



Certification of Polynomial Middle Product

Pascal Giorgi

► **To cite this version:**

| Pascal Giorgi. Certification of Polynomial Middle Product. 2017. <lirmm-01538453>

HAL Id: lirmm-01538453

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-01538453>

Submitted on 13 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Certification of Polynomial Middle Product

Pascal Giorgi

Abstract—Polynomial multiplication and its variants are a key ingredient in effective computer algebra. While certifying a polynomial product is a well known task, it was not yet clear how to do a similar approach for its middle product variant. In this short note, we present a new algorithm that provides such a certification with the same complexity and probability that polynomial multiplication certification. Furthermore, we extend our algorithm to certify any operations that compute only a certain chunk of the product.

Index Terms—Theory of computation, computations on polynomials, arithmetic, polynomial multiplication, middle product, probabilistic certification



1 INTRODUCTION

Polynomial multiplication is a fundamental tool in computer algebra as it often plays a central role in most efficient algorithms. Improving its complexity remains a major challenge of the domain. In order to speed-up some algorithms, one might be interested in computing only a certain part of the product. For instance, this is the case with polynomial division or inversion where computing the middle terms of a specific product improves the complexity by a constant factor. This specific operation is called the middle product in [1]. Let $f, g \in \mathbb{K}[X]$ be two polynomials defined over a field \mathbb{K} such that $\deg f = s-1$, $\deg g = 2s-2$. The middle product of degree $s-1$ to $2s-2$ from the product fg denoted $MP_s(f, g)$ corresponds to the coefficients of degree $s-1$ to $2s-2$ from the product fg . Let $fg = \sum_{i=0}^{3s-3} h_i X^i$ then $MP_s(f, g) = h_{s-1} + h_s X + h_{s+1} X^2 + \dots + h_{2s-2} X^{s-1}$. Let $M(n)$ denote the complexity function for the multiplication of two polynomials of $\mathbb{K}[X]$ of degree at most n . Computing $MP_s(f, g)$ through full products requires $2M(s) + O(s)$ operations in \mathbb{K} . As shown in [1], dedicated algorithms can compute $MP_s(f, g)$ twice faster. One remarkable property of middle product is to be the transposed problem of polynomial multiplication using the Tellegen principle [2]. This strong result tells us that any polynomial multiplication algorithm can be turned into an algorithm for middle product with the same asymptotic complexity, i.e. $M(s) + O(s)$. Since the seminal work of Karatsuba [3], many fast polynomial multiplication algorithms have been designed in order to reach a quasi-linear time complexity [4, Chapter 8]. As of today, the best result over finite fields is $O(d \log d \delta^{\log^* d} \log p)$ operations¹ for the product of degree d polynomials [5]. A common feature of all these algorithms is to be much more complex than the naive product, meaning their implementation could be complicated and errors prone. Furthermore, the derivation of algorithms for middle product using Tellegen principle might also lead to some implementation errors.

A classic way to check implementation is to use a *posteriori* certification. The idea is to provide an algorithm

that can check the result with an asymptotically better complexity than the operation itself. The simplicity of the algorithm must ensure the robustness of its implementation. For instance, such certification algorithm can also serve when one want to check a computation delegated to an untrusted cloud.

In order to check a polynomial product fg one can pick a random point α and check that $f(\alpha)g(\alpha) = (fg)(\alpha)$. If not, it is clear that the product is wrong. If the results agree, it is well known through Zippel-Schwartz-Lipton-DeMillo lemma [6], [7], [8] that the product fg is correct with a probability greater than $1 - \frac{d}{N}$ where N corresponds to the number of sampling points for α and $\deg fg < d$. Assuming $N > d$, one can decrease the probability to $1 - \frac{d^k}{N^k}$ by picking k different points. One advantage of this certification is that polynomial evaluation has a linear time complexity and can be implemented easily through Horner's rules.

