Octaveによる
動的システムシミュレーション入門

２００２年２月

島根大学総合理工学部
電子制御システム工学科

吉田和信

目 次

第 1 章　Octave 入門 **1**
1.1 Octave の起動と終了 1
1.2 数値の入出力 2
1.3 行列の入出力 3
1.4 行列の和・差・積・転置・べき乗 4
1.5 行列関数 6
1.6 単位行列・零行列・対角行列 9
1.7 条件判定・繰り返し 9
1.7.1 if, else, elseif 9
1.7.2 switch 11
1.7.3 while 12
1.7.4 for 12
1.8 種々の行列の作成法 14
1.8.1 行列要素の指定 14
1.8.2 行列のサイズ 15
1.8.3 行列の結合と削除 16
1.9 数学関数 20
1.10 多項式に関する関数 24
1.10.1 多項式の表現，零点，特性方程式 24
1.10.2 多項式曲線のあてはめ 26
1.11 ユーザ定義関数の書き方 27
1.12 ファイルに対する入出力 29

第 2 章　シミュレーションの基礎 **31**
2.1 オイラー法 31
2.2 ルンゲクッタ法 33
2.3 伝達関数から状態方程式を作る方法 35
2.3.1 １次系 35
2.3.2 2 次系 35
2.3.3 分母と分子の次数が等しい場合 37
2.4 位相面図の描き方 38
2.5 非線形制御の例：可変長振子系の振動制御問題 42

2.6 制御系設計用関数 47
2.6.1 極配置法 47
2.6.2 オブザーバ 48
2.6.3 LQG 制御 53
2.7 伝達関数に基づく解析法 57
2.7.1 システムの伝達関数表現 57
2.7.2 伝達関数の結合 59
2.7.3 零点，極，ゲイン 64
2.7.4 過渡応答 65
2.7.5 周波数応答 66
2.7.6 根軌跡 69
2.7.7 制御系の設計例 71

第1章　Octave入門

GNU Octave は Matlab に似たフリー数値計算ソフトウェアである．主として，John W. Eaton によって 1992 年頃から本格的に開発が進められ，現在，安定バージョンとして 2.0.16 が，開発中バージョンとして 2.1.35 が提供されている．

Octave は，プログラミングで必要とされる基本言語 (ほぼ Matlab 互換) を持ち，対話形式とバッチ形式の両方で使える．ベクトル，行列，複素数の取り扱いが簡単で，行列式や固有値などの行列に関する計算，線形および非線形方程式の求解，多項式演算，微分方程式の求解，gnuplot を用いたグラフ表示等の機能を持つ．Matlab と同じ名前・機能の関数を多く持つが，Octave 固有の関数もあり，特徴を出している．また，制御工学，信号処理などの分野における関数も用意されている．

本テキストは，先に作成したテキスト「MATLAB による動的シミュレーション入門」[1] を Octave 用に書き換えたものである．概ね，[1] に沿っているが，特に，2.7 節の古典制御理論の部分を Octave 関数を用いて加筆し，Matlab の数学的関数に関する関数の部分は Octave が相当する関数を持たないため割愛した．本テキスト作成にあたり使用した Octave のバージョンは 2.1.35 であり，2.5 節までのプログラムはバージョン 2.0.16 でも実行できることを確認している．

では，Octave によるプログラミングを学習しよう．

1.1　Octaveの起動と終了

コマンドウィンドウで

```
octave
```

と入力すれば，Octave を起動できる．Octave を終了するには

```
octave:1> quit
```

または

```
octave:1> exit
```

と入力する．

何らかのトラブルで Octave が暴走した場合，`Ctrl-c` (`Ctrl` キーを押しながら `c` を押す) で中断できる．

Octave Window では，コマンドを入力して実行する．バッチ処理のプログラムはエディタで作り，このファイル (以後 M ファイルという) を拡張子 m を付けて「ファイル名.m」として作業ディレクトリに保存する．それから，Octave Window で「ファイル名」(拡張子は付けない) を入力することにより実行する．

1.2　数値の入出力

変数に数値を代入するには次のようする．

```
octave:1> a = 1
a = 1
octave:2> b = 1.23
b = 1.2300
octave:3> c = 3;
```

変数名として，英数字と '_' (アンダースコア) が使える．英字の大文字と小文字は区別され，先頭文字に数字は使えない．また，長さは無制限である．文の最後に ';' を付ければ，出力を抑制できる．

出力書式の指定もできる (%より右はコメントなので打たなくてよい)．

```
octave:4> x = pi            % 円周率
x = 3.1416                  % デフォルト表示
octave:5> format long       % long 表示
octave:6> x
x = 3.14159265358979
octave:7> format short      % short 表示
octave:8> x
x = 3.14
octave:9> format            % デフォルト表示
octave:10> x
x = 3.1416
octave:11> format long e    % 浮動小数点形式の long 表示
octave:12> x
x = 3.14159265358979e+00
octave:13> format short e   % 浮動小数点形式の short 表示
octave:14> x
x = 3.14e+00
octave:15> format
```

複素数も扱え，虚数単位 $\sqrt{-1}$ として

```
i, j, I, J
```

が使える．これらの記号がすでに変数として使われている場合，変数の定義が優先される．

```
octave:16> y = 1+2i
y = 1 + 2i
octave:17> z = i
z = 0 + 1i
octave:18> z*z
ans = -1
```

書式付で出力するにはC言語のスタイルが使える．

```
octave:19> printf("hello, Octave\n");
hello, Octave
octave:20> x = pi;
octave:21> printf("variable x = %f\n",x);
variable x = 3.141593
octave:22> y = 1 + 2i;
octave:23> printf("complex y = %f + %fi\n",real(y),imag(y));
complex y = 1.000000 + 2.000000i
```

1.3 行列の入出力

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

を入力して Octave Window に出力しよう．

```
octave:1> A = [1 2 3;4 5 6;7 8 9]
A =
  1  2  3
  4  5  6
  7  8  9
```

行列要素の入力方法に注意されたい．行ごとに入力し，‘;’ で次の行となる．Octave では配列を定義する必要がなく，必要な数だけ自動的に確保される．

次に，これを M ファイルで実行してみよう．まず，適当なエディタを開き，プログラムを入力する．プログラムは以下のようである．

ファイル名　file1.m

```
A = input('Enter matrix A ')
```

このファイルを例えば file1.m として Octave の作業ディレクトリ (pwd で表示されるディレクトリ) に保存しよう . そして , Octave Window に戻りつぎのように入力する

```
octave:1> file1
Enter matrix A [1 2 3;4 5 6;7 8 9]
A =
  1  2  3
  4  5  6
  7  8  9
```

A がスカラーのとき , '[ ]' を省略できる . 行列をプログラムの中に書く場合

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

とすればよい .

演習

1. 次の行列を入力してみよう .

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \quad \boldsymbol{C} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$$

1.4　行列の和・差・積・転置・べき乗

$\boldsymbol{A}$, $\boldsymbol{B}$ 行列の和・差は , 単に

```
X = A + B
Y = A - B
```

また , 積は

```
X = A*B
```

と書く . もちろん , $\boldsymbol{A}$, $\boldsymbol{B}$ の次元は , 各演算が定義できるように整合性を持つとするが , 一方がスカラーの場合は例外で行列 A , スカラー t に対して

```
X = A*t
```

という計算ができる . この場合 , A のすべての要素に t が掛けられる .

$\boldsymbol{A}$ の共役転置行列は

```
X = A'
```

で求まる．また，$\boldsymbol{A}$ の k 乗 (k は任意の実数でよい) は

```
X = A^k
```

で計算される．

例題 **1.1** $\boldsymbol{A}, \boldsymbol{B}$ を入力して，$\boldsymbol{A}+\boldsymbol{B}$, $\boldsymbol{A}-\boldsymbol{B}$ を計算するプログラムを作成せよ．また，つぎの行列について，具体的に計算してみよ．

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 & -1 & 1 \\ 2 & 2 & -1 \\ -1 & 2 & 1 \end{bmatrix}$$

(解答例)

```
A = input('Enter matrix A ')
B = input('Enter matrix B ')
X = A + B
Y = A - B
```

(実行結果)

```
Enter matrix A [1 2 3; 4 5 6; 7 8 9]
A =
  1  2  3
  4  5  6
  7  8  9
Enter matrix B [1 -1 1; 2 2 -1; -1 2 1]
B =
   1  -1   1
   2   2  -1
  -1   2   1

X =
   2   1   4
   6   7   5
   6  10  10

Y =
  0  3  2
  2  3  7
  8  6  8
```

演習

1. $\boldsymbol{A}$, $\boldsymbol{B}$ を入力して，$\boldsymbol{AB}$ を計算するプログラムを作成せよ．そして，つぎの場合を計算してみよ．

$$(1)\quad \boldsymbol{A}=\begin{bmatrix}1&1&1&1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1\\2\\3\\4\end{bmatrix}\qquad (\text{答})\boldsymbol{AB}=10$$

$$(2)\quad \boldsymbol{A}=\begin{bmatrix}1&2&3&4\\5&6&7&8\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1&1\\2&-2\\3&-3\\4&4\end{bmatrix}\qquad (\text{答})\boldsymbol{AB}=\begin{bmatrix}30&4\\70&4\end{bmatrix}$$

$$(3)\quad \boldsymbol{A}=\begin{bmatrix}1&1&1\\0&1&1\\0&0&1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1\\2\\3\end{bmatrix}\qquad (\text{答})\boldsymbol{AB}=\begin{bmatrix}6\\5\\3\end{bmatrix}$$

2. 次の $\boldsymbol{A}$, k に対する $\boldsymbol{A}^k$ を計算してみよ．

$$(1)\quad \boldsymbol{A}=\begin{bmatrix}2&1\\-1&2\end{bmatrix},\quad k=5\qquad (\text{答})\begin{bmatrix}-38&41\\-41&-38\end{bmatrix}$$

$$(2)\quad \boldsymbol{A}=\begin{bmatrix}2&1\\-1&2\end{bmatrix},\quad k=-2\qquad (\text{答})\begin{bmatrix}0.12&-0.16\\0.16&0.12\end{bmatrix}$$

1.5　行列関数

正則行列 $\boldsymbol{A}$ の逆行列は，単に

```
X = inv(A)
```

で求まる．Octave には，このような便利な関数が多数用意されている．よく使用される関数を表 1.1 に示す．関数の使用法の詳細については，必要に応じて，help コマンドで，例えば

```
octave:1> help inv
```

と打って調べるか，オンラインマニュアル(http://www.octave.org/doc/octave_toc.html)を参照する．ちなみに、線形方程式

$$\boldsymbol{Ax}=\boldsymbol{b}$$

の解は

表 1.1: 代表的な行列関数

関数	意味
norm	行列またはベクトルノルム
rank	行列のランク（階数）
det	行列式
inv	逆行列
eig	固有値と固有ベクトル
expm	行列指数関数

```
x = A\b
```

で計算できる．

例題 **1.2** 固有値問題

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & -1 \\ 4 & 2 & 0 \end{bmatrix}$$

を解け．
（解答）

```
octave:1> A = [1 2 1;2 1 -1;4 2 0]
A =
  1   2   1
  2   1  -1
  4   2   0
octave:2> [V,lambda] = eig(A)
V =
  -0.57735   0.54944   0.40825   % V = [v1 v2 ... vn]
   0.57735   0.13736  -0.40825
   0.57735   0.82416   0.81650
lambda =
  -2.00000   0.00000   0.00000   % lambda の対角要素は固有値
   0.00000   3.00000   0.00000
   0.00000   0.00000   1.00000
octave:3> lambda = eig(A)        % 固有値のみ計算
lambda =
  -2.00000
   3.00000
   1.00000
```

演習

1. 次の行列について，逆行列と行列式を計算せよ（逆行列 $\boldsymbol{X}$ の検算には $\boldsymbol{XA} = \boldsymbol{I}$ が利用できる）．

$$(1)\quad \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 0 & 1 & 3 & 2 \\ -1 & 1 & 2 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$（答）\boldsymbol{X} = \begin{bmatrix} 0.2 & 0.4 & -0.8 & 0.2 \\ 0.5 & -0.5 & 0.5 & 0 \\ -0.1 & 0.3 & -0.1 & 0.4 \\ -0.1 & 0.3 & -0.1 & -0.6 \end{bmatrix}, \quad |\boldsymbol{A}| = 10$$

$$\boldsymbol{A} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & 0 \end{bmatrix} \quad （答）\boldsymbol{X} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \end{bmatrix}, \quad |\boldsymbol{A}| = -1$$

2. 次の行列のランクを求めよ．

$$(1)\quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad （答）2 \quad (2)\quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad （答）2$$

$$(3)\quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix} \quad （答）2$$

$$(4)\quad \boldsymbol{A} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix} \quad （答）3$$

$$(5)\quad \boldsymbol{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix} \quad （答）4$$

3. 次の $\boldsymbol{A}$ について，e^A を計算せよ．

$$(1)\quad \boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \quad （答）e^A = \begin{bmatrix} 0.65970 & 0.53351 \\ -0.53351 & 0.12619 \end{bmatrix}$$

$$(2)\quad \boldsymbol{A} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \quad （答）e^A = \begin{bmatrix} 0.36788 & 0.36788 & 0.18394 \\ 0 & 0.36788 & 0.36788 \\ 0 & 0 & 0.36788 \end{bmatrix}$$

1.6 単位行列・零行列・対角行列

n 次単位行列 $\boldsymbol{X} = \boldsymbol{I}(n \times n)$ を作る場合

```
X = eye(n,n)
```

また，零行列 $\mathbf{0}(n \times m)$ の場合

```
zeros(n,m)
```

とする．同様に，すべての要素が 1 の $n \times m$ 行列は

```
ones(n,m)
```

である．

対角行列

$$\boldsymbol{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

の場合

```
v=[1 9 4 4 2];
D=diag(v,0)
```

と書く．

1.7 条件判定・繰り返し

1.7.1 if, else, elseif

if の使い方を例題によって説明しよう．

例題 **1.3** a, b を入力して，$b \neq 0$ ならば

$$y = a/b \tag{1.1}$$

を出力するプログラムを作成せよ．

(解答例)

```
a = input('Enter a ');
b = input('Enter b ');
if b ~= 0
   y = a/b
endif
```

例題 **1.4** x を入力して

$$y = \begin{cases} 1 & \text{if} \quad x \geq 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \tag{1.2}$$

を出力するプログラムを作成せよ．

(解答例)

```
x=input('Enter x ');
if x >= 0
   y = 1;
else
   y = -1;
endif
y
```

例題 **1.5** x を入力して

$$y = \begin{cases} 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \tag{1.3}$$

を出力するプログラムを作成せよ．

(解答例)

```
x=input('Enter x ');
if x > 0
   y=1;
elseif x<0
   y=-1;
else
   y=0;
endif
y
```

`if` ループは何重にもできる．また，`if` の中で `elseif` を何回でも使える．`if` や `while` で使われる関係演算子を表 1.2 に示す．

表 1.2: 関係演算子

演算子	意味
<	$<$
<=	$\leq$
>	$>$
>=	$\geq$
==	$=$
~=	$\neq$

1.7.2 switch

例題 **1.6** n を入力し

$n=-1$	ならば	`minus one`
$n=0$	ならば	`zero`
$n=1$	ならば	`one`
その他の場合		`other value`

を出力するプログラムを作成せよ．

(解答例)

```
n=input('Enter n ');
switch n
case -1
   disp('minus one');
case 0
   disp('zero');
case 1
   disp('one');
otherwise
   disp('other value');
endswitch
```

`switch` は次のような使い方もできる．

```
n=input('Enter n ');
switch n
case 1
   disp('1');
case {2,3,4}
```

```
   disp('2 or 3 or 4');
case 5
   disp('5');
otherwise
   disp('other value');
endswitch
```

1.7.3　while

while は次の構文で使う．

```
while  条件
   statements
endwhile
```

while は条件が成立する限り，*statements* を実行する．while ループは break でいつでも中断できる．

例題 **1.7** 与えられた正数 a を $\epsilon = 0.001$ よりも小さくなるまで 2 で割り続けるプログラムを作成せよ．

(解答例)

```
eps=0.001;
a=input('Enter a ');
while a>=eps
   a=a*0.5
endwhile
```

1.7.4　for

for は一般に次の構文で使われる．

```
for index = start : increment : end
    statements
endfor
```

index を増分 *increment* で変えながら，*statements* を *end* まで実行する．増分は負でもよい．増分を省略すると $increment = 1$ となる．

