# 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

Implementation of Pallalelized Arbitrary Precision ODE Solvers using Extrapolation Methods and Evaluation of its Performance on PC clusters

幸谷智紀*

Tomonori KOUYA*

(2004 年 2 月 27 日受理)

Abstract: In this paper, we firstly describe how to constuct “cs-pccluster2” as our new Pentium IV-based PC cluster. Secondly, the performance of cs-pccluster2 are evaluated by using BNCpack and MPIBNCpack. BNCpack is a multiple presicion numerical calculation library with GMP+MPFR, and MPIBNCapck is parallelized BNCpack with MPI. Thirdly, our arbitrary precision ODE solvers based on extrapolation methods and its availability are described by numerical examples of the resonance problem. Lastly, we apply the ODE solvers to a linear ODE on cs-pccluster2 and evaluate the performance.

## 1. 初めに

本稿では，Pentium IV を用いた PC cluster(cs-pccluster2) の概要と，その上で動作する補外法を用いた並列分散型任意精度常微分方程式 (ODE) Solver について述べる。ここで言う「任意精度」とは，今井ら [6] が使用している「無限精度」という用語と同義である。この二つの用語は,「理論誤差 (打切り誤差)」を任意に調整することが可能な数値計算アルゴリズムに,「丸め誤差」の大きさを任意に調節できる多倍長浮動小数点数を組み合わせて使用する，ということを意味する。「無限精度」という用語は今井らが Trade mark として利用しているので，本稿では「任意精度」という用語をこの意味で使用する。

昨年度は，Pentium III を用いた PC cluster(cs-pccsluter[16]) の上で動作する，並列分散型多倍長数値計算ライブラリ MPIBNCpack[12, 4] を開発し，その有用性について議論してき

*理工学部 情報システム学科

た。このライブラリは，GMP[1], MPFR[2], BNCpack[3] を IEEE754/多倍長数値計算部分に用い，並列分散化には MPI の実装系である mpich[11] を用いて構築されたものである。

cs-pccluster は，著者が本学に着任した際，卒業研究用及び学生実験用に購入した PC を組み合わせて構築したものである。PC cluster は，単なるネットワークにつながっている PC 群であり，コストパフォーマンスの高い，汎用性のある技術である。本学のような大学においては，通常の講義・実習実験・卒業研究全てに利用しつつ，研究用の大規模な数値計算を行う，という利用のされ方が望ましいが，現状の cs-pccluster では能力的に不足しつつあると感じていた。ことに，近年盛んになりつつある動画を中心としたマルチメディア用途には適さない。そのため，よりコストパフォーマンスの高い汎用 PC cluster に移行すべく，学内研究費を申請し，今回認められるに至ったのである。

以下，学内研究費によって実現された cs-pccluster2 と，その上で動作する任意精度 ODE solver について解説する。

## 2. cs-pccluster2 の概要

cs-pccluster は，前述の通り，著者が 1999 年に着任した際に購入した PC を組み合わせて構築したものである。購入当時においては最新の CPU(Pentium III 1GHz, P3) であったが，e-Learning には不可欠な，動画を用いたマルチメディア教材を自在に編集するのは不可能である。そのためには Hyper-Threading(HT) 機能を備え，大きな Cache メモリを積んだ Pentium IV(P4) レベルの CPU が最低でも必要である。これは Pentim III と同じ IA-32 アーキテクチャでに改良を加えたものであり，コンシューマ向けの CPU という位置づけなので，SMP マシンにも利用される Xeon や Opteron といった CPU に比べると価格が安く抑えられている。次期 CPU としてアナウンスされていた Prescott は，パフォーマンス向上に疑問符が付く割には発熱及び消費電力の問題があるという報道 [7] もあり，購入するのは現状の Pentium IV がふんだんに供給されている本年 (2003 年) が好機であると思われた。

また，ネットワーク技術も FastEthernet(100BASE) から GigabitEthernet(GbE, 1000BASE) へ移りつつあり，安価な GbE Network Interface Card(NIC) や Switching Hub が多く出回るようになっている。そのパフォーマンスを遅延なく発揮させるには現状の 32bit PCI バスに刺さった NIC ではなく，CSA 接続された GbE インターフェースか，バス幅が拡張された PCI Express をフルに活用する GbE NIC が必要である。しかし後者は一部のサーバ向け高価格マザーボードに搭載される程度にしか普及しておらず，前者に比べて将来性はあるものの，コストパフォーマンスはまだ高止まりしている感がある。一方，CSA 接続の GbE を備えた低価格マザーボードは 2003 年頃から出回るようになっていた。

以上の方針に従い，既存の 100BASE NIC や機器，グラフィックボードを流用して，極

Table 1: cs-pccluster2 のハードウェア

| | 親: cs-northpole | 子: cs-room443-01〜10 |
|---|---|---|
| CPU | Pentium IV 2.8C GHz | |
| Mother Boad | Gigabyte GA-8IPE1000 Pro2 | |
| L2 Cache (KB) | 512 | |
| RAM(MB) | 1024 | 512 |
| SATA IDE HDD(GB) | 80(ST380013AS ) | |
| 100BASE-TX NIC | Intel Pro/100, 3COM 3c905 | 3COM 3c905 |
| 100BASE SW | Allied-Telesis CenterCom 8124XL | |
| 1000BASE-T | Intel Kenai II(CSA) | |
| 1000BASE L2 SW | Planex FXG-16TX | |

