多項式の並列計算について*

丸山　清**

Abstract

In this paper, the parallel evaluation of general polynomials is considered. If an unlimited number of processors is availabbe, then for any given number of steps s, $s>1$, polynomials of degree as large as $C2^{s^{-\delta}}$ can be evaluated, where $C=\sqrt{2}$ and $\delta\approx\sqrt{2s}$. Also, polynomials of degree n can be evaluated in $\log_2 n+\sqrt{2\log_2 n}+0(1)$ steps. If only K, $0(K)=\sqrt{n}$, processors are available, then polynomials of degree n can be evaluated in

$$2n/K+\log_2 K+\sqrt{2\log_2 K}+0(1)$$

steps.

1. まえがき

n次多項式の計算については，以前より相当研究がなされており Pan[7], Winograd[8] その他により，一般のn次多項式の計算に，$2n$ 個の演算の必要なことが知られている．また，直列計算に関しては，Horner's rule が最適であることも知られている．しかしながら，もし任意の数の演算処理装置が与えられると仮定するならば，多項式の項の計算に $\lceil\log_2(n+1)\rceil$ のステップが必要で，さらに $(n+1)$ 個の項の加算に必要なステップ数が $\lceil\log_2(n+1)\rceil$ であるため，全体として $\lceil 2\log_2(n+1)\rceil$ ステップにより，n次多項式の計算が行なえる．また，これよりも少ないステップ（加算および掛算の処理時間を同一とみなし，それをステップと呼ぶことにする）にて計算可能なことは明らかである．

まず任意数の処理装置が与えられるものと仮定し，一般n次多項式の並列計算について考えてみる．一般n次多項式の並列計算について最もよく知られている二つの方法は，Estrin[3] の方法と Dorn[2] による K^{th} order Horner's rule であり，おのおの $\lceil 2\log_2(n+1)\rceil$，および $\lceil\log_2 n\rceil+\lceil\log_2(n+1)\rceil+1$ のステップの必要なことが知られている．最近 Muraoka[6] によって考え出された，tree method および，folding method に従えば，より少ないステップにより，n次多項式の計算の可能なことが知られているが，特に folding method については，少なくとも $1.46\log_2 n+0(1)$ のステップの必要なことが知られている．ただし，$0(1)$ はコンスタントを意味し，一般に，関数の大きさとして，もしも，$\limsup_{n\to\infty}(f(n)/g(n))=k$ となるコンスタント $k>0$ が存在する場合は，$f(n)=0(g(n))$ で表わす．さらに Munro と Paterson[5] により，いかなる計算方法を用いたとしても，一般n次多項式の計算には，少なくとも $\lceil\log_2 n\rceil+1$ のステップが必要であることが知られている．

一般n次多項式は式(1)のごとく表わされる．ただし，x, $a_0, a_1, \cdots\cdots, a_n$ は，代数的に独立な実数とする．

$$P_n(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots\cdots+a_1 x+a_0. \qquad (1)$$

一般に，n次多項式の計算に関して，次の二つの方法が考えられる．

(1) 与えられたn次多項式 $P_n(x)$ の計算に必要な最小のステップ数 s.

(2) 与えられたステップ数 s で，計算可能な多項式 $P_N(x)$ の最高次数 N.

