{-1} \\ \hat{B}^{-1} &\leqslant (1-t)^{-n} \\ \hat{Q} &\leqslant |A|\,(1-t)^{-n-1}, \end{aligned}$$

whence

$$|Q| \leqslant |A| \binom{n+m}{n}$$

and

$$|R| \leqslant |B|\,|Q| + |A| \leqslant \left(2^n \binom{n+m}{n} + 1\right) |A| \leqslant \left(\frac{4\,m}{n}\right)^n |A|.$$

□

3. MULTI-POINT EVALUATION

Consider a complex polynomial

$$P = P_0 + \cdots + P_{n-1}\,z^{n-1} \in \mathbb{C}[z],$$

which has been normalized so that $|P| \leqslant 1$. Let

$$z_0, \ldots, z_{n-1} \in \mathbb{C}$$

be pairwise distinct points with $|z_i| \leqslant 1$ for all i. The problem of multi-point evaluation is to find an efficient algorithm for the simultaneous evaluations of P at z_0, ..., z_{n-1}. A 2^{-p}-approximation of $P(z_i)$ will also be called a 2^{-p}-*evaluation* of P at z_i. In order to simplify our exposition, we will assume that $n = 2^l$ is a power of two; this assumption only affects the complexity analysis by a constant factor.

3.1. The binary splitting algorithm

An efficient and classical algorithm for multi-point evaluation [AHU74] relies on binary splitting: let $Q_{\mathrm{lo}} = (z - z_0) \cdots (z - z_{n/2-1})$, $Q_{\mathrm{hi}} = (z - z_{n/2}) \cdots (z - z_{n-1})$. Denoting by $A \bmod B$ the remainder of the Euclidean division of A by B, we compute

$$\begin{aligned} P_{\mathrm{lo}} &= P \bmod Q_{\mathrm{lo}} \\ P_{\mathrm{hi}} &= P \bmod Q_{\mathrm{hi}} \end{aligned}$$

Then

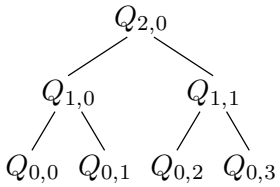
$$P(z_i) = \begin{cases} P_{\mathrm{lo}}(z_i), & \text{for } 0 \leqslant i < n/2 \\ P_{\mathrm{hi}}(z_i), & \text{for } n/2 \leqslant i < n \end{cases}$$

In other words, we have reduced the original problem to two problems of the same type, but of size $n/2$.

For the above algorithm to be fast, the partial products Q_{lo} and Q_{hi} are also computed using binary splitting, but in the opposite direction. In order to avoid recomputations, the partial products are stored in a binary tree of depth $h = \log_2 n$. At depth k, we have 2^k nodes, labeled by polynomials $Q_{h-k,0}, \ldots, Q_{h-k,2^{h-k}-1}$, where

$$Q_{k,i} = (z - z_{2^k i}) \cdots (z - z_{2^k (i+1)-1}).$$

For instance, for $n = 4$, the tree is given by



We have

$$\begin{aligned} Q_{0,i} &= z - z_i \\ Q_{k,i} &= Q_{k-1,2i}\, Q_{k-1,2i+1} \end{aligned} \tag{6}$$

For a given polynomial P of degree $< n$ we may now compute

$$\begin{aligned} P_{h,0} &= P \\ P_{k,i} &= P_{k+1,\lfloor i/2 \rfloor} \bmod Q_{k,i} \end{aligned} \tag{7}$$

This second computation can be carried out in place. At the last stage, we obtain

$$P_{0,i} = P(z_i).$$

More generally,

$$P_{k,i} = P \bmod Q_{k,i}, \tag{8}$$

for all k and i.

LEMMA 11. *Let $P \in \mathbb{C}[z]$ and $z_0, \ldots, z_{n-1} \in \mathbb{C}$ be such that $|P| \leqslant 1$, $\deg P < n$ and $|z_i| \leqslant 1$ for all i. Given $p \in \mathbb{N}$, we may compute 2^{-p}-evaluate P at $z_0, \ldots, z_{n-1}$ in time $O(\mathsf{I}(n\,(p+n))\log n)$.*

PROOF. In the algorithm, let $q = p + 5\,n + 4 = p + O(n)$ and assume that all multiplications (6) and all euclidean divisions (7) are computed up to an absolute error $\leqslant 2^{-q}$. Let $\tilde{P}_{k,i}$ and $\tilde{Q}_{k,i}$ denote the results of these approximate computations. We have

$$|\tilde{Q}_{k,i} - Q_{k,i}| \leqslant |\tilde{Q}_{k-1,2i}|\,|\tilde{Q}_{k-1,2i+1} - Q_{k-1,2i+1}| + |Q_{k-1,2i+1}|\,|\tilde{Q}_{k-1,2i} - Q_{k-1,2i}| + 2^{-q}.$$

It follows by induction that

$$\begin{aligned} |Q_{k,i}| &\leqslant 2^{2^k} \\ |\tilde{Q}_{k,i}| &\leqslant 2^{2^k+1} \qquad (9) \\ |\tilde{Q}_{k,i} - Q_{k,i}| &\leqslant 2^{2^k+2-q}. \qquad (10) \end{aligned}$$

From (8) and lemma 10, we have

$$|P_{k,i}| \leqslant \left(\frac{4\,n}{2^k}\right)^{2^k} \leqslant 4^n. \qquad (11)$$

In view of (7) and lemma 9, we also have

$$|\tilde{P}_{k,i} - P_{k,i}| \leqslant 2^{3\cdot 2^k+1}\,|\tilde{P}_{k+1,\lfloor i/2\rfloor} - P_{k+1,\lfloor i/2\rfloor}| + 2^{5\cdot 2^k+1}\,|P_{k+1,\lfloor i/2\rfloor}|\,|\tilde{Q}_{k,i} - Q_{k,i}| + 2^{-q}.$$