To the best of our knowledge, the certification of the middle product has not been investigated while it is closely related to polynomial multiplication. In this short note, we demonstrate that one can achieve similar linear time certificate for the middle product. One motivation of this work came from our experiment to compute the kernel of a large sparse matrix arising in discrete logarithm computation. In particular, one part of the computation was relying on polynomial middle product with matrix coefficients [9]. Unfortunately, our code failed to produce correct results when polynomial degrees were above 500 000. Since quadratic time certificate was not an option for such parameters we decided to develop fast approach to certify the middle product.

We start the next section by giving a matrix interpretation to the certification of polynomial product. Using this interpretation, we will define in the following sections our probabilistic certificate for the middle product. Finally, in the last section we show that our method extends to any other operation that computes any partial chunk of polynomial multiplication.

1. \log^* is the iterated logarithm function

4 TOEPLITZ MATRIX-VECTOR PRODUCT WITH POWERS

Let $f \in \mathbb{K}[X]$ of degree $s - 1$, we denote L_f and U_f the following triangular Toeplitz matrices:

$$\underbrace{\begin{pmatrix} f_{s-1} & f_{s-2} & \cdots & f_0 \\ & \ddots & \ddots & \vdots \\ & & \ddots & f_{s-2} \\ & & & f_{s-1} \end{pmatrix}}_{U_f}, \underbrace{\begin{pmatrix} f_0 & & & \\ f_1 & \ddots & & \\ \vdots & \ddots & \ddots & \\ f_{s-1} & \cdots & f_1 & f_0 \end{pmatrix}}_{L_f}$$

where $f = f_0 + f_1X + \cdots + f_{s-1}X^{s-1}$ and $U_f, L_f \in \mathbb{K}^{s \times s}$.

Lemma 4.1: Let $\vec{\alpha}_s = [1, \alpha, \dots, \alpha^{s-1}] \in \mathbb{K}^{1 \times s}$. The matrix-vector products $\vec{\alpha}_s U_f$ and $\vec{\alpha}_s L_f$ can be computed in $O(s)$ operations in \mathbb{K} .

Proof: It obvious that the lemma is correct for $s = 1$. Let us assume the lemma correct for dimension $s-1$ and write $f = f_0 + X\hat{f}$, with $f_0 \in \mathbb{K}$ and $\hat{f} \in \mathbb{K}[X]$ of degree $s-2$. One can rewrite U_f as follow:

$$U_f = \begin{pmatrix} U_{\hat{f}} & f_0 \\ & \vdots \\ & f_{s-1} \end{pmatrix}.$$

There, multiplying a vector $\vec{\alpha}_s$ by U_f is equivalent to compute the row vector $[\vec{\alpha}_{s-1} U_{\hat{f}}, \vec{\alpha}_s \cdot [f_0, \dots, f_{s-1}]]$.

From the Toeplitz structure of U_f it is easy to see that $\vec{\alpha}_s \cdot [f_0, \dots, f_{s-1}]$ is equal to $\alpha y + f_0$ where y is the last column of $\vec{\alpha}_{s-1} U_{\hat{f}}$. By induction, it follows immediately that the complexity is linear in the matrix dimension s . For the matrix L_f the proof is similar remarking that

$$L_f = \begin{pmatrix} f_0 & & & \\ \vdots & & & \\ f_{s-1} & L_{(f \bmod X^{s-1})} & & \end{pmatrix}$$

and that $\vec{\alpha}_s \cdot [f_0, \dots, f_{s-1}] = \alpha^{-1}y + \alpha^{s-1}f_{s-1}$ where $y = \vec{\alpha}_{s-1} L_{(f \bmod X^{s-1})}$. \square

One may remark that computing $\vec{\alpha} U_f$ performs exactly the same operations as calculating $f(\alpha)$ using Horner's rule. The same remark applied for $\vec{\alpha} L_f$ but with the evaluation of $rev(f) = f(1/X)X^s$ in $X = \alpha$.

Corollary 4.2: The transposed operations $U_f \vec{\alpha}^T$ and $L_f \vec{\alpha}^T$ can also be computed in $O(s)$ operations in \mathbb{K} .

