情報演習 II　応用演習　MATLAB コース

MATLAB，SIMULINK の基本操作

滑川　徹　　小林 泰秀

平成 15 年 6 月 13 日 ver.1.2

Contents

(注意) 本稿は MATLAB ver **5.3.1** に基づいて記述しています．情報処理センター (pine) の MATLAB のバージョンは **6.5** (03/06/01 現在) であり，GUI 機能が強化されたため，多少表示が異なりますが，基本的な計算機能は同じで，本稿を基に類推できると思います．

1 MATLABの基本操作

1.1 MATLAB とは

MATLAB は, 科学技術計算用の高性能言語です．主な特長および応用分野を以下に示します.

特長

1. 次元の設定を必要としない配列を基本データ要素とする対話型システム．これにより, (非対話型言語である）C や Fortran に比べプログラミングが容易.
2. 行列計算，可視化に優れ, さらに GUI 機能を含む.
3. *Toolbox* と名付けられたアプリケーションに特化したライブラリを有しいる.（システム制御，信号処理, ニューラルネット, ウェーブレット, シミュレーション等[1].）

主な MATLAB の応用分野

1. 数学, 計算理論，工学全般（特に制御工学）
2. アルゴリズム開発
3. モデリング, シミュレーション
4. データ解析, 診断, 可視化
5. 科学・工学用グラフィックス
6. GUI 構築を含むアプリケーション開発

MATLAB と言う名前は, **MAT**rix **LAB**oratory の略で，元々行列計算に対する最先端ソフトウェアである LINPACK, EISPACK により開発された行列ソフトウェアを利用，発展させて現在の MATLAB になっている.

1.2 MATLAB の起動方法と終了方法

注意：(情報処理センターの環境に依存するので，以下の表示が多少異なるかもしれません．)

まず計算サーバ (pine) へリモートログインします．そして pine の Terminal 上で `matlab` と打ち込みます.

```
pine% matlab
```

MATLAB の画像が一瞬表示されます. 数行の英語のコメントの後, 今まで % となっていたプロンプトが>>となります. この状態が MATLAB が立ち上がってる状態です.

```
>>
```

この状態で様々な MATLAB のコマンドを使うことが可能となります.

MATLAB を使い終わったら, 元の状態に戻す必要がありますから．まずは MATLAB を終了させてみましょう. 終了には, `exit` コマンドを使います.

```
>> exit
```

すると今まで >>となっていた部分が, % に戻ります. これで MATLAB が終了します.

1.3 MATLAB の簡単な使い方

まず行ベクトル

$$t = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \tag{1}$$

を定義してみましょう. Window 上で

```
>> t=[1 2 3]
```

と打ってください. すると以下のように表示されます.

[1] pine には simulink しかインストールされていません

```
>> t=[1 2 3]

t =
    1   2   3
>>
```

次に列ベクトル

$$u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \tag{2}$$

を定義してみます. この場合は [ ] 内の数字に;(セミコロン) をつけます.

```
>> u=[1;2;3]

u =
    1
    2
    3

>>
```

では, 行列

$$V = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{3}$$

はどうすればよいでしょうか? 行ベクトルを縦に並べればいいので,

```
>> V=[1 2 3 ; 4 5 6 ; 7 8 9]

V =
    1 2 3
    4 5 6
    7 8 9

>>
```

とすればよいです.

次のステップに進みます. 次は設定したベクトルを使い計算します. まず前述の t を列ベクトルにします. すなわち, t の転置

$$t^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \tag{4}$$

を行います. それには以下のコマンドがあります.

```
>> t'
```

すると,

```
>> t'

ans =
     1
     2
     3

>>
```

となります. ans とは, answer の省略形であり, 直前のコマンドの出力を指します.
　では, 上記の t^T と t の積を行うことで行列の掛け算が簡単に行えることを確認します. 積の演算子は*です.

```
>> t'*t

ans =
     1   2   3
     2   4   6
     3   6   9

>>
```

となります.
　今まではコマンドを打つとその出力がすぐに得られていましたが, 出力を省略する方法があります.

```
>> t=[1 2 3];
```

と入力すると

```
>> t=[1 2 3];
>>
```

と, プロンプトだけが表示されますが, MATLAB 内には t が記憶されています.
　では, MATLAB 内の変数はどうやって調べるのでしょうか？それには who コマンドを使います.

```
>> who

Your variables are:

V         ans       t         u

>>
```

MATLAB の中に変数として登録されていることが分かります.
　次に進む前に, MATLAB 内の変数を消去してみます. 変数を消すには clear コマンドを使用します.

```
>> clear
```

who コマンドで確認すると, 先程は Your variables are: と表示されたのが, 今回は何も表示されないはずです. これで MATLAB 内の変数が消えました.
　次にアレイ状に変数 k を定義します.

```
>> k=0:0.1:10;
```

