アルゴリズム 64－6
（1998．9．16）

Fast Remainder Calculation in Polynomial Multiplication

伊豆 哲也, 野呂 正行

富士通研究所HPC研究センター
〒211-8588 川崎市中原区上小田中4-1-1
Email: izu@flab.fujitsu.co.jp, noro@para.flab.fujitsu.co.jp

Abstract

多項式 $f(x), g(x)$ の積 $f(x)g(x)$ に関する剰余 $f(x)g(x) \bmod x^m$ の計算法において，三角積という手法を用いた高速な計算アルゴリズムを提案する．本アルゴリズムは部分的に従来の高速積計算法が適用できる点と，並列計算が可能である点が特徴的である．計算量的には従来の高速計算法の域を越えられていないが，現実的な大きさの多項式に対しては一定の効果を持つことがわかった．三角積の応用例として，(1) モジュラー多項式の計算，(2) 一般的な剰余 $f(x)g(x) \bmod n(x)$ の計算，(3) 有限体 $GF(q)$ における演算の計算への適用について述べる．

Fast Remainder Calculation in Polynomial Multiplication

IZU Tetsuya, NORO Masayuki

FUJITSU Laboratories Ltd.
4-1-1, Kamikodanaka,
Nakahara-ku, Kawasaki, 211-8588, Japan
Email: izu@flab.fujitsu.co.jp, noro@para.flab.fujitsu.co.jp

Abstract

We propose a faster algorithm to calculate $f(x)g(x) \bmod x^m$, where $f(x), g(x)$ are polynomials. In the algorithm, we can use existing fast multiplication algorithms, and can use parallel computation. We also give three exapmles: (1) the calculation of the modular polynomial, (2) fast calculation of general remainder $f(x)g(x) \bmod n(x)$, (3) calculation in the $GF(q)$.

1 はじめに

多項式の積計算はさまざまな場面で必要となる，基本的な演算の一つである．これまでにさまざまな高速計算法(Karatsuba法など)が提案されているが，実際に計算する場合には，用途，データサイズ，計算機環境などに合わせてアルゴリズム(またはその組み合わせ)を選択しなくてはならない．

本稿では，多項式の積のある次数以下の項のみが必要であるような場面，つまり多項式 $f(x), g(x)$ の積 $f(x)g(x)$ に関する剰余

$$f(x)g(x) \bmod x^m$$

における高速計算法を提案する．ポイントは剰余に必要となる組み合わせのみを計算する点であるが，部分的に従来の高速積計算法を適用できるという意味では従来法の拡張であり，さらには並列計算が可能であるという特徴を併せもっている．

本稿が想定している剰余計算が必要となる場面はいくつか考えられ，そのそれぞれで三角積が効力を発揮することができる．例として，モジュラー多項式と呼ばれる多項式の具体的な計算例，一般的な剰余

$$f(x)g(x) \bmod n(x)$$

への適用可能性，有限体 $GF(q)$ における演算の実現法の3例について述べる．特に有限体における演算は，暗号や符号といった応用数学の場面で実際に必要となる演算である．

本稿の構成は以下の通りである．第2節で多項式の積計算に関するアルゴリズムを準備した後，第3節で三角積の手法を紹介し，その計算量を考察する．第4節では三角積を用いた適用例について述べる．

2 準備

本節では，多項式の積計算に関するアルゴリズムとして，Karatsuba法と離散フーリエ変換を用いた方法(DFT法)を紹介する．詳細については基本的な教科書(例えば[Knu81],[GCL92]など)を参照されたい．

