ある20480次代数方程式の係数の計算に対する多倍長演算の並列化

益本博幸[†] 藤野清次[††]
小野令美[†††] 児島彰[††]

Chebyshevの数値積分公式の分点を決める代数方程式の次数Nを無限に大きくしたとき，その係数が漸近する式が1981年森口によって発表されている．しかし，代数方程式の次数Nについては2048次までしか確認されていない．そこで，本論文の目的の1つは，その代数方程式の次数Nを計算機環境の許容限度まで大きくして，係数が漸近する式に計算で求めた値がどのくらい近いかを数値的に確認することにある．この係数の計算では桁落ちが起きるため，本研究では演算をすべて多倍長演算で行った．さらに，一般に多倍長演算は時間がかかるので，多倍長演算の並列化を試みることも本論文のもう1つの目的である．数値実験には，PC（Pentium Proプロセッサ200MHz）および分散メモリ型並列計算機：IBM SP2（48プロセッサ）を使用した．多倍長演算に対するいくつかの並列化手法を検討し，プログラムを実装した結果，次数が20480次まで上記の漸近式の正当性と並列計算の有効性が確認された．

Parallelisim of Multi-precision Arithmetic for Computation of Coefficients of Algebraic Equation of 20480 Degrees

HIROYUKI MASUMOTO,[†] SEIJI FUJINO,[††] HARUMI ONO[†††] and AKIRA KOJIMA[††]

In this paper we verify numerically a certain asymptotic estimation derived by S. Moriguti for the coefficients of algebraic equation which stem from N-points Chebyshev Integration formula. Since a cancellation in floating-point arithmetic is significant for calculating precisely of this algebraic equation of very high degrees, all computations were carried out with multiple precision arithmetic. However, it needs much computation time. Therefore some parallelization methods were investigated from the viewpoint of efficiency on distributed memory parallel computer. As a result, soundness of the asymptotic estimation and effectiveness of parallel computation were confirmed.

1. はじめに

Chebyshevの数値積分公式[3),7)]

$$\int_{-1}^{1} f(x)dx \approx \frac{2}{N}\sum_{i=1}^{N} f(x_i) \qquad (1)$$

の分点を零点に持つ方程式を$F_N(z)$とおく．このとき$F_N(z)$は

$$F_N(z) = a_0 z^N + a_2 z^{N-2} + \cdots + \begin{cases} a_{N-1}z & (N:\text{奇数}) \\ a_N & (N:\text{偶数}) \end{cases} \qquad (2)$$

となり，各項の係数a_{2k}は，

$$\begin{cases} a_0 = 1 \\ a_{2k} = -\dfrac{N}{2k}\displaystyle\sum_{j=1}^{k} \frac{1}{2j+1} a_{2(k-j)} \\ (k = 1, 2, 3, \cdots, \frac{N}{2}) \end{cases} \qquad (3)$$

で表される．ただし奇数項の係数は0である．代数方程式$F_N(z)$の次数Nを$N \to \infty$に近づけたとき，係数を漸近的に表す式が森口によって与えられている[11)]．本論文の目的は，この代数方程式の次数Nをできるだけ大きくして式(3)の計算値が漸近値に近づく様子を数値的に確かめることにある．計算には，PC（Pentium Proプロセッサ）および分散メモリ型

† 株式会社アイテック阪神
Ai-tech Hanshin Ltd.
†† 広島市立大学情報科学部
Faculty of Information Sciences, Hiroshima City University
††† 元千葉大学工学部
Former Chiba University

並列計算機：IBM SP2（48プロセッサ）を使用し，前者ではC++言語，後者ではFortran90を用いて多倍長演算を行った．

高次の代数方程式に関する先行研究としては，共著者の1人小野による研究[12),13)]が知られている．そこでは，2048次までのものについてDurand-Kerner法を用いて全根を求め，$N \to \infty$ の極限で解の並ぶ曲線にそれらの根が近づいていく様子と方程式の係数の大きさの様子が示されている．また，解を求めるために特別に用意された多倍長演算システムを使う必要があったこと，根を求める計算および係数を求める計算(3)の過程において大きな桁落ちが発生することなどが報告されている．たとえば，$N=2048$ 次のときは，3〜5桁の精度の解を求めるのに必要な桁数の係数を求める計算には840桁（およそ $\frac{2\times N}{5}$ に相当）の桁数が必要であった．本報告では代数方程式の係数 a_k の計算のみなので，係数の有効桁数を5桁程度でよいことにすれば，計算に必要な桁数はもっと少なくて済む．たとえば，次数が同じ $N=2048$ のとき演算桁数を410桁（この桁数は $\frac{1.0\times N}{5}$ に相当する）にとり，これとその2倍の演算桁数で計算した結果と比較した結果，有効数字が6桁あることが確認できた．このような検討をもとに，次数 N の大きさに応じて演算桁数を決める基準を見い出した．本報告では，次数が20480次までの係数 a_k について上述の森口による漸近式の妥当性を確かめることにする．

さらに，このような超高次の代数方程式の係数を精度良く求めるには多倍長演算が必要になる．しかし，一般に多倍長演算には多くの演算時間が必要である．そこで，分散メモリ型並列計算機による並列計算を行った．そのためのいくつかの並列化の方法について検討を行い，その実装を行った．

