社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－HPC－99　(14)
2004／7／31

補外法に基づく数値積分法の実装と性能評価

幸谷智紀*　鈴木千里 *

概要

本稿では，分割数列として harmonic 数列を用いた補外法 (GBS アルゴリズム) に基づく数値積分法を分散メモリ並列マシンとしての PC Cluster の上で実装し，多倍長浮動小数点数を用いて Kahaner のテスト問題を解いた結果について報告する。多倍長浮動小数点ライブラリには GNU MP(GMP)+MPFR をベースとして実装した MPIBNCpack を使用した。Kahaner の 21 ものテスト問題の中には，積分区間内に特異点を含むものもあるため，簡易な Overflow 対策も施した。この結果，多くの問題で計算時間が減少することが確認できた。

Implementation of Numerical Integration Methods based on Extrapolation with harmonic sequences and its Performance Evaluation

Tomonori Kouya*　Chisato Suzuki*

Abstract

In this paper, we describe our multiple precision numerical integration programs based on extrapolation methods with harmonic sequences and its performance on PC clusters by using Kahaner's 21 test problems. The multiple precision library used in our implementation is MPIBNCpack based on GNU MP(GMP) and MPFR. In addtion, we implement simple overflow handling mechanism in our programs, because some test problems have the integration interval which contain singular points. Consequently, our programs can reduce computational time in many test problems.

1　初めに

GBS(Gragg-Bulirsch-Stoer) アルゴリズムは，補外法に基づいた常微分方程式の初期値問題向けの解法であり，補外法の段数を変化させることによって任意次数の近似解を得る事ができる。

補外法の計算時間を左右するのは初期系列の計算量であり，これは積分区間の分割数を決定する数列，分割数列によって決定される。この分割数列が急速に発散するような場合は段数を上げるに従って積分計算に要する時間も急速に増える一方，初期系列に含まれる打切り誤差は急速に減少し，丸め誤差についても補外計算部分での増大が抑えられる効果が見られる。即ち，計算時間と打切り誤差・丸め誤差の性質はトレードオフの関係にある。Burlisch, Stoer らは常微分方程式の初期値問題について，３種の分割数列 (Romberg, Bulirsch, harmonic) を比較した [5]。また，Romberg 数列を用いた補外法の数値的性質については室伏 [6] が詳しく調査している。

しかし，並列計算が容易な一次元積分問題について，丸め誤差の調整が容易な多倍長浮動小数点数を用いた場合の網羅的な数値実験を行った研究はない。今回我々は分散メモリの PC Cluster において，PE 数に応じた改良 harmonic 分割数列を用いた並列分散多倍長積分ルーチンを実装し，Kahaner のテスト問題に適用してその性能評価を行い，その有用性を確認することにした。

多倍長浮動小数点ライブラリとしては，GMP[2] と MPFR[1] を元に構築した MPIBNCpack[7] を用いる。MPFR には IEEE754 と同様，オーバーフロー (±Inf)

*静岡理工科大学
*Shizuoka Institute of Science and Technology

や非数 (NaN) を扱うことができるため，これを利用した簡易な特異点対策も併せて行い，全てのテスト問題を計算できるようにした。

2 簡易化したGBSアルゴリズム

GBS アルゴリズムは常微分方程式の初期値問題

$$\left\{ \begin{array}{rcl} \dfrac{d\mathbf{y}}{dx} & = & \mathbf{f}(x,\mathbf{y}) \\ \mathbf{y}(x_0) & = & \mathbf{y}_0 \end{array} \right. \tag{1}$$

の任意の次数の近似解を，補外法を用いて求めるものである。このアルゴリズムを一次元の数値積分

$$\int_a^b f(x)\ dx \tag{2}$$

に適用するには，(1) を (2) の形式で表現できればよい。従って以下では

$$\left\{ \begin{array}{rcl} \dfrac{dy}{dx} & = & f(x) \\ y(a) & = & f(a) \end{array} \right. \tag{3}$$

