MATX を用いた制御系設計を支援する計算機環境

A Computer Environment for Design of Control Systems using MATX

○正　古賀雅伸 (東工大)　　清田洋光 (東工大)
中山尚登 (東工大)　　三平満司 (東工大)

Tokyo Institute of Technology
2–12–1 Oh-okayama, Meguro-ku, Tokyo 152, JAPAN

This paper introduces a computer environment for design of control systems, which was developed using a programming language MATX, through examples of analysis, design, and simulation of a simple control system, i.e., an inverted pendulum. The real-time control facility of MATX is demonstrated by showing an experiment of the inverted pendulum.

Key Words: CACSD, Simulation, Real-time control, programming language

1　はじめに

著者らは，科学や工学に必要な数値および数式計算をサポートする記述性に優れたプログラミング言語 MATX を開発し，制御系の解析・設計・シミュレーションに使用してきた [1]。

MATX の処理系にはインタプリタ (matx) とコンパイラ (matc) がある。ライブラリやユーティリティなどを含めたパッケージ全体を MATX と書き，インタプリタ (matx) と区別する。コンパイラは MATX のソースファイル (mm ファイル) から C 言語のファイルを生成する。その C 言語のファイルを C コンパイラを使ってコンパイルし，ユーザの C 言語のプログラムとリンクして実行プログラムを作ることができる。生成された実行ファイルはインタプリタより，実効速度・メモリの使用効率の点で優れている [2]。

近年，CAD により設計されたコントローラをディジタルコントローラで実現する作業を自動化する試みが行われている。これは，制御系の設計から実現までに要する時間の短縮と CAD により設計されたコントローラと実現されたコントローラの一貫性を保つことを目指している。シミュレーション環境で作成したコントローラを簡単にリアルタイム制御環境に移植できることが望まれている。MATX にリアルタイム処理機能を付加した RTMATX(リアルタイム MATX) を用いれば，PC/AT 互換機や PC9801 の DOS 上で動作するリアルタイム処理プログラムを簡単に作成できる。プログラムのほとんど全てを MATX 言語で記述できるだけでなく，シミュレーション用に作成した関数をほとんどそのままリアルタイム制御に用いることができるので，コントローラの実現が非常に容易になる [3]。

本論文では，簡単な制御対象 (倒立振子) の制御系の解析・設計・シミュレーションを例題として，科学技術計算プログラミング言語 MATX を用いて開発された制御系 CAD を紹介する。また，MATX のリアルタイム制御機能により倒立振子の実機を用いた実験方法を示す。

2　MATX の紹介

2.1　MATX の特徴

MATX の特徴として，

1. 行列，多項式，有理関数，多項式行列，有理関数行列などのクラスが標準で提供されており，計算アルゴリズムを簡潔に表現でき，記述性が高い。
2. 言語仕様が C 言語の仕様と非常に似ているので覚えやすい。たとえば，制御文 (`if`, `else`, `for`, `while`, `do`, `switch`) は全く同じである。
3. インタプリタだけなくコンパイラも提供されているので目的に応じて効率良く作業を進めることができる。
4. C 言語のプログラムをリンクできるので今までに C 言語で書かれたソフトウェア資産を活用できる。
5. EISPACK や LINPACK と同じアルゴリズムの固有値計算，LU 分解，QR 分解，QZ 分解，特異値分解や RKF45，FFT 等の有用関数が提供されている。
6. 行列の入力編集を簡単に行うための行列エディタが提供されている。
7. 常微分方程式で記述される非線形システム (線形システムを含む) の連続系およびサンプル値系 (連続 + 離散) のシミュレ-ションを簡単に実行できる。
8. RTMATX を使えば，実時間制御用プログラムを直接記述できる。

などがあげられる。

2.2 MATX の構造

MATX の構造を図 1に示す。 MATX には，3 つユーザ・インターフェース・ツール (インタプリタ matx，コンパイラ matc，行列エディタ mated) がある。これらのツールはクラスライブラリ **MaTX-Lib** を用いて作成されている。MaTX-Lib は，複素数クラス，多項式クラス，有理多項式クラス，行列クラス，リストクラスの C 言語のクラス・ライブラリの集合である。

