数式処理ソフトを用いた制御系設計支援システムの開発

Matrix Fraction Description Approach CAD System for Control System Analysis and Design

日比野　伸介　(Shinsuke Hibino)
指導教官：　山田　実 講師　(Dr. Minoru Yamada)

In this study, matrix fraction description approach CAD system for control system analysis and design, named Symbolic Control Toolbox (SCT), was developed to offer the symbolic/numeric manipulation so-called a hybrid CAD system. SCT provides symbolic CAD facilities for control system analysis and design by using Symbolic Math Toolbox which allows the users to access Maple Kernel. Using SCT together with other existing MATLAB Toolboxes, the hybrid CAD environment can be easily obtained. SCT consists of three function groups which are algebraic manipulation function group, solving equation function group and control system analysis and design function group based on matrix fraction description approach. Some examples are given to verify the effectiveness of this CAD system.

1.　はじめに

現在、制御理論は古典制御理論と同様、周波数応答に注目した制御系設計理論（代数的設計法）が再び活発に研究されだしてきた[1]。これは、古典的な設計法に理論的な基礎づけをしようという問題意識が起こったことによる。代数的設計法において、伝達関数行列は既約な多項式や安定有理関数行列の比として表され（既約行列分解表現)、様々な制御問題は代数方程式の可解性に帰着される[2]。

一方、制御理論の発展に伴い、ますます高度化、複雑化する制御系の設計にとって計算機援用設計システム（CAD）はもはや必要不可欠なものとなっており、様々なところで研究開発されてきた。しかしそれらのほとんどは数値計算や図形処理には威力を発揮したが、パラメータや上述の代数方程式など変数 s や z といった記号をそのまま扱う数式処理には問題があった[3]。また、数式処理を行った結果に対して数値処理や図形処理を必要とする場合などが多くあり、このため数式・数値処理機能が統合されたいわゆるハイブリッド制御系 CAD が必要である[4]。

MATLAB は、高度な数値演算と多彩なグラフィック機能を備えたソフトであり、数値制御 CAD として世界中で幅広く利用されている。また、Symbolic Math Toolbox（SMT）により数式処理ができる機能を持つ。そこで本研究では、MATLAB の数式・数値統合制御系 CAD 環境を提供するために SMT を利用した行列既約分解法による制御系設計支援システム Symbolic Control Toolbox（SCT）の開発を行う。

2.　Symbolic Control Toolbox

本研究では、SMT の関数を主に利用して数式処理による制御系解析･設計 Toolbox である Symbolic Control Toolbox (SCT) の開発を行った。SCT は、行列既約分解法に基づいて実現した　制御系解析・設計関数群とそれに必要な数式処理を実現した関数群（　代数方程式求解関数群、　基本代数操作関数群）の 3 つの関数郡からなる Toolbox である。SCT の関数は MATLAB の関数や他の Toolbox などとの相互利用が容易に行える。また、SCT の関数は多くの MATLAB の関数や他の Toolbox の関数と同様に M-file 形式 (テキストファイル形式）で書かれており、ファイルの容量が少なくてすみ、コンパイルといった作業が必要ないため、ハードウェアが違う環境においても問題なく動作する[5]。

2.1　基本代数操作関数群

この関数群は、制御系解析・設計に必要な基本的な代数操作を行う (**Table 1**)。これは、後に述べる代数方程式求解関数群を開発する際に必要となる、行列の行（列）標準形[6] を求める関数である。行（列）標準形を求めるには最小公倍因子を

求める関数が必要であり、最小公倍因子は、ユークリッドの互除法にて最大公約因子を求めることで与えられる。また、この関数群には基本行（列）変換や、関数 maple を利用して、スミスホームを求める関数も含まれる。