By induction over $h - k$, we obtain

$$|\tilde{P}_{k,i} - P_{k,i}| \leqslant (h-k)\,2^{5n+4-q}.$$

Using our choice of q, we conclude that

$$|\tilde{P}_{0,i} - P(z_i)| \leqslant 2^{-p} \qquad (0 \leqslant i < n).$$

Let us now estimate the complexity of the algorithm. In view of (9) and lemma 1, we may compute $\tilde{Q}_{k,i}$ in time $O(\mathsf{I}(2^k\,(q + O(2^k)))) = O(\mathsf{I}(2^k\,(q+n)))$. In view of (11) and lemma 7, the computation of $\tilde{P}_{k,i}$ takes a time $O(\mathsf{I}(2^k\,(q + 2\,n + 2^k))) = O(\mathsf{I}(2^k\,(q+n)))$. For fixed k, the computation of all $\tilde{Q}_{k,i}$ and $\tilde{P}_{k,i}$ thus takes a time $O(\mathsf{I}(2^k\,(q+n))\,n\,2^{-k}) = O(\mathsf{I}(n\,(q+n)))$. Since there are $h = \log_2 n$ stages, the time complexity of the complete algorithm is bounded by $O(\mathsf{I}(n\,(q+n))\log n) = O(\mathsf{I}(n\,(p+n))\log n)$. □

3.2. Low precision methods

If $p = o(n)$, then the bound from lemma 11 reduces to $O(\mathsf{I}(n^2)\log n)$, which is not very satisfactory. Indeed, in the very special case when $z_j = \mathrm{e}^{2\pi \mathrm{i} j/n}$ for $0 \leqslant j < n$, we may achieve the complexity $O(\mathsf{I}(p)\,n\log n)$, by using the fast Fourier transform. In general, we may strive for a bound of the form $O(\mathsf{I}(p\,n)\log n)$. We have not yet been able to obtain this complexity, but we will now describe some partial results in this direction. Throughout the section, we assume that $p = o(n)$.

PROPOSITION 12. *With the notations of proposition 11, assume that $|z_0| = \cdots = |z_{n-1}| = 1$. Then we may 2^{-p}-evaluate P at $z_0, \ldots, z_{n-1}$ in time $O(\mathsf{I}(p\log^3 n)\,p\,n)$.*

PROOF. Let $N = 4\,n$, $D = p+2$ and $\omega_j = \mathrm{e}^{2\pi \mathrm{i} j/N}$ for all $0 \leqslant j < N$. Each of the points z_k is at the distance of at most $1/(2\,n)$ to one of the points ω_j. We will compute $P(z_k)$ using a Taylor series expansion of P at ω_j. We first observe that

$$\frac{|P^{(i)}(\omega_j)|}{i!} \leqslant \binom{n}{i} |P| \leqslant \binom{n}{i} \leqslant n^i. \tag{12}$$

Using D fast Fourier transforms of size N, we first compute $n^i\ 2^{-D}$-approximations of $P^{(i)}(\omega_j)$ for all $i < D$ and $j < N$. This can be done in time

$$O(\mathsf{I}(D \log n)\,(D \log n)\,N \log N) = O(\mathsf{I}(p \log^3 n)\,p\,n).$$

From (12) it follows that

$$\begin{aligned} \left| P(z_k) - \sum_{i<D} \frac{P^{(i)}(\omega_j)}{i!}\,(z_k - \omega_j)^i \right| &= \left| \sum_{i \geqslant D} \frac{P^{(i)}(\omega_j)}{i!}\,(z_k - \omega_j)^i \right| \\ &\leqslant \sum_{i \geqslant D} 2^{-i} \leqslant 2^{1-D} \leqslant 2^{-p-1}. \end{aligned}$$

The 2^{-p}-evaluation of n sums of the form $\sum_{i<D} \frac{P^{(i)}(\omega_j)}{i!}\,(z_k - \omega_j)^i$ using Horner's rule takes a time $O(n\,\mathsf{I}(p \log n)\,D) = O(\mathsf{I}(p \log n)\,p\,n)$. □

PROPOSITION 13. *With the notations of proposition 11, we may 2^{-p}-evaluate P at $z_0, \ldots, z_{n-1}$ in time $O(\mathsf{I}(n^{3/2}\,p^{3/2} \log^3 n))$.*

PROOF. The proof of the above proposition adapts to the case when $1 - \frac{1}{n} < |z_k| \leqslant 1$ for all k. Let us now subdivide the unit disk into T annuli $1 - \frac{t+1}{n} < |z| \leqslant 1 - \frac{t}{n}$ and the disk of radius $1 - \frac{T}{n}$. For a fixed annulus, we may evaluate P at each of the z_k inside the annulus in time $O(\mathsf{I}(p \log^3 n)\,p\,n)$. For $|z| \leqslant 1 - \frac{T}{n}$, we have

$$\left| \sum_{i \geqslant V} P_i\,z^i \right| \leqslant \frac{n}{T} \left(1 - \frac{T}{n}\right)^V \leqslant 2^{-p-1},$$

provided that $T\,V \gg n\,p \log 2$. Consequently, taking $V \geqslant p$, we may evaluate P at the remaining points in time $O((n/V)\,\mathsf{I}(V^2) \log V)$. Under the condition $T\,V \gg n\,p \log 2$, and up to logarithmic terms, the sum

$$T\,\mathsf{I}(p \log^3 n)\,p\,n + (n/V)\,\mathsf{I}(V^2) \log V$$

becomes minimal for $T = O(n^{1/2}\,p^{-1/2})$ and $V = O(n^{1/2}\,p^{3/2})$. □

PROPOSITION 14. *With the notations of proposition 11, assume that there exists an $\omega \in \mathbb{C}$ with $|\omega| \leqslant 1$ and $|z_i - \omega| \leqslant 1/(2\,n)$ for all i. Then we may 2^{-p}-evaluate P at $z_0, \ldots, z_{n-1}$ in time $O(\mathsf{I}(n\,p \log n))$.*

