高速剰余変換による多数桁乗算

後保 範[†]

多数桁乗算に関する計算アルゴリズムを提案する．そのアルゴリズムは高速フーリエ変換（FFT）に剰余理論を取り込んだ方法を基礎にした．基礎とする方法を高速剰余変換（Fast Modulo Transformation，FMT）と名付ける．FMT は 1 の原始 n 乗根 ω の上に作成したもので，複素数だけでなく整数でも定義できる．ω を複素数にすると FMT は FFT と同じ計算式になる．FMT は剰余理論に基づいているため多数桁乗算への応用に対する見通しが良い．多数桁乗算への FMT の応用として整数 FMT，巡回乗算，2 段階 FMT，複素 FMT の直接利用および分割乗算を示す．整数 FMT による巡回乗算は ± 1 の巡回だけでなく自然数 α に対して $\pm\alpha$ で巡回する乗算も可能と判明した．さらに，多数桁乗算の入力値を m 個に分割し，互いに異なる m 個の自然数 α を用いて，$\pm\alpha$ で巡回する $2m$ 個の分割した乗算から元の多数桁乗算を復元することが可能となる．多数桁乗算に 2 段階 FMT および分割乗算を利用すると使用メモリ量を大きく削減できる．

High-precision Multiplication by Fast Modulo Transformation

Yasunori Ushiro[†]

In this paper, I would like to present new algorithms for high-precision multiplication method. They are based on Fast Fourier Transformation (FFT) and remainder theorem. The based method is called a Fast Modulo Transformation (FMT). The FMT is composed on a primitive root of one (ω) and It is possible to define not only by a complex number but by a integer. If ω is a complex number, the computational formulation of the FMT is same as the FFT. The FMT has the wide scope about a high-precision multiplication, becouse it is based on remainder theorm. I present integer FMT, cyclic multiplications, two level FMT, direct use of complex FMT and divide multiplications for the FMT application. If α are natural numbers, I show that the FMT can achieve not only ± 1 cyclic multiplication but $\pm\alpha$ cyclic multiplications. In addition, the FMT can divide cyclic multiplications of $\pm\alpha$ for a original long multiplication, when α are different natural numbers. Two level FMT and divide multiplications are possible to reduce a memory size of a high-precision multiplication.

1. はじめに

多数桁乗算の計算法は筆算による方法，中国剰余定理，Karatsuba 法，高速フーリエ変換（FFT）およびこれらを組み合わせた方法が知られている．計算桁数が多い場合はこれらの中で FFT が最も計算量が少ない．しかし，FFT を使用した多数桁乗算は FFT 計算で多くのメモリを使用し，分割して使用すると演算量が増加する．このため，大きい桁数の乗算には桁数を適当に分割し，分割した部分から元の桁数の乗算に復元するのに Karatsuba 法[4]を使用することが多かった．そこで，多数桁乗算で FFT の高速性を生かしながらメモリの使用量を減少させる方法を検討した．その結果，FFT に剰余理論を取り込んだ高速剰余変換（Fast Modulo Transformation，FMT）[1]が有効なことが判明した．FMT は複素数の他に整数でも利用できる．複素数の上で定義すると FMT と FFT は同じ計算式となる．FMT は剰余理論に基づいて作成されているため，多数桁乗算，すなわち畳込み演算に利用すれば中間結果の定義が明白で FFT より見通しが良い．この利点から多数桁乗算への FMT の応用として整数 FMT[3]，巡回乗算，2 段階 FMT，複素 FMT の直接利用および分割乗算が導出できた．複素 FMT の直接利用による多数桁乗算では，通常の乗算のほかに半分の要素数で負巡回乗算もできる．FFT でも正巡回乗算と負巡回乗算[6]は知られているが，整数 FMT による巡回乗算は ± 1 の巡回だけでなく自然数 α に対して $\pm\alpha$ で巡回する乗算が可能と判明した．実用的な巡回 FMT の係数も求めることができた．さらに，多数桁乗算の入力値を m 個に分割し，互いに異なる m 個の自然数 α を用いて，$\pm\alpha$ で巡回する $2m$ 個の

† 株式会社日立製作所エンタープライズサーバ事業部
Enterprise Server Division, Hitachi Ltd.

分割した乗算から元の多数桁乗算が復元できることも分かった．多数桁乗算に 2 段階 FMT を使用すると，8 バイト整数に 2 進 60 桁[7]を詰めて 1 兆桁の計算も可能である．さらに，$2m$ 個の分割乗算を組み合わせると使用メモリ量を大きく削減できる．FMT と FFT は導出方法は異なるが，多数桁乗算への適用では同じ計算式となるため FFT に整数上での計算を含めると，4 章の FMT の適用方法はそのまま FFT にも適用できる．ただし，巡回する多数桁乗算の計算では，FMT が上位要素を巡回させるのに対し，FFT は下位要素を巡回させることに注意を要する．

以下，2 章に高速剰余変換（FMT）の方法，3 章に FMT による畳込み演算の方法，すなわち多数桁乗算の方法，4 章に多数桁乗算への FMT の適用方法を示す．

2. 高速剰余変換（FMT）

n 個の素数 $P_1, P_2, \cdots, P_n$ を使用して各剰余から元の値を復元するのが中国剰余定理である．FMT では剰余（mod）を整数だけでなく記号にも拡張し，記号 E と数 ω，k に対して $f \pmod{E-\omega^k}$ を多項式 f 中の記号 E を ω^k に置き換えることで定義する．さらに，ω を FFT の基本原理である 1 の原始 n 乗根とし，素数の代わりに $E-\omega^k$ を利用する．順変換は多項式の剰余を求めることで，逆変換は剰余から元の多項式の係数を求めることで定義する．

