FACTA UNIVERSITATIS (NIŠ)
SERIES: ELECTRONICS AND ENERGETICS VOL. 10, No.2 (1997), 205-213

RELATIONS BETWEEN MATRIX MULTIPLICATION, CONVOLUTION AND LARGE–NUMBERS MULTIPLICATION

Zdenka Babić

Abstract. This paper presents a better way of reformulating matrix multiplication as polynomial multiplication and convolution. If the elements of the product matrix are known to be bounded, matrix multiplication and convolution can be done, by using a scaling factor, with a single large–number multiplication. The degrees of polynomials and size of numbers being multiplied are smaller than earlier. Also, adaptation a size of scaling factor and multiplier's word–length with matrix dimensions, makes a number of multiplication smaller, because we know beforehand that some products will be zero. With increasing multiplier's word–length decreases number of multiplication, so we can easily adapt this algorithm to various architectures. The algorithm is especially suitable for implementing with parallel multipliers.

1. Introduction

The problems of determining the minimum number of multiplication needed to multiply matrices and the design of fast algorithms for multiplying matrices have both been active research area. Some algorithms with reduced number of multiplication, as APA and algorithm presented in [1], use scaled or sum of scaled matrix elements. In [1] the product of two $N \times N$ matrices is reformulated as the linear convolution of a sequence of length N^2 and a sparse sequence of length $N^3 - N + 1$. This results in a sequence of length $N^3 + N^2 - N$, from which elements of the product matrix can be obtained.

Theorem 1 [1]:

Let $\underline{A}, \underline{B}$ and $\underline{C} = \underline{AB}$ all be $N \times N$ matrices with elements $\{a_{i,j}\}$, $\{b_{i,j}\}$ and $\{c_{i,j}\}$, respectively. Define arrays $\underline{a} = \{a_i\}$, $\underline{b} = \{b_i\}$, $\underline{c} = \{c_i\}$, where:

$$a_{i,j} = a_{i+jN}; \qquad b_{i,j} = b_{N-1-i+jN}; \qquad 0 \le i,\ j \le N-1. \tag{1}$$

Manuscript received February 27, 1997.
Manuscript revised July 22, 1997.
The author is with Faculty of Electrical Engineering, Patre 5, Banjaluka, Republika Srpska.

The matrix multiplication $\underline{C} = \underline{AB}$ is computed by the polynomial multiplication

$$\left(\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{i+jN}x^{i+jN}\right)\left(\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} b_{N-1-i+jN}x^{N(N-1-i+jN)}\right) = \sum_{i=0}^{N^3+N^2-N-1} c_i x^i \tag{2}$$

where the elements of $\underline{C}$ are read off of the $\{c_i\}$ computed in (2), using

$$c_{i,j} = c_{N^2-N+i+jN^2}; \qquad 0 \le i;\ j \le N-1. \tag{3}$$

Let us consider 2×2 matrices for clarity; extensions to larger matrices will be clear. The 2×2 multiplication

$$\begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{bmatrix}\begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} \\ c_{1,0} & c_{1,1} \end{bmatrix} \tag{4}$$

is implemented by the polynomial multiplication (equivalent to a convolution)

$$\begin{aligned}&(a_{0,0} + a_{1,0}x + a_{0,1}x^2 + a_{1,1}x^3) \times (b_{1,0} + b_{0,0}x^2 + b_{1,1}x^4 + b_{0,1}x^6)\\ &= * + *x + c_{0,0}x^2 + c_{1,0}x^3 + *x^4 + *x^5 + c_{1,1}x^7 + *x^8 + *x^9.\end{aligned} \tag{5}$$

Replacing x with sz, where s is scaling factor, and taking the result $\bmod(z^6 - 1)$ produce:

$$\begin{aligned}&(a_{0,0} + a_{1,0}sz + a_{0,1}s^2z^2 + a_{1,1}s^3z^3) \times ((b_{1,0} + b_{0,1}s^6) + b_{0,0}s^2z^2 + b_{1,1}s^4z^4)\\ &= (* + c_{0,1}s^6) + (*s + c_{1,1}s^7)z + (c_{0,0}s^2 + *s^8)z^2\\ &+ (c_{1,0}s^3 + *s^9)z^3 + *z^4 + *z^5; \qquad \bmod(z^6 - 1)\end{aligned} \tag{6}$$

