数式処理 *J.JSSAC (2005)*
Vol. 11, No. 3,4, pp. 99 - 117

Noda2005 論文

SCILAB へのべき級数の導入と その制御系設計への応用

北本 卓也*

山口大学

概　　要

'Matlab' is one of the most popular commercial software among the researchers in the area of control and system engineering. Although Matlab has no special functions as a programming language, it is equipped with many state-of-the-art packages, which makes Matlab competitive with its clones. In this paper, we focus on 'SCILAB', one of the clones of Matlab, which is developed by INRIA. Unlike other clones, the SCILAB has packages which are powerful enough to match those of Matlab. The SCILAB also has the function to define a new data type and its operators in the programming, which is similar to 'class' in C++ programming language. In this paper, utilizing the function, we introduce 'power series arithmetic' into the SCILAB, following the next three guidelines.

1. The power series are truncated at the total degree of variables.
2. Computations of the power series should be efficient.
3. The power series should be treated as if they are numbers.

We developed the packages for controller designs which utilizes the power series arithmetic. With the packages, the H_2 optimal controller is obtained in the form of truncated power series of the design parameters. We present a design example of H_2 optimal controller with restrictions in its overshoot and undershoot (such design restrictions require trial and error process in a conventional controller design method). We also present a technique to detect numerical errors due to power series arithmetic.

1　序論

Mathematica や Maple 等の数式処理システムは近年かなり著名になり，これらの数式処理の特色を生かして工学上の問題を解決する研究も行われるようになってきた．しかしながら，これらの応用はまだ研究段階にとどまっており，現場レベルでの数式処理システムの応用はまだきわめて少ない状況にある．特に，制御系設計の分野においてはいまだに MATLAB 等の数値計算を基礎とするシステムが主役の座を占めており，数式処理システムの活躍の場は少ない．この理由としては，１つにはよく言われるように数式処理システムが柔軟性に欠けることが挙げられる．具体的には，工学上の問題が主に浮動小数点数を取り扱うのに対して，数式処理システムのアル

*kitamoto@yamaguchi-u.ac.jp

ゴリズムの多くが誤差を全く含まない整数，または有理数をベースにしているため，浮動小数点数を含む演算がそのままの形では行えないことである．これらの問題を解決するために，近年では数値数式融合演算に関する研究が盛んになってきている．

しかしながら，制御系設計の研究者の話を聞いてみると，数式処理システムが活用されない原因はその「数式処理の柔軟性の欠如」だけにあるのではなく，MATLAB のような制御系設計専門のパッケージが少ないことにもあることがわかる．具体的には，ロバスト制御系設計では今では基本的な道具となった LMI（線形行列不等式）や μ 解析など計算を行うパッケージが MATLAB にはあるが，*Mathematica* や Maple などの汎用数式処理システムにはそれに該当するものがない（*Mathematica* には Control System Professional 2 というパッケージがあるが，これは最新の制御系設計理論の結果を含んでいるとは言えず，LMI や μ 解析など計算は行えない）．また，制御系設計の研究者の多くは MATLAB を使うことに慣れており，*Mathematica* や Maple などの他の文法をもつシステムには不慣れなため近寄りづらいことも制御系設計の研究者を数式処理システムから遠ざける遠因であろう．

そこで本稿では，MATLAB とよく似た文法とほとんど同じ機能（パッケージ）を併せ持つ SCILAB に注目し，この SCILAB にべき級数演算を導入することにより制御系設計の研究者にとって近づきやすく，使いやすい “擬似” 数式処理システムを作成する（“擬似” とつけたのは，このシステムでは通常の数式処理システムとは数式の取り扱いが異なるからである．このシステムでは打ち切りべき級数のみを取り扱い，普通の意味での数式は取り扱うことが出来ない）．

なお, 現在では MATLAB から数式処理システム Maple を直接使うためのパッケージ (Extended symbolic math toolbox) が開発され，以前に比べはるかに容易に MATLAB から Maple にアクセスできるようになっている．Extended symbolic math toolbox を用いた制御系の MATLAB ツールも開発されつつあり，この Extended symbolic math toolbox を用いて本稿のようなべき級数演算パッケージを Maple 上で実現するということも技術的には可能である（このようなパッケージは制御系設計者から見たインターフェースが MATLAB であり，制御系設計者から見たハードルが低いという点で本稿のパッケージと長所を共有するので有用ではないかと思われる）．

本稿は以下の構成をとる．まず，2 章で制御系設計の研究者の標準的な道具となっている MATLAB の概要を説明する．次に 3 章で MATLAB の代替ソフトウェアとして今回取り上げる SCILAB について概説し，その SCILAB へのべき級数演算の導入について述べる．4 章ではそのべき級数を係数とする多項式やべき級数を要素とする行列の SCILAB への導入について説明する．5 章では，SCILAB へ導入したべき級数演算のベンチマークテストを実施し，6 章では，べき級数演算を活用した制御系の設計例を示す．最後に 7 章で結論を述べる．

2 MATLAB について

MATLAB は行列を簡単に取り扱えるシステムとして昔から広く使われている有料のソフトウェアである．特に制御系設計の研究者，実務者の間では制御系設計の標準的な CAD ソフトウェアの役割を果たしている．行列の取り扱いが簡単になるような工夫がされており，行列演算が得意で（MATLAB では通常の数は 1×1 の行列として扱われる），浮動小数点演算に基礎を置いている（整数型もあることはあるが，ほとんど使われない）．プログラム言語としてはインタ

プリタ型の言語であり，一言で言うと *Mathematica* から数式処理機能を除いて，制御系設計のための専用パッケージを付け加えたようなものである．プログラム言語の文法は手続き型言語の極めてオーソドックスなもので，プログラム言語として特に際立つものはなく，GNU の Octave や本稿で取り挙げる SCILAB などの クローンがたくさん存在する．

MATLAB が制御系設計の標準的な CAD として使われている一番大きな理由は専用パッケージが充実していることである．例えば最新の制御系設計の結果を用いた「ロバストコントロールパッケージ」,「LMI（行列線形不等式）」,「μ 解析パッケージ」などがある．また，Simulab を用いれば，ブロック線図から簡単にシミュレーションを行えるようになっている．ただし，これらのパッケージは１つ１つが別売りされており，一揃いの制御系設計の機能を MATLAB でそろえようとすると，合計金額は高価になってしまう．

3 SCILAB について

3.1 概要

SCILAB は INRIA（フランス国立コンピュータ科学・制御研究所）で作成された高機能は行列演算システムであり，ソースコードも公開されているフリーソフトウェアである．MATLAB とよく似た文法，ほぼ同じ行列演算機能（LAPACK 等のパッケージを利用）を持っている．制御系設計パッケージが充実しており，MATLAB の「ロバストコントロールパッケージ」「LMI（行列線形不等式）パッケージ」「μ 解析パッケージ」と同機能のパッケージを持っている．また，MATLAB の Simulab と同様にブロック線図からシミュレーションを行えるシミュレータ SCICOS を含んでおり，制御系設計の面で MATLAB と対等に渡り合えるだけの機能を持っている．