2.1 基本 FMT 変換

$n-1$ 次の E の多項式 f に対して $E-\omega^k$ $(k=0,1,\cdots,n-1)$ の剰余に関する変換である．

2.1.1 順 変 換

多項式 $f \equiv a_{n-1}E^{n-1}+\cdots+a_1E+a_0$ に対して $f_k \equiv f \pmod{E-\omega^k}$ を求める．各 f_k は f 中の E に ω^k を代入して式 (1) が得られる．

$$\begin{aligned} f_k &= a_{n-1}\omega^{(n-1)k}+\cdots+a_1\omega^k+a_0 \\ &= \sum_{l=0}^{n-1} a_l\omega^{lk}, \\ & \quad k=0,1,\cdots,n-1 \end{aligned} \tag{1}$$

2.1.2 逆 変 換

式 (1) の f_k から元の多項式 f の係数 a_j を求める．この計算には下記の ω が原始 n 乗根の条件を利用する．

$$\sum_{k=0}^{n-1}\omega^{(l-j)k} = \begin{cases} n: & l=j \\ 0: & l\neq j \end{cases} \tag{2}$$

式 (1) および式 (2) から次式が得られる．

$$\begin{aligned} \sum_{k=0}^{n-1} f_k\omega^{-kj} &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} a_l\omega^{kl}\omega^{-kj} \\ &= \sum_{l=0}^{n-1} a_l \sum_{k=0}^{n-1}\omega^{(l-j)k} = na_j \end{aligned} \tag{3}$$

したがって，係数 a_j を求める逆変換は次式のように計算できる．

$$\begin{aligned} a_j &= (f_{n-1}\omega^{-(n-1)j}+\cdots+f_1\omega^{-j}+f_0)/n \\ &= \sum_{k=0}^{n-1} f_k\omega^{-kj}/n, \\ & \quad j=0,1,\cdots,n-1 \end{aligned} \tag{4}$$

2.2 拡張 FMT 変換

E の $n-1$ 次の多項式 f と整数 m，$s(s=n/m)$ および有理数 q に対して，ω を原始 s 乗根とするとき，$s-1$ 次の多項式 $f^{(q,s)} \equiv f \pmod{E^s-\omega^{qs}}$ を定義する．拡張 FMT 変換はこの $s-1$ 次の多項式 $f^{(q,s)}$ に対する $E-\omega^{k+q}$ $(k=0,1,\cdots,s-1)$ の剰余変換である．

2.2.1 順 変 換

多項式 $f^{(q,s)} \equiv f \pmod{E^s-\omega^{qs}} \equiv a_{s-1}^{(s)}E^{s-1}+\cdots+a_1^{(s)}E+a_0^{(s)}$ に対して $f_k^{(q,s)} \equiv f^{(q,s)} \pmod{E-\omega^{k+q}}$ を求める．各 $f_k^{(q,s)}$ は $f^{(q,s)}$ 中の E に数 ω^{k+q} を代入して式 (5) が得られる．

$$\begin{aligned} f_k^{(q,s)} &= a_{s-1}^{(s)}\omega^{(s-1)(k+q)}+\cdots+a_1^{(s)}\omega^{k+q}+a_0^{(s)} \\ &= \sum_{l=0}^{s-1} a_l^{(s)}\omega^{(k+q)l}, \\ & \quad k=0,1,\cdots,s-1 \end{aligned} \tag{5}$$

ただし，$a_l^{(s)} = a_l+a_{l+s}\omega^{qs}+\cdots+a_{l+(m-1)s}\omega^{(m-1)qs}$ である．

2.2.2 逆 変 換

$f_k^{(q,s)}$ から $s-1$ 次の多項式 $f^{(q,s)}$ の係数 $a_j^{(s)}$ を求める．式 (5) および式 (2) から次式が得られる．

$$\begin{aligned} \sum_{k=0}^{s-1} f_k^{(q,s)}\omega^{-kj} &= \sum_{k=0}^{s-1}\sum_{l=0}^{s-1} a_l^{(s)}\omega^{(k+q)l}\omega^{-kj} \\ &= \sum_{l=0}^{s-1} a_l^{(s)}\omega^{ql}\sum_{k=0}^{s-1}\omega^{(l-j)k} \\ &= s\omega^{qj}a_j^{(s)} \end{aligned} \tag{6}$$

したがって，係数 $a_j^{(s)}$ を求める逆変換は次式のように計算できる．

$$a_j^{(s)} = (f_{s-1}^{(q,s)}\omega^{-(s-1)j} + \cdots + f_1^{(q,s)}\omega^{-j} + f_0^{(q,s)})\ \omega^{-qj}/s = \omega^{-qj}\sum_{k=0}^{s-1} f_k^{(q,s)}\omega^{-kj}/s,\quad j = 0, 1, \cdots, s-1 \tag{7}$$

3. FMT による畳込み演算

多数桁乗算は適当な桁数で分割し，それぞれ分割した値を多項式の係数とする畳込み演算に帰着できる．多数桁乗算では畳込み演算結果に分割した桁数単位に桁上げする処理を追加すればよい．このため，多数桁乗算と畳込み演算は同じものとして記述する．