例題 **1.8** n は正整数とする．1 から n までの和を求めるプログラムを作成せよ．

(解答例)

```
n=input('Enter n ');
s = 0;
for i = 1:n
   s = s + i;
endfor
s
```

for も break によっていつでも中断できる．

C 言語の goto に相当する文は Octave にはないが，本節で述べた構文を組み合わせることによって，任意のプログラムを記述できる．

MATLAB 互換のプログラムを書きたい場合 ,endif ,endswitch ,endwhile ,endfor の代わりに end を使う．

演習

1. 実数 x と正整数 n が与えられたとき

   $$y = 1 + x + x^2 + \cdots + x^n$$

   を計算するプログラムを作成せよ．
   （ヒント）次のプログラムが使える．

   ```
   a = 1;
   y = 1;
   for i = 1:n
     a = a*x;
     y = y + a;
   endfor
   ```

2. 実数 $x(|x| < 1)$ と正数 ϵ が与えられたとき

   $$y = 1 + x + x^2 + \cdots$$

   を計算するプログラムを作成せよ．ただし，終了基準は

   $$|x^k| < \epsilon$$

   とする．

3. 正整数 n が与えられたとき，n の階乗

   $$n! = 1 \cdot 2 \cdot 3 \cdots n$$

   を計算するプログラムを作成せよ．
   （ヒント）次のプログラムが使える．

```
y = 1;
for i = 1:n
  y = y*i;
endfor
```

または

```
x = 1:n;
y = prod(x)        % x の要素の積を計算する関数
```

でもよい．

4. 正整数 n, $r(n \geq r)$ が与えられたとき，組み合わせの数

$$ {}_nC_r = \frac{n(n-1)(n-2)\cdots(n-r+1)}{1\cdot 2\cdot 3\cdots r} $$

を計算するプログラムを作成せよ．

5. 実数 x に対して，次の級数を計算するプログラムを作成せよ．ただし，ϵ を与えた正数とするとき，新たに加える項の絶対値が ϵ より小さくなるまで計算するものとする．

(1) $y = 1 + x + x^2/2! + x^3/3! + \cdots$ （$x = 2$ のとき $y \to 7.38906$）

(2) $y = 1 - x^2/2! + x^4/4! - + \cdots$ （$x = 2$ のとき $y \to -0.41615$）

(3) $y = x - x^3/3! + x^5/5! - + \cdots$ （$x = 2$ のとき $y \to 0.90930$）

6. 次の級数を計算するプログラムを作成せよ．ただし，ϵ を与えた正数とするとき，新たに加える項の絶対値が ϵ より小さくなるまで計算するものとする．

(1) $y = \dfrac{1}{1\cdot 2} + \dfrac{1}{2\cdot 3} + \dfrac{1}{3\cdot 4} + \cdots \quad (= 1)$

(2) $y = \dfrac{1\cdot 2}{1\cdot 3} + \dfrac{1\cdot 2\cdot 3}{1\cdot 3\cdot 5} + \dfrac{1\cdot 2\cdot 3\cdot 4}{1\cdot 3\cdot 5\cdot 7} + \cdots \quad \left(= \dfrac{\pi}{2}\right)$

1.8　種々の行列の作成法

1.8.1　行列要素の指定

行列の要素は

```
x(i)      A(i,j)
```

などと指定する．よって，n を正整数とすると

$$ \boldsymbol{x} = \begin{bmatrix} 1 & 2 & \cdots & n \end{bmatrix} $$

を生成するプログラムは

```
x = [];
for i=1:n
  x(i) = i;
endfor
x = x'
```

と書ける．また

```
x = 1:n
```

と書くこともできる．後者の方が Octave らしい書き方で，前者に比べ計算速度も速い．

ある区間を分割した点をベクトルとして与える以下の関数がある．これらは，関数や微分方程式の解を計算する座標点を与える際によく利用される．

```
octave:1> x = linspace(0,1,5)  % [0, 1] を (5-1) 等分する．
x =
  0.00000  0.25000  0.50000  0.75000  1.00000
octave:2> y = logspace(-1,1,5)
                % [1e-1, 1e+1] を対数的に (5-1) 等分する．
y =
  0.10000  0.31623  1.00000  3.16228  10.00000
```

今度は n 次 Jordan 細胞 $\boldsymbol{J}_n(\lambda)$ を作ってみよう．Jordan 細胞は例えば

$$\boldsymbol{J}_3(\lambda) = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix} \quad \boldsymbol{J}_4(\lambda) = \begin{bmatrix} \lambda & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & \lambda \end{bmatrix}$$

という規則的な形をしている．

```
n=input('Enter n ');
lambda=input('Enter lambda ');
J=eye(n,n)*lambda;
for i=1:n-1
  J(i,i+1) = 1;
endfor
J
```

1.8.2 行列のサイズ

いま，プログラム中で $\boldsymbol{A}$ 行列が与えられていて，このサイズ (行と列の数) を知りたいとき

```
d=size(A)
```

とする．すると，$\boldsymbol{A}$ が 3×4 行列の場合，答えが

```
d =
  3  4
```

となる．すなわち，d(1)=3 d(2)=4 という意味である．
$\boldsymbol{A}$ と同じサイズの零行列 $\boldsymbol{Z}$ を作る場合

```
Z=zeros(size(A))
```

とする．同様に，$\boldsymbol{A}$ とサイズが同じで，要素がすべて 1 の行列 $\boldsymbol{S}$ は

```
S=ones(size(A))
```

で作れる．

1.8.3　行列の結合と削除

$\boldsymbol{A}$ に $\boldsymbol{B}$ を追加して新しい行列 $\boldsymbol{C}$ を作る場合以下のようにする．

$$\boldsymbol{C} = \left[\begin{array}{cc} \boldsymbol{A} & \boldsymbol{B} \end{array} \right]$$

の場合

```
C = [A B]
```

また

$$\boldsymbol{C} = \left[\begin{array}{c} \boldsymbol{A} \\ \boldsymbol{B} \end{array} \right]$$

の場合

```
C = [A;B]
```

と書く．
$\boldsymbol{A}$ の第 i 列を削除した行列 $\boldsymbol{X}$ を作る場合

```
X = A;
X(:,i)=[]
```

同様に，$\boldsymbol{A}$ の第 i 行を削除した行列 $\boldsymbol{X}$ を作る場合

```
X = A;
X(i,:)=[]
```

とする．また，$\boldsymbol{A}$ の第 i 列から第 j 列までを削除した行列 $\boldsymbol{X}$ を作る場合

```
X = A;
X(:,i:j)=[]
```

とする．

例題 1.9 動的システム

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{1.4}$$

ただし

$$\boldsymbol{A}(n \times n)\ ,\ \boldsymbol{B}(n \times r)$$

の可制御性行列は次式で定義される．

$$\boldsymbol{U}_c := [\ \boldsymbol{B} \quad \boldsymbol{A}\boldsymbol{B} \quad \cdots \quad \boldsymbol{A}^{n-1}\boldsymbol{B}\ ] \tag{1.5}$$

この行列を作るプログラムを作成せよ．そして，次の $(\boldsymbol{A}, \boldsymbol{B})$ に対する可制御性行列を計算してみよ．

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

(解答例)

```
n=input('Enter n ');
A=input('Enter A ');
B=input('Enter B ');
Uc=B;
X=B;
for i=1:n-1
   X=A*X;
   Uc=[Uc X];
endfor
Uc
```

(実行例)

```
Enter n 3
Enter A [0 1 0; 0 0 1; 1 0 0]
Enter B [1;0;1]
Uc =
  1  0  1
  0  1  1
  1  1  0
```

演習

1. 次の行列を任意の正整数 n に対して作るプログラムを書いてみよう．

$$\boldsymbol{R}_1 = 1 \quad \boldsymbol{R}_2 = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \quad \boldsymbol{R}_3 = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix} \quad \cdots$$

2. $\boldsymbol{v} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$ とする．

```
diag(v,1), diag(v,-1), diag(v,2), diag(v,-2)
```

がどのような行列になるか調べてみよう．

3. 任意の正整数 n について，対角行列

$$\boldsymbol{D} = \begin{bmatrix} 1 & & & \\ & 2 & & \boldsymbol{0} \\ & & \ddots & \\ \boldsymbol{0} & & & n \end{bmatrix}$$

を作るプログラムを作成せよ．
（ヒント）

```
n = input('Enter n ');
v = 1:n;
D = diag(v,0)
```

4. 動的システム

$$\left.\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) \end{aligned}\right\} \tag{1.6}$$

ただし

$\boldsymbol{A}(n \times n)$，$\boldsymbol{C}(m \times n)$

の可観測性行列

$$\boldsymbol{U}_o := \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{n-1} \end{bmatrix} \tag{1.7}$$

を作るプログラムを作成せよ．そして，次の $(\boldsymbol{A}, \boldsymbol{C})$ に対する可観測性行列を計算してみよ．

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad （答）\boldsymbol{U}_o = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5. 動的システム

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}(t) & = & \boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) & = & \boldsymbol{C}\boldsymbol{x}(t) \end{array}\right\} \tag{1.8}$$

の次数を n とするとき，可制御性条件および可観測性条件は

$$\mathrm{rank}\,\boldsymbol{U}_c = n, \quad \mathrm{rank}\,\boldsymbol{U}_o = n \tag{1.9}$$

である．可制御性・可観測性を判定するプログラムを作成せよ．そして，次のシステムの可制御性・可観測性を判定してみよ．

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

(答)$\mathrm{rank}\,\boldsymbol{U}_c = 3, \quad \mathrm{rank}\,\boldsymbol{U}_o = 4$ したがって，不可制御・可観測である．

6. $\boldsymbol{A}(n\times n)$, $\boldsymbol{x}(n\times 1)$ とする．差分方程式

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad k = 0, 1, \ldots$$

の解を $k = 0, 1, \ldots, t_f$ について求めるプログラムを作成せよ．そして，次の $\boldsymbol{A}$, $\boldsymbol{x}_0$, t_f について実行してみよ．

$$\boldsymbol{A} = \begin{bmatrix} 0.8 & 1 \\ 0 & 0.8 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad t_f = 20$$

(ヒント)

```
x = input('Enter x ');
tf = input('Enter tf ');
A = [0.8 1; 0 0.8]
y = [];
for i = 1:tf+1
   y(:,i) = x;
   x = A*x;
endfor
y
```

7. 差分方程式

$$y(k+2) + a_1 y(k+1) + a_0 y(k) = 0, \quad y(0) = y_0, \quad y(1) = y_1, \quad k = 0, 1, \ldots$$

の解 $y(k)$, $k = 0, 1, \ldots, t_f$ を求めるプログラムを作成せよ．そして，次のデータについて実行してみよ．

$$a_0 = 0.25, \quad a_1 = -1, \quad y(0) = 1, \quad y(1) = 1, \quad t_f = 20$$

8. n を正整数とする．フィボナッチ数列

   $1, 1, 2, 3, 5, 8, \ldots$

   (第 1 項 = 第 2 項 = 1 とし，第 i 項を第 $i-2$,$i-1$ 項の和により求める)

   を第 n まで作るプログラムを作成せよ．

9. 実数 x と正整数 n が与えられたとき，ベクトル

   $$\boldsymbol{y} = \left[\begin{array}{ccccc} 1 & x & x^2 & \ldots & x^n \end{array} \right]$$

   を作るプログラムを作成せよ．

10. [0, 1] の範囲で n 個の乱数 (実数)

    $x_1, x_2, x_3, \ldots, x_n$

    を発生させ，この内の最大値と最小値を求めよ．そして，小さい順に並べ替えよ．
    (ヒント)

```
n=input('Enter n ');
x = rand(1,n)          % 1 × n の一様分布乱数行列を発生させる関数
xmax = max(x)          % 最大値を求める関数
xmin = min(x)          % 最小値を求める関数
y = sort(x)            % 小さい順に並べ替える関数
```

11. [0, 1] の範囲で n 個の乱数 (実数)

    $x_1, x_2, x_3, \ldots, x_n$

    を発生させ，大きい順に並べ替えよ．
    (ヒント) 大きい順に並べ替える関数は用意されていないので，やはり `sort` を利用する．

1.9　数学関数

sin, cos などの数学関数が一通り用意されている．また，このような関数はベクトル計算もできる．例えば，$\sin(t)$ を種々の t に対して計算する場合

```
y = [];
i = 0;
for t = 0:0.1:10
    i = i + 1;
    y(i) = sin(t);
endfor
y = y'
```

表 1.3: 代表的数学関数

関数	意味
sin(x)	$\sin x$
cos(x)	$\cos x$
tan(x)	$\tan x$
asin(x)	$\arcsin x$
acos(x)	$\arccos x$
atan(x)	$\arctan x$
atan2(y,x)	4 象限逆正接
sinh(x)	$\sinh x$
cosh(x)	$\cosh x$
tanh(x)	$\tanh x$
exp(x)	e^x
log(x)	$\ln x$
log10(x)	$\log_{10} x$
sqrt(x)	$\sqrt{x}$
abs(x)	$\|x\|$
real(x)	複素数の実部
imag(x)	複素数の虚数部
sign(x)	符号関数

の代わりに

```
t = 0:0.1:10;
y = sin(t);
```

とできる．後者の方が Octave に適した書き方で，計算速度も前者より速い．ちなみに，後者のプログラムで得た t，y をグラフ表示するには

```
grid "on"
plot(t,y)
```

とする (図 1.1)．代表的な数学関数を表 1.3 に示す．

例題 **1.10** 関数

$$y(t) = e^{-0.5t} \cos 5t \tag{1.10}$$

のグラフを描け．ただし，t の範囲を $[0, 10]$，プロット点の刻み幅を 0.01 とする．

(解答例)

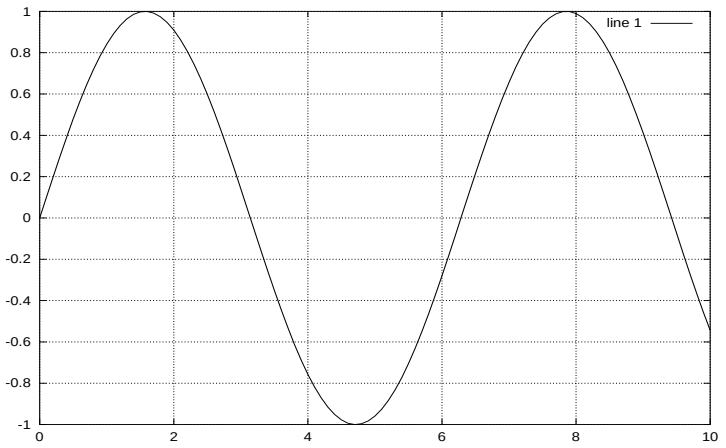

図 1.1: $y = \sin t$ のグラフ

```
t = 0:0.01:10;
y = exp(-0.5*t).*cos(5*t);
grid "on"
plot(t,y)
```

‘.*’ は要素毎の掛け算を表す．グラフを図 1.2 に示す．

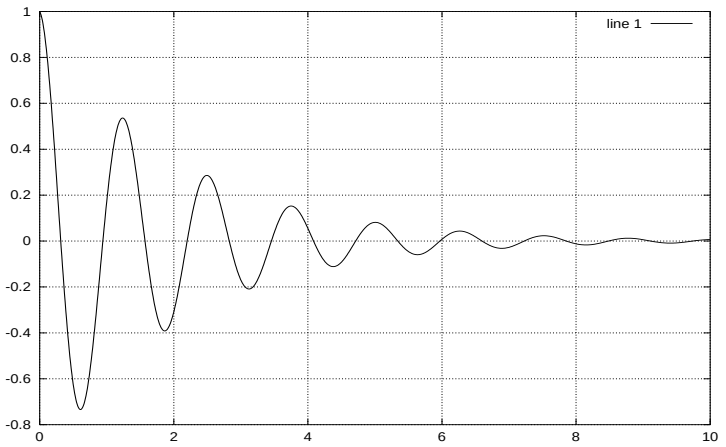

図 1.2: $y = e^{-0.5t} \cos 5t$ のグラフ

例題 1.11 図 1.3 に示す関数は

$$f(x) = \frac{4k}{\pi} \left(\sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x + \cdots \right) \tag{1.11}$$

とフーリエ級数展開される．この級数を第 n 項まで計算した結果をグラフ表示せよ．ただし，グラフ表示の範囲を $[-10, 10]$，プロット点の刻み幅を 0.02 とする．