3.2 べき級数型の導入

3.2.1 SCILAB における新しいデータ型を導入する機能

SCILAB は C++ のように新しいデータ型やそれに対する演算を導入する機能を持っている．ここでいう新しいデータ型とは，C++ のクラスや C 言語の構造体と同様に複数のデータを組み合わせて１つのデータとしてみなしたものである．例えば，2 つの実数型データを組み合わせて複素数という新しい型を作ったりすることが出来る．

新しいデータ型は `tlist` という関数を用いて作られる．`tlist` は以下の形で呼び出される．

```
tlist([‘型の名前’,‘データ 1 の名前’,…,‘データ n の名前’], データ 1, データ 2, …, データ n)
```

`tlist` の一番初めの引数は文字列を要素とするベクトルであり，一番初めの要素が新しいデータ型の名前を表している（この名前はプログラムの作成者が自由に定めることが出来る）．ベクトルの２番目以下の要素は新しいデータ型が含むデータの名前である．引数の２番目以下の データ 1, …, データ n が実際に含まれるデータを示している．例えば，複素数 $1 + 2i$ は次のように定義できる．

```
tlist([ ‘ complex ’ , ‘ real ’ , ‘ imag ’ ], 1.0, 2.0)
```

上のように定義した複素数を変数 c に代入したとすると，その実部，虚部はそれぞれ c.real，c.imag でアクセスできる．

3.2.2 SCILAB へのべき級数の実現とその方針

前節で説明した機能を用いて，打ち切りべき級数を SCILAB に導入した（打ち切りべき級数のデータ型の名前は “ps” とした）．べき級数の実現方法を決める方針として，次のものを定めた．

(i) 多変数のべき級数を取り扱い，打ち切り次数は全次数を用いる．
(ii) 計算をなるべく効率的に行う．
(iii) 通常の数や行列となるべく同じように取り扱えるようにする．

(ii) の計算効率の向上の観点から，新しく作成するべき級数 “ps” 型では打ち切りべき級数を数式として取り扱わず，その係数のみをベクトルとして保持することにした．例えば，べき級数 $\alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 x^2 + \alpha_4 xy + \alpha_5 y^2$ はベクトル $[\alpha_0\ \alpha_1\ \alpha_2\ \alpha_3\ \alpha_4\ \alpha_5]$ として保持される．このようにべき級数をベクトルとして保持した際の問題は，べき級数同士の乗除算である．というには，２変数以上のべき級数を全次数で打ち切った場合，その乗算は不規則になるからである．例えば，x, y を変数とするべき級数を全次数 3 次で打ち切った場合，

$$\begin{aligned} p_1 &= \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 x^2 + \alpha_4 xy + \alpha_5 y^2 \\ p_2 &= \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 xy + \beta_5 y^2 \end{aligned}$$

の加算の結果は

$$p_1 + p_2 = \alpha_0 + \beta_0 + (\alpha_1 + \beta_1)x + (\alpha_2 + \beta_2)y + (\alpha_3 + \beta_3)x^2 + (\alpha_4 + \beta_4)xy + (\alpha_5 + \beta_5)y^2$$

となり，その係数の計算は規則的になるが，乗算の結果は

$$\begin{aligned} p_1 \times p_2 = {} & \alpha_0\beta_0 + (\alpha_0\beta_1 + \alpha_1\beta_0)x + (\alpha_2\beta_0 + \alpha_0\beta_2)y + (\alpha_0\beta_3 + \alpha_1\beta_1 + \alpha_3\beta_0)x^2 \\ & + (\alpha_0\beta_4 + \alpha_1\beta_2 + \alpha_2\beta_1 + \alpha_4\beta_0)xy + (\alpha_0\beta_5 + \alpha_2\beta_2 + \alpha_5\beta_0)y^2 \end{aligned}$$

となり，係数同士の演算がかなり不規則になってしまう．乗算の際の係数同士の演算規則は，ベクトルの各要素がべき級数のどの項に対応しているかがわかっていれば決められるが，べき級数の乗算が起こるたびに，この係数同士の演算規則を計算していたのでは (ii) の効率性に反する．よって係数同士の演算規則を乗算表としてあらかじめ計算しておくことにした．また，新しく定義するデータ型「べき級数」のデータとしてこの乗算表や変数名，変数の数などのべき級数の構造に関する情報を含めると，「べき級数」のデータ量が多くなりコピーを作るときの手間がかかるため，これも上の (ii) の効率性に反する（データのコピーは代入を行う際や，計算を行う際のテンポラリー変数の作成などで頻繁に使われるので，この操作が効率的であることは非常に重要である）．よって，新しいデータ型 “ps_str” を新たに作成し，こちらにべき級数の乗算表や変

数名や変数の数などの基本情報とべき級数の種類を表す自然数 ID を保持し，べき級数の型 “ps” には，係数をベクトルで保存したものとこの ID のみを保持するようにした（図 3.2.2 を参照）.

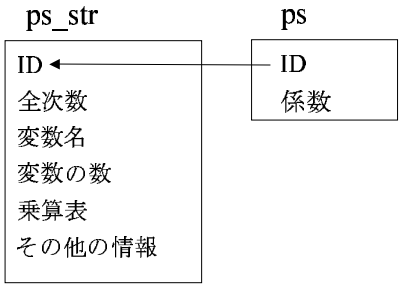

表 1: データ構造

下は実際にべき級数を生成する例である.

```
id = ps_str(['x','y'],2);
p1 = ps(id,[1,2,3,-1,2,-2]);
p2 = ps(id,[3,4,-1,2,-1,1]);
```

１行目の関数 ps_str() はべき級数の基本情報（乗算表など）を生成し，それに対する ID を返す関数である．１行目の命令で，x, y を変数とし打ち切り全次数を 2 としたべき級数の基本情報を生成し，それに対応する ID を変数 id に代入している．２行目の関数 ps() は，その ID と係数のベクトルを元に実際にべき級数を生成する関数である．上の例では，べき級数 $1 + 2x + 3y - x^2 + 2xy - 2y^2$ を変数 p1 に代入している．３行目の命令は，２行目の命令と同様にべき級数 $3 - 4x - y + 2x^2 - xy + y^2$ を変数 p2 に代入している.

方針 (iii) については，次節で述べる「べき級数の演算子の導入」で対応した.