3.1 畳込み演算への FMT の適用

次式の f，g で示す E の $n-1$ 次多項式の係数の畳込み演算で作成される E の $2n-1$ 次の多項式を h とする．

$$f \equiv a_{n-1}E^{n-1} + \cdots + a_1E + a_0$$
$$g \equiv b_{n-1}E^{n-1} + \cdots + b_1E + b_0$$
$$h \equiv f \cdot g \equiv c_{2n-1}E^{2n-1} + \cdots + c_1E + c_0 \tag{8}$$

このとき，h の係数 c_j は f，g の係数 a_i，b_{j-i} で次式のように表される．

$$c_j = \sum_{i=\max(0,j-n+1)}^{\min(j,n-1)} a_i b_{j-i},\quad j = 0, 1, \cdots, 2n-1 \tag{9}$$

3.1.1 適用方法

原始 $2n$ 乗根 ω を用いて次式のように計算する．ここで，$k = 0, 1, \cdots, 2n-1$，$j = 0, 1, \cdots, 2n-1$ である．

(1) FMT 順変換で f_k，g_k を計算する．

$$f_k \equiv f \pmod{E - \omega^k} = a_{n-1}\omega^{(n-1)k} + \cdots + a_1\omega^k + a_0$$
$$g_k \equiv g \pmod{E - \omega^k} = b_{n-1}\omega^{(n-1)k} + \cdots + b_1\omega^k + b_0 \tag{10}$$

(2) 各 k ごとに f_k と g_k の乗算を次式で計算する．

$$h_k = f_k \cdot g_k \tag{11}$$

(3) FMT 逆変換で h_k から多項式 h の係数 c_j を求める．

$$c_j = (h_{2n-1}\omega^{-(2n-1)j} + \cdots + h_1\omega^{-j} + h_0)/2n \tag{12}$$

3.1.2 畳込み演算になることの証明

式 (11) に式 (10) および式 (9) を代入し，式 (2) の n を $2n$ に置き換えた ω が原始 $2n$ 乗根の条件から次式が成立する．

$$\sum_{k=0}^{2n-1} h_k\omega^{-jk} = \sum_{k=0}^{2n-1} f_k g_k \omega^{-jk} = \sum_{k=0}^{2n-1}\left(\sum_{i=0}^{n-1} a_i\omega^{ik}\right)\left(\sum_{l=0}^{n-1} b_l\omega^{lk}\right)\omega^{-jk} = \sum_{i=0}^{n-1} a_i \sum_{l=0}^{n-1} b_l \sum_{k=0}^{2n-1} \omega^{(i+l-j)k}$$
$$= 2n \sum_{i=max(0,j-n+1)}^{min(j,n-1)} a_i b_{j-i} = 2nc_j,\quad j = 0, 1, \cdots, 2n-1 \tag{13}$$

この式から式 (12) が導かれる．

3.2 拡張 FMT 変換と畳込み演算の関係式

式 (8) の $n-1$ 次の多項式 f，g を E の $2n-1$ 次ではなく整数 m，$s(s = n/m)$ および有理数 q に対して，ω を原始 s 乗根として $s-1$ 次に畳み込むことを考える．

$$h^{(q,s)} \equiv h \pmod{E^s - \omega^{sq}} \equiv f \cdot g \pmod{E^s - \omega^{sq}} \equiv c_{s-1}^{(q,s)}E^{s-1} + \cdots + c_1^{(q,s)}E + c_0^{(q,s)} \tag{14}$$

3.2.1 適用方法

原始 s 乗根 ω を用いて次式のように計算する．ここで，$k = 0, 1, \cdots, s-1$，$j = 0, 1, \cdots, s-1$ である．

(1) $s-1$ 次の多項式 $f^{(q,s)}$，$g^{(q,s)}$ に変更する．

$$f^{(q,s)} \equiv f \pmod{E^s - \omega^{sq}} = a_{s-1}^{(s)}E^{(s-1)} + \cdots + a_1^{(s)}E + a_0^{(s)}$$
$$g^{(q,s)} \equiv g \pmod{E^s - \omega^{sq}} = b_{s-1}^{(s)}E^{(s-1)} + \cdots + b_1^{(s)}E + b_0^{(s)} \tag{15}$$

ここで，$a_l^{(s)} = a_l + a_{l+s}\omega^{qs} + \cdots + a_{l+(m-1)s}\omega^{(m-1)qs}$ で $b_l^{(s)} = b_l + b_{l+s}\omega^{qs} + \cdots + b_{l+(m-1)s}\omega^{(m-1)qs}$，$l = 0, 1, \cdots, s-1$ である．

(2) 拡張 FMT 順変換で $f_k^{(q,s)}$，$g_k^{(q,s)}$ を計算する．

$$f_k^{(q,s)} \equiv f^{(q,s)} \pmod{E - \omega^{k+q}} = a_{s-1}^{(s)}\omega^{(s-1)(k+q)} + \cdots + a_1^{(s)}\omega^{k+q} + a_0^{(s)}$$
$$g_k^{(q,s)} \equiv g^{(q,s)} \pmod{E - \omega^{k+q}} = b_{s-1}^{(s)}\omega^{(s-1)(k+q)} + \cdots + b_1^{(s)}\omega^{k+q} + b_0^{(s)} \tag{16}$$

(3) 各 k ごとに $f_k^{(q,s)}$ と $g_k^{(q,s)}$ の乗算を次式で計算する．

表 1 P が 2^{24} の近くの原始 n 乗根となる整数 n, ω, P の例
Table 1 n, ω, P in which P is primitive root near 2^{24}.