2. チェビシェフによる2つの積分公式

まず，有限，または無限区間 (a,b) における積分

$$\int_a^b w(x)f(x)dx = \sum_{i=1}^{N} w_i(x_i)f(x_i) \qquad (4)$$

について考える．ここで $w(x)$ は重み関数とする．合計 $2N$ 個のパラメータ：重み w_i と分点 x_i のとり方によりいろいろな公式が生まれる．

2.1 チェビシェフ・ガウスの積分公式

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}dx \qquad (5)$$

のように，積分区間 $[-1,1]$ で，重み関数 $w(x)=\frac{1}{\sqrt{1-x^2}}$ と与え，分点をチェビシェフ多項式 $T_N(x)$ の零点にとる公式はチェビシェフ・ガウスの積分公式（則）と呼ばれる[1),14)]．ここで，N 次のチェビシェフ多項式 $T_N(x)$ の零点は，

$$x_j = \cos\frac{(2j-1)\pi}{2N}, \quad (j=1,2,\cdots,N) \qquad (6)$$

となる．また，チェビシェフ多項式 $T_N(x)$ は次の漸化式を満たす．

$$T_{N+1}(x) - 2xT_N(x) + T_{N-1}(x) = 0. \qquad (7)$$

ここで，$T_N(x)$ については，最高次の係数は 2^{N-1}，以下交互に正負の符号をとり，最低次の項は N が偶数 p ならば $(-1)^p$，奇数 $2p+1$ ならば $(-1)^p(2p+1)x$ になることなどいくつもの事実が知られている[9)]．

2.2 チェビシェフの数値積分公式

一方，N 個の重みを一定値 $w_i=(b-a)/N$ にとり，分点 x_i $(i=1,\cdots,N)$ を定めるのが，本研究で取り扱うチェビシェフの数値積分公式（Chebyshev quadrature）である[1),10)]．積分区間 $[-1,1]$ の場合には，積分公式は

$$\int_{-1}^{1} f(x)dx \approx \frac{2}{N}\sum_{i=1}^{N} f(x_i) \qquad (8)$$

のように表される．また，その分点を零点に持つ方程式の係数は式(3)のように表される．前に述べたチェビシェフ多項式の係数の場合は，式(7)のようにきれいな漸化式の形で表されため，これらの係数の性質についてもよく調べられている．

一方，本研究で取り扱うチェビシェフの数値積分公式では，分点を定める多項式を式(7)のような漸化式で表すことが難しい．そのため，次数 $N=8$ と $N\geq 10$ のとき，方程式の根には実根が N に達しなくて共役複素数の根が混じると述べられているが[10)]，後者の場合に，理論的にあるいは数値的に確かめられていない，などまだ分からないことも多い．

3. チェビシェフの数値積分公式における係数 a_k の漸近式

次数を N とし，$\alpha=k/N$ をおく．N が大きいとき係数 a_k は次のように表されることが森口により示されている[11)]．奇数の k に対しては $a_k=0$．偶数の k に対しては，N が大きいとき漸近的に α がほぼ0.75位までは，

$$v\cdot \operatorname{arccot} v = 1-\alpha+\frac{1}{N} \qquad (9)$$

を満たす v と α を用いて，

表1 指標 k が偶数のときの漸近式の数値一覧表
Table 1 Asymptotic estimation (10) when index k is even.

α	v	arccot v	$\frac{\sqrt{1+v^2}}{e^{\alpha}v^{1-\alpha}}$	$\log_{10}\frac{\sqrt{1+v^2}}{e^{\alpha}v^{1-\alpha}}$
.0135	5.00	0.1974	1.0282	0.0121
.0352	3.00	0.3218	1.0578	0.0244
.2151	1.00	0.7854	1.1405	0.0571
.3823	0.60	1.0304	1.0909	0.0378
.5244	0.40	1.1903	0.9858	−0.0062
.7258	0.20	1.3734	0.7673	−0.1150

$$a_k = (-1)^{k/2}\left(\frac{\sqrt{1+v^2}}{e^{\alpha}v^{1-\alpha}}\right)^N \cdot\sqrt{\frac{2}{N\pi}}/\sqrt{\frac{1-\alpha}{v^2}-\frac{1}{1+v^2}} \qquad (10)$$

となる．したがって，N が大きいときは $(\frac{\sqrt{1+v^2}}{e^{\alpha}v^{1-\alpha}})^N$ が $|a_k|$ の主要な部分となる．一方，α が0.75付近の臨界値を超えるときは

$$a_k \approx -\left(\frac{2}{e}\right)^N \qquad (11)$$

である．**表1** に N=2048のときの $|a_k|$ が最大となる付近での漸近式(10)の中の主な値を示す．

4. 演算桁数と有効桁数の関係

4.1 演算桁数を変えたときの係数の値について

図1 に次数 N=256のとき，演算桁数Mを，M=50桁 (50/256≒0.195)，M=60桁 (60/256≒0.234)，M=70桁 (70/256≒0.273)，そしてM=102桁 (102/256≒0.398) と変化させたときの係数 a_k の絶対値をプロットしたものを示す．横軸には係数 a_k の番号：k をとり，縦軸は $\log_{10}|a_k|$ の大きさを表す．図から，上の2本の線で表されたM=50桁とM=60桁のときは，桁落ちの影響を受けて計算値は解と比べてオーダまで値が違うこと，一方，M=70桁のときは桁数を十分にとった図中の最下線のM=102桁の線と重なって見えること，などが分かる．また，解の精度もM=70桁のときの結果はM=102桁のそれと比較して有効数字8桁目まで完全に一致していた．また，**表2** に次数 N=1024のときの係数 a_{1024} の値について，演算桁数Mを M=210桁 (210/1024≒0.205) から M=225桁 (225/1024≒0.220) まで変化させ，それらと M=410桁 (410/1024≒0.400) のときの結果とを比較（表中の下線部が値が一致したところ）したものを示す．この表から，桁数を増加させるに従って有効桁数がだんだんと大きくなり，有効桁数が6桁ぐらいであれば，その比率が0.210のときの演算桁数 (M=215桁) で十分であることが分かる．