とすることで, k は 101 個の要素をもつ行ベクトルとなります. この k は, 0 から 10 までを 0.1 刻みで分割することを表します.
　ここで, sin 関数を使ってみます. sin の出力結果は y に代入されます.

```
>> y=sin(k);
```

1×101 の大きさを持つ sin の出力 y が得られます. すなわち, アレイとは, 連続値を近似しているとも言えます. whos コマンドで確認してみましょう. whos は, who よりも詳細な情報が得られます.

```
>> whos
  Name   Size   Bytes Class

  k      1x101  808  double array
  y      1x101  808  double array

Grand total is 202 elements using 1616 bytes

>>
```

Name, Size で確認が出来るはずです. 最後にグラフを使用してみます. グラフを描画するには, plot を用います. ( ) の中に描きたい変数を入力します. この時, 先に入力した方（引数）が横軸に, 後に入力した方が縦軸になります.

```
>> plot(k,y)
```

MATLAB にはこの他にも様々な数学関数やグラフィックス関数がありますので是非積極的に使ってください.

1.4 m-file について

MATLAB には幾つか便利な機能がありますが, その中の 1 つである “m-file” について説明します. m-file とは, 拡張子に “.m” を持ち, MATLAB 上で読み込むことの出来るファイルを指します. この m-file の利点は, シェル上に何度も同じコマンドを打ち込む必要がなく, 使い慣れたエディタ (mule, vi, etc) で編集が可能であることです. 作成方法は, エディタで作成したファイルを保存する時に, 拡張子 “.m” をつけるだけです. 実際のコマンドラインと比較してみましょう. 簡単な計算を例に出します. a=5, b=3, c=a+b とします.

シェル上でコマンドを打ち込む場合

```
>> a=5;
>> b=3;
>> c=a+b

c=
      8

>>
```

m-file を用いる場合

まずはエディタ (ke 等) を開いて, ファイルの中に

```
a=5;
b=3;
c=a+b
```

と書きます. この時ファイル名は “filename.m” とします. filename は分かりやすい名前にしておきましょう. ここでは “test.m” とします. 次に, MATLAB プロンプト上で m-file から拡張子 “.m” を除いたファイル名を打ち込みます.

```
>> test

c=
      8

>>
```

こうすると, m-file の中身を MATLAB プロンプト上で打ち込んだ時と同様の結果が得られます. この例ではプログラム自体が短いものでしたが, 長いプログラムを作る時や同じプログラムで数値を換えて出力させたい時などに有用です.

注意点

- ファイル名に 数式, MATLAB で用いられるコマンド名を使わないでください (1-2.m, sin.m 等).
- ファイルの保存場所と MATLAB 上で作業しているディレクトリが一致していないと m-file が使えません.[2]

[2] matlab path を通しておけば使えます.

1.5 図の保存方法

MATLAB では色々な図 (figure) が描けますが, その保存方法を説明します. まずは図を表示します. 表示の仕方は幾つかありますが, ここでは省略して図が表示されているものとして進めていきます. まずは分かりやすくするために図にタイトルを入れます.

```
>> title('*******')
```

'*******' には何の図か, 名前, 名列番号等を入れておきましょう. 表示されているの中にタイトルが表示されます. また

```
>> xlabel('*******')
>> ylabel('*******')
```

とすることで, 図の x 軸と y 軸に名前をつけることができます. ここでは, 試しに

```
>> xlabel('k')
>> ylabel('sin(k)')
```

としてみてください. 次にその図を保存します.

```
>> print -dps filename
```

これで, 図が ps ファイル (Postscript 形式) として保存されました.

2 MATLAB による行列と線形代数

2.1 節では, データの定義について，復習を兼ねて，まとめています. 2.2 節では, 行列と線形代数について, 行列演算について説明します.

2.1 データの定義

MATLAB でのデータの取り扱いや適用範囲は, 非常に柔軟です. MATLAB 上で扱う変数には, 以下のような特徴があります.

- スカラやベクトルデータを含めて, データはすべて配列として取り扱われる.
- 変数の型宣言 (実数, 複素数, 文字) 等は不要である.
- 数値データは, 演算において倍精度浮動小数点型で取り扱われる.

2.1.1 数値の入力方法

入力規則

- 配列の要素間は, ブランク (空文字) またはカンマ (,) で区切る.
- 配列の各行の終了は, 改行 (リターン), またはセミコロン (;) で定義する.
- 配列は, 要素全体を鍵括弧 ([ ]) で囲んで定義する.
- 1 行で記述しきれない場合は, MATLAB の継続記号として, 最後にピリオド 3 つ (...) 連続して入力し，次の行に継続する.
- 計算結果を画面 (MATLAB コマンドウィンドウ) 上に表示させたくない場合は, コマンドラインの最後にセミコロン (;) をつける.

例題 2.1

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}$ を定義する.

解) 定義の方法はたくさんありますが, 幾つかの例を以下に示します.

