数理解析研究所講究録
第1638巻 2009年 159-168

常微分方程式の精度保証における数式処理について
On symbolic computation in numerical verification for ODEs

山本 野人 (電気通信大) ・松田 望 (電気通信大情報工学専攻)
Nobito Yamamoto, Nozomu Matsuda,
Department of Computer Science, The University of Electro-Communications
FAX: 042-443-5349 E-mail: yamamoto@im.uec.ac.jp

1 Introduction

本稿では、常微分方程式の精度保証法に現れる繁雑ではあるが自動化が可能な計算について考察する。

近年、数値解析および数学解析のさまざまな局面で、精度保証付き数値計算を利用した結果が見られるようになった。しかしながら、多くの人が精度保証を利用する研究を望みながらもなかなか手を出しにくい、ということも良く耳にする。精度保証用のソフトウェアとしては、Unix マシン用の PROFIL や MATLAB 上で稼働する INTLAB という使いやすいものがあり、これを用いれば手軽に精度保証付き計算が可能であると著者らは主張してきたが、精度保証技術の普及のためにはより応用に則したソフトウェア開発の必要があると感じられる。

常微分方程式初期値問題の精度保証法については、Lohner 法がもっとも広く使われている。これについては、Lohner 自身が開発した AWA というソフトウェアパッケージがあるが、我が国ではそれほど普及していない。力学系の研究者などは、PROFIL などを用いて自身でプログラムを組むことが多いようである。Lohner 法は Taylor 展開を利用した誤差解析に基づく方法なので、連立あるいは高階の微分方程式に対しては、ベクトル値関数の高階微分項の処理がかなり繁雑となる。AWA などでは、この部分を自動微分で処理するようであるが、実行速度の点で問題があると思われる。

著者らが開発している中尾理論 [7] に基づく常微分方程式の精度保証法 [2] においては、多項式補間を用いる関係上、補間の次数が高次にわたるとプログラミングに至るまでの数式処理がやはり繁雑になる。また、精度保証の過程で Newton 型反復を用いる際には、区間巾の過剰な拡大を避けるために、一次項をまとめる作業をプログラミングの前に行っておく必要がある。これも手計算で行うと面倒なものである。

精度保証技術の普及を目指すには、区間演算ソフトの提供だけでなく、これらの繁雑さを軽減するための工夫を使いやすい形で提示することが重要であろう。そこで本稿では、特に常微分方程式の精度保証法に現れる式変形のうち手計算では繁雑になることが多い部分を例示し、これを数式処理ソフトと区間演算ソフトの併用によって処理する方法を紹介する。

第2章では常微分方程式境界値値問題の精度保証法について概観する。これは Lohner が 1987 年に出版した論文 [3] に基づくものであるが、オリジナルの方法では精度保証の過程に、

連立一次方程式として区間行列を係数とするものが現れる。これは Lohner 自身が指摘しているように計算コストの点で不利であるので、ここではこの部分を準 Newton 法に置き換え、係数行列に区間値が現れないようにする方法を紹介する。なお、オリジナルの方法は、よく知られている初期値問題に対する Lohner 法とは異なるものである。

第 3 章では、上記の方法のうち、プログラミングにかかる前に数式変形によって処理しなければならない部分を説明し、これを MATLAB 上の数式処理機能 Symbolic Math Toolbox と精度保証付き区間演算ソフトウェア INTLAB によって半自動的に処理するアイデアについて述べる。

2 常微分方程式境界期値問題の精度保証法

次のような正規形の境界値値問題に対する精度保証法を取り上げよう。

$$\begin{cases} \dfrac{d}{dt}\mathbf{u}(t) &= \mathbf{f}(t,\mathbf{u}), \qquad a<t<b, \\ \mathbf{r}(\mathbf{u}(a),\mathbf{u}(b)) &= 0 \end{cases}$$