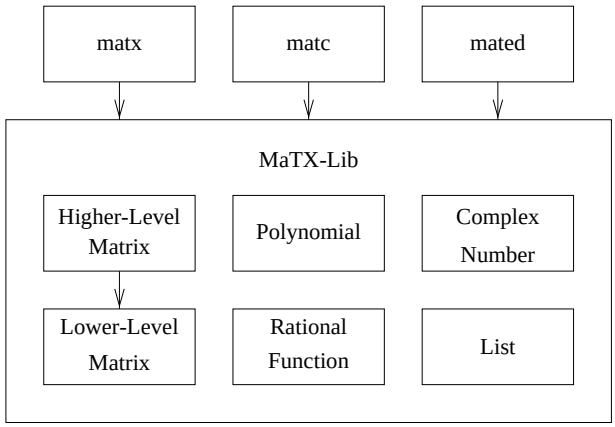

図 1: MATX の構造

2.3 プログラム作成手順

図 2は，MATX におけるプログラムの作成手順を示している。プログラムを作成するには，仕様にしたがってアルゴリズムをエディタ (vi, emacs) を使って mm ファイル (test.mm) の中に関数として実現する。 インタープリタ

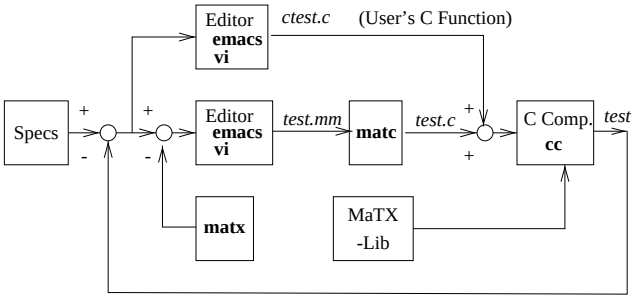

図 2: プログラム作成手順

(matx) を使えば，ファイル単位，関数単位，または行単位に会話的に mm ファイルをテストすることができる。動作確認の終った mm ファイルは， matc によって C 言語に変換される。 C 言語のプログラム (ctest.c) を matc の出力コードにリンクすれば， MATX の関数から簡単に C 言語の関数を呼び出すことができる。全ての C 言語のファイルが完成した後は， C コンパイラによってコンパイルし，実行ライブラリ (MaTX-Lib) とリンクして実行プログラム (test) を作成する。最後に，実行ファイルを実行して全ての仕様が満たされているかチェックする。

3 倒立振子の安定化制御

図 3に示す倒立振子の安定化制御について考える。まず， MATX を用いた制御系設計，ディジタル制御器によるサンプル値系のシミュレーションの方法を示す。次に，RTMATX を用いたリアルタイム制御の方法を示す。

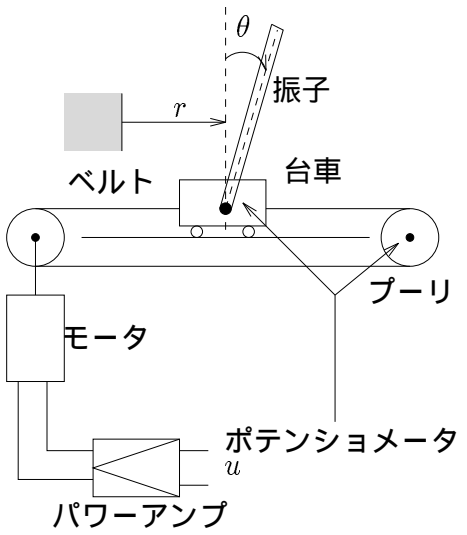

図 3: 倒立振子制御系の実験装置

3.1 制御系の設計

状態変数として台車の変位 r，振子の角度 θ，台車の速度 $\dot{r}$，振子の角速度 $\dot{\theta}$ をとり，状態ベクトルを

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T = \begin{bmatrix} r & \theta & \dot{r} & \dot{\theta} \end{bmatrix}^T$$

とし，r と θ を出力変数とすると，倒立振子の非線形状態方程式と出力方程式が得られる。この方程式を平衡点近傍 $(x = 0, u = 0)$ で線形近似すると倒立振子の線形モデル

$$\dot{x} = Ax + Bu \quad (1)$$

$$y = Cx \quad (2)$$

が得られる。線形モデルを設定するプログラムを図 4に示す。線形モデルに対して 2 次形式評価関数

$$J = \int_0^\infty x^T Q x + u^T R u \, dt, \quad Q^T = Q \geq 0, R^T = R > 0$$

を最小にする最適状態フィードバック

$$u = -Fx = -R^{-1} B^T P x$$

を用いて制御を行う。ただし， P はリカッティ方程式

$$A^T P + PA - PBR^{-1}B^T P + Q = O$$

```
Func List linear_plant()
{
    Matrix A, B, C;

    A = [[ 0  0   1   0  ]
         [ 0  0   0   1  ]
         [ 0 a32 a33 a34 ]
         [ 0 a42 a43 a44 ]];

    B = [[  0 ]
         [  0 ]
         [ b3 ]
         [ b4 ]];

    C = [[ c1  0 0 0 ]
         [  0 c2 0 0 ]];

    return {A, B, C};
}
```