(a) 配列の要素間にブランクをあけた場合

```
>> A=[1 2 3 ; 4 5 6 ; 7 8 0]
A=

  1   2   3
  4   5   6
  7   8   0
>>
```

(b) 配列の要素間をカンマ (,) で区切った場合

```
>> A=[1, 2, 3 ; 4, 5, 6 ; 7, 8, 0]
A=

  1   2   3
  4   5   6
  7   8   0
>>
```

(c) 配列の各行の終了を改行 (リターン) にした場合

zeros	零行列を作成
ones	要素がすべて 1 の行列を作成
eye	単位行列を作成
diag	対角行列を作成
magic	魔方陣行列を作成
rand	一様分布の乱数を作成
:	線形等間隔のベクトルを作成

Table 1: 演算子, 配列操作関数

```
>> A=[1 2 3
4 5 6
7 8 0]
A=

  1   2   3
  4   5   6
  7   8   0
>>
```

(d) 1 行ではなく, ピリオド 3(...) により複数行にした場合

```
>> A=[1 2 3; 4 5 6; 7 ...
8 0]

A=

  1   2   3
  4   5   6
  7   8   0
>>
```

注意

(1) 文字列 (または文字変数) は大文字と小文字を区別するので, A と a は別の変数として認識する.

(2) 変数の文字数制限は 31 文字.

(3) 数字および演算子で始まる変数名は認識されない.

(4) 虚数単位 i のような永久変数 (pi(円周率), eps(浮動小数点相対精度), ..etc) と同じ変数名を使用しない.

(5) 関数 (sin, ..etc), コマンド (pwd, ..etc) と同じ変数名を使用しない.

(6) 同じ変数名でデータを定義すると, 値が上書きされる.

2.1.2 MATLAB の関数を使う方法

MATLAB は, 一般行列関数や特殊行列関数として Table 1 のような関数を提供しています. また, 行列要素の抽出や付け足し等の操作も容易に行えます.

例題 2.2

(1) 基本行列の定義

(a) 2 行 3 列の零行列 $X1$ を定義する.
入力引数として, 行数と列数を与える. 1 つの値のみを与えた場合は, 正方行列を作成する.

```
>> X1=zeros(2,3)

X1=

   0   0   0
   0   0   0
>> X11=zeros(2)

X11=

   0   0
   0   0
>>
```

(b) 3 行 3 列の単位行列 $X2$ を定義する.

```
>> X2=eye(3)

X2=

   1   0   0
   0   1   0
   0   0   1
>>
```

(c) 対角要素に 2, 5, 7 の値をもつ対角行列 $X3$ を定義する.

```
>> X3=diag([2 5 7])

X3=

   2   0   0
   0   5   0
   0   0   7
>>
```

(d) 逆に対角要素を取り出す.

```
>> X4=diag(X2)

X4=

   1
   1
   1
>>
```

(2) 等間隔のベクトルを定義する.

(a) Y1 を 25 から 0 まで -5 刻みで変化する数列を定義する.
等差数列のベクトルを定義する場合, コロン記号 (:) を用いることができる. 初期値, 公差, 最終値をそれぞれコロン記号で区切って定義する.

```
>> Y1=25:-5:0

Y1=

   25   20   15   10   5   0
>>
```

(b) Y2 として 1 から 5 まで 1 ずつ変化する数列を定義する.
公差を省略した場合は, 公差 1 のベクトルになる. また, 公差が 1 と考えれば, (a) と同様に考えることができる.

```
>> Y2=1:5

Y2=

   1   2   3   4   5
>> Y2=1:1:5

Y2=
   1   2   3   4   5
```

(c) Y3 は, 0 から 3 刻みに増加する数列で, 要素が最終値と一致しない場合を考える.
最終値が数列要素と一致しない場合, 最終値以下で条件を満たす値がベクトルとして出力される.

```
>> Y3=0:3:10

Y3=

   0   3   6   9
>>
```

(3) 配列の大きさを調べる.

配列の大きさを知るには, 関数 size を用いる. 出力引数を 1 つ与えると, 2 要素のベクトルとして [行数 列数] の形で出力される. 出力引数を 2 つ与えると, 行数と列数がそれぞれスカラ値として出力される. また, 行数と列数のうち大きい方を出力する関数 length もある.

```
>> d=size(X1)

d=

  2   3
>> length(X1)

d=
  3
```

(4) 配列要素の抽出や, 入れ替え, 追加を行う.
行と列のインデックスを与えることによって, その要素を抽出したり, 入れ替えることができる. ここでは, はじめに 3×3 の乱数行列 Z1 を定義し, Z1 に対し, これらの操作を行う. 準備として適当な行列 $Z1$ を定義する.

```
>> Z1=rand(3)

Z1=

   0.2190   0.6793   0.5194
   0.0470   0.9347   0.8310
   0.6789   0.3835   0.0346
>>
```

(a) 入れ替え (変数 Z1 の 3 行 2 列目の値を 1 に変換する).

```
>> Z1(3,2)=1

Z1=
```

```
   0.2190   0.6793   0.5194
   0.0470   0.9347   0.8310
   0.6789   1.0000   0.0346
>>
```

(b) 抽出 (変数 Z1 の 3 行 2 列目の値を抽出, 変数 Z1 の 1 列目の値を抽出する).

```
>> a=Z1(3,2)

a=

   1
>> b=Z1(:,1)

b=

   0.2190
   0.0470
   0.6789
>>
```

コロン記号は, すべての行 (または列) を意味するので, 変数 b には Z1 のすべての行の 1 列目の要素が入る. Z1(:) とすると, Z1 のすべての要素を列ベクトルとして表す.