ここに $\mathbf{u}, \mathbf{f}, \mathbf{r}$ は n 元ベクトル値関数であり、$\mathbf{f}, \mathbf{r}$ は $\mathbf{u}$ について 1 階微分可能であるとする。境界条件を表す関数 $\mathbf{r}$ は線形でなくても構わない。

時間分点を、

$$a = t_0 < t_1 < \cdots < t_{m-1} < t_m = b, \qquad m \in \mathbf{N}$$

と取る。次に m 個の n 元ベクトルを考え、これらを縦に並べたベクトルを

$$\mathbf{s} = (\mathbf{s_0}, \mathbf{s_1}, \cdots, \mathbf{s_{m-1}}, \mathbf{s_m})^T$$

と置く。また、

$$\mathbf{u}(t_{k+1}; t_k, \mathbf{s_k})$$

を $\mathbf{u}(t_k) = \mathbf{s}_k$ を初期値として微分方程式を $t = t_{k+1}$ まで解いたときの真の解とする。

実際の計算の過程では、これらを精度保証付きで算定する必要がある。そのためには初期値問題に対する Lohner 法を用いるのが通常であるが、ここでは 1 ステップの計算なので、いわゆる Wrapping Effect についてはそれほど考慮しなくてもよいと思われる。そこで、より簡便な方法として、Lohner 法のもととなる Taylor 展開に基づく精度保証法を用いることにする。これについては後述する。

与えられた $\mathbf{s}$ に対し、次のように $F(\mathbf{s})$ を定める。

$$F(\mathbf{s}) := \begin{bmatrix} \mathbf{u}(t_1; t_0, \mathbf{s_0}) - \mathbf{s_1} \\ \mathbf{u}(t_2; t_1, \mathbf{s_1}) - \mathbf{s_2} \\ \vdots \\ \mathbf{u}(t_m; t_{m-1}, \mathbf{s_{m-1}}) - \mathbf{s_m} \\ r(\mathbf{s_0}, \mathbf{s_m}) \end{bmatrix}$$

方程式 $F(\mathbf{s})=\mathbf{0}$ の解 $\mathbf{s}$ が存在すれば、境界値問題の解 $\boldsymbol{u}(t)$ も存在し、$\mathbf{s_j}$ が $\boldsymbol{u}(t_j)$ の値を与える。この $\mathbf{s}$ についての方程式を解くために、オリジナルの方法では Newton 法を用いている。しかし、この方法では係数行列が区間行列となる連立一次方程式を解かなくてはならず、計算時間が増大する。そこで、ここでは準 Newton 法を用いて次のように定式化する。

まず、$\mathbf{s}$ をひとつ取って固定し、これを $\tilde{\boldsymbol{s}}$ とする。$\dfrac{\partial F}{\partial \mathbf{s}}(\tilde{\boldsymbol{s}})$ に相当するものを算定して、これを $F'(\tilde{\boldsymbol{s}})$ とする。すなわち、

$$F'(\tilde{\boldsymbol{s}}) := \begin{bmatrix} V_0(t_1) & -I & 0 & 0 & \cdots & 0 \\ 0 & V_1(t_2) & -I & 0 & \cdots & 0 \\ 0 & 0 & V_2(t_3) & -I & \cdots & 0 \\ \vdots & \cdot & \cdot & \cdot & \cdot & \vdots \\ 0 & \cdots & \cdots & 0 & V_{m-1}(t_m) & -I \\ \dfrac{\partial r}{\partial \mathbf{s_0}}(\tilde{\boldsymbol{s}}_\mathbf{0}, \tilde{\boldsymbol{s}}_\mathbf{m}) & 0 & \cdot & \cdot & 0 & \dfrac{\partial r}{\partial \mathbf{s_m}}(\tilde{\boldsymbol{s}}_\mathbf{0}, \tilde{\boldsymbol{s}}_\mathbf{m}) \end{bmatrix}$$

ただし、$V_k(t_{k+1})$ は $n \times n$ 行列

$$\frac{\partial \mathbf{u}(t_{k+1}; t_k, \tilde{\boldsymbol{s}}_\mathbf{k})}{\partial \mathbf{s_k}}$$

