システム/制御/情報, Vol. 55, No. 5, pp. 196–203, 2011

「制御系解析・設計における数値計算／数式処理ソフトウェアの活用」特集号

解　説

数式処理を用いた非線形制御系の解析・設計

藤本　健治*

1. はじめに

線形システムの制御理論が整備されるとともに，制御系の解析・設計を行うためのソフトウェアも進歩してきた．行列を扱える数値計算ソフトウェア MATLAB/Simulink を用いれば，線形制御則の設計や非線形の制御対象を含めたシミュレーションなど，制御系の解析・設計に関する実用的な操作を容易に実行することができる．これに比べて非線形制御系の解析・設計については，学術的には多くの手法が提案されているが，それらを実用的に実行するためのソフトウェアは整備されているとはいえない状況である．非線形制御系の解析・設計を行うためには，関数の微分・積分などの操作が必要であり，線形制御の場合とは異なり Maple, Mathematica, Maxima などの数式処理ソフトウエアが必要となる．本稿では，その一例として Maple をとりあげ，実際の解析・設計に必要なツールとその使い方を紹介する．

現代制御において安定性や制御性能の評価は，行列の演算と固有値や行列のたかだか2次の方程式によって記述されていた．そのため，行列と固有値を扱える数値計算ソフトウェアが重要な役割をはたしている．一方，非線形制御においては安定性を保証するのはリアプノフ関数であり，可制御性などを調べるには微分幾何の道具を用いる必要がある．これらの道具は微分方程式および偏微分方程式で記述されるため，微積分が扱える数式処理のソフトウェアが重要となる．残念ながら，MATLAB のように制御系の解析・設計に特化したコマンド群は用意されていないが，非線形制御において必要な典型的な操作については，十分な数のコマンドが用意されている．また C, Fortran, MATLAB といった他のソフトウェアとの連携もしやすいものとなっている．

本稿ではいくつかの典型的な非線形制御系の解析・設計問題を例にとって，プログラムの例文を交えながらその使い方を紹介したい．制御系の性能の解析問題は微分操作のみで実行できるものが多く，非常に簡単に実行できるが，補償器設計問題は積分操作（偏微分方程式の求解も含む）が必要なものが多く，多少のテクニックを要する．とくに設計問題としては，実装が比較的容易なフィードバック線形化と，問題の記述が一般的で多くの問題に適用できる最適制御の二つの手法に焦点をしぼって，解説を行う．

* 名古屋大学 大学院 工学研究科

Key Words: Maple, nonlinear control, partial differential equation, optimal control, feedback linearization.

2. Maple にできること

制御系の設計・解析において広く用いられているソフトウェアは，行列の数値計算を行える MATLAB, Scilab, Octave などである．これらと比べた Maple, Mathematica などの数式処理ソフトウェアの大きな利点の一つはシンボリック演算ができることである．たとえば $f(x)=\sin x$ の x に関する微分 $\mathrm{d}f/\mathrm{d}x=\cos x$ や積分 $\int f(x)\mathrm{d}x=-\cos x$ の演算が可能であり，これらの操作を必要とする問題には非常に有用である．以下では Maple の操作の基本と扱える微分・積分演算を簡単にまとめておく．

2.1 操作の基本

Maple を使いこなすうえで，まず知っておく必要があるのはヘルプブラウザである．このヘルプは例題が充実しているため，初めて使用するコマンドは一度ヘルプを確認するとよい．また文末にコロン「:」をつけると出力は表示せず，文末にセミコロン「;」をつけると出力を表示する（Maple 最新版では何もつけなくてもこの動作）．本稿のサンプルプログラムにおいても，紙面の都合により重要でない部分はコロンをつけている．

また Maple のコマンドは複数のパッケージに分かれている．たとえばパッケージ名「Package」に含まれる「Function」という関数を使いたい場合には，以下の2通りの方法がある．

- `Package[Function](引数);`
- `with(Package):`
  `Function(引数);`

非線形制御においてよく用いられるパッケージは，`LinearAlgebra`（線形代数），`VectorCalculus`（ベクトル解析），`DEtools`（微分方程式），`PDEtools`（偏微分方程式），`DifferentialGeometry`（微分幾何）などである．

また Maple を使って，制御系の設計解析に必要な演算を行ったあとに，計算結果を他のソフトウェアに移したいことがよくある．たとえば設計は Maple を用いて行うが，数値シミュレーションや実機への実装は

MATLAB, C, Fortran などを用いて行いたいという場合である．これらソフトウェアにおいては，数式がそれぞれ違ったフォーマットで記述されており，同じ関数でも違う表現になっているのが普通である．Maple 上の関数は，CodeGeneration というパッケージを使うと各種ソフトウェアに適した形で書き出すことが可能である．たとえば，以下のコマンドで，Maple の arcsin(x) という関数を MATLAB の表現 asin(x) に改めて出力している．

入力
```
CodeGeneration[Matlab](arcsin(x));
```
出力
```
cg = asin(x);
```

2.2 微分

非線形制御においてもっともよく使う演算は微分操作である．たとえば，もっとも簡単なスカラ関数の（偏）微分 $\partial\sin(x_1^2+x_1x_2)/\partial x_1$ の演算は以下のように行う．

入力
```
diff(sin(x1^2+x1*x2),x1);
```
出力
```
cos(x1^2 + x1 x2)(2 x1 + x2)
```

また VectorCalculus パッケージを用いると，多変数関数 $F:\mathbb{R}^n\to\mathbb{R}^m$ の偏微分が可能である．以下では関数 F のヤコビ行列を求めている．