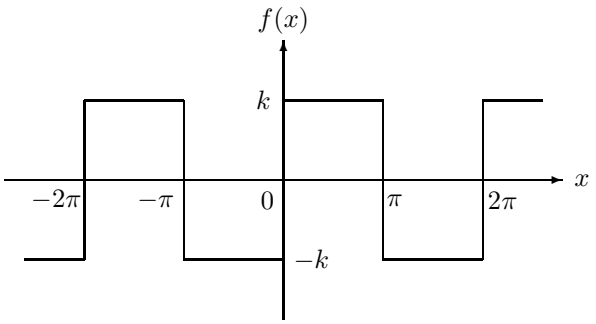

図 1.3: 矩形波関数

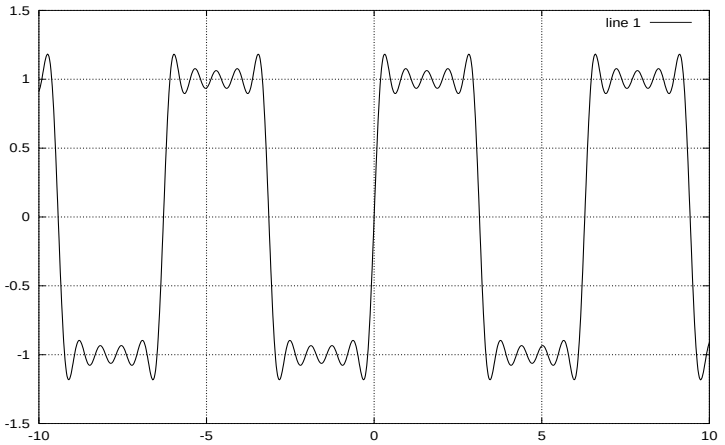

図 1.4: 例題 1.11 $k = 1$, $n = 5$ の場合

（解答例）

```
k = input('Enter k ');
n = input('Enter n ');
x = -10:0.02:10;
y = zeros(size(x));
a = 1;
for i=1:n
   y = y + sin(a*x)/a;
   a = a + 2;
endfor
```

```
y = y*4*k/pi;
grid "on"
plot(x,y)
```

演習

1. 図 1.5 に示す関数は

$$f(x) = \frac{\pi}{2} - \frac{4}{\pi}\left(\cos x + \frac{1}{3^2}\cos 3x + \frac{1}{5^2}\cos 5x + \cdots\right) \tag{1.12}$$

とフーリエ級数展開される．この級数を第 n 項まで計算した結果をグラフ表示せよ．ただし，グラフ表示の範囲を $[-10, 10]$，プロット点の刻み幅を 0.02 とする．

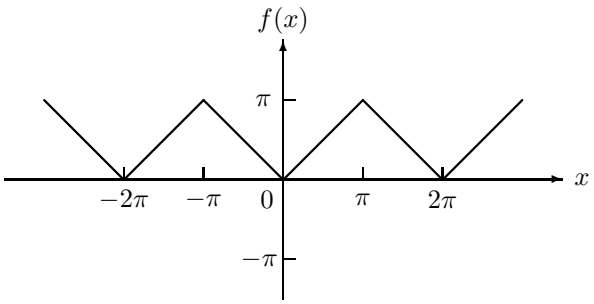

図 1.5: 三角波関数

1.10　多項式に関する関数

多項式に関する代表的な関数を表 1.4 に示す．

1.10.1　多項式の表現，零点，特性方程式

Octave では，多項式を降べき順に係数を並べたベクトルとして表現する．すなわち，多項式

$$p(x) = -x^4 + 20x^2 - 20x + 5 \tag{1.13}$$

は

```
octave:1> p = [-1 0 20 -20 5]
```

で表される．

この多項式の零点 (多項式 $= 0$ の根) は `roots` を用いて次のように求められる．

表 1.4: 多項式に関する関数

関数	説明
roots	根を求める
poly	指定根を与える多項式
poly	正方行列に対する特性多項式
polyval	多項式の値
polyvalm	行列多項式の値
polyfit	多項式によるデータの最小二乗内挿
conv	多項式の掛け算
deconv	多項式の割り算
residue	有理関数の部分分数展開

```
octave:2> r = roots(p)
r =
  -4.92606
   3.89858
   0.57356
   0.45393
```

逆に，零点から多項式の係数を求めるには

```
octave:3> p1 = poly(r)
p1 =
    1.0000e+00   -2.2760e-15  -2.0000e+01   2.0000e+01   -5.0000e+00
```

とする．poly は，正方行列 $\boldsymbol{A}$ に対する特性多項式を求めるときにも使用できる．例えば

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 2 \end{bmatrix}$$

の場合

```
octave:4> A = [0 1 0; -1 0 0; 0 -1 2];
octave:5> p2 = poly(A)
p2 =
   1  -2  1  -2
```

と計算できる．すなわち，特性多項式は

$$p_2(x) = |x\boldsymbol{I} - \boldsymbol{A}| = x^3 - 2x^2 + x - 2$$

である．

$p(x)$ の $x = 1$ における値は，polyval を用いて

```
octave:6> y = polyval(p,1)
y = 4
```

と計算できる．

例題 1.12 polyval 関数を使って $p(x) = -x^4 + 20x^2 - 20x + 5$ のグラフを描いてみよう．ただし，x の範囲を $[-5, 5]$，プロット点の刻み幅を 0.02 とせよ．

(解答例)

```
p =[ -1 0 20 -20 5];
x = -5:0.02:5;
y = polyval(p,x);
grid "on"
plot(x,y)
```

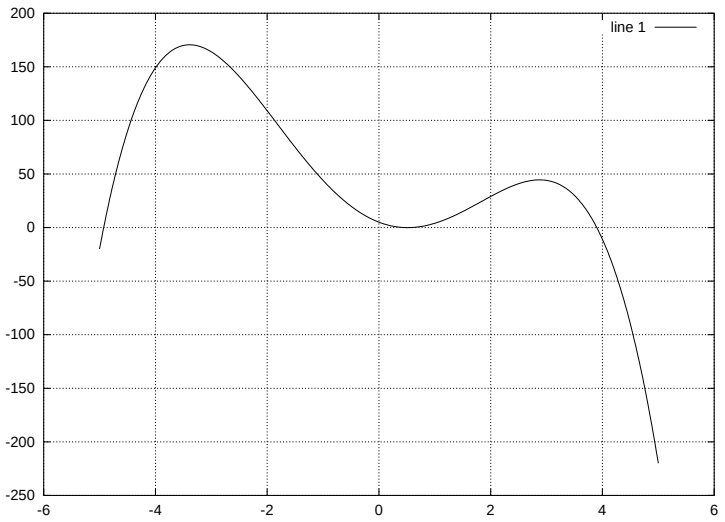

図 1.6: $p(x) = -x^4 + 20x^2 - 20x + 5$ のグラフ

1.10.2　多項式曲線のあてはめ

polyfit は，平面上の与えられたデータ点の集合を n 次多項式曲線で最小二乗近似する関数である．次のデータを考えよう．

$$\boldsymbol{x} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

$$\boldsymbol{y} = \begin{bmatrix} 0 & 3 & 4 & 10 & 18 & 27 & 41 & 59 & 79 & 93 & 101 \end{bmatrix}$$

このデータを n 次多項式曲線で近似するプログラムは次のように書ける．

```
n = input('Enter n ');
x = 0:10;
y = [0 3 4 10 18 27 41 59 79 93 101];
p = polyfit(x,y,n)
x1 = 0:0.1:10;
y1 = polyval(p,x1);
grid "on"
plot(x,y,'+',x1,y1)     % (x,y) データを+で表示 ( 他に , o,*,x など )
                        % (x1,y1) データを線で結ぶ
```

$n = 3$ とした場合の実行例を図 1.7 に示す．

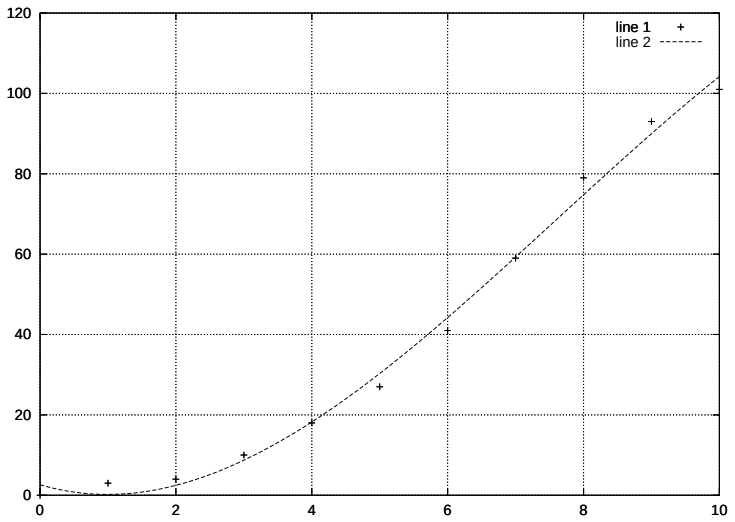

図 1.7: データ点に対する多項式曲線のあてはめ

1.11 ユーザ定義関数の書き方

ユーザ定義関数の書き方を例題で見てみよう．

例題 **1.13** n 個のデータ $x_1, x_2, \cdots, x_n$ の平均 $\bar{x}$ と分散 σ^2 は次式で求められる．

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1.14}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{1.15}$$

ベクトル $\boldsymbol{x}$ を

$$\boldsymbol{x} := \left[\begin{array}{cccc} x_1 & x_2 & \cdots & x_n \end{array}\right]$$

と定義する．$\boldsymbol{x}$ を入力して，$\bar{x}$, σ^2 を出力する関数を作成せよ．

(解答例)

ファイル名：heikin.m

```
function [a, b] = heikin(x)
% 平均と分散を求める関数　a:平均　b:分散
n = length(x);  % x の要素数
a = sum(x)/n;   % sum(x):要素の総和
b = sum((x-a).^2)/n;
endfunction
```

この関数を呼ぶプログラムの例を次に示す．

```
octave:1> x = rand(1,5)
x =
  0.70536  0.31711  0.52546  0.80152  0.14361
octave:2> [mean,sigma2] = heikin(x)
mean = 0.49861
sigma2 = 0.058838
```

引数 (複数の場合コンマで区切る) を関数に渡し，戻り値 (複数の場合コンマで区切る) を関数から得る．引数と戻り値以外の関数中の変数はメインプログラムや他の関数プログラムに対して独立している．もちろん，引数や戻り値は行列でもよい．

関数名も変数名と同様，英字で始め，他に数字やアンダーバーの記号が使える．ファイル名は

関数名.m

とする．

function 後のコメントは help コマンドで参照される．

関数をファイルに保存して，M ファイルとして使用する場合，MATLAB との互換性を考慮して，endfunction を省略できる．

1.12 ファイルに対する入出力

数値などのファイルへの出力には fprintf，ファイルからの入力には fscanf 関数を使う．引数の書き方は C 言語に従う．ベクトルを扱えるところが C 言語と大きく違う点である．以下に例を示す．

```
octave:1> x = 0:0.1:1;
octave:2> y = [x; exp(x)];
octave:3> fid = fopen("exp.dat','w');
octave:4> fprintf(fid,'%6.2f %12.8f \n',y);
octave:5> fclose(fid);
```

以上を実行すると，カレントディレクトリに次の内容のファイル exp.dat が出力される．

```
 0.00   1.00
 0.10   1.11
 0.20   1.22
 0.30   1.35
 0.40   1.49
 0.50   1.65
 0.60   1.82
 0.70   2.01
 0.80   2.23
 0.90   2.46
 1.00   2.72
```

exp.dat を変数名 a で読み込む場合，次のようにする．

```
octave:6> fid = fopen("exp.dat");
octave:7> a = fscanf(fid,'%f %f',[2 inf]);   % It has two rows now.
octave:8> a = a';
octave:9> fclose(fid);
```

読み込み専用ファイルでオープンするには fopen で"r"を指定してもよいが，これはデフォルトなので省略することもできる．[2 inf] は，「2 次元配列でファイルの最後まで読み込め」というオプションである．

第2章　シミュレーションの基礎

本章では，オイラー法またはルンゲクッタ法を用いて常微分方程式を解くプログラムの作り方を紹介する．このような基本的なプログラミングを知っておけば，制御系を計算機制御する際の計算機制御用プログラムを作るときにも役立つ．また，微分方程式の数値解法の学習にもなる．ちなみに，Octave には `lsode` という常微分方程式を解く関数が用意されており，`lsode` 関数を用いて同様のプログラムを書くことができる．

2.1　オイラー法

次の連立一階常微分方程式（状態方程式）を考える．

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{2.1}$$

初期状態 $\boldsymbol{x}_0$ からの (2.1) 式の解（状態軌道）を求める問題を初期値問題という．

オイラー法とは，(2.1) 式を差分近似，すなわち，無限小 dt を有限の小さな正数 Δt で置き換えて，$\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$ から Δt のちの状態 $\boldsymbol{x}(t+\Delta t)$ を

$$\boldsymbol{x}(t+\Delta t) \simeq \boldsymbol{x}(t) + (\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t))\Delta t \tag{2.2}$$

と近似計算する方法である．すなわち，オイラー法による近似は，$\boldsymbol{x}(t+\Delta t)$ を t のまわりでテイラー級数展開して，Δt の 1 次項までを考慮することに相当する．オイラー法はプログラムが容易であるが，刻み幅 Δt をある程度小さく与える必要がある．(2.1) 式をオイラー法を用いて解くプログラムの例を以下に示す．

例題 **2.1** 初期値問題 (2.1) をオイラー法で解くプログラムを作成せよ．そして，次の問題を解いてみよう．

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad \boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad u(t) = 1, \quad t \geq 0$$

ただし，刻み幅 $\Delta t = 0.05$，終了時刻 $t_f = 20$ とする．

（解答例）

ファイル名：`ex2_1.m`

```
A = [0 1; -1 -1];
```

```
B = [0;1];
u=1;
dt = 0.05;
tf=20;
x=[0;0];
xx = [];
i=0;
for t=0:dt:tf
   i=i+1;
   xx(:,i)=x;
   dx = A*x + B*u;
   x = x + dx*dt;
endfor
t=0:dt:tf;
grid "on"
plot(t,xx)
```

シミュレーション結果は plot によりグラフ表示される．

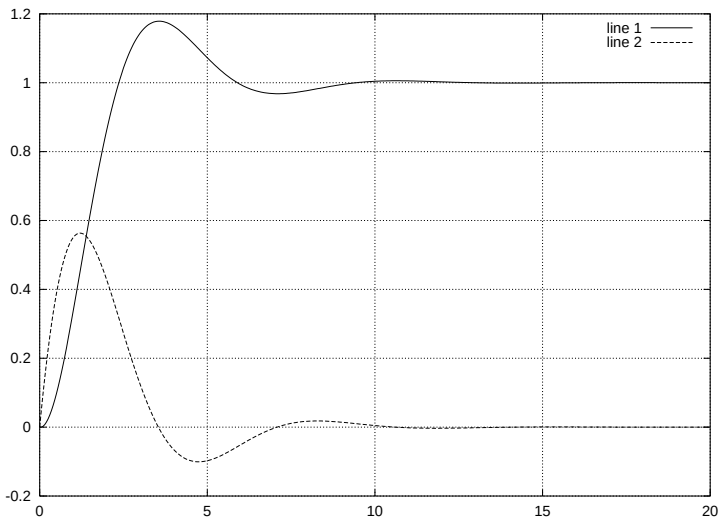

図 2.1: シミュレーション結果

演習

1. 次の初期値問題をオイラー法で解いてみよ．ただし，終了時刻 $t_f = 15$ とする．

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{x}(t), \quad \boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

2. 次の初期値問題をオイラー法で解いてみよ．ただし，終了時刻 $t_f = 20$ とする．刻み幅を種々変えてみよ．

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \boldsymbol{x}(t), \quad \boldsymbol{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

2.2 ルンゲクッタ法

以下で示す（4 次）ルンゲクッタ法は，$\boldsymbol{x}(t + \Delta t)$ を t のまわりでテイラー級数展開して，Δt の 4 次までの項を残したことに相当する近似法を用いる方法である．

一般的な連立一階微分方程式

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{2.3}$$

に対するルンゲクッタ法はつぎのように与えられる．

$$\boldsymbol{x}(t + \Delta t) \simeq \boldsymbol{x}(t) + \frac{\boldsymbol{d}_1 + 2\boldsymbol{d}_2 + 2\boldsymbol{d}_3 + \boldsymbol{d}_4}{6} \tag{2.4}$$

ただし

$$\left.\begin{array}{rcl} \boldsymbol{d}_1 &=& \boldsymbol{f}(\boldsymbol{x}(t))\Delta t \\ \boldsymbol{d}_2 &=& \boldsymbol{f}(\boldsymbol{x}(t) + \boldsymbol{d}_1/2)\Delta t \\ \boldsymbol{d}_3 &=& \boldsymbol{f}(\boldsymbol{x}(t) + \boldsymbol{d}_2/2)\Delta t \\ \boldsymbol{d}_4 &=& \boldsymbol{f}(\boldsymbol{x}(t) + \boldsymbol{d}_3)\Delta t \end{array}\right\} \tag{2.5}$$

ルンゲクッタ法はオイラー法に比べ格段に精度が良く，刻み幅をオイラー法よりも大きくとることができる．通常，シミュレーションでは，ルンゲクッタ法が使われる．

例題 **2.2** 初期値問題 (2.1) をルンゲクッタ法で解くプログラムを作成せよ．そして，次の問題を解いてみよ．

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad \boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad u(t) = 1, \quad t \geq 0$$