PROOF. We will only provide a sketch of the proof. As in the proof of proposition 12, we first compute the Taylor series expansion of P at ω up to order $D = p+2$. This task can be performed in time $O(\mathsf{I}(n\,(p + p \log n)))$ via a division of P by $(z-\omega)^D$ followed by a Taylor shift. We next evaluate this expansion at $z_i - \omega$ for all i. According to proposition 16, this can be done in time $O(\mathsf{I}(n\,p) \log p)$. □

PROPOSITION 15. *Let $c > 0$ be a fixed constant. With the notations of proposition 11, assume that $|z_j - \mathrm{e}^{2\pi \mathrm{i} j/n}| < c/n$ for all j. Then we may 2^{-p}-evaluate P at $z_0, \ldots, z_{n-1}$ in time $O(\mathsf{I}(p + \log^2 n)\,n \log^2 n)$.*

PROOF. Again, we will only provide a sketch of the proof. The main idea is to use the binary splitting algorithm from the previous subsection. Let $\omega = \mathrm{e}^{2\pi \mathrm{i}/n}$. If $z_i = \omega^i$ for all i, then we obtain $Q_{k,i} = z^{2^k} - \omega^{i2^k}$ and the algorithm reduces to a fast Fourier transform. If each z_i is a slight perturbation of ω^i, then $Q_{k,i}$ becomes a slight perturbation of $z^{2^k} - \omega^{i2^k}$. In particular, the Taylor coefficients $(\hat{Q}_{k,i}^{-1})_N$ of $\hat{Q}_{k,i}^{-1}$ only have a polynomial growth $O(N^{O(c)})$ in N. Consequently, the Euclidean division by $Q_{k,i}$ accounts for a loss of at most $O(\log n)$ instead of $O(n)$ bits of precision. The binary splitting algorithm can therefore be carried out using a fixed precision of $p + O(\log^2 n)$ bits. □

Sometimes, it is possible to decompose $n = n_1 + \cdots + n_k$, such that a fast multi-point evaluation algorithm is available for each set of points $\{z_{n_1+\cdots+n_{i-1}}, \ldots, z_{n_1+\cdots+n_i-1}\}$. This leads to the question of evaluating P at $m < n$ points.

PROPOSITION 16. *Let $P = P_0 + \cdots + P_{n-1}\, z^{n-1} \in \mathbb{C}[z]$ and $z_1, \ldots, z_m \in \mathbb{C}$ be such that $m < n$, $|P| \leqslant 1$ and $|z_i| \leqslant 1$ for all i. Let $\mathsf{E}(m, p)$ be the time needed in order to compute 2^{-p}-evaluations of polynomials $Q = Q_0 + \cdots + Q_{m-1} z^{m-1} \in \mathbb{C}[z]$ with $|Q| \leqslant 1$ at $z_0, \ldots, z_{m-1}$. Then 2^{-p}-evaluations of P at $z_0, \ldots, z_{m-1}$ can be computed in time $O(\mathsf{E}(m, p)\, n/p + \mathsf{I}(p)\, n)$.*

PROOF. Without loss of generality, we may assume that m and n are powers of two. We decompose P as follows:

$$\begin{aligned} P &= \Pi_0 + \Pi_1\, z^m + \cdots + \Pi_{n/m-1}\, z^{(n/m-1)m} \\ \Pi_i &= P_{im} + \cdots + P_{(i+1)m-1}\, z^{m-1}. \end{aligned}$$

We may compute 2^{-p-2}-evaluations of the Π_i at $z_0, \ldots, z_{m-1}$ in time $O(\mathsf{E}(m, p)\, n/m)$. We may also 2^{-p-2}-approximate $z_0^m, \ldots, z_{n/m-1}^m$ in time $O(\mathsf{I}(p)\, n \log m/m) = O(\mathsf{I}(p)\, n)$, using binary powering. Using Horner's rule, we may finally 2^{-p}-evaluate

$$P(z_i) = \Pi_0(z_i) + \Pi_1(z_i)\, z_i^m + \cdots + \Pi_{n/m-1}(z_i)\, (z_i^m)^{n/m-1}$$

for $i = 0, \ldots, m-1$ in time $O(m\, \mathsf{I}(p)\, n/m) = O(\mathsf{I}(p)\, n)$. □

4. COMPOSITION OF POWER SERIES

4.1. Evaluation of truncated power series

Let $f = f_0 + f_1 z + \cdots$ be a convergent power series with radius of convergence $\rho_f > 0$. Given $R < \rho_f$, we denote

$$\|f\|_R = \sup_{|z| \leqslant R} |f(z)|.$$

Using Cauchy's formula, we have

$$|f_n| = \left| \frac{1}{2\,\pi\,\mathrm{i}} \int_{|z|=R} \frac{f(t)\,\mathrm{d}t}{t^{n+1}} \right| \leqslant \frac{\|f\|_R}{R^n}, \tag{13}$$

For $z \in \mathbb{C}$ with $|z| \leqslant r < R$, it follows that

$$|f_n\, z^n + f_{n+1}\, z^{n+1} + \cdots| \leqslant \frac{\|f\|_R}{1 - r/R} \left(\frac{r}{R}\right)^n. \tag{14}$$

Let $\mathsf{T}_1(n, p)$ be the time needed to compute 2^{-p}-approximations of $f_0, \ldots, f_{n-1}$. Given $r < R$, let $\mathsf{T}_2(n, p)$ be the time needed to 2^{-p}-evaluate f at $r\,\mathrm{e}^{2\pi \mathrm{i} k/n}$ for $k \in \{0, \ldots, n-1\}$.

LEMMA 17.

a) *We have* $\mathsf{T}_2(n,p) = \mathsf{T}_1(O(p),O(p)) + O(n\,\mathsf{I}(p)\log p)$.

b) *We have* $\mathsf{T}_1(n,p) = \mathsf{T}_2(O(p),O(p)) + O(\mathsf{I}(p)\,p\log p + n)$.