入力
```
with(VectorCalculus):
F := [tan(x2),u/cos(x2)];
Jacobian(F,[x1,x2,u]);
```
出力
$$\left[\tan(x2)\quad \frac{u}{\cos(x2)}\right]$$
$$\begin{bmatrix} 0 & 1+\tan(x2)^2 & 0 \\ 0 & \dfrac{u\sin(x2)}{\cos(x2)^2} & \dfrac{1}{\cos(x2)} \end{bmatrix}$$

2.3 積分

積分とは微分と逆の演算であり，微分方程式を解く手続きでもある．たとえば，スカラ関数 $\cos(x_1^2+x_1x_2)(2x_1+x_2)$ の x_1 に関する不定積分を求める操作は以下のように書ける．

入力
```
int(cos(x1^2+x1*x2)*(2*x1+x2), x1);
```
出力
```
sin(x1^2 + x1 x2)
```

スカラの場合とは異なり，多変数関数の積分はいつも可能というわけではない．たとえば，関数 $f:\mathbb{R}^n\to\mathbb{R}$ の微分方程式

$$\frac{\partial f}{\partial x}=g(x)^{\mathrm{T}} \tag{1}$$

を満たす関数 f が存在するのは，

$$\frac{\partial g}{\partial x}=\frac{\partial g}{\partial x}^{\mathrm{T}} \tag{2}$$

が成立するときのみである．このように多変数の「積分」の問題は通常「偏微分方程式」として表現され，かつ常に可解かどうかわからない．非線形制御に現れる多くの問題はこのような積分問題である．たとえば，(1) 式において

$$g(x)=\begin{pmatrix} x_2^2 \\ 2x_1x_2 \end{pmatrix},\quad x=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

の場合に f を求める問題は，以下のように書ける．2行目で (2) 式を確認し，3行目で解 f を導出している．なお，出力結果の $\_C1$ は積分定数である．

入力
```
g := [x2^2, 2*x1*x2];
Jacobian(g,[x1,x2]);
pdsolve([diff(f(x1,x2),x1) = g[1],
         diff(f(x1,x2),x2) = g[2]]);
```
出力
$$\begin{bmatrix} 0 & 2\,x2 \\ 2\,x2 & 2\,x1 \end{bmatrix}$$
$$\left\{f(x1,x2)=x2^2x1+\_C1\right\}$$

3. 非線形制御と微分積分

非線形制御において微分積分の演算はなくてはならないツールである．その典型的な使い方を例題を用いて紹介する．本稿では制御対象としてつぎのような入力にアファインな（入力に関して1次式の）状態方程式を考える．

$$\frac{\mathrm{d}x}{\mathrm{d}t}=F(x,u)=f(x)+g(x)u \tag{3}$$

ここで，状態 $x(t)\in\mathbb{R}^n$，入力 $u(t)\in\mathbb{R}^m$ である．非線形制御では，状態推定は未解決の問題の一つであるので，本稿では状態 x が観測できるものとする．

3.1 物理システムのモデル化と偏微分

典型的な非線形の制御対象の一つにロボットを含むメカトロニクス系がある．これらの系は停留条件などの偏微分を用いて運動方程式が記述されるため，Maple などのツールが直接役に立つ．たとえば Lagrange の運動方程式

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L(q,\dot{q})}{\partial\dot{q}}^{\mathrm{T}}-\frac{\partial L(q,\dot{q})}{\partial q}^{\mathrm{T}}=u$$

や電気回路を表す Brayton-Moser 方程式 [1]

$$Q(x)\dot{x}=-\frac{\partial P(x)}{\partial x}^{\mathrm{T}}+g(x)u$$

などは，Maple の微分機能を使って直接状態方程式を導ける．ロボットなどは式がとても複雑になり，手計算では不可能に近い計算量の操作をソフトウェアに代行させることができる．

また得られた状態方程式 (3) を平衡点 $x=x_0$, $u=u_0$ ($0=f(x_0,u_0)$ を満たす点) のまわりで近似線形化

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{x}=\underbrace{\left.\frac{\partial F}{\partial x}\right|_{(x,u)=(x_0,u_0)}}_{A}\bar{x}+\underbrace{\left.\frac{\partial F}{\partial u}\right|_{(x,u)=(x_0,u_0)}}_{B}\bar{u} \quad (4)$$

する際にも，Maple などを用いれば，物理パラメータの変数をそのまま含んだ A, B 行列が簡単に計算できる．ここで $\bar{x}\approx x-x_0$, $\bar{u}\approx u-u_0$ は平衡点からの変位である．たとえば2状態1入力の系に対して A, B 行列の計算は以下のように書ける．ここで subs および evalf は，それぞれ代入と浮動小数点による評価を行う命令である．

入力

```
1  A := evalf(subs({x1=0,x2=0,u=0},Jacobian(F,[x1,x2])));
2  B := evalf(subs({x1=0,x2=0,u=0},diff(F,u)));
```

出力

$$\begin{bmatrix} 0. & 1. \\ 0. & 0. \end{bmatrix}$$
$$\begin{bmatrix} 0. & 1. \end{bmatrix}$$

3.2 微分幾何と偏微分

線形システム (4) の可制御性は可制御性行列

$$M_c:=\left(B,AB,\cdots,A^{n-1}B\right)$$

がフルランクか否かで判別できる．非線形制御においても，これと同様の局所的な可制御性（可到達性）は，微分幾何の演算を用いて調べることができる．具体的には以下の分布（状態 x に依存した部分空間）のランクが最大の n に一致するかどうかで判別できる [2,3].