3.3 べき級数の演算子の導入

SCILAB では，C++ のオペレータオーバーローディングのように新しく導入した型に対する演算子を定義できる．SCILAB のシステムは新しく定義された型に対する演算が行われると，自動的にあらかじめ決められた名前の関数を呼ぶようになっている．例を挙げると，変数 x と y の加算を定義するためには %<x の型名>_a_<y の型名> という名前の関数を定義する．ここで <x の型名>，<y の型名> はそれぞれ変数 x と y の型名 を表す．例えばべき級数 （ps 型）同士の加算を定義するためには %ps_a_ps() という名前の関数を定義する．p1, p2 を ps 型 の変数とすると，p1+p2 という表現は自動的に %ps_a_ps(p1,p2) で置き換えられ，計算が実行される．ps 型はべき級数の種類を表す ID と係数のベクトルからなるので，%ps_a_ps() の定義は次のようにすればよい.

```
function c = %ps_a_ps(a,b)
if (a.id ~= b.id) then
  error("%ps_a_ps: ids are different!!");
end
c=ps(id,a.coef+b.coef);
endfunction
```

上の関数では，まず ID が一致するかどうかをチェックし，一致しなければエラーメッセージを出力して演算を終了する．ID が一致している場合は，与えられた引数と同じ ID をもち，与えられた引数の係数ベクトルの和を係数ベクトルに持つべき級数を新たに作成し，それを関数値として返している．

べき級数同士の減算, 乗算, 除算については %ps_a_ps() の a（加算を表す）をそれぞれ s（減算を表す), m（乗算を表す), r（除算を表す）で置き換えた名前の関数 %ps_s_ps(), %ps_m_ps(), %ps_r_ps() を同様に定義すればよい（乗算と除算の場合の計算は，前節で述べた乗算表を用いて行う）．このほか，べき乗 ^（p を用いる）や等号演算子 ==（o を用いる）などの演算子も定義できる．表 2 に演算子とそれに対応する文字の代表的なものの一覧表を示す．

表 2: 演算子と対応する文字

演算	演算子	文字	演算	演算子	文字
加算	+	a	等式の否定	<>	n
減算	-	s	不等式	<	1
乗算	*	m	不等式	>	2
除算	/	r	不等式	<=	3
べき乗	^	p	不等式	>=	4
共役転置	'	t	取り出し	()	e
等式	==	o	代入	()	i

関数 %ps_a_ps() はあくまでべき級数同士の加算を定義しているので，べき級数と数値の加算には使えない．よって p1 を ps 型とするとき，p1+1 というようなべき級数と数値の加算を行うためには %ps_a_s()（数値は s 型である）を別に定める必要がある．このため，このようなべき級数と数値の演算を行う関数も定義した（ちなみに 1+p1 を演算を行う関数は%ps_a_s() でなく，%s_a_ps() である）.

4 多変数多項式，多項式行列の導入

SCILAB ではすでに poly 型という多項式を扱う型が存在するが，1 変数多項式に限られており，主変数の係数が数値であるものしか取り扱えない．1 変数多項式ではべき級数演算を生かすことが出来ないので，主変数の係数がべき級数である新しい型を定義し，multpoly 型と名づけた．また，SCILAB ではもちろん行列は取り扱えるが，要素が数値であるものしか取り扱えないので，要素がべき級数である新しい型を定義し，こちらは psmat 型と定義した．multpoly

型は，データとして主変数名と係数（ps 型）のリストの２つをデータを保持することにした．psmat 型は，データとして行数，列数，要素（ps 型）のリストの３つをデータとして保持する．

実際に multpoly 型のデータを作るときには， multpoly(変数名, 係数のリスト) とする．例えば，次の命令は x を主変数とする 2 次の多項式 p0 + p1*x + p2*x^2 を作り，f に代入する．（ただし，p0，p1，p2 が ps 型の変数）

```
f = multpoly('x',list(p0,p1,p2))
```

また，psmat 型のデータを作るときには， psmat（行数，列数，要素のリスト） とする．例えば，次の命令は，要素がべき級数である 2×2 の行列 $\begin{bmatrix} p11 & p12 \\ p21 & p22 \end{bmatrix}$ を作り，変数 m に代入する．（ただし，p11，p12，p21，p22 が ps 型の変数）

```
m = psmat(2,2,list(p11,p12,p21,p22))
```

multpoly 型に対する加減乗除等の演算子をべき級数の時と同様に定義し，記号的ニュートン法でべき級数根を計算する関数 multpoly_newton() を作成した．以下にそのソースコードを示す（説明のため，一列目に本来のソースコードには含まれていない行番号入れている）．

```
1:  function xn = multpoly_newton(p,x0)
2:  xn = x0;
3:  tdeg = ps_str_data(p.coef(1).id).tdeg;
4:  pd = p';
5:  i = 1;
6:  while (i-1)<tdeg
7:    i = 2*i;
8:    xn = xn-p(xn)/pd(xn);
9:  end;
10: endfunction
```

一番目の引数 p は multpoly 型の変数であり，multpoly_newton() はこの p のべき級数根を計算する．二番目の引数 x0 は数値であり，求めるべき級数根の定数部である． 2 行目の xn = x0 で x0 をべき級数根 xn を x0 で初期化する．3 行目ではべき級数の全次数を tdeg に代入している． 4 行目の pd = p' は p の主係数による微分を pd に代入している（' は本来，行列の共役転置を取る演算子なのであるが，multpoly 型では，これを微分演算子として定義している）．5-9 行目でニュートン法の繰り返しによりべき級数根を計算している．8 行目の p(xn) は p の主変数に xn を代入したときの値（結果は ps 型）を表している（この () は本来，行列やベクトルの要素の取り出しに使われる演算子であるが，ここでは主変数の値の代入に用いている）．

ソースコードからわかるように，通常の数式処理システムとほとんど同じ形でべき級数根を計

算するプログラムが書けている．これは SCILAB へのべき級数の導入が自然な形で行えたことを示している．

psmat 型に対しても，multpoly 型と同様に，加減乗除等の演算子を定義し，逆行列のべき級数展開を計算する関数 psmat_inv() を作成した．以下にそのソースコードを示す（上と同様に，１列目に行番号を入れている）．

```
1:  function pn = psmat_inv(a)
2:  if (a.row ~= a.col) then
3:    error(" Matrix is not square!!");
4:  end
5:  a0 = psmat_const(a);
6:  e = eye(a0);
7:  pn = inv(a0);
8:  tdeg = ps_str_data(a(1,1).id).tdeg;
9:  i = 1;
10: while (i-1)<tdeg
11:   i = 2*i;
12:   pn = pn+pn*(e-a*pn);
13: end;
14: endfunction
```

上のソースコードの説明は省略するが，これも通常の数式処理システムとほとんど同じ形でプログラムが書けている．

5 ベンチマークテスト

SCILAB 上でべき級数演算を行う本稿のパッケージの演算効率を調べるために，べき級数の加減乗除のベンチマークテストを行った．また，比較のため *Mathematica* Ver 5.0 でも同様のテストを行い，その結果を比較した．

5.1 テスト内容

まずはじめに全次数を定め，その全次数のべき級数を２つランダムに生成する（係数は -1 から 1 の間の実数を乱数で生成させる）．それから，その２つのべき級数同士の加減乗除にかかる時間を計測する．ただし，正確を期すため，加減は 10000 回，乗除は 100 回連続して計算し，最後に計算時間を計算回数で割る．これを 10 組行い，その平均時間をテスト結果とした．テストを行ったマシンは CPU Pentium M 1.6GHz, メモリ 1GByte の Windows マシンである．1〜3 変数のべき級数に対して，それぞれ全次数を 2 から 18 まで 4 つずつ増やしながら，上のテストを行った．