PROOF. We first consider the case when $r = 1$. Let N be the smallest power of two with

$$N \geqslant \frac{p + \log_2\left(\frac{2\,\|f\|_R}{1-1/R}\right)}{\log R}. \tag{15}$$

For all $|z| \leqslant 1$, the bound (14) implies

$$|f(z) - f_0 - \cdots - f_{N-1}\,z^{N-1}| \leqslant 2^{-p-1}. \tag{16}$$

The computation of $2^{-p-\log_2 N-2}$-approximations of f_0, ..., f_{N-1} takes a time $\mathsf{T}_1(N,\ p + O(\log\ p))$. The 2^{-p-1}-evaluation of $f_0 + \cdots + f_{N-1}\ z^{N-1}$ at N primitive roots of unity can be done using $\lceil n/N \rceil$ fast Fourier transforms of size N. This requires a time $O(n\,\mathsf{I}(p)\log p)$. If $r = 1$, we thus obtain the bound

$$\mathsf{T}_2(n,p) = \mathsf{T}_1(O(p), p + O(\log p)) + O(n\,\mathsf{I}(p)\log p).$$

The general case is reduced to the case $r = 1$ via a change of variables $z = r\,z'$. This requires computations to be carried out with an additional precision of $-N\log_2 r = O(N) = O(p)$ bits, leading to the complexity bound in (a).

As to (b), we again start with the case when $r = 1$. Taking N to be the smallest power of two with (15), we again obtain (16) for all $|z| \leqslant 1$. If $N < n$, then we also notice that $|f_i| \leqslant 2^{-p-1}$ for all $i \geqslant N$. We may 2^{-p-2}-evaluate $f_0 + \cdots + f_{N-1}\ z^{N-1}$ at the primitive N-th roots of unity in time $\mathsf{T}_2(N,\ p + 2)$. We next retrieve the coefficients of the polynomial $f_0 + \cdots + f_{N-1}\,z^{N-1}$ up to precision $\leqslant 2^{-p-1}$ using one inverse fast Fourier transform of size N. In the case when $r = 1$, we thus obtain

$$\mathsf{T}_1(n,p) = \mathsf{T}_2(O(p), p + O(1)) + O(\mathsf{I}(p)\,p\log p + n).$$

In general, replacing p by $p + O(N) = O(p)$ yields the bound in (b). □

4.2. Composition of power series

Let us now consider two convergent power series $f, g \in \mathbb{C}[[z]]$ with $g_0 = 0$, $\|f\|_1 \leqslant 1$ and $\|g\|_1 \leqslant 1$. Then $h = f \circ g$ is well-defined and $\|h\|_1 \leqslant 1$. In fact, the series f, g and h still converge on a compact disc of radius $R > 1$. Let $B \in \mathbb{R}^{>}$ be such that $\|f\|_R \leqslant B$, $\|g\|_R \leqslant B$ and $\|h\|_R \leqslant B$. Then (14) becomes

$$|f_N z^N + f_{N+1}\,z^{N+1} + \cdots| \leqslant \frac{B}{1-1/R}\,R^{-N} \tag{17}$$

and similarly for g and h. In view of lemma 17, it is natural to use an evaluation-interpolation scheme for the computation of 2^{-p}-approximations of $h_0, \ldots, h_{n-1}$.

For a sufficiently large N and $\omega = \mathrm{e}^{2\pi\mathrm{i}/N}$, we evaluate g on the N-th roots of unity $1, \omega, \ldots, \omega^{N-1}$ using one direct FFT on $g_0, \ldots, g_{N-1}$. Using the fact that $|g(\omega^i)| \leqslant 1$ for all i and the algorithm for multi-point evaluation from the previous section, we next compute $h(1), \ldots, h(\omega^{N-1})$. Using one inverse FFT, we finally recover the coefficients of the polynomial $H = H_0 + \cdots + H_{N-1}\,z^{N-1}$ with $H(\omega^i) = h(\omega^i)$ for all $i < N$. Using the tail bound (17) for f, g, h and a sufficiently large $N = O(p)$, the differences $|H_i - h_i|$ can be made as small as needed. More precisely, the algorithm goes as follows:

Algorithm `compose`(f, g, n, p)

INPUT: $f, g \in \mathbb{C}[[z]]$ with $g_0 = 0$ and $n, p \in \mathbb{N}$,
such that we have bounds $\|f\|_1 \leqslant 1$, $\|g\|_1 \leqslant 1$ and
$\|f\|_R \leqslant B$, $\|g\|_R \leqslant B$ and $\|f \circ g\|_R \leqslant B$ for certain $B, R > 1$
OUTPUT: 2^{-p}-approximations for $(f \circ g)_0, ..., (f \circ g)_{n-1}$

Step 1. [Determine auxiliary degree and precision]
Let $N \in 2^{\mathbb{N}}$ be smallest with $N \geqslant \left(p + 2 + \log_2 \frac{Bn}{1-1/R}\right) / \log_2 R$ and $N \geqslant n$
Let $q := N \log_2 R - \log_2 \frac{B}{R-1}$, so that $2^{-q} = \frac{B R^{-N}}{1-1/R}$

Step 2. [Evaluate g on roots of unity $1, ..., \omega^{N-1}$]
Let $\omega := \mathrm{e}^{2\pi \mathrm{i}/N}$
Compute a 2^{-q}-approximation $(z_0, ..., z_{N-1}) \approx \mathrm{FFT}_\omega(g_0, ..., g_{N-1})$ with $|z_i| \leqslant 1$
We will show that $|z_i - g(\omega^i)| \leqslant 2^{1-q}$ for all i

Step 3. [Evaluate f on $g(1), ..., g(\omega^{N-1})$]
Let $F := f_0 + \cdots + f_{N-1} z^{N-1}$
Compute a 2^{-q}-approximation $(v_0, ..., v_{N-1}) \approx (F(z_0), ..., F(z_{N-1}))$
We will show that $|v_i - f(g(\omega^i))| \leqslant (2 + 2n)\ 2^{-q}$ for all i

Step 4. [Interpolate]
Compute a 2^{-q}-approximation $(u_0, ..., u_{N-1}) \approx \mathrm{FFT}_\omega^{-1}(v_0, ..., v_{N-1})$
We will show that $|u_i - (f \circ g)_i| \leqslant (4 + 2n)\ 2^{-q}$ for all $i < n$
Return $(u_0, ..., u_{n-1})$

