第 5 章

信号処理

本章では, 信号処理, 特に低域通過フィルタ (LPF: Low Pass Filter) に関する実験を行う. センサを用いて物理量を計測する場合, 観測量には必ずと言ってよいほど雑音が重畳する. 雑音に埋もれた観測量から信号成分を抽出するのが LPF の役割である. 本章では, ロータリエンコーダを用いたモータの回転速度計測における信号処理の実験を行う. 信号処理は, 前期のセンサ実験や後期のロボット実験で使用するほとんどのセンサで必要となる技術なので, 応用できるレベルまで理解度を深めてほしい.

5.1 ロータリエンコーダの概要

今回はホールセンサを利用したインクリメンタル型のロータリエンコーダを使用する. ここでは, ホールセンサの動作原理と, ロータリエンコーダの概要を簡単にまとめる.

5.1.1 ホールセンサ

ホールセンサとは, ホール効果を利用して磁界を検出するセンサである. 磁場 B の中を動く電荷 (電流 i) はローレンツ力 F_L を受ける (図 5.1). フレミングの左手の法則より, 電荷は手前の電極の方向への力を受ける. P 型半導体の場合, 多数キャリアが正孔 $(+)$ なので, 電極 E_a が正に帯電する. N 型半導体の場合, 多数キャリアが電子 $(-)$ なので, 電極 E_a が負に帯電する. よって, 電極 E_a と 電極 E_b の間に起電力 V_H (ホール電圧) が発生する (ホール効果). 実験で使用するホールセンサ DRV5023A[3] はアンプを内蔵しており, このホール電圧 V_H を増幅し, 2 値化したものを出力とするので, Arduino[2] のディジタルポートから直接読み込むことができる.

5.1.2 ロータリエンコーダ

ロータリエンコーダは, 角度や角速度を計測するセンサである. 物理量を計測するセンサとスケールの組み合わせはセンサ位置を 2 進数のようなディジタル符号に符号化するので, エンコーダと呼ばれる.

高速で回転する場合は, 物理量を非接触で計測するのが望ましいので, 光学センサや磁気センサが用いられることが多い. 光学センサを用いる光電方式では, さらに, 透過型, 反射型などに分かれる. 磁気センサを用いる磁気方式では, ギヤの歯を検出するものもある. 本実験では, 磁石を回転子に取り付け磁場の向きを磁気センサで計測することにする. 絶対的な角度を計測するアブソリュート型 (図 5.2) と, 角度の増減だけを計測するインクリメンタル型に分けられる. 通常, アブソリュート型では, その解像度に合わせた数だけのセンサが必要である.

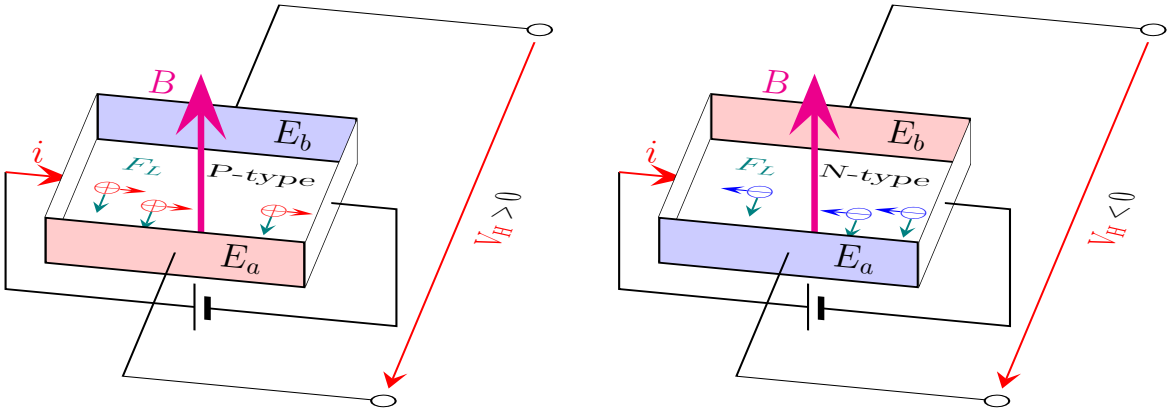

図 5.1　ホール効果 (P 型半導体 (左) と N 型半導体 (右) の場合)

Gray コード

アブソリュート型では, 通常の 2 進数 (BCD: Binary Coded Decimal) ではなく, 隣接する符号が 1bit だけ異なる Gray コード (図 5.3) が用いられることが多い. 例えば, 4bit の BCD だと 0 と 15 の間は 4bit が一斉に変化するが, センサの設置位置が少しずれていると, どの bit が先に変化してもおかしくない. そうすると, あたかも, 0 と 15 の間にそれら以外の数が存在するように見えてしまう. Gray コードではそのようなことは起こらない.