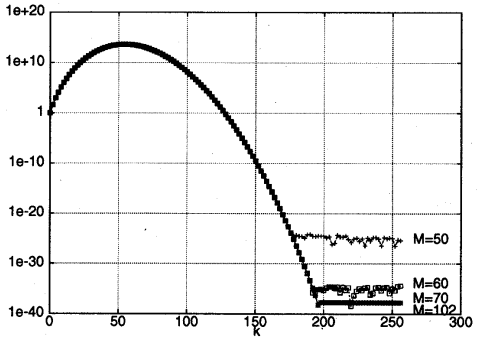

図1 演算桁数Mを変えたときの各係数 a_k $(k=1,\cdots,N)$ の絶対値（縦軸の目盛は $\log_{10}|a_k|$）
Fig. 1 Absolute value of coefficients a_k $(k=1,\cdots,N)$ for some digits of computation with the vertical axis of $\log_{10}|a_k|$.

表2 次数 N=1024のときの係数 a_{1024} の値
Table 2 Values of coefficient a_{1024} at $N=1024$.

演算桁数	比率	有効桁数	係数 a_{1024} の値
210	.205	1.50	$0.1435\ 3844 \times 10^{-140}$
215	.210	6.29	$0.1391\ 5320\ 6352 \times 10^{-140}$
220	.215	11.8	$0.1391\ 5313\ 5615\ 1877 \times 10^{-140}$
225	.220	16.0	$0.1391\ 5313\ 5615\ 4347 \times 10^{-140}$
410	.400	—	$0.1391\ 5313\ 5615\ 4347 \times 10^{-140}$

表3 係数 a_N の計算値の有効桁数が6桁以上になる演算桁数の次数に対する比率
Table 3 Ratios of necessary digits vs. degree of polynomial for calculation of a_N with more than 6 digits of accuracy.

次数 N	演算桁数M	比率
256	65	0.254
512	115	0.224
1024	215	0.210
2048	410	0.200
4096	800	0.195

また，**表3** は，各次数 N ごとに，演算桁数Mを変化させて係数 a_N の有効桁数が6桁以上あるときの演算桁数の次数に対する比率を調べたものである．この表から，次数を大きくするに従ってその比率は単調に小さくなっていくことが分かる．この結果から，次数をもっと大きくしたときでも必要な演算桁数は $\frac{1.1\times N}{5}$ 桁程度で十分であることが分かる．

4.2 十分な演算桁数がある場合

前の節での考察を基に，次数 N=256のときは演算

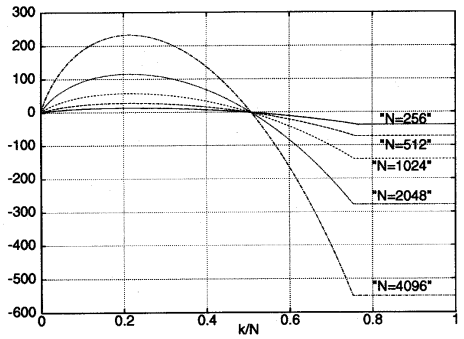

図2 演算桁数が十分あるときの係数 a_k の絶対値（縦軸の目盛は10のべき乗の値）

Fig. 2 Absolute value of coefficients a_k with sufficient digits of computation with the vertical axis of a power of 10.

桁数を70桁 (70/256≒0.274) に，次数がそれ以上大きいときは，次数 N ごとに演算桁数を $\frac{1.1\times N}{5}$ 桁とって計算した結果を**図2**に示す．ただし，横軸の目盛の0が係数 a_0 に，同じく目盛の1が係数 a_N に対応している．また，縦軸には $\log_{10}|a_k|$ をとり，目盛は10のべき乗の値を表す．図の左側の上に凸の部分で，5本の線のうち一番下の実線が256次のとき，そして一番上にある1点鎖線が4096次のときを各々表す．この図から，すべての次数の場合において α=k/N (x 軸) が0.75付近になるまで減少し，桁落ちの影響がないことが分かる．

4.3 漸近評価式と計算値との比較

図2から，$\alpha = k/N$ がおよそ0.75までは式(10)の値に，それよりも大きいところでは式(11)の値に近づいていくことが分かる．これをはっきり示すために，縦軸方向の値を $\frac{1}{N}\log_{10}|a_k|$ に変更したものを**図3**に示す．図中で"true"のグラフは漸近評価式(10)で $\frac{1}{N}\log|a_k|$ の $N\to\infty$ の極限の曲線 $(0\le\frac{k}{N}\le 0.75)$ である．方程式の次数が2048次から20480次まで $N=2^n$ $(n=11,12,13,14)$ と $N=20480$ に対する6種類のデータの曲線を表示した．$\alpha=0.75$ まではこれらの線が重なって識別不可能なほど値が近いことが分かる．

さらに，$\alpha=0.75$ よりも大きなところの6つのデータの差異を細かく見るために拡大図を**図4**に示す．縦軸方向の目盛は，図3と同様に，$\frac{1}{N}\log_{10}|a_k|$ とする．実線が2048次のとき，1点鎖線が20480次のときの結果を示している．これらの図からも，(i) α の値が0.75よりも小さいとき計算値は漸近式(10)の値に次数が高くなればなるほど近づくこと，さらに (ii) α の値が0.75よりも大きいときは，高次になればなるほど計算値は漸近式(11)に近づくこと，などが分かる．

