*Article Submitted to Journal of Symbolic Computation*

# FFT-like Multiplication of Linear Differential Operators

Joris Van Der Hoeven[1]

[1] *Dépt. de Mathématique (C.N.R.S.)*
*Université Paris-Sud*
*91364 Orsay Cedex, France*
*e-mail :* `Joris.VANDERHOEVEN@math.u-psud.fr`

**Abstract**

It is well known that integers or polynomials can be multiplied in an asymptotically fast way using the discrete Fourier transform. In this paper, we give an analogue of fast Fourier multiplication in the ring of skew polynomials $C[x, \delta]$, where $\delta = x\frac{\partial}{\partial x}$. More precisely, we show that the multiplication problem of linear differential operators of degree $n$ in $x$ and degree $n$ in $\delta$ can be reduced to the $n \times n$ matrix multiplication problem.

## 1. Introduction

Let $C$ be an effective ring, which means that all ring operations can be performed effectively. It is classical [Cooley and Tukey (1965); Schönhage and Strassen (1971); Knuth (1981)] that polynomials of degree $n$ in $C[x]$ can be multiplied in time $O(n \log n \log \log n)$ using FFT multiplication. If $C$ contains sufficiently many $2^k$-th roots of unity, this complexity further reduces to $O(n \log n)$. Notice that these complexities are measured in terms of operations in $C$.

In this paper, we consider the skew polynomial ring $C[x, \delta]$, where $\delta = x\frac{\partial}{\partial x}$. We assume that $C$ is an effective $\mathbb{Q}$-algebra, so that both the ring operations and the scalar multiplication with rationals can be performed effectively. We show that the multiplication problem of polynomials of degree $n$ in $x$ and degree $n$ in $\delta$ can be reduced to the problem of multiplying (a fixed finite number of) $n \times n$ matrices. Fast algorithms for $n \times n$ matrix multiplication are described in [Strassen (1969); Pan (1984); Coppersmith and Winograd (1990); Knuth (1981)] and the lowest time resp. space complexities, which can currently be achieved by such algorithms, are $O(n^\alpha)$ with $\alpha < 2.376$ and $O(n^2)$.

More precisely, we will prove theorem 1.1 below. This theorem should be compared to the naive algorithm for the multiplication of linear differential operators, which has time complexity $O(n^3 \log n \log \log n)$.