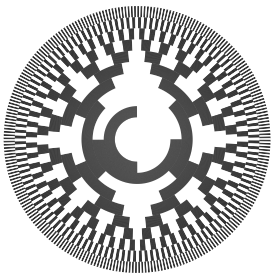

図 5.2　アブソリュート型ロータリエンコーダ (10bit)

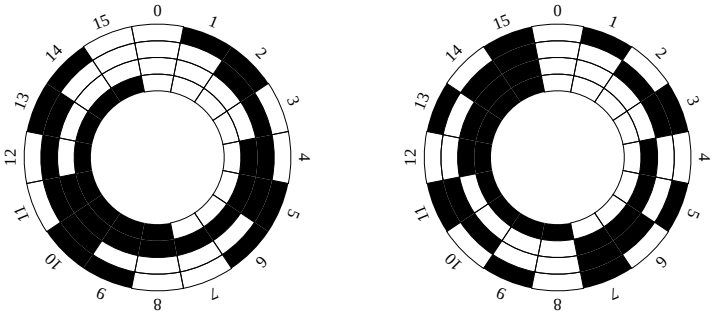

図 5.3　Gray コード (左) と BCD(右) (4bit の場合)

インクリメンタル型における増減方向の弁別

インクリメンタル型のロータリエンコーダにおいて, センサを 2 つ, 1/4 波長ずらした点 (A 点と B 点) に配置することで, 増減方向を検出することができる (図 5.4). 時計回りに回転しているとき (図 5.4 左) A 点が白から黒に変わる瞬間, B 点は黒である. 一方, 反時計回りに回転しているとき (図 5.4 右) A 点が白から黒に変わる瞬間, B 点は白である. このようにセンサを 2 つ 1/4 波長ずらした点に配置することによって, 回転方向を検出することができる.

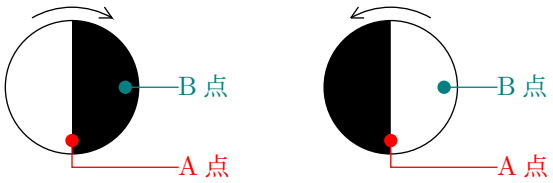

図 5.4　インクリメンタル型ロータリエンコーダの増減方向の弁別

実験で使用するロータリエンコーダ

本実験で使用するロータリエンコーダは, 磁気センサを用いたインクリメンタル型であり, 回転方向の検出は行わない. 磁気センサは 1 つだけ使用する. 回転子に磁石が埋め込まれており, 回転子の角度に応じてホールセンサを貫く磁束の向きが変わる (図 5.5) ので, これを検出して回転速度を検出する.

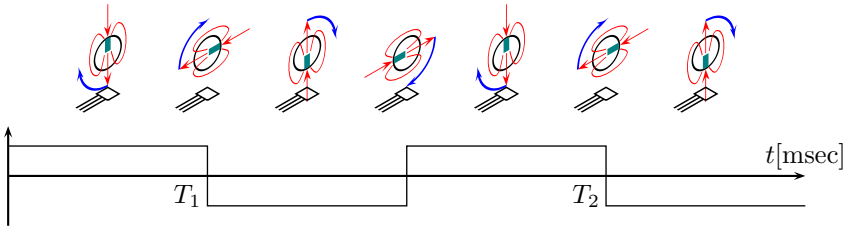

図 5.5　実験で使用するロータリエンコーダ

ホールセンサ DRV5023A の出力はオープンドレインとなっているので, 10kΩ の抵抗で `+5V` にプルアップしておいてから, Arduino の割込みポート (ここでは, `D2`) に接続し (図 5.6), κ 回目の割り込みの発生した時刻 T_κ[msec] を計測する (図 5.5).