力コストをかけず，Table 1 に示すようなスペックの PC，及び GbE 機材を購入することとした。また，既存の Web サーバ (cs-www, cs-hera 等) を生かした Web cluster の実験環境も併せて整えることが出来るよう，ネットワーク構成を考えた結果，Fig.1 のようになった。

PC は全てパーツ単位で購入し，著者のゼミ学生 (情報システム学科 3 年生 8 名) に組立作業を行ってもらった。その様子を Fig.2 に示す。当学科のネットワークシステム実験では，当学科教員の小嶋が長年 PC の組み立て作業を指導しており，当ゼミ学生は全員はその体験者である。そのため，特に指導することもなく組立作業は 1 コマ (90 分) 程度で終了した。後に若干の不具合が見られたが，故障するマシンは一台もなく，現在も順調に動作している。

なお，組み立て後にこれらのマシンを PC cluster として動作させるためのソフトウェア設定については，かなり煩雑になるのでここでは紹介しない。詳細は Web にて公開している文書 [17] とそこに掲載してある文献・URI を参照されたい。ごく簡単に述べると，OS には Vine Linux 2.6r3*，MPI には mpich-1.2.5.2[11] を用い，NIS 及び NFS でアプリケーションディレクトリとホームディレクトリを共有するようになっている。

*後にファイルシステムに不具合が見つかって，2.6r4 がリリースされた。

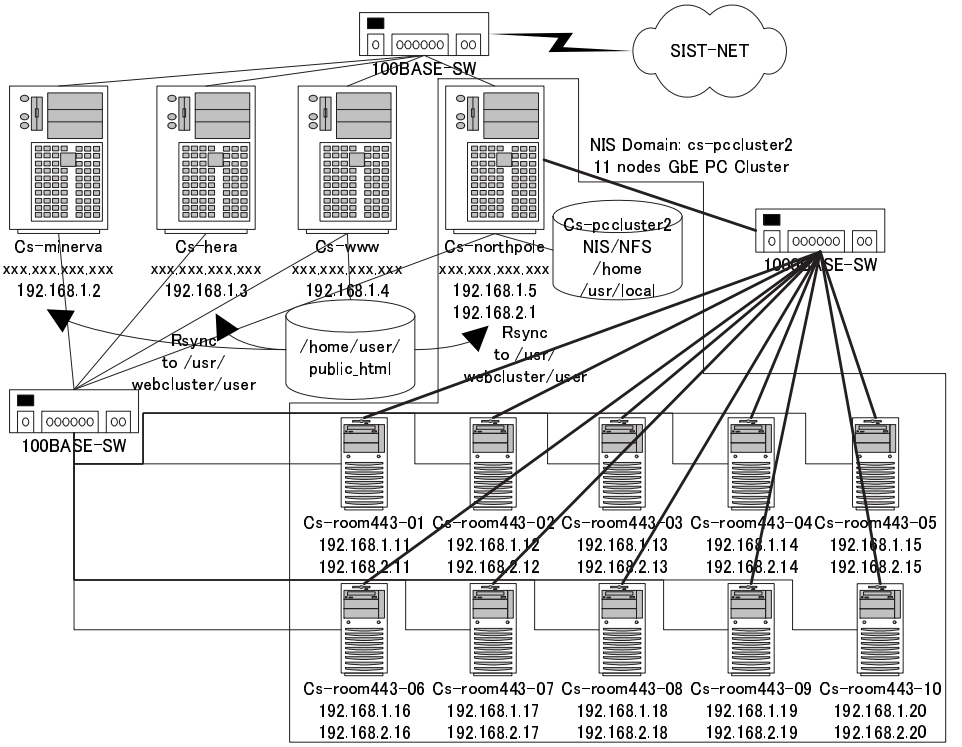


Fig. 1: cs-pccluster2

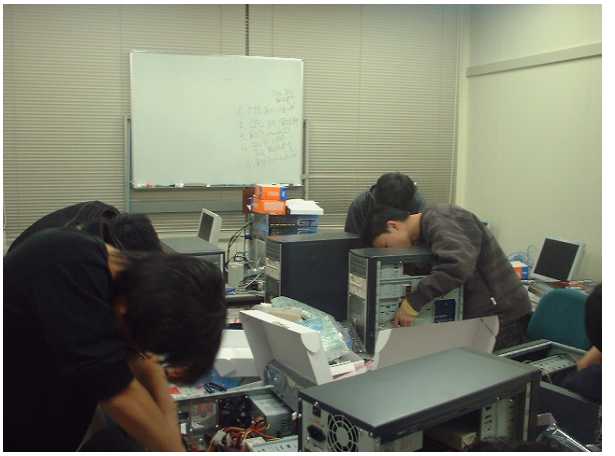

Fig. 2: PC 組み立て作業の様子

### 3. BNCpack, MPIBNCpack による性能評価

ここでは BNCpack[3] と MPIBNCpack[4] を用いたベンチマークテストの結果について，簡単に述べる。

まず，PC 単体での多倍長浮動小数点演算性能 (Fig.3) と，GbE を用いた 1 対 1 同期通信 (MPI_Send/MPI_Recv) の性能評価 (Fig.4) の結果を示す。これらは既存の P3 マシン，100BASE においても同様の結果となっている。