Mathematica のべき級数演算に関する注意であるが，*Mathematica* は基本的に 1 変数のべき級数しか取り扱えない．2 変数以上のべき級数を扱う場合は各変数について，決められた次数まで

展開した級数を取り扱うことになる（つまり，1 つのべき級数の係数が別のべき級数であるという形になる）．例えば，x, y の変数を持つべき級数を全次数 2 次で打ち切ると

$$\alpha_{0,0} + \alpha_{1,0}x + \alpha_{0,1}y + \alpha_{2,0}x^2 + \alpha_{1,1}xy + \alpha_{0,2}y^2$$

となるが，各変数について 2 次で打ち切ると次のようになる．

$$\alpha_{0,0} + \alpha_{0,1}y + \alpha_{0,2}y^2 + (\alpha_{1,0} + \alpha_{1,1}y + \alpha_{1,2}y^2)x + (\alpha_{2,0} + \alpha_{2,1}y + \alpha_{2,2}y^2)x^2$$

このように，各変数についての次数で打ち切ったものは全次数で打ち切ったものより項数が多くなり，ベンチマークテストとしては不利になるが，*Mathematica* には全次数でべき級数を打ち切る機能がない（ダミー変数を用いるトリックについては下で述べる）ので，今回はそれぞれの変数について全次数まで級数展開したものを用いることにした．なお，ダミー変数 t を用いて，各変数 x_i を $x_i t$ で置き換え，t の次数で打ち切ることで *Mathematica* でも擬似的に全次数のべき級数を導入することが出来る．このようにして擬似的な全次数のべき級数を *Mathematica* に導入したものについてもベンチマークテストを行ったが，各変数を全次数まで展開したものに比べ，かえって計算効率が落ちてしまったのでテスト結果には載せていない．

5.2 テスト結果と考察

1 変数のべき級数に対するベンチマークテストの結果を表 3 へ示す．

表 3: 1 変数のべき級数に対するベンチマークテスト結果

全次数	2	6	10	14	18
加算（*Math.*）	0.01702	0.01793	0.01873	0.01963	0.02053
加算（本システム）	0.1187	0.1189	0.1193	0.1198	0.1202
計算時間比	0.1434	0.1508	0.1570	0.1639	0.1708
減算（*Math.*）	0.04056	0.04216	0.04376	0.04536	0.04707
減算（本システム）	0.1184	0.1187	0.1191	0.1195	0.1198
計算時間比	0.3426	0.3552	0.3674	0.3796	0.3929
乗算（*Math.*）	0.02414	0.04917	0.09203	0.15242	0.23113
乗算（本システム）	0.2704	0.5808	1.1116	1.8627	2.8241
計算時間比	0.0893	0.0847	0.0828	0.0818	0.081842
除算（*Math.*）	0.0715	0.11296	0.17535	0.25707	0.36513
除算（本システム）	0.2403	0.5207	0.9914	1.6624	2.5437
計算時間比	0.297545	0.216939	0.176871	0.154638	0.143543

「演算（*Math.*）」 は *Mathematica* でその演算にかかった時間,「演算（本システム）」は同じく本システムでその演算にかかった時間をそれぞれ示している．時間の単位は ms（ミリセカンド）である．「計算時間比」は「演算（*Math.*）」/「演算（本システム）」であり，本システムの演算が *Mathematica* より何倍早いかを表している．表 3 よりわかるように，1 変数の場合はどの演算においても *Mathematica* による演算が数倍早い（「計算時間比」が 1 以下である）．「計算時間比」

の平均値は 0.202 であるので，本システムの 1 変数のべき級数の演算速度は *Mathematica* のそれと比べ 5 倍程度遅いといえる．

2 変数のべき級数に対するベンチマークテストの結果を表 4 へ示す．

表 4: 2 変数のべき級数に対するベンチマークテスト結果

全次数	2	6	10	14	18
加算（*Math.*）	0.05828	0.10805	0.16554	0.23023	0.29873
加算（本システム）	0.119	0.1205	0.1236	0.1264	0.131
計算時間比	0.490	0.897	1.34	1.82	2.28
減算（*Math.*）	0.12839	0.27079	0.42481	0.60356	0.77751
減算（本システム）	0.1185	0.1204	0.123	0.1262	0.1308
計算時間比	1.083	2.25	3.45	4.78	5.94
乗算（*Math.*）	0.14	0.961	4.146	12.488	30.114
乗算（本システム）	0.4006	3.1445	13.93	41.9603	99.5031
計算時間比	0.349	0.306	0.298	0.298	0.303
除算（*Math.*）	0.641	3.175	12.778	37.634	87.295
除算（本システム）	0.3605	2.784	12.7383	38.7057	92.8535
計算時間比	1.78	1.14	1.00	0.970	0.940

表 4 の「計算時間比」をみると，*Mathematica* の方が早い場合（計算時間比 < 1）と本システムの方が早い場合（計算時間比 > 1）の両方があり，演算と全次数によってどちらが早いかは異なっている． 計算時間比の平均は 1.59 なので，本システムの 2 変数のべき級数の演算速度は *Mathematica* のそれと同等であるといえる．

3 変数のべき級数に対するベンチマークテストの結果を表 5 へ示す．

表 5: 3 変数のべき級数に対するベンチマークテスト結果

全次数	2	6	10	14	18
加算（*Math.*）	0.18717	0.68768	1.69453	3.37365	6.50616
加算（本システム）	0.1187	0.1238	0.1372	0.1654	0.2083
計算時間比	1.58	5.55	12.4	20.4	31.2
減算（*Math.*）	0.40999	1.57677	3.81649	7.50279	13.4212
減算（本システム）	0.1187	0.1238	0.1371	0.1639	0.2093
計算時間比	3.45	12.7	27.8	45.8	64.1
乗算（*Math.*）	0.7	24.74	260.18	1443.17	5524.54
乗算（本システム）	0.6009	12.9987	109.377	526.437	1824.02
計算時間比	1.16	1.90	2.38	2.74	3.028
除算（*Math.*）	3.91	84.02	817.77	4412.65	16663.5
除算（本システム）	0.5107	11.847	101.576	494.191	1720.47
計算時間比	7.66	7.09	8.05	8.93	9.69

表 5 の「計算時間比」よりわかるように，全ての場合について本システムの演算速度が勝っている．計算時間比の平均は 13.9 であり，本システムの 3 変数のべき級数の演算速度は *Mathematica* のそれよりも高速であるといえる．

変数が多くなるにつれ，本システムが優位になっていったのは *Mathematica* が全次数で打ち切るべき級数演算をサポートしていないことにその理由がある（変数が多くなるにつれ，全次数で打ち切るべき級数と各変数の次数で打ち切るべき級数の項数の差は大きくなる）．よって，4 変数以上の場合でも本システムの方がべき級数の演算が高速であると考えられる．故に 1 変数のべき級数の場合を除いて，本システムは *Mathmatica* と同等かそれ以上の演算速度を持っているといえる．また，1 変数のべき級数の場合においても 5 倍程度の速度低下ですんだのは，*Mathematica* のべき級数の加減乗除の演算が内部コードで高速に行われていることを考慮すると（本システムでは SCILAB のプログラムとして実行されている）予想外に健闘している．これには，乗算表をあらかじめ計算しておくことや，べき級数の構造に関するデータをべき級数そのものでなく別のデータに保存している等の工夫が貢献していると考えられる．