THEOREM 18. *Let $f, g, h \in \mathbb{C}[[z]]$ be power series with $g_0 = 0$ and $h = f \circ g$. Assume that $\|f\|_1 \leqslant 1$ and $\|g\|_1 \leqslant 1$. Given $p \in \mathbb{N}$, we may compute 2^{-p}-approximations for $h_0, ..., h_{n-1}$ (as a function of $f_0, ..., f_{n-1}$ and $g_0, ..., g_{n-1}$) in time $O(\mathsf{I}((n+p)^2) \log(n+p))$.*

PROOF. Let us first prove the correctness of the algorithm. The choice of q and the tail bound (17) for g imply $|g_N z^N + g_{N+1} z^{N+1} + \cdots| \leqslant 2^{-q}$. This ensures that we indeed have $|z_i - g(\omega^i)| \leqslant 2^{1-q}$ for all i at the end of step 2. For $i < N$, we also have

$$\begin{aligned} |v_i - h(\omega^i)| &\leqslant |v_i - F(z_i)| + |F(z_i) - F(g(\omega^i))| + |F(g(\omega^i)) - f(g(\omega^i))| \\ &\leqslant 2^{-q} + \|F'\|_1 |z_i - g(\omega^i)| + \frac{\|f\|_R R^{-N}}{1-1/R} \\ &\leqslant (2 + 2n)\ 2^{-q}. \end{aligned}$$

This proves the bound stated at the end of step 3. As to the last bound, let $H = h_0 + \cdots + h_{N-1} z^{N-1}$. Then $|H(\omega^i) - h(\omega^i)| \leqslant 2^{-q}$ and $(h_0, ..., h_{N-1}) = \mathrm{FFT}_\omega^{-1}(H(1), ..., H(\omega^{N-1}))$. Using the fact that $|\mathrm{FFT}_\omega^{-1}(a)| \leqslant |a|$ for all vectors a of length N, we obtain

$$\begin{aligned} |u_i - h_i| &\leqslant |\mathrm{FFT}_\omega^{-1}(H(1) - h(1), ..., H(\omega^{n-1}) - h(\omega^{n-1}))_i| + \\ &\quad |\mathrm{FFT}_\omega^{-1}(v_0 - h(1), ..., v_{n-1} - h(\omega^{n-1}))_i| + \\ &\quad |u_i - \mathrm{FFT}_\omega^{-1}(v_0, ..., v_{n-1})_i| \\ &\leqslant (4 + 4n)\, 2^{-q}. \end{aligned}$$

This proves the second bound. Our choice of N implies the correctness of the algorithm.

As to the complexity bound, the FFT transform in step 2 can be done in time

$$O(\mathsf{I}(q)\, N \log N) = O(\mathsf{I}(p + O(\log N))\, N \log N) = O(\mathsf{I}(N\,(p+N)) \log N).$$

By lemma 11, the multi-point evaluation in step 3 can be performed in time

$$O(\mathsf{I}(N\,(q+N)) \log N) = O(\mathsf{I}(N\,(p + O(N))) \log N) = O(\mathsf{I}(N\,(p+N)) \log N).$$

The inverse FFT transform in step 4 can be performed within the same time as a direct FFT transform. This leads to the overall complexity bound

$$O(\mathsf{I}(N(p+N))\log N) = O(\mathsf{I}((n+p)^2)\log(n+p)),$$

since $N = O(n+p)$. □

4.3. Variants of the main theorem

COROLLARY 19. *Let $f, g, h \in \mathbb{C}[[z]]$ be convergent power series with $g_0 = 0$ and $h = f \circ g$. Given $p \in \mathbb{N}$, we may compute 2^{-p}-approximations for $h_0, \ldots, h_{n-1}$ (as a function of $f_0, \ldots, f_{n-1}$ and $g_0, \ldots, g_{n-1}$) in time $O(\mathsf{I}((n+p)^2)\log(n+p))$.*

PROOF. Let $R > 0$ be sufficiently small such that $S = \|g\|_R < +\infty$ and $T = \|f\|_S < +\infty$. Consider the series $G(z) = g(Rz)/S$ and $F(z) = f(Sz)/T$, so that

$$(f \circ g)(z) = T(F \circ G)(z/R).$$

Let $k = \lceil p - n\log_2 \min(1, R) + \log_2 \max(T, 1) + 1 \rceil$. By the theorem, we may compute 2^{-k}-approximations $\tilde{H}_0, \ldots, \tilde{H}_{n-1}$ of $(F \circ G)_0, \ldots, (F \circ G)_{n-1}$ in time $O(\mathsf{I}((n+k)^2)\log(n+k)) = O(\mathsf{I}((n+p)^2)\log(n+p))$. Using $O(n)$ additional k-bit multiplications, we may compute

$$\tilde{h_0} = T\tilde{H}_0, \ldots, \tilde{h_{n-1}} = T\frac{\tilde{H}_{n-1}}{R^{n-1}}$$

with $|\tilde{h_i} - h_i| \leqslant 2^{1-k}\, T/\min(1, R)^{n-1} \leqslant 2^{-p}$ for all i. This can again be done in time $O(n\,\mathsf{I}(k)) = O(\mathsf{I}((n+p)^2))$. □

COROLLARY 20. *Let $f, g \in \mathbb{C}[[z]]$ be convergent power series with $g_0 = 0$, such that $h = f \circ g \in \mathbb{N}[[z]]$ is an ordinary generating function. Then we may compute $h_0, \ldots, h_{n-1}$ (as a function of $f_0, \ldots, f_{n-1}$ and $g_0, \ldots, g_{n-1}$) in time $O(\mathsf{I}(n^2)\log n)$.*