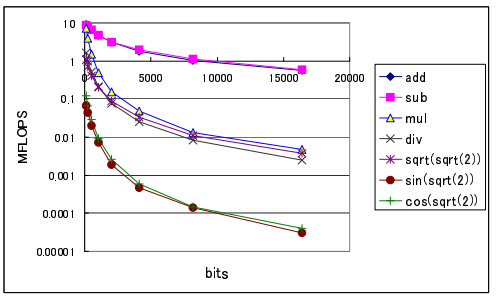


Fig. 3: P4 マシンにおける MPFR 演算性能

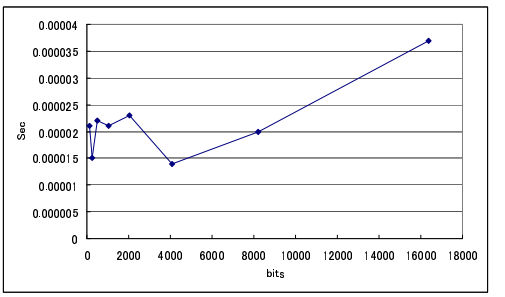


Fig. 4: GbE における 1 対 1 同期通信

次に，同じベンチマークテストプログラムを用いて，多倍長浮動小数点演算性能の速度向上比 (Fig.5) と，1 対 1 同期通信の速度向上比 (Fig.6) を計測した結果を示す。

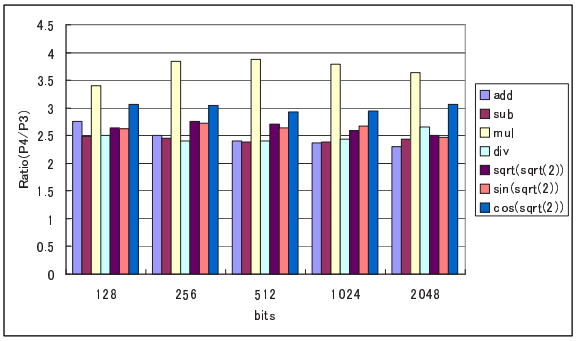


Fig. 5: MPFR 演算性能 P4 vs. P3

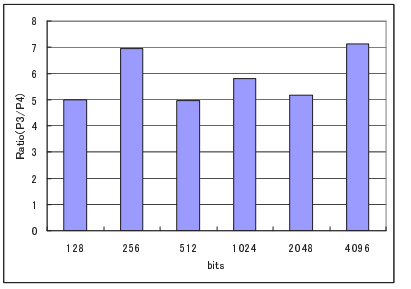


Fig. 6: 1 対 1 同期通信 P4 vs. P3

MPFR の速度向上比は，概ね 2.3 倍〜3.8 倍となっている。演算別に見ると，特に乗算の性能向上が高くなっていることが分かる。この向上率は，CPU の clock 数の向上率 2.8 倍とほぼ一致している。

同期通信は 5 倍〜7 倍の性能向上が見られる。10 倍になっていないのは，Swiching Hub もしくは NIC 等のハードウェアか，Linux kernel における TCP/IP 通信やそれを利用する mpich といったソフトウェアのどこかボトルネックになる原因があるものと思われるが，詳細は今のところ不明である。

最後に，cs-pccluster と cs-pccluster2 を用いて，128 次元と 256 次元の実正方行列同士の乗算を行った結果を示す。並列行列積のアルゴリズムについては [15] を参照されたい。

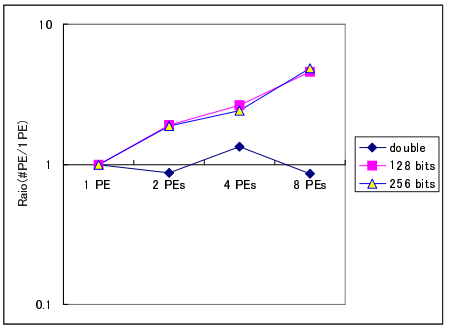


Fig. 7: 128 次元の行列積の並列性能

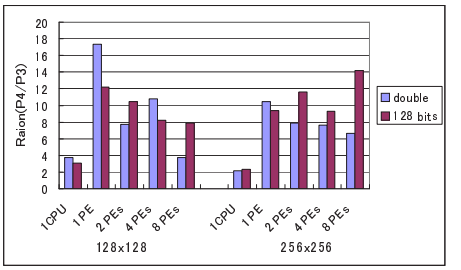


Fig. 8: 行列積の性能 P4 vs. P3

この場合，1CPU(BNCpack, mul_dmatrix 関数, mul_mpfmatrix 関数) ではせいぜい 2 倍程度向上しているだけだが，MPI を用いた並列行列積 (MPIBNCpack, _mpi_mul_dmatrix