に相当し、

$$\begin{cases} \dfrac{d}{dt}V_k(t) &= \dfrac{d\mathbf{f}}{d\mathbf{u}}(\mathbf{u}(t; t_k, \tilde{\boldsymbol{s}}_\mathbf{k}))V_k(t), \qquad t \in [t_k, t_{k+1}] \\ V_k(t_k) = \mathbf{I} \end{cases}$$

を近似的に解いて求めるものとする。ここに $\mathbf{I}$ は $n \times n$ 単位行列である。

これらを用いて、

$$T(\mathbf{s}; \tilde{\boldsymbol{s}}) \;:=\; (F'(\tilde{\boldsymbol{s}}))^{-1}(F'(\tilde{\boldsymbol{s}})\mathbf{s} - F(\mathbf{s}))$$

と定め、$\mathbf{s}$ についての不動点方程式

$$\mathbf{s} \;=\; T(\mathbf{s}; \tilde{\boldsymbol{s}})$$

を精度保証付きで解く。$\tilde{\boldsymbol{s}}$ はパラメータとみなす。この方程式は $\mathbf{R}^{n(m+1)}$ 上の方程式であるので、Brouwer の不動点定理を用いることが出来て、

$$T([\mathbf{s}]; \tilde{\boldsymbol{s}}) \;\subset\; [\mathbf{s}]$$

を満たす $n(m+1)$ 元区間ベクトル $[\mathbf{s}]$ を見出すことができれば、$[\mathbf{s}]$ の中に真の解を持つことがわかる。

なお、区間演算を行う上では、T に現れる $F'(\tilde{\boldsymbol{s}})[\mathbf{s}] - F([\mathbf{s}])$ の項は、区間ベクトル $[\mathbf{s}]$ の一次項について整理する必要がある。このことについては次章で詳説する。

以上を整理して手順を示そう。

1. 近似解 $\tilde{\boldsymbol{u}}$ を算定する。これについてはシューティング法を用いるのが一般である。また、$\mathbf{s_0} = \tilde{\boldsymbol{u}}(t_0)$, $\mathbf{s_1} = \tilde{\boldsymbol{u}}(t_1)$, $\cdots$, $\mathbf{s_{m-1}} = \tilde{\boldsymbol{u}}(t_{m-1})$, $\mathbf{s_m} = \tilde{\boldsymbol{u}}(t_m)$ と取って

$$[\mathbf{s}]^0 \quad := \quad (\mathbf{s_0}, \mathbf{s_1}, \cdots, \mathbf{s_{m-1}}, \mathbf{s_m})$$

を Newton 反復の初期値とする。これは区間巾が 0 の区間ベクトルとみなす。

2. 整数 $\nu = 0, 1, 2, \cdots$ に関する反復を考えよう。$[\mathbf{s}]^\nu$ に対して、$\tilde{\boldsymbol{s}}^\nu \in [\mathbf{s}]^\nu$ を取り、

$$T([\mathbf{s}]^\nu; \tilde{\boldsymbol{s}}^\nu) \quad \subset \quad [\mathbf{s}]^\nu$$

をチェックする。T を構成する F については、微分方程式を 1 ステップだけ解いたときの真の解 $\mathbf{u}(t_{k+1}; t_k, [\mathbf{s_k}]^\nu)$ を精度保証付きで算定する必要がある。その結果、F は区間値ベクトルになる。一方、行列 $F'(\tilde{\boldsymbol{s}})$ の算定については、$V_k(t_{k+1})$ および $\mathbf{u}(t_{k+1}; t_k, \tilde{\boldsymbol{s}}_\mathbf{k})$ としてその近似値を用いてよく、区間演算を使う必要はない。この行列が区間になっていないことが、オリジナルの Lohner による方法との違いである。包含関係が成立すれば精度保証に成功したので、停止する。