If we choose scaling factor s that satisfies condition $\mid c_{i,j} \mid < s^6$ and $\mid * \mid < s^6$, then the $c_{i,j}$ and $*$ in (6) may be separated from each other without error. The equitation (6) is equivalent to 6–point circular convolution, and can be calculated using pseudo–number–theoretical transforms using only 6 multiplications. Number of multiplication decreases with increasing scaling factor and, extremely, with $\mid c_{i,j} \mid < s$ and $\mid * \mid < s$, we have circular convolution in one point. Take $z = 1$ and matrix multiplication can be done with single multiplication:

$$\begin{aligned}&(a_{0,0} + a_{1,0}s + a_{0,1}s^2 + a_{1,1}s^3) \times (b_{1,0} + b_{0,0}s^2 + b_{1,1}s^4 + b_{0,1}s^6)\\ &= * + *s + c_{0,0}s^2 + c_{1,0}s^3 + *s^4 + *s^5 + c_{0,1}s^6 + c_{1,1}s^7 + *s^8 + *s^9.\end{aligned} \tag{7}$$

The 3×3 matrix multiplication

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} \\ c_{1,0} & c_{1,1} & c_{1,2} \\ c_{2,0} & c_{2,1} & c_{2,2} \end{bmatrix} \tag{8}$$

is implemented by a single multiplication with $\mid c_{i,j} \mid < s$ and $\mid * \mid < s$ as follows:

$$\begin{aligned} &(a_{0,0} + a_{1,0}s + a_{2,0}s^2 + a_{0,1}s^3 + a_{1,1}s^4 + a_{2,1}s^5 + a_{0,2}s^6 + a_{1,2}s^7 + a_{2,2}s^8) \\ &\times (b_{2,0} + b_{1,0}s^3 + b_{0,0}s^6 + b_{2,1}s^9 + b_{1,1}s^{12} + b_{0,1}s^{15} + b_{2,2}s^{18} + b_{1,2}s^{21} + b_{0,2}s^{24}) \\ &= \ldots + c_{0,0}s^6 + c_{1,0}s^7 + c_{2,0}s^8 + \ldots + c_{0,1}s^{15} + c_{1,1}s^{16} + c_{2,1}s^{17} + \ldots \\ &+ c_{0,2}s^{24} + c_{1,2}s^{25} + c_{2,2}s^{26} + \ldots \end{aligned} \tag{9}$$

The numbers being multiplied are massive, and dependent on the degree of scaling factor s, that is maximally $N^3 - N$ for $N \times N$ multiplication. This paper shows that the largest degree can be smaller and equal to $N^3 - N^2 + N - 1$, so that the numbers being multiplied decrease. For 2×2 matrix multiplication the degree of s is five instead six, for 3×3 matrix multiplication twenty instead twenty–four, and so on.

2. A better way of reformulating of matrix multiplication as convolution

If we use another way to form polynomial coefficients then in [1], the degree of scaling factor decreases. The number being multiplied decrease also, so that the number of multiplication can be smaller.

Theorem 2:

Let $\underline{A}, \underline{B}$ and $\underline{C} = \underline{AB}$ all be $N \times N$ matrices with elements $\{a_{i,j}\}$, $\{b_{i,j}\}$ and $\{c_{i,j}\}$, respectively. Define sequences $\underline{a} = \{a_i\}, \underline{b} = \{b_i\}, \underline{c} = \{c_i\}$, where:

$$a_{i,j} = a_{Ni+j}; \qquad b_{i,j} = b_{N-1-i+jN}; \qquad 0 \le i,\ j \le N-1. \tag{10}$$

The matrix multiplication $\underline{C} = \underline{AB}$ is computed by the polynomial multiplication

$$\left(\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_{Ni+j} x^{Ni+j} \right) \left(\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} b_{N-1-i+jN} x^{N^2 j+N-1-i} \right) = \sum_{i=0}^{N^3+N-2} c_i x^i \tag{11}$$

where the elements of $\underline{C}$ are read off of the $\{c_i\}$ computed in (11), using

$$c_{i,j} = c_{N^2 j+Ni+N-1}; \qquad 0 \le i;\ j \le N-1. \tag{12}$$

Proof:

The coefficient of $x^{N^2 j_3 + N i_3 + N - 1}$ in (11) is the sum of the products of the coefficients of $x^{N i_1 + j_1}$ in the first polynomial and the coefficients of $x^{N^2 j_2 + N - 1 - i_2}$ in the second polynomial such that

$$\begin{aligned} &(N i_1 + j_1) + (N^2 j_2 + N - 1 - i_2) = N^2 j_3 + N i_3 + N - 1; \\ &0 \le i_k,\ j_k \le N - 1;\ 1 \le k \le 3. \end{aligned} \tag{13}$$

The solution to (13) is readily seen to be

$$i_1 = i_3,\ j_2 = j_3; \qquad 0 \le j_1 = i_2 \le N - 1. \tag{14}$$

Comparing (10), (12) and (14) shows that (11) implements $\underline{C} = \underline{AB}$.