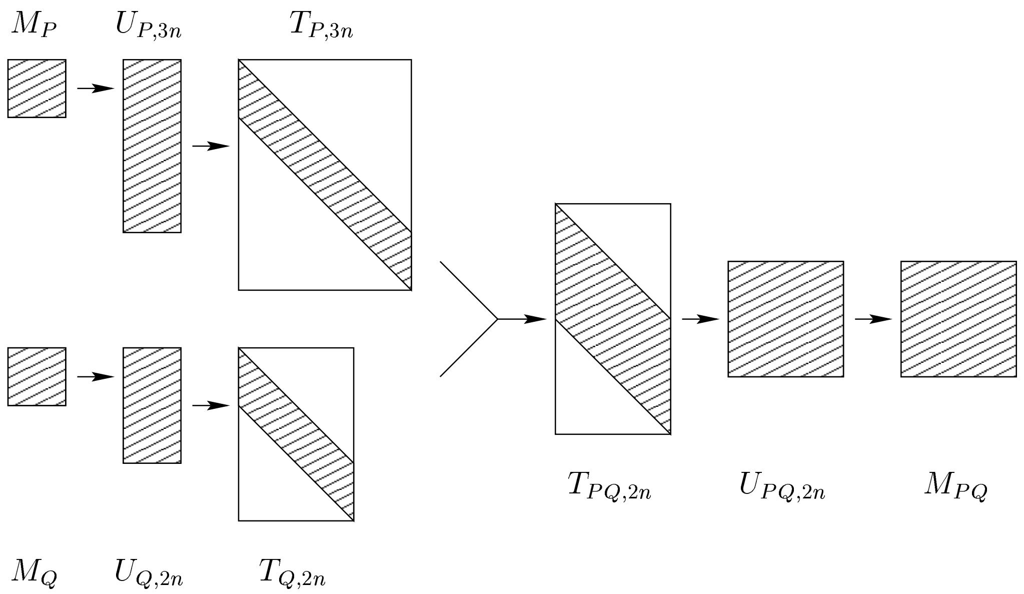


**Figure 1:** Schematic representation of FFT multiplication of linear differential operators.

THEOREM 1.1: *Assume that there exists an algorithm which multiplies two $n \times n$ matrices in time $M(n)$ and space $S(n)$. Then two linear differential operators in $C[x, \delta]$ of degree $n$ in $x$ and degree $\delta$ can be multiplied in time $O(M(n))$ and space $O(S(n))$.*

Classically, FFT multiplication proceeds by evaluating the multiplicands in $2^k$-th roots of unity, multiplying these evaluations, and interpolating the results. In the non commutative case, the idea is to evaluate the linear differential operators at powers of $x$. Roughly speaking, this comes down to interpreting linear differential operators of degree $n$ in $x$ and degree $n$ in $\delta$ as linear mappings from $C \oplus \cdots \oplus C x^n$ into $C \oplus \cdots \oplus C x^{2n}$. In this context, the direct and inverse Fourier transforms correspond to multiplications with a Vandermonde matrix or its inverse, as well as some additional reordering of coefficients.

For the reader's convenience, we have illustrated our algorithm in figure 1. The significance of this figure will become clear when reading sections 2, 3 and 4.

## 2. The direct Fourier transform

Consider a linear differential operator

$$P = \sum_{i=0}^{n} \sum_{j=0}^{n} P_{i,j} x^j \delta^i. \tag{1}$$

We associate the following matrix with $P$:

$$M_P = \begin{pmatrix} P_{0,0} & \cdots & P_{0,n} \\ \vdots & & \vdots \\ P_{n,0} & \cdots & P_{n,n} \end{pmatrix}. \tag{2}$$

Let $V_{m,n}$ denote the Vandermonde matrix

$$V_{m,n} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & k & \cdots & k^n \\ \vdots & \vdots & & \vdots \\ 1 & m & \cdots & m^n \end{pmatrix}. \tag{3}$$

Since $\delta^l(x^k) = k^l x^k$, the coefficient $(U_{P,m})_{i,j}$ of the matrix $U_{P,m} = V_{m,n} M_P$ coincides with the coefficient of $x^i$ in the evaluation of $P_{0,j} + \cdots + P_{n,j}\delta^n$ at $x^i$. Now we define the "Fourier transform" of $P$ at order $m$ by

$$T_{P,m} = \begin{pmatrix} (U_{P,m})_{0,0} & & & 0 \\ \vdots & (U_{P,m})_{1,0} & & \\ (U_{P,m})_{0,n} & \vdots & \ddots & \\ & (U_{P,m})_{1,n} & & (U_{P,m})_{m,0} \\ & & \ddots & \vdots \\ 0 & & & (U_{P,m})_{m,n} \end{pmatrix} \tag{4}$$

This matrix has the following property: given a polynomial $A = A_0 + \cdots + A_m x^m$, represented by the column matrix $M_A$ with entries $A_0, \ldots, A_m$, the evaluation $P(A)$ of $P$ at $A$ is represented by the column vector $M_{P(A)} = T_{P,m} M_A$.

## 3. The inner multiplication

Now consider two differential operators, $P$ given by (2) and

$$Q = \sum_{i=0}^{n} \sum_{j=0}^{n} Q_{i,j} x^j \delta^i. \tag{5}$$

In order to multiply $P$ and $Q$, we compute the Fourier transforms of $P$ at order $3n$ and of $Q$ at order $2n$. This yields two matrices $T_{P,3n}$ and $T_{Q,2n}$. We claim that their product $T_{P,3n} T_{Q,2n}$ coincides with the Fourier transform $T_{PQ,2n}$ of $PQ$ at order $2n$. Indeed, for each polynomial $A$ of degree $2n$, we have $T_{PQ,2n} M_A = M_{PQ(A)} = M_{P(Q(A))} = T_{P,3n} M_{Q(A)} = T_{P,3n} T_{Q,2n} M_A$. Our claim follows by counting dimensions. It remains to be shown how to retrieve $PQ$ from $T_{PQ,2n}$.