項番	$n=\alpha=2^{17}$		$n=\beta=3\cdot 2^{15}$		$n=\gamma=3^2\cdot 2^{14}$	
	ω	P	ω	P	ω	P
1	770	$149\alpha+1$	26	$173\beta+1$	175	$114\gamma+1$
2	201	$153\alpha+1$	866	$187\beta+1$	434	$132\gamma+1$
3	70	$155\alpha+1$	562	$194\beta+1$	12	$136\gamma+1$
4	20	$164\alpha+1$	637	$204\beta+1$	604	$147\gamma+1$
5	1498	$165\alpha+1$	100	$220\beta+1$	459	$150\gamma+1$

表 2 P が 2^{48} の近くの原始 n 乗根となる整数 n, ω, P の例
Table 2 n, ω, P in which P is primitive root near 2^{48}.

項番	$n=\alpha=2^{35}$		$n=\beta=3\cdot 2^{32}$		$n=\gamma=3^2\cdot 2^{31}$	
	ω	P	ω	P	ω	P
1	360	$8319\alpha+1$	9375	$22100\beta+1$	8426	$14636\gamma+1$
2	933	$8334\alpha+1$	5549	$22226\beta+1$	226	$14771\gamma+1$
3	1714	$8549\alpha+1$	9520	$22254\beta+1$	4987	$14896\gamma+1$
4	474	$8756\alpha+1$	3991	$22314\beta+1$	9607	$14913\gamma+1$
5	744	$8759\alpha+1$	6955	$22316\beta+1$	7806	$14994\gamma+1$

$$h_k^{(q,s)} = f_k^{(q,s)} \cdot g_k^{(q,s)} \tag{17}$$

(4) 拡張 FMT 逆変換で $h_k^{(q,s)}$ から多項式 $h^{(q,s)}$ の係数 $c_j^{(q,s)}$ を求める．

$$c_j^{(q,s)} = \left(h_{s-1}^{(q,s)}\omega^{-(s-1)j} + \cdots + h_1^{(q,s)}\omega^{-j} + h_0^{(q,s)}\right) \omega^{-qj}/s \tag{18}$$

3.2.2 畳込み演算との関係式

式 (12) の c_j と式 (18) の $c_j^{(q,s)}$ の関係は，式 (8) と式 (14) の定義から次式のようになる．

$$c_j^{(q,s)} = c_j + c_{j+s}\omega^{sq} + \cdots + c_{j+(2m-1)s}\omega^{(2m-1)sq}, \quad j = 0, 1, \cdots, s-1 \tag{19}$$

3.2.3 畳込み演算になることの証明

ω が 1 の原始 s 乗根の条件および式 (15)，式 (16)，式 (17) を式 (18) に代入して次式が成立する．式 (20) で式 (2) の右辺が s になるのは，$i+l-j=0$ および $i+l-j=s$ の 2 つの場合がある．

$$\begin{aligned}
c_j^{(q,s)} &= \sum_{k=0}^{s-1} h_k^{(q,s)}\omega^{-jk}\omega^{-qj}/s \\
&= \sum_{k=0}^{s-1} f_k^{(q,s)} g_k^{(q,s)} \omega^{-j(k+q)}/s \\
&= \sum_{k=0}^{s-1} \left(\sum_{i=0}^{s-1} a_i^{(s)}\omega^{i(k+q)}\right)\left(\sum_{l=0}^{s-1} b_l^{(s)}\omega^{l(k+q)}\right) \omega^{-j(k+q)}/s \\
&= \sum_{i=0}^{s-1} a_i^{(s)} \sum_{l=0}^{s-1} b_l^{(s)}\omega^{(i+l-j)q} \sum_{k=0}^{s-1} \omega^{(i+l-j)k}/s \\
&= \sum_{i=0}^{j} a_i^{(s)} b_{j-i}^{(s)} + \omega^{sq} \sum_{i=j+1}^{s-1} a_i^{(s)} b_{j+s-i}^{(s)} \\
&= \sum_{i=0}^{j}\sum_{k=0}^{m-1} a_i\omega^{sqk} \sum_{l=0}^{m-1} b_{j-i}\omega^{sql} + \omega^{sq} \sum_{i=j+1}^{s-1} \sum_{k=0}^{m-1} a_i\omega^{sqk} \sum_{l=0}^{m-1} b_{j+s-i}\omega^{sql} \\
&= c_j + c_{j+s}\omega^{sq} + \cdots + c_{j+(2m-1)s}\omega^{(2m-1)sq}, \quad j = 0, 1, \cdots, s-1
\end{aligned} \tag{20}$$

したがって，式 (20) は式 (19) に等しくなる．

4. 多数桁乗算への FMT の適用方法

4.1 整数 FMT

整数 FMT の 1 つは $\omega = 2$，$P = \omega^{n/2} + 1$ とすると，ω が法 P のもとで 1 の原始 n 乗根となるのを利用する方法である．ω は 2 以上の任意の整数でも可能である．この方法の問題点は要素数 n が大きくなると，P が指数的に大きくなることである．この問題点を解消するには，適当な大きさの素数 P 上で 1 の原始 n 乗根となる ω を利用すればよい．計算機で利用しやすい原始 n 乗根となる n，ω，P の例を**表 1** および**表 2** に示す．表 1 は倍精度浮動小数点を使用して，P 以下の任意の整数の乗算とその P による剰余が正確に計算できるため，簡単にプログラムが作成できる利点がある．しかし，要素数 n を大きくできない欠点がある．表 2 は倍精度浮動小数点で，P 以下の任意の整数の乗算とその剰余はできないがその部分だけ工夫すれば，倍精度浮動小数点で FMT の計算が