The degrees of the polynomials multiplied in (11) are $N^3 - N$ and $N^3 - N^2 + N - 1$, and the degree of the resulting polynomial is $N^3 + N - 2$.

The 2×2 matrix multiplication is implemented as:

$$\begin{aligned} &(a_{0,0} + a_{0,1} s + a_{1,0} s^2 + a_{1,1} s^3) \times (b_{1,0} + b_{0,0} s + b_{1,1} s^4 + b_{0,1} s^5) \\ &* + c_{0,0} s + * s^2 + c_{1,0} s^3 + * s^4 + c_{0,1} s^5 + * s^6 + c_{1,1} s^7 + * s^8, \end{aligned} \tag{15}$$

and the 3×3 matrix multiplication as:

$$\begin{aligned} &(a_{0,0} + a_{0,1} s + a_{0,2} s^2 + a_{1,0} s^3 + a_{1,1} s^4 + a_{1,2} s^5 + a_{2,0} s^6 + a_{2,1} s^7 + a_{2,2} s^8) \\ &\times (b_{2,0} + b_{1,0} s + b_{0,0} s^2 + b_{2,1} s^9 + b_{1,1} s^{10} + b_{0,1} s^{11} + b_{2,2} s^{18} + b_{1,2} s^{19} + b_{0,2} s^{20}) \\ &= \ldots + c_{0,0} s^2 + c_{1,0} s^5 + c_{2,0} s^8 + \ldots c_{0,1} s^{11} + c_{1,1} s^{14} + c_{2,1} s^1 7 + \ldots \\ &+ c_{0,2} s^{20} + c_{1,2} s^{23} + c_{2,2} s^{26} + \ldots \end{aligned} \tag{16}$$

It is properly to choose scaling factor in a form $s = 2^p$, so that we have only shifting instead multiplying by s. For example, if the elements of the product matrix are known to be bounded so that we can choose $p = 6$, we can realize the 2×2 matrix multiplication as a single multiplication of numbers 36 bits long. The algorithm presented in [1] uses 42–bit numbers for this operation. And, if $N = 3$, 126 bits are needed, instead 162.

3. Matrix multiplication using number theoretic transform (NTT)

We have seen that matrix multiplication can be reformulated as the circular convolution of arrays which elements are matrix elements or scaled

matrix elements. The length of arrays depends on scaling factor. The convolution computation must be exact, because any round–off error makes the algorithm wrong. Therefore, NTT (especially Fermat transform) is suitable to use for the convolution computation.

For the N–point cyclic convolution, N multiplications are needed. But, there are strong relations between module $M = 2^{2^n} + 1$ and length of transformation $N = 2^{n+1}$, that limit selection of scaling factor and use for matrix multiplication.

The idea behind pseudo–number–theoretic transforms, presented in [1], is to replace the module $2^n \pm 1$ with a large prime divisor of $2^n \pm 1$. For example, we might use to prime $5419 = (2^{21} + 1)/(9)(43)$ as a module. Note that reduction mod(5419) could easily be carried out by first reducing mod$(2^{21} + 1)$, and then reducing only the reminder mod(5419) as desired. The major advantage of pseudo–number–theoretic transform is the greater choice of transform lengths. For 5419, we have transform of length 2,3,6,7,14 and 21 using 2 as a base, since 2 is clearly a 42-nd root of unity mod(5419).

In purpose to comparing, we repeat numerical example explained in [1], for 2×2 matrix multiplication using NTT with only six multiplication.

Numerical example 1:

Consider a 2×2 matrix multiplication:

$$\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 9 & 8 \\ 7 & 6 \end{bmatrix} = \begin{bmatrix} 46 & 40 \\ 62 & 54 \end{bmatrix} \tag{17}$$

$| c_{i,j} | < 64$ and $| * | < 64$, so we can use $s = 2$ and module 5419. Reformulating matrix multiplication as polynomial multiplication we have:

$$\begin{aligned} &(2 + 3x + 4x^2 + 5x^3) \times (7 + 9x^2 + 6x^4 + 8x^6) \\ &= 14 + 21x + \underline{46}x^2 + \underline{62}x^3 + 48x^4 + 63x^5 + \underline{40}x^6 + \underline{54}x^7 + 32x^8 + 40x^9 \end{aligned} \tag{18}$$

where $c_{i,j}$ are underlined. Scaling by $s = 2$, and taking the result mod$(z^6 - 1)$ and mod(5419), the polynomial (6) for this problem is:

$$\begin{aligned} &(2 + 6z + 16z^2 + 40z^3) \times (519 + 36z^2 + 96z^4) \\ &= 2574 + 1535z + 2957z^2 + 4719z^3 + 768z^4 + 2016z^5; \\ &\quad \mathrm{mod}(z^6 - 1); \quad \mathrm{mod}(5419). \end{aligned} \tag{19}$$