$$D_c^n(x):=\mathrm{span}\left\{g(x),\mathrm{ad}_f g(x),\cdots,\mathrm{ad}_f^{n-1}g(x)\right\}$$

なお，この ad の演算はリー（かっこ）積とよばれ，局所座標系では次式で定義される．

$$[f,g]:=\frac{\partial g}{\partial x}f-\frac{\partial f}{\partial x}g$$
$$\mathrm{ad}_f^{k+1}g:=[f,\mathrm{ad}_f^k g]$$
$$\mathrm{ad}_f^0 g:=g$$

また，ベクトル場 $\dot{x}=f(x)$ に沿った，x の関数 $\phi(x)$ の k 階の時間微分は，

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\phi(x)=\mathrm{L}_f^k\phi(x)$$

のようにリー微分 $\mathrm{L}_f^k\phi$ を用いて計算することができる．その定義はつぎで与えられる．

$$\mathrm{L}_f^{k+1}\phi(x):=\frac{\partial}{\partial x}(\mathrm{L}_f^k\phi)f(x)$$
$$\mathrm{L}_f^0\phi(x):=\phi(x)$$

なおこれらの演算は，DifferentialGeometry パッケージの中に LieBracket, LieDerivative としてコマンドが用意されているほか，手続き型関数を用いてつぎのように自分専用のコマンドとして定義することもできる．

入力

```
1  MyL := proc(phi,f,x,k)
2   local i, result:
3   with(VectorCalculus):
4   result := phi:
5   for i from 1 to k do
6    result := evalm(Jacobian([result],x) &* f)[1]:
7   end do:
8   result;
9  end proc:
10 MyL(x1,F,[x1,x2],2);
```

出力

$$\frac{(1+\tan(x2)^2)u}{\cos(x2)}$$

この例では，1〜9行目でリー微分を計算する MyL という関数を定義して，最後の行で実際に $\mathrm{L}_F^2 x_1$ を計算している．このように，可制御性などの性能の解析は微分演算が主体であり，数式処理ソフトウェアをそのまま用いることができる．

3.3 リアプノフ関数と偏微分方程式

古典制御ではラプラス変換と複素関数論，現代制御では線形代数と多変数の1次もしくは2次方程式が重要な役割を果たしている．非線形制御でこれに代わるツールは偏微分方程式である．ここでは典型的な非線形制御における問題を偏微分方程式の形をもとに分類してみよう．

ここでは制御対象として (3) 式の状態方程式を考える．この系を状態フィードバック

$$u=k(x) \quad (5)$$

で漸近安定化する問題は，フィードバック系

$$\frac{\mathrm{d}x}{\mathrm{d}t}=f(x)+g(x)k(x) \quad (6)$$

に対して，リアプノフ関数 $V(x)$ を見つける問題である．リアプノフ関数とは，それ自身正定関数，すなわち

$$V(x)\begin{cases}=0 & (x=0)\\ >0 & (x\neq 0)\end{cases}$$

を満たす関数（$V\succ 0$ と書く）であり，その時間微分 $\dot{V}$ は負定関数（$-\dot{V}$ が正定関数であり $\dot{V}\prec 0$ と書く）となるものである．リアプノフ関数を見つけることができれば，制御系 (3) は漸近安定であることがわかる．この $\dot{V}$ を書き下すと，

$$\dot{V}=\frac{\partial V}{\partial x}(f(x)+g(x)k(x))$$

となり，$W(x):=-\dot{V}(x)$ とおくと

$$\frac{\partial V}{\partial x}(f(x)+g(x)k(x))=-W(x) \tag{7}$$

を満たす $V(x), W(x), k(x)$ で，$V, W\succ 0$ となるものを求める問題となる．これは1階の偏微分を含む未知関数に関する2次（双1次）の方程式であり，非線形系の安定化フィードバック設計問題（以下「安定化問題」とよぶ）の多くはこのように1階2次の偏微分方程式となる．なお，状態フィードバック $k(x)$ が与えられているときにフィードバック系 (6) の安定性を調べる問題（以下，「安定性解析問題」とよぶ）は，未知関数 $V(x)$, $W(x)$ に関する1階1次の偏微分方程式となる．たとえば，閉ループ系が $\dot{x}=-\sinh(x)$ となるスカラ系に対して $W(x)=x^2$ と選んでリアプノフ関数 $V(x)$ を求めてみよう．コマンド `dsolve` は，スカラ変数の微分方程式を解いてくれる．

入力

```
dsolve(-diff(V(x),x)*sinh(x) + x^2 = 0);
```

出力

$$V(x)=x^2\ln(1-\mathrm{e}^x)+2x\mathrm{polylog}(2,\mathrm{e}^x)-2\mathrm{polylog}(3,\mathrm{e}^x)$$
$$-x^2\ln(1+\mathrm{e}^x)-2x\mathrm{polylog}(2,-\mathrm{e}^x)+2\mathrm{polylog}(3,-\mathrm{e}^x)+\_C1$$

このように非線形制御の問題の多くは1階の偏微分方程式となり，安定性解析問題は1次，安定化問題は2次となることが多い．1次は比較的容易に解けるが，2次は難しい問題である．