(c) 追加 (変数 Y2 の 10 番目に値 10 を追加, 変数 X3 と 変数 Z1 を結合する.)

```
>> Y2(10)=10

Y2=

   1    2    3    4    5    0    0    0    0   10
>> Z2=[X3 Z1]

Z2=

   2.0000   0        0        0.2190   0.6793   0.5194
   0        5.0000   0        0.0470   0.9347   0.8310
   0        0        7.0000   0.6789   1.0000   0.0346
>>
```

行列要素の追加は, ある特定のインデックスに入る数値を与えることによっても行うことができる (Y2 参照). このとき, 指定しなかった要素には 0 が入る. また, Z2 のように要素数が合う場合は, 行列を要素として入れることもできる. 要素数が合わない場合には, 下記のようなエラーメッセージが表示される.

```
>> [A X1]
??? All matrices on a row in the bracketed expression must have
      the same number of rows.
```

2.2 行列と線形代数

MATLAB では, 行列の四則演算を行うために Table 2, 3 のような関数が用意されています. また, 四則演算だけではなく, 行列操作を行う関数や基本的な数学関数などもあります.

例題 3.1 次の各行列演算をおこなう.

(1) 次の行列の加算を求める.

$$\begin{pmatrix} 1 & 6 & 3 \\ 7 & 0 & 2 \\ 4 & 5 & 9 \end{pmatrix} + \begin{pmatrix} 7 & 2 & -1 \\ 6 & 8 & 3 \\ 9 & 2 & 2 \end{pmatrix} \tag{5}$$

+	配列の加算
−	配列の減算
*	行列の乗算
/	行列の右割り
\	行列の左割り
^	行列のべき乗
.*	配列の乗算
./	配列の右割り
.\	配列の左割り
.^	配列のべき乗
′	複素共役転置
.′	配列の複素共役転置
reshape	行列の変形

Table 2: 演算子, 配列操作関数

sin	正弦値
cos	余弦値
abs	絶対値
log	自然対数
log10	常用対数
sqrt	平方根
inv	逆行列
eig	固有値と固有ベクトル
lu	LU 分解
chol	Cholesky 分解
qr	QR 分解
svd	特異値分解
trace	対角要素の和

Table 3: スカラー関数, 行列関数

解) はじめに行列 X を $X = \begin{pmatrix} 1 & 6 & 3 \\ 7 & 0 & 2 \\ 4 & 5 & 9 \end{pmatrix}$ と定義する. MATLAB では

```
>> X=[1 6 3 ; 7 0 2 ; 4 5 9]
```

と打てば定義され, 以下のように表示されます.

```
>> X=[1 6 3 ; 7 0 2 ; 4 5 9]

X =
    1   6   3
    7   0   2
    4   5   9
>>
```

もちろん, 結果を表示したくないならばセミコロン (;) をつけて

```
>> X=[1 6 3 ; 7 0 2 ; 4 5 9];
```

と打てば, 次のように結果は表示されません.

```
>> X=[1 6 3 ; 7 0 2 ; 4 5 9];
>>
```

次に行列 Y を $Y = \begin{pmatrix} 7 & 2 & -1 \\ 6 & 8 & 3 \\ 9 & 2 & 2 \end{pmatrix}$ と定義する. MATLAB では

```
>> Y=[7 2 -1 ; 6 8 3 ; 9 2 2]
```

と打てば定義され, 以下のように表示されます.

```
>> Y=[7 2 -1 ; 6 8 3 ; 9 2 2]

Y =
```

```
    7   2   -1
    6   8   3
    9   2   2
>>
```

最後に X と Y の加算をします. MATLAB では'+' を用いて

```
>> X+Y
```

と打つことにより加算がおこなわれます. 表示は以下のようになります.

```
>> X+Y

ans =
    8    8    2
    13   8    5
    13   7    11
>>
```

これにより行列の和が計算できました.

$$\begin{pmatrix} 1 & 6 & 3 \\ 7 & 0 & 2 \\ 4 & 5 & 9 \end{pmatrix} + \begin{pmatrix} 7 & 2 & -1 \\ 6 & 8 & 3 \\ 9 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 8 & 8 & 2 \\ 13 & 8 & 5 \\ 13 & 7 & 11 \end{pmatrix} \tag{6}$$

(2) 次の行列積を求める.