前者を **primal problem** と呼び，後者をその **dual problem** と呼ぶことにする．

2. 並列計算の方法に関して

2.1 基本事項

式(1)に示す一般n次多項式は，式(2)のごとく書くことができる．

$$P_n(x)=\sum_{i=1}^{q-1} Q_{ni}(x)\,x^{mi}+P_m(x). \qquad (2)$$

* On the Parallel Evaluation of Polynomials

** Digital Computer Laboratory, Department of Computer Science, University of Illinois, Urbana-Champaign

ただし,

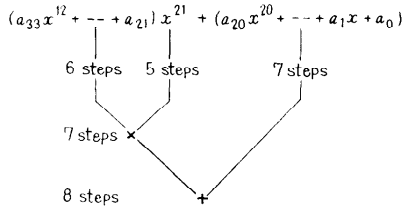
$$n=m+\sum_{i=1}^{q-1}(n_i+1),$$

$$m_i=m+i+\sum_{j<i}n_j.$$

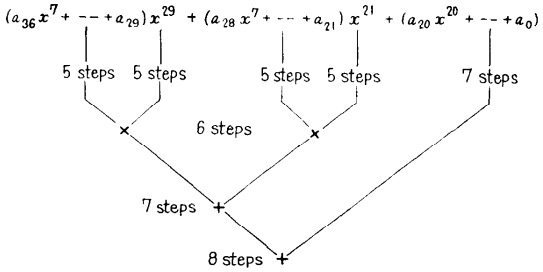
式(2)にて $Q_{n_i}(x)x^{m_i}$ を **segment** と呼び，右辺第1項全体を ***q*-cut** と呼ぶ．したがって一つの segment を s' ステップ以内にて計算するためには，$Q_{n_i}(x)$ および x^{m_i} の計算が $(s'-1)$ ステップ以内に終了することが必要である．ここで両方の条件を満足する segment を **consistent** と呼ぶ．

式(2)を $P_n(x)$ を計算する computation tree と考え，$\sum$ の項を **LHT**，$P_m(x)$ を **RHT** と呼ぶ．また，$N(s)$ という記号を用い，q-cut にて s ステップ以内に計算可能な多項式の最高次数を表わす．たとえば，$q=2$ として与えられる Muraoka の folding method は，次のごとく簡単に表わすことができる．$N(s-1)$ と $N(s)$ をもって $(s-1)$ ステップ，および s ステップに計算可能な多項式の次数とすれば，$(s+1)$ ステップに，計算可能な多項式の次数 $N(s+1)$ は，$N(s-1)+1+N(s)$ で与えられ，$N(s-1)+1$，および $N(s)$ は $P_{N(s+1)}(x)$ を計算する computation tree の LHT および，RHT に相当する．

例題として**図1**に，2個の異なった computation tree を示す．図1(a)は，2-cut により33次の多項式が8ステップで計算可能なことを示し，図1(b)は，3-cut により36次の多項式が8ステップで計算可能なことを示す．この二つの例からもわかるごとく，適切な q の選択により，与えられたステップ内で計算できる多項式の最高次数を高めることができる．



(a) The computation tree for Muraoka's folding method, a 2-cut at step 8, which evaluates a polynomial of degree 33.



(b) The computation tree by a 3-cut at step 8, which evaluates a polynomial of degree 36.

図 1 Examples of Computation Trees at Step 8.

2.2 Dual Problem の構成

一般に与えられた q の大きさが $2^t<q-1\leq 2^{t+1}$，$q\geq 3$ であるならば，q-cut を用いて $(s+1)$ ステップ内に計算できる多項式の次数は，式(3)で与えられる．

$$N(s+1)=N(s)+\sum_{i=0}^{M+L}(N(s-k-1+M-i)+1)A_i. \tag{3}$$

ただし，

$$q-1=\sum_{i=0}^{M+L}A_i.$$

ここで，A_i は $(s-k+M-i)$ ステップの計算時間を必要とする segment の数を示し，A_{M+L} は2の倍数とする．さらに $(s-k-L)$ は，LHT 内のある segmemt を計算するに必要な最小のステップ数を表わし，$(s-k+M)$ は，LHT 内のある segment を計算するに必要な最大のステップを表わすものとする．(詳細は Maruyama[4] 参照.)

定義 1

式(3)に相当する多項式の computation tree において，もし LHT 内のすべての segment が consistent であるならば，そのような computation tree のことを ***q*-consistent** であると呼ぶ．

定義 2

式(3)に相当する computation tree において，次の二つの条件を満たす tree を **LOF** computation tree と呼ぶ．

(1) binary balanced tree である．すなわち $q-1=A_0+A_1$.

(2) A_0 内の segment の x の最小累数が，A_1 内の segment の x の最大累数よりも大きい．

図2に一般の LOF conputation tree を示した．次に，computation tree に関して知[4]られている，三つの性質を記載する（証明略）．

補題 1

式(3)に相当する computation tree を考え，LHT が balanced computation tree でないものと仮定する．もしも，A_{M+j} 内の二つの segment を l レベル

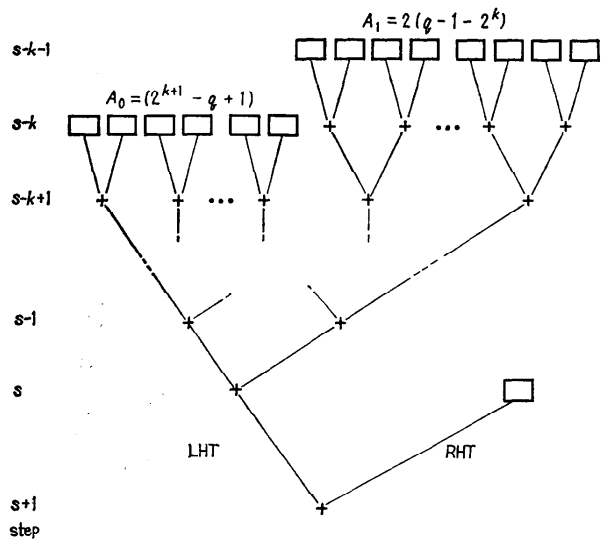


図 2 A General LOF Computation Tree. Where ▭ Denotes a Segment.