3.4 テイラー展開を用いた近似解法

前節 **3.3** で述べたように非線形制御系の設計問題のほとんどは偏微分方程式を解く問題となる．まれに解析解が見つかる場合もあるが，多くの場合は解析的に解くことができず，近似解法を用いることになる．近似解法のうち，もっともポピュラーな方法はテイラー展開を用いる方法である．求めるべき未知関数 $\phi:\mathbb{R}^n\to\mathbb{R}$ を

$$\phi(x)=\sum_{k_1,\ldots,k_n} c_{k_1,\ldots,k_n}x_1^{k_1}\cdots\cdots x_n^{k_n}$$

のように x_i に関する多項式で展開し，その係数 $c_{i,\ldots,j}$ を求める方法である．この ϕ を解きたい偏微分方程式に代入し，偏微分方程式全体をテイラー展開・係数比較することで，問題は係数 $c_{i,\ldots,j}$ に関する代数方程式に変換される．偏微分の方程式が未知パラメータ ϕ に関する p 次式である場合には，対応する代数方程式も係数 $c_{i,\ldots,j}$ に関する p 次式になる．したがって，たとえば $p=1$ である場合には1次の代数方程式となり非常に容易に解くことができるが，$p\geq 2$ の場合には解くのに工夫が必要となる．たとえば **3.3** で扱ったリアプノフ関数を用いた安定化問題 (7) などのように，未知パラメータに関して双線形な場合には，片方を固定して交互に求解していく方法がよく用いられる．

前節 **3.3** と同じ例題をテイラー展開を用いて解いてみよう．解 V の候補としてテイラー展開の4次までの関数 $V_4=c_1x+c_2x^2+c_3x^3+c_4x^4$ を考え，これを $W=x^2$, $k=0$ とともにリアプノフ方程式 (7) に代入して係数 $c_1,\ldots,c_4$ を求めよう．コマンド `mtaylor`, `match` はそれぞれ，指定次数以下のテイラー展開と係数比較を行ってくれる命令である．

入力

```
1  V4 := c1*x + c2*x^2 + c3*x^3 + c4*x^4:
2  eq4 := mtaylor(-diff(V4,x)*sinh(x)+x^2,x,5):
3  match(0 = eq4, x, 'sol4'):
4  sol4;
```

出力

$$\left\{c1=0,\ c2=\frac{1}{2},\ c3=0,\ c4=-\frac{1}{24}\right\}$$

ここまででテイラー展開をもとにした手法を説明したが，同様に既知の基底関数 $\psi_k(x)$, $k=1,2,\ldots$ の線形和として近似解を求める方法も提案されている．この手法は一般には**ガラーキン近似**の一種であり，分布定数系のモデル化などの分野で広く研究されている．テイラー展開を用いた近似解法は，この基底 $\psi_k(x)$ として x の要素の単項式を選んだ場合と考えられる．

基底関数展開やテイラー展開をもとに近似を行った場合には，得られた近似解の正定性などの性質を事前に保証することができない．この点を改善するのが SoS (Sum of Squares)[4] とよばれる手法である．そのアイデアは以下のとおり．基底関数の組を並べたベクトル

$$\psi(x):=(\psi_1(x),\ldots,\psi_N(x))^{\mathrm{T}}$$

を作り，このベクトル $\psi(x)$ の2次形式として解を表現する．

$$\phi(x)\equiv\psi(x)^{\mathrm{T}}P\psi(x),\quad P^{\mathrm{T}}=P\in\mathbb{R}^{N\times N}$$

このようにパラメータを対称行列 P で表現することで，関数 ϕ の正定性とパラメータ行列 P の正定性が等価となる．解きたい問題をこの行列 P の LMI (Linear Matrix Inequality) などの半正定値行列の最適化問題に変換できれば，効率よく正定関数の偏微分方程式を解くことができる．

4. フィードバック線形化

前章の **3.3** で述べたように，非線形制御系の設計問題の多くは1階2次の偏微分方程式となり，一般には解くことが困難である．また **3.4** で紹介したテイラー展開などの近似解法を用いても，代数の2次方程式となり，やはり簡単には解くことができない．この章では，設計問題が1階の偏微分方程式となり非線形制御手法の中では比

較的容易に実装できるフィードバック線形化手法[2,5,3]を紹介する．

4.1 入出力線形化

入出力線形化とは，(3)式のような非線形系に対して，出力信号 $y=h(x)$ が与えられているときに，フィードバック入力変換 $u=\alpha(x)+\beta(x)\bar{u}$ を用いて，新しい入出力 $\bar{u}\mapsto y$ の挙動が線形システムとなるようにする手法である．こうすることで，出力信号 y は制御入力 $\bar{u}$ を使って線形の制御則で制御することが可能になる．

いま y の高階の時間微分を計算していって，はじめて入力 u の影響が現れる微分の階数を ρ とする．この ρ を**相対次数**とよぶ．この条件をリー微分を使って表すと，

$$\begin{aligned}
\dot{y} &= \mathrm{L}_{f+gu}h(x) = \mathrm{L}_f h(x) + \underbrace{\mathrm{L}_g h(x)}_{=0} u \\
\ddot{y} &= \mathrm{L}_f^2 h(x) + \underbrace{\mathrm{L}_g \mathrm{L}_f h(x)}_{=0} u \\
&\vdots \\
y^{(\rho-1)} &= \mathrm{L}_f^{\rho-1} h(x) + \underbrace{\mathrm{L}_g \mathrm{L}_f^{\rho-2} h(x)}_{=0} u \\
y^{(\rho)} &= \mathrm{L}_f^{\rho} h(x) + \underbrace{\mathrm{L}_g \mathrm{L}_f^{\rho-1} h(x)}_{\neq 0} u
\end{aligned} \tag{8}$$