関数, _mpi_mul_mpfmatrix 関数) では，5 倍〜13 倍の速度向上比が得られている。

この原因は，通信時間の減少や，演算の高速化といった原因のほかに，MPI(mpich) におけるオーバーヘッドが減少していることが寄与している。実際，mpich-1.2.5.2 を用いた場合，cs-pccluster において，512 次元の行列積を 1PE で MPI を用いた行列積を行うと，IEEE754 倍精度で約 4 倍の計算時間を要していた。これが，cs-pccluster2 になると，殆どオーバーヘッドは見られない。同じ MPI 実装を用いて，同じ系統の OS やハードウェアを使っているにも関わらず，この違いはどこから来るのか？

以上の性能評価によって，cs-pccluster2 の性能は予想以上に良いことが判明した。

## 4. 補外法に基づく任意精度 ODE Solver の実装と性能評価

ここでは ODE の初期値問題 (Initial Value Problem, IVP)

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

の近似解を任意の次数で求めることが出来る補外法について解説し，実際の数値例を元にその有用性を述べる。

IVP 向けの近似解法は様々なものが提案されているが，代表的なものとしては，1 ステップ前の近似解を元に次のステップの計算を進める一段法と，複数ステップ前の近似解を元に次のステップを計算する多段法が挙げられる。前者は，ステップサイズの制御が容易に行える半面，近似解の次数を上げるに従い，関数 $\mathbf{f}(x, \mathbf{y})$ の評価回数が急激に増大するという短所がある。後者はその点，関数計算は少なくて済むが，ステップサイズを変えながら計算を進めるのは，不可能ではないが面倒である。

今回実装した補外法は，一段法を使って初期系列を求め，補外計算によって次数を上げる計算方法である。以下，実装した GBS アルゴリズムとステップサイズ制御方法，線型 ODE における並列計算の効果について解説する。

### 4.1 GBS アルゴリズム

補外 (Extrapolation) という考え方は，IVP のみならず，数値微分・積分についても適用可能なものである。その実装方法については適用分野毎に提案されているものがある。IVP においては，提案者の頭文字を取った GBS(Gragg-Bulirsch-Stoer) アルゴリズムが最良のものとされている [10]。これは陽的 Euler 法と中点法を初期系列に用いた補外法であり，今回実装した solver は全てこれに基づいたものである。

簡単のため，初期値 $\mathbf{y}_0$ から，ステップサイズ $H$ だけ進んだ，厳密解 $\mathbf{y}(x_0 + H)$ の近似解 $\mathbf{y}_H$ を求めるものとする。この時，このステップサイズ $H$ を，自然数列 $\{n_1, n_2, ..., n_j, ...\}$

を用いて $h_j := H/n_j$ の小区間に等分割し，最初は陽的 Euler 法を用いて

$$\mathbf{y}_1 := \mathbf{y}_0 + h_j \mathbf{f}(x_0, \mathbf{y}_0) \tag{2}$$

を求め，以降は中点法

$$\mathbf{y}_{i+1} := \mathbf{y}_{i-1} + 2h_j \mathbf{f}(x_0 + ih_j, \mathbf{y}_i) \tag{3}$$

を用いて，$\mathbf{y}_{n_j+1}$ まで求める。最後に平滑化

$$T_{j1} := \frac{1}{4}(\mathbf{y}_{n_j-1} + 2\mathbf{y}_{n_j} + \mathbf{y}_{n_j+1}) \tag{4}$$

を行って，補外の初期系列 $T_{11}, T_{21}, T_{31}, ...$ を得る。これを元に補外計算

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

を行い，$h_j$ 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。この計算はベクトルの成分ごとに行われる。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，$2m$ 次の近似解 $\mathbf{y}_H$ となることが知られている。

GBS アルゴリズムを実装する方法として，3 つのステップサイズの分割数列が提案されている。

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

**Bulirsch 数列**　$\{2, 4, 6, 8, 12, ...\}$

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

このうち，最も関数評価回数が少ないのが harmonic 数列である。室伏ら [9] 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いる。今回実装した solver は，Romberg 数列を用いたものと harmonic 数列を用いたものである。

これらのどれを用いても，同じ段数 $m$ ならば打切り誤差の次数は等しくなる。実際 1 次元 ODE の IVP である

$$\begin{cases} \frac{dy}{dx} &= y \\ y(0) &= 1 \\ \text{積分区間} &: [0,1] \end{cases} \tag{7}$$

に対して，$H = 1/$分割数 とし，10 進 100 桁計算した $T_{88}, T_{99}, T_{10,10}$ の相対誤差を求めると，段数が同じであれば同じ傾きになることが分かる。しかし，誤差の大きさは Romberg 数列を用いた方が小さく出来る (Fig.9 左)。