3. 上の包含関係が満たされなければ、$T([\mathbf{s}]^\nu; \tilde{\boldsymbol{s}}^\nu)$ の区間巾を少し大きくしたものを $[\mathbf{s}]^{\nu+1}$ とし、その中心値を $\tilde{\boldsymbol{s}}^{\nu+1}$ として反復を継続する。

Taylor 展開法による初期値問題の精度保証法について

ここでは、$\mathbf{u}(t_{k+1}; t_k, [\mathbf{s_k}])$ を精度保証つきで算定するための方法として Taylor 展開を利用するものについて述べておく。Taylor 展開法の打ち切り誤差の評価のためには、各ステップ間 $[t_k, t_{k+1}]$ で真の解を包含する区間ベクトル値関数 $[\mathbf{U}]$ が必要になる。

ステップ巾を $h = t_{k+1} - t_k$, $t = t_k$ での解の値 $\mathbf{s_k}$ を含む区間を $[\mathbf{s_k}]$ とする。$\mathbf{u}$ の積分方程式を導いて不動点定理を適用すれば、次が成立するとき $[\mathbf{U}]$ は $\mathbf{u}(t)$ $(t_k \le t \le t_{k+1})$ を含むことがわかる。

$$[\mathbf{U}] \quad \supset \quad [\mathbf{s_k}] + [0, h]\mathbf{f}([t_k, t_{k+1}], [\mathbf{U}]).$$

これを満たす区間値の定数関数を $[\mathbf{U}]$ として用いることにしよう。以下これを $[\mathbf{U}_{k+1}]$ と記す。さらに、$\mathbf{u}(t_{k+1}; t_k, [\mathbf{s_k}])$ を包み込む $[\mathbf{U}_{k+1}]$ よりも精密な区間ベクトル $[\mathbf{u}_{k+1}]$ を以下のように求める。

$$\begin{aligned}[\mathbf{u}_{k+1}] \quad &= \quad [\mathbf{s_k}] + \sum_{j=1}^{(p-1)} h^j \mathbf{f}^{(j)}(t_k, [\mathbf{s_k}]) \\ &\quad + h^p \mathbf{f}^{(p)}([t_k, t_{k+1}], [\mathbf{U}_{k+1}]). \end{aligned} \tag{1}$$

ただし、

$$\begin{aligned}\mathbf{f}^{(1)} \quad &= \quad \mathbf{f}, \\ \mathbf{f}^{(j+1)} \quad &= \quad \frac{1}{j+1}\left(\frac{\partial \mathbf{f}^{(j)}}{\partial t} + \frac{\partial \mathbf{f}^{(j)}}{\partial \mathbf{u}}\mathbf{f}\right). \end{aligned} \tag{2}$$

この式は、関数 $\mathbf{f}$ がベクトル値であるので、j が大きくなるにつれてかなり複雑なものとなる。

3 計算の難所と数式処理による対策

3.1 手計算での処理における難所

前章で記した境界値問題の精度保証法は、以下のような作業量の多い式変形計算を含んでいる。

- 1 ステップの初期値問題を解き、$\mathbf{u}(t_{k+1}; t_k, [\mathbf{s_k}])$ を精度保証付きで算定する必要がある。これには Taylor 展開法を用いるが、前述したとおり、(2) の算定はかなり面倒である。
- 区間演算を行ううえでは、T に現れる $F'(\tilde{\boldsymbol{s}})[\mathbf{s}] - F([\mathbf{s}])$ の項は、区間ベクトル $[\mathbf{s}]$ の一次項について整理しなければならない。この作業は (1) 式まで考慮して行うので、相当に繁雑となる。

二つめの作業を行う理由は次のとおりである。

区間を変数とするこのような計算では、しばしば区間の巾の過剰な増大が現れる。これは、区間変数に関しては分配則が成り立たないことに起因するものであることが多い。特に区間に対する非線形関数の値域の包含は、しばしば大きな過大評価となる (dependency problem と呼ばれる)。これを軽減するための伝統的な手法は平均値形式 [6] とそのバリエーションであるが、Taylor Model も有力な方法である [4]。