ただし，刻み幅 $\Delta t = 0.05$，終了時刻 $t_f = 20$ とする．

（解答例）

ファイル名：`ex2_2.m`

```
A = [0 1; -1 -1];
B = [0; 1];
u=1;
dt = 0.05;
tf=20;
x=[0 0]';
xx = [];
i=0;
for t=0:dt:tf
   i=i+1;
   xx(:,i)=x;
   xt = x;
   for j=1:4
      f = A*x + B*u;
      d(:,j) = f*dt;
      x = xt + d(:,j)*0.5;
      if j==3
         x = xt + d(:,j);
      endif
   endfor
   x = xt + (d(:,1) + d(:,2)*2 + d(:,3)*2 + d(:,4))/6;
endfor
t=0:dt:tf;
grid "on"
plot(t,xx)
```

シミュレーション結果は図 2.1 とほぼ同じである．

演習

1. 次の初期値問題をルンゲクッタ法で解け．ただし，終了時刻 $t_f = 15$ とする．また，オイラー法の結果と比較してみよ．

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{x}(t), \quad \boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

2. 次の初期値問題をルンゲクッタ法で解け．ただし，終了時刻 $t_f = 20$ とする．刻み幅を種々変えてみよ．また，オイラー法の結果と比較してみよ．

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \boldsymbol{x}(t), \quad \boldsymbol{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

2.3 伝達関数から状態方程式を作る方法

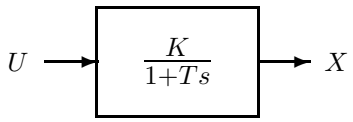

図 2.2: 1 次系の伝達関数

2.3.1 1 次系

伝達関数に対応する状態方程式を求めることを実現というが，まず，1 次系の伝達関数 (図 2.2) から実現の方法を説明しよう．ブロック線図から

$$\frac{X}{U} = \frac{K}{1+Ts} \tag{2.6}$$

これを変形すると

$$sX = \frac{1}{T}(KU - X) \tag{2.7}$$

となる．この表現を時間領域での表現に直すと，結局，(2.6) 式に対する状態方程式

$$\dot{x}(t) = \frac{1}{T}(Ku(t) - x(t)) \tag{2.8}$$

を得る．

2.3.2 2 次系

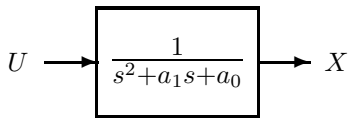

図 2.3: 2 次系の伝達関数

図 2.3 に示す分子が 1 の伝達関数の実現を考える．ブロック線図から

$$\frac{X}{U} = \frac{1}{s^2 + a_1 s + a_0} \tag{2.9}$$

これを変形して

$$s^2 X = -a_0 X - a_1 s X + U \tag{2.10}$$

を得る．さらに，これを時間領域表現すると

$$\ddot{x}(t) = -a_0 x(t) - a_1 \dot{x}(t) + u(t) \tag{2.11}$$

となる．さらに，状態方程式で表すため，状態を

$$x_1 := x, \quad x_2 := \dot{x} \tag{2.12}$$

と定義すると，(2.11) 式から

$$\left.\begin{array}{rcl} \dot{x}_1(t) & = & x_2(t) \\ \dot{x}_2(t) & = & -a_0 x_1(t) - a_1 x_2(t) + u(t) \end{array}\right\} \tag{2.13}$$

また

$$\boldsymbol{x} := \left[\begin{array}{c} x_1 \\ x_2 \end{array}\right] \tag{2.14}$$

を定義して，ベクトル表現すれば

$$\dot{\boldsymbol{x}}(t) = \left[\begin{array}{cc} 0 & 1 \\ -a_0 & -a_1 \end{array}\right] \boldsymbol{x}(t) + \left[\begin{array}{c} 0 \\ 1 \end{array}\right] u(t) \tag{2.15}$$

となる．

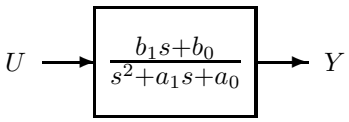

図 2.4: 一般的な 2 次系の伝達関数

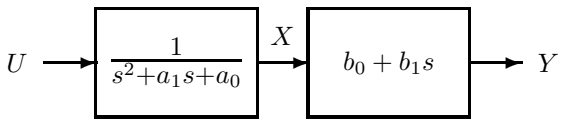

図 2.5: 一般的な 2 次系の伝達関数

上の議論を基にして，図 2.4 の一般的な 2 次系の実現を考えよう．このブロック線図は，図 2.5 のように等価変換できる．図 2.5 から

$$Y = (b_0 + b_1 s)X \tag{2.16}$$

すなわち

$$y(t) = b_0 x(t) + b_1 \dot{x}(t) \tag{2.17}$$

また，X/U の実現は (2.15) 式として求まっているので，結局，図 2.4 の伝達関数の実現は

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{2.18}$$

$$y(t) = \begin{bmatrix} b_0 & b_1 \end{bmatrix} \boldsymbol{x}(t) \tag{2.19}$$

となる．n 次系の伝達関数に対しても同様な方法によって状態方程式を得ることができる．

2.3.3 分母と分子の次数が等しい場合

$$G(s) = \frac{N(s)}{D(s)} \tag{2.20}$$

において，$D(s)$ と $N(s)$ の次数が等しい場合，割り算を実行して

$$G(s) = d + \frac{N'(s)}{D(s)} \tag{2.21}$$

と表す (図 2.6) . d は定数であり，$N'(s)$ の次数は $D(s)$ の次数よりも小さくなる．よっ

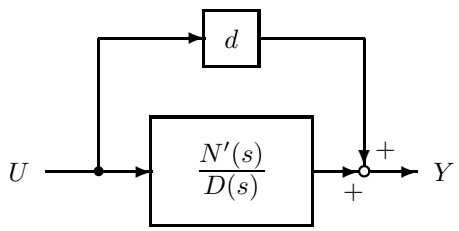

図 2.6: 分母と分子が同じ次数の伝達関数

て，$N'(s)/D(s)$ の実現を

$$\left.\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}u(t) \\ y'(t) &= \boldsymbol{c}\boldsymbol{x}(t) \end{aligned}\right\} \tag{2.22}$$

とすると，$G(s)$ の実現は

$$\left.\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}u(t) \\ y(t) &= \boldsymbol{c}\boldsymbol{x}(t) + du(t) \end{aligned}\right\} \tag{2.23}$$

となる．

演習

1. 次の伝達関数の実現を求めよ．

$$(1)\quad G(s)=\frac{s}{s^2+1} \qquad (2)\quad G(s)=\frac{1+T_1 s}{1+T_2 s}$$

$$(3)\quad G(s)=\frac{s^2+1}{s^3+2s^2+2s+1} \qquad (4)\quad G(s)=\frac{1}{(1+Ts)^2}$$

2. ルンゲクッタ法を用いて 1. の伝達関数のステップ応答をそれぞれ求めよ。ただし

$$T_1=5,\quad T_2=1,\quad T=1$$

とせよ．ステップ応答を求めるには

$$\boldsymbol{x}(0)=\mathbf{0},\quad u=1$$

とする。

2.4　位相面図の描き方

2 次系の状態軌道を状態平面 (x_1, x_2) に描いたものを位相面図という．位相面図は，2 次系の挙動を調べるためによく用いられ，特に，非線形系の解析に威力を発揮する．

例題 **2.3** 例題 2.2 に対する位相面図を求めるプログラムを作成せよ．

(解答例)

ファイル名：ex2_3.m

```
A = [0 1; -1 -1];
B = [0; 1];
u=1;
dt = 0.05;
tf=20;
x=[0 0]';
xx = [];
i=0;
for t=0:dt:tf
   i=i+1;
   xx(:,i)=x;
   xt = x;
   for j=1:4
      f = A*x + B*u;
      d(:,j) = f*dt;
      x = xt + d(:,j)*0.5;
```

```
        if j==3
           x = xt + d(:,j);
        endif
     endfor
     x = xt + (d(:,1) + d(:,2)*2 + d(:,3)*2 + d(:,4))/6;
  endfor
  grid "on"
  plot(xx(1,:),xx(2,:))
```

グラフ表示結果を図 2.7 に示す．

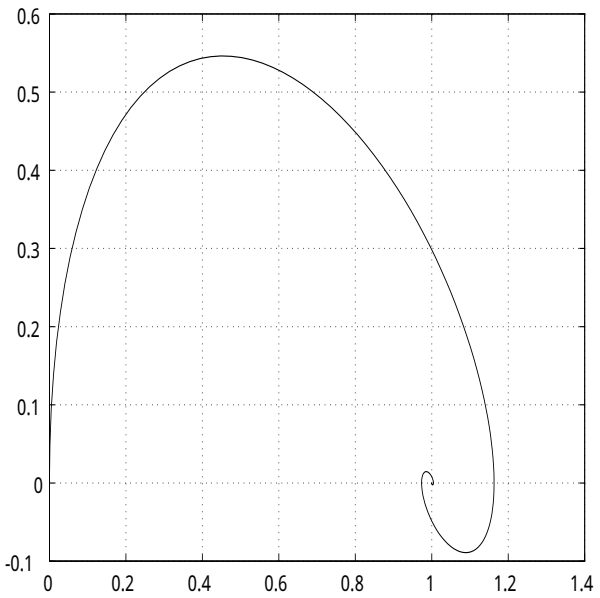

図 2.7: 位相面図 (例題 2.3)

例題 **2.4** 振子長 l の単振子系の運動方程式は，安定平衡点からの振れ角を θ とすると

$$\ddot{\theta} + \frac{g}{l}\sin\theta = 0 \tag{2.24}$$

と表される．ルンゲクッタ法を用いて，振子系の自由振動の位相面図を描け．ただし，$l = 1\mathrm{m}$, $g = 9.8\mathrm{m/s^2}$ とし，初期状態を

$$\theta(0) = 2.5\mathrm{rad}, \quad \dot{\theta}(t) = 0$$

とする．また，シミュレーション終了時刻は $5\mathrm{s}$ とする．

(解答) まず，(2.24) 式を状態方程式に直す．

$$x_1 := \theta, \quad x_2 := \dot{\theta} \tag{2.25}$$

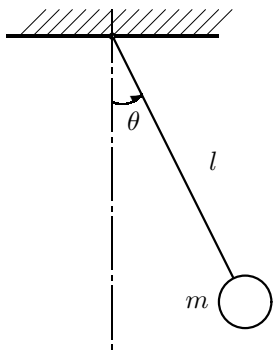

図 2.8: 単振子

と用いると，(2.24) 式は

$$\left.\begin{array}{rcl} \dot{x}_1 &=& x_2 \\ \dot{x}_2 &=& -(g/l)\sin x_1 \end{array}\right\} \tag{2.26}$$

すなわち

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad u(t) = -\frac{g}{l}\sin x_1(t) \tag{2.27}$$

となる．シミュレーションプログラムは次のとおりである．

ファイル名：ex2_4.m

```
A = [0 1; 0 0];
B = [0; 1];
dt = 0.05;
tf=4;
x=[2.5 0]';
l=1;
g=9.8;
a1=g/l;
xx = [];
i=0;
for t=0:dt:tf
   i=i+1;
   xx(:,i)=x;
   xt = x;
   for j=1:4
```

```
      u = -a1*sin(x(1));
      f = A*x + B*u;
      d(:,j) = f*dt;
      x = xt + d(:,j)*0.5;
      if j==3
         x = xt + d(:,j);
      endif
   endfor
   x = xt + (d(:,1) + d(:,2)*2 + d(:,3)*2 + d(:,4))/6;
endfor
grid "on"
axis([-pi pi -10 10])  % axis([xmin xmax ymin ymax])
plot(xx(1,:),xx(2,:))
```

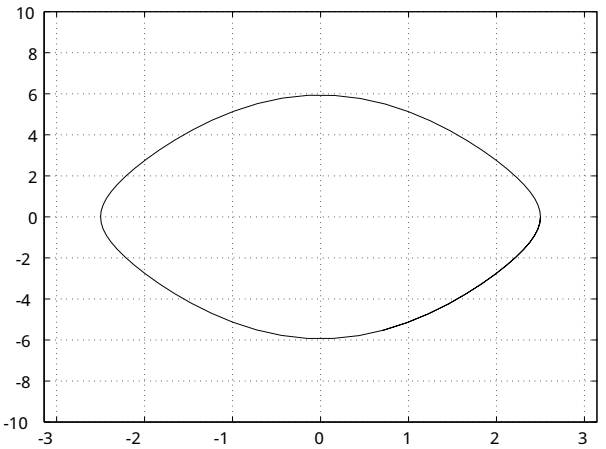

図 2.9: 位相面図 (例題 2.4)

例題 2.4 に対するシミュレーション結果を図 2.9 に示す．

演習

1. 例題 2.4 において，初期状態を種々変えてシミュレーションしてみよ．
2. 例題 2.4 において，ルンゲクッタ法の計算精度を調べるため，終了時刻を大きくして，軌跡を重ねて描いてみよ．また，刻み幅を種々変えて同じ実験を行ってみよ．

   前のグラフ結果を残すには，plot の上に

   ```
   hold on
   ```

を書く．

3. 例題 2.4 のプログラムを修正して時間応答を求めよ．

4. ファンデアポール方程式

$$\ddot{x}(t) - \epsilon(1 - x(t)^2)\dot{x}(t) + x(t) = 0, \quad x(0) = 0.1, \quad \dot{x}(0) = 0 \tag{2.28}$$

の位相面図を描け．ただし，$\epsilon = 1$ とする．

2.5　非線形制御の例：可変長振子系の振動制御問題

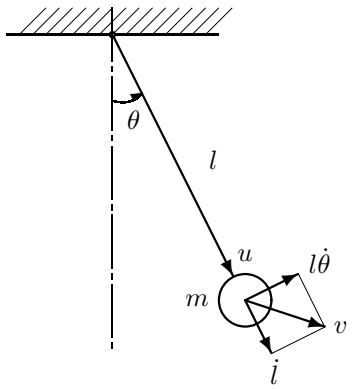

図 2.10: 可変長振子系

可変長振子系 (図 2.10) の運動方程式は，ラグランジュの方法により，つぎのように求められる。

ラグランジュ関数 L は，運動エネルギ T とポテンシャルエネルギ U を用いて

$$L = T - U \tag{2.29}$$

と計算される。この系に対する T および U は

$$T = \frac{1}{2}m(\dot{l}^2 + l^2\dot{\theta}^2) \tag{2.30}$$

$$U = -mgl\cos\theta \tag{2.31}$$

である。よって

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{2.32}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{l}}\right) - \frac{\partial L}{\partial l} = u \tag{2.33}$$

から，つぎの運動方程式を得る。

$$ml^2\ddot{\theta} + 2ml\dot{l}\dot{\theta} + mgl\sin\theta = 0 \tag{2.34}$$

$$m\ddot{l} - ml\dot{\theta}^2 - mg\cos\theta = u \tag{2.35}$$

ただし，θ は振子の振れ角，l は支点から重りまでの距離，m は重りの質量，g は重力加速度，u は重りの駆動力である。振子の棒の質量は無視している。l はつぎの制限を受けるものとする。

$$0 < l_0 \le l \le l_1 \tag{2.36}$$

設計を容易にするため，(2.35) 式をつぎのように線形化する。

$$\ddot{l} = \mu \tag{2.37}$$

μ は新しい入力である。この線形化を行う u は次式で与えられる。

$$u = m\mu - ml\dot{\theta}^2 - mg\cos\theta \tag{2.38}$$

問題は，(2.36) 式の制限のもとで，振子系の振動を速やかに減衰させる l の制御則を求めることである。

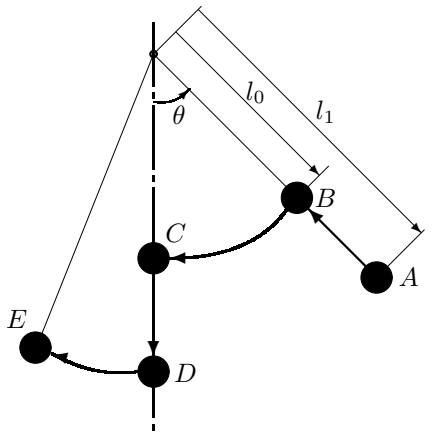

図 2.11: 最適制御

振子振動の減衰を最大にする最適制御は，図 2.11 に示すように，振子長を $\dot{\theta} = 0$ のとき l_1 から l_0 へ，$\theta = 0$ のとき l_0 から l_1 へ瞬間的に移動する制御である．実際には，このように重りを瞬間的に移動する制御の実現は不可能であるが，これを近似的に行う制御系をシミュレーションしてみよう．

