講　座

ロボット工学ソフトウェア利用の手引 (第 3 回)
科学技術計算プログラミング言語 MATX

MATX - A High-Performance Programming Language for Scientific and Engineering Computation

古賀雅伸*　*東京工業大学
Masanobu Koga**　Tokyo Institute of Technology

1. はじめに

制御系の設計において，外乱やプラントパラメータの変化に対する閉ループ系のシミュレーションを行うことは重要であるので，制御系設計に計算機を使用することはシミュレーションから始まった．代表的なものに非線形システムのシミュレーションを行うために開発された Simnon [1] がある．残念ながら Simnon は行列演算を効率良く記述することができなかった．制御系の解析・設計におけるベクトルや行列演算を簡単に行いたいという要望から行列を基本データ構造とする言語 Matlab [2] [3] や Matrix_X [4] が生まれた． Matlab や Matrix_X は LINPACK や EISPACK 等の信頼性のある行列計算アルゴリズムを元に構築された行列演算言語であり，紙に書いた行列演算をほとんどそのままプログラムとして使用できるという記述性の良さから近年多く使用されるようになった．

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． MATX は行列の演算や成分の操作法について Matlab といくつかの特徴的性質を共有している．しかしながら， Matlab がデータの型を「区別しない」言語であるのに対し， MATX はデータの型を「区別する」言語である． Matlab の唯一のデータ型は行列であり，多項式などのデータの演算や操作には特別な演算子を使うか専用の関数を呼び出す．一方， MATX のデータ型には，整数，実数，複素数，文字列，多項式，有理多項式，行列，配列，リストがあり，多くの重要なデータの定義や操作を数学的な代数的表現で記述できる．

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．

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．多くの制御系 CAD は便利で使いやすいシミュレーション環境を提供しており，そこ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 R_TMATX(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．

本稿は，日本ロボット学会誌の特別付録 “ソフトウェアライブラリー (CD-ROM 版)” [5] に掲載された，科学技術計算プログラミング言語 MATX の使用方法を例題を通して述べる． 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, MIPS, NEWS)， 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 や EPSON) の DOS 上で使うこともできる．

2. MATX の概要

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．

原稿受付　1996 年 8 月 21 日
キーワード：制御系 CAD, 制御系解析・設計, 数値計算, 数式処理
*〒 152 東京都目黒区大岡山 2-12-1
*2–12–1 Oh-okayama, Meguro-ku, Tokyo 152

(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．

3. 倒立振子の安定化制御

図1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．

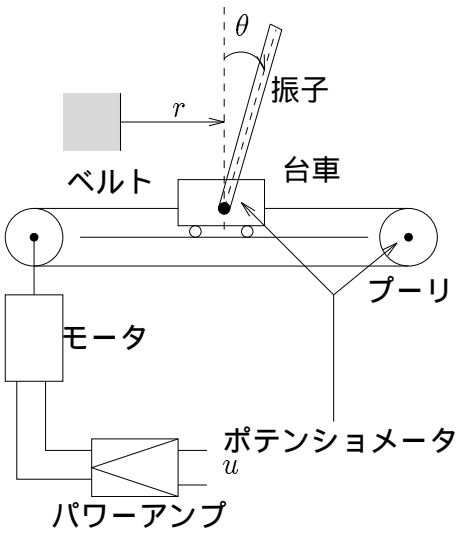

図1 倒立振子制御系の実験装置

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

$$\dot{x} = Ax + Bu \qquad (1)$$
$$y = Cx \qquad (2)$$

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

```
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2 倒立振子の線形モデル

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

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

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

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)

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

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．

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

制御系の設計を実行するプログラムを図3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 3　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 4参照)．組み込み関数 OdeHybrid() を呼び出すことでサンプル値系のシミュレーションを実行できる (図 5参照)．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

一般にディジタルコントローラによるリアルタイム制御では，コントローラはサンプル点毎に制御対象の出力を取り込み，制御入力の計算 (コントローラの出力の計算，コントローラの状態の更新) を実行し，制御対象に出力する．ところで，MATX によるサンプル値系のシミュレーションでは，制御対象の入出力方程式，コントローラの入出力方程式および状態方程式を記述した link_eqs() の計算がサンプル点毎に実行される．このようにサンプル値系のシミュレーションプログラムでサンプル点毎に実行される処理とリアルタイム制御プログラムでサンプル点毎に実行される処理がほとんど一致する．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．

```
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 4　システムの状態方程式と入出力方程式

```
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 5　シミュレーションのメイン関数

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 6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．

```
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 6　リアルタイム制御 (リアルタイム処理関数)

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．

```
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 7　リアルタイム制御 (メイン関数)

次に，図 7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．ただし，雛型のままでも (1 回の) 実験ができる．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，割り込みキーを押されたときに呼ばれる関数の (break_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，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次のような特徴をもつ．

- シミュレーションプログラムとデータを共有できる．
- シミュレーションプログラムと同一のユーザインタフェースをもつ．
- 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

倒立振子の安定化制御を例題として科学技術計算プログラミング言語 MATX を紹介した．これまで主に制御系の解析・設計，シミュレーションに応用されてきたが，倒立振子やロボットの実時間制御プログラムの記述などにも応用が始まっている．今後，フリーソフトウェアである MATX が大学や研究所だけでなく数値・数式計算を必要とする高校，高専などで役立てられることを期待している．

最後に MATX の最新版の入手先を示す．

FTP:
 matx.mei.titech.ac.jp:/pub/MaTX
WWW:
 http://www-sampei.mei.titech.ac.jp/~koga/MaTX.html

参考文献

[1] K. J. Astrom. Computer aided modeling, analysis and design of control systems – a perspective. *Control Systems Magazine*, pp. 4–16, 1983.

[2] C. Moler, J. Littel, and S. Bangert. *PC-MATLAB – User's Guide*. N. Main St., Sherborn, MA 01770, USA, 3.1 edition, 1987.

[3] Cleve Moler. *MATLAB – User's Guide*. Alberquerque, USA, 1980.

[4] Robert Walker, Charles Gregory Jr., and Sunil Shah. MATRIXx: A data analysis, system identification, control design and simulation package. *Control Systems Magazine*, pp. 30–37, 1982.

[5] 日本ロボット学会学会誌編集委員会, 荒井裕彦, 小笠原司. ロボット工学ソフトウェアライブラリ cd-rom について. 日本ロボット学会誌, Vol. 14, No. 1, pp. 42–47, 1996.

古賀　雅伸 (Masanobu Koga)

1965 年 9 月 21 日生まれ．1993 年 3 月東京工業大学大学院博士後期課程修了 (制御工学専攻)．同年 4 月東京工業大学工学部助手，1994 年 6 月同大学大学院情報理工学研究科助手となり現在に至る．システム・制御理論の応用，ロボット工学，制御系 CAD などの研究に従事．計測自動制御学会，システム制御情報学会，IEEE の会員．博士 (工学)．

(日本ロボット学会正会員)