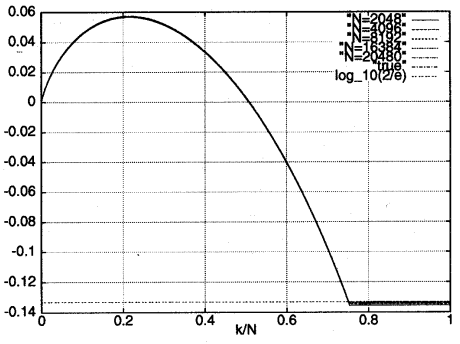

図3 縦軸方向の値を $\frac{1}{N}\log_{10}|a_k|$ にした場合の漸近式の極限の曲線と計算値

Fig. 3 Asymptotic estimation (10) and computational values with the vertical axis of $\frac{1}{N}\log_{10}|a_k|$.

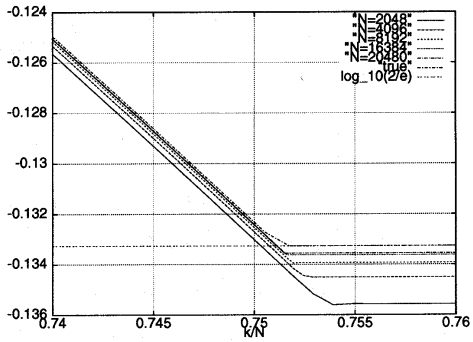

図4 $\alpha=0.75$ 付近の拡大図（縦軸方向の目盛は $\frac{1}{N}\log_{10}|a_k|$）

Fig. 4 Enlarged graph near at $\alpha=0.75$ with the vertical axis of $\frac{1}{N}\log_{10}|a_k|$.

5. PCによる計算

計算の効率を上げるため，式(3)に現れる $(2j+1)$ による割算を乗算：$\frac{1}{2j+1}$ に書き換えて，高速化を図ったので以下その結果を報告する．

5.1 除算の乗算への書き換え

この書き換えを行ったとき，作業領域として，新たに $N/2$ 個の配列（メモリ）が必要になり，演算可能な次数 N が半減することが予想される．各次数における CPU 時間を**表4**に示す．使用した計算機（PC）は Pentium Pro プロセッサ（200 MHz）である．使用言語は C++言語である．多倍長演算は平山による MPPACK を改良して使用した[4),5)]．この表から，速度がおよそ4.4〜7.7倍も向上し大きな効果があることが分かった．そこで，以下の計算では除算を乗算に

表4 除算を乗算に置き換えたときのCPU時間（単位：秒）
Table 4 CPU times of multiplication in place of division.

次数	除算	乗算	比（除算/乗算）
256	2.15	0.28	7.67
512	13.6	1.97	6.93
1024	105.8	18.9	5.59
2048	1027.0	232.6	4.42
4096	18157.0	3253.0	5.58

表5 高次のときのCPU時間（単位：時間），演算桁数と必要なメモリ容量
Table 5 CPU times, digits of computation and necessary memory in case of polynomial with high degrees.

次数	CPU時間	演算桁数	メモリ（Mbyte）
4096	0.91	905	2.96
8192	13.3	1805	11.8
16384	213.4	3610	47.3
20480	561.1	4520	74.1

置き換えてすべての計算を行った．ただ，このように多倍長演算の除算を乗算に置き換えないでも，ほぼ乗算と同じ程度の速さで計算できることがOzawa[15]の研究で知られている（謝辞参照）．また，共著者の1人小野[16]により除算の高速化への試みがなされているが，本研究では，上のような簡単な置き換えだけで十分な高速化が達成できたので，除算の特別なルーチンは作成しなかった．さらに大きな次数の計算を行うときは，計算時間短縮のため考慮すべきであると思われる．4096次以上のときのCPU時間（単位：時間），演算桁数そして必要なメモリ容量を**表5**に示す．

たとえば，16384次では213時間（約9日），20480次では561時間（約23日）も計算に時間を要した．これ以上の大きな次数のときの計算は計算機環境の制約と，PCの搭載メモリ容量（96 Mbyte）を上回るため実行できなかった．

6. 多倍長演算の並列化について

PCでは多くの計算時間が必要であったので，並列計算機による多倍長演算の高速化を図った．並列化の方法としては，2つの方法を検討し，それらを並列計算機上に実装した．ここでは，第1の方法を**内部加算法**（Inner Summation，略してIS法），そして第2の方法を**外部加算法**（Outer Summation，略してOS法）と呼ぶことにする．また，前者のIS法ではメモリ分割の方法として，**サイクリック分割**と**ブロック分割**と呼ばれる2種類の分割の方法についても検討した．

6.1 IS法について

IS法は，式(3)の係数 a_{2k}：

$$\sum_{j=1}^{k} \frac{1}{2j+1} a_{2(k-j)} \tag{12}$$

の計算を各プロセッサごとに小さく分割して割り当て，その部分和をプロセッサごとに計算し，次に各部分和を通信を使って集計して全体の総和を求め，そして次の係数 $a_{2(k+1)}$ を求める手順を繰り返す方法である．このとき，式(12)はプロセッサの台数と各プロセッサに割り当てる係数も含んだ次の形の式で表すことができる．ただし，$k = 1, 2, \cdots, \frac{N}{2}$ とする．

$$\sum_{i=1}^{p} \sum_{j=1}^{\gamma} \frac{1}{2(p(j-1)+i)+1} a_{2(k-(p(j-1)+i))}. \tag{13}$$