平均値形式とは、以下のようなものである。

区間ベクトル $[\boldsymbol{v}]$ についての関数 g の値域 $\{\, g(\boldsymbol{v}) \mid \boldsymbol{v} \in [\boldsymbol{v}] \,\}$ を包含する区間を $[g([\boldsymbol{v}])]$ と書く。このような区間のひとつとして次式の右辺で得られるものがある。

$$[g([\boldsymbol{v}])] \quad \subset \quad g(\hat{\boldsymbol{v}}) + [g'([\boldsymbol{u}])]([\boldsymbol{u}] - \hat{\boldsymbol{v}}),$$

ここに $\hat{\boldsymbol{v}}$ は $[\boldsymbol{v}]$ に含まれる任意のベクトルである。この区間包囲を $g([\boldsymbol{v}])$ の $\hat{\boldsymbol{v}}$ における平均値形式と言う。

ここでは、

$$F'(\tilde{\boldsymbol{s}})[\mathbf{s}] - F([\mathbf{s}]) \quad = \quad \begin{bmatrix} V_0(t_1)[\mathbf{s_0}] - \mathbf{u}(t_1; t_0, [\mathbf{s_0}]) \\ V_1(t_2)[\mathbf{s_1}] - \mathbf{u}(t_2; t_1, [\mathbf{s_1}]) \\ \vdots \\ V_{m-1}(t_m)[\mathbf{s_{m-1}}] - \mathbf{u}(t_m; t_{m-1}, [\mathbf{s_{m-1}}]) \\ \dfrac{\partial \boldsymbol{r}}{\partial \mathbf{s_0}}(\mathbf{s_0}, \mathbf{s_m})[\mathbf{s_0}] + \dfrac{\partial \boldsymbol{r}}{\partial \mathbf{s_m}}(\mathbf{s_0}, \mathbf{s_m})[\mathbf{s_m}] - \boldsymbol{r}([\mathbf{s_0}], [\mathbf{s_m}]) \end{bmatrix}$$

に (1) を代入したものに平均値形式を適用することになる。その繁雑さたるや想像に難くないであろう。

以上の様な繁雑な計算が関っていては、精度保証の方法と区間演算ソフトを手にしただけで気軽にこれを行なうことに躊躇するのも無理はない。そこで、数式処理によって半自動的にこの処理を行ない、精度保証のプログラムコードを生成することを考えよう。本稿では、MATLAB 上の数式処理機能 Symbolic Math Toolbox と精度保証付き区間演算ソフトウェア INTLAB を合わせて使う。なお、読者の便を考慮して、MATLAB のプログラムコードを例示する。

3.2 Symbolic Math Toolbox と区間演算

Symbolic Math Toolbox は、MATLAB 上で数式処理と可変精度演算を行うためのパッケージである。Symbolic Math Toolbox によって様々な数式処理が可能になるが、ここでは本稿で使用する関数の微分について簡単な例を示す。なお、以下では混乱を避けるために、数学的な意味での関数を「関数」、プログラムのサブルーチンとして動作する関数を「コマンド関数」と呼ぶことにする。

Symbolic Math Toolbox での数式処理は、「シンボリックオブジェクト」に対して行われる。新しいシンボリックオブジェクト x, y を宣言するには、以下のように syms 命令を用いる。

```
>> syms x y
```

シンボリックオブジェクトに対する演算の結果も、シンボリックオブジェクトとなる。以下では、関数 $f(x, y)$ を定義している。

```
>> f = exp(x) * (sin(x) + cos(y))
f =
exp(x)*(cos(y) + sin(x))
```

コマンド関数 diff によって関数の微分が行える。以下では、f を x について微分している。

```
>> f1 = diff(f, x)
f1 =
exp(x)*cos(x) + exp(x)*(cos(y) + sin(x))
```