この条件を満たすとき，上式の最後の行(8)より，フィードバック入力変換

$$u = \underbrace{-\frac{\mathrm{L}_f^{\rho} h(x)}{\mathrm{L}_g \mathrm{L}_f^{\rho-1} h(x)}}_{=\alpha(x)} + \underbrace{\frac{1}{\mathrm{L}_g \mathrm{L}_f^{\rho-1} h(x)}}_{=\beta(x)} \bar{u} \tag{9}$$

によって $\bar{u}\mapsto y$ がつぎのように線形系（ρ 次の積分器）になることがわかる．

$$y^{(\rho)} = \bar{u}$$

また y の時間微分を並べたものを新たな状態変数として

$$\begin{aligned}
\bar{x} &= \Phi(x) := (y, \dot{y}, \cdots, y^{(\rho-1)})^{\mathrm{T}} \\
&= (h(x), \mathrm{L}_f h(x), \cdots, \mathrm{L}_f^{\rho-1} h(x))^{\mathrm{T}} \in \mathbb{R}^{\rho}
\end{aligned} \tag{10}$$

のように定義すると，$\bar{x}$ の挙動は $\bar{u}$ を入力とした線形系となる．この変換法を**入出力線形化**という．

$$\dot{\bar{x}} = \begin{pmatrix} 0 & 1 & & 0 \\ \vdots & \ddots & \ddots & \\ \vdots & & & 1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \bar{x} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \bar{u} \tag{11}$$

この系は，線形制御則を用いて制御可能である．ここで必要なのは，(9)式のフィードバック則と(10)式の新しい座標 $\bar{x}$ の計算であり，これらはリー微分だけで計算可能であるので，**2.2**で紹介したツールを使って計算できる．

ただしここで注意してほしいのは，もとの系(3)の次元 n と線形化された系(11)の次元 $\rho(\leq n)$ が異なる点である．この差 $n-\rho$ の次元の状態変数の動きが見えなくなっており，y を0にするように安定化制御を行ったとしても，見えない $n-\rho$ 個の状態は不安定になるということも起こり得る．この見えない状態の挙動を**ゼロダイナミクス**とよび，入出力線形化で制御を行うためにはゼロダイナミクスが漸近安定である必要がある．このゼロダイナミクスは線形系の零点の挙動に対応しており，(3)に出力 $y=h(x)$ を組み合わせた系の入出力 $u\mapsto y$ の線形近似系が安定な零点をもてば，ゼロダイナミクスは漸近安定であることがしられている．したがって，この安定性もMapleを用いて計算できる．なお，ゼロダイナミクスが漸近安定である系を，線形の場合の類似の性質から**最小位相系**とよぶ．

例題としてつぎのような2次元系を考えよう[1]．

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \tan x_2 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1/\cos x_2 \end{pmatrix} u \tag{12}$$

この系に対して出力 $y=x_1$ を用いて入出力線形化してみよう．**3.2**で定義したMyLというリー微分を計算するコマンドを使って，相対次数を計算してみよう．

入力

```
1  xx := [x1,x2]:
2  f := [tan(x2),0]:
3  g := [0,1/cos(x2)]:
4  h := x1:
5  MyL(h,g,xx,1);
6  MyL(MyL(h,f,xx,1),g,xx,1);
```

出力

$$\begin{matrix} 0 \\ \dfrac{1+\tan(x2)^2}{\cos(x2)} \end{matrix}$$

この計算から $\mathrm{L}_g h=0, \mathrm{L}_g \mathrm{L}_f h\neq 0$ すなわち相対次数は $\rho=2$ であることがわかる．すなわちこの場合は，$\rho=n$ であるのでゼロダイナミクスは存在しない．よって線形化の新座標(10)およびフィードバック(9)の α, β, Φ はつぎのように計算できる．

入力

```
1  alpha := -MyL(h,f,xx,2)/MyL(MyL(h,f,xx,1),g,xx,1);
2  beta := 1/MyL(MyL(h,f,xx,1),g,xx,1);
3  Phi := [h,MyL(h,f,xx,1)];
```

出力

$$\begin{matrix} 0 \\ \dfrac{\cos(x2)}{1+\tan(x2)^2} \end{matrix}$$

$$[\ x1 \quad \tan(x2)\ ]$$

[1] この例は文献[5]から引用した．

4.2 厳密な線形化

前節で紹介した入出力線形化法は，Maple の微分の機能だけを用いて実装できるため非常に簡単に利用することができるが，ゼロダイナミクスという制御できない状態ができてしまうという問題があった．最小位相系であれば安定化制御は可能であるが，ゼロダイナミクスの挙動を自由に操ることができず，結果として制御対象の全状態を精密に制御することはできない．入出力線形化においては，出力 $y=h(x)$ は設計パラメータとして設計者が与えるものであったが，ここでは $\rho=n$ となってゼロダイナミクスが存在しないような出力を見つける問題を考えよう．$\rho=n$ の場合の入出力線形化を，**厳密な線形化**とよび，その時の出力 $y=\phi(x)$ を**線形化出力**とよぶ．(8) 式から，線形化出力関数 ϕ の満たすべき条件は $\rho=n$ として

$$\begin{aligned} &\mathrm{L}_g\mathrm{L}_f^k\phi(x)=0, \quad k=0,1,\ldots,n-2 \\ &\mathrm{L}_g\mathrm{L}_f^{n-1}\phi(x)\neq 0 \end{aligned}$$