Indeed, by transposed matrix product we have $(U_f \vec{\alpha}^T)^T = \vec{\alpha} L_{rev(f)}$ and $(L_f \vec{\alpha}^T)^T = \vec{\alpha} U_{rev(f)}$.

Corollary 4.3: Let T_f be a full Toeplitz matrix, one can compute $T_f \vec{\alpha}^T$ or $\vec{\alpha} T_f$ in $O(s)$ operations rather than $M(s)$ operations with the classical fast approach [11].

Indeed, T_f is a sum of an upper and lower triangular Toeplitz matrix, and Lemma 4.1 can be applied.

5 A LINEAR TIME CERTIFICATION ALGORITHM

Let $f, g, h \in \mathbb{K}[X]$ such that $\deg f = \deg h = s - 1$, $\deg g = 2s - 2$. The following algorithm provides a probabilistic verification for $h = MP_s(f, g)$ that requires a linear number of operation.

Algorithm **CertifiedMP**(f, g, h) :

- 1) choose a random α from a finite subset $S \subset \mathbb{K}$ and set $\vec{\alpha}_s \leftarrow [1, \alpha, \dots, \alpha^{s-1}]$
- 2) $y_1 \leftarrow (\vec{\alpha}_s U_f) \cdot [g_0, \dots, g_{s-1}]$
- 3) $y_2 \leftarrow \alpha (\vec{\alpha}_{s-1} L_{(f \bmod X^{s-1})}) \cdot [g_s, \dots, g_{2s-2}]$
- 4) return true if $h(\alpha) = y_1 + y_2$, false otherwise

Lemma 5.1: Algorithm **CertifiedMP**(f, g, h) ensures that $h = MP_s(f, g)$ with a probability greater or equal to $1 - s/|S|$. The algorithms uses $O(s)$ operations in \mathbb{K} and $\lceil \log_2 |S| \rceil$ random bits.

Proof: The correctness of algorithm **CertifiedMP** comes from the definition of $MP_s(f, g)$ as a linear application when f is fixed. Indeed, this corresponds to a linear application from $\mathbb{K}^{2s-1} \rightarrow \mathbb{K}^s$ where its matrix representation in the canonical basis of $\mathbb{K}[X]$ is:

$$\mathcal{M}_f = \underbrace{\begin{pmatrix} f_{s-1} & f_{s-2} & \cdots & f_0 \\ & \ddots & \ddots & \vdots \\ & & \ddots & f_{s-2} \\ & & & f_{s-1} \end{pmatrix}}_{U_f} \underbrace{\begin{pmatrix} f_0 & & & \\ \vdots & \ddots & & \\ f_{s-2} & \cdots & f_0 & \end{pmatrix}}_{L_{(f \bmod X^{s-1})}}.$$

Let $\vec{h} = \mathcal{M}_f [g_0, g_1, \dots, g_{2s-2}]^T$, one can read the coefficients of $h = MP_s(f, g)$ from \vec{h} . Splitting \mathcal{M}_f and g in two parts, we get

$$\vec{h} = U_f \begin{pmatrix} g_0 \\ \vdots \\ g_{s-1} \end{pmatrix} + \begin{pmatrix} 0 & \cdots & 0 \\ L_{(f \bmod X^{s-1})} \end{pmatrix} \begin{pmatrix} g_s \\ \vdots \\ g_{2s-2} \end{pmatrix}$$

Therefore, multiplying this equation on the left by $\vec{\alpha}_s$ gives $h(\alpha) = y_1 + y_2$ and proves the correctness of our algorithm. Using Lemma 3.1, the probability that $h(\alpha) = y_1 + y_2$ when $l \neq h$ is less than $\frac{s}{|S|}$, which then gives a probability of success greater than $1 - \frac{s}{|S|}$ as promised.