だけ computation tree の根に近づけるならば，式(3)で与えられる $N(s+1)$ の値を少なくとも $(\alpha^t-1)(2-\alpha)\times N(s-k-i-j)>0$ だけ増加することができる．ただし $0\leq j\leq L$．ここで，α を**増増率**と呼び，$N(s-k-j)+1$ と $N(s-k-1-j)+1$ の比で定義され，一般に $1<\alpha\leq 2$ である．例えば，Muraoka の folding method の場合，$\alpha(s)$ は $(\sqrt{5}+1)/2$ となる．

補題 2

式(3)に相当する computation tree において，その LHT を LOF でない computation tree とする．この場合 LHT は balanced tree でもよい．そして，そのような computation tree にて計算できる多項式の最高次数を $\tilde{N}(s+1)$ で示せば，次のことがいえる．すなわち，条件 $N(s+1)>\tilde{N}(s+1)$ かつ $q>\tilde{q}$ を満足する q-consistent な LOF computation tree が少なくとも一つ存在する．

補題 3

式(3)に相当する computation tree が，balanced computation tree であり，かつ q-consistent であるとすれば，同じ q に対する LOF computation tree も q-consistent である．

補題1，補題2は balanced tree により，$(s+1)$ ステップ内に計算できる最高次数は，balanced tree でない computation tree で与えられるそれよりも大きいことを意味し，補題3は，LOF computation tree が一般の balanced tree よりも，多項式の並列計算に適していることを意味する．したがって，LOF computation tree を，多項式の並列計算に使用する．

2.3 アルゴリズムに関して

定理 1

LOF computation tree が q-consistent であるとする．すなわち式(4)の不等式が保たれたとすると式(5)が成立する．

$$s-k-2\geq\lceil\log_2(N(s)+(2q-2^{k+1}-3)\times(N(s-k-2)+1)+1)\rceil. \quad (4)$$

$$N(s+1)=N(s)+2(q-1-2^k)(N(s-k-2)+1)+(2^{k+1}-q+1)(N(s-k-1)+1). \quad (5)$$

証明

LOF computation tree および，q-consistency の定義により，定理が明らかとなる．

系 1

LOF computation tree が q-consistent であると仮定すれば，q-cut で与えられる $N(s+1)$ は，$(q-i)$-cut で与えられる $N(s+1)$ よりも大きい．ただし，i の値は $1\leq i\leq q-3$ とする．

証明

q-consistent な LOF computation tree は，明らかに $(q-i)$-consistent である．したがって，q-cut により与えられる $N(s+1)$ の値が $(q-1)$-cut により与えられる $N(s+1)$ の値よりも大きいことを示せば十分である．

系1より，$N(s+1)$ の値をより大きくするためには，q-consistent な LOF computation tree の q の値を，より大きくすることであるといえる．この方法は，Muraoka の folding method の一般化であり，**multi-folding method** と呼ぶ．(これと似た方法が，Munro と Paterson[5] により，ほとんど同時に，かつ独立に考え出されている．) multi-folding method が満たされなければならない q-consistent の条件，式(4)を利用して，$(q+1)$-consistent な LOF computation tree を得ることができる．この方法を **modified multi-folding method** と呼び，multi-folding method よりもすぐれていることが知られている[4]．

表1に，与えられたステップ s 以内で計算できる多項式の最高次数を，Muraoka の folding method，および multi-folding method についてしめした．さら

表 1 Degrees of Polynomials Which Can Be Evaluated in Given Steps, Those Degrees Marked With * Have Been Found By Munro And Paterson.