$$\begin{pmatrix} 2 & 6 \\ 8 & 3 \\ 7 & 1 \end{pmatrix} \begin{pmatrix} 2 & 8 & 7 \\ 6 & 3 & 1 \end{pmatrix} \tag{7}$$

解) はじめに行列 A を $A = \begin{pmatrix} 2 & 6 \\ 8 & 3 \\ 7 & 1 \end{pmatrix}$ と定義する. MATLAB 上では

```
>> A=[2 6 ; 8 3 ; 7 1]
```

と A を定義され, 以下のように表示されます.

```
>> A=[2 6 ; 8 3 ; 7 1]

A =
    2   6
    8   3
    7   1
>>
```

また, 2 つめの行列は A の転置行列 (A^T) であることがわかる.

$$A^T = \begin{pmatrix} 2 & 8 & 7 \\ 6 & 3 & 1 \end{pmatrix} \tag{8}$$

MATLAB で行列の転置をするには, シングルクォート (') で行なうので,

```
>> A*A'
```

とすることで行列の積が計算できます.

```
>> A*A'

ans =
     40    34    20
     34    73    59
     20    59    50
>>
```

よって,

$$\begin{pmatrix} 2 & 6 \\ 8 & 3 \\ 7 & 1 \end{pmatrix} \begin{pmatrix} 2 & 8 & 7 \\ 6 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 40 & 34 & 20 \\ 34 & 73 & 59 \\ 20 & 59 & 50 \end{pmatrix} \tag{9}$$

となります.

(3) $A = \begin{pmatrix} 3 & 6 \\ -2 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 8 & 0 \\ 5 & 7 \end{pmatrix}$ としたとき, 要素ごとの積を求める.

解)(2) の問題との違いは, 行列の積であるか要素ごとの積であるかである. たとえば, A, B 行列が

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \tag{10}$$

で与えられているとする. (2) の問題で求めたのは,

$$AB = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} \tag{11}$$

であり, ここで求めるのは

$$\begin{pmatrix} A_{11}B_{11} & A_{12}B_{12} \\ A_{21}B_{21} & A_{22}B_{22} \end{pmatrix} \tag{12}$$

であることに注意する.

行列 A と B を定義します.

```
>> A=[3 6 ; -2 1];
>> B=[8 0 ; 5 7];
```

MATLAB で要素ごとの乗算をするには, 演算子 ($*$) の前にドット (.) をつけるので

```
>> A.*B
```

とすることで計算でき, 以下のように表示されます.

```
>> A.*B

ans =
     24     0
    -10     7
>>
```

(4) 行列 A の対角要素の和と, 行列 A の固有値の和が等しいことを確認する.

解) 行列 A を作ります. 別にどんな行列でもいいのですが, ここではコマンドの紹介も含めて, 魔方陣行列を作ります. MATLAB で魔方陣行列を作るには magic(N) を用います. magic(N) は, 1 から N^2 までの整数を使って, 行方向, 列方向, 対角方向の和が等しくなる N 行 N 列の行列を作成します.

行列 A を 3 行 3 列の行列とすると $N = 3$ とすればいいので

```
>> A=magic(3)
```

とすることで行列 A を定義します.

```
>> A=magic(3)

A=
   8   1   6
   3   5   7
   4   9   2
>>
```

行列 A は

$$A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \tag{13}$$

と定義されました. 次に行列 A の対角要素の和を求めます. MATLAB で行列 A の対角要素の和を求めるには, Trace(A) を用います. そこで

```
>> trace(A)
```

とすれば対角要素の和が求まります.

```
>> trace(A)

ans=15
>>
```

対角要素の和は 15 でした. 次に行列 A の固有値と固有ベクトルを求めます. 行列 A の固有値と固有ベクトルを求めるには eig(A) を用います.

```
>> eig(A)
```

と打てば, 正方行列 A の固有値からなるベクトルを出力します. 固有ベクトルも求めたい場合は,

```
>> [V,D] = eig(A)
```

とします. [V,D] = eig(A) は, 固有値からなる対角行列 D と $AV = VD$ となるような固有ベクトルからなるフル行列 V を出力します.

```
>> [V,D]=eig(A)

 V=
   0.5774   0.8131   -0.3416
   0.5774   -0.4714   -0.4714
   0.5774   -0.3416   0.8131

 D=
   15.0000       0         0
       0      4.8990       0
       0         0      -4.8990
>>
```

最後に, 行列 D の対角要素は固有値であるのでその和を求めます. 対角要素の和をもとめるには Trace を用いればよいので,

```
>> trace(D)

 ans=
     15.0000
```

とすることで, 固有値の和は 15 であることがわかります. よって, 行列 A の対角要素の和と, 行列 A の固有値の和が等しいことが確認できます.