Table 1 基本代数操作関数群

関 数 名	機 能
erop	基本行変換、任意の行をα倍する。
ecop	基本列変換、任意の列をα倍する。
eroad	基本行変換、I 行に J 行をα倍して加える。
ecoad	基本列変換、I 列に J 列をα倍して加える。
eroch	基本行変換、I 行と J 行をいれかえる。
ecoch	基本列変換、I 列と J 列をいれかえる。
gcds	ユークリッドの互除法にて最大公約因子を求める。
denoms	有理関数行列から分母部分を求める。
numers	有理関数行列から分子部分を求める。
gcdr	各行ごとに最大公約因子を求める。
gcdc	各列ごとに最大公約因子を求める。
lcmr	各行ごとに最小公倍因子を求める。
lcmc	各列ごとに最小公倍因子を求める。
mcm	行列の最大公約因子を求める。
lcm	行列の最小公倍因子を求める。
sym2polys	多項式行列からシンボリックな係数ベクトルの抽出。
ref	行標準形 (エルミート行正準形) を求める。
cef	列標準形を求める。
smith	多項式環でスミスホームを求める。
psmith	プロパー安定有理関数環でスミスホームを求める。
monic	多項式行列からモニック多項式を求める。

2.2 代数方程式求解関数群

この関数群は、主に基本代数操作関数群を利用し、既約行列分解や代数方程式の解を求めるものである (**Table 2**)。代数的設計法を行う際、制御系設計問題はユニラテラル方程式やバイラテラル方程式といった代数方程式の可解性に帰着される。既約行列分解表現はこの代数方程式を解く際に重要な表現形式である。つまり、この関数群は本研究の心臓部ともいえる。

例えば左ユニラテラル方程式を解くとは、与えられた多項式行列 (プロパー安定有理関数行列) A, B, C に対して、次式を満たす特殊解 X_0, Y_0 を求めることである。

$$AX_0 + BY_0 = C \tag{1}$$

この方程式の一般解は、

$$X = X_0 + B_l T, \quad Y = Y_0 - A_l T \tag{2}$$

で表され、$AB_l = BA_l$ となる B_l, A_l も求める。ここで T は任意行列である。右ユニラテラル方程式についても同様である。

これらの代数方程式を解くには基本代数操作関数群のスミスホームを求める関数や代数方程式求解関数群の関数 axeqb などが必要となる。

Table 2 代数方程式求解関数群

関 数 名		機 能
多項式環	プロパー安定有理関数環	
axeqb	paxeqb	$AX = B$ の X を求める。
gcld	pgcld	最大共通左因子を求める。
gcrd	pgcrd	最大共通右因子を求める。
lcf	plcf	左既約行列分解表現を求める。
rcf	prcf	右既約行列分解表現を求める。
doubly	pdoubly	重既約分解表現を求める。
lunil	plunil	左ユニラテラル方程式を解く。
runil	prunil	右ユニラテラル方程式を解く。
bil-w	pbil-w	バイラテラル方程式を解く (Wolovich)。

2.3 制御系解析・設計関数群

この関数群は、行列既約分解法に基づいて実現された制御系解析･設計関数群であり、上述の２つの関数群を主に利用した関数群である (**Table 3**)。関数 stabilizer は多項式環上で代数方程式求解関数群の重既約分解表現を求めることで安定なコントローラを設計している [7]。同様に関数 pstabilizer はプロパー安定有理関数環上で求めている。

Table 3 制御系解析・設計関数群

関 数 名	機 能
＜モデルの変換＞	
D2P-2D	遅れ (進み) 系の 2D 有理関数を進み (遅れ) 系に変換する。
filts	伝達関数行列を連続時間系の LTI モデルに変換する。
filtd	伝達関数行列を離散時間系の LTI モデルに変換する。
＜コントローラの設計＞	
stabilizer	多項式環で、安定化コントローラを求める。
pstabilizre	プロパー安定有理関数環で安定化コントローラを求める。
emm	モデルマッチング問題を解く。

3. 実行例

本章では、本 Toolbox において開発した左既約分解表現を求める関数 `lcf` の実行例と安定化コントローラを求める関数 `pstabilizer` を適用して制御系を設計し、設計した制御系を Simulink を用いてシミュレーションを行った例を示す。