いま，r を l の目標値とする．r から l までの伝達関数が

$$G(s) = \frac{1}{(1+Ts)^2} \tag{2.39}$$

となるように，l のサーボ系を次のように構成する．

$$\mu = k_1(r - l) - k_2\dot{l} \tag{2.40}$$

ただし

$$k_1 = \frac{1}{T^2}, \quad k_2 = \frac{2}{T}, \quad , T > 0 \tag{2.41}$$

である．r は l_0 または l_1 の値をとるものとし，次式に従って切り換える．

$$r = \begin{cases} l_0 & \text{if} \quad \theta\dot{\theta} < 0 \\ l_1 & \text{if} \quad \theta\dot{\theta} > 0 \end{cases} \tag{2.42}$$

以上で得られた制御系の微分方程式を状態方程式表現すると，(2.34) 式から

$$\ddot{\theta} = -(g\sin\theta + 2\dot{l}\dot{\theta})/l \tag{2.43}$$

また，(2.40) 式を (2.37) 式へ代入して

$$\ddot{l} = -k_1 l - k_2\dot{l} + k_1 r \tag{2.44}$$

を得る．状態を

$$\boldsymbol{x} := \begin{bmatrix} \theta & \dot{\theta} & l & \dot{l} \end{bmatrix}^T \tag{2.45}$$

と定義して，(2.43)，(2.44) 式を状態方程式で表すと次式を得る．

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.46}$$

ただし

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k_1 & -k_2 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.47}$$

$$\boldsymbol{u} = \begin{bmatrix} -(g\sin\theta + 2\dot{l}\dot{\theta})/l \\ k_1 r \end{bmatrix} \tag{2.48}$$

例題 2.5 上述の考え方に従って，可変長振子系の振動制御シミュレーションプログラムを作成せよ（状態変数の時間応答をグラフ表示するようにせよ）．そして，つぎの条件でシミュレーションしてみよ．

$l_0 = 0.8\text{m}, \quad l_1 = 1.2\text{m}, \quad , \quad g = 9.8\text{m/s}^2,$

$\boldsymbol{x}(0) = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T$

刻み幅 $= 0.005\text{s}, \quad$ 終了時刻 $= 10\text{s}$

$T = 0.1, \quad 0.05, \quad , 0.01$

(解答例)

```
T=input('Enter T: ');
k1 = 1/T^2; k2 = 2/T;
l0 = 0.8; l1 = 1.2;
g = 9.8;
A = [0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 -k1 -k2];
B = [0 0; 1 0; 0 0; 0 1];
dt = 0.005;
tf = 10;
x = [1 0 1 0]';
xx = [];
i = 0;
for t=0:dt:tf
   i=i+1;
   xx(:,i)=x;
   xt = x;
   for j=1:4
      if x(1)*x(2)<0
          r = l0;
      else
         r = l1;
      endif
      u = [(-g*sin(x(1)) - 2*x(4)*x(2))/x(3); k1*r];
      f = A*x + B*u;
      d(:,j) = f*dt;
      x = xt + d(:,j)*0.5;
      if j==3
         x = xt + d(:,j);
      endif
   endfor
   x = xt + (d(:,1) + d(:,2)*2 + d(:,3)*2 + d(:,4))/6;
endfor
t = 0:dt:tf;
grid "on"
axis([0 10 -0.8 1.2])
plot(t,xx(1,:),t,xx(3,:))
```

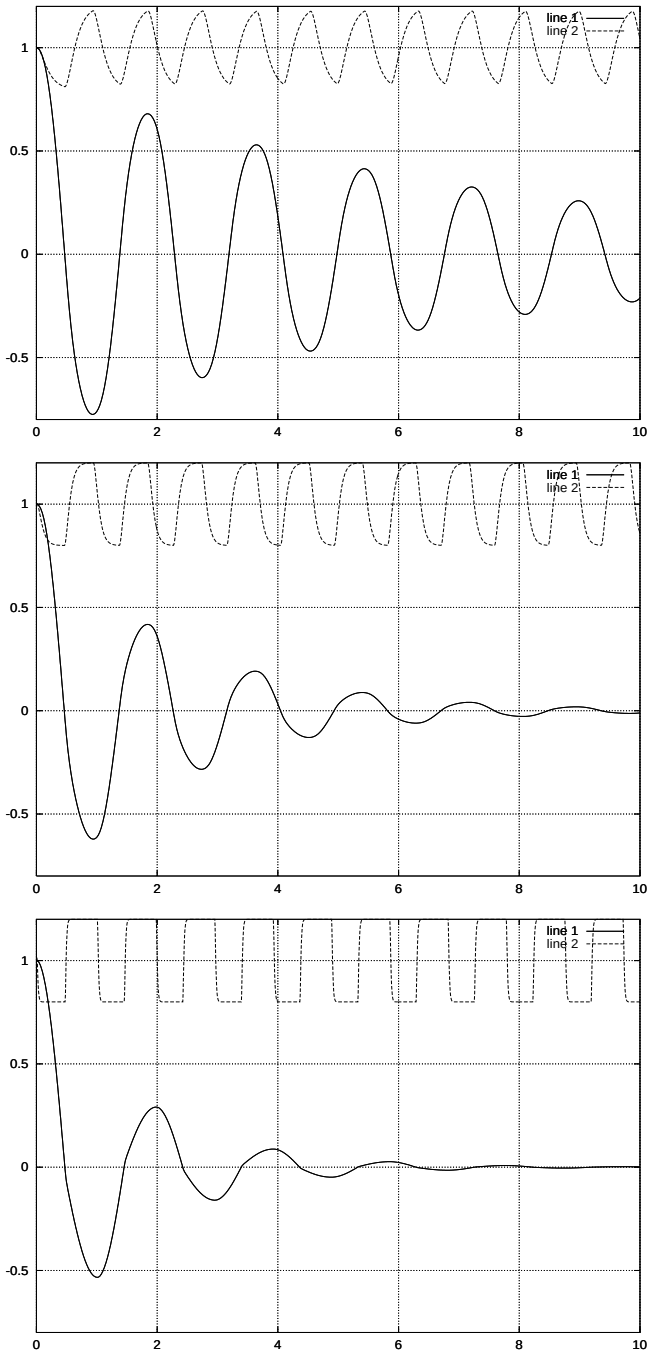

図 2.12: シミュレーション結果 (上から $T = 0.1,\ 0.05,\ 0.01$)

演習

1. 例題 2.5 のシミュレーションプログラムについて、位相面図を出力させるプログラムに変更せよ。そして、可変長振子系が安定化制御された場合の位相面軌道を観察せよ。

2. 振子の振り付け (不安定化) は、l を安定化の逆のパターンで変化させることによって行われる。不安定化のプログラムを作成せよ。そして、小さな初期角度、例えば

$$\boldsymbol{x}(0) = \left[\begin{array}{cccc} 0.1 & 0 & 1 & 0 \end{array} \right]^T$$

を与えてシミュレーションしてみよ。時間応答、位相面軌道をそれぞれ表示するプログラムを作成せよ。

2.6 制御系設計用関数

2.6.1 極配置法

制御対象

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}u(t) \tag{2.49}$$

に対して制御則

$$u(t) = -\boldsymbol{k}\boldsymbol{x}(t) \tag{2.50}$$

が与えられたとすると，閉ループ系は

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{b}\boldsymbol{k})\boldsymbol{x}(t) \tag{2.51}$$

となる．この閉ループ系をレギュレータという場合がある．$(\boldsymbol{A}, \boldsymbol{b})$ が可制御ならば，閉ループ系のシステム行列 $\boldsymbol{A} - \boldsymbol{b}\boldsymbol{k}$ の固有値を実軸に対して対称な任意の位置に配置できる．$\boldsymbol{A} - \boldsymbol{b}\boldsymbol{k}$ の固有値をレギュレータ極という．

表 2.1: 極配置関数

関数	説明
`place`	1 入力系用極配置関数

極を制御対象極から大きく離れて指定すると，フィードバックゲインの要素が大きくなり，制御系の感度を上げることになるので，そのような極端な極配置は行わないようにする．

例題 2.6

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

について，レギュレータ極を $-1 \pm j$ に配置するフィードバックゲインを求めよ．

(解答)

ファイル名：ex2_6.m

```
A = [0 1; 0 0];
b = [0; 1];
c = eye(2,2);            % c = I とする．
d = [];
sys = ss2sys(A,b,c,d);   % 状態方程式を sys に設定する．
P = [-1+i -1-i];
k = place(sys,P)
```

このプログラムを実行すれば

```
octave:1> ex2_6
k =
  2   2
```

と結果が表示される．このフィードバックゲインが指定極を実現しているかどうかは eig によって確認できる．

```
octave:2> eig(A-b*k)
ans =
  -1.0000 + 1.0000i
  -1.0000 - 1.0000i
```

2.6.2　オブザーバ

一般に制御対象は次式で記述される．

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}(t) & = & \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}u(t) \\ y(t) & = & \boldsymbol{c}\boldsymbol{x}(t) + du(t) \end{array}\right\} \tag{2.52}$$

$y(t)$ は出力であり，観測される変数である．上述の状態フィードバック制御 $u(t) = -\boldsymbol{k}\boldsymbol{x}(t)$ では，$\boldsymbol{x}(t)$ が必要であるが，実際には，$u(t)$ と $y(t)$ を利用して推定した状態 $\boldsymbol{\xi}(t)$ を $\boldsymbol{x}(t)$ の代わりに用いることになる．$u(t), y(t)$ から $\boldsymbol{\xi}(t)$ を得る機構をオブザーバという．

オブザーバは次式で構成される．

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{b}u(t) + \boldsymbol{l}(y(t) - \boldsymbol{c}\boldsymbol{\xi}(t) - du(t)) \tag{2.53}$$

誤差ベクトルを

$$\boldsymbol{e}(t) := \boldsymbol{x}(t) - \boldsymbol{\xi}(t) \tag{2.54}$$

と定義すると，(2.52)，(2.53) 式から

$$\dot{\boldsymbol{e}}(t) = (\boldsymbol{A} - \boldsymbol{l}\boldsymbol{c})\boldsymbol{e}(t) \tag{2.55}$$

を得る．$(\boldsymbol{c}, \boldsymbol{A})$ が可観測ならば，$\boldsymbol{l}$ を選んで $(\boldsymbol{A} - \boldsymbol{l}\boldsymbol{c})$ の固有値を実軸に対して対称な任意の位置に配置できる．$(\boldsymbol{A} - \boldsymbol{l}\boldsymbol{c})$ の固有値をオブザーバ極という．

$(\boldsymbol{A} - \boldsymbol{l}\boldsymbol{c})$ の固有値は，$(\boldsymbol{A}^T - \boldsymbol{c}^T\boldsymbol{l}^T)$ の固有値に等しいので，$(\boldsymbol{A}^T - \boldsymbol{c}^T\boldsymbol{l}^T)$ に対して place を使って ($\boldsymbol{A} \to \boldsymbol{A}^T, \boldsymbol{b} \to \boldsymbol{c}^T$ と置き換える) 指定の極を実現する $\boldsymbol{l}$ を求めることができる．オブザーバ極はレギュレータ極より左に配置するのがよいとされる．

例題 **2.7**

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

について，$(\boldsymbol{A} - \boldsymbol{l}\boldsymbol{c})$ の固有値を $(-2, -3)$ に配置する $\boldsymbol{l}$ を求めよ．

(解答)

ファイル名：ex2_7.m

```
A = [0 1; 0 0];
c = [1 0];
Ao = A';
bo = c';
co = eye(2,2);
do = [];
syso = ss2sys(Ao,bo,co,do);
P = [-2 -3];
l = place(syso,P);
l = l'
```

このプログラムを実行すれば

```
octave:1> ex2_7
l =
  5
  6
```

と結果が表示される．オブザーバ極は

```
octave:2> eig(A-l*c)
ans =
  -3.0000
  -2.0000
```

と確認される．

(2.52)，(2.53) 式において，$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{\xi}$ とおくとレギュレータ・オブザーバ併合系の状態方程式

$$\begin{bmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\boldsymbol{\xi}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{b}\boldsymbol{k} \\ \boldsymbol{l}\boldsymbol{c} & \boldsymbol{A} - \boldsymbol{l}\boldsymbol{c} - \boldsymbol{b}\boldsymbol{k} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\xi} \end{bmatrix} \tag{2.56}$$

を得る．また，この式に座標変換

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{e} \end{bmatrix} \tag{2.57}$$

を施すと

$$\begin{bmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\boldsymbol{e}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{b}\boldsymbol{k} & \boldsymbol{b}\boldsymbol{k} \\ \boldsymbol{0} & \boldsymbol{A} - \boldsymbol{l}\boldsymbol{c} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{e}(t) \end{bmatrix} \tag{2.58}$$

となる．よって，レギュレータ極とオブザーバ極は独立に設定できることがわかる．

例題 **2.8** システム

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x}(t)$$

に対して

レギュレータ極　：　$(-1 \pm j)$

オブザーバ極　：　$(-2,\ -3)$

と設計したレギュレータ・オブザーバ併合系の時間応答をシミュレーションするプログラムを作成せよ．ただし，初期状態は

$$\begin{bmatrix} \boldsymbol{x}(0) \\ \boldsymbol{\xi}(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

とする．

(解答)

ファイル名：ex2_8.m

```
n = 2;
A = [0 1; 0 0];
b = [0; 1];
c = [1 0];
d = [];
sys = ss2sys(A,b,c,d);
Ao = A';
bo = c';
co = eye(2,2);
do = [];
syso = ss2sys(Ao,bo,co,do);
Pr = [-1+i -1-i];
Po = [-2 -3];
k = place(sys,Pr);
l = place(syso,Po);
l = l';
A1 = [A -b*k;l*c A-l*c-b*k];
dt = 0.01;
tf=10;
x=[1 0 0 0]';
xx = [];
k=0;
for t=0:dt:tf
   k=k+1;
   xx(:,k)=x;
   xt = x;
   for j=1:4
      f = A1*x;
      d(:,j) = f*dt;
      x = xt + d(:,j)*0.5;
      if j==3
         x = xt + d(:,j);
      endif
   endfor
   x = xt + (d(:,1) + d(:,2)*2 + d(:,3)*2 + d(:,4))/6;
endfor
```

```
t = 0:dt:tf;
grid "on"
plot(t,xx)
```

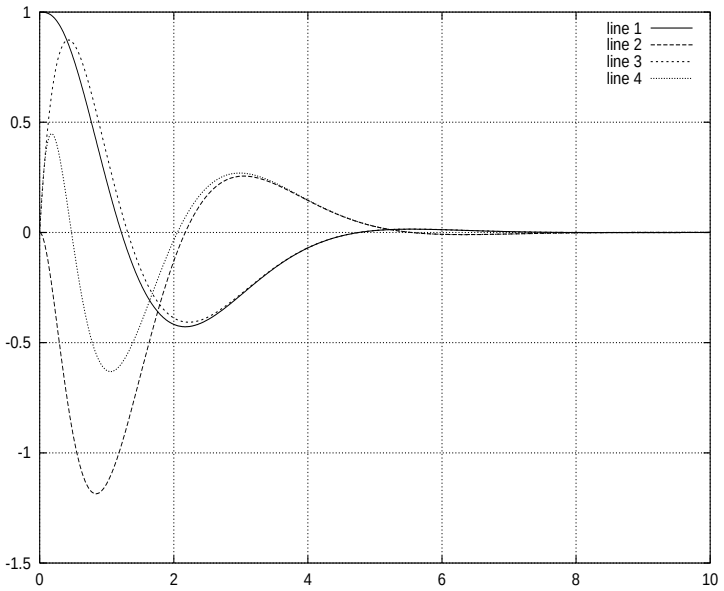

図 2.13: 時間応答 (例題 2.8)

演習

1. システム

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t)$$

に対して

$$\text{レギュレータ極} \quad : \quad (-1 \pm j, -1)$$
$$\text{オブザーバ極} \quad : \quad (-2, -3, -4)$$

と設計したレギュレータ・オブザーバ併合系の時間応答をシミュレーションするプログラムを作成せよ．ただし，初期状態は

$$\begin{bmatrix} \boldsymbol{x}(0) \\ \boldsymbol{\xi}(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

とする．

2.6.3 LQG 制御

制御対象の入出力にノイズが混入する場合，2 次形式評価関数に基づいてレギュレータ・オブザーバ併合系の最適設計を行う方法を紹介しよう．

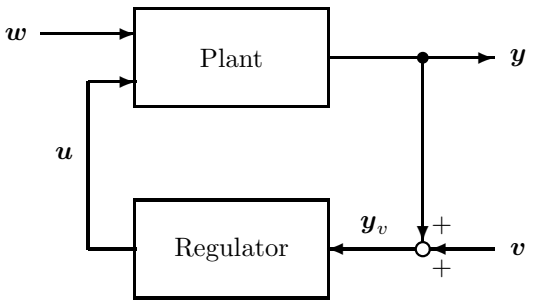

図 2.14: LQG レギュレータ

図 2.14 において，$\boldsymbol{v}$, $\boldsymbol{w}$ は白色雑音である．制御系の状態方程式は以下のとおりである．

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}(t) & = & \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{G}\boldsymbol{w}(t) \\ \boldsymbol{y}_v(t) & = & \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{H}\boldsymbol{w}(t) + \boldsymbol{v}(t) \end{array}\right\} \tag{2.59}$$