PROOF. It suffices to take $p = 2$ in corollary 19 and round the results. □

COROLLARY 21. *Let $f, g \in \mathbb{C}[[z]]$ be convergent power series with $g_0 = 0$ such that $h = f \circ g = \sum_{n \in \mathbb{N}} \frac{h_n}{n!} z^n$ is an exponential generating function. Then we may compute $h_0, \ldots, h_{n-1} \in \mathbb{N}$ (as a function of $f_0, \ldots, f_{n-1}$ and $g_0, \ldots, g_{n-1}$) in time $O(\mathsf{I}(n^2 \log^2 n))$.*

PROOF. Applying corollary 19 for $p = \lceil \log_2 n! \rceil + 2 = O(n \log n)$, we obtain $(2\,n!)^{-1}$-approximations $\tilde{\varphi}_0, \ldots, \tilde{\varphi}_{n-1}$ of $h_0, \ldots, h_{n-1}/(n-1)!$ in time $O(\mathsf{I}(n^2 \log^2 n))$. Using $O(n)$ additional p-bit multiplications of total cost $O(n\,\mathsf{I}(n \log n)) = O(\mathsf{I}(n^2 \log n))$, we may compute

$$\tilde{h_0} = 0!\,\tilde{\varphi}_0, \ldots, \tilde{h_{n-1}} = (n-1)!\,\tilde{\varphi}_{n-1},$$

with $|\tilde{h_i} - h_i| < \frac{1}{2}$ for all i. The h_i are obtained by rounding the $\tilde{h_i}$ to the nearest integers. □

COROLLARY 22. *Let $f, g, h \in \mathbb{C}[[z]]$ be power series with $g_0 = 0$ and $h = f \circ g$. Let u_n and v_n be positive increasing functions with $|f_n| \leqslant 2^{u_n}$ and $|g_n| \leqslant 2^{n v_n}$. Given $p \in \mathbb{N}$, we may compute 2^{-p}-approximations for $h_0, \ldots, h_{n-1}$ (as a function of $f_0, \ldots, f_{n-1}$ and $g_0, \ldots, g_{n-1}$) in time $O(\mathsf{I}((n + u_n + n\,v_n + p)^2)\log(n + u_n + n\,v_n + p))$.*

PROOF. Without loss of generality, we may assume that $f_i = g_i = 0$ for $i \geqslant n$. In the proof of corollary 19, we may therefore choose $R = 2^{-v_n - 1}$, which yields $S \leqslant 2$ and $T \leqslant 2^{u_n + n}$. It follows that $k = \lceil p - n\log_2 \min(1, R) + \log_2 \max(T, 1) + 1 \rceil \leqslant p + u_n + n\,v_n + O(n)$ and we conclude in a similar way as in the proof of corollary 19. □

5. RELAXED COMPOSITION

Let $f, g, h \in \mathbb{C}[[z]]$ be such that $g_0 = 0$ and $h = f \circ g$. Given an order $n \in \mathbb{N}$, we have shown in the previous section how to compute $h_0, ..., h_{n-1}$ efficiently, as a function of $f_0, ..., f_{n-1}$ and $g_0, ..., g_{n-1}$. Since h_i only depends on $f_0, ..., f_i$ and $g_0, ..., g_i$, we may consider the *relaxed composition problem* in which $f_0, ..., f_{n-1}$ and $g_0, ..., g_i$ are given progressively, and where we require each coefficient h_i to be output as soon as $f_0, ..., f_i$ and $g_0, ..., g_i$ are known. Similarly, in the *semi-relaxed composition problem*, the coefficients $g_0, ..., g_{n-1}$ are known beforehand, but $f_0, ..., f_{n-1}$ are given progressively. In that case, we require h_i to be output as soon as $f_0, ..., f_i$ are known. The relaxed and semi-relaxed settings are particularly useful for the resolution of implicit equations involving functional composition. We refer to [vdH02, vdH07] for more details about relaxed computations with power series.

5.1. Semi-relaxed composition

Let $n \in 2^{\mathbb{N}}$. In this section, we consider the problem of computing the semi-relaxed composition $h = f \circ g$ up to order n. If $n = 1$, then we simply have $h_0 = f_0$. For $n \geqslant 2$, we denote

$$\begin{aligned} f_{\text{lo}} &= f_0 + \cdots + f_{n/2-1}\, z^{n/2-1} \\ f_{\text{hi}} &= f_{n/2} + \cdots + f_{n-1}\, z^{n/2-1}, \end{aligned}$$

and similarly for g and h. Our algorithm relies on the identity

$$h = f_{\text{lo}} \circ g + (f_{\text{hi}} \circ g)\, (g/z)^{n/2}\, z^{n/2} + O(z^n). \tag{18}$$

The first part h_{lo} of h is computed recursively, using one semi-relaxed composition at order $n/2$:

$$h_{\text{lo}} = f_{\text{lo}} \circ g_{\text{lo}} + O(z^{n/2}).$$

As soon as f_{lo} is completely known, we compute $f_{\text{lo}} \circ g$ at order $n/2$, using one of the algorithms for composition from the previous section. We also compute $(g/z)^{n/2}$ at order $n/2$ using binary powering. We are now allowed to recursively compute the second part h_{hi} of h, using

$$h_{\text{hi}} = (f_{\text{lo}} \circ g)_{\text{hi}} + (f_{\text{hi}} \circ g_{\text{lo}})\, (g/z)^{n/2} + O(z^{n/2}).$$

This involves one semi-relaxed composition $f_{\text{hi}} \circ g_{\text{lo}}$ at order $n/2$ and one semi-relaxed multiplication $(f_{\text{hi}} \circ g_{\text{lo}})\, (g/z)^{n/2}$ at order $n/2$.