ここで，$\gamma = (k + (p - i))/m$ は整数で小数点以下切捨て，i はプロセッサの番号，p はプロセッサの台数をそれぞれ表すものとする．たとえば，プロセッサが4台のときプロセッサごとの処理は次のようになる．指標 i には各プロセッサの番号を代入し指標 p には4を代入する．

$$\begin{aligned} &\sum_{j=1}^{(k+3)/4} \frac{1}{2(4(j-1)+1)+1} a_{2(k-(4(j-1)+1))} \\ &+ \sum_{j=1}^{(k+2)/4} \frac{1}{2(4(j-1)+2)+1} a_{2(k-(4(j-1)+2))} \\ &+ \sum_{j=1}^{(k+1)/4} \frac{1}{2(4(j-1)+3)+1} a_{2(k-(4(j-1)+3))} \\ &+ \sum_{j=1}^{k/4} \frac{1}{2(4(j-1)+4)+1} a_{2(k-(4(j-1)+4))} \end{aligned} \tag{14}$$

上の式中の4つの項の和は，上から順番に番号1，2，3，4が付けられたプロセッサに割り当てられた計算式に対応している．IS法の計算手順を**図5**に示す．この図は係数 a_{20} を求めるときのサイクリック分割によるIS法（以下，IS-B法と略す）の計算の動きの概略を表す．

逐次処理（並列化せず）の処理手順は次の(a)，(b)のように表せる．

[逐次処理の処理手順]

(a) S_j について演算を行う．このとき，S_j の演算結果は順次変数SUMに加算される．

(b) すべての S_j について計算が終了した時

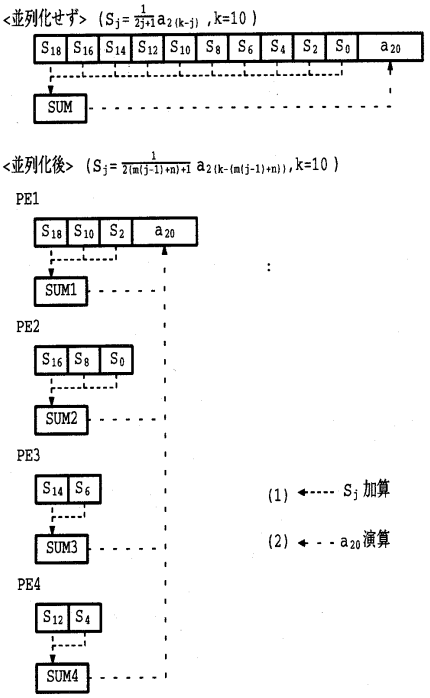

図5 IS-C法（サイクリック分割）の計算の動きの概略（4プロセッサ使用）

Fig. 5 Outline of procedure of the Inner Summation method of cyclic partitioning with four processors.

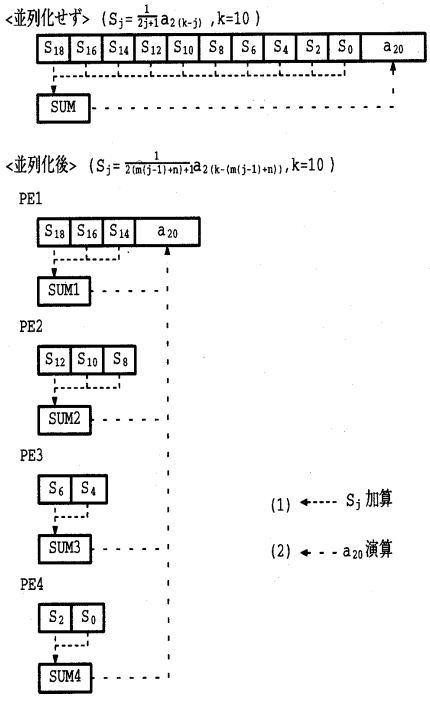

図6 IS-B法（ブロック分割）の計算の動きの概略（4プロセッサ使用）

Fig. 6 Outline of procedure of the Inner Summation method of block partitioning with four processors.

点で，変数SUMから係数 a_{20} の値が求まる．

一方，並列化したときの処理手順は以下の(a)～(d)のように表せる．

[並列化したときの処理手順]

(a) 各プロセッサは割り当てられた S_j のみ演算を行う．このとき，S_j の演算結果は各プロセッサごとに用意された変数（SUM1～SUM4）に加算される（図5の中で短破線の矢印で表す）．

(b) 割り当てられた演算がすべて終了した時点で，プロセッサ：PE2～PE4は変数の値をPE1に送り，PE1が代表して係数 a_{20} の値を求める（図5の中で長破線の矢印で表す）．

(c) プロセッサPE1が演算を行っている間，他のプロセッサはPE1の演算が終了するまで待機している．PE1は演算が終了した時点で，求まった係数 a_{20} の値を他のプロセッサに送る．

(d) 他のプロセッサは，プロセッサPE1から値が送られてきた時点で待機を解除し，次の係数の計算に移る．

さらに，ブロック分割をしたときのIS法（以下，IS-B法と略す）の計算の動きの概略を**図6**に示す．この方法は，各プロセッサに割り当てる計算式だけが違うだけで計算手順はサイクリック分割のIS-C法とまったく同じである．