以下 R を可換環とし，R において加算と乗算を実行するのに必要な時間(R にのみ依存する定数)をそれぞれ t_{add}, t_{mul} とおく．

2.1 Karatsuba法

$f(x), g(x)$ を高々 $n-1$ 次の多項式とし，積 $f(x)g(x)$ を計算するのに必要な計算量を $M(n)$ で表す．初等的な方法を用いると，積 $f(x)g(x)$ は R における n^2 回の乗算と $(n-1)^2$ 回の加算で求められるから，$M(n)$ の上限は

$$M(n) \leq n^2 t_{mul} + (n-1)^2 t_{add}$$

で与えられる．

Karatsubaは巧妙な式変形を用いることによって，高速な計算方法を実現した[Kar63]．いま $n = 2m$ とすると，$f(x), g(x)$ は

$$\begin{aligned} f(x) &= f_l(x) + f_u(x)x^m, \\ g(x) &= g_l(x) + g_u(x)x^m \end{aligned}$$

と表せる(ただし $f_l(x), f_u(x), g_l(x), g_u(x)$ は高々 $m-1$ 次の多項式)．このとき積 fg は

$$\begin{aligned} &(f_l + f_u x^m)(g_l + g_u x^m) \\ &= f_l g_l \\ &\quad + (f_l g_l + f_u g_u + (f_u - f_l)(g_l - g_u))\, x^m \\ &\quad + f_u g_u x^{2m} \end{aligned}$$

となる．多項式 $f_l, f_u, g_l, g_u, f_u - f_l, g_l - g_u$ はすべて高々 $m-1$ 次だから，Karatsuba法を用いた場合の計算量 $M_K(n)$ は

$$M_K(n) \leq 3M(m) + (5n-4)t_{add}$$

と評価できる．$n = 2^k$ としてKaratsuba法を繰り返し適用すると，$M(1) = t_{mul}$ により

$$M_K(2^k) = 3^k t_{mul} + (8 \cdot 3^k - 10 \cdot 2^k + 2)t_{add}$$

を得る．よってKaratsuba法の漸近的な計算量は

$$M_K(n) = O(n^{\log_2 3}(t_{mul} + 8t_{add}))$$

と表される．

Karatsuba法は非常に単純な方法であり，多項式の分割によるオーバーヘッドもほとんど見られない．このため，一般に n が数～10数次程度から効力を発揮する．

2.2 離散フーリエ変換を用いた方法

可換環 R の要素を成分とする長さ N のベクトル

$$\begin{aligned} \boldsymbol{f} &= (f_0, f_1, \ldots, f_{N-1}), \\ \boldsymbol{g} &= (g_0, g_1, \ldots, g_{N-1}) \end{aligned}$$

に対し, 長さ N の畳み込みベクトル

$$\boldsymbol{h} = (h_0, h_1, \ldots, h_{N-1}) = \boldsymbol{f} \otimes \boldsymbol{g}$$

を以下のように定義する:

$$h_k = \sum_{i+j \equiv k \pmod N} f_i g_j \quad (0 \le k \le N-1).$$

ここで多項式 $F(x), G(x), H(x)$ を

$$\begin{aligned} F(x) &= \sum_{i=0}^{N-1} f_i x^i, \quad G(x) = \sum_{i=0}^{N-1} g_i x^i, \\ H(x) &= \sum_{i=0}^{N-1} h_i x^i \end{aligned}$$

と定義すると, 畳み込みの定義により

$$H(x) \equiv F(x)G(x) \pmod{x^N - 1}$$

となる. したがって, 何らかの方法によって畳み込みが高速に計算できるのであれば, 多項式の積計算に応用することが可能であろう.

この畳み込みを高速に実現する計算法が離散フーリエ変換である. 高々 $n-1$ 次の多項式

$$F(x) = \sum_{i=0}^{n-1} f_i x^i, \ G(x) = \sum_{i=0}^{n-1} g_i x^i$$

が与えられたときに, $N > 2n-2$ となる N を選ぶ. そして $F(x), G(x)$ を $N-1$ 次の多項式と見なしたときの各係数のなすベクトル

$$\begin{aligned} \boldsymbol{f} &= (f_0, f_1, \ldots, f_{N-1}), \\ \boldsymbol{g} &= (g_0, g_1, \ldots, g_{N-1}) \end{aligned}$$