という形式の初期値問題を考えることにする。

我々の簡易化した GBS アルゴリズムは，次のような手順で計算が進められる。

初期値 $f(x_0)(= f(a))$ からステップ幅 H_1 だけ進んだ地点の解 $y(x_1)(= y(x_0+H_1))$ の近似値 y_1 を求めるものとする。

まず，ステップ幅 H_1 を定め，これを更に小ステップに分割して初期系列を求める。この分割数を定める分割数列を $\{n_1,\ n_2,\ ...,\ n_j,\ ...,\ n_m\}$ とすると，各分割数 n_j に対して初期系列 T_{j1} を台形公式

$$T_{j1} = h_j \left\{ \frac{1}{2}\left(f(x_0)+f(x_1)\right) + \sum_{i=1}^{n_j-1} f(x_0+ih_j) \right\} \tag{4}$$

を用いて求める。ここで小ステップ h_j は $h_j = H_1/n_j$ である。

こうして求められた初期系列 $T_{11}, T_{21}, ..., T_{m1}$ ををを用いて補外計算

$$T_{j,k+1} := T_{jk} + \frac{T_{jk} - T_{j-1,k}}{\left(\frac{n_j}{n_{j-k}}\right)^2 - 1} \tag{5}$$

を行い，h_1 に関して 2 次の漸近展開として表現できる打ち切り誤差項を削っていく [5]。この過程は補外表

$$\begin{array}{c|cccc} \text{初期系列} & & & & \\ T_{11} & & & & \\ T_{21} & T_{22} & & & \\ T_{31} & T_{32} & T_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ T_{m1} & T_{m2} & T_{m3} & \cdots & T_{mm} \end{array} \tag{6}$$

として表現される。この m を補外における段数と呼ぶ。最後に得られた T_{mm} は，

$$T_{mm} = \int_{x_0}^{x_1} f(x)dx + O(h_1^{2m})$$

となることが知られている。

この補外計算の過程で，逐次に収束判定を

$$|T_{j,k+1} - T_{jk}| \le \varepsilon_R\, |T_{j,k+1}| \tag{7}$$

という条件式を用いて行う。ここで ε_R は相対誤差の許容度を定める正定数である。この収束条件を満足すれば，そこで補外計算を打ち切って y_1 を決定し，次の区間の積分計算 $y_2 = y_1 + \int_{x_1}^{x_2} f(x)dx$ を行う。収束条件を満足しなければ，$H_1 := H_1/2$, $m := m+1$ としてもう一度最初から計算をやり直す。

こうして，積分区間を $[x_0, x_1]$(ステップ幅 $x_1 - x_0 = H_1$) $\to [x_1, x_2]$(同 H_2) $\to \cdots \to [x_{N-1}, x_N]$(同 H_N) と順次進めながら近似値 $y_1,\ y_2,\ ...,\ y_N$ を求め，最後の y_N を (2) の近似値として採用する。

Bulirsch, Stoer らはこの GBS アルゴリズムに 3 種類の分割数列を適用して比較検討を行っている。そのうち，もっとも数値的性質が優れているのは Romberg 数列，計算時間の点で優れているのは harmonic 数列であると結論付けている。

Romberg 数列 $\{2, 4, 8, 16, 32, ..., 2^j, ...\}$

harmonic 数列 $\{2, 4, 6, 8, 10, ..., 2j, ...\}$

室伏ら [6] が提唱する NIM 法は，Romberg 数列を用いた GBS アルゴリズムに基づいており，その特性を生かした収束判定が可能であることを示している。よって我々の簡易化 GBS アルゴリズムは harmonic 数列を基本の分割数列として採用する。

今回対象とする一次元積分問題では，初期系列の計算には台形公式のみが用いられるため，常微分方程式

の場合とは異なり，この部分の並列化は容易に行える。即ち，(4) 式の，端点を除く関数計算を各 PE で並列に実行すればよい。更に，分割数列を使用 PE 数に応じたものに改良すれば，遊んでしまう PE もなくなり，Romberg 数列に比較して劣る収束性能もある程度は改善できるものと予想される。また，多倍長浮動小数点数での利用に限れば，補外計算における丸め誤差の伝播と拡大もさして問題にならない。必要に応じて，伝播による丸め誤差の拡大に見合った程度の桁数を増やしても計算時間は Romberg 数列を用いた場合よりも少なくなると考えられる。