6.2 パイプライン法による並列化

一般に，**パイプライン法**による並列化は，計算に必要なデータ（ここでは係数の値）が同じプロセッサ内にある場合は計算を続け，必要とするデータ（ここでは係数の値をさす）が同じプロセッサ内にないと分かった時点で初めてそのデータを持っている他のプロセッサからデータを送ってもらうという並列化の方法である[17]．パイプライン法の計算の動きの概略を**図7**に示す．図中の丸印のことを以下では要素と呼ぶことにする．

パイプライン法の計算手順は，たとえば，図7中のプロセッサ：PE3にある●の要素が計算対象の場合，

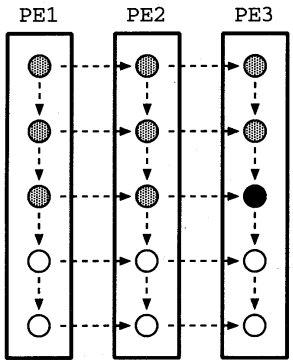

図7 パイプライン法における計算の動きの概略
Fig. 7 Outline of procedure of the pipeline method.

以下の(a)～(d)のように表せる.

[パイプライン法の処理手順]

(a) 同じPE3にある計算がすでに終了した要素(網掛けの丸印)を用いて計算をする.

(b) PE2の要素(網掛けの丸印)の計算終了を待つ.ただし,PE2の計算はPE1にある要素(網掛けの丸印)の計算終了を待たなければならない.

(c) PE2から計算が終わった要素(網掛けの丸印)の値を通信で送ってもらう.

(d) これで初めて●の値が計算できる.

6.3 OS法

パイプライン法を応用した並列化方法がOS法である.ここでは,式(3)で表された係数 a_{2k} は,$k=pm+i$ のとき

$$a_{2k} = -\frac{N}{2k}\sum_{j=1}^{k}\frac{a_{2k-2j}}{2j+1} = -\frac{N}{2k}\left\{\sum_{j=1}^{i-1}\frac{a_{2k-2j}}{2j+1}+\sum_{j=i}^{k-1}\frac{a_{2k-2j}}{2j+1}+\frac{a_0}{2k+1}\right\} \quad (m=1,\cdots,\frac{N}{2p}-1) \tag{15}$$

とプロセッサの番号:i とその台数:p を含んだ式で表される.並列化した場合に,他のプロセッサでの計算結果が必要な部分は,式(15)の括弧{ }内の第1番目の和:

$$\sum_{j=1}^{i-1}\frac{a_{2k-2j}}{2j+1} \tag{16}$$

にあたる.すなわち,他のプロセッサPEでの計算が終わった後,必要な値を通信で受けとり,次に自分のプロセッサPE内での計算が始められることになる.プロセッサの台数が4の場合には,指標 i には各PE番号そして指標 p には4が入る.すなわち,PEの番号が4のときは指標 $i=4$ である.$L=2(4m+4)$ とおいたとき,プロセッサ L での計算式は以下のように示される.

$$a_L = -\frac{n}{L}\sum_{j=1}^{4m+4}\frac{a_{L-2j}}{2j+1} = -\frac{n}{L}\left\{\sum_{j=1}^{3}\frac{a_{L-2j}}{2j+1}+\sum_{j=4}^{4m+3}\frac{a_{L-2j}}{2j+1}+\frac{a_0}{L+1}\right\} \tag{17}$$

たとえば,係数 a_{24} のとき,すなわち $m=2$ としたとき,プロセッサ:PE4の担当する計算の具体的な式は次のようになる.

$$\begin{aligned} a_{24} &= -\frac{n}{24}\sum_{j=1}^{12}\frac{a_{24-2j}}{2j+1} \\ &= -\frac{n}{24}\left\{\sum_{j=1}^{3}\frac{a_{24-2j}}{2j+1}+\sum_{j=4}^{11}\frac{a_{24-2j}}{2j+1}+\frac{a_0}{24+1}\right\} \\ &= -\frac{n}{24}\left\{\left(\frac{a_{22}}{3}+\frac{a_{20}}{5}+\frac{a_{18}}{7}\right)\right. \\ &\quad +\left(\frac{a_{16}}{9}+\frac{a_{14}}{11}+\frac{a_{12}}{13}+\frac{a_{10}}{15}\right) \\ &\quad \left.+\left(\frac{a_8}{17}+\frac{a_6}{19}+\frac{a_4}{21}+\frac{a_2}{23}\right)+\frac{a_0}{25}\right\} \end{aligned} \tag{18}$$

実際には,3つの係数:a_{18}, a_{20}, a_{22} の値が他のプロセッサでまだ求まっていない場合は,計算終了後ただちにデータの送信が行われ,そこで初めて係数 a_{24} の計算が始まることになる.

OS法の計算手順を以下の(a)～(c)に示す.全体の計算時間がおよそ $\frac{1}{4}$(ただし,PE数が4台の場合)に短縮されると期待できる.

[OS法の計算手順]

(a) 各PEは,割り当てられた S_j を演算する.他のPEで演算している結果が必要になったときは,そこで実行を停止し待機する.

(b) S_j の演算が終わった段階で係数 a_{2k} を求める.

(c) 各PEは,係数 a_{2k} の演算が終了した時点で,その結果を他のPEに送信する.これにより,他のPEは中断していた計算を再開する.たとえば,PE1が送信した時点で,PE2は必要な値が全部揃うた

表6 SP2/48の仕様
Table 6 Specifications of SP2 with 48 processors.