と書けるが，これは ϕ に関する高階の偏微分方程式となり，解きやすい問題ではない．しかしリー代数の基本公式を用いるとつぎのような 1 階の線形な偏微分方程式に書き直すことができ，Maple を用いて比較的容易に解を求めることができる．

$$\mathrm{L}_{\mathrm{ad}_f^k g}\phi(x)=0, \quad k=0,1,\ldots,n-2 \tag{13}$$

$$\mathrm{L}_{\mathrm{ad}_f^{n-1} g}\phi(x)\neq 0 \tag{14}$$

さらに上の偏微分方程式が可解かどうかの確認も，微分演算のみで行うことができ，つぎのような二つの条件の成立と上の偏微分方程式の可解性が等価であることがしられている（**フロベニウスの定理**）．

- D_c^n のランクが n.
- D_c^{n-1} が**対合的**，すなわち

$$a,b\in D_c^{n-1} \quad \Rightarrow \quad [a,b]\in D_c^{n-1}$$

これらの演算はやはり微分演算のみを用いているため，Maple を使って簡単に実装可能である．さて，この手法を (12) 式の系に適用してみよう．$n=2$ であることから，D_c^1 は 1 次元の分布となるため，対合性の条件は自動的に成立する．また D_c^2 のランク条件も (12) 式の線形近似系が可制御であることから自動的に満たされる．すなわち (13), (14) 式を満たす ϕ の存在がわかる．(13) 式は $\mathrm{L}_g\phi=0$, (14) 式は $\mathrm{L}_{[f,g]}\phi\neq 0$ となるため，下記のプログラムの 1 行目で，等式である (13) 式を解き，つぎに 3 行目で，その解が (14) 式を満たすことを確認している．

入力

```
1 sol := pdsolve(evalm(Jacobian([phi(x1,x2)],xx)&*g)[1]=0);
2 fg := evalm(Jacobian(g,xx)&*f-Jacobian(f,xx)&*g):
3 evalm(Jacobian([subs(sol,phi(x1,x2))],xx)&*fg)[1];
```

出力

$$\phi(x1,x2)=\_F1(x1) - \frac{\left(\frac{\mathrm{d}}{\mathrm{d}x1}\_F1(x1)\right)(1+\tan(x2)^2)}{\cos(x2)}$$

計算結果における $\_F1(x1)$ は $x1$ の任意の関数を表す．この計算によって，前節 **4.1** で選んだ出力 $\phi(x)=x_1$ が解の一つになっていることが確認できる．

5. 最適制御

最適制御は状態空間モデルが提案された当初からよくしられた問題であり，長い歴史を有する制御手法である [6]．汎用性のある手法であるが，以下に述べるように解くのに工夫が必要である．

5.1 最適制御とハミルトン・ヤコビ方程式

最適制御問題とは，制御対象 (3) に対してつぎのような評価関数を設定し，これを最小とする制御入力 $u=u^\star(x)$ を求める問題である．

$$J=\int_0^\infty \underbrace{\left(q(x(t))+\frac{1}{2}u(t)^{\mathrm{T}}R(x(t))u(t)\right)}_{L(x,u)}\mathrm{d}t \tag{15}$$

この問題に対する解も古くからしられており，次式の**ハミルトン・ヤコビ方程式**

$$\frac{\partial V}{\partial x}f(x)-\frac{1}{2}\frac{\partial V}{\partial x}g(x)R(x)^{-1}g(x)^{\mathrm{T}}\frac{\partial V}{\partial x}^{\mathrm{T}}+q(x)=0 \tag{16}$$

を満たすスカラ関数 $V:\mathbb{R}^n\to\mathbb{R}$ を用いて，最適入力は以下で与えられることがわかっている．

$$u^\star(x)=-R(x)^{-1}g(x)^{\mathrm{T}}\frac{\partial V}{\partial x}^{\mathrm{T}} \tag{17}$$

ただしハミルトン・ヤコビ方程式 (16) を満たす解 V は，一般に複数あるが，ここではそのうち $\dot{x}=f+gu^\star$ を漸近安定にする正定解を用いるものとする．

ハミルトン・ヤコビ方程式 (16) を解く最適制御は，安定化問題の一種であり，未知関数 V に関する 1 階 2 次の偏微分方程式である．2 次であるので一般には解くことが難しいため，これをつぎのような漸化式の形に書き換える．

$$\begin{aligned} &\frac{\partial V_{[k+1]}}{\partial x}\left(f(x)+g(x)u^\star_{[k]}(x)\right)+L^\star_{[k]}(x)=0 \\ &u^\star_{[k]}=-R(x)^{-1}g(x)^{\mathrm{T}}\frac{\partial V_{[k]}}{\partial x}^{\mathrm{T}} \\ &L^\star_{[k]}=q(x)+\frac{1}{2}u^\star_{[k]}{}^{\mathrm{T}}R(x)\,u^\star_{[k]} \end{aligned}$$

ただしパラメータ $k=0,1,2,\ldots$ は繰り返し計算における繰り返し回数を表しており，k の増加にともなって解の精度がよくなっていく．このような方法は**漸近近似** [7] とよばれ，同様の手法が多数提案されている [8,9]．この