これを，1CPU 環境 (Pentium IV 2.8 GHz, MPFR 2.0.3, gcc 3.3.2) で，次の積分に適用して確認することにする。これは後述する Kahaner のテスト問題の 1 番目にあたるものである。

$$\int_0^1 \exp(x)dx = e - 1 \tag{8}$$

比較のため，分割数列として，Romberg，harmonic 以外に，次のものを使用した。

harmonic 数列 $\{2, 4, 6, 8, 10, ..., 2j, ...\}$

harmonic 数列 2 $\{4, 8, 12, 16, 20, ..., 4j, ...\}$

harmonic 数列 3 $\{8, 16, 24, 32, 40, ..., 8j, ...\}$

段数と相対誤差の推移を示したものが図 1，計算時間を示したものが図 2 である。

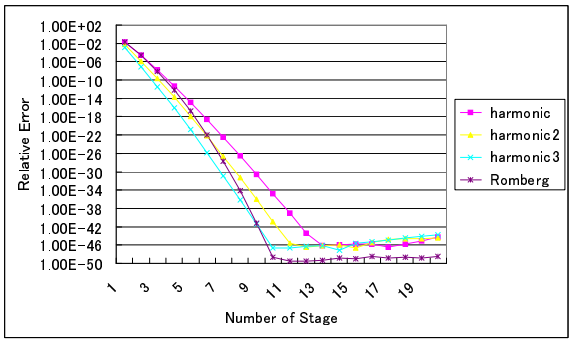

図 1: Relative Errors

harmonic 数列を用いた場合でも，等差を大きくすることで Romberg 数列を用いたものに匹敵する収束性を確保できることが分かる。また，丸め誤差については Romberg 数列のものに比較して伝播による悪化

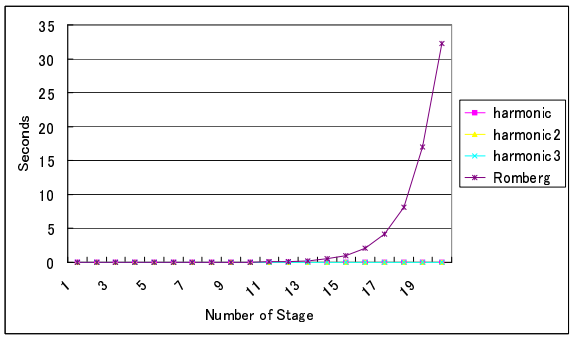

図 2: Computational Time

が見られるが，必要な精度に対して十分に桁数が確保できる多倍長浮動小数点数が利用できれば特に問題にはならない。それに対して，計算時間は Romberg 数列を用いた場合に比較して非常に優れていることが分かる。

今回は分散メモリの PC Cluster における利用を目的としているため，分割数列としては次のものを使うことにする。使用する PE 数を P とした時は

改良 harmonic 数列 : $\{P{+}1, 2(P{+}1), ..., j(P{+}1), ...\}$ (9)

となる。

従って，初期系列 T_{j1} の計算においては両端点を除いてちょうど jP 回の関数計算が必要となるため，各 PE には j 回の関数計算を要求することになる。これによって，使用する PE 数が多くなれば収束性も向上し，並列性が増して通信の overhead があっても全体の計算時間を少なくする効果が期待できる。

3 Kahaner のテスト問題と特異点に対する対策

一次元積分問題を網羅的に調べるため，Kahaner のテスト問題 (表 1) を使用する。

但し，この問題の中には積分区間内で関数値が Overflow するものも含まれている。二宮 [3] は不連続点も考慮した異常点検出アルゴリズムを提唱しているが，今回は MPFR の Inf/NaN 検出関数を用いて，積分区間の端点で Overflow した時は積分区間内へずらして計算するだけの簡単な Overflow 対策のみを施して対