From Lemma 4.1 and the cost of dot product in dimension at most s , one can deduce the complexity of $O(s)$. Since the bitsize of α is less than $\log_2 |S|$, this concludes the proof. \square

Remark 1: Assuming $|S| > 2s$, one can run k times Algorithm **CertifiedMP**(f, g, l) on same inputs to raise the probability to $1 - \frac{1}{2^k}$.

6 A MORE GENERAL RESULT

Following our previous result we are able to generalize our algorithm to certify any operations that compute only a certain consecutive chunk of a polynomial product. This is for instance the case for the so-called short product operation that correspond to the multiplication of power series at a given precision [12], [13].

Let $f, g \in \mathbb{K}[X]$ of degree $s - 1$, the short product of f and g is denoted $SP_s(f, g) = fg \bmod X^s$. Similarly, one can define the high short product of f and g to be $HP_s(f, g) = fg \operatorname{div} X^{s-1}$, corresponding to the s highest term of the product fg . Assuming f is fixed, one can define these two operations as linear applications from $\mathbb{K}^s \rightarrow \mathbb{K}^s$ with the matrix L_f for $SP_s(f, g)$ and the matrix U_f for $HP_s(f, g)$. As before, picking a random element $\alpha \in S \subset \mathbb{K}$, one can check the two short product operations by checking respectively $h(\alpha) = (\vec{\alpha}L_f) \cdot \vec{g}$ or $h(\alpha) = (\vec{\alpha}U_f) \cdot \vec{g}$. Indeed, using Lemma 4.1 one can achieve a complexity of $O(s)$ operations in \mathbb{K} and a probability of success greater than $1 - s/|S|$.

Without loss of generality, assuming that $\deg f = m \geq \deg g = n$ and $s \mid n$. One can define a partial product operation on f and g as $PP_s(f, g, i) = (fg \operatorname{div} X^i) \bmod X^s$. This operation corresponds to extracting the s consecutive terms of the product fg starting from the monomial X^i . Assuming f is fixed, this operation is a linear application from $\mathbb{K}^n \rightarrow \mathbb{K}^s$ where its matrix has the form

$$\mathcal{M}_f = \begin{pmatrix} T_{f_0} & T_{f_1} & \dots & T_{f_{n/s-1}} \end{pmatrix} \in \mathbb{K}^{s \times n}$$

such that each $T_{f_k} \in \mathbb{K}^{s \times s}$ for $k \in [0, \dots, n/s - 1]$ is a Toeplitz matrix formed from the coefficients of the polynomial f . More precisely, we have

$$T_{f_k} = \begin{pmatrix} f_{i-ks} & f_{i-ks-1} & \dots & f_{i-(k+1)s+1} \\ f_{i-ks+1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & f_{i-ks-1} \\ f_{i-(k-1)s-1} & \dots & f_{i-ks+1} & f_{i-ks} \end{pmatrix}$$

with $f_j = 0$ when $j < 0$ or $j > m$ and f_j is the coefficient of the polynomial f at X^j otherwise. Let $h = PP_s(f, g, i)$ and \vec{g}, \vec{h} be the vector of the coefficients of the polynomial g and h . By definition of \mathcal{M}_f we have $\vec{h} = \mathcal{M}_f \vec{g}$. Here again, applying a vector $\vec{\alpha}$ to this equality provides us a way to certify the partial product operation.

The following algorithm provides a probabilistic certificate for $h = PP_s(f, g, i)$ with a complexity of $O(n)$:

Algorithm **CertifiedPP**(f, g, h, s, i) :

- 1) choose a random α from a finite subset $S \subset \mathbb{K}$ and set $\vec{\alpha}_s \leftarrow [1, \alpha, \dots, \alpha^{s-1}]$
- 2) for k from 0 to n/s

$$y_k \leftarrow (\vec{\alpha}_s T_{f_k}) \cdot [g_{ks}, \dots, g_{(k+1)s-1}]$$
- 3) return true if $h(\alpha) = \sum_{k=0}^{n/s-1} y_k \alpha^k$, false otherwise

Lemma 6.1: Algorithm **CertifiedPP**(f, g, h, s, i) ensures that $h = PP_s(f, g, i)$ with a probability greater or equal to $1 - s/|S|$. The algorithm uses $O(n)$ operations in \mathbb{K} and $\lceil \log_2 |S| \rceil$ random bits.