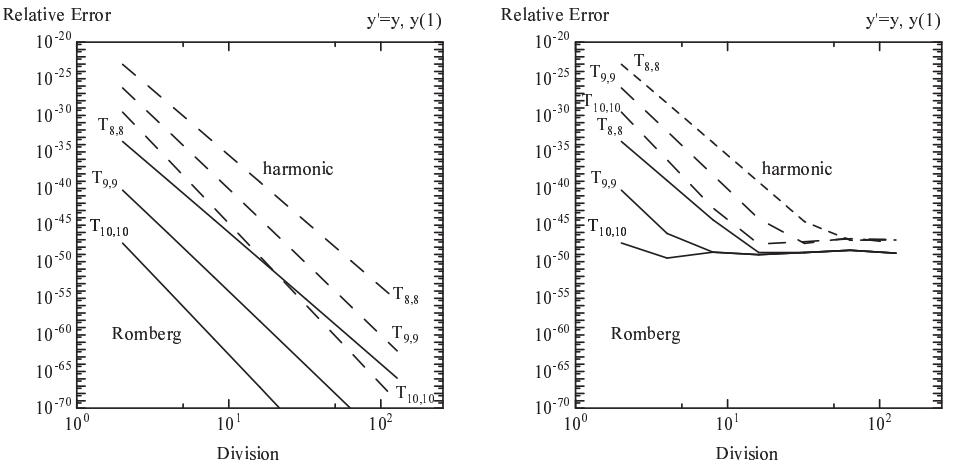


Fig. 9: 補外法の相対誤差 (左:10 進 100 桁, 右:10 進 50 桁)

更に，harmonic 数列については集積丸め誤差が増大するという欠点が指摘されている。これを先の例題を用いて 10 進 50 桁計算してみると，harmonic 数列を用いた場合，10 進 2 桁程度，丸め誤差が増大していることが確認できる (Fig.9 右)。

しかし，多倍長計算においては任意に桁数を調整できるため，2 桁程度の丸め誤差ならば，2 桁増やして計算すればよいということになる。BNCpack, MPIBNCpack で使用している MPFR の場合，IA-32 マシンであれば，32bit 単位で仮数部が割り当てられるため，この程度の桁数の増加は計算時間に影響するほどのものではない。実際に 2 桁増やして計算した例を Fig.10 に示しておく。Romberg 数列を用いたものと同程度の誤差になることが示される。

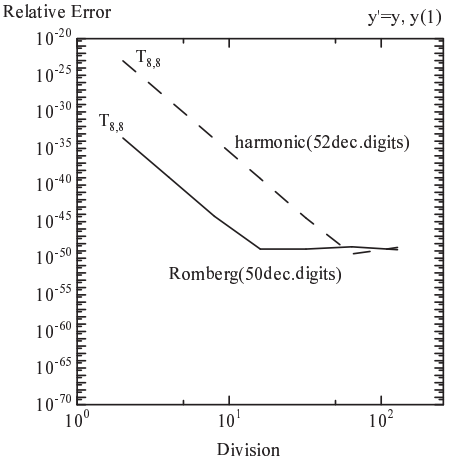


Fig. 10: 補外法の相対誤差 (10 進 50 桁 (Romberg), 52 桁 (harmonic) 計算)

### 4.2 停止則と積分区間制御

補外法における収束判定は，初期系列ではなく，補外計算の部分で行うのが普通である。今回は，ごく単純に

$$\|T_{j,k+1} - T_{jk}\|_\infty \le \varepsilon_R \|T_{j,k+1}\|_\infty + \varepsilon_A \tag{8}$$

となった時に収束したとみなすことにした。ここで $\varepsilon_R$ は相対誤差の大きさを判断する正の実数値であり，この大きさをユーザが指定することによって停止条件が決定される。$\|T_{j,k+1}\|_\infty$ が 0 に近い値になった時には，保険として $\varepsilon_A$ を指定することで停止させる。今回の数値例は全て $\varepsilon_A = 0$ として実行した。

収束判定はステップサイズを調整するためにも使用される。もし，$m$ 段で収束条件を満足しなければ，$H := H/2$ として再度初期系列の計算から実行する。これを収束するまで繰り返す。

収束条件を満足し，次のステップに移動した際には $H := 2H$ として初期系列の計算から実行する。これで収束すればそのまま次のステップへ，そうでなければ $H := H/2$ としてやりなおす。

室伏 [9] は，Romberg 数列を用いた補外法が丸め誤差の蓄積が少ないことを利用して，桁数ギリギリの近似解を得るために $\varepsilon_R = 0$ とすることを提案している。これを「イコール判定」と呼ぶ。このイコール判定を用いた Romberg 数列による補外法のアルゴリズム

が NIM 法である。丸め誤差の蓄積を勘案して，最低限マシンイプシロンの定数倍程度の $\varepsilon_R$ を指定するのが数値計算の常道であるが，NIM 法ではこれが有効に働くことが知られている。

その例として，次の共振問題を取り上げる。

$$\begin{aligned} \frac{d}{dx}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= \begin{bmatrix} y_2 \\ \alpha y_1(-y_1 \sin x + 2y_2 \cos x) \end{bmatrix} \\ \mathbf{y}(0) &= [1\ \alpha]^T \\ \text{積分区間} &: [0, 37] \end{aligned} \tag{9}$$

ここで $\alpha$ は実定数である。解析解は

$$\mathbf{y}(x) = \begin{bmatrix} \frac{1}{1-\alpha \sin x} \\ \frac{\alpha \cos x}{(1-\alpha \sin x)^2} \end{bmatrix} \tag{10}$$

