計算機アーキテクチャ　147－29
ハイパフォーマンス　89－29
コンピューティング
（2002．3．8）

# GNU MPを用いたODE Solverのパフォーマンスについて

幸谷智紀 *　満田賢一郎 **　鈴木千里 *
* 静岡理工科大学　**(株) システム計画研究所

2002年3月7日-8日
HOKKE-2002

**概 要**

本講演では，多倍長計算ライブラリ GNU MP(GMP) を用いた ODE Solver のパフォーマンスについて，それを構築するまでの背景も含めて述べる。

## On Performance of ODE Solver using GNU MP

Tomonori Kouya*　Ken-ichiro Mitsuda**　Chisato Suzuki*
*Shizuoka Institute of Science and Technology
**Research Institute of System Planning, Inc.

**Abstract**

In this paper, we describe the performance of our ODE solver using GNU MP, one of the fastest free multi-precision libraries, and background and reason why we build it.

## 1　初めに

一般に数値計算は，ハードウェアで直接処理出来る，有効桁数が固定された浮動小数点数を使用して行われる。それ以上の精度が要求される場合のみ，ソフトウェアで実現される任意精度の浮動小数点演算（多倍長計算）が実行される。その理由としては

1. 計算速度が低下する
2. 現行の精度で間に合っている
3. 必要となる記憶領域が増える

ということが挙げられる。従って，多倍長計算は数値計算における一種のオプションとして考えられている。

歴史を降り返ってみると，ハードウェアの高機能化に合わせて，サポートされる浮動小数点数の有効桁数は増えてきている。ユーザにとっては，計算速度があまり変化しないのであれば，精度は高いに越したことはない。そこで，多倍長計算が利用できるようになってくると，次のような疑問が浮かんでくる。

1. 浮動小数点数の有効桁数に鋭敏なアルゴリズムは，多倍長計算を利用することで，現在最良とされている頑健なアルゴリズムよりも速度面で凌駕する可能性があるのではないか？

2. 悪条件問題において，浮動小数点数の精度が数値解の精度に与える影響はどの程度のものか？　特に最良の精度との関係を図示するとどのようなものになるのか？

これらを理論的に追求するのは困難であると思われるが，数値実験を網羅的に行うことである程度の見積もりを得ることは可能であろう。そのための土台として，我々は GNU MP(GMP) を用いて多倍長計算が実行できる数値計算ライブラリの雛形，BNCpack を作成した。これらの点をふまえて今回は BNCpack で実装しつつある ODE Solver のパフォーマンスについて考えてみたい。

## 2　ODE Solver のアルゴリズムについて

ここで考える常微分方程式の初期値問題を

$$\begin{cases} \frac{dy}{dx} & = & f(x, y) \\ y(x_0) & = & y_0 \in R^m \end{cases} \tag{1}$$

とする。積分区間を $[x_0, x_{\mathrm{end}}]$ とすると，標準的な数値解法では，離散化を行い，初期値から分割した積分区間を逐次近似解を計算しつつ進んでいくことになる。ここでは簡単のため，積分区間は $h = (x_{\mathrm{end}} - x_0)/n, x_i = x_0 + ih$ に $n$ 等分割されているものと考える。

$n \longrightarrow \infty$ となる時，近似値 $y_n$ は $O(h^p)$ で解 $y(x_n)$ に収束する。この時，このアルゴリズムの次数は $p$ である。現在多く実装されているアルゴリズムはこの $p$ が固定のものと可変なものに分類される。

**固定次数** —

- **一段法** — 陽的・陰的 Runge-Kutta 法など
- **多段法** — PC 法など

**可変次数** —

- **補外法** — NIM 法など
- **Taylor 展開法**

収束が早まるため，次数は高い方が望ましいが，計算量は増える。可変次数のアルゴリズムは同次数の固定次数アルゴリズムよりも一般には計算量が多いとされている。関数 $f(x, y)$ の評価回数も少ない方が好ましいが，一段法は多段法よりも多く，一段法をベースとする補外法は更に多くなる。