s \ $N(s)$	Folding Method	Multi-folding Method	q	Best Known	q	Lower Bound
1	0	0	—	0*	—	
2	1	1	2	1*	2	2
3	2	2	2	2*	2	3
4	4	4	2	4*	2	4
5	7	7	2	7*	2	4
6	12	12	2	12*	2	5
7	20	20	2	21*	—	6
8	33	36	3	37*	3	7
9	54	62	3	63*	3	7
10	88	104	3	107*	3	8
11	143	183	4	187*	—	9
12	232	320	4	327	4	10
13	376	572	5	575	5	11
14	609	992	5	1,007	5	11
15	986	1,728	5	1,759	5	12
16	1,596	3,059	6	3,119	6	13
17	2,583	5,489	7	5,575	7	14
18	4,180	9,767	7	9,895	7	15
19	6,764	17,454	8	17,703	8	16
20	10,945	31,286	9	31,783	9	16
21	17,710	55,915	10	56,743	9	17
22	28,656	101,095	11	102,111	11	18
23	46,367	182,875	12	185,047	12	19
24	75,024	330,839	13	335,031	13	20
25	121,392	602,873	15	607,423	14	21
26	196,417	1,096,807	16	1,109,143	16	22
27	317,810	1,991,463	17	2,017,047	17	22
28	514,228	3,619,735	18	3,662,215	18	23
29	832,039	6,603,699	20	6,680,511	20	24
30	1,346,268	12,071,699	22	12,216,343	22	25
31	2,178,308	22,129,325	24	22,373,607	24	26
32	3,524,577	40 738,153	27	41,071,247	26	27
33	5,702,886	75,027,401	29	75,758,895	29	28
34	9,227,464	138,391,057	32	139,896,991	32	29
35	14,930,331	254,222,609	33	257,087,903	33	29

に，現在知られている最高の次数も加えた．ここで，＊印の付いている次数は，Munro と Paterson[5] により与えられた次数であり，これらの値と modified multi-folding method を使用して，12 ステップ以後に計算できる多項式の最高次数を求めた．また，multi-folding method で与えられる次数の多項式の計算に絶対必要なステップ数，すなわち，理論的な **lower bound** をも示した．

表 1 からも知れるとおり，$N(s)$ と $N(s-1)$ の比として定義される増幅率 $\alpha(s)$ は，multi-folding method の場合，s の増加とともに，2 に近づく，また 7 ステップ以後になると，multi-folding method が folding method に比べて，著しくすぐれていることがわかる．

3. 多項式の並列計算における BOUNDS に関して

定理 2

もしも，任意の演算処理装置が与えられるものと仮定するならば，次のことがいえる．

$$N(D_r)\geq 2^{D_{r-1}}.$$

ただし，$D_r=r(r+1)/2+i$ であり，$r\geq 2$，$r-1\geq i\geq -1$ とする．

証明

D_r ステップ以内に，次数 $2^{D_{r-1}}$ の多項式の計算ができることを，帰納法を用いて証明する．ここで，$i=0$ の場合の証明は，Munro と Paterson[5] によっても与えられている．

$r=2$ の場合は $N(3+i)\geq 2^{1+i}$ が，$1\geq i\geq -1$ に対して成立する．$r\leq k$ に対して定理が成立するものと仮定し，$r=k+1$ の場合にも，定理が成立することを示す．

帰納法の仮定より，D_k ステップにおいて，次数が，$2^{D_{k-1}}$ 以下である任意の多項式が計算でき，かつ，次数 2^{D_k} 以下である任意の x の累乗をも計算できる．したがって，次数 2^{D_k} 以下の任意の多項式は，次式のごとく表わされる．

$$P(x)=\sum_{j=0}^{2^{k-1}} Q_j(x)x^{j2^{D_{k-1}}}.$$

ただし，$Q_j(x)$ は次数 $2^{D_{k-1}}$ 以下の任意の多項式を示す．上式より明らかなように，$P(x)$ の最高次数は 2^{D_k} となり，D_{k+1} ステップで計算できることがわかる．

ここで興味のあることは，Brent[1] の 758 頁に書かれている式(4)を書き替えることにより，次数 $2^{r(r-1)/2}-1$ の多項式が，$r(r+1)/2$ ステップ内に計算できると解釈できる．

定理 2 より，次の性質は明らかである．

系 2 (dual property)

与えられた 2 以上のステップにおいて，次数 $C2^{s-\delta}$ の多項式が計算できる．ただし，$C=\sqrt{2}$ そして $\delta\approx\sqrt{2s}$.