図 4: 倒立振子の線形モデル

の正定対称解である。直接測定できない状態を推定するためゴピナスの設計法で最小次元オブザーバ (の極は Po)

$$\begin{aligned} \dot{z} &= A_h z + [B_h\ J_h] \begin{bmatrix} y \\ u \end{bmatrix} \\ x_h &= C_h z + D_h y \end{aligned}$$

を設計する。ディジタル制御を行うためオブザーバを (サンプリング時間 T で) 離散化して以下のように求める。

$$\begin{aligned} z[k+1] &= A_{hd}\, z[k] + [B_{hd}\ J_{hd}] \begin{bmatrix} y[k] \\ u[k] \end{bmatrix} \\ x_h[k] &= C_h z[k] + D_h y[k] \end{aligned}$$

制御系の設計を実行するプログラムを図 5 に示す。

```
Func List design(A, B, C, Q, R, Po, T)
  Matrix A, B, C, Q, R, Po;
  Real T;
{
  Matrix P, F;
  Matrix Ah, Bh, Ch, Dh, Jh, Ahd, Bhd;

  P = Riccati(A, B, Q, R);
  F = R~ * B' * P;

  {Ah,Bh,Ch,Dh,Jh} = Gopinath(A,B,C,Po);
  Bh = [Bh Jh];

  {Ahd, Bhd} = Discretize(Ah, Bh, T);
  return {F, Ahd, Bhd, Ch, Dh};
}
```

図 5: 制御系の設計

3.2 シミュレーション

前節で設計したディジタルコントローラで連続時間の制御対象である倒立振子を制御するサンプル値系のシミュレーションを行う。そのためには，システムの状態方程式と入出力方程式をそれぞれ関数として記述する。具体的には，制御対象の状態方程式 (非線形常微分方程式) を diff_eqs() に，オブザーバの状態方程式 (差分方程式) と状態フィードバック則を link_eqs() に記述する (図 6参照)。組み込み関数 OdeHybrid() を呼び出すことでサンプ

```
Func void diff_eqs(dx, t, x, u)
  Matrix dx, x, u;
  Real t;
{
  Real c2, s2, dm;

  c2 = cos(x(2));  s2 = sin(x(2));
  dm = 1 + alpha*s2^2;

  dx = [[            x(3)                ]
        [            x(4)                ]
        [(a35*s2*x(4)^2 + a34*c2*x(4)
          + a33*x(3) + a32*s2*c2)/dm  ]
        [(a45*s2*c2*x(4)^2 + a44*x(4)
          + a42*s2 + a43*c2*x(3))/dm  ]]
          [[      0         ]
        +  [      0         ]
           [   b3*u(1)/dm  ]
           [ b4*c2*u(1)/dm ]];
}

Func void link_eqs(u, t, x)
  Matrix u, x;
  Real t;
{
 Matrix y, xh;

 y  = C*x;                 // 出力方程式
 xh = Ch*z + Dh*y;         // 状態推定
 u  = - F*xh;              // 状態フィードバック
 z  = Ah*z + Bh*y + Jh*u; // オブザーバの更新
}
```

図 6: システムの状態方程式と入出力方程式

ル値系のシミュレーションを実行できる (図 7参照)。ただし，dt はサンプリング周期，x20 はオブザーバの初期状態である。関数 OdeHybrid() は link_eqs() をサンプル

```
Func void main()
{
  Real t0, tf, ts, dt;
  Matrix x10, x20, X0;
  Array Ts, Xs, UYs;

  x2 = x20;           // オブザーバの初期状態
  X0 = x10;           // 制御対象の初期状態
  {Ts,Xs,UYs} = OdeHybrid(t0, tf, dt, X0,
                    diff_eqs, link_eqs, ts);
}
```

図 7: シミュレーションのメイン関数

点毎に呼び出し，サンプル点間で繰り返し diff_eqs() を呼び出しながら 4 次のルンゲクッタ法を用いてきざみ ts 毎にシミュレーション計算を行い，状態の時系列 Xs，入出

力の時系列 UYs，計算を行った時刻の系列 Ts を返す。

状態方程式と入出力方程式を 2 つの関数として別々に記述するので，サンプル値系のシミュレーションプログラムに最小限の変更を加えるだけで連続系のシミュレーションを実行できることを付け加えておく。