6 制御系設計への応用

6.1 制御系設計の対象と H_2 最適制御

本稿では，線形微分方程式で与えられた次のシステムを制御系設計対象とする．

$$\left\{ \begin{array}{rcl} \frac{d}{dt}x(t) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) + Du(t) \end{array} \right. \tag{1}$$

ここで，$x(t), u(t), y(t)$ はそれぞれ，状態変数，制御入力，観測出力 を表し，A, B, C, D は定数行列である．制御系設計の目的は，$t > 0$ において $x(t)$, $y(t)$ が望ましい値をとるように $u(t)$ を定めることである．状態 $x(t)$ が使える（すなわち $x(t)$ が何らかの方法で観測，もしくは推定できる）場合には，$u(t) = -Kx(t)$ (ただし，K は行列） とすることが一般的であり（これを状態フィードバックという），この場合には，制御系設計は $x(t)$, $y(t)$ が望ましい値をとるように行列 K を決めることになる．この K の決め方には様々な方法があるが，本稿では H_2 最適制御を取り扱う．H_2 最適制御とは下の評価関数

$$\int_0^\infty \left\{ x(t)^T Q x(t) + u(t)^T R u(t) \right\} \, dt \tag{2}$$

(ただし, Q, R は固有値が全て正の対称行列で $x(t)^T$, $u(t)^T$ はそれぞれ $x(t)$, $u(t)$ の転置を表す) を最小化する入力 $u(t)$ を求め, これを (1) の入力 $u(t)$ とする制御系設計法である．(2) を最小化する $u(t)$ は以下の手順で求まることが知られている．まず, 代数 Riccati 方程式 $PA + A^T P - PBR^{-1}B^T P + Q = 0$ を P について解き，その解を求める（この方程式を満たす解 P は複数あるが，そのうち固有値が全て正の対称行列を選ぶ）．そして，$K = B^T P$ で K を定めれば $u(t) = -Kx(t)$ が求める $u(t)$ となる．

6.2 制御系設計用のパッケージ

べき級数演算を用いた制御系の設計を行うため，次の関数を作成した．

- psmat_inv(a) — psmat 型の引数 a の逆行列のべき級数展開を返す（計算のアルゴリズムについては，[2],[7] を参照）.
- psmat_eig(a) — psmat 型の引数 a の固有値と固有ベクトルのべき級数展開を返す.（計算のアルゴリズムについては，[2] [6],[8],[12] を参照）
- psmat_lyap(a,b) — 引数 a,b を psmat 型，または数値行列としたとき，Lyapunov 方程式 a*x+x*a^T = b の解 x のべき級数展開を返す.
- psmat_are(a,b,c) — 引数 a,b,c を psmat 型, または数値行列としたとき, 代数的 Riccati 方程式 a^T*p+p*a-p*b*p+c = 0 の解 p のべき級数展開を返す.（計算のアルゴリズムについては，[3],[5] を参照）
- psmat_expm(a) — psmat 型の引数 a の行列指数関数のべき級数展開を返す.（計算のアルゴリズムについては，[9] を参照）
- step(a,b,c,d,t) — 数値行列 a,b,c,d と実数ベクトル t を引数にとり，A = a, B = b, C = c, D = d としたときの （1） のシステムのステップ応答を計算する.
- impulse(a,b,c,d,t) — 数値行列 a,b,c,d と実数ベクトル t を引数にとり，A = a, B = b, C = c, D = d としたときの （1） のシステムのインパルス応答を計算する.
- eval_y(a,b,c,d,t) — 数値行列 a,b,c,d と実数 t を引数にとり，A = a, B = b, C = c, D = d としたときの （1） のシステムのステップ応答の t =t における $y(t)$ の値（実数値）を返す.
- psmat_eval_y(a,b,c,d,t) — psmat 型の変数 a,b,c,d と ps 型の変数 t を引数にとり，A = a, B = b, C = c, D = d としたときの （1） のシステムのステップ応答の t =t における $y(t)$ の値（ps 型）を返す.（計算のアルゴリズムについては，[9] を参照）
- psmat_peaktime(a,b,c,d,t) — psmat 型の変数 a,b,c,d と実数値 t を引数にとり，A = a, B = b, C = c, D = d としたときの （1） のシステムのステップ応答のおける $y(t)$ のピーク値 y_{peak} （ps 型）とそのピーク値を取るときの t（すなわち $y(t) = y_{\text{peak}}$ を満たす t）の値（ps 型）を返す. $y(t)$ のピーク値は複数あるが，ピーク値を取るときの t の定数項が t に最も近い物を選ぶ.（計算のアルゴリズムについては，[9] を参照）

6.3 制御系設計への応用例

（1） のシステムにおいて，A, B, C, D を以下のように置く.

$$A = \begin{bmatrix} -1 & 1 \\ -3 & 1 \end{bmatrix},\ B = \begin{bmatrix} -1 \\ 0 \end{bmatrix},\ C = \begin{bmatrix} 1 & 2 \end{bmatrix},\ D = 0 \tag{3}$$

このシステムに対して，状態フィードバックの H_2 最適制御により制御系を設計する.「**6.1 制御系設計の対象と H_2 最適制御**」で述べたように，重み行列 Q, R を適当に定め，（2） を最小化する $u(t) = -Kx(t)$ を求めればよい. そこで，まず Q, R を以下のように置き，

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ R = 1 \tag{4}$$

代数 Riccati 方程式 $PA + A^TP - PBR^{-1}B^TP + Q = 0$ を P について解くと，

$$P = \begin{bmatrix} 2.12 & -1.29 \\ -1.29 & 1.61 \end{bmatrix} \tag{5}$$

を得る．よって

$$K = B^TP = \begin{bmatrix} -2.12 & 1.29 \end{bmatrix} \tag{6}$$

となり，

$$u(t) = -Kx(t) = 2.12x_1(t) - 1.29x_2(t) \tag{7}$$

を得る．このとき，得られた制御系のステップ応答は図 6 のようになる（ただし，図 6 の $y(t)$ は $y(\infty) = 1$ となるように正規化している）．