可能である．また，表 1 に比べて要素数 n も大きくとれる．さらに，表 1 は約 4 倍，表 2 は約 16 倍大きな P を選んでも IEEE の倍精度浮動小数点を使用して計算可能であるが，FMT のプログラム作成では多少 P を超える値の乗算が可能なようにしておく方がプログラム作成が容易になる．このため表 1 と表 2 の値を選んだ．これらの表の 1/2，1/3，$\cdots$ 倍の n に適用するには，P はそのままで ω を法 P のもとで ω^2，ω^3，$\cdots$ に置き換えればよい．表 1，表 2 の複数個の P による FMT の畳込み演算結果は，各 P が素数のため中国剰余定理で結果を重ね合わせ，各要素の桁数を拡大することができる．

4.2 正巡回と負巡回乗算

拡張 FMT 変換による畳込み演算を使用する．式 (19) において $s=n$，$m=1$ で $q=0$ のとき正巡回 FMT，$q=1/2$ のとき負巡回 FMT と呼び，それぞれに対応する乗算を正巡回乗算および負巡回乗算と呼ぶ．正巡回 FMT と負巡回 FMT を求める目的は多数桁乗算が 2 分割でき FMT での使用メモリ量を半減することである．さらに，M，n を整数とするとき，任意の整数乗算結果の M^n-1 や M^n+1 における剰余が直接計算できるため，この形の合成数の素因数分解の計算にも利用できる．また，負巡回 FMT は 2 段階 FMT の下位 FMT として必要な計算手段である．式 (19) に $s=n$，$m=1$ での q に 0 および 1/2 を代入すると $\omega^0=1$，$\omega^{n/2}=-1$ で次式のようになる．

$$\begin{aligned} c_j^{(0,n)} &= c_j + c_{j+n} \\ c_j^{(1/2,n)} &= c_j - c_{j+n}, \\ & \qquad j=0,1,\cdots,n-1 \end{aligned} \tag{21}$$

正巡回乗算の係数 $c_j^{(0,n)}$ と負巡回乗算の係数 $c_j^{(1/2,n)}$ から次式のように元の乗算の $2n-1$ 次の係数 c_j が計算できる．

$$\begin{aligned} c_j &= (c_j^{(0,n)} + c_j^{(1/2,n)})/2 \quad ; \\ &\quad \text{元の乗算の下位要素の係数} \\ c_{j+n} &= (c_j^{(0,n)} - c_j^{(1/2,n)})/2 \quad ; \\ &\quad \text{元の乗算の上位要素の係数,} \\ &\quad j=0,1,\cdots,n-1 \end{aligned} \tag{22}$$

4.3 2 段階 FMT による多数桁乗算

FFT を利用した多数桁乗算の問題点は FFT 利用時に大量のメモリを使用することである．これを解消する方法として 2 段階 FMT が有効である．2 段階 FMT による乗算とは，1 要素に多数桁を持つ N 要素の乗算を $P=\omega^{N/2}+1$ を法とする整数 FMT で行い，そこで発生する式 (17) に対応する各要素ごとの乗算にも FMT を使用する方法である．P を法とする整数 FMT を上位 FMT，各要素ごとの乗算に使用する FMT を下位 FMT と呼ぶ．ω として 2 のべき乗を使用すれば，上位 FMT は加減算とシフト演算だけとなる特長がある．ここで重要なのは下位 FMT の乗算結果がそのまま法 P の剰余となることである．乗算結果が法 P の剰余となるためには，下位 FMT に式 (23) を満たす負巡回 FMT を利用すればよい．ここで下位 FMT の要素数を n，1 要素に入れる α 進数を k 桁とし，上位 FMT の要素数を N，原始 N 乗根を $\omega=\alpha^l$ とする．

$$l \cdot N = 2n \cdot k \tag{23}$$

4.3.1 計 算 方 法

2 段階 FMT による多数桁乗算の基底は 2 進数である必要はなく，10 進数でも他の基底でも可能であるが，説明を簡単にするため 2 進数を仮定して説明する．基底を変更するときは，ω の値を基底数のべき乗にする必要がある．

(1) 記号の定義

ここで使用する記号を下記のように定義する．

N：上位 FMT の要素数，n：下位 FMT の要素数

F_P，F_P^{-1}：上位正巡回 FMT の順変換および逆変換

F_N，F_N^{-1}：上位負巡回 FMT の順変換および逆変換

f_N，f_N^{-1}：下位負巡回 FMT の順変換および逆変換

$\bullet$：通常の乗算（畳み込み演算による全要素乗算）

$\otimes$：項別乗算（対応する要素の乗算）

(2) 上位 FMT の計算

上位 FMT の要素数を N としたとき，上位 FMT の 1 要素に 2 進 mN 桁を記憶できるようにする．このとき，上位 FMT は $\omega=2^{2m}$ で $P=\omega^{N/2}+1$ として構成する．ともに N 個の要素を持つ A と B の多数桁の乗算を下記のように表す．乗算結果は $2N$ 個の要素になる．

$$A \bullet B = (c_{2N-1}, c_{2N-2}, \cdots, c_N, \cdots, c_1, c_0)$$

A と B の多数桁の乗算を $2N$ 個の要素の FMT ではなく，N 個の要素の正巡回 FMT と，負巡回 FMT による乗算を利用して下記のように計算する．