を考える. ここで ω を 1 の $N/2$ 乗根とするとき, ベクトル

$$\begin{aligned} \overline{\boldsymbol{f}} &= (F(1),\ F(\omega),\ \ldots,\ F(\omega^{N-1})), \\ \overline{\boldsymbol{g}} &= (G(1),\ G(\omega),\ \ldots,\ G(\omega^{N-1})) \end{aligned}$$

を $\boldsymbol{f}, \boldsymbol{g}$ の離散フーリエ変換という. このとき $\overline{\boldsymbol{f}}, \overline{\boldsymbol{g}}$ の内積

$$(\ F(1)G(1),\ F(\omega)G(\omega),\ \ldots,\ F(\omega^{N-1})G(\omega^{N-1})\ )$$

は, $\boldsymbol{f}, \boldsymbol{g}$ の畳み込み $\boldsymbol{h} = \boldsymbol{f} \otimes \boldsymbol{g}$ を離散フーリエ変換した結果 $\overline{\boldsymbol{h}}$ に等しい. よって逆フーリエ変換することによって $\boldsymbol{h}$ が求められ, 結果として $H(x) \equiv F(x)G(x) \pmod{x^N - 1}$ を得ることができる. この計算法を離散フーリエ変換を用いた方法 (DFT 法) と呼ぶ. DFT 法による計算量は, 漸近的に

$$M_F(n) \le 2n \log_2 (2n)(3t_{mul} + 6t_{add})$$

で与えられる.

DFT 法は本質的に高速な積計算法であるが, 複雑な計算を必要とするため, 実装は簡単ではない. また, 変換におけるオーバーヘッドがあるため, 一般には n が大きくないと効果が現れない. しきい値は環境に依存するが, およそ 100 ～ 300 次程度であろう.

3 三角積

高々 $n-1$ 次の多項式 $f(x), g(x)$ が与えられたとき, その積に関する剰余 $f(x)g(x) \bmod x^m$ の計算法を考える. もちろん積 $f(x)g(x)$ を求めてから $m-1$ 次以下の部分をとりだしてもよいが, そもそも x^m より高次の項に対応する部分の計算は不用である. そこでわれわれは必要な項のみを高速に計算する方法—三角積—を提案する.

以下 $f(x)g(x) \bmod x^m$ を計算するのに必要な計算量を $T(m)$ で表す.

3.1 アルゴリズム

剰余 $f(x)g(x) \bmod x^m$ を求めるのに, 必要となる高々 $m(m+1)/2$ 組の係数の積を独立に計算して加える方法が考えられる. この方法による計算量は

$$T(m) \le \frac{m(m+1)}{2} t_{mul} + \frac{m(m-1)}{2} t_{add}$$

と評価できるが, m に関する指数時間アルゴリズムである.

いま2つの多項式 $f(x)$, $g(x)$ の係数を図示し, 積 $f(x)g(x)$ を計算するのに必要な係数同士の積を点で表すことにすると, その全体は長方形をなしている. このとき剰余計算に必要な係数同士の積を表す点は三角形を作っており, その内部に適当な大きさの長方形が存在している. 長方形は多項式と多項式の(通常の)積を意味するから, この部分には通常の高速積計算法を用いることが可能である. つまり係数同士を独立に計算するより, 三角形を適当な大きさで分割して, まとめて考えたる方法が考えられる(分割統治法の1種). このような分割を用いた剰余計算法を三角積と呼ぶことにする.

この分割は帰納的である. つまり分割後にできた三角形の部分に再び三角積を用いることによって, 長方形の部分を増やすことができる. それぞれの長方形は積の高速計算法が適用できるので, 全体としての効率を上げられることになる.