Assume now that $|f_i| \leqslant 1$ for all i and $\|g\|_1 \leqslant 1$. Consider the problem of computing 2^{-p}-approximations of $h_0, ..., h_{n-1}$. In our algorithm, it will suffice to conduct the various computations with the following precisions:

- We recursively compute 2^{-p}-approximations of $(f_{\text{lo}} \circ g_{\text{lo}})_0, ..., (f_{\text{lo}} \circ g_{\text{lo}})_{n/2-1}$.
- We compute 2^{-p-1}-approximations of $(f_{\text{lo}} \circ g)_{n/2}, ..., (f_{\text{lo}} \circ g)_{n-1}$.
- We recursively compute 2^{-p-n-2}-approximations of $(f_{\text{hi}} \circ g_{\text{lo}})_0, ..., (f_{\text{hi}} \circ g_{\text{lo}})_{n/2-1}$.
- We compute $2^{-p-\lceil \log_2 n \rceil - 2}$-approximations of the coefficients of $((g/z)^{n/2})_{\text{lo}}$.
- We compute 2^{-p-1}-approximations of the coefficients of $((f_{\text{hi}} \circ g)\, (g/z)^{n/2})_{\text{lo}}$.

Let us show that this indeed enables us to obtain the desired 2^{-p}-approximations for $h_0, ..., h_{n/2-1}$. This is clear for the coefficients of h_{lo}. As to the second half, we have the bounds

$$\begin{aligned} \|f_{\text{hi}}\|_1 &\leqslant (n/2)\, |f_{\text{hi}}| \leqslant n/2 \\ \|f_{\text{hi}} \circ g\|_1 &\leqslant \|f_{\text{hi}}\|_{\|g\|_1} \leqslant n/2 \\ |(f_{\text{hi}} \circ g)_{\text{lo}}| &\leqslant \|f_{\text{hi}} \circ g\|_1 \leqslant n/2 \end{aligned}$$

and

$$\begin{aligned}(g_{\rm lo}/z)^{n/2} &\leqslant \frac{1}{(1-z)^{n/2}}\\ |((g_{\rm lo}/z)^{n/2})_{\rm lo}| &\leqslant 2^n.\end{aligned}$$

These bounds justify the extra number of bits needed in the computations of $(g/z)^{n/2}$ and $(f_{\rm hi}\circ g_{\rm lo})_{\rm lo}$ respectively.

Let us finally analyze the complexity of the above algorithm. We will denote by $\mathsf{M}(n,p)$ the complexity of multiplying two p-bit integer polynomials of degrees $<n$. Using Kronecker multiplication, we have $\mathsf{M}(n,\ p) = O(\mathsf{I}(n\ p))$. We denote by $\mathsf{M}_{\rm semi}(n,\ p)$ the cost of a semi-relaxed multiplication of two p-bit integer polynomials of degrees $<n$. Using the fast relaxed multiplication algorithm from [vdH02], we have $\mathsf{M}_{\rm semi}(n,\ p) = O(\mathsf{M}(n,\ p)\ \log\ n)$. We will denote by $\mathsf{C}(n,\ p)$ and $\mathsf{C}_{\rm semi}(n,\ p)$ the cost of classical and semi-relaxed composition for f, g, n and p as described above. By theorem 18, we have $\mathsf{C}(n,\ p) = O\ (\mathsf{I}((n\ +\ p)^2)\ \log(n\ +\ p))$. The complexity of the semi-relaxed composition satisfies the following recursive bound:

$$\begin{aligned}\mathsf{C}_{\rm semi}(n,p) &\leqslant \mathsf{C}_{\rm semi}(n/2,p+1)+\mathsf{C}_{\rm semi}(n/2,p+n+2)+\mathsf{C}(n,p)+\\ &\quad \mathsf{M}(n,p+n+2)\log_2 n+\mathsf{M}_{\rm semi}(n/2,p+n+2)+O(n\,(p+n))\\ &\leqslant 2\,\mathsf{C}_{\rm semi}(n/2,p+n+2)+\mathsf{C}(n,p)+O(\mathsf{I}(n\,(p+n))\log n).\end{aligned}$$

Using theorem 18, it follows by induction that

$$\mathsf{C}_{\rm semi}(n,p)\leqslant 2^k\,\mathsf{C}_{\rm semi}(n\,2^{-k},p+O(n))+O(2^k\,\mathsf{I}((n+p)^2)\log(n+p)). \tag{19}$$

From [BK75, vdH02], we also have

$$\mathsf{C}_{\rm semi}(n,p)=O(\mathsf{I}(n\,(p+n))\,\sqrt{n\log^3 n})). \tag{20}$$

Applying (19) for $2^k\approx n^{1/5}$ and (20) on $\mathsf{C}_{\rm semi}(n\,2^{-k},p+O(n))$, we obtain

$$\begin{aligned}\mathsf{C}_{\rm semi}(n,p) &= O(n^{1/5}\,\mathsf{I}((n+p)^2)\log(n+p))+O(\mathsf{I}(n^{4/5}\,(n+p))\,n^{2/5}\log^{3/2} n))\\ &= O(\mathsf{I}((n+p)^{11/5})\log^{3/2}(n+p)).\end{aligned}$$

We have proved the following theorem:

THEOREM 23. *Let f, g, $h\in\mathbb{C}[[z]]$ be power series with $g_0=0$ and $p\in\mathbb{N}$. Assume that $\|f\|_1\leqslant 1$ and $\|g\|_1\leqslant 1$. The computation of the semi-relaxed composition $h=f\circ g$ at order n and up to an absolute error $\leqslant 2^{-p}$ can be done in time $O(\mathsf{I}((n+p)^{11/5})\log^{3/2}(n+p))$.*

5.2. Relaxed composition

Let $f,g\in\mathbb{C}[[z]]$ be as in the previous section and $n\in 2^{\mathbb{N}}$. Assume also that $g_1\neq 0$. In order to compute the relaxed composition $h=f\circ g$, we again use the formula (18), in combination with

$$f_{\rm lo}\circ g=f_{\rm lo}\circ g_{\rm lo}+\frac{(f_{\rm lo}\circ g_{\rm lo})'}{g_{\rm lo}'}\,g_{\rm hi}\,z^{n/2}+O(z^n)$$

The first $n/2$ coefficients of $f\circ g$ are still computed recursively, by performing a relaxed composition $f_{\rm lo}\circ g_{\rm lo}$ at order $n/2$. As soon as $f_{\rm lo}$ and $g_{\rm lo}$ are completely known, we may compute the composition $f_{\rm lo}\circ g_{\rm lo}$ at order n using the algorithm from section 4. Differentiation and division by $g_{\rm lo}'$ also yields $(f_{\rm lo}\circ g_{\rm lo})'/g_{\rm lo}'$ at order $n/2$. The product $((f_{\rm lo}\circ g_{\rm lo})'/g_{\rm lo}')\,g_{\rm hi}$ can therefore be computed using one semi-relaxed multiplication.