3.3 リアルタイム制御

RTMATX を使用すれば，サンプル値系のシミュレーションプログラムに最小限の変更を加えたコードからリアルタイム制御プログラムを生成できる。加えるべき変更とは制御対象の出力の取り込み処理と制御入力の制御対象への出力処理である。

ここでは倒立振子のサンプル値系のシミュレーションプログラムに最小限の変更を加えたコードからリアルタイム制御プログラムを生成する方法を示す。まず，図

```
Func void on_task()
{
 Matrix u, y, xh;

 y  = sensor();          // センサから入力
 xh = Chd*z + Dhd*y;     // オブザーバの推定
 u  = - F*xh;            // 制御則計算
 actuator(u);            // 制御出力
 z = Ahd*z+Bhd*y+Jhd*u; // オブザーバの更新
}
```

図 8: リアルタイム制御 (リアルタイム処理関数)

8に示すように link_eqs() を変更し on_task() とする。sensor() は制御対象の出力をセンサーで取り込む関数，actuator() は制御入力をアクチュエータへ出力する関数であり，コンピュータの入出力ボードの仕様にしたがって作成する。制御則を変更してもこれらの関数を変更する必要がないので，入出力ボードから先のハードウェアの仕様をこれらの関数に隠蔽できる。また，RTMATX が提供するコンピュータの入出力ポートを直接アクセスする関数を用いれば，sensor() と actuator() も RTMATX の関数として直接記述できる。

次に，図 9に示すリアルタイム制御のためのメイン関数の雛型を必要に応じて変更する。メイン関数では，コントローラなどのパラメータの設定，変数の初期化，サンプリング周期の設定，サンプル点毎に呼び出す関数 (on_task()) の設定をした後，実験装置の初期化を行う。rtStart() が呼ばれると，on_task() がサンプル点毎に呼び出される。off_task_loop() では，変数の表示やユーザからのコマンドの入力を受け付ける。

最後に，RTMATX のコンパイラを用いて on_task() や main() などの関数を記述したソースファイルを C 言語のファイルに変換し，リアルタイム割り込みルーチンを含むライブラリとリンクする。

入出力ボードの仕様に従い sensor() と actuator() をあらかじめ作成しておき，main() の雛型を用いるとすれば，シミュレーションプログラムに加えるべき変更は，

```
Func void main()
{
  para_init();              // パラメータ設定
  var_init();               // 変数の初期化
  rtSetClock(stime);        // サンプリング時間
  rtSetTask(on_task);       // リアルタイム処理
  rtSetBreak(break_task); // Ctrl-C の処理
  machine_ready();          // 実験装置の準備

  rtStart();                // リアルタイム開始
  off_task_loop();          // 画面表示など
  rtStop();                 // リアルタイム停止

  machine_stop();           // 実験装置の停止
}
```

図 9: リアルタイム制御 (メイン関数)

link_eqs() に sensor() と actuator() の呼び出しを追加するだけとなる。

このようにして生成されたリアルタイム制御プログラムはシミュレーションプログラムと同じライブラリをリンクしているので，シミュレーションプログラムとデータを共有でき，同一のユーザインタフェースをもつ。また，全ての制御系解析・設計ルーチンを組み込めるので，プログラムをコンパイルし直さずにコントローラの修正と実験を繰り返し実行できる。

4 おわりに

MATX は C 言語で書かれており，UNIX が動くワークステーション (Sun, HP, DEC, MIPS, SONY)，PC-UNIX(BSD/OS, FreeBSD, Linux) や Windows 95, Windows NT が動くパソコンで動作する。GO32 などのメモリ・エクステンダを使えば，PC/AT 互換機や国産パソコン (PC9801) の DOS 上で使うこともできる。MATX はフリーソフトウェアであり，以下の場所から入手できる。

```
FTP: matx.mei.titech.ac.jp:/pub/MaTX
WWW: http://matx.mei.titech.ac.jp/
```

参 考 文 献

[1] 古賀雅伸, 鳥海洋章, 三平満司. オンラインパラメータ変更を考慮したリアルタイム制御系ＣＡＤ. 第３５回計測自動制御学会学術講演会, pp. 181–182, 1996.

[2] Masanobu Koga, Mitsuji Sampei, and Katsuhisa Furuta. A compiler of matlab to MaTX: Compiling and linking of m-files to an executable program. *Proc. of CACSD'94, Tucson, Arizona, U.S.A.*, pp. 137–142, 1994.

[3] Masanobu Koga, Hiroaki Toriumi, and Mitsuji Sampei. Real-time cad of control systems achieving cooperation of modeling and design of controllers. *Proc. of CACSD'96, Dearborn, Michigan, U.S.A.*, pp. 457–462, 1996.