3.2 三角積の例

例として $n=6$, $m=5$ の場合を考える. このとき $f(x)g(x) \bmod x^5$ を求めるのに必要な係数の積の組み合わせは, 表の ∘, • 印の部分である.

	f_0	f_1	f_2	f_3	f_4	f_5
g_0	•	•	•	∘	∘	
g_1	•	•	•	∘		
g_2	•	•	•			
g_3	∘	∘				
g_4	∘					
g_5						

表の • の部分の長方形は2次式と2次式の積に対応しており, この部分には高速計算法が適用できる.

3.3 計算量

三角積の計算量 $T(m)$ を考察する. 簡単のため $n=m=2^k$ の場合で考える. このとき $T(m)$ に関する漸化式は

$$T(2^k) = M(2^{k-1}) + 2T(2^{k-1}) + 2\cdot 2^{k-1}t_{add}$$

で与えられるので, $T(1)=t_{mul}$ により

$$T(2^k) = M(2^{k-1}) + 2M(2^{k-1}) + \ldots$$
$$+2^{k-1}M(1) + 2^k t_{mul} + k\cdot 2^k t_{add}$$

を得る.

多項式の積計算にKaratsuba法を用いると, この式に

$$M_K(2^k) = 3^k t_{mul} + (8\cdot 3^k - 10\cdot 2^k + 2)t_{add}$$

を代入して

$$T_K(2^k) =$$
$$3^k t_{mul} + (8\cdot 3^k - (4k+6)2^k - 2)t_{add}$$

を得る.

$M_K(2^k)$ と $T_K(2^k)$ の計算量を比較すると, 本質的な差異は見られず, 三角積が優れているとは言い難い. しかし,

$$C_{add,K} = 8\cdot 3^k - 10\cdot 2^k + 2$$
$$C_{add,T} = 8\cdot 3^k - (4k+6)2^k - 2$$

として, その比率を比べると以下のようになる:

k	$C_{add,K}/C_{add,T}$	k	$C_{add,K}/C_{add,T}$
1	3.000	6	1.328
2	2.429	7	1.234
3	1.971	8	1.168
4	1.667	9	1.121
5	1.465	10	1.087

よって200次程度の多項式において, 加法の計算回数が2割程度少なくてすむことがわかる.

t_{mul} と t_{add} の比率はシステム構成によって異なるが, $t_{mul}=4t_{add}$ として代入すると,

$$M'_K(2^k) = 3\cdot 3^k - \frac{5}{2}2^k + \frac{1}{2}$$
$$T'_K(2^k) = 3\cdot 3^k - (k+\frac{3}{2})2^k - \frac{1}{2}$$

となり, その比率 $M'_K(2^k)/T'_K(2^k)$ は次のようになる:

k	$M'_K(2^k)/T'_K(2^k)$	k	$M'_K(2^k)/T'_K(2^k)$
1	1.286	6	1.188
2	1.400	7	1.141
3	1.382	8	1.104
4	1.317	9	1.076
5	1.248	10	1.056

よって200次程度の多項式において, 全体の計算量が1割程度低減できていることがわかる.

一方, 多項式の積計算にDFT法を使用した場合には,

$$M_F(2^k) = 6(k+1)2^k(t_{mul} + 2t_{add})$$

を代入して,

$$T_F(2^k) =$$

$$(k^2+3k+1)2^{k-1}t_{mul} + (k^2+5k)2^{k-1}t_{add}$$

を得る. この場合は, 三角積を用いるよりも, DFT法を用いた法が計算量的に優れていることがわかる.

三角積の積計算にどのようなアルゴリズムかを用いるかによって, 三角積の計算量は変化する. 現実的には分割等のオーバーヘッドが存在するため, k がある程度の範囲でしか有効でないと思われる.

4 応用例

三角積の適用例として, モジュラー多項式を計算した実験結果と, 一般的な剰余 $f(x)g(x) \bmod n(x)$ の高速計算の実現について述べる.