正巡回乗算：

$$F_P^{-1}(F_P(A) \otimes F_P(B)) = (d_{N-1}, d_{N-2}, \cdots, d_1, d_0)$$

負巡回乗算：

$$F_N^{-1}(F_N(A) \otimes F_N(B)) = (e_{N-1}, e_{N-2}, \cdots, e_1, e_0)$$

A と B の多数桁乗算の $2N$ 個の要素 c_{j+N}，c_j は下記で計算できる．

$$c_{j+N} = (d_j - e_j)/2$$

$$c_j = (d_j + e_j)/2, \quad j=0,1,\cdots,N-1$$

正巡回乗算と負巡回乗算で現れる要素単位の項別乗算

($\otimes$) は下位 FMT を利用して行う．

(3) 下位 FMT の計算

上位 FMT の要素単位に実施される項別乗算に下位 FMT は利用される．そのため，下位 FMT を利用する多数桁の乗算は法 P の上で実行される必要がある．$P = \omega^{N/2} + 1 = 2^{mN} + 1$ のため，下位 FMT の要素数を n，下位 FMT の 1 要素に詰める 2 進桁数を k とすると下記関係が必要となる．

$$m \cdot N = n \cdot k$$

この条件を満たせば，下位 FMT に実数 FMT を使用するか，整数 FMT を使用するかは自由である．要素数 n でこの条件のもとに，多数桁乗算結果が法 P を満たすためには，負巡回 FMT を下位乗算に使用すればよい．すなわち，2 進 k 桁を 1 要素とする a と b の nk 桁どうしの乗算は下記のように n 要素の負巡回 FMT を使用することにより，法 P のもとで nk 桁で求まる．

$$a \cdot b \pmod{P} = f_N^{-1}(f_N(a) \otimes f_N(b))$$

4.3.2 使用メモリ量の比較

具体的なメモリ使用量の比較例として，入力 A，B は 0.5 兆桁で出力 C が 1 兆桁（10 進換算）となる 3 種類の多数桁乗算を大規模並列計算機で計算するとして比較する．TB はテラバイトの略である．

(1) 2 段階 FMT による 1 兆桁乗算

上位 FMT は正巡回 FMT と負巡回 FMT を重ね合わせる．

上位 FMT は整数 8 バイトに 2 進 60 桁を詰めて計算できる．

FMT（A）および計算結果	：0.5TB
FMT（B）	：0.5TB
正巡回結果の保存	：0.25TB
下位 FMT および通信ワーク	：0.05TB
合計	：1.3TB

(2) 実 FFT による 1 兆桁乗算

乗算入力の各要素の値は絶対値が最小になるように正，負の符号付で表示する．正，負の値で平均化されるため結果 C の各要素で保持する桁数は理論限界より長くできる．それでも，実数 8 バイトに 2 進 9 桁詰めが限度である．正巡回 FFT と負巡回 FFT の重ね合わせはしない．

FFT（A）および計算結果	：3.0TB（2 進 9 桁）
FFT（B）	：3.0TB
FFT ワーク	：3.0TB（並列化）
合計	：9.0TB

(3) 実 FFT と Karatsuba 法による 1 兆桁乗算

Karatsuba 法の適用回数が増加するとメモリ量は減るが演算量は増加する．1 回 Karatsuba 法を使用すると，FFT 領域は半減し，Karatsuba 法のワークは増加する．

	FFT	ワーク	合計	演算量比
0 回適用：	9.0TB	+ 0.0TB	= 9.0TB	1.0 倍
1 回適用：	4.5TB	+ 1.0TB	= 5.5TB	1.5 倍
2 回適用：	2.3TB	+ 1.5TB	= 3.8TB	2.3 倍
3 回適用：	1.2TB	+ 1.7TB	= 2.9TB	3.4 倍
4 回適用：	0.6TB	+ 1.8TB	= 2.4TB	5.1 倍

4.4 複素 FMT の直接使用による多数桁乗算

複素 FFT による多数桁乗算では，複素共役性を利用して実 FFT に変換する必要がある．これに対し拡張 FMT 変換を利用すると複素 FMT で直接多数桁乗算が可能となる．複素 FMT で直接多数桁乗算が可能なことの利点は，実 FMT を経由しないためプログラムが簡単になることである．特に，並列計算機において複素 FMT（または FFT）の計算結果を実 FMT（または FFT）に変換するのは，先頭と中間の要素を特別処理することで，対応する要素位置が 1 ずれるため結構面倒な工夫をする必要がある．それに対して，複素 FMT による直接多数桁計算では，要素位置は正確に要素数の半分か 1/4 ずれるだけのため並列処理も簡単になる．

式 (19) に ω を複素数として $s = n$，$m = 1$ と $q = 1/4$ を適用すると，$\omega^{n/4} = i$ から次式のようになる．ここで i は虚数単位である，

$$c_j^{(1/4,n)} = c_j + i \cdot c_{j+n}, \quad j = 0, 1, \cdots, n-1 \tag{24}$$

式 (24) は，入力値の虚部をゼロにした複素 FMT による乗算結果 $c^{(1/4)}$ の実部が元の多数桁乗算の下位要素 (c_j) に対応し，虚部が上位要素 (c_{j+n}) に対応することを示している，