アルゴリズムを比較する重要な指標として，安定性がある。様々な安定性の定義がなされているが，A 安定性と呼ばれる指標については，今回のように浮動小数点数の有効桁数

に依存して決まる最良の近似解の精度をベースに考える場合はあまり重要視する必要はない，と我々は考えている。

実際，田中 [7] が提唱している陽的 Runge-Kutta 法用の丸め誤差評価値は，丸め誤差が近似解の精度に影響する時には $h << 1$ である，ということが前提となっている。逆に言えば $h << 1$ でなければ丸め誤差の影響も目立つ物ではないし，この評価値もあまり意味がないということである。以上の事を踏まえてると，多倍長計算を導入せざるを得ない程の悪条件問題にアルゴリズムを適用した時，A 安定性というものがどれ程の影響を与えるかは今の時点では判然としない。また影響があるにしろ，A 安定ではない陽的なアルゴリズムも，次数を上げることで安定性を増すことが知られている [3][6]。

同じ有効桁数の浮動小数点数を使う場合，同次数のアルゴリズムどうしを，計算時間，近似解の精度という観点で比較する意味はあると思われる。

## 3 Rössler Model について

Rössler Model は

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -(y+z) \\ x + 0.2y \\ 0.2 + z(x-\mu) \end{bmatrix} \tag{2}$$

という形で表される。これは最初に発見された Lorenz Model

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sigma x + \sigma y \\ -xz + rx - y \\ xy - bz \end{bmatrix} \tag{3}$$

を単純化したものとして提唱された。解曲線が Lorenz Model が羽を広げた蝶のように 2 つのリングが捻れて繋がっているのに対し，Rössler Model はそれが一つしかない。

数値計算の例題としてこの 2 つを取り上げる時には，丸め誤差の影響で，積分区間が広がるにつれて近似解の精度が悪くなるサンプルとして語られる。その明確な理由を述べたものは寡聞にして知らないが，向川 [5] は，Chaos の定義に使用される Lyapunov 指数との関連について述べている。Wilkinson 流の後退誤差解析の観点から見れば，誤差が増大する理由の説明になっていると思われる。但し，数値計算のアルゴリズムの流れの中でどのように誤差が増大するのか，そのメカニズムについての説明は別に行われる必要があるだろう。

## 4 GMP と ODE Solver の時間計測

GNU MP は LGPL に基づいて配布されている多倍長計算ライブラリである [1]。2002 年 2 月現在で，最新バージョンは 4.0.1 になっているが，現在流通している Linux distribution に入っているものはまだ 3.x 系列が多いようである。Ver.3 系とは，C 言語の関数インターフェースの互換性は保ちつつ，入出力関数の追加，要望の多かった C++クラスインターフェースが同封されるようになった点が大きな変化である。我々の ODE Solver は β バージョンではあるが，BNCpack[2] に取り込んであり，元々は GMP が Ver.2.x であった時に

構築されたものである。GMP のドキュメントに明示してある通り，ごく標準的な，GMP の浮動小数点数演算関数のみ使用してあるため，Ver.3.x でも Ver.4.x でも問題なく動作することが確認されている。

GMP の演算ルーチンそのものの改良も行われたようであるが,

**CPU** Intel Pentium III 800MHz

**RAM** 256MB

**OS** RedHat 7.2J

**C compiler** GCC 2.96

**GMP** Ver.3.1.1, Ver.4.0.1 (`./configure; make; make install` でインストール)

という環境下で比較したところ，劇的な速度向上は見られなかった。従って，今回の GMP のバージョンアップは，ユーザーサイドの使い勝手を向上させたという面が大きいと言える。

ベンチマークテストの詳細については講演時に述べる。

## 参考文献


[1] The GNU MP Home Page, `http://swox.com/gmp/`

[2] BNCpack, `http://member.nifty.ne.jp/tkouya/na/bnc/`

[3] 大林 昇, 常微分方程式の高精度数値解法の研究, 博士論文, 1995.

[4] 幸谷智紀, 常微分方程式および固有値問題における高精度計算法の研究, 博士論文, 1997.

[5] 向川 均，カオスと天気予報, pp.126–pp.132 (合原・相沢 編, カオス研究の最前線, 1999).

[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, 1998.

[7] 田中 正次 他, 8 段 6 次陽的 Runge-Kutta 法の最適化について, 情報処理学会論文誌, Vol.34, No.1.