である。これで分かるように，$\alpha$ が 1 に近づくと，どちらの要素も共振点 $(2n+1)\pi/2$ $(n \in \mathbb{Z})$ において急激に増加する。実際，解析解の Taylor 展開を求めてみると係数が急激に大きくなる級数になり，収束半径はごく小さいものであると判断できる。室伏 [9] の数値実験によれば，$\alpha = 0.99999999$ が IEEE754 倍精度で収束するギリギリの線である。従って，このような問題は多倍長計算を用いて行うのがふさわしい。Romberg 数列と harmonic 数列を用いた補外法を用いて，この共振問題を解いた結果を示す。

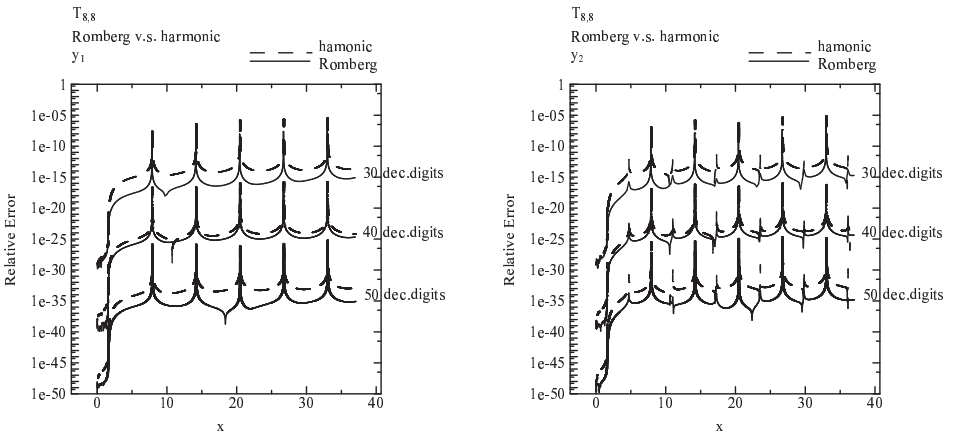


Fig. 11: 共振問題 $y_1$(左) と $y_2$(右) の相対誤差 (10 進 30, 40, 50 桁計算)

まず，10 進 30, 40, 50 桁計算した時の近似解の相対誤差の変化を Fig.11 に示す。ここでは全て $\varepsilon_R = 0$ とし，イコール判定を用いた。この結果から，どちらの補外法において

も前述したステップサイズ制御がうまく働き，積分区間において破綻することなく計算できていることがわかる。また，共振点においては刻み幅制御を行っていても誤差が増大していることも併せて見て取れる。どの精度の計算結果でも，大体 15 桁程度は精度が悪化する傾向がある。この場合，全般的に，harmonic 数列の方が，若干相対誤差が大きくなっている。

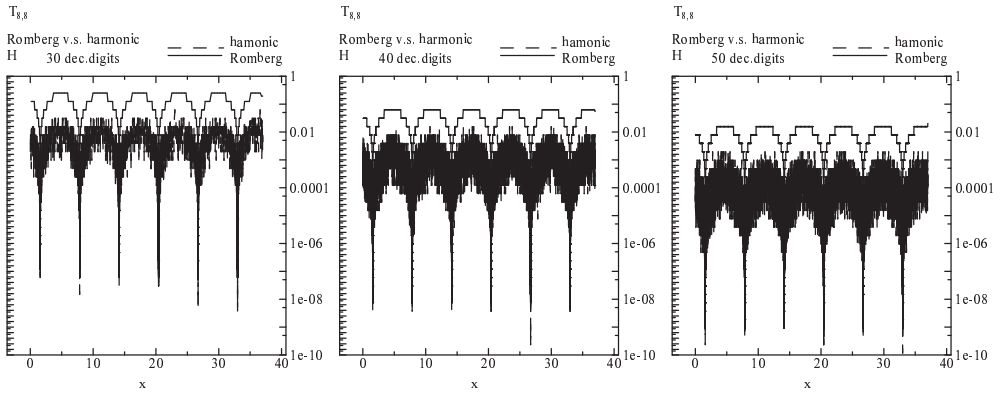


Fig. 12: ステップサイズ $H$ の変化 (10 進 30 桁 (左), 40 桁 (中央), 50 桁 (右))

しかしこの時のステップサイズの変化は，Romberg, harmonic 数列で全く異なる (Fig.12)。この場合，どちらも共振点付近でのみステップサイズが急激に小さくなっているが，それ以外の所では Romberg 数列の方はうまく制御できている一方，harmonic 数列を用いた方はステップサイズ制御がうまくいっておらず，振幅の激しい振動を起こしている。しかも全般的に，harmonic 数列のステップサイズは Romberg 数列のものよりもかなり小さく抑えられてしまっている。このため，本来は計算量が少なくて済む harmonic 数列の補外法の方が，トータルの計算時間は Romberg 数列に比べて圧倒的に増大してしまう。