同様に高階微分も可能である。f を x について 2 階微分するには、以下のようにする。

```
>> f2 = diff(f, x, 2)
f2 =
2*exp(x)*cos(x) - exp(x)*sin(x) + exp(x)*(cos(y) + sin(x))
```

f2 の式は簡略化できる。式の簡略化にはコマンド関数 simplify を用いる。

```
>> f2 = simplify(f2)
f2 =
exp(x)*(2*cos(x) + cos(y))
```

シンボリックオブジェクトに値を代入して計算を行うには、通常はコマンド関数 subs を用いる。しかし、subs は INTLAB の区間オブジェクトに対応していない。そこで、ここでは式に現れる変数を代入する値で上書きしてから、コマンド関数 eval で計算するという方法を取る。以下では、$x = 1, y = 2$ のときの f2 の値を計算している。

```
>> x = 1;
>> y = 2;
>> eval(f2)
ans =
   1.806183496074957
```

ここで、代入する値として INTLAB の区間オブジェクトを用いれば、Symbolic Math Toolbox で求めた式を精度保証付きで計算できる。

```
>> x = intval(1);
>> y = intval(2);
>> eval(f2)
intval ans =
[   1.80618349607495,   1.80618349607496]
```

ただし、ここで式全体が正しく精度保証付きで計算されているかどうかに注意する必要がある。例えば、

```
>> syms x
>> f = (6 * x + 1) / 3
f =
2*x + 1/3
```

のような場合、x に区間オブジェクトを代入して f の値を計算しても、1/3 の部分は精度保証なしの近似計算が行われる。このような場合、例えば、

```
>> syms x a
>> f = (6 * x + a) / 3
f =
a/3 + 2*x
>> a = intval(1);
```

のようにすることで、正しく精度保証付き計算が行える。

シンボリックオブジェクトに対する区間ベクトルの代入

前述したように、 Symbolic Math Toolbox と INTLAB を組み合わせて使う場合、シンボリックオブジェクトに対する値の代入が、通常の方法では行えない。このため、シンボリックオブジェクトに対して区間ベクトルを代入する場合には、さらに工夫が必要である。この代入を行うための、以下のようなコマンド関数を作成した。ただし、この方法はシンボリックオブジェクト内の変数名が一定の規則に従っている場合にのみ可能な、やや一般性を欠くものである。

```
function y = subsIntval(f, name, value)
% y = subsIntval(f, name, value)
%
% f      : sym 関数
% name   : f 内に現れるベクトル変数名
%          これが例えば x なら、'x' のようにクォテーションで括って与える。
%          ただし、その要素名は x1, x2, ... のように名付けられているとする。
```

```
% value : intval ベクトル
% y     : f(value) の値

  for i = 1 : length(value)
    eval(sprintf('%s%d = value(%d);', name, i, i));
  end

  y = eval(f);
```

3.3 Taylor展開法の生成

Symbolic Math Toolbox と INTLAB を組み合わせて、与えられた f に対して式 (1) を計算するプログラムは、以下のようになる。

$f^{(j)}$ の計算時に t と u を同時に代入する必要があり、先述の subsIntval では対応できないため、新たに専用のコマンド関数 evalFj を作成した。なお、 jacobian は、Jacobi 行列を求める Symbolic Math Toolbox のコマンド関数である。

```
y = sk;
hj = h;
fj = f;

for j = 1 : p - 1
  y = y + hj * evalFj(fj, tk, sk);

  hj = hj * h;
  fj = (diff(fj, t) + jacobian(fj, u) * f) / (j + 1);
end

y = y + hj * evalFj(fj, infsup(inf(tk), sup(tk1)), Uk1);
```

ここに、

```
function y = evalFj(fj, t, u)
  for i = 1 : length(u)
    eval(sprintf('u%d = u(%d);', i, i));
  end

  y = eval(fj);
```