この制御系は，最適レギュレータとカルマンフィルタを併合した系であり，LQG レギュレータと呼ばれる．構造はレギュレータ・オブザーバ併合系と同じであるが，レギュレータゲインとオブザーバゲインの計算法が異なり，それぞれ，LQ 最適ゲイン，カルマンゲインを用いることになる．

a. LQ 最適ゲイン

二次形式評価関数

$$J(\boldsymbol{u}) = \int_0^\infty (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + 2\boldsymbol{x}^T\boldsymbol{N}\boldsymbol{u} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})dt \tag{2.60}$$

を最小化する制御は

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t) \tag{2.61}$$

で与えられる．LQ(linear quadratic) 最適ゲイン $\boldsymbol{K}$ は，リカッチ方程式と呼ばれる行列代数方程式の解から得られる．$\boldsymbol{K}$ は `lqr` によって次のように計算できる．

```
[K,S,Er] = lqr(A,B,Q,R,N)
```

ただし，S はリカッチ方程式の解，Er は $(\boldsymbol{A}-\boldsymbol{BK})$ の固有値である．

b. カルマンゲイン

白色雑音 $\boldsymbol{w}$, $\boldsymbol{v}$ の平均値は $\boldsymbol{0}$ でつぎの共分散行列を持つとする．

$$E(\boldsymbol{w}(t)\boldsymbol{w}(\tau)^T)=\boldsymbol{Q}_n\delta(t-\tau),\quad E(\boldsymbol{v}(t)\boldsymbol{v}(\tau)^T)=\boldsymbol{R}_n\delta(t-\tau),$$

$$E(\boldsymbol{w}(t)\boldsymbol{v}(\tau)^T)=\boldsymbol{N}_n\delta(t-\tau) \tag{2.62}$$

状態の推定を $\hat{\boldsymbol{x}}(t)$ で表すと，カルマンフィルタは

$$\lim_{t\to\infty}E\left((\boldsymbol{x}(t)-\hat{\boldsymbol{x}}(t))(\boldsymbol{x}(t)-\hat{\boldsymbol{x}}(t))^T\right) \tag{2.63}$$

という基準によって設計される．すなわち，カルマンフィルタは状態推定誤差の共分散を最小にする最適フィルタであり，次式で与えられる．

$$\dot{\hat{\boldsymbol{x}}}(t)=\boldsymbol{A}\hat{\boldsymbol{x}}(t)+\boldsymbol{B}\boldsymbol{u}(t)+\boldsymbol{L}(\boldsymbol{y}_v(t)-\boldsymbol{C}\hat{\boldsymbol{x}}(t)-\boldsymbol{D}\boldsymbol{u}(t)) \tag{2.64}$$

$\boldsymbol{L}$は，あるリカッチ方程式解（LQ 最適ゲインのリカッチ方程式とは異なる）から得られる．制御は，$\boldsymbol{u}(t)=-\boldsymbol{K}\hat{\boldsymbol{x}}(t)$ を用いることになるので，これを (2.64) 式に代入すると LQG レギュレータの状態方程式

$$\left.\begin{array}{rcl}\dot{\hat{\boldsymbol{x}}}(t) &=& \{\boldsymbol{A}-\boldsymbol{LC}-(\boldsymbol{B}-\boldsymbol{LD})\boldsymbol{K}\}\hat{\boldsymbol{x}}(t)+\boldsymbol{L}\boldsymbol{y}_v(t)\\ \boldsymbol{u}(t) &=& -\boldsymbol{K}\hat{\boldsymbol{x}}(t)\end{array}\right\} \tag{2.65}$$

が得られる．$\boldsymbol{L}$ は lqr によって次のように計算できる．

```
[L,P,Eo] = lqr(A',C',G*Qn*G',Rn,Nn)
L = L'
```

ただし，P はリカッチ方程式の解，Eo は $(\boldsymbol{A}-\boldsymbol{LC})$ の固有値である．

上述のように，LQG 制御は，システムと出力に白色雑音が混入する場合に，二次形式評価関数の観点から最適となるように設計されたものである．制御系の構造は，レギュレータ・オブザーバ併合系と同じであるが，レギュレータゲインとオブザーバゲインをリカッチ方程式解から構成するという点に特徴がある．LQG 制御系は，白色雑音に対して好ましい特性を持つだけでなく，制御対象のパラメータ変化に対しても良好なロバスト性を持つことが知られているので，白色雑音を想定しない場合の制御系設計においてもしばしば利用される．

例題 2.9 システム

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}0&1\\0&0\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t)$$

$$y(t)=\begin{bmatrix}1&0\end{bmatrix}\boldsymbol{x}(t)$$

に対して，LQ 最適ゲインとカルマンゲインを利用して，レギュレータ・オブザーバ併合系を設計し，制御系の時間応答をシミュレーションするプログラムを作成せよ．つぎのデータを使ってシミュレーションしてみよ．

$$\text{LQ 最適ゲイン} \quad : \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1, \quad N = \mathbf{0}$$

$$\text{カルマンゲイン} \quad : \quad \boldsymbol{Q}_n = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R_n = 1, \quad N_n = \mathbf{0}, \quad \boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{x}(0) \\ \hat{\boldsymbol{x}}(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$$

(解答)

ファイル名：ex2_9.m

```
n = 2;
% System matrices
A = [0 1; 0 0];
B = [0; 1];
C = [1 0];
D = 0;
G = eye(n,n);
% Weighting matrices for optimal regulator
Q = eye(n,n);
R = 1;
% Weighting matrices for Kalman estimator
Qn = eye(n,n)*10;
Rn = 1;
[K,S,Er] = lqr(A,B,Q,R);
[L,P,Ee] = lqr(A',C',G*Qn*G',Rn);
L = L';
A1 = [A -B*K;L*C A-L*C-B*K];
dt = 0.01;
tf=10;
x = zeros(2*n,1);
x(1) = 1;
xx = [];
k=0;
for t=0:dt:tf
   k=k+1;
   xx(:,k)=x;
```

```
   xt = x;
   for j=1:4
      f = A1*x;
      d(:,j) = f*dt;
      x = xt + d(:,j)*0.5;
      if j==3
         x = xt + d(:,j);
      endif
   endfor
   x = xt + (d(:,1) + d(:,2)*2 + d(:,3)*2 + d(:,4))/6;
endfor
t = 0:dt:tf;
grid "on"
plot(t,xx)
```

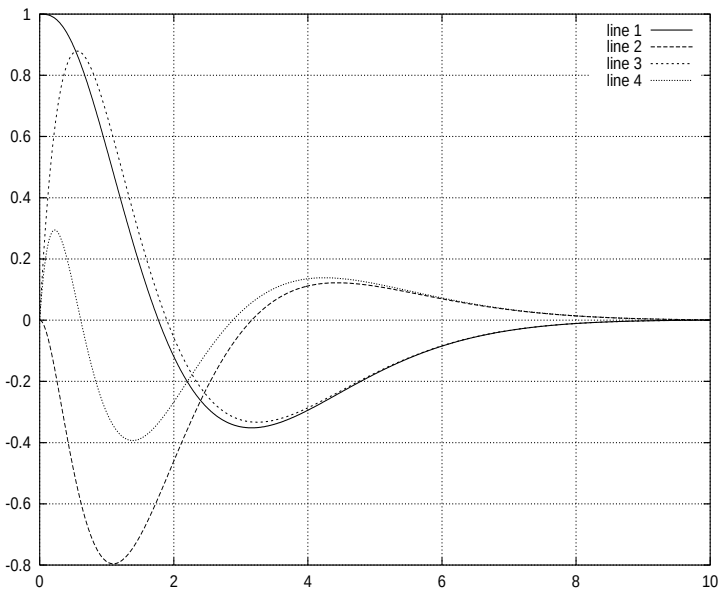

図 2.15: 時間応答 (例題 2.9)

演習

1. システム

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t)$$

に対して，LQ 最適ゲインとカルマンゲインを利用して，レギュレータ・オブザーバ併合系を設計し，時間応答をシミュレーションするプログラムを作成せよ．ただし，つぎのデータを用いるものとする．

$$\text{LQ 最適ゲイン} \quad : \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R = 1, \quad N = \boldsymbol{0}$$

$$\text{カルマンゲイン} \quad : \quad \boldsymbol{Q}_n = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \quad R_n = 1, \quad N_n = \boldsymbol{0},$$

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{x}(0) \\ \hat{\boldsymbol{x}}(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

2.7 伝達関数に基づく解析法

2.7.1 システムの伝達関数表現

(1) 係数を入力する方法

伝達関数

$$G(s) = \frac{5s^2 + 6s + 7}{s^3 + 2s^2 + 3s + 4}$$

を入力するには `tf2sys` 関数を用いる．

```
octave:1> sys = tf2sys([5 6 7],[1 2 3 4]);
octave:2> sysout(sys,"tf")
Input(s)
      1:u_1
Output(s)
```

```
      1:y_1
transfer function form:
5*s^2 + 6*s^1 + 7
-------------------------
1*s^3 + 2*s^2 + 3*s^1 + 4
```

一般に，伝達関数の分母の係数を降べき順に並べた 1 次元配列を den，分子の係数を降べき順に並べた 1 次元配列を num とすると，伝達関数は

```
sys = tf2sys(num,den);
```

で設定できる．ただし，Octave で入力できる伝達関数には

分子の次数 $\leq$ 分母の次数

という制限があるので注意する．sysout 関数は，システムを表示する関数である．"tf" で通常の伝達関数表示を指定している．

比例要素

$$G(s) = K$$

を設定するには

```
sys = tf2sys(K,1);
```

とするが，特に，$G(s) = 1$ に対しては

```
sys = ugain(1);
```

という関数が用意されている．

(2)　極・零点・ゲインを入力する方法

伝達関数

$$G(s) = \frac{10(s+4)(s+5)}{(s+1)(s+2)(s+3)}$$

を極-零点形式で入力するには zp2sys 関数を用いることができる．

```
octave:1> sys = zp2sys([-4 -5],[-1 -2 -3],10);
octave:2> sysout(sys,"zp")
Input(s)
      1:u_1
Output(s)
      1:y_1
zero-pole form:
10 (s + 4) (s + 5)
-------------------------
  (s + 1) (s + 2) (s + 3)
```

一般に，極を並べた 1 次元配列を poles，零点を並べた 1 次元配列を zeros，ゲインを K とすると，伝達関数は

```
sys = zp2sys(zeros,poles,K);
```

で入力できる．sysout 関数で"zp"と指定すると，伝達関数が極-零点形式で表示される．

演習

1. 次の伝達関数を入力するプログラムを書け．

$$(1)\quad 20 \qquad (2)\quad \frac{1}{s} \qquad (3)\quad \frac{4}{s^2} \qquad (4)\quad \frac{1}{s^2+1}$$

$$(5)\quad \frac{2s^2+5s}{s^3+2s+1} \qquad (6)\quad \frac{s^3-2s+2}{s^4+s^3+2s^2+s+1}$$

$$(7)\quad \frac{5s(s+2)}{(s+0.1)(s+0.5)(s+1)} \qquad (8)\quad \frac{0.1(s+0.1)}{(1+0.1s)(1+0.2s)}$$

2.7.2 伝達関数の結合

表 2.2: 伝達関数の結合を求める関数

関数	説明
sysmult	直列結合
sysadd	並列結合 (和)
syssub	並列結合 (差)
buildssic	一般的結合

伝達関数の結合を求めるため，表 2.2 の関数が用意されている．使用例を以下に示す．sys は，U から Y までの伝達関数を表す．

1. 直列結合 (図 2.16)

```
sys = sysmult(sys1,sys2)
```

2. 並列結合 (和)(図 2.17)

```
sys = sysadd(sys1,sys2)
```

3. 並列結合 (差)(図 2.18)

```
sys = syssub(sys1,sys2)
```

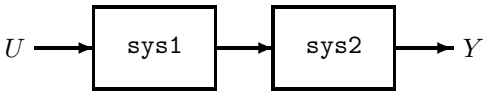

図 2.16: 直列結合

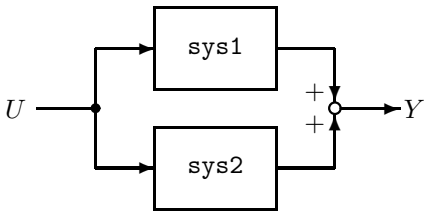

図 2.17: 並列結合 (和)

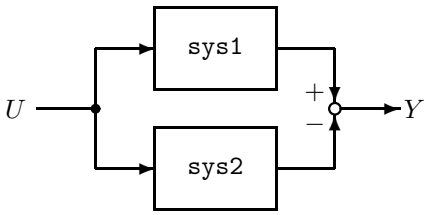

図 2.18: 並列結合 (差)

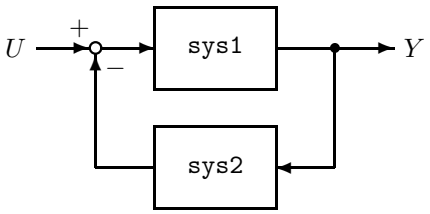

図 2.19: フィードバック結合 (負のフィードバック)

4. フィードバック結合 (負のフィードバック)(図 2.19)

   `buildssic` 関数を使って次のように書ける．

   ```
   sys = buildssic([1 -2;2 1],[],1,1,sys1,sys2)
   ```

   `buildssic` 関数の引数の意味は次のとおりである．

   `[1 -2;2 1]`　`sys1` には `sys2` の出力が負符号で入力される．そして，`sys2` には `sys1` の出力が入力される．この引数は行列として与えることに注意する (要素数が不足する場合，0 で埋める)．

[],1,1　出力リストとして新たに何も加えない．結合系の出力は sys1 の出力とする．そして，結合系の入力は sys1 の入力とする．

sys1,sys2　ブロックの伝達関数．この順にしたがって，ブロックの番号が割り振られる．伝達関数は 8 つまで入力できる．

この関数を使って，MATLAB の feedback 関数と同様な働きをする関数を定義できる．

```
function sys = feedback(sys1,sys2)
    sys = buildssic([1 -2;2 1],[],1,1,sys1,sys2);
endfunction
```

これを feedback.m として Octave の作業ディレクトリに保存しておけば，毎回定義しなくてすむ．feedback 関数は次のように使う．

```
sys = feedback(sys1,sys2);
```

5. フィードバック結合 (正のフィードバック)(図 2.20)

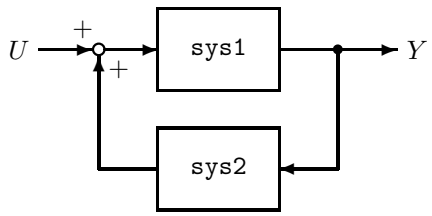

図 2.20: フィードバック結合 (正のフィードバック)

同様に，buildssic 関数を用いて

```
sys = buildssic([1 2;2 1],[],1,1,sys1,sys2);
```

と書く．もちろん，これを例えば pfeedback という関数として定義して使用してもよい．

```
function sys = pfeedback(sys1,sys2)
    sys = buildssic([1 2;2 1],[],1,1,sys1,sys2);
endfunction
```

6. 一般的結合の例

buildssic 関数を使えば複雑なブロック線図の伝達関数を容易に求めることができる．

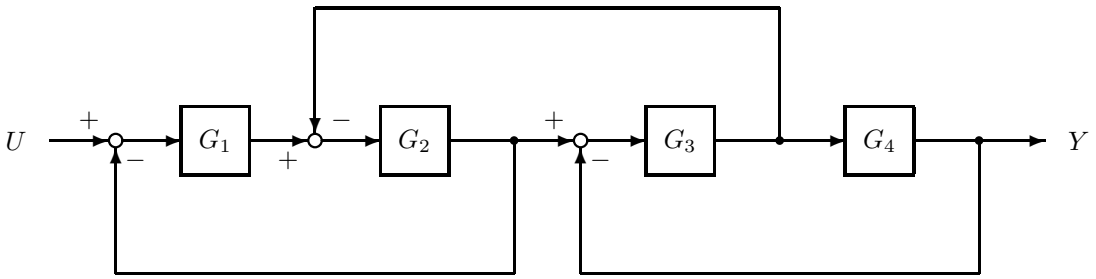

図 2.21: 例題のブロック線図

例題 2.10 図 *2.21* のブロック線図について , U から Y までの伝達関数を求めよう . ただし

$$G_1 = \frac{1}{0.1s+1}, \quad G_2 = \frac{1}{s(s+1)}, \quad G_3 = \frac{1}{0.2s+1}, \quad G_4 = \frac{2}{s^2+4s+2}$$

とする .