4.1 モジュラー多項式の計算

4.1.1 モジュラー多項式

複素上半平面上の点 z に対し $q = e^{2\pi z\sqrt{-1}}$ と定め, 楕円モジュラー関数

$$\frac{1}{q} + 744 + 196884q + 21493760q^2 + \ldots$$

を z, q の多項式 $j(z), j(q)$ として同一視する. このとき素数 l に対するモジュラー多項式 $\Phi_l(X,j)$ とは,

$$\Big(X - j(lz)\Big)\prod_{k=0}^{l-1}\Big(X - j\Big(\frac{z+k}{l}\Big)\Big)$$

$$= X^{l+1} + \sum_{k=1}^{l+1}(-1)^k s_k(j) X^{l-k+1}$$

で定められる多項式のことをいう. ここで係数 $s_k(j)$ は j の整数係数多項式であり,

$$J_l = \Big\{j(lz), j\Big(\frac{z}{l}\Big), j\Big(\frac{z+1}{l}\Big), \ldots, j\Big(\frac{z+l-1}{l}\Big)\Big\}$$

の基本対称式としてあらわされる.

対称式に関するNewton公式を利用すると, $j(q)$ のべき乗と $s_k(j)$ の間の関係式を導くことができる. したがってモジュラー多項式 $\Phi_l(X,j)$ を計算するには, $j(q), j^2(q), \ldots, j^{l+1}(q)$ のそれぞれの l 次の項までを求める必要がある. 詳細については[INY98]を参照されたい.

4.1.2 計算実験

例えば Φ_{113} を考える. このモジュラー多項式を求めるには, $j(q), j^2(q), \ldots, j^{114}(q)$ を求める必要がある. ただし $j^i(q)$ は113次の項まで求められれば十分なので, $j(q)\cdot j^{i-1}(q) \bmod q^{114}$ という剰余計算を繰り返すことになる. 実際に $j^{113} \times j \bmod q^{114}$ の計算を行った場合, Karatsuba法では13040秒, 三角積を用いた方法では6660秒で計算で, 三角積による効果は約2倍であった. これは $j(q)$ の係数が高次の項になるにつれて増大するため, 三角積のように不要な計算を行わない影響が大きかったためと思われる.

4.1.3 補足

モジュラー多項式は数学において研究の対象となる多項式である[La87]. このため古くから純粋な興味により計算実験が行われてきている. しかし近年では楕円曲線暗号における安全な楕円曲線を構成する際に, この多項式はデータベースとして利用されるようになっている.

われわれはモジュラー多項式の計算実験を行い, $l = 2, 3, \ldots, 119$ に対する $\Phi_l(X,j)$ を全て求めることに成功した. これはわれわれの知る限りでの世界記録である.

4.2 剰余計算

三角積は $f(x)g(x) \bmod x^m$ の高速計算法であったが, これを拡張することによって, もっと一般的な剰余 $f(x)g(x) \bmod n(x)$ を高速に計算することが可能となる.

4.2.1 逆多項式

多項式 $f(x)$ に対し

$$f^*(x) = x^{deg(f(x))} f(1/x)$$

で定義される関数 $f^*(x)$ を $f(x)$ の逆多項式という. $f(x)$ と $f^*(x)$ の次数は同じで, 係数の順序が逆になっている.

4.2.2 逆多項式を用いた剰余計算

$H-1$ 次多項式 $h(x)$ と $N-1$ 次多項式 $n(x)$ に対し $h(x) \bmod n(x)$ を計算することは,

$$h = nq + r \quad 0 \leq deg(r) < N-1$$

となる多項式 r を求めることに他ならず, q がわかればよい. ここで両辺の次数が $H-1$ 次であることに注意すると, ある定数 $\lambda > 1$ が存在して

$$h^* = n^* q^* + x^{H-N+\lambda} r^*$$

と表せる. ここで与えられた n に対し, $s^* n \equiv 1 \pmod{x^{H-N+1}}$ となる多項式 s をあらかじめ求めておいたとすると,

$$\begin{aligned} sh^* &\equiv sn^* q^* \pmod{x^{H-N+1}} \\ &= q^* \end{aligned}$$

となる. q, q^* は $H-N+1$ 次なので,

$$q(x) = (sh^* \bmod x^{H-N+1})^*$$

から q を求めることができ, 欲しかった r が求められる.