これは，前述のように，harmonic 数列を用いた補外法の丸め誤差の蓄積量が増大する，という性質が影響していると思われる。イコール判定は，近似解における打切り誤差が丸め誤差以下に抑えられるまでステップサイズを小さくする手法であるため，丸め誤差に関する性質の違いが顕著に現れてしまうのであろう。

では，精度を十分に取って丸め誤差を小さくし，打切り誤差でステップサイズが制御できるようにするとどうなるか？　10 進 50 桁で，$\varepsilon_R = 10^{-25}, 10^{-45}$ とした場合の結果を示す (Fig.13)。

Fig.13 の左のグラフによって近似解の相対誤差は適切に抑えられていることが示される。また，中央及び右のグラフによって，harmonic 数列のステップサイズ制御が正常に実行されていることも分かる。この場合は，若干 harmonic 数列の方がステップサイズが

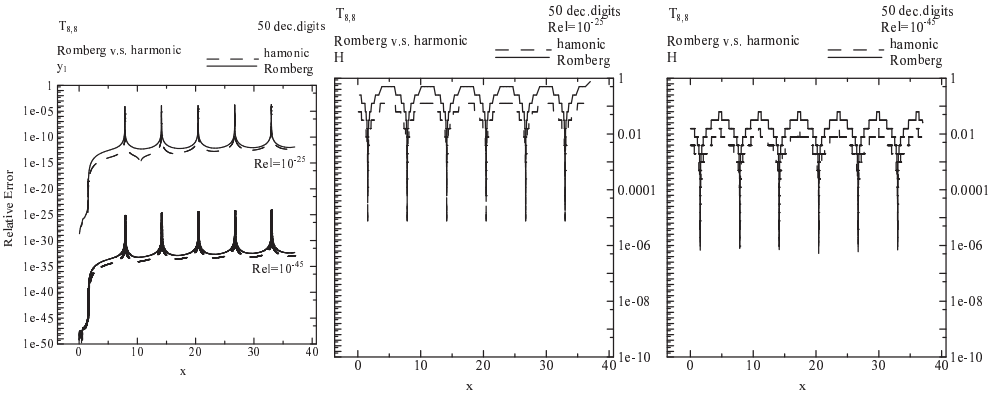


Fig. 13: 共振問題 $y_1$ の相対誤差 (10 進 50 桁計算, 左) とステップサイズ $H$ の変化 (中央:$\varepsilon_R = 10^{-25}$, 右:$\varepsilon_R = 10^{-45}$)

小さくなっているもの，初期系列の計算量が少なくて済んでいる分，高速に計算できている。

以上の結果により，浮動小数点数の精度ギリギリの数値解を求めたいときには Romberg 数列に基づく補外法を用いてイコール判定を行うのがよく，今回のように多倍長計算を用いて丸め誤差の増大を気にすることがない場合は，harmonic 数列に基づく補外法を，打切り誤差に基づくステップサイズの制御を行いつつ実行する方が計算時間が少なくて済む，ということが分かる。

### 4.3 線型 ODE による性能評価

IVP は並列計算に向いていないと言われている [10]。各ステップ毎に近似計算を行い逐次的に計算を進めるため，並列性を持たせるには 1 ステップ内での効率が上がるようアルゴリズムを工夫する必要がある。分散メモリの PC cluster の場合にはベクトル $\mathbf{y}$ を各 PE に分散配置させることが多いが，これを行うと関数 $\mathbf{f}(x, \mathbf{y})$ を計算する前にそれらをまとめて全ての PE にばら撒く必要が出てくる。MPI では Allgather を使うことになるが，ここで並列性が犠牲になり，通信時間のボトルネックが発生する。

従って，もし分散メモリの並列マシンで効果を出そうと思うのであれば，大規模かつ $\mathbf{f}(x, \mathbf{y})$ の計算そのものに並列化の効果がある問題でなければならず，全ての ODE が PC cluster で実行するのに適している訳ではないことになる。ここではその効果が端的に現れると予測される次の大規模線型 ODE を例にベンチマークテストを行うことにする。

$$\begin{aligned} \frac{d\mathbf{y}}{dx} &= A\mathbf{y} \\ \mathbf{y}(0) &= [1\ 1\ \cdots\ 1]^T \\ \text{積分区間} &: [0, 0.001] \end{aligned} \tag{11}$$

ここで実正方行列は

$$A = \begin{bmatrix} n & n-1 & \cdots & 1 \\ n-1 & n-1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

とする。補外法は A 安定ではない解法なので，行列の次元が上がるにつれてステップサイズを小さくする必要が出てくる。ここでは並列化の効果のみを見るため，$n = 128$ の時，$H = 0.001/32$ として固定し，Romberg 数列を使った $T_{88}$ の計算時間を計測した結果を示す (Fig.14)。