(答)

$$G(s) = \frac{100}{s^6 + 20s^5 + 131s^4 + 367s^3 + 600s^2 + 630s + 300}$$

(解答例)

```
octave:1> s1 = tf2sys(1,[0.1 1]);
octave:2> s2 = tf2sys(1,[1 1 0]);
octave:3> s3 = tf2sys(1,[0.2 1]);
octave:4> s4 = tf2sys(2,[1 4 2]);
octave:5> sys = buildssic([1 -2 0;2 1 -3;3 2 -4;4 3 0],[],
>4,1,s1,s2,s3,s4);
octave:6> sysout(sys,"tf")
Input(s)
      1:u_1
Output(s)
      1:y_1
transfer function form:
100
-------------------------------------------------------------
1*s^6 + 20*s^5 + 131*s^4 + 367*s^3 + 600*s^2 + 630*s^1 + 300
```

演習

1. 図 2.22 のブロック線図について，U から Y までの伝達関数を求めよ．ただし

$$G_1 = \frac{1}{s+1}, \quad G_2 = 5, \quad G_3 = \frac{1}{s^2+2s+1}, \quad G_4 = \frac{2s+3}{s+1}$$

とする．
（答）

$$G(s) = \frac{s+1}{s^4+9s^3+23s^2+34s+24}$$

2. 図 2.23 のブロック線図について，U から Y までの伝達関数を求めよ．ただし

$$G_1 = \frac{2}{s}, \quad G_2 = \frac{s+1}{0.01s+1}, \quad G_3 = \frac{1}{s^2+2s+1}, \quad G_4 = \frac{1}{0.1s+1}$$

とする．
（答）

$$G(s) = \frac{100s^3+1102s^2+1220s+2000}{s^5+112s^4+1222s^3+3220s^2+3020s+2000}$$

3. 図 2.24 のブロック線図について，U から Y までの伝達関数を求めよ．ただし

$$G_1 = \frac{10}{s+0.2}, \quad G_2 = \frac{1}{s^2+s+1}, \quad G_3 = \frac{2s+1}{s+2}, \quad G_4 = \frac{5s+11}{0.01s+1}$$

とする．
（答）

$$G(s) = \frac{20s^2+2010s+1000}{s^5+103.2s^4+825.6s^3+2774s^2+4040.6s+2500}$$

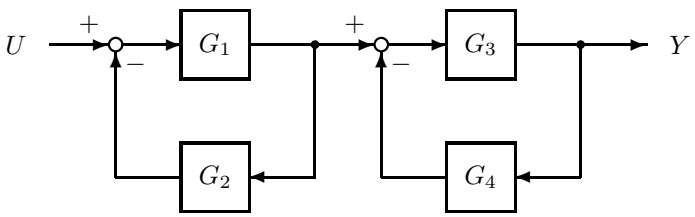

図 2.22: 問題 1 のブロック線図

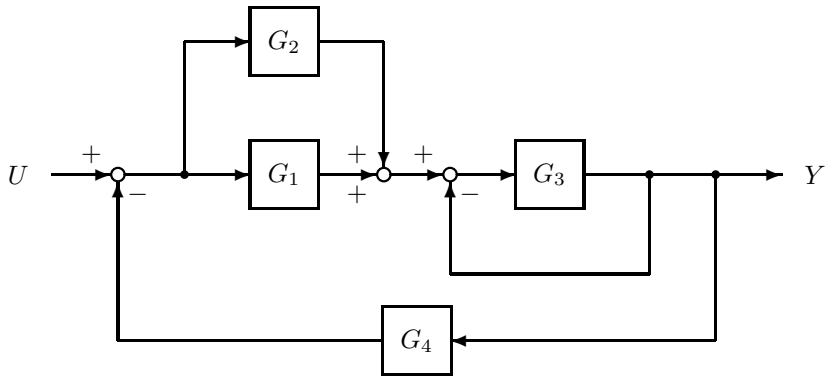

図 2.23: 問題 2 のブロック線図

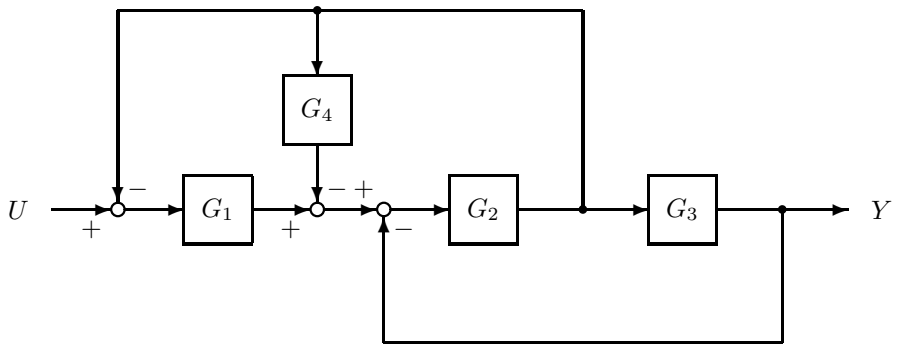

図 2.24: 問題 3 のブロック線図

2.7.3　零点，極，ゲイン

sys2zp 関数を用いて，伝達関数の零点，極，ゲインを計算できる．使用例は次のとおりである．

```
octave:1> sys = tf2sys([2 4],[1 1 2 1]);
octave:2> [z,p,k] = sys2zp(sys)
z = -2
p =
  -0.21508 + 1.30714i
  -0.21508 - 1.30714i
  -0.56984 + 0.00000i
k = 2
```

演習

1. 2.7.2 項の演習問題における伝達関数の極と零点を計算せよ．
   (答) (1)　極 $\{-6,\ -1.45340,\ -0.77330 \pm 1.46771i\}$, 零点 $\{-1\}$
   (2)　極 $\{-100.112,\ -8.53326,\ -2.41586,\ -0.46946 \pm 0.86526i\}$,
   零点 $\{-10,\ -0.51 \pm 1.31905i\}$
   (3)　極 $\{-94.7946,\ -3.0007 \pm 1.7971i,\ -1.2020 \pm 0.8432i\}$,
   零点 $\{-100,\ -0.5\}$

2.7.4　過渡応答

表 2.3: 過渡応答を計算する関数

関数	説明
`impulse`	インパルス応答
`step`	ステップ応答

システムのインパルス応答とステップ応答を求める関数を紹介する．これらの関数は，例えば，次のように使用する．

```
octave:1> sys = tf2sys(1,[1 1]);
octave:2> impulse(sys)
```

上のように書くと時間範囲と計算点数は、自動的に選択される (図 2.25) . 終了時刻を指定する場合，次のように書く．

```
octave:3> ts = 10;
octave:4> impulse(sys,1,ts)     % 1 はインパルスを与える入力番号
```

`step` 関数の使い方も同様である．

例題 **2.11**　次の伝達関数を持つシステムのインパルス応答を求めよ．

$$G(s) = \frac{1}{s^2 + 0.4s + 1}$$

(解答)

```
octave:1> sys = tf2sys(1,[1 0.4 1]);
octave:2> impulse(sys,1,30,200)      % 200 はインパルス応答のデータ数
```

インパルス応答を図 2.26 に示す．

演習

1. 2.7.2 項の演習問題における伝達関数のインパルス応答とステップ応答を求めよ．終了時刻 `ts` は応答の概形が把握できるように選定せよ．

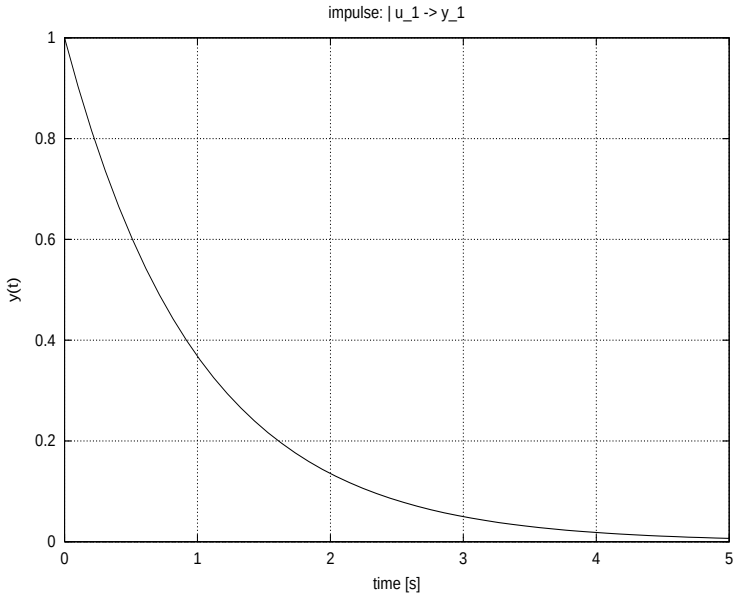

図 2.25: インパルス応答

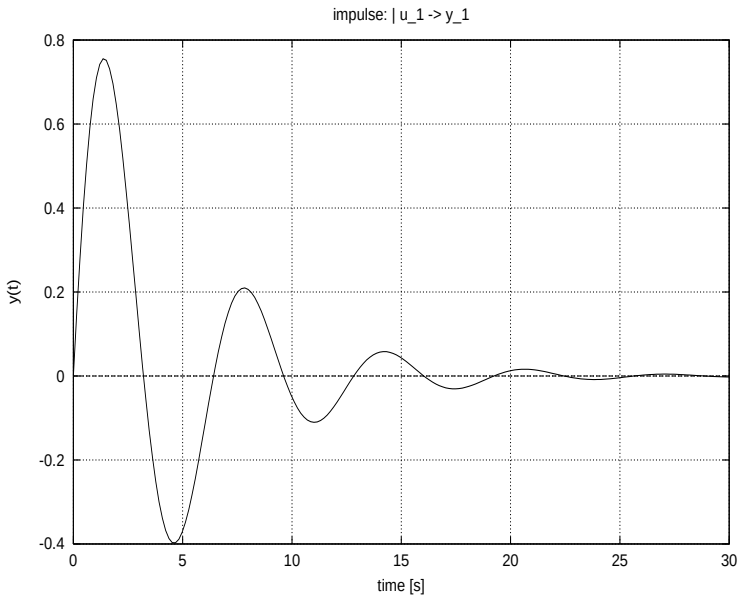

図 2.26: インパルス応答

2.7.5　周波数応答

これらの関数を用いて伝達関数

$$G(s) = \frac{1}{s^2 + 0.4s + 1}$$

表 2.4: 周波数応答を計算する関数

関数	説明
nyquist	ナイキスト線図
bode	ボード線図

のナイキスト線図とボード線図を求めよう．

```
octave:1> sys=tf2sys(1,[1 0.4 1]);
octave:2> nyquist(sys)              % 図 2.27
octave:3> bode(sys)                 % 図 2.28
```

これらの関数では，指定した周波数ベクトルに対して周波数応答を求めることもできる．

```
octave:4> w = logspace(-2,2,200);
octave:5> bode(sys,w)               % 図 2.29
```

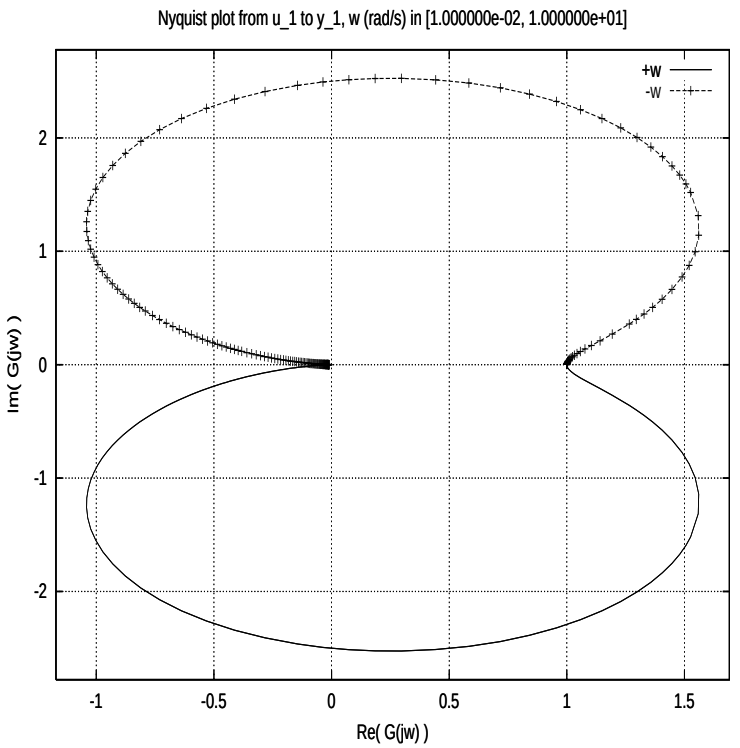

図 2.27: ナイキスト線図

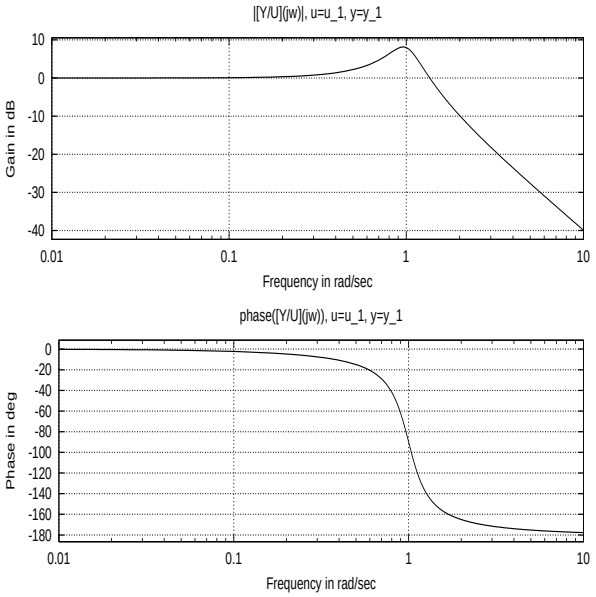

図 2.28: ボード線図

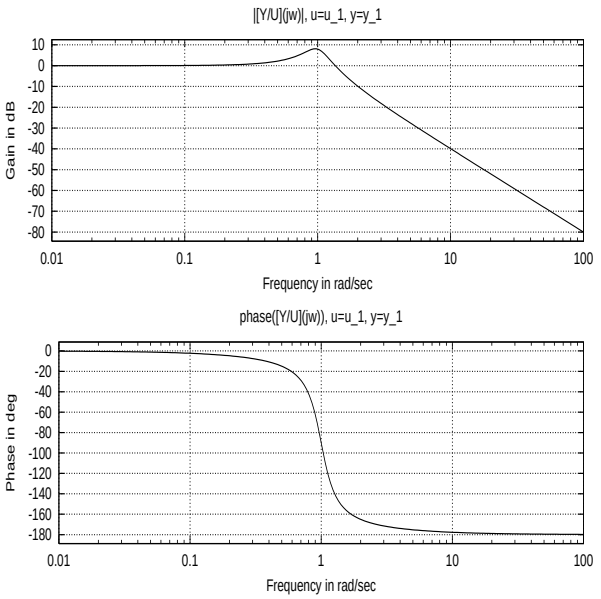

図 2.29: ボード線図

また，次の伝達関数のボード線図を求めよう．

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

```
octave:1> sys = tf2sys(1,[1 1 2 1]);
octave:2> bode(sys)                    % 図 2.30
```

位相は常に $-180°$ ~ $180°$ の範囲で計算されるので，この例のように，範囲を超えて位相が変化する場合，注意が必要である．

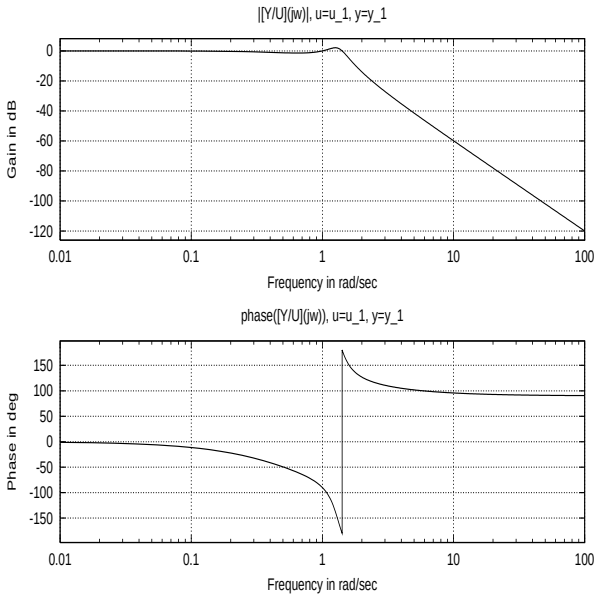

図 2.30: ボード線図

演習

1. 2.7.2 項の演習問題における伝達関数のナイキスト線図とボード線図を求めよ．

2.7.6 根軌跡

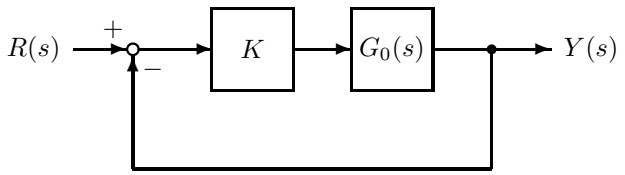

図 2.31: フィードバック系

図 2.31 のフィードバック系において，K を 0 から ∞ まで変化させたときのフィードバック系の極の軌跡を根軌跡という．根軌跡は `rlocus` 関数により計算できる．