手法の各ステップでは$u^\star(x)$（および$L^\star(x)$）を固定し，1階1次の$V_{[k+1]}$に関する偏微分方程式を解けばよい形になっている．したがって，これまでに説明したMapleのテクニックが直接使えることになる．繰り返し計算の初期値として$V_{[0]}$を与える必要があるが，これは解くべき問題を線形近似（制御対象は線形近似し，評価関数は2次のテイラー近似）した場合の解を用いるのが一般的である．とくに，古くからしられているテイラー展開に基づく解法[8]は，各ステップkにおいて$V_{[k]}$のテイラー展開の$k+2$次以下の項だけを求めていく方法となっている．テイラー展開を用いた近似解法は，アルゴリズムの作成が容易であるため学術研究ではよく用いられるが，一般に真の解への収束が遅いことが多く，問題に応じて適切な基底を選ぶことが望ましい[9]．

ハミルトン・ヤコビ方程式を解くこれ以外の方法としては，状態空間をメッシュ状に区切って区分ごとに近似解を求める方法[10–13]が比較的多く研究されており，また常微分方程式に帰着させる[14,15]方法なども提案されている．

5.2 制御リアプノフ関数と逆最適制御

さて，前節**5.1**で説明した最適制御手法は，さまざまな制御問題に適用できる汎用性のある手法であるが，その計算は基本的に漸近的なアルゴリズムとなり，計算を繰り返して解の精度をよくしていく方法である．繰り返し計算の各ステップにおける演算は，1階1次の偏微分方程式となり，Mapleなどの数式処理ソフトウェアで実装可能であるが，どのくらいの計算を行うとどのくらいの精度の解が得られるのかわからないといった問題点がある．ここではできるだけ労力をかけずにある種の最適制御則を求めることができる逆最適補償器[16,17]とその実装方法を紹介する．

制御対象(3)に対してそのリアプノフ関数$V:\mathbb{R}^n\to\mathbb{R}$の候補がすでに与えられているとする．すなわち，ある状態フィードバック則$u=k(x)$が存在し，閉ループ系がVをリアプノフ関数として漸近安定になるものとする．このようなリアプノフ関数の候補のことを**制御リアプノフ関数**とよび，正確にはつぎの関係を満たすものである．

$$\mathrm{L}_gV=0 \quad\Rightarrow\quad \mathrm{L}_fV<0$$

さて，そのような制御リアプノフ関数Vが既知である場合には，どのような制御を行えばよいであろうか．非線形最適化における**最急降下法**のように，関数Vの値を最も大きく減少させる入力の方向，すなわち$-(\text{ゲイン})\times\mathrm{L}_gV^{\mathrm{T}}$の方向に制御入力を加えてやることが方針として考えられる．このような形の制御則をL_gV**補償器**とよぶ．たとえば最適制御則(17)は，ゲインが$R(x)^{-1}$の場合のL_gV補償器であると見なすことができる．このようなL_gV補償器のうち，つぎのようなゲインを選んだものが**逆最適補償器**とよばれるものである．

$$u=\begin{cases}-\dfrac{\mathrm{L}_fV+\sqrt{\mathrm{L}_fV^2+\|\mathrm{L}_gV\|^4}}{\|\mathrm{L}_gV\|^2}\mathrm{L}_gV^{\mathrm{T}} & (\mathrm{L}_gV\neq0)\\ 0 & (\mathrm{L}_gV=0)\end{cases} \tag{18}$$

この制御則は，実は(15)式の評価関数においてq, Rをつぎのように選んだ場合の最適制御の解になっていることがわかる．容易に確かめられるのでご確認いただきたい．

$$\begin{aligned}q(x)&=\|\mathrm{L}_gV\|^2+\sqrt{\mathrm{L}_fV^2+\|\mathrm{L}_gV\|^4}\\ R(x)&=\frac{\|\mathrm{L}_gV\|^2}{2(\mathrm{L}_fV+\|\mathrm{L}_gV\|^2+\sqrt{\mathrm{L}_fV^2+\|\mathrm{L}_gV\|^4})}I\end{aligned} \tag{19}$$

したがってリアプノフ関数の候補さえ与えられれば，前節のようにハミルトン・ヤコビ方程式を解かずにある種の最適制御補償器を作ることができる．ただ，評価関数のパラメータq, Rは，上式のように決まってしまっており，設計者が指定できるものではないため設計自由度はほとんどない．フィードバック線形化など，他の手法で設計した制御則を作り直して，より性能の良い制御則を作れるため，**リアプノフ再設計法**ともよばれる．

この手法は微分の操作のみで実装できるため，Mapleを用いて非常に簡単に実行することができる．たとえば，(12)式の系をフィードバック線形化した後の線形系(11)はリアプノフ関数$\bar{x}^{\mathrm{T}}P\,\bar{x}$を用いて制御できるとしよう．すなわち線形のリアプノフ方程式

$$P(A+BK)+(A+BK)^{\mathrm{T}}P=-Q\prec0 \tag{20}$$

を満たす状態フィードバック$\bar{u}=K\bar{x}$が存在する．このとき，もとの系(3)の制御リアプノフ関数の一つは

$$V(x):=\bar{x}^{\mathrm{T}}P\,\bar{x}=\Phi(x)^{\mathrm{T}}P\,\Phi(x)$$

で与えられるため，このVを(18)式に代入することで逆最適補償器が得られる．いま，線形化された(11)式は2次の積分器であるので，リアプノフ方程式(20)におけるK, Qを

$$K=(\,-1,\,-2\,),\quad Q=\begin{pmatrix}1&0\\0&1\end{pmatrix}$$

と選ぶことで，リアプノフ方程式(20)を満たすPを

$$P=\frac{1}{2}\begin{pmatrix}3&-1\\-1&1\end{pmatrix}$$

のように求めることができる[1]．この例ではL_gVがスカラであるので，上記のフィードバックを求める手順はMaple上でつぎのように書ける．

[1] リアプノフ方程式はMATLAB, Mapleいずれを用いても簡単に解くことができる．

入力

```
P := [[3,-1],[-1,1]]/2:
V := simplify(evalm(Phi &* P &* Phi));
LgV := simplify(MyL(V,g,xx,1)):
LfV := simplify(MyL(V,f,xx,1)):
u = -(LfV + sqrt(LfV^2+LgV^4))/LgV;
```