応することにした (図 3)。

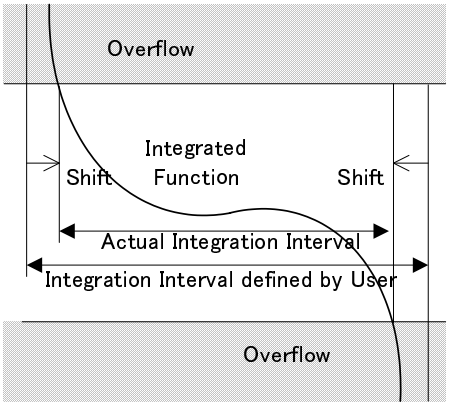

図 3: Overflow 対策

表 1: Kahaner's 21 Test Problems

No.	Integrated function	$[a, b]$
1	$\exp(x)$	$[0, 1]$
2	$\lfloor \min(x/0.3, 1) \rfloor$	$[0, 1]$
3	$\sqrt{x}$	$[0, 1]$
4	$0.92 \cosh x - \cos x$	$[-1, 1]$
5	$(x^4 + x^2 + 0.9)^{-1}$	$[-1, 1]$
6	$x\sqrt{x}$	$[0, 1]$
7	$(\sqrt{x})^{-1}$	$[0, 1]$
8	$(x^4 + 1)^{-1}$	$[0, 1]$
9	$2(2 + \sin(31.4159x))^{-1}$	$[0, 1]$
10	$(1 + x)^{-1}$	$[0, 1]$
11	$(1 + \exp(x))^{-1}$	$[0, 1]$
12	$x(\exp(x) - 1)^{-1}$	$[0, 1]$
13	$\sin(314.159x)(\sin(3.14159x))^{-1}$	$[0.1, 1]$
14	$\sqrt{50}\exp(-50 \times 3.14159x^2)$	$[0, 10]$
15	$25\exp(-25x)$	$[0, 10]$
16	$50/3.14159/(2500x^2 + 1)$	$[0, 10]$
17	$50(\sin(50 \times 3.14159x)$ $/(50 \times 3.14159x))^2$	$[0.01, 1]$
18	$\cos(\cos x + 3\sin x + 2\cos(2x)$ $+3\sin(2x) + 3\cos(3x))$	$[0, 3.1415927]$
19	$\log x$	$[0, 1]$
20	$(x^2 + 1.005)^{-1}$	$[-1, 1]$
21	$(\cosh(10(x - 0.2)))^{-2}$ $+(\cosh(100(x - 0.4)))^{-4}$ $+(\cosh(1000(x - 0.6)))^{-6}$	$[0, 1]$

4 数値実験

使用した計算機環境は以下の通りである。

cs-pccluster2(10 Nodes, 10 CPUs) Pentium IV 2.8cGHz, 512MB RAM, 1000BASE-T, mpich-1.2.5.2, gcc-3.3.2

VTPCC(8 Nodes, 16 CPUs) Xeon 3.06GHz Dual, 1GB RAM, 1000BASE-T, mpich-1.2.5, gcc-3.2

なお, 多倍長浮動小数点数ライブラリには, どちらも GMP 4.1.2 と MPFR 2.0.3 を用い, 台形則には MPIB-NCpack の mpf_trapozoidal_fs 関数に前述の Overflow 対策を施したものを使用した。従って, 計算時間を除いては, どちらの計算機環境においても精度及びステップ数はほぼ同じになる。

4.1 収束性と相対誤差による比較

計算に際しては, $x = a$ から $x = b$ 方向へ計算したもの (Forward, F) と, その逆方向へ計算したもの (Backward, B) の両方を, それぞれ 10 進 50 桁 (2 進 333bits) で実行した。段数 m は, $16 \leq m \leq 32$ の範囲で変動するように制限されている。その結果, 全ての問題について, F 方向, B 方向のどちらかは収束することが確認できた (表 2)。