## 4. The inverse Fourier transform

The matrix $U_{PQ,2n}$ is easily obtained as a function of $T_{PQ,2n}$ using the formula $(U_{PQ,2n})_{i,j} = (T_{PQ,2n})_{j,i+j}$. We finally compute the coefficients of $PQ$ using the formula

$$M_{PQ} = V_{2n,2n}^{-1} U_{PQ,2n}. \tag{6}$$

Let us show how to invert the Vandermonde matrix $V_{2n,2n}$ quickly. One formally verifies that the inverse of a general Vandermonde matrix

$$V = \begin{pmatrix} 1 & \lambda_0 & \cdots & \lambda_0^n \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_n & \cdots & \lambda_n^n \end{pmatrix}, \tag{7}$$

is given by the formula

$$V^{-1} = \begin{pmatrix} (-1)^n \frac{\Sigma_{n;0}}{D_0} & \cdots & (-1)^n \frac{\Sigma_{n;n}}{D_n} \\ \vdots & & \vdots \\ \frac{\Sigma_{0;0}}{D_0} & \cdots & \frac{\Sigma_{0;n}}{D_n} \end{pmatrix}, \tag{8}$$

where

$$\begin{aligned} \Sigma_{d;i} &= \Sigma_d(\lambda_0, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_n); \\ D_i &= (\lambda_i - \lambda_0) \cdots (\lambda_i - \lambda_{i-1})(\lambda_i - \lambda_{i+1}) \cdots (\lambda_i - \lambda_n) \end{aligned}$$

and $\Sigma_d$ is the symmetric polynomial of degree $d$

$$\Sigma_d(\alpha_1, \ldots, \alpha_n) = \sum_{1 \leqslant i_1 < \cdots < i_d \leqslant n} \prod_{j=1}^{d} \alpha_{i_j}. \tag{9}$$

The $\Sigma_{d;i}$ may now be computed recursively in time and space $O(n^2)$ using the formulas

$$\Sigma_{0;i} = 1; \tag{10}$$

$$\Sigma_{d;i} = \Sigma_d(\lambda_0, \ldots, \lambda_n) - \lambda_i \Sigma_{d-1;i}. \tag{11}$$

The $D_i$ may be computed naively in time and space $O(n^2)$. Using $O(n^2)$ final divisions in $\mathbb{Q}$, this proves that $V_{2n,2n}^{-1}$ can be computed in time and space $O(n^2)$.

## 5. Multiplication in other skew polynomial rings

The FFT multiplication from the previous sections may be adapted or generalized to a certain number of other skew polynomial rings. We will briefly describe two other examples.

### 5.1. Differentiation with respect to $x$

Consider the multiplication problem in the ring $C[x, \partial]$, where $\partial = \frac{\partial}{\partial x}$ satisfies $\partial x = x\partial + 1$. The fact that $\partial$ (unlike $\delta$) is not graduated in $x$ as an operator on $C[x]$ implies that the algorithm from the previous sections can not be generalized in a direct way. Nevertheless, elements of $C[x, x^{-1}, \partial]$ may be rewritten as elements of $C[x, x^{-1}, \delta]$ and vice versa in an efficient manner. Moreover our algorithm for FFT multiplication in $C[x, \delta]$ is easily adapted to the ring $C[x, x^{-1}, \delta]$. Indirectly, we thus obtain an FFT-like algorithm for the multiplication of linear differential operators in $C[x, \partial]$.

The fast conversion of elements in $C[x, x^{-1}, \partial]$ to elements in $C[x, x^{-1}, \delta]$ and vice versa is based on the fact that we have formulas

$$\partial^i = x^{-i} \sum_{j=0}^{i} \alpha_{i,j} \delta^j; \qquad (12)$$

$$\delta^i = \sum_{j=0}^{i} \beta_{i,j} x^j \partial^j, \qquad (13)$$

with $\alpha_{i,j}, \beta_{i,j} \in \mathbb{Z}$. The coefficients $\alpha_{i,j}$ and $\beta_{i,j}$ can be computed by induction in linear time. Using (12) and (13) in a convenient way, the conversion problem then again reduces to matrix multiplication.