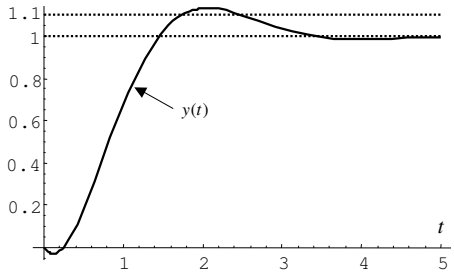

表 6: 初期の制御系のステップ応答

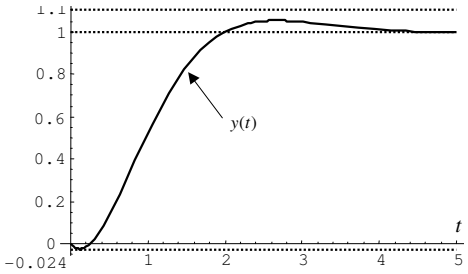

表 7: 改善後の制御系のステップ応答

図 6 からわかるように，$t = 2$ 辺りにあるオーバーシュート（1 を超えた $y(t)$ の値）の最大値が 10% (=1.1) を超えており，逆応答（0 より小さい $y(t)$ の値）も生じている．そこで，重み行列 Q の対角成分の値を調整することで，下記の制御系設計目標を達成することを考える．

オーバーシュートを 10% 未満に抑え，かつ逆応答もなるべく小さく抑える (8)

従来の設計法ではこのような設計目標を達成するためには，試行錯誤のプロセスを必要としたが，べき級数演算を用いればこれをシステマティックに行うことが出来る．具体的には，べき級数演算を用いることで Q に設計パラメータを記号として残したまま制御入力 $u(t)$ を求めることができ，ステップ応答におけるピーク値を設計パラメータのべき級数の形で計算できる．あとは，(8) を満たすように，パラメータの値を決めればよい．実際，本稿の制御系設計用パッケージを用いて図 6 のステップ応答を図 7 のように改善することが出来た（改善後の制御系では，(8) の条件が満たされている）．以下では，その設計プロセスを説明する．

まず，重み行列 Q, R を以下のように置く（下のように対角成分を置いたのは $|z_1|, |z_2|$ の値をなるべく小さく保ったまま，対角成分の値を広範囲で動かすためである）．

$$Q = \begin{bmatrix} (1+z_1)^5 & 0 \\ 0 & (1+z_2)^5 \end{bmatrix},\ R = 1 \tag{9}$$

ここで，z_1, z_2 は設計パラメータであり，$|z_1| \le 0.5$, $|z_2| \le 0.5$ を満たすとする．($0.03125 \le (1+z_1)^5, (1+z_2)^5 \le 7.59375$ より，Q の対角成分が 0.03125 から 7.59375 の範囲で H_2 最適コン

トローラを探すことになる）．z_1, z_2 は先に述べたようにステップ応答におけるピーク値が望ましい値になるように決めたいので，べき級数の形で残したまま H_2 最適制御系を構成し，ステップ応答におけるピーク値を z_1, z_2 のべき級数の形で求めることにする．

以下に実際の制御系設計のプロセスを示す．以下は SCILAB へのコマンド入力である．なお，説明のため第 1 列目に行番号（“数値:” の箇所）を入れているが，もちろん実際のコマンドには行番号は含まれない．まず，システム（1），(3）を定義する．

```
1: a = [-1 1; -3 1];
2: b = [-1 0]';
3: c = [1 2];
4: d = 0;
```

次に，重み行列 Q, R を（9） のように定義する．

```
1: id = ps_str(['z1','z2'],7);
2: q = psmat(id,zeros(2,2));
3: q1 = ps(id);
4: q1(0) = [1 1];
5: q(1,1) = q1*q1*q1*q1*q1;
6: q2 = ps(id);
7: q2(0) = [1 0 1];
8: q(2,2) = q2*q2*q2*q2*q2;
9: r = [1];
```

1 行目では，べき級数の基本的な情報を定義している．パラメータは z_1, z_2 の 2 つである．打ち切り次数は全次数 7 次とした（打ち切り次数を 7 に定めた理論的な根拠はない）．2-8 行目では，Q を定義している．まず，2 行目で Q を psmat 型（べき級数を要素にもつ行列）の 2×2 の零行列に置く．3-4 行目では q1 を $(1 + z_1)$ としている（べき級数の係数は [定数項, z_1 の係数, z_2 の係数, $\cdots$] と並んでいるので，[1 1] は $(1 + z_1)$ を表している）．よって 5 行目は Q の (1,1) 成分に $(1 + z_1)^5$ を代入する命令になる．同様にして，6-8 行で Q の (2,2) 成分に $(1 + z_2)^5$ を代入している．9 行目では R に 1 を代入している．

これでシステムの準備が出来たので，次の命令で H_2 最適制御系を構成する．

```
1: p = psmat_are(a,b*inv(r)*b',q);
2: k = b'*p;
3: at = a-b*k;
```

1 行目の命令で代数的 Riccati 方程式 $A^T P + PA + PBR^{-1}B^T P - Q = 0$ の解 P を計算している．Q にはパラメータ z_1, z_2 が含まれるので，計算結果の p は psmat 型（z_1, z_2 の打ち切りべき級数）

となる．2 行目では，実際のコントローラゲイン K を計算している（K も psmat 型である）．実際のコントローラは $u(t) = -Kx(t)$ で与えられる．3 行目では，入力 $u(t)$ にこのコントローラを適用した制御系の A 行列 $(= A - BK)$ を計算し，変数 at に代入している（at も psmat 型）．

次にこのシステムのステップ応答のピーク値を計算する．

```
1: [y1,t1]=psmat_peaktime(at,b,c,d,0.1);
2: [y2,t2]=psmat_peaktime(at,b,c,d,2);
```

図 6 より $z_1 = 0, z_2 = 0$ のとき，制御系のステップ応答は $t = 0.1$, $t = 2.0$, $t = 4$ の辺りでピーク値を取っている（$t = 0.1$ の時は逆応答のピーク値）．そこで，1-2 行目でそのピーク値とピーク値を取る時間を計算している．ピーク値はピーク値を取る時間はともに psmat 型（z_1, z_2 の打ち切りべき級数）である．参考のため，t1 と y1 を以下に示す．

```
t1  =
0.1224596 - 0.0105167*z1 - 0.0228435*z2 - 0.0160519*z1^2 + 0.0228602*z
     1*z2 - 0.0086916*z2^2 - 0.0048062*z1^3 + 0.0207401*z1^2*z2 - 0.02
     49321*z1*z2^2 + 0.0175755*z2^3 + 0.0104859*z1^4 - 0.0246366*z1^3*
     z2 + 0.0149730*z1^2*z2^2 - 0.0004266*z1*z2^3 - 0.0124474*z2^4 + 0
     .0045135*z1^5 - 0.0363986*z1^4*z2 + 0.0879816*z1^3*z2^2 - 0.09596
     83*z1^2*z2^3 + 0.0581038*z1*z2^4 - 0.0090761*z2^5 - 0.0106686*z1^
     6 + 0.0397802*z1^5*z2 - 0.0281488*z1^4*z2^2 - 0.0641130*z1^3*z2^3
      + 0.1386888*z1^2*z2^4 - 0.1077031*z1*z2^5 + 0.0369955*z2^6 - 0.0
     039249*z1^7 + 0.0579790*z1^6*z2 - 0.2246491*z1^5*z2^2 + 0.3786463
     *z1^4*z2^3 - 0.2959000*z1^3*z2^4 + 0.0528626*z1^2*z2^5 + 0.053303
     9*z1*z2^6 - 0.0415234*z2^7
y1  =
-0.0588569 + 0.0067010*z1 + 0.0137098*z2 + 0.0101002*z1^2 - 0.0147790*
     z1*z2 + 0.0043691*z2^2 + 0.0026767*z1^3 - 0.0129860*z1^2*z2 + 0.0
     164857*z1*z2^2 - 0.0110891*z2^3 - 0.0070773*z1^4 + 0.0169150*z1^3
     *z2 - 0.0101725*z1^2*z2^2 - 0.0003118*z1*z2^3 + 0.0087567*z2^4 -
     0.0027660*z1^5 + 0.0236845*z1^4*z2 - 0.0581702*z1^3*z2^2 + 0.0630
     380*z1^2*z2^3 - 0.0376935*z1*z2^4 + 0.0048071*z2^5 + 0.0072892*z1
     ^6 - 0.0275811*z1^5*z2 + 0.0206004*z1^4*z2^2 + 0.0412952*z1^3*z2^
     3 - 0.0913448*z1^2*z2^4 + 0.0717595*z1*z2^5 - 0.0237037*z2^6 + 0.
     0024990*z1^7 - 0.0385361*z1^6*z2 + 0.1507337*z1^5*z2^2 - 0.254123
     9*z1^4*z2^3 + 0.1980022*z1^3*z2^4 - 0.0342933*z1^2*z2^5 - 0.03812
     40*z1*z2^6 + 0.0285312*z2^7
```