3.1 lcf

目的 与えられた有理関数行列 $W \in R(s)^{m\times n}$ に対して左既約分解表現 $W = D^{-1}N$ の $D \in R[s]^{m\times m}$, $N \in R[s]^{m\times n}$ を求める。このとき、派生する $P \in R[s]^{m\times m}$, $Q \in R[s]^{n\times m}$ も求める。ここで $R[s]^{m\times n}$ は実係数多項式からなる $m\times n$ 行列の集合である。

$$W = D^{-1}N \tag{3}$$

$$DP + NQ = I \tag{4}$$

入力 W

出力 N, D, P, Q

例題として

$$W = \begin{bmatrix} \dfrac{1}{s} & \dfrac{s+2}{(s+1)^2} \\ \dfrac{1}{s+1} & \dfrac{s+1}{(s+2)^2} \end{bmatrix} \tag{5}$$

の左既約分解表現 N, D を求める。ここで、

```
[D,N,P,Q] = lcf(W)
```

と入力すると、D, N, P, Q が求まるが、

```
[D,N] = lcf(W)
```

と入力すると、D, N のみ求まる。**Fig. 1** に結果を示す。1 行目で W を入力し、2 行目で関数 `lcf` を用いて左既約分解表現 D, N を求めている。5 行目で $D^{-1}N$ を計算すると W となっており、3 式が成り立っていることがわかる。

3.2 pstabilizer

目的 プラントに対して、**Fig. ??**のフィードバックシステムを安定化するコントローラのクラスを求める [7]。

入力 プラント P

出力 フィードバックシステムを安定化するコントローラのクラス $\Omega(P)$ を表す $X_{PR}, Y_{PR}, X_{PL}, Y_{PL}, D_{PR}, N_{PR}, D_{PL}, N_{PL}$

ここで、

$$P = N_{PR}D_{PR}^{-1} = D_{PL}^{-1}N_{PL} \tag{6}$$

```
》
W=[1/s (s+2)/(s+1)^2;1/(s+1) (s+1)/(s+2)^2];
》[D,N]=lcf(W);
》pretty(D)

[       3     2                              ]
[      s + 2 s + s,                0          ]
[                                             ]
[    4     3     2         3      2           ]
[-2 s - 7 s - 8 s - 3 s, s  + 5 s  + 8 s + 4]
》pretty(N)

[      2                       2              ]
[     s  + 2 s + 1,           s  + 2 s        ]
[                                             ]
[    3     2               3      2           ]
[-2 s - 6 s - 4 s + 1,-2 s  - 6 s  - 4 s + 1]
》simple(inv(D)*N)
ans =

[           1/s, (s+2)/(s+1)^2]
[       1/(s+1), (s+1)/(s+2)^2]
```

Fig. 1 関数 `lcf` の実行例

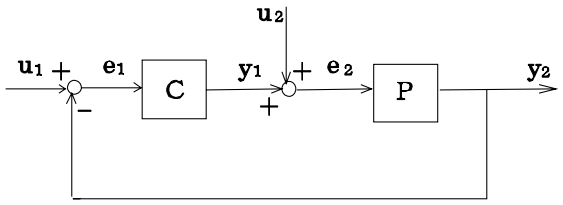

Fig. 2 フィードバックシステム

$$X_{PR}N_{PR} + Y_{PR}D_{PR} = I \tag{7}$$

$$N_{PL}X_{PL} + D_{PL}Y_{PL} = I \tag{8}$$

$$\Omega(P) = \{C = (X_{PL} + D_{PR}Q)(Y_{PL} - N_{PR}Q)^{-1}, \det(Y_{PL} - N_{PR}Q) \neq 0\} \tag{9}$$

$$= \{C = (Y_{PR} - RN_{PL})^{-1}(X_{PR} + RD_{PL}), \det(Y_{PR} - RN_{PL}) \neq 0\} \tag{10}$$

Q, R は任意のプロパー安定有理関数行列である。

例題としてプラントを

$$P = \begin{bmatrix} \dfrac{1}{s} & \dfrac{s+2}{s^2-1} \end{bmatrix} \tag{11}$$

としたときのコントローラのクラスを求める。**Fig. 3** に結果を示す。1 行目で P を入力し、2 行目で関数 `pstabilizer` を用いて、コントローラのクラスを表す際に必要なパラメータを求めている。