例題 3.2 次の連立 1 次方程式を解く.

$$\begin{cases} x + 5y = 7 \\ 2x + 4y = 8 \end{cases} \tag{14}$$

解) まず, 式 (14) を線形の方程式として見やすくします.

$$\begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \end{bmatrix} \tag{15}$$

ここで,

$$A = \begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix}, \qquad X = \begin{bmatrix} x \\ y \end{bmatrix}, \qquad b = \begin{bmatrix} 7 \\ 8 \end{bmatrix} \tag{16}$$

とおくことにより, 次のように書き換えられます.

$$AX = b \tag{17}$$

MATLAB では, はじめに行列 A と B を定義します.

```
>> A=[1 5 ; 2 4];
>> b=[7 ; 8];
```

いま求めたいのは X ですが, 式 (17) から

$$X = A^{-1}b \tag{18}$$

であることがわかります. ここで, A の逆行列が存在するかどうかを確認する必要があり, 方法としては行列 A のランクを確認して, フルランクであれば OK です.

```
>> n=rank(A)
```

とすれば求まり,

```
>> n=rank(A)
n=
  2
```

と表示されます. A は 2 行 2 列の行列ですからフルランクであることがわかります. 次に X を求めます. MATLAB では行列除算演算子 (\, もしくは ¥) を用います.

```
>> X=A\b
```

とすることで解 X が求まります.

```
>> X=A\b

X=2.000
  1.000
```

注意として, 式 (18) から行列 A の逆行列を求めれば行列除算演算子 (\) を用いなくても求まりそうです. MATLAB では A の逆行列は inv(A) により求まり,

```
>> X=inv(A)*b
```

とする方法もあります. ところが, この逆行列を用いた解法は計算時間と数値精度の観点から見ると良い方法ではありません.

演習問題

(1) 以下のデータを MATLAB の関数を使って定義せよ.

1-1)

$$\begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{pmatrix} \tag{19}$$

1-2)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 & 0 & 3 \end{pmatrix} \tag{20}$$

1-3)

$$\begin{pmatrix} 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 4 \\ 6 & 6 & 6 & 6 \\ 8 & 8 & 8 & 8 \\ 10 & 10 & 10 & 10 \end{pmatrix} \tag{21}$$

(2) 次の連立方程式を求めよ.

$$\begin{cases} 2x + y + 5z = 5 \\ 2x + 2y + 3z = 7 \\ x + 3y + 3z = 6 \end{cases} \tag{22}$$

(3) 解の公式を用いて, 次の 2 次方程式の解を求めよ.

$$x^2 + 6x + 10 = 0 \tag{23}$$

3 Simulink の基本操作

本章は Simulink と呼ばれる MATLAB のソフトウェアの使い方を簡単にまとめています[3]. 第 3.1 節では, Simulink の概要を紹介しており, 第 3.2 節では, 実際に簡単なモデルを作成し, シミュレーションを行います.

3.1 Simulink とは

Simulink は, ダイナミックシステムのモデル化, シミュレーション, 解析を行うためのソフトウェアパッケージです. モデル化のために Simulink は, マウスのクリックアンドドラック操作を用いて, ブロック線図としてモデルを作成するためのグラフィカルユーザインターフェイス (GUI) を有しています. この GUI を使用すると, 紙と鉛筆を用いて描くのと同じようにモデルを描画できます.

デモモデルの実行はじめに, Simulink の概要を把握するためにデモモデルを実行してみましょう. MATLAB コマンドウィドウで

```
>> thermo
```

と入力するとデモモデルが立ち上がります. 次に Thermo Plots をダブルクリックし, Scope を開きます. シミュレーションを開始するには, **Simulation** メニューをプルダウンし, **Start** コマンドを選択します. シミュレーションが実行されると, 住宅の温度出力と外気の温度が Indoor vs. Outdoor Temp の Scope ブロックに, 累積暖房出力が Heat Cost($) の Scope ブロックに表示されます.

シミュレーションを終了するには, **Simulink** メニューから **Stop** コマンドを選択します. ただし, 初期設定では, 10 秒でとまるので **Stop** コマンドを実行する必要はありません.

シミュレーションの実行が終了したら, **File** メニューから **Close** を選択してモデルを閉じます.

3.2 簡単なモデルの作成

ここでは, 簡単なモデルを作成するための方法を説明します. いまから, Fig. 1 に示すようなモデルを作成します.

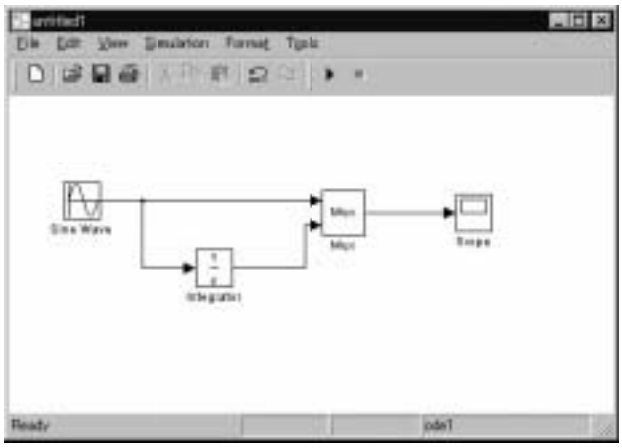

Fig.1: Simulink モデル

3.2.1 Simulink の起動

Simulink を起動させます. MATLAB コマンドウィンドウで

```
>> simulink
```

と入力すると, Fig. 2 に示すように Simulink ブロックライブラリウィンドウが表示されます. ここで作成するモデルは, つぎのライブラリを使用します.