プロセッサ間のネットワーク	HPS スイッチ（40 Mb/s）
プロセッサ	POWER2（66 MHz）
性能（倍精度）	12.67 Gflops
メモリ	6.14 Gbyte

表7 演算時間と通信時間とその合計時間（単位：秒）およびそれらの割合（下段括弧内，単位：%）
Table 7 Computation, communication time and their summation (in seconds) and their ratios in parenthesis.

次数	IS-C法			OS法		
	演算時間	通信時間	合計時間	演算時間	通信時間	合計時間
512	.491	.252	.743	.453	.248	.701
	(66.1)	(33.9)		(64.6)	(35.4)	
1024	3.53	0.46	3.99	3.48	0.45	3.93
	(88.5)	(11.5)		(88.5)	(11.5)	
2048	34.2	1.0	35.2	34.0	1.3	35.3
	(97.2)	(2.8)		(96.3)	(3.7)	
4096	396.1	2.4	399	397.3	7.2	405
	(99.4)	(0.60)		(98.2)	(1.8)	

め，再び計算を開始する．

7. 並列計算機による数値実験

PCで使用したC++言語のプログラムを使って計算できる並列計算機の環境を探したが見つけ出すことができなったので，以下の計算はFortran90で記述したプログラムによる多倍長演算で行った．並列計算機は，日本原子力研究所計算科学技術推進センターのIBM SP2システム（48プロセッサ）を使用した[2]．**表6**に使用したIBM SP2/48のシステムの仕様を表す．Fortran90のコンパイラはIBM独自のXL Fortran (verion 4)と呼ばれるものを使用した．プログラムの最適化のレベルは–O3である．ここで，HPSスイッチとは小規模なクロスバーを複数段接続した多段結合型のアーキテクチャである．

表7に，プロセッサ数が4台のときのIS-C法とOS法における実行時間の中の演算時間（単位：秒）と通信時間（単位：秒）およびそれらの割合（下段の括弧内，単位%）を示す．この表から，次数Nが大きくなるに従って，演算に要する時間が急激に増加すること，たとえば，次数$N=4096$のときは計算全体のほとんど（98～99%）が演算時間で占められること，などが分かる．

表8にプロセッサ数が4台のときの逐次処理（CPU時間，単位：秒）および3つの並列化方法の実行時間（単位：秒）とその比率を示す．括弧内の数字は逐次処理の速度を1としたときの比率である．方程式の次数Nは512次から8192次まで変化させた．この表から以下のことが観察される．ただし，次数が8192次のとき乗算処理はFFT（Fast Fourier Transform）にて行った．

表8 逐次処理のCPU時間および3つの並列化方法の実行時間（単位はいずれも秒）およびそれらの比率（括弧内）
Table 8 CPU time (in seconds) of sequential process and execution time of three parallel methods, and their ratios in parenthesis.

次数	逐次	IS-C法 (比)	IS-B法 (比)	OS法 (比)
512	1.96	0.74 (2.65)	0.77 (2.55)	0.70 (2.80)
1024	14.1	3.99 (3.53)	4.00 (3.53)	3.93 (3.59)
2048	137.1	35.2 (3.89)	36.9 (3.72)	35.3 (3.88)
4096	1582.	399. (3.96)	400. (3.96)	405. (3.91)
8192	不可	6319. (—)	6329. (—)	6330. (—)

表9 プロセッサ数の違いによる実行時間の変化（単位：秒）
Table 9 Execution time in seconds with increase of processors.

PE数	4096次		8192次	
	実行時間	比	実行時間	比
1	1583.2	1.0	20724.3	1.0
4	398.8	4.0	5194.2	4.0
8	203.6	7.8	2619.4	7.9
16	106.5	14.9	1336.8	15.5
32	60.3	26.3	697.4	29.7
48	49.7	31.8	523.1	39.6

(1) IS-C法とIS-B法そしてOS法は，各々通信に要する時間は多少異なるが，係数の計算に必要な演算の回数自体は同じはずである．そのため，大きな次数Nのときは，演算時間が計算全体に占める割合が多くなるので，これら3つの方法の実行時間もまたほぼ等しくなると思われる．

(2) したがって，次数Nが大きいとき台数効果に相当する約4倍に近い速度の向上が達成されている．

次にプロセッサ数を増加させて実行時間の変化の様子を調べた．**表9**に結果を示す．表中の比はプロセッサ数が1台のときのCPU時間に対する比率を表している．この表からプロセッサ数を増加させることによって実行時間は大幅に短縮することが分かる．

表10は，48プロセッサを使用したときの，(i) FFTを使用しない場合と，(ii) FFTを使用した場合の実行時間（単位：秒）を比較したものである．次数Nが大きくなると，FFTを使用した方がよいことが分かる．IBM SP2（48プロセッサ）では，およそ2020桁（=9180次）よりも大きい場合はFFTを使用した方

表10 48プロセッサを用いた場合のFFTの効果（単位：秒）
Table 10 Speedups by FFT on SP2 with 48 processors.

次数	a：使用せず	b：FFT使用	比 ($\frac{b}{a}$)
16384	6603	4555	0.69
20480	15493	7255	0.47

表11 実行時間（48プロセッサ，単位：秒）
Table 11 Execution time on SP2 with 48 processors.

次数	実行時間
4096	49.7
8192	523.1
16384	4554.8
20480	7255.2

表12 3つのケースの実行時間（単位：時間）とその比率
Table 12 Execution time of three cases and ratios.