出力

$$\frac{3}{2}x1^2 - x1\tan(x2) + \frac{1}{2}\tan(x2)^2 - \frac{(-(-3x1+\tan(x2))\tan(x2)+\sqrt{\cdots})\cos(x2)^4}{-x1\cos(x2)+\sin(x2)}$$

ここではリアプノフ関数 $V(x)$ とフィードバック入力 $u(x)$ を計算したが，結果が複雑になるため一部「…」と表記して省略した．また，どのような評価関数を最適化した補償器になっているかは，Maple で (19) 式を計算することで確認できる．

6. おわりに

本稿では Maple を用いた非線形制御系の解析・設計手法を解説した．非線形制御に必要な Maple の機能である微分積分の使い方を，実際の制御系の解析・設計問題を交えながら説明した．制御系の解析問題は1階1次の偏微分方程式に，設計問題は1階2次の偏微分方程式になることが多く，それらの典型的な解法のアルゴリズムを示した．ここで扱ったもの以外にもさまざまな非線形制御手法が提案されているが，本質的に同様の使い方で Maple などの数式処理ソフトウェアによって実装できるものが多い．非線形制御を学ぶための入門用の教科書としては，[5,3] などがあるが，いずれも概論的な内容である．洋書であれば，非線形制御一般 [18]，フィードバック線形化 [2] などが有名である．この記事をきっかけに非線形制御手法にふれていただければ幸いである．

(2011年2月10日受付)

参考文献

[1] 藤本：メカニカルシステムの非線形制御における未解決問題；計測と制御，Vol. 42, No. 2, pp. 100–106 (2003)

[2] A. Isidori: *Nonlinear Control Systems*, Springer-Verlag, third edition (1995)

[3] 島，石動，山下，渡邉，川村，横道：非線形システム制御論，コロナ社 (1997)

[4] P. A. Parrilo : *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, PhD thesis, California Institute of Technology, Pasadena, CA, USA (2000)

[5] 石島，島，石動，山下，三平，渡辺：非線形システム論，計測自動制御学会 (1993)

[6] 吉川，井村：現代制御論，昭晃堂 (1994)

[7] R. W. Beard, G. N. Saridis and J. T. Wen : Approximate solutions to the time-invariant Hamilton-Jacobi-Bellman equation; *Journal of Optimization Theory and Applications*, Vol. 96, pp. 589–626 (1998)

[8] D. L. Lukes : Optimal regulation of nonlinear dynamical systems; *SIAM J. Control*, Vol. 7, pp. 75–100 (1969)

[9] 水野，藤本：Chebyshev多項式を用いたHamilton-Jacobi方程式の近似解法；計測自動制御学会論文集，Vol. 44, No. 2, pp. 113–118 (2008)

[10] G. N. Saridis and C.-S. G. Lee : An approximation theory of optimal control for trainable manipulators; *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, pp. 152–159 (1979)

[11] R. Abgrall : Numerical discretization of first order Hamilton-Jacobi equation on triangular meshes; *Comm. Pure. Appl. Math*, Vol. 49, pp. 1339–1373 (1996)

[12] P. Laosuwan, M. Sampei, M. Koga and E. Shimizu: A numerical computational approach of Hamilton-Jabobi-Isaacs equation in nonlinear H-infinity control problems; 第25回制御理論シンポジウム，pp. 335–340 (1996)

[13] 大作，三平，清水，富田：非線形 H_∞ 制御によるセミアクティブサスペンション；計測と制御，Vol. 39, No. 2, pp. 126–129 (2000)

[14] A. J. van der Schaft : *L_2-Gain and Passivity Techniques in Nonlinear Control*, Springer-Verlag (2000)

[15] N. Sakamoto and A. J. van der Schaft : Analytical approximation methods for the stabilizing solution of the Hamilton-Jacobi equation; *IEEE Trans. Autom. Contr.*, Vol. 53, No. 10, pp. 2335–2350 (2008)

[16] R. A. Freeman and P. V. Kokotovic : *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*, Birkhäuser (1996)

[17] E. D. Sontag : *Mathematical Control Theory*, Springer, second edition (1998)

[18] H. K. Khalil : *Nonlinear Systems*, Macmillan Publishing Company, third edition (1996)

著者略歴

藤本 健治 (正会員)

1996年京都大学大学院工学研究科修士課程応用システム科学専攻修了，1997年同大学院博士後期課程を中途退学．1997年京都大学大学院工学研究科助手などを経て，2007年より名古屋大学大学院工学研究科准教授．1999年オーストラリア国立大学客員研究員，1999-2000年および2002年デルフト工科大学客員研究員．非線形制御の研究に従事．博士（情報学）．2000，2009年計測自動制御学会論文賞，2003年計測自動制御学会制御部門大会賞，2005年 The IFAC Congress Young Author Prize，2007年計測自動制御学会制御部門パイオニア賞を受賞．日本機械学会，日本鉄鋼協会，IEEE の会員．