関数 pstabilizer を実行して得た結果を関数 filts を用いて連続時間系の LTI(Liner Time Invariant) モデルに変換し、Simulink を用いて各ブロック (コントローラ及びプラントのパラメータ、入力、応答の表示) を設定する。そして数値計算によりシステムの応答を求めると **Fig. 4** となり、安定な応答が得られていることがわかる。このように Symbolic Control Toolbox で設計した制御系に対し、他の Toolbox(ここでは、Simulink) で処理が容易に行える。

4. おわりに

本研究では、Symbolic Math Toolbox を利用して多項式環・プロパー安定有理多項式環上の双方で行列既約分解法による Symbolic Control Toolbox の開発を行った。これにより MATLAB の数式・数値処理統合制御系 CAD 環境を提供できた。ここで関数 stabilizer などの制御系解析・設計関数を 1 つ開発するにも、基本代数操作関数や代数方程式求解関数といったさまざまな関数を用意しておく必要があり、安定化コントローラなどの制御問題を解くには、大変な時間と労力が必要となることがわかった。

本研究では代数演算や基本制御問題に重点をおいてたため、制御系の解析・設計に関する関数がまだ十分充実しているとはいいがたい。例えば、任意パラメータの設定や 2D 制御系のための関数などについても開発する余地がある。また、数式処理を必要とする問題が他にも数多く存在していると考えられるため、今後それらを整理し、関数の充実をはかる必要がある。

参 考 文 献

(1) 前田 肇, 杉江 俊治, アバンスト制御のためのシステム制御理論, (1990), 朝倉書店.
(2) 日野 秀樹, 行列既約分解法による制御系設計支援システム, (1996), 豊橋技術科学大学 修士論文.
(3) 斎藤, 阿部, 数式処理とそのシステム制御工学への応用, **28**-2,(1989),168-172, 計測と制御.
(4) 高橋, 藤田, 斎藤, 阿部, 数式・数値処理による制御系 CAD, **111**-7,(1991),292-298, 電気学会論文誌.
(5) Xu Li, Minoru Yamada, Osami Saito, Development of n-D Control System Toolbox for Use with MATLAB, (1999),CCA/CACSD'99.
(6) 伊藤 正美, 木村 英紀, 細江 繁幸, 線形制御系の設計理論, (1978),(社) 計測自動制御学会.
(7) Vladmir Kucera, Discrete Linear Control, (1979),A Wiley-Interscience Publication.


```
》P=[1/s (s+2)/(s^2-1)];
》
[Xpr,Ypr,Xpl,Ypl,Dpr,Npr,Dpl,Npl]=pstabilizer(P)

Xpr =
[          1/4]
[ 4/3*s/(s+1)]

Ypr =
[         1/4,                              0]
[ -4/3/(s+1), 1/6*(-3+4*s+3*s^2)/(s+1)^2]

Xpl =
[ 1/6*(3+s)*(1+3*s)/(s+1)^2]
[         -2/3*(1+3*s)/(s+1)]

Ypl =
-1/2

Dpr =
[           4*s/(s+1), -2*s*(s+2)/(s+1)^3]
[                   0,       2*(s-1)/(s+1)]

Npr =
[            4/(s+1), 2*s*(s+2)/(s+1)^3]

Dpl =
-2*s*(s-1)/(s+1)^2

Npl =
[   -2*(s-1)/(s+1)^2, -2*s*(s+2)/(s+1)^3]
```

Fig. 3 関数 pstabilizer の実行例

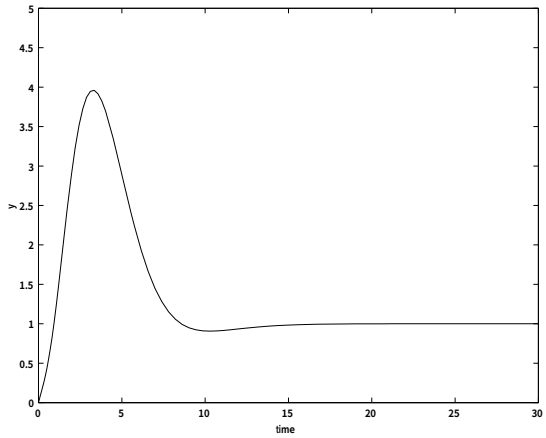

Fig. 4 フィードバックシステムの応答