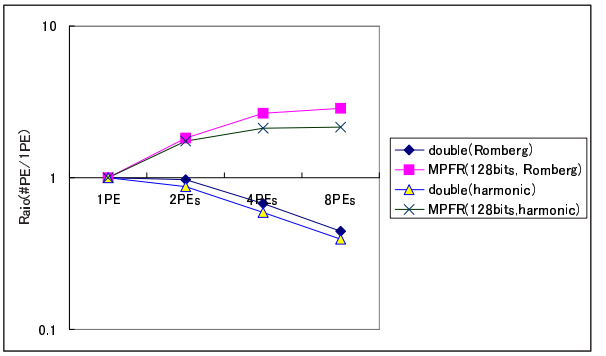


Fig. 14: 線型 ODE の並列化効率 (2 進 128 桁計算)

先に示した Fig.7 の結果から容易に推測できるが，この次元数では IEEE754 倍精度 (double) においては並列行列演算の効果がまだ現れないため，ここでも同様に結果になっている。一方，多倍長計算の方は順当に並列化の効果が現れている。

従って，今回実装した任意精度かつ並列可能な ODE solver は，大規模かつ関数 $\mathbf{f}(x, \mathbf{y})$ に並列化の効果がある問題に対して十分効果を発揮するものと結論付けられる。

## 5. 結論と今後の課題

最初に述べたように，PC cluster を用いて並列分散数値計算可能な環境を構築することはさほど難しいことではないが，分散メモリを自在に使いこなす技法の蓄積についてはまだまだ不十分であると感じる。ことに，今回作成した多倍長計算をベースとした，任意精度数値計算アルゴリズムの実装はごく限られたものしか存在せず，GMP + MPFR, BNCpack，MPIBNCpack というソフトウェアを独力で積み上げなければ実現できないものである。以上のベンチマークテストの結果，これらのソフトウェアが cs-pccluster2 上において有効に働くことが示された。

今後の課題としては，今回構築した任意精度 ODE Solver に関するものと，それを実行する基盤である PC cluster の応用技法に関するものが挙げられる。

### 5.1 任意精度 ODE Solver についての課題

今回実装した任意精度 ODE solver は，A 安定ではない一段法に基づいて作られたものであるため，十分精度のある近似解が必要な場合は有用であるが，あまり近似解の精度を必要としない場合は Stiff ODE に対して多大な計算時間を必要とする。従って，任意精度かつ A 安定な数値計算アルゴリズムを考え出す必要がある。

### 5.2 PC cluster の応用技法に関する課題

今回使用した PC は，擬似的に 2CPU の SMP 環境を提供できる Hyper Threading 機能を備えた，Pentium 4 2.8CGHz を使用しており，最大 22 プロセスを同時に実行することも可能である。しかし，これらは研究用のみならず，毎週 1 回実施される学生実験や，夏季休暇中に行われる高校生向け実験講座，高大一貫実習，そしてセミナーに卒研と，殆ど休むことなく教育用途でも利用されるものである。それでも，長期間に渡って実行しなければならない巨大な計算を取り扱う必要が出てくるため，教育と研究が両立できるような実行体制を確立しなければならない。

従って，長期の計算を実行しつつ教育に支障がないよう，1CPU には 1MPI プロセスのみを実行させるようにし，パイプラインの利用率を上げつつ他のプロセスがストレスなく実行できるように運用したい。その効果については，今後の課題とし，ベンチマークテストを重ねていきたい。

### 謝辞

本研究は，学内研究費の支援を受けて行われた。支援をして頂いた関係者に深く感謝する。また，PC の組み立てを行ってくれた幸谷ゼミ生 8 名の諸君と，助言をして頂いた永坂秀子先生にも感謝申し上げる。

**参考文献**


1) GNU MP, http://swox.com/gmp/
2) MPFR Project, http://www.mpfr.org/
3) BNCpack マニュアル, http://na-inet.jp/na/bnc/bnc.pdf
4) MPIBNCpack マニュアル, http://na-inet.jp/na/bnc/mpibnc.pdf
5) 石川　裕　他,「Linux で並列計算しよう」, 共立出版, 2002.
6) 今井仁司,「応用解析における多倍長計算」, 数学 Vol.55, No.3, 2003.
7) 後藤弘茂の Weekly 海外ニュース, http://pc.watch.impress.co.jp/docs/2003/1114/kaigai045.htm
8) P.S.Pacheco・秋葉博　訳,「MPI 並列プログラミング」, 培風館, 2001.
9) 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.
10) E.Harier, S.P.Nørsett, G.Wanner, Solving Ordinary Differential Equations I (Second Ed.), Springer-Verlag, 1991.
11) MPICH, http://www-unix.mcs.anl.gov/mpi/mpich/
12) 幸谷智紀,「PC cluster を用いた多倍長数値計算ライブラリの並列分散化」, Linux Conference 2003.
13) 幸谷智紀,「CG 法における最短計算時間を探索する試み」, SWoPP 2003.
14) 幸谷智紀,「PC cluster を用いた多倍長数値計算ライブラリの並列分散化」, FIT 2003.
15) 幸谷智紀,「A Tutorial of BNCpack & MPIBNCpack」, http://na-inet.jp/tutorial/
16) 幸谷智紀,「Vine Linux による PC cluster の構築」, http://na-inet.jp/na/bnc/mpipc.pdf
17) 幸谷智紀,「Vine Linux による PC cluster の構築 Version 2」, http://na-inet.jp/na/bnc/mpipc2.pdf