### 5.2. The shift operator

Consider the ring $C[x, S]$ with $Sx = (x+1)S$. This ring corresponds to the ring of polynomials in $x$ with the shift operator $S : P(x) \mapsto P(x+1)$. The ideas from sections 2, 3 and 4 apply to this case as well, but in an "adjoint" way.

Indeed, instead of evaluating the skew polynomials at powers of $x$, we rather consider their *adjoint evaluations* at powers of $S$. Just like the evaluation of a polynomial $P$ in $x$ and $\delta$ at $x^i$ is obtained by setting $\delta = 0$ in the product $Px^i$, the adjoint evaluation of a polynomial $P$ in $x$ and $S$ at $S^i$ is obtained by setting $x = 0$ in the product $S^i P$. Similarly, we may define the adjoint evaluations of skew polynomials in $C[x, S]$ at polynomials in $S$.

We now observe that the adjoint evaluation of $x^i S^j$ at $S^k$ is $k^i S^{j+k}$, just like the evaluation of $x^j \delta^i$ at $x^k$ is $k^i x^{j+k}$. This analogy may be exploited in a straightforward way so as to adapt the FFT multiplication algorithm from sections 2, 3 and 4 to the present case.

## 6. Final remarks

1. Our multiplication method can be adapted to a few other skew polynomial rings, such as $C[x, Q]$ with $Qx = qxQ$ and $q \in C^*$.
2. It would be interesting to see whether the ideas of this paper can also be used in order to prove that the matrix multiplication problem may be

reduced to the multiplication problem of skew polynomials, since this would prove that these problems are essentially equivalent. This is certainly the case, if we assume that the multiplications of constants in $C$ with rational numbers may be neglected with respect to other constant multiplications. Indeed, this assumption implies that the cost of the direct and inverse FFT transforms can be neglected w.r.t. the cost of the inner multiplication, since the direct and inverse FFT transforms only involve multiplications of constants in $C$ with rational numbers.

3. Assume that the skew polynomials $P$ and $Q$ have unequal degrees $n$ and $m$ in $x$ resp. $\delta$. If $n$ and $m$ are very different, then $PQ$ may be computed using the naive formula

$$PQ = P * Q + \frac{\partial P}{\partial \delta} * x \frac{\partial Q}{\partial x} + \cdots + \frac{1}{r!} \frac{\partial^r P}{\partial \delta^r} * \left( x \frac{\partial}{\partial x} \right)^r Q, \tag{14}$$

where $r = \min(n, m)$ and

$$P * Q = \sum_{i_1=0}^{n} \sum_{i_2=0}^{n} \sum_{j_1=0}^{m} \sum_{j_2=0}^{m} P_{i_1,j_1} Q_{i_2,j_2} x^{j_1+j_2} \delta^{i_1+i_2} \tag{15}$$

stands for the “commutative product” of $P$ and $Q$.

If $m = kn$ or $n = km$, with small $k \geqslant 1$, then the technique from the previous section may be adapted in order to yield a multiplication algorithm of complexity $O(k^2 M(n))$ resp. $O(k^2 M(m))$, where $M(n)$ is the time needed to multiply two $n \times n$ matrices. Currently, we do not know, if this bound can be further reduced to $O(k \log k \log\log k M(n))$ resp. $O(k \log k \log\log k M(m))$.

## References

Cooley, J., Tukey, J. (1965). An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, **19**:297–301.

Coppersmith, D., Winograd, S. (1990). Matrix multiplication via arithmetic progressions. *J.S.C.*, **9**:251–280.

Knuth, D. (1981). *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 2-nd edition.

Pan, V. (1984). *How to multiply matrices faster*, volume 179 of *Lect. Notes in Math.*. Springer.

Schönhage, A., Strassen, V. (1971). Schnelle Multiplikation grosser Zahlen. *Computing 7*, **7**:281–292.

Strassen, V. (1969). Gaussian elimination is not optimal. *Numer. Math.*, **13**:352–356.