After the 6–point NTT, the products are:

$$
\begin{aligned}
&k=0\,[2+6+16+40][519+36+96]=[64][651]\cong 3731\\
&k=1\,\left[2+6(128)+16(128)^2-40\right]\left[519+36(128)^2-96(128)\right]=[2767][3641]\cong 4197\\
&k=2\,\left[2+6(128)^2-16(128)+40\right]\left[519-36(128)+96(128)^2\right]=[4175][2684]\cong 4627\\
&k=3\,[2-6+16-40][519+36+96]=[-28][651]\cong 3448\\
&k=4\,\left[2-6(128)+16(128)^2+40\right]\left[519+36(128)^2-96(128)\right]=[1306][3641]\cong 2683\\
&k=5\,\left[2-6(128)^2-16(128)-40\right]\left[519-36(128)+96(128)^2\right]=[2571][2684]\cong 2177
\end{aligned}
\tag{20}
$$

The separation of the elements $c_{i,j}$ of the product of matrix is as follows:

$$
\begin{aligned}
14+2^6\underline{40}\cong 2574; \quad 2^1(21)+2^7\underline{54}\cong 1535; \quad &\mathrm{mod}(5419)\\
2^2\underline{46}+2^8(32)\cong 2957; \quad 2^3\underline{62}+2^9(40)\cong 4719; \quad &\mathrm{mod}(5419)
\end{aligned}
\tag{21}
$$

where the $c_{i,j}$ are again underlined.

4. Polynomial multiplication as a large–numbers multiplication

We have seen that the matrix multiplication can be reformulated as polynomial multiplication or convolution, and than implemented using only one large–numbers multiplication if the scaling factor s ensured separability of everything in the product. Clearly, we can multiply large numbers with parallel multipliers with arbitrary word–length. Some modules have advantages for certain matrix dimensions.

Numerical example 2:

Let us illustrate this algorithm with same 2×2 matrix multiplication as in Example 1:

$$
\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}\begin{bmatrix} 9 & 8 \\ 7 & 6 \end{bmatrix}=\begin{bmatrix} 46 & 40 \\ 62 & 54 \end{bmatrix} \tag{22}
$$

For clarity, choose we first $s=100$. Based on theorem 2:

$$
(5040302)(8000600090007)=4032\underline{54}\ \underline{40}6348\underline{62}\ \underline{46}2114 \tag{23}
$$

where $c_{i,j}$ are underlined. The choice $s=64$ instead of $s=100$ produces similar result in binary notation, and the result is smaller than that of (23) by a factor of $(100/64)^{10}=86.7$.

Since $c_i<64$ we can choose scaling factor $s=2^6$. Then, form the arrays as follow from (10):

$$
a(n)=\{2,4,3,5\}\ \text{and}\ b(n)=\{7,9,0,0,6,8\}. \tag{24}
$$

Based on (15) we need to multiply two large numbers, in binary notation:

$$\begin{aligned} A &= 000101000011000100000010, \\ B &= 00100000011000000000000001001000111. \end{aligned} \tag{25}$$

Results of each multiplication's must be exact, because any error may cause unpredictable error in matrix product. All bits in massive number multiplication are valid. Rounding is not permissible. We need to take care about this in implementation. If floating–point multipliers are used, it is not possible to take advantages of full capacity of multipliers. Floating–point multipliers are usually more expensive. The increased cost results from the more complex circuitry required within the floating–point processor, which implies a larger silicon die. Integer multipliers are faster and realized with less hardware then floating–point multipliers, therefore they are recommended for exact massive number multiplication.

Make a preposition that we have 12–bit multipliers. A segmentation by 12 bits allows us to write A and B as follows:

$$\begin{aligned} A &= a_1 2^{12} + a_0, \qquad a_1 = 323, \qquad a_0 = 102; \\ B &= b_2 2^{24} + b_1 2^{12} + b_0, \qquad b_2 = 518, \qquad b_1 = 0, \qquad b_0 = 583 \end{aligned} \tag{26}$$

where a_1, a_0 and b_2, b_1, b_0 are 12–bit long numbers.