y1, y2 はこのままでは少しわかりにくいので，$y(\infty) = 1$ となるように正規化する．

```
1: tmp = c*psmat_inv(at)*b;
2: nf = -1/tmp(1);
3: ny1 = y1*nf;
4: ny2 = y2*nf;
```

$y(\infty) = -C(A-BK)^{-1}B$ であるので，正規化するためには $1/(-C(A-BK)^{-1}B)$ を y1, y2 に掛ければよい．上では，1 行目で $C(A-BK)^{-1}B$ を計算し，2 行目で $-1/(C(A-BK)^{-1}B)$ を nf に代入している．3-5 行目で，nf を y1, y2 に掛けて，正規化されたものを ny1, ny2 に代入している．|ny1|（逆応答），ny2-1（オーバーシュート）の等高線図を z_1z_2 平面に描いたものをそれぞれ 図 8,9 に示す．暗い箇所が値が小さい所を表している．

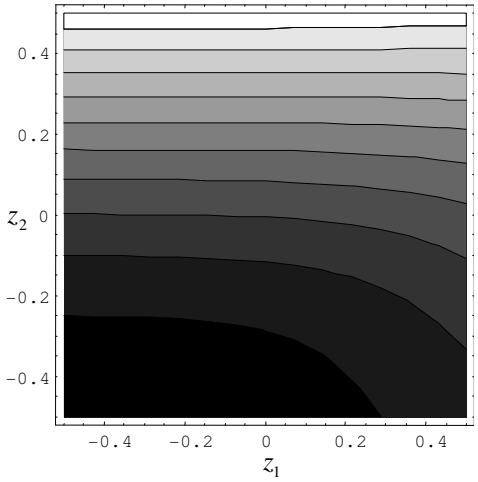

表 8: オーバーシュートの等高線図

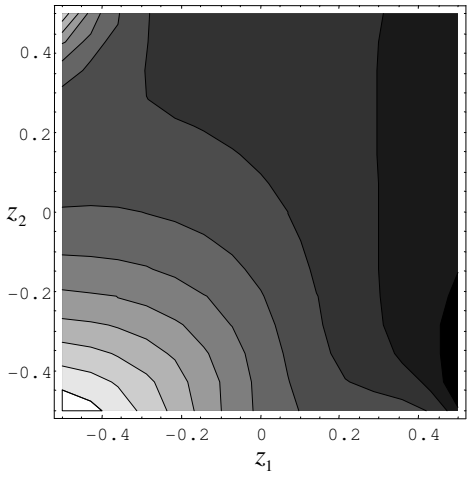

表 9: 逆応答の等高線図

|ny1|, ny2-1 はぞれぞれ, システムの逆応答の大きさ, システムのオーバーシュートを表しているので, ともに出来るだけ小さいほうが望ましい．これらをともに小さくするために，|ny1|<0.024（逆応答の 2.4%未満），ny2-1<0.1（オーバーシュートが 10%未満）を満たす範囲を描くと図 1 を得る（灰色部分が両方の条件を満たす領域である）．この図より，$z_1 = 0.3, z_2 = -0.4$ とし（このときの ny1= −0.02389，ny2= 1.07932），コントローラゲイン K を求める．

```
1: q = zeros(2,2);
2: q(1,1) = (1+0.3)^5;
3: q(2,2) = (1-0.4)^5;
4: p = riccati(a,b*b',q,'c');
5: k = b'*p;
```

1-3 行目で重み行列 Q を設定し直し，4 行目で Riccati 方程式を解いている（ riccati() は SCILAB で準備されている関数である．今回は全て数値行列のなので，こちらを用いた）．最後に 5 行目

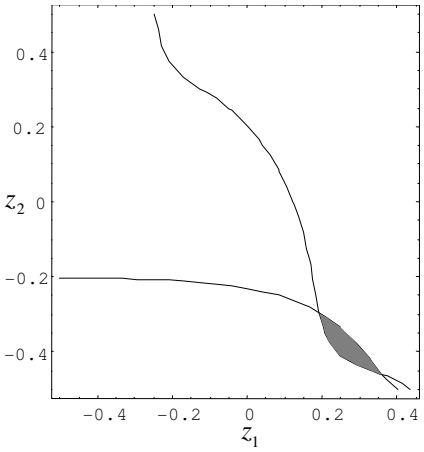

図 1: 条件を満たす領域

で K を計算している．このコントローラを用いて構成した制御系のステップ応答を描くと図 7（$y(\infty) = 1$ となるように正規化している）を得る．図 7 より，逆応答，オーバーシュートともにそれぞれ，2.4%，10% 以下に抑えられている事が確認できる．以上より設計仕様を満たす制御系が 1 つ見つかったわけであるが，仕様を満たす制御系を全て見つけるという観点からは以下の問題が残っている．

- ここでは $|z_1| \leq 0.5$, $|z_2| \leq 0.5$ すなわち，Q の対角成分 $(1+z_1)^5, (1+z_2)^5$ が $0.03125 \leq (1+z_1)^5, (1+z_2)^5 \leq 7.59375$ を満たす範囲で制御系を探しており，この範囲以外での検索は行っていない．すなわち，この範囲以外で設計仕様を満たす制御系が存在する可能性がある．
- べき級数演算では打ち切り誤差は避けられないが，この誤差のため Q の対角成分が $0.03125 \leq (1+z_1)^5, (1+z_2)^5 \leq 7.59375$ を満たす範囲以内においても設計仕様を満たす制御系を見逃している可能性がある．