さらに，$n-1$ の代わりに $n/2-1$ 次で畳み込むことを考える．式 (19) に $s = n/2$，$m = 2$ と $q = 1/4$ を代入すると，$\omega^{n/8} = \omega^{s/4} = i$ から次式のようになる．

$$\begin{aligned} c_j^{(1/4,n/2)} &= c_j + c_{j+n/2}\omega^{n/8} + c_{j+n}\omega^{n/4} \\ &\quad + c_{j+3n/2}\omega^{3n/8} \\ &= (c_j - c_{j+n}) + i \cdot (c_{j+n/2} - c_{j+3n/2}), \\ &\quad j = 0, 1, \cdots, n/2-1 \end{aligned} \tag{25}$$

n 要素の整数入力値に $(\bmod E^{n/2} - i)$ を施すと，入力の上位 $n/2$ 要素を虚部に下位 $n/2$ 要素を実部にした $n/2$ 要素の複素数となる．式 (25) は，この $n/2$ 要素の複素 FMT の乗算結果 $c_j^{(1/4,n/2)}$ の実部が元の

表 3 $n = 2^{20}$ で $\pm\alpha$ で巡回する乗算となる整数 FMT の係数の例
Table 3 Examples of integer FMT that is $\pm\alpha$ cyclic multiplication.

α	P	q	ω	ω^q
2	$280049478 \cdot n + 1$	1/93349826	243728030273313	196084934801470
3	$322504032 \cdot n + 1$	1/80626008	278086974322725	45280667942168
4	$283546518 \cdot n + 1$	1/94515506	237787923135329	113109652731064
5	$205984108 \cdot n + 1$	1/205984108	43678380610349	192726477692165
6	$282686616 \cdot n + 1$	1/282686616	20159304809371	288630981189601
7	$222921552 \cdot n + 1$	1/111460776	161161411978886	163047718620128
8	$280049478 \cdot n + 1$	1/93349826	243728030273313	70308039835214
9	$322504032 \cdot n + 1$	1/40313004	93762856436902	259915309872599
10	$314771758 \cdot n + 1$	1/314771758	149271167055472	94022866127931

n 要素の負巡回乗算の下位要素 $(c_j - c_{j+n})$ に，虚部が上位要素 $(c_{j+n/2} - c_{j+3n/2})$ に対応していることを示している．したがって，複素 FMT で負巡回乗算が可能となる．

4.5 ± 自然数での巡回乗算

α を任意の自然数とするとき，整数 FMT では $\pm\alpha$ で巡回する多数桁乗算の計算が可能である．$\alpha = 1$ のときは正および負巡回乗算になる．これは，$s = n$，$m = 1$ および q を有理数として，法 P 上での原始 n 乗根 ω を使用する整数 FMT で式 (19) が成立するためである．$\pm\alpha$ で巡回する乗算の第 1 の目的は多数桁の分割乗算に利用することである．それ以外に，M，n を整数とするとき，任意の整数乗算結果の $M^n - \alpha$ や $M^n + \alpha$ における剰余が直接計算できるため，合成数の素因数分解の計算にも利用できる．$\pm\alpha$ で巡回する乗算ができるためには，要素数 n に対して法 P と $\alpha \equiv \omega^{nq} \pmod P$ となる自然数の組 P，ω，ω^q および有理数 q があればよい．この例として，$n = 2^{20}$ で 2 から 10 までの各 α に対応する値を**表 3** に示す．表 3 の n が 2^{19}，$2^{18}, \cdots$ に適用するには α，P，q はそのままでω，ω^q の値をそれぞれ 2 乗，4 乗，$\cdots$ すればよい．表 3 の記号を使用すると，整数 FMT での巡回乗算は $\omega^{n/2} = -1$ のため次式のようになる．ここで，c_j，$c_j^{(q,n)}$ は式 (9) および式 (18) で定義したものである．

$$\begin{aligned} c_j^{(q,n)} &= c_j + \alpha \cdot c_{j+n} \\ c_j^{(q+1/2,n)} &= c_j - \alpha \cdot c_{j+n}, \\ & \qquad j = 0, 1, \cdots, n-1 \end{aligned} \tag{26}$$

4.6 巡回乗算による多数桁の分割乗算

多数桁の分割乗算の目的は計算メモリ量の削減である．さらに重要な用途と思われるのは，Disk などの外部記憶装置を利用してメモリの何十倍もの桁数の乗算が実用的な時間で可能になることである．分割乗算を利用すると演算量を増加させないで，Disk への入出力回数を大幅に削減した超多数桁の乗算が可能になる．

式 (22) で正巡回乗算と負巡回乗算から元の乗算の係数 c_j が計算できることを示した．これをさらに進めて，$2m$ 個の異なる巡回乗算から元の乗算の係数 c_j が計算できる．

式 (19) で $s = n/m$ および $q = k/2m$ を代入すると次式が得られる．ただし，ω は原始 s 乗根で ω_n は原始 n 乗根を示し，$\omega_n = \omega^{1/m}$ である．

$$\begin{aligned} c_j^{(k/2m,n/m)} = c_j &+ c_{j+n/m}\omega_n^{kn/2m} + \cdots \\ &+ c_{j+(2m-1)n/m}\omega_n^{(2m-1)kn/2m}, \\ & j = 0, 1, \cdots, n/m-1, \\ & k = 0, 1, \cdots, 2m-1 \end{aligned} \tag{27}$$