- Sources ライブラリ (Sine Wave ブロック)
- Sinks ライブラリ (Scope ブロック)
- Linear ライブラリ (Integrator ブロック)
- Connections ライブラリ (Mux ブロック)

[3] **(注意)** 本稿は MATLAB ver **5.3.1** に基づいて記述している. 情報処理センター (pine) の MATLAB のバージョンは **6.1.0** (02/06/01 現在)

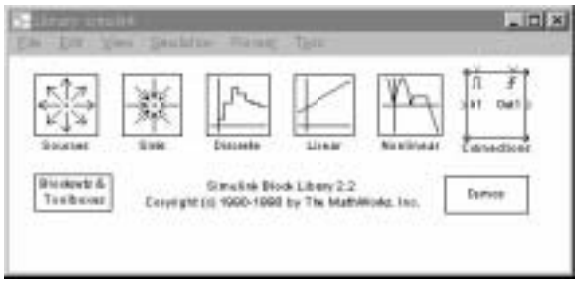

Fig.2: Simulink ブロックライブラリウィンドウ

3.2.2 新規モデルウィンドウの作成

モデル作成のためのウィンドウを作成します. Simulink ブロックライブラリウィンドウの **File** メニューから **New** の **Model** を選択します. すると, Fig. 3 に示すような新規のウィンドウが開きます.

Fig.3: 新規モデルウィンドウ

3.2.3 ブロックのコピー

いま, 作成した新規のモデルウィンドウにブロックライブラリからブロックをコピーします.

Sources ライブラリを開いて Sine Wave ブロックにアクセスします. ブロックライブラリを開くには, ライブラリのアイコンをダブルクリックします. Sources ライブラリ (Fig. 4) では, すべてのブロックが信号源です.

Fig.4: Sources ライブラリ

ブロックライブラリまたは他のモデルからブロックをコピーすることによって, モデルに追加します. Sine Wave ブロックをコピーします. Sine Wave ブロック上にカーソルを合わせ, マウスボタンを押します. そして, ブロックをモデルウィンドウ内までドラッグし, モデルウィンドウ内においてブロックを配置したい位置でマウスボタンを解除します. Sine Wave ブロックのコピーがモデルウィンドウの中に表示されます (Fig. 5).

同様に残りのブロックをモデルウィンドウにコピーします.

すべてのブロックがモデルウィンドウにコピーされると Fig. 6 のようになります. ブロックアイコンをみると, Sine Wave ブロックの右側に角括弧 (>) があり, Mux ブロックの左側には 3 つの角括弧があるのがわかります. ブロックから >記号が出ている場合は**出力ポート**です. 記号がブロックをさしている場合は**入力ポート**です. 信号は出力ポートから出て, 信号は出力ポートから出て, 接続線を通じて別のブロックの入力ポートに入ります. ブロックが接続されるとポートは消えます.

Fig.5: Sine Wave

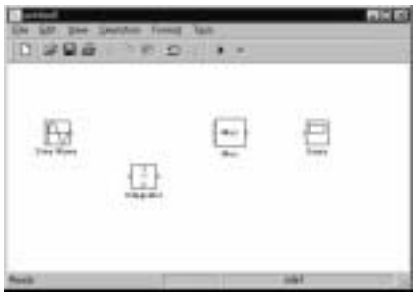

Fig.6: モデル

3.2.4 ブロックのパラメータの変更

Mux ブロックには入力ポートが 3 つありますが, 入力信号は 2 つしか必要ないので変更します. 入力ポートを変更するには, Mux ブロックをダブルクリックしてダイアログボックスをオープンします (Fig. 7). **Number of inputs** パラメータ値を 2 に変更した後, **Close** ボタンをクリックします.

Fig.7: Mux ブロック

3.2.5 ブロックの結線

次にブロックを結線します. Sine Wave ブロックを Mux ブロックの上部の入力ポートに結線します. ポインタの位置を Sine Wave ブロックの右側の出力ポートに合わせます. カーソルの形がクロスヘア (十文字) に変わります. マウスボタンを押して, カーソルを Mux ブロックの最上部の入力ポートに移動します. マウスを押している間はラインは破線となり, カーソルを Mux ブロックに近づけるとカーソルの形が二重のクロスヘア変わります. ここでマウスを解除します. ブロックが結線されました (Fig. 8).