表 2: Convergent

	$\varepsilon_R = 1.0 \times 10^{-15}$		$\varepsilon_R = 1.0 \times 10^{-30}$	
No.	F	B	F	B
1	○	○	○	○
2	○	○	○	○
3	×	○	×	○
4	○	○	○	○
5	○	○	○	○
6	×	○	×	○
7	○	×	○	×
8	○	○	○	○
9	○	○	○	○
10	○	○	○	○
11	○	○	○	○
12	○	○	○	○
13	○	○	○	○
14	○	○	○	×
15	○	○	○	○
16	○	○	×	○
17	○	○	○	○
18	○	○	○	○
19	○	×	○	×
20	○	○	○	○
21	○	○	○	○

表 3: Relative errors and numbers of steps

	$\varepsilon_R = 1.0 \times 10^{-15}$		$\varepsilon_R = 1.0 \times 10^{-30}$	
No.	ReErr	N	ReErr	N
1	3.8E-20	2	9.5E-37	2
2	0.0E+00	251	0.0E+00	251
3	9.8E-18	113	7.5E-33	113
4	4.9E-21	2	1.3E-37	2
5	1.4E-18	2	3.0E-33	4
6	9.0E-19	69	2.6E-33	69
7	4.4E-18	163	1.0E-25	220
8	1.4E-18	2	7.7E-34	2
9	6.5E-17	8	2.8E-33	26
10	7.4E-19	2	1.6E-33	2
11	2.1E-20	2	5.9E-36	2
12	7.5E-20	2	5.5E-36	2
13	5.3E-18	16	5.5E-33	32
14	2.8E-15	4	4.7E-31	10
15	1.9E-17	6	6.6E-34	19
16	5.8E-17	8	7.0E-33	11
17	3.4E-17	15	1.2E-33	32
18	1.9E-17	4	4.2E-32	8
19	5.7E-18	163	5.4E-33	167
20	1.6E-18	2	4.6E-35	4
21	5.4E-17	16	2.1E-33	27

数値解の相対誤差 (ReErr)，及び，総ステップ数 (N) を表 3 に示す。F 方向で収束した場合はその結果を，そうでない場合のみ，B 方向の結果を使っている。

今回使用したテスト問題では，停止則に指定した ε_R の値と同程度の相対誤差になっていることが分かった。しかし，特異点を含む問題では，ステップ数が極端に多くなっており，効率的な処理が出来ているとは言いがたい。

4.2 計算時間について

表 3 のステップ数から，次の条件に当てはまる問題は並列化の効果が薄いことが容易に予想できる。

No.1, 4, 5, 8, 10, 11, 12 ･･･2 ステップで収束している問題。最大段数が多い (32 段) ため，最小のステップ数で解が得られている問題では並列化の効果はない。

No.2, 3, 6, 7, 19 ･･･特異点を含む問題。特異点の処理に時間が掛かる上，特異点の周辺での収束性が悪いため，PE 数を増やしてもさほど効果が得られない。

従って，これらを除いた問題について計算時間の低減効果を見ることにする。我々の提唱する分割数列は PE 数が増えるに従って収束性が増し，総ステップ数 (N) も減ることが期待される。実際，図 4 に見られるように，これらの問題については総ステップ数が減っていることが分かる。

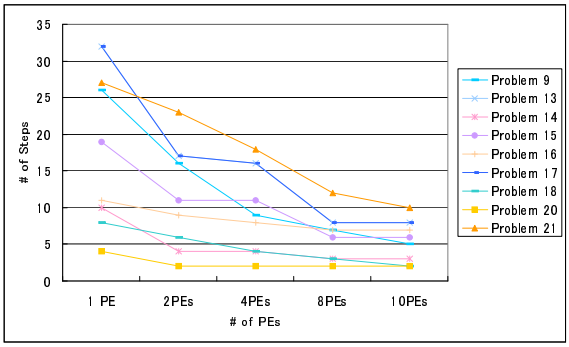

図 4: Steps

総ステップ数が減れば，全体の計算時間も減ることが期待される。実際，Dual CPU Node の PC Cluster(VTPCC，図 6) でも，1CPU Node のもの (cs-