この式から逆に $2m$ 分割したときの元の $2n-1$ 次の係数は次式のようになる．

$$\begin{aligned} c_{j+nk/m} = \sum_{l=0}^{2m-1} c_j^{(l/2m,n/m)}\omega_n^{-lkn/2m}/2m, \\ j = 0, 1, \cdots, n/m-1, \\ k = 0, 1, \cdots, 2m-1 \end{aligned} \tag{28}$$

具体的例として $m = 2$ で 4 個に分割して計算する例を次式に示す．ここで，ω_n は原始 n 乗根で，$\omega_n^{-n/2} = -1$ である．

$$\begin{aligned} c_j &= (c_j^{(0,n/2)} + c_j^{(1/4,n/2)} + c_j^{(1/2,n/2)} \\ &\quad + c_j^{(3/4,n/2)})/4 \\ c_{j+n/2} &= (c_j^{(0,n/2)} + c_j^{(1/4,n/2)}\omega_n^{-n/4} - c_j^{(1/2,n/2)} \\ &\quad - c_j^{(3/4,n/2)}\omega_n^{-n/4})/4 \\ c_{j+n} &= (c_j^{(0,n/2)} - c_j^{(1/4,n/2)} + c_j^{(1/2,n/2)} \\ &\quad - c_j^{(3/4,n/2)})/4 \\ c_{j+3n/2} &= (c_j^{(0,n/2)} - c_j^{(1/4,n/2)}\omega_n^{-n/4} - c_j^{(1/2,n/2)} \\ &\quad + c_j^{(3/4,n/2)}\omega_n^{-n/4})/4, \\ &\quad j = 0, 1, \cdots, n/2-1 \end{aligned} \tag{29}$$

これは，元の $2n$ 個の要素の多数桁乗算結果が 4 分

割されて，それぞれ $n/2$ 個の要素の乗算として求められることを示している．

5. まとめ

高速剰余変換（FMT）を利用し，多数桁乗算において FFT と同じ演算量でメモリ使用量を削減する方法を考案した．整数 FFT と正負の巡回乗算はすでに FFT の応用として知られているが，負巡回の性質を利用した 2 段階 FMT，複素 FMT の直接利用による多数桁乗算，自然数 α に対して $\pm\alpha$ 巡回乗算およびそれを使用した分割乗算の提案は他に見当たらない．また倍精度浮動小数点で表示できる法 P のもとで $\pm\alpha$ で巡回する FMT の係数を具体的に提示できた．2 段階 FMT および FMT による分割乗算により，多数桁乗算のメモリ使用量が FFT に比較して大幅に削減でき，当初の目的が達成できた．FMT は複素数で定義すると FFT と同じ計算式となる．一方，剰余理論をベースにしている FMT は多数桁乗算の中間結果の定義が明白であり，FFT より見通しが良い点が特長である．本論文の多数桁乗算への適用例を示す，FORTRAN と C のプログラムを http://www.dept.edu.waseda.ac.jp/math/ushiro/research/fmtapply.htm に掲載する．今後の課題として，2 段階 FMT と分割乗算を組み合わせてメモリ容量を大幅に超えた桁数の乗算を，外部記憶装置を使用して計算するパッケージの作成と性能評価がある．

謝辞 本論文に対する貴重なコメントをいただき，円周率世界記録達成のプログラムに本論文の 2 段階 FMT（FFT）を採用させていただいた，東京大学金田康正教授（円周率計算プロジェクトリーダ）に，謹んで感謝の意を表する．

参考文献

1) 後　保範：FMT（高速剰余変換）(2002). http://www.dept.edu.waseda.ac.jp/math/ushiro/ushiro/method/fmt.htm
2) 後　保範，金田康正，高橋大介：級数に基づく多数桁計算の演算量削減を実現する分割有理数化法，情報処理学会論文誌，Vol.41, No.6, pp.1811–1819 (2000).
3) 後　保範，長堀文子：ベクトル計算機による整数上の FFT 計算，情報処理学会全国大会，No.27, pp.1293–1294 (1983).
4) 高橋大介，金田康正：分散メモリ計算機による円周率の 515 億桁計算，情報処理学会論文誌，Vol.39, No.7, pp.2074–2083 (1998).
5) Knuth, D.E.: *The Art of Computer Programming*, Vol.2, Addison Wesley (1981).
6) 大浦拓哉：円周率公式の改良と高速多倍長計算の実装，情報処理学会研究報告，98-HPC-74, pp.25–30 (1998).
7) 後　保範：世界記録のための工夫点 (2002). http://www.dept.edu.waseda.ac.jp/math/ushiro/pirecord/pistudy.htm

(平成 15 年 3 月 24 日受付)
(平成 15 年 10 月 16 日採録)

後　保範（正会員）

1945 年生．1967 年早稲田大学理工学部数学科卒業．同年（株）日立製作所入社．以来，スーパーコンピュータ用数値計算ライブラリーの開発および応用技術の開発に従事．現在，同社エンタープライズサーバ事業部に勤務．2002 年 11 月，東大金田康正教授ほかと共同で SR8000/MPP を使用し円周率 1 兆 2411 億桁の世界記録を達成．早稲田大学教育学部数学専修非常勤講師．