例として

$$G_0 = \frac{0.2(s+2)}{s(s+1)(s+5)}$$

に対する根軌跡を計算する．

```
octave:1> sys = zp2sys([-2],[0 -1 -5]);
octave:2> rlocus(sys)
```

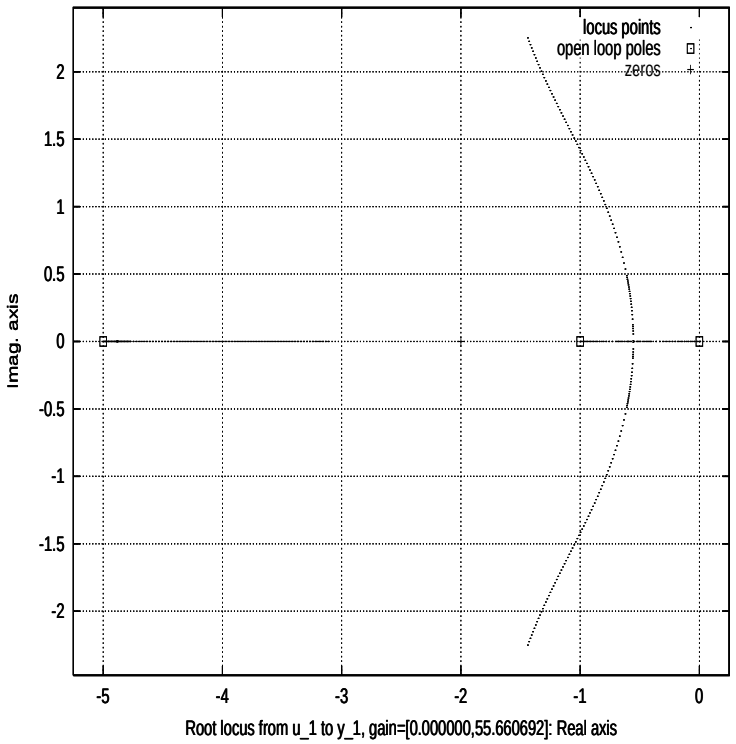

図 2.32: 根軌跡

rlocus 関数では，ゲインの増分，初期値，最終値を指定することもできる．

```
octave:3> rlocus(sys,1,0,500)
```

演習

1. 図 2.31 において $G_0(s)$ が以下のとき，フィードバック系の根軌跡を描け．

$$(1)\quad G_0(s) = \frac{s+3}{(s+1)(s+2)} \qquad (2)\quad G_0(s) = \frac{1}{s(s^2+2s+2)}$$

$$(3)\quad G_0(s) = \frac{s+1}{s(s+2)(s^2+2s+2)} \qquad (4)\quad G_0(s) = \frac{1}{(s+1)(s+2)(s+3)(s+4)}$$

2.7.7 制御系の設計例

ＤＣサーボモータの速度フィードバック制御

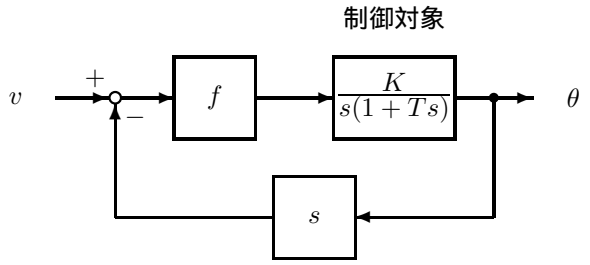

図 2.33: ＤＣサーボモータ制御系

図 2.33 にＤＣサーボモータ制御系を示す．制御対象の $K/\{s(1+Ts)\}$ は，ＤＣサーボモータ，モータドライバ，減速ギア系から構成されている．θ は出力角 v は目標角速度である．v から θ までの伝達関数を計算すると次式となる．

$$G(s) = \frac{1}{\frac{T}{fK}s^2 + \frac{1+fK}{fK}s} \tag{2.66}$$

よって，f を十分大きく設定した場合

$$G(s) \simeq \frac{1}{s} \tag{2.67}$$

となり，制御系はほぼ積分要素して振舞う．

実際の設計では，フィードバック回路の微分要素 s は，実現可能な次の近似微分で置き換える．

$$\frac{s}{1+T_d s}, \quad 0 < T_d \ll 1 \tag{2.68}$$

パラメータを

$$K = 15,\ T = 0.2,\ T_d = 0.01,\ f = 0.1,\ 10$$

と与えた場合の制御系のボード線図を図 2.34，図 2.35 に示す．$f = 10$ の場合，$\omega < 10\mathrm{rad/s}$ の低周波域でゲイン曲線の勾配が $-20\mathrm{dB/dec}$，位相が約 $-90°$ となり，この周波数域で積分要素を近似していることがわかる．

```
octave:1> K = 15; T = 0.2; Td = 0.01;
octave:2> f = 0.1;
octave:3> s1 = tf2sys(f,1);
octave:4> s2 = tf2sys(K,[T 1 0]);
```

```
octave:5> s3 = tf2sys([1 0],[Td 1]);
octave:6> s4 = sysmult(s1,s2);
octave:7> sys = feedback(s4,s3);          % p61 で定義した関数
octave:8> w = logspace(0,4,300);
octave:9> bode(sys,w)                     % 図 2.34
octave:10> f = 10;
octave:11> s1 = tf2sys(f,1);
octave:12> s4 = sysmult(s1,s2);
octave:13> sys = feedback(s4,s3);
octave:14> bode(sys,w)                    % 図 2.35
```

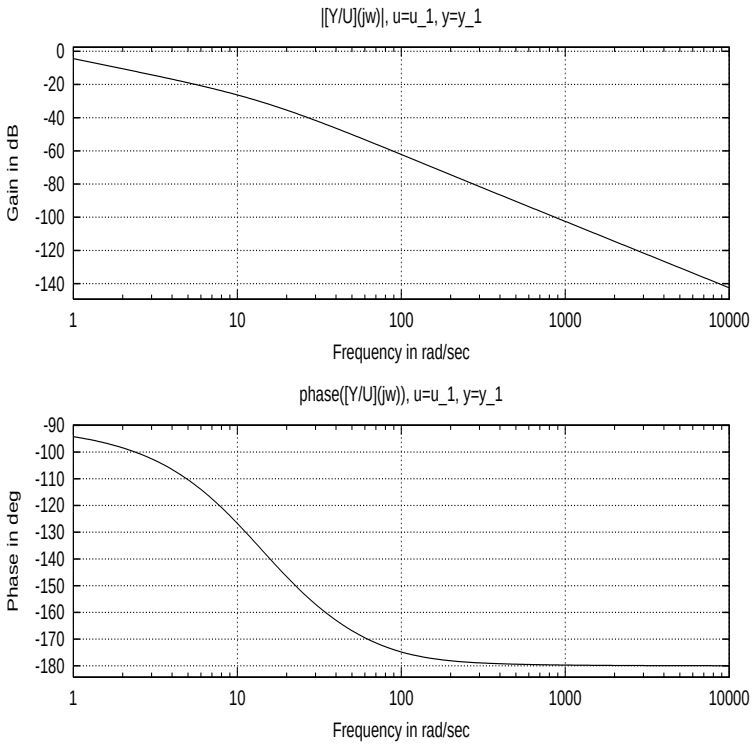

図 2.34: ボード線図 ($f = 0.1$)

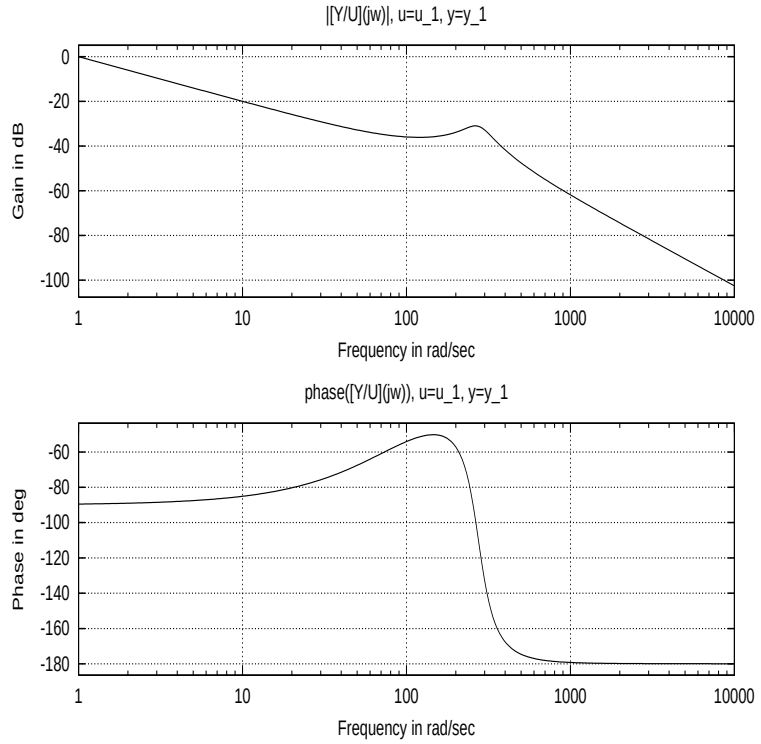

図 2.35: ボード線図 ($f = 10$)

PID 制御系の設計

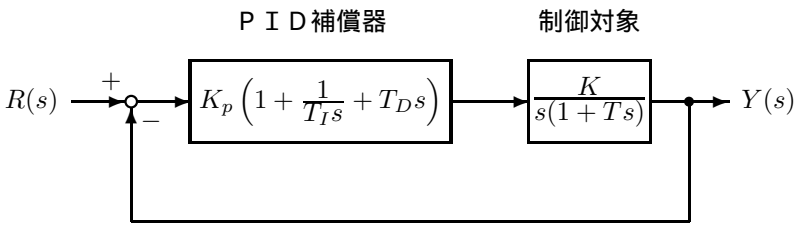

図 2.36: ＰＩＤ制御系

図 2.36 において，制御対象のパラメータは

$$K = 15, \quad T = 0.2$$

とする．以下では，ジーグラ・ニコルスの過渡応答法によってＰＩＤ補償器のパラメータを設計する．

制御対象のステップ応答曲線に対して最大勾配の接線を引くとき，接線の勾配を R 接線が時間軸と交わる時刻を L とする．過渡応答法は，R, L によりＰＩＤパラメータを次のように決める方法である．

$$K_P = \frac{1.2}{RL}, \quad T_I = 2L, \quad T_D = 0.5L \tag{2.69}$$

この制御対象の場合，R, L は次式で決定できる．

$$R = K, \quad L = T \tag{2.70}$$

シミュレーションを行うため，ＰＩＤ補償器の微分要素 s を

$$\frac{s}{1 + T_d s}, \quad T_d = 0.01 \tag{2.71}$$

で近似する．このとき，ＰＩＤ補償器の伝達関数は次式となる．

$$\begin{aligned} C(s) &= K_p \left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + T_d s} \right) \\ &= K_P \frac{T_I (T_d + T_D) s^2 + (T_I + T_d) s + 1}{T_I T_d s^2 + T_I s} \end{aligned} \tag{2.72}$$

制御系のステップ応答を図 2.37 に，また，ボード線図を図 2.38 に示す．

```
octave:1> K = 15; T = 0.2; Td = 0.01;
octave:2> R = K; L = T;
octave:3> KP = 1.2/(R*L); TI = 2*L; TD = 0.5*L;
octave:4> s1 = tf2sys(KP,1);
octave:5> s2 = tf2sys([TI*(Td+TD) TI+Td 1],[TI*Td TI 0]);
octave:6> s3 = sysmult(s1,s2);
octave:7> s4 = tf2sys(K,[T 1 0]);
octave:8> s5 = sysmult(s3,s4);
octave:9> s6 = ugain(1);
octave:10> sys = feedback(s5,s6);
octave:11> step(sys)              % 図 2.37
octave:12> bode(sys)              % 図 2.38
```

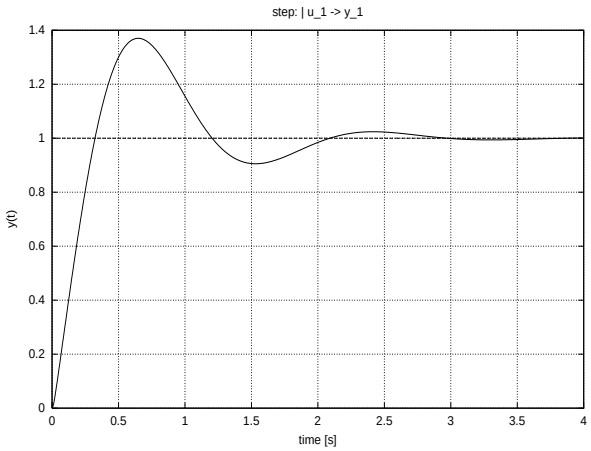

図 2.37: ＰＩＤ制御系のステップ応答

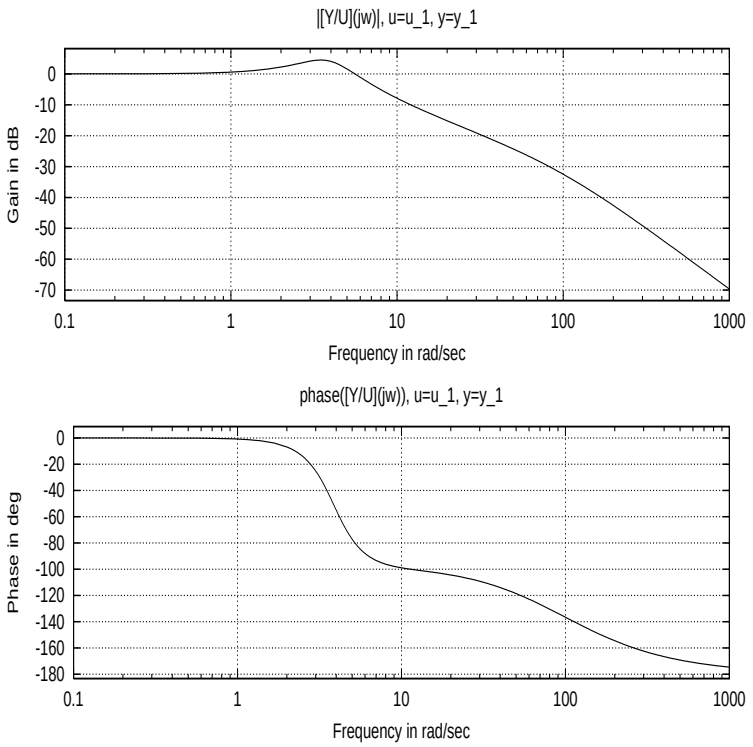

図 2.38: ＰＩＤ制御系のボード線図

関連図書

[1] 吉田和信：MATLAB による動的システムシミュレーション入門，
http://www.ecs.shimane-u.ac.jp/ kyoshida/matlab.htm, 2001.

[2] John W. Eaton: GNU Octave A high-level interactive language for numerical computations Edition 3 for Octave version 2.0.5,
http://www.octave.org/doc/octave_toc.html, 1997.

[3] 佐久間元敬他：線形代数教科書，共立出版，1978.

[4] 志賀浩二：数学 30 講シリーズ 1 微分積分 30 講，朝倉書店，1988．

[5] 伊理正夫他：別冊・数学セミナー 現代応用数学の基礎 1 ，日本評論社，1987.

[6] 吉田和信他:重心移動による振子系の振動制御，システム制御情報学会論文誌，pp.470-477, 2000.

[7] 加藤寛一郎：最適制御入門 レギュレータとカルマンフィルタ，東京大学出版会，1987.

[8] 日高照晃他：機械力学–振動の基礎から制御まで–，朝倉書店，2000.

[9] 奥山佳史他：制御工学–古典から現代まで–，朝倉書店，2001.

[10] 浪速智英：Octave/Matlab で見るシステム制御，科学技術出版，2000.

Octave による動的システムシミュレーション入門

2002 年 2 月 1 日　初版

ⓒ吉田和信