次数	a：PC	b：1PE	c：48PE	比 ($\frac{a}{c}$)
4096	0.903	0.439	0.013	69.5
8192	13.33	6.671	0.145	91.9
16384	213.4	—	1.265	168.6
20480	561.1	—	2.015	278.5

がよいことが分かった．**表11**は，4096次から20480次までのそれぞれの方程式の係数を48プロセッサを使って求めるのに要した実行時間を示したものである．

表12は，PC（Pentium Pro）とSP2（プロセッサ1台：1PE）とSP2（プロセッサ48台：48PE）の3つのケースの実行時間（単位：時間）を比較したものである．この表から，たとえば，20480次の場合，PCで561時間（およそ23日）の処理が並列計算をした結果，わずか2時間1分で終了したことが分かる．

8. まとめ

Chebyshevの数値積分公式の分点を決める代数方程式の次数Nを無限大に近づけていくとき，その係数a_kが漸近する式の妥当性を次数$N=20480$まで数値的に確認できた．また，多倍長演算においては，演算時間の計算全体に対して占める割合が非常に大きく，次数Nが大きいときには台数効果に相当する速度の向上が得られる，など並列化の効果が大きいことが分かった．また，検討した2つの並列化手法の効率の違いは小さいことも分かった．

謝辞 代数方程式の係数a_kの漸近式について貴重なコメントをいただいた東京大学名誉教授森口繁一氏，多倍長演算プログラムパッケージMPPACKの使用に関して有益な助言を頂戴した神奈川工科大学助教授平山弘氏，さらに多倍長演算について多くのご助言とご協力をたまわった愛知工業大学教授秦野和郎氏に感謝の意を表する．また，並列計算機IBM SP2使用の便宜と多くの助言をいただいた日本原子力研究所計算科学技術推進センターならびにCOMPACS運用グループの方々に心より感謝の意を表する．多倍長演算の除算の高速化についての貴重な文献[15]のご教示や有益なるご指摘を多数いただいた査読者に深く感謝する．なお，この研究の一部は，広島市立大学特定研究費（1997年度番号：9744）および文部省平成10年度科学研究費補助金（基盤研究（B），課題番号10440031）の援助を受けた．ここに記して謝意を表する．

参考文献

1) 赤坂 隆：数値計算，コロナ社 (1967).
2) 青山幸也：RS-6000 SP 並列プログラミング虎の巻（MPI版），日本IBM (1997).
3) Berstein, S.: Sur les formules de quadrature de Cotes et Tchebycheff, *Comptes Rendus (Doklady) de l'Academie des Sciences de l'URSS*, Vol.14, pp.323–326 (1937).
4) 平山 弘：C++言語による高精度計算パッケージの開発，日本応用数理学会論文誌，Vol.5, pp.307–318 (1995).
5) 平山 弘：高精度計算プログラムマニュアル（C++版，Fortran版）(1997).
6) 伊理正夫，山下 浩，寺野隆雄，小野令美：大域的収束性をもつ代数方程式の解法，京都大学数理解析研究所講究録，No.339, pp.43–69 (1978).
7) Kuzmin, R.O.: Sur la méthode de quadrature de Tchebycheff, *C.R. Acad. Sci.*, Vol.202, pp.272–273 (1936).
8) Kuzmin, R.O.: Sur la distribution des racines des polynomes dans la méthode de quadrature de Tchebycheff, *Bulletin de l'academie des Sciences de l'URSS*, Vol.4, pp.429–444 (1938).
9) 森 正武：数値解析，共立出版 (1973).
10) 森口繁一：数値計算工学，岩波書店 (1989).
11) 森口繁一：日科技連，CP研究部会資料 (June 1981).
12) 小野令美：Durand-Kerner法とAberth法を用いた超高次方程式の数値計算，情報処理学会論文誌，Vol.20, No.5, pp.399–404 (1979).
13) 小野令美：Durand-Kerner-Aberth法を用いたある種の超高次方程式の解の数値計算，情報処理学会論文誌，Vol.22, No.2, pp.165–168 (1981).
14) 長田直樹：数値微分積分法，現代数学社 (1987).
15) Ozawa, K.: A Fast $O(n^2)$ division algorithm for multiple precision floating-point arithmetic, *J. of Information Processing*, Vol.14, pp.345–356 (1991).
16) 戸田英雄，小野令美：高精度計算用の除算のアルゴリズムに関して，電子技術総合研究所彙報，Vol.42, No.4, pp.66–71 (1978).

17) 富田眞治：並列コンピュータ工学，昭晃堂 (1996).

(平成10年11月2日受付)
(平成11年 9 月2日採録)

益本　博幸

1975年生．1996年3月国立弓削商船高等専門学校卒業．1998年3月広島市立大学情報科学部卒業．現在（株）アイテック阪神に勤務．

藤野　清次（正会員）

1950年生．1974年京都大学理学部（数学専攻）卒業．1993年博士（工学）東京大学．1994年3月広島市立大学情報科学部教授．1994年8～9月 Karlsruhe 大学客員研究員．数値計算，並列計算に興味を持つ．著書「反復法の数理」(共著朝倉書店）等．日本応用数理学会会員．

小野　令美（正会員）

1932年生．1954年御茶の水女子大学理学部数学科卒業．1985年工学博士（東京大学）．1997年3月千葉大学工学部定年退官．数値計算，データ解析に興味を持っている．著書「入門数値計算」(共著，オーム社)．日本応用数理学会，応用統計学会会員．

児島　　彰（正会員）

1965年生．1992年京都大学工学部情報工学科卒業．1993年京都大学大学院工学研究科情報工学専攻修士課程修了．1996年京都大学大学院工学研究科情報工学専攻博士課程認定退学．同年広島市立大学情報科学部情報工学科助手．並列化コンパイラ，並列処理の研究に従事．