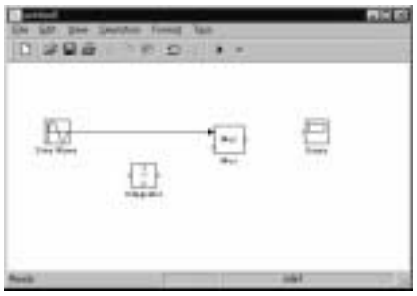

Fig.8: ブロックの結線 (1)

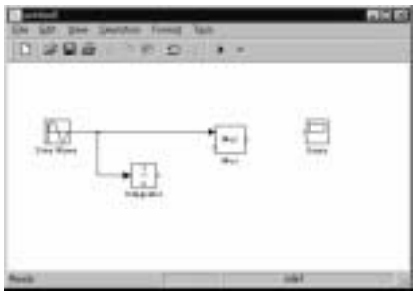

Fig.9: ブロックの結線 (2)

Fig. 9 をみるとラインがブロックの出力ポートを他のブロックの出力ポートに結線してあります. ただし, 1 本のラインは**ライン**を別のブロックの入力ポートに結線しています. これは, **分岐線**とよばれ, Sine Wave 出力を Integrator ブロックに接続し, Sine Wave ブロックから Mux ブロックに渡されるのと同じ信号を伝搬します.

分岐線の描画は, 次の手順に従ってください.

1 ポインタの位置を Sine Wave ブロックと Mux ブロックとの間のライン上に合わせる.

2 **Ctrl** キーを押しながらマウスボタンを押す. マウスボタンは, そのままにして, ポインタを Integrator ブロックの入力ポート, または, Integrator ブロックそのものにまでドラッグする.

3 マウスボタンを解除する. Simulink は, 起点と Integrator ブロックの入力ポートとの間にラインを引く.

ブロックの接続ができました. モデルは Fig. 10 のようになります.

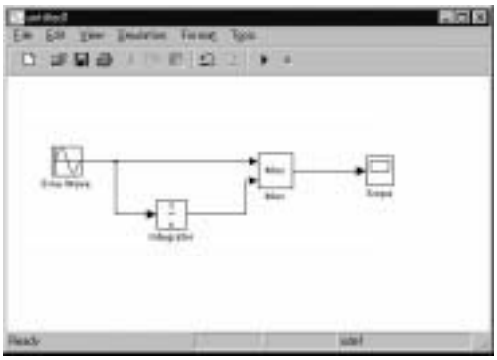

Fig.10: ブロックの結線 (3)

3.2.6 シミュレーションの実行

次に Scope ブロック (Fig. 12) を開いて (ダブルクリックして) シミュレーション出力を表示します. 10 秒間のシミュレーションを実行します. **Simulation** メニューから **Parameter** を選択して, シミュレーションパラメータを設定します. 表示されるダイアログボックス上で **Stop time** が 10.0(デフォルト値) に設定されていることに注意します. **Close** をクリックして **Simulation Parameter** ダイアログボックスを閉じます.

Fig.11: シミュレーションの設定

Fig.12: Scope

Simulation メニューから **Start** を選択し, Scope ブロックの入力を追跡します. シミュレーションは, **Simulation Parameters** ダイアログボックスで指定した終了時間に達するか, Simulation メニューから **Stop** を選択すると停止します.

このモデルを保存するには, **File** のメニューから **Save** を選択し, ファイル名と保存場所を入力します.

演習問題

(1) 1 次系の応答

1 次系のステップ応答を描きなさい. Fig. 13 に構成図を載せておく.

- Step ブロックをダブルクリックすることで Fig. 14 のように設定を行う. Step ブロックはシミュレーション時間が **Step time** パラメータ値より小さい場合, ブロックの出力は **Initial value** パラメータ値となる. シミュレーション時間が **Step time** 以上の場合, 出力は **Final value** パラメータ値となる.
- Transfer Fcn ブロックも同様に設定を行なうことができる. **Transfer Fcn** は伝達関数である. 例えば,

```
Numerator:[1]
Denominator:[2 3]
```

とすると, この伝達関数 $G(s)$ は

$$G(s) = \frac{1}{2s+3} \tag{24}$$

を表す. また,

```
Numerator:[4]
Denominator:[1 2 3]
```

とすると, この伝達関数 $G(s)$ は

$$G(s) = \frac{4}{s^2 + 2s + 3} \tag{25}$$

を表す.

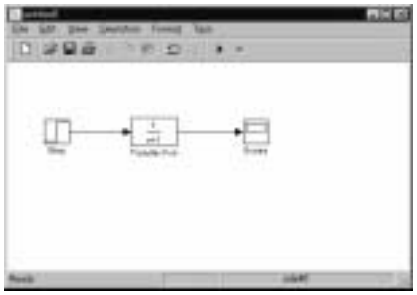

Fig.13: 1 次系の応答

Fig.14: Scope ブロック

Fig.15: Transfer Fcn ブロック

(2) 2 次系の応答
2 次系のステップ応答を描きなさい.