上の第 1 の問題点は，計算範囲を広げることで解決できる．計算範囲を広げるといっても $|z_1|, |z_2|$ の範囲を広げるのではなく（こうするとべき級数の打切り誤差の影響が大きくなる），$|z_1|, |z_2|$ の範囲はそのままで対角成分の形を変えることで $0.03125 \leq (1+z_1)^5, (1+z_2)^5 \leq 7.59375$ 以外の範囲を検索する．例えば 対角成分を $(2+z_1)^5, (2+z_2)^5$ とおけば $|z_1| < 0.5, |z_2| < 0.5$ の時，$7.59375 \leq (2+z_1)^5, (2+z_2)^5 \leq 97.6563$ の範囲を検索できる．つまり，対角成分が 0.03125 から 97.6563 の範囲でコンローラを検索するときは，$|z_1| < 0.5, |z_2| < 0.5$ の条件の下で対角成分が $\left\{(1+z_1)^5, (1+z_2)^5\right\}, \left\{(2+z_1)^5, (1+z_2)^5\right\}, \left\{(1+z_1)^5, (2+z_2)^5\right\}, \left\{(2+z_1)^5, (2+z_2)^5\right\}$ の 4 つ場合を調べれば良い．

打ち切り誤差の問題は次のようにチェックする．原点 $(z_1, z_2) = (0, 0)$ から最も遠い $(z_1, z_1) = (-0.5, -0.5), (-0.5, 0.5), (0.5, -0.5), (0.5, 0.5)$ の 4 つの点で実際にアンダーシュート，オーバーシュートの最大値を計算し，これと ny1,ny2 の各点での値を比較することにより ny1,ny2 の打

ち切り誤差をチェックする．例えば，$(z_1, z_2) = (-0.5, -0.5)$ での打切り誤差が実用上問題ないほど小さい場合は $-0.5 \le z_1, z_2 \le 0$ における ny1, ny2 の誤差は実用上問題がないほど小さいと判定する．この方法で先ほどの設計例における ny1, ny2 の打切り誤差をチェックすると $(z_1, z_2) = (-0.5, -0.5), (0.5, -0.5), (0.5, 0.5)$ での打ち切り誤差は問題なかったが，$(z_1, z_2) = (-0.5, 0.5)$ における相対誤差が 30% にも上ったので $-0.5 < z_1 < 0,\ 0 \le z_2 \le 0.5$ における ny1, ny2 の値は誤差で信用できないと判定された．よって，対角成分が $0.03125 \le (1 + z_1)^5 \le 1,\ 1 \le (1 + z_2)^5 \le 7.59375$ の範囲には設計仕様を満たすコントローラが存在する可能性がある．そこで対角成分を $(0.75 + z_1)^5, (1.25 + z_2)^5$ と置いて $|z_1| < 0.25, |z_2| < 0.25$ の範囲でもう一度同様に仕様を満たすコントローラを探したが，見つからなかった．以上より，$0.03125 \le (1 + z_1)^5, (1 + z_2)^5 \le 7.59375$ の範囲以内においては求まったコントローラ以外に仕様を満たすコントローラはないと判定された．

7 結論

SCILAB は制御系設計の標準的ツールとなっている MATLAB に匹敵する機能，パッケージを持ったオープンソースのソフトウェアである．SCILAB には，C++ のクラスと同様に新しいデータ型とそれに対する演算を定義する機能がある．本稿では，この機能を用いて SCILAB にべき級数演算パッケージを導入した．このパッケージの特徴は

(a) 多変数のべき級数（全次数で打ち切り）が自然な形で扱える．
(b) べき級数の乗算表をあらかじめ計算しておくことや，べき級数の構造に関するデータを別のデータ型に定義しておく等の工夫により，SCILAB 上のプログラムとしては高速に計算を行う．
(c) べき級数を通常の数と同様に自然な形で扱うことが出来る

である．実際にベンチマークテストの結果より，*Mathematica* Ver 5.0 と比較しても 2 変数以上のべき級数の演算速度に関しては同等かそれ以上であることが見て取れた（1 変数のべき級数の演算速度に関しては *Mathematica* の 1/5 程度である）．

このべき級数演算パッケージを用いて，制御系設計用パッケージを作成した．パッケージの中には，多項式行列の逆行列や固有値，固有ベクトルをべき級数の形で計算する関数や，代数 Riccati 方程式をべき級数の形で解く関数，システムのピーク値やそのピーク値をとる時間をパラメータのべき級数で計算する関数などが含まれる．これらの関数を用いて，実際に制御系設計を行った．設計目的は，H_2 最適制御でステップ応答のオーバーシュートと逆応答を制限する制御系を構成することである（従来の設計法でこの設計目的を達成するには，重み行列を変えながらステップ応答を調べるという試行錯誤的な手法を取らざるを得ない）．本稿のべき級数演算パッケージを用いることにより，設計パラメータを記号の形で残したまま，オーバーシュートや逆応答の最大値をべき級数の形で計算することが可能になり，効率的に目的とする制御系を構成することが出来た．

参 考 文 献

[1] Scilab Group: Scilab 入門,
ftp://ftp.inria.fr/INRIA/Scilab/contrib/SCIJAPANESE/intro-jp.pdf

[2] Kitamoto, T.: Approximate eigenvalues, eigenvectors and inverse of a matrix with polynomial entries, *Jpn. J. Indus. Appl. Math.*, **11**(1), 1994, 73-85.

[3] 北本: 近似固有値, 固有ベクトルとその最適制御への応用, 電子情報通信学会論文誌, **J78-A**(4), 1995, 531–534.

[4] 北本: 近似代数を用いた H_2 最適制御系の解析と設計, 電子情報通信学会論文誌, **J81-A**(2), 1998, 289–292.

[5] 北本: 代数的 Riccati 方程式のべき級数解の計算法について, 電子情報通信学会論文誌, **J81-A**(3), 1998, 445–447.

[6] 北本: 近似固有値の計算法について, 電子情報通信学会論文誌, **J81-A**(4), 1998, 803–806.

[7] 北本：近似逆行列の高次収束への拡張について, 電子情報通信学会論文誌, **J84-A**(2), 2001, 243–245.

[8] Kitamoto, T.: On computation of approximate eigenvalues and eigenvectors, *IEICE Trans. Fundamentals*, **E85-A**(3), 2002, 664–675.

[9] Kitamoto, T.: Computation of the peak of time response in the form of formal power series, *IEICE Trans. Fundamentals*, **E86-A**(12), 2003, 3240–3250.

[10] Kung, H.T., Traub, J.F, All algebraci function can be computed fast, *J. ACM*, **25**, 1978, 245–260.

[11] Lipson, J.D., Newton's method: A great algebraic algorithm, *Proceedings of ACM Symposium on Symbolic and Algebraic Computations*, 1976, 260–270.

[12] Yokoyama, K., Takeshima, T.: On Hensel construction of eigenvalues and eigenvectors of matrices with polynomial entries, *Proceedings of the ISSAC '93*, ACM PRESS, 1993, 218-224.