Multiplying $A \times B$ gives:

$$\begin{aligned} &(a_1 2^{12} + a_0)(b_2 2^{24} + b_1 2^{12} + b_0) \\ &= a_1 b_2 2^{36} + (a_0 b_2 + a_1 b_1) 2^{24} + (a_1 b_0 + a_0 b_1) 2^{12} + a_0 b_0 \end{aligned} \tag{27}$$

Therefore, we need six 12–bit multiplications. The number of multiplication can be smaller if we choose the module on the appropriate way. It is easy to see that b_1 is independent on matrix elements values, and is always zero. Since, we need only four 12–bit matrix multiplications.

For our example:

$$C = A \times B = 167314 \cdot 2^{36} + 133644 \cdot 2^{24} + 188309 \cdot 2^{12} + 150414. \tag{28}$$

For clarity, write C in binary notation:

$$C = 101000 \| 110110 \| 110010 \| 101000 \| 111001 \| 111110 \| 011001 \| 101110 \| 001110 \tag{29}$$

A segmentation by 6 bits produces array:

$$c(n) = \{22, 46, 25, 62, 57, 40, 50, 54, 40\}. \tag{30}$$

Based on (12) we can read off elements of matrix $\underline{C}$:

$$\begin{aligned} c_{0,0} &= c(1) = 46 \qquad c_{0,1} = c(5) = 40 \\ c_{1,0} &= c(3) = 62 \qquad c_{1,1} = c(7) = 54 \end{aligned} \tag{31}$$

Comparing with the example 1, we can see that the direct and inverse NTT is avoided, the word–length is not increased, but number of multiplication is 4 instead 6. Generally, it is possibly implement the 2×2 matrix multiplication with four multiplications if the word–length of multipliers is two times greater then the number of bits needed to present elements of matrix products.

The Strassen algorithm [3] for 2×2 matrix multiplication uses 7 multiplications and 18 additions instead then obvious 8 multiplications and four additions. If processors with implemented hardware multipliers are used, number of multiplications is not more critical factor, because time consumption for multiplication is the same as for addition. In this case Strasssen algorithm has not advantages.

Algorithm presented in this paper needs only four multiplications and no addition. The savings in number of operations is made with detriment in multipliers word–length. In the real signal processing samples are of 8, 12 or 16 bit accuracy. Improvement in digital hardware is much faster then in A/D conversion, so we have now RISC [4], VPP ULSI [5] and signal processors [6],[7] with 32×32 bit multipliers. There is a redundancy in computation precision, because finally precision is limited by A/D conversion. In such cases, when accuracy of samples is bounded to 8 bits, and speed is a critical factor, 32×32 multipliers can be used for 2×2 matrix multiplication with only four multiplications.

5. Conclusion

The algorithm presented in this paper gives a better way of linking matrix multiplication, convolution, and large–number multiplication. Matrix multiplication is reformulated as a linear convolution and polynomial multiplication, and, finally, by using a scaling factor, as a single large–number multiplication. The polynomial degrees and size of numbers are smaller than used in [1]. Also, adaptation a size of scaling factor and multipliers word–length with matrix dimensions, makes a number of multiplication smaller, because we know beforehand that some products will be zero. For example, for 2×2 matrix multiplication, number of multiplication is 4, instead than obvious 8, 7 in Strassen's algorithm, and 6 in [1]. The flexibility of this

algorithm allows trade– off between word–lengths and number of multiplication, that is important for adapting this algorithm to various computer architectures, especially for parallel processors.

REFERENCES

1. A.E. Yagle: *Fast algorithms for matrix multiplication using pseudo–number–theoretic transforms.*IEEE Transactions on Signal Processing, Vol.43, No.1, January 1995.
2. D.E. Knuth: *The Art of Computer Programming: Semi Numerical Algorithms.* Reading MA: Addison-Weslay, 1981.
3. R.E. Blahut: *Fast Algorithms for Digital Signal Processing.* Addison–Weslay Publishing Company, 1985.
4. F. Murabayashi, T. Hotta, S. Tanaka, T. Yamauchi, H. Yamada, T. Nakano, Y. Kobayashi, and T. Bandoh: *BiCMOS Circuit Techniques for a 120–MHz RISC Microprocrssor.* IEEE Jornal on Solid State Circuits, Vol.29, No. 3, March 1994.
5. F. Okamoto, Y. Hagihara, C. Ohkubo, N. Nishi, H. Yamada, and T. Enomoto: *A 200–MFLOPS 100–MHz 64-b BiCMOS Vector–Pipelined Processor (VPP) ULSI.* IEEE Jornal on Solid State Circuits, Vol.26, No. 12, December 1991.