このアルゴリズムからわかる通り, 逆多項式を用いる際に $\bmod x^m$ のタイプの計算を繰り返すことになるので, 三角積の効果を期待することができる.

4.2.3 計算実験

例として $p = 2^{240} + 115$, $n(x) = x^3 + x + 1$ とした場合の, $x^p \bmod n(x)$ の計算実験を行った. Karatsuba法による計算時間は418秒だったのに対し, 三角積を用いた計算時間は336秒であった. $n(x)$ を変化させたところ, 三角積による平均的な効果は1割5分程度であった.

4.3 有限体 $GF(q)$ における演算

4.3.1 有限体

有限個の要素を持つ集合 K の要素間に和, 差, 積, 商が定義されているとき, K を有限体という. このとき, その要素の数は素数のべきになる. 要素の数が $q = p^m$ (pは素数) であるような有限体を $GF(q)$ と表す.

有限体 $GF(q)$ の元は, m 個の $GF(p)$ の組

$$(a_0, a_1, \ldots, c_{m-1}) \quad c_i \in GF(p)$$

と表される.

4.3.2 有限体における演算

有限体 $GF(q)$ における演算は次のようにして定義することができる. まず, $GF(q)$ の元

$$(a_0, a_1, \ldots, c_{m-1}) \quad c_i \in GF(p)$$

と, $GF(p)$ 係数多項式

$$a_0 + a_1 x + \ldots + c_{m-1} x^{m-1} \quad c_i \in GF(p)$$

を同一視する.

$GF(q)$ における和と差は, $GF(p)$ 上の多項式の和と差を計算することによって定義する. 一方, あらかじめ定めておいた m 次の既約多項式 $n(x)$ に対し, $n(x)$ で割った余りを考えることによって $GF(p)$ 係数多項式の積と商が計算できるので, その結果として $GF(q)$ における積と商も定義できる.

4.3.3 三角積の利用

有限体 $GF(q)$ における演算は, 多項式 $n(x)$ による剰余が基本となっている. このため前節で説明した方法を用いることによって, 有限体における演算の高速化が考えられる.

5 まとめ

多項式の剰余 $f(x)g(x) \bmod x^m$ の高速計算法を提案した. この方法はKaratsuba法の拡張になっている. 部分的に従来の高速積計算法が適用できる点と, 並列計算が可能である点が特徴的である.

実際の積計算を実現させる際には, 本方式を含めたしきい値のみきわめを行わなければならないが, これは課題である. また計算機では多項式と多倍長表現された整数の表現方法は本質的に同じでありので, 整数 F,G に対する積 FG の剰余 $FG \bmod M$ の高速計算への適用可能性についても考えていきたい.

参考文献

[GCL92] Keith O. Geddes, Stephen R. Czapor, George Labahn, "Algorithms for Computer Algebra", Kluwer Academic Publishers, (1992).

[Ito95] Hideji ITO, "Computation of the Modular Equation", *Proc.Japan Acad.*, 71A, pp.48-50(1995).

[INY98] 伊豆哲也, 野呂正行, 横山和宏, "モジュラー多項式の計算", SCIS98 予稿集, IEICE.

[Kar63] A.Karatsuba, "Multiplication of Multidigit Numbers on Automata", Soviet Physics doklady, 7, pp. 595-596(1963).

[Knu81] D.E.Knuth, "The Art of Computer Programming", Vol.2., Seminumerical Algorithms, 2nd ed., Addison-Wisley, (1981).

[La87] S.Lang, "Elliptic Functions (Second Edition)", *GTM* 112(1987), Springer-Verlag.