pccluster2, 図 fig:sec-cs-pccluster2) でも，計算時間は減っていることが判明した。

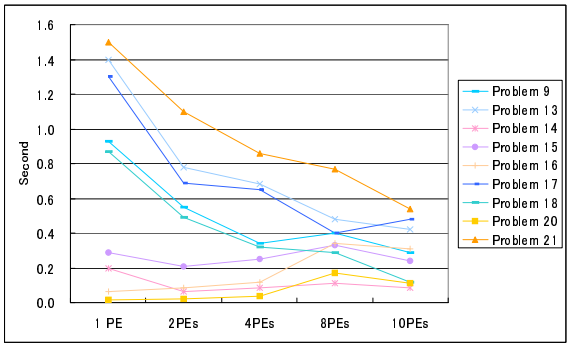

図 5: cs-pccluster2

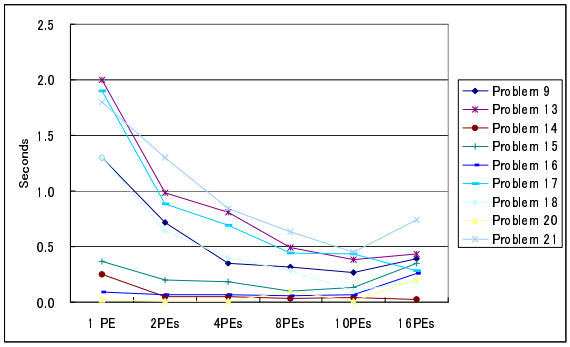

図 6: VTPCC

5　今後の課題

今回実装した GBS アルゴリズムは常微分方程式の初期値問題向けのものであり，それをほぼそのまま数値積分に適用しても，それなりの実行性能と精度が確保できることは確認できたが，数値積分アルゴリズムとしてはまだ改良する必要があり，特に，特異点の処理を高速に実行する必要がある。今後の課題としては，この結果を土台として，harmonic 数列を用いた補外アルゴリズムを,「適応型」積分アルゴリズムにまで改良することが挙げられる。

謝辞

今回使用した cs-pccluster2 は，静岡理工科大学学内研究費の助成を受けて構築されたものである。VTPCC は名古屋大学三井研究室に設置されたものを使用した。また,「計算ソフトとその周辺ワークショップ」参加者，職業能力総合大学校　室伏誠先生，永坂秀子先生からは本研究に対して意見を頂いた。ここで御礼申し上げる。

参考文献

[1] MPFR Project, http://www.mpfr.org/

[2] GNU Multiple Precision Library, http://swox.com/gmp/

[3] 二宮市三, 適応型ニュートン・コーツ積分法の改良, 情報処理 Vol.21, No.5, 1980.

[4] 日比野・長谷川・二宮・細田・佐藤, 二宮法と FLR 法との結合による新しい適応型積分法, 情報処理学会論文誌 Vol.44, No.10, 2003.

[5] E.Harier, S.P.Nørsett, G.Wanner, Solving Ordinary Differential Equations I (Second Ed.), Springer-Verlag, 1991.

[6] 室伏誠, 有限桁計算における Richardson の補外法による丸め誤差評価の研究, 博士論文 (日本大学), 1998.

[7] BNCpack/MPIBNCpack, http://na-inet.jp/na/bnc/

[8] 幸谷智紀, 多倍長数値計算ライブラリ BNCpack とその並列化について, (To appear).

[9] 幸谷智紀, PC Cluster 上における多倍長数値計算ライブラリ BNCpack の並列分散化, Linux Conference 2003 抄録集 Vol.1, CP-14, 2003.

[10] 幸谷智紀, 補外法を用いた並列任意精度 ODE Solver の実装と PC Cluster における性能評価, 静岡理工科大学紀要, 2004.

[11] 幸谷智紀, 新しい PC Cluster の性能評価及び数値積分法への応用, 第 13 回計算ソフトとその周辺ワークショップ, 2004.

[12] 幸谷智紀, harmonic 数列を用いた補外法の性能評価, 第 33 回数値解析シンポジウム, 2004.