この例のように、代入については、場合によっては個別に対応する必要がある。

3.4 平均値形式

与えられた区間 $[v]$ と関数 f について、$\mathbf{f}([v])$ の平均値形式による区間包囲を求めるプログラムは、以下のようになる。

```
function y = meanVec(f, u, nameU, v)
% y = meanVec(f, u, nameU, v)
%
% u       : sym ベクトル
% f       : u のベクトル値関数
% nameU   : u の各要素の名前 ( u1, u2, ... なら 'u' )
% v       : u に代入する intval ベクトル
% y       : f(v) の平均値形式による区間包囲

  midV = mid(v);

  j = jacobian(f, u);

  % f(mid(v)) を精度保証付きで計算する。
  y = subsIntval(f, nameU, intval(midV));

  % u に区間ベクトルを代入し、残りの部分を精度保証付きで計算する。
  y = y + subsIntval(j, nameU, v) * (v - midV);
```

3.5 使用法

以上のプログラムを前章で説明した境界値問題の精度保証法に利用するには、つぎの手順を踏めば良い。

1. $[\mathbf{s}]$ を構成するベクトル $[\mathbf{s_k}]$ を用いて、前章で述べた $[\mathbf{U}_k]$ を算定する。
2. これを (1) に代入したものを上述のプログラムによって生成し、さらに $F'(\tilde{s})[\mathbf{s}] - F([\mathbf{s}])$ を計算するプログラム (m-file) を作成する。
3. 次に $F'(\tilde{s})[\mathbf{s}] - F([\mathbf{s}])$ に対する平均値形式による区間包囲を計算するプログラムコードを生成する。
4. これらを利用して精度保証のためのプログラムを作成して実行する。

これを利用すれば、精度保証法の実装がより容易になるはずである。なお、実際の問題に対する数値例などは別の機会に譲ることとしたい。

4 おわりに

近年、区間演算のための強力な計算技術として多項式の数式処理過程を含む Taylor Model 法が現れ、さまざまな問題に適用されつつある [1]。このことは、本稿が関連する RIMS 研究集会「数値解析における理論・手法・応用」における講演「常微分方程式の精度保証：Taylor Model 法の中尾理論への導入」で論じた。

この方法はシンボリックな式変形を本質的に用いるものであり、数式処理を組み込んだ精度保証用ソフトウェアパッケージが必須である。開発者 M.Berz と K.Makino は、COSY Infinity[5] と呼ばれるパッケージを製作しているが、これはかなり膨大なもののようである。

普及の点から考えれば、比較的使用しやすい MATLAB 上の数式処理機能 Symbolic Math Toolbox と精度保証付き区間演算ソフトウェア INTLAB とを混交して使用して実現できることが望ましい。本稿が、そのためのきっかけとなれば幸いである。

参考文献

[1] J.Hoefkens, M.Berz, K.Makino, *Verified High-Order Integration of DAEs and Higher-Order ODEs*, In:W.Kramer,J.W.Gudenberg (eds.) Scientific Computing, Validated Numerics, Interval Methods, Kluwer Academic/Plenum Publishers, New York, 2001

[2] 小森喬・山本野人「常微分方程式境界値問題の精度保証法の初期値問題への適用」日本応用数理学会論文誌, Vol 18, No.3, 2008

[3] R.J.Lohner, *Enclosing the solutions of ordinary initial and boundary value problems*, In:E.Kaucher, U.Kulisch, Ch.Ullrich (eds.) Computerarithmetic, 225-286, Teubner, Stuttgart(1987)

[4] K.Makino, M.Berz, *Efficient control of the dependency problem based on Taylor model methods*, Reliab. Comput. 5, No.1, 3-12, 1999

[5] K.Makino, M.Berz, *COSY INFINITY Version 9*, Nuclear Instruments and Methods, A558, 346-350, 2005

[6] R.E.Moore, *Interval Analysis*, Prentice Hall, Englewood Cliffs, N.J.,1966

[7] M.T.Nakao, *Numerical verification methods for solutions of ordinary and partial differential equations*, Numer. Funct. Anal. Optim. 22(3&4), 321-356, 2001