Proof: From the definition of \mathcal{M}_f we know that $\vec{h} = \mathcal{M}_f \vec{g}$ corresponds to the partial product operation $h = PP_s(f, g, i)$. There, multiplying both side of the equation gives $\vec{\alpha} \cdot \vec{h} = h(\alpha) = (\vec{\alpha} \mathcal{M}_f) \cdot \vec{g}$. Since $\sum_{k=0}^{n/s-1} y_k \alpha^k$ corresponds, in our algorithm, exactly to $(\vec{\alpha} \mathcal{M}_f) \cdot \vec{g}$, this proves the correctness of our algorithm.

Assuming $h \neq PP_s(f, g, i)$, the value $h(\alpha) - (\vec{\alpha} \mathcal{M}_f) \cdot \vec{g}$ is a non zero polynomial of $\mathbb{K}[\alpha]$ of degree less than s . Hence, such polynomial can be zero only for s values of $\alpha \in S \subset \mathbb{K}$ which gives the expected probability. Finally, the complexity of our algorithm is dominated by step 2. Each loop costs exactly $O(s)$ operations in \mathbb{K} by using Corollary 4.3. Since the size of the loop is n/s , the final complexity is $O(n)$ as promised. \square

For some specific cases, one is able to reduce the complexity of $PP_s(f, g, i)$. Indeed, depending on the value of i some Toeplitz matrices T_{f_k} will be zero. Using the structure of \mathcal{M}_f , one can prove that the number of non zero matrices is given by $\lceil \frac{i}{s} \rceil$ if $i < n$ and $\lceil \frac{m-i}{s} \rceil$ if $i > m$. For such cases, the complexity drops down to $O(s \lceil \frac{i}{s} \rceil)$ and $O(s \lceil \frac{m-i}{s} \rceil)$ which are below $O(n)$.

REFERENCES

- [1] G. Hanrot, M. Quercia, and P. Zimmermann, "The middle product algorithm i," *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, no. 6, pp. 415–438, 2004.
- [2] A. Bostan, G. Lecerf, and E. Schost, "Tellegen's principle into practice," in *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*. ACM, 2003, pp. 37–44.
- [3] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," *Soviet Physics-Doklady*, vol. 7, pp. 595–596, 1963.
- [4] J. v. z. Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed. New York, NY, USA: Cambridge University Press, 2013.
- [5] D. Harvey, J. V. D. Hoeven, and G. Lecerf, "Faster polynomial multiplication over finite fields," *J. ACM*, vol. 63, no. 6, pp. 52:1–52:23, Jan. 2017.
- [6] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ser. EUROSAM '79. London, UK, UK: Springer-Verlag, 1979, pp. 216–226.
- [7] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, Oct. 1980.
- [8] R. A. Demillo and R. J. Lipton, "A probabilistic remark on algebraic program testing," *Information Processing Letters*, vol. 7, no. 4, pp. 193 – 195, 1978.
- [9] P. Giorgi and R. Lebreton, "Online order basis algorithm and its impact on the block Wiedemann algorithm," in *ISSAC '14: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*. ACM, 2014, pp. 202–209.
- [10] T. Kimbrel and R. K. Sinha, "A probabilistic algorithm for verifying matrix products using $o(n^2)$ time and $\log_2 n + o(1)$ random bits," *Information Processing Letters*, vol. 45, no. 2, pp. 107 – 110, 1993.
- [11] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.
- [12] T. Mulders, "On short multiplications and divisions," *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, no. 1, pp. 69–88, 2000.
- [13] G. Hanrot and P. Zimmermann, "A long note on mulders short product," *Journal of Symbolic Computation*, vol. 37, no. 3, pp. 391 – 401, 2004.