次に Primal Problem である，次数 n の多項式の計算に必要な，最小のステップ数について考える．定理 2 より次の不等式が導ける．ただし，$T_K(n)$ は，すべてのアルゴリズムを考えた場合，n 次多項式を計算するに必要な最小のステップ数に相当し，

$$T_*(n)=\min_K T_K(n)$$

は，任意の数の演算処理装置が与えられる場合に相当する．

$$T_\infty(n)\leq \log_2 n+\sqrt{2}\log_2 n/\sqrt{\log_2 n-i}+0(1)$$

ただし，$\log_2 n=r(r-1)/2+i$，$r\geq 2$，$r-1\geq i\geq -1$．

したがって，Munro と Paterson の得た定理と同じ定理3が導ける．

定理 3 (primal property)

$$T_\infty(n)\leq \log_2 n+\sqrt{2\log_2 n}+0(1).$$

定理 4

もしもK個の演算処理装置が与えられたものと仮定すれば，次の不等式が成立する．ただし，$0(K)=\sqrt{n}$．

$$T_K(n)\leq 2n/K+\log_2 K+\sqrt{2\log_2 K}+0(1).$$

証明

一般の，n次多項式を次のように書く．ただし，A_iは，次数 $n/(K-1)$ の多項式であり，$X=x^{n/(K-1)}$ とする．

$$P_n(x)=A_0(x)+A_1(x)X+\cdots\cdots+A_{K-2}(x)X^{K-2}$$

まず，$A_0, A_1, \cdots\cdots, A_{K-2}$ を，Horner's rule を用いて，$2n/(K-1)$ ステップで計算し，同時に X をも計算する．そして，multi-folding method を用い，さらに $\log_2(K-2)+\sqrt{2\log_2(K-2)}+0(1)$ ステップで $P_n(x)$ の計算を終了する．したがって全体として，

$$2n/(K-1)+\log_2(K-2)+\sqrt{2\log_2(K-2)}+0(1)$$

ステップ必要となり，定理が導ける．

4. むすび

Muraoka の folding method の一般化として，multi-folding method と呼ばれる多項式の computation tree を考えてみた．この方法により，大きな次数の多項式に関して，理論的に求められるステップ数(下限)に近いステップ数で，多項式の計算ができることが知れた．

与えられたステップ数内において計算できる多項式の，現在知られている最高の次数を，表1に示した．しかし，著者の知識ではあるが，Munro と Paterson[5] により考えだされた特質の computation tree を考慮すれば，15ステップ以上において計算できる多項式の次数 $N(s)$ を，ほんの少し高めることが可能である．

本論文のおもなる結果を，定理2, 3, 4に示したが，これらの結果は，本論文においては考慮しなかったが，割算をも含む多項式の計算に拡張できるものと思われる．しかし，本論文で示した multi-folding method は，並列計算を許す多項式の computation tree のモデルとして，現在知る限り最も適切な方法であり，たとえ限られた数の演算処理装置しか与えられなくとも，下限に非常に近いステップ数により，多項式の計算が行なえる．

謝辞 本研究課題を与えて下さったイリノイ大学 Computer Science 教授 Dr. D. Kuck ならびにご援助いただいた Dr. Y. Muraoka (現在電気通信研究所勤務中) に深謝する．

参考文献

1) R. Brent: "On the Addition of Binary Numbers," IEEE Trans. on Computers, vol. C-19, pp. 758—759 (August 1970).
2) W. Dorn: "Generalizations of Horner's Rule for Polynomial Evaluation," IBM Journal of Resarch and Developmeut, vol. 6, pp. 239—245 (April 1962).
3) G. Estrin: "Organization of Computer Systems the Fixed plus Variable Stucture Computer," Proc. of Western Joint Computer Conference, pp. 33—40 (May 1960).
4) K. Maruyama: "Parallel Methods and Bounds of Evaluating Polynomials," DCS Report No. 437, University of Illinois at Urbana-Champaign, (March 1971).
5) I. Munro & M. Paterson: "Optimal Algorithms for Parallel polynomial Evaluation," IBM Research Reort RC 3497, (August 1971). (It also appears in the IDEE-SWAT Proceedings of Oct. 1971.)
6) Y. Muraoka: "Parallelism Exposure and Exploitation in Programs," Ph. D. Thesis, Department of Computer Science, Universty of Illinois at Urbana-Champaign, pp. 33—41 (1971).
7) V. Pan: "Methods of Computing Values of Polynomials," Russian Mathematical Surveys vol. 21, pp. 105—136 (January-February 1966).
8) S. Winograd: "On the Number of Multiplications Required to Compute Cetrain Functions," Proceedings of National Academy of Sciences 58, pp. 1840—1842 (1968).

(昭和47年2月7日受付)