Let $q = p + 2\,n + 2\,\lceil \log_2 n \rceil + 2$. This time, the intermediate computations should be conducted with the following precisions:

- We recursively compute 2^{-q}-approximations of the coefficients of $h_{\rm lo}$.
- We compute 2^{-q-1}-approximations of the coefficients of $g_{\rm hi}$.
- We compute 2^{-p-3}-approximations of the coefficients of $((f_{\rm lo} \circ g_{\rm lo})'/g'_{\rm lo})_{\rm lo}$.
- We compute 2^{-p-2}-approximations of the coefficients of $(((f_{\rm lo} \circ g_{\rm lo})'/g'_{\rm lo})\, g_{\rm hi})_{\rm lo}$.

Indeed, we have

$$\begin{aligned}
g'_{\rm lo} &\trianglelefteqslant (1-z)^{-2} \\
((g'_{\rm lo})^{-1})_{\rm lo} &\trianglelefteqslant (1-z)^{-n} \\
|((g'_{\rm lo})^{-1})_{\rm lo}| &\leqslant 4^n \\
|((f_{\rm lo} \circ g_{\rm lo})')_{\rm lo}| &\leqslant (n/2)\,|(f \circ g)_{\rm lo}| \leqslant n/2 \\
|((f_{\rm lo} \circ g_{\rm lo})'/g'_{\rm lo})_{\rm lo}| &\leqslant n^2\, 4^n \leqslant 2^{q-p-2}.
\end{aligned}$$

Denoting by $\mathsf{C}_{\rm rel}(n,p)$ the complexity of relaxed composition, we obtain

$$\begin{aligned}
\mathsf{C}_{\rm rel}(n,p) &\leqslant \mathsf{C}_{\rm rel}(n/2,p) + \mathsf{C}_{\rm semi}(n/2, p+n+2) + \mathsf{C}(n,q) + \\
&\quad O(\mathsf{M}(n/2,q)) + \mathsf{M}_{\rm semi}(n/2,q) + \\
&\quad \mathsf{M}(n, p+n+2)\log_2 n + \mathsf{M}_{\rm semi}(n/2, p+n+2) + O(n\,(p+n)) \\
&\leqslant \mathsf{C}_{\rm rel}(n/2,p) + O(\mathsf{I}((n+p)^{11/5})\log^{3/2}(n+p))
\end{aligned}$$

It follows that

$$\mathsf{C}_{\rm rel}(n,p) = O(\mathsf{I}((n+p)^{11/5})\log^{5/2}(n+p)).$$

THEOREM 24. *Let $f, g, h \in \mathbb{C}[[z]]$ be power series with $g_0 = 0$, $g_1 \neq 0$ and $p \in \mathbb{N}$. Assume that $\|f\|_1 \leqslant 1$ and $\|g\|_1 \leqslant 1$. The computation of the relaxed composition $h = f \circ g$ at order n and up to an absolute error $\leqslant 2^{-p}$ can be done in time $O(\mathsf{I}((n+p)^{11/5})\log^{5/2}(n+p))$.*

REMARK 25. From [BK75, vdH02], we have

$$\mathsf{C}_{\rm rel}(n,p) = O(\mathsf{I}(n\,(p+n))\,\sqrt{n\log^3 n})) = O(\mathsf{I}((n+p)^{5/2})\log^3(n+p)).$$

Theorem 24 improves on this bound in the frequent case when $n \asymp p$. Unfortunately, we have not yet been able to prove a quasi-linear bound $\mathsf{C}_{\rm rel}(n,p) = \tilde{O}((n+p)^2)$.

REMARK 26. For certain types of functional equations, one may avoid to apply theorem 24. For instance, power series reversion can directly be reduced to composition using Newton's method [BK75]. The solutions of certain other equations, such as

$$f(z) = \mathrm{e}^z + f(z^5\, f(z^7)),$$

can be evaluated efficiently on small disks. Consequently, the Taylor coefficients of f can be computed efficiently using lemma 17.

BIBLIOGRAPHY

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Massachusetts, 1974.

[Ber98] D.J. Bernstein. Composing power series over a finite ring in essentially linear time. *JSC*, 26(3):339–341, 1998.

[BK75] R.P. Brent and H.T. Kung. $O((n \log n)^{3/2})$ algorithms for composition and reversion of power series. In J.F. Traub, editor, *Analytic Computational Complexity*, Pittsburg, 1975. Proc. of a symposium on analytic computational complexity held by Carnegie-Mellon University.

[BK77] R.P. Brent and H.T. Kung. Fast algorithms for composition and reversion of multivariate power series. In *Proc. Conf. Th. Comp. Sc.*, pages 149–158, Waterloo, Ontario, Canada, August 1977. University of Waterloo.

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

[BSS08] Alin Bostan, Bruno Salvy, and Éric Schost. Power series composition and change of basis. In J. Rafael Sendra and Laureano González-Vega, editors, *ISSAC*, pages 269–276, Linz/Hagenberg, Austria, July 2008. ACM.

[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.

[Für07] M. Fürer. Faster integer multiplication. In *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*, pages 57–66, San Diego, California, 2007.

[Kro82] L. Kronecker. Grundzüge einer arithmetischen Theorie der algebraischen Grössen. *Jour. für die reine und ang. Math.*, 92:1–122, 1882.

[Sch82] A. Schönhage. Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients. In J. Calmet, editor, *EUROCAM '82: European Computer Algebra Conference*, volume 144 of *Lect. Notes Comp. Sci.*, pages 3–15, Marseille, France, April 1982. Springer.

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

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

[vdH07] Joris van der Hoeven. New algorithms for relaxed multiplication. *JSC*, 42(8):792–802, 2007.