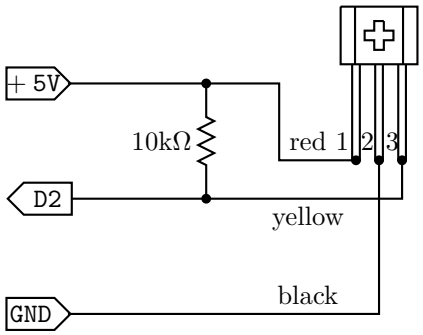

図 5.6　ホールセンサ回路

モータやエンジンなどの回転速度を表すとき, 1 分間に回転した回数を表す rpm (rotations per minute) という単位がよく用いられる. 1 秒間当たりの回転数 sec^{-1} に変換するには 60[sec] で割ればよい. ($60\mathrm{rpm} = 60\mathrm{min}^{-1} = 1\mathrm{sec}^{-1} = 1\mathrm{Hz}$.) モータの回転速度 ϕ_κ[rpm] は次式で与えられる.

$$\phi_\kappa[\mathrm{rpm}] = \frac{1000[\mathrm{msec/sec}] \times 60[\mathrm{sec/min}]}{(T_\kappa - T_{\kappa-1})[\mathrm{msec/rotation}]}, \tag{5.1}$$

ここで, T_κ は (最新の) 割り込みが発生した時刻 [msec], $T_{\kappa-1}$ は 1 回前の割り込みが発生した時刻 [msec] である. ϕ_κ を `float` 型の大域変数として定義しておき, 割り込みが発生するたびに, ISR(Interrupt Service Routine) の中で上式を計算すれば, モータの回転速度を得ることができる. 時刻 t における ϕ_κ の値を $\phi(t)$ と

おくと

$$\phi(t) = \phi_\kappa \qquad \text{for} \qquad T_\kappa \le t < T_{\kappa+1}. \tag{5.2}$$

また, $\phi(t)$ をサンプリング周期 T_s でサンプルした値を $\phi[k]$ と書くことにする:

$$\phi[k] = \phi(kT_\mathrm{s}). \tag{5.3}$$

雑音対策

DC モータが回転していると, その構造上スパイク状の雑音が発生する. すなわち, δ 関数的な雑音が一瞬だけ信号に重畳する. 割込みにより D2 ポート電圧の変化を監視する場合, この一瞬だけの雑音でも割込みが発生してしまう. そこで, 割込みが発生して ISR が呼ばれたとき, ISR の中で, もう一度 D2 ポートを読んで実際にポート電圧が変化しているかどうかを確認し, 実際に電圧が変化しているときだけ, ϕ_κ を計算するようにする (リスト 5.1).

5.2　信号処理とディジタルフィルタ

信号処理の基礎としてローパスフィルタを簡単に解説する. また, FIR フィルタと IIR フィルタについても簡単にまとめる.

5.2.1　ローパスフィルタ

雑音や量子化誤差の影響を低減するため, しばしば, 読み取ったセンサ値を平滑化してから利用する. 一般に雑音のパワースペクトルは低周波領域から高周波領域まで万遍なく分布しているのに対して, 信号の周波数成分は低周波領域にのみ分布していることが多い. そのため, 雑音の低減にはローパスフィルタ (LPF: Low Pass Filter, 低域通過フィルタ) を用いることが多い.

図 5.7に示す duty 比を PWM 変調した電圧をモータに入力したときの回転速度 $\phi(t)$[rpm] を図 5.8に示す. (演習 5.2を参照) Duty 比を制御入力 $u(t)$ とし, 出力を $\phi(t)$ とする. 図 5.8の青い点線 ($\cdots$) が平滑化する前のデータである. 量子化誤差の影響が大きいことがわかるであろう. ホールセンサの割り込みはモータの回転に同期して発生するので, 2000[rpm] $\sim$ 3000[rpm] で回転しているとき, 33.3[Hz] $\sim$ 50[Hz] で割り込みが発生している. 対象 (モータ) の時定数 T_M は, このグラフより約 0.5 秒程度である. 雑音除去のための LPF の帯域幅は対象の帯域幅 (= 時定数の逆数) の数倍はほしいので, LPF の時定数を $T_\mathrm{c} = 1/8$ とすると, LPF の伝達関数 $F(s)$ は

$$F(s) = \frac{1}{1 + T_\mathrm{c}s} = \frac{1}{1 + 0.125s} \tag{5.4}$$

となる. ここで, s はラプラス変換の s であり, 初期値の影響を無視すれば微分作用素 d/dt と思ってもよい. これを 33.3[Hz] で離散化すると,

$$F_{d1}(z) = \frac{1}{4.6866 - 3.6866z^{-1}} \tag{5.5}$$

となり, 50[Hz] で離散化すると,

$$F_{d2}(z) = \frac{1}{3.0332 - 2.0332z^{-1}} \tag{5.6}$$

となる. ここで, z は進み作用素であり, 離散時間信号 x_κ ($\kappa = 0, 1, 2, \ldots$ はサンプル番号 (サンプリング周期は一定でなくてもよい)) に対して, $zx_\kappa = x_{\kappa+1}$ である. (付録 Aを参照.) 一般につぎの形のフィルタ:

$$F_d(z) = \frac{1}{\alpha - (\alpha - 1)z^{-1}} \tag{5.7}$$

リスト 5.1 ロータリエンコーダによる回転速度の計測

```
#include <MsTimer2.h>
const int D2 = 2;                          // 外部割込みピンの定義
const int D6 = 6;                          // PWM 出力ディジタルピンの定義
float dutyH=0.25, dutyL=0.2, duty=dutyL; // duty 変数の定義．duty 比変更
boolean isHigh=false;
int Ts = 40;                               // サンプリング周期 [msec]
int iteration = 0;
unsigned long timep = 0;
volatile float motrpm = 0;
//volatile float motrpmf = 0;
volatile boolean isD2High=false;
void countUp() {                           // 外部割込み (D2) の ISR
  if ((isD2High)&&(digitalRead(D2)==LOW)) {   // 雑音対策: 本当に LOW か?
    unsigned long timen = millis();
    motrpm = 1000.0*60.0/(float)(timen-timep);
//    motrpmf = (3*motrpmf+motrpm)/4.0;
    timep=timen;
    isD2High=false;
  } else if ((!isD2High)&&(digitalRead(D2)==HIGH)) {    // 本当に HIGH か?
    isD2High=true;
  }
}
void control() {                           // タイマ割込み (MsTimer2) の ISR
  iteration++;
  if (iteration>=100) {                    // 4[sec] ごと
    iteration=0;
    if (isHigh) {
      isHigh=false;
      duty = dutyL;
    } else {
      isHigh=true;
      duty =  dutyH;
    }
  }
  analogWrite(D6,duty*255);
  Serial.println((unsigned)motrpm);
}
void setup() {
  analogWrite(D6,dutyH*255);
  attachInterrupt(0,countUp,CHANGE);     // 外部割込みピン:D2
  MsTimer2::set(Ts, control);            // 40[msec] ごと
  MsTimer2::start();
  Serial.begin(115200);
}
void loop() {}
```

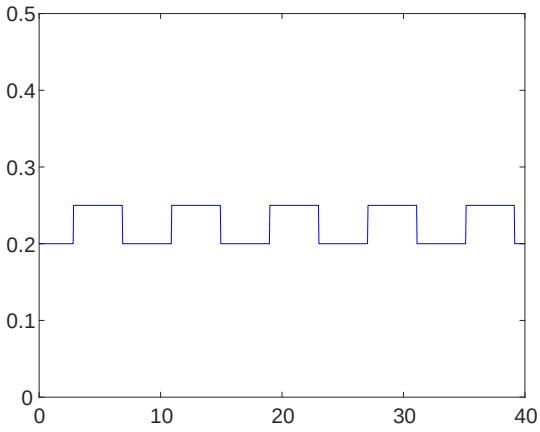

図 5.7　モータの Duty 比 $u(t)$,
横軸:t[sec], 縦軸:$u(t)$

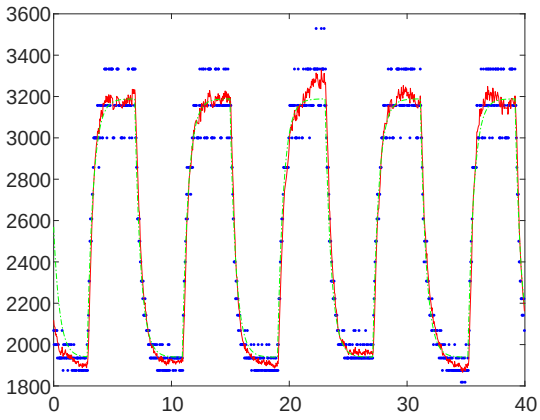

図 5.8　モータの回転速度,
横軸:t[sec], 縦軸:$\phi(t)$ [rpm](···),
$y(t)$ [rpm](—) $\hat{y}(t)$[rpm](-·-)

はゲイン 1 の 1 次の LPF である. これに ϕ_κ を通した出力を

$$y_\kappa = F_d(z)\phi_\kappa \tag{5.8}$$

とすると, $F_d(z)$ はつぎのように実装される.

$$y_\kappa = \frac{\alpha-1}{\alpha}y_{\kappa-1} + \frac{1}{\alpha}\phi_\kappa \tag{5.9}$$

割り算の際に数値計算の精度がなるべく落ちないように, α は 2 のべき乗に選ぶことが多い. そこで, 今回は $\alpha = 4$ と選ぶことにする. フィルタの初期値を $y_0 = 0$ とすると,

$$y_\kappa = \frac{3}{4}y_{\kappa-1} + \frac{1}{4}\phi_\kappa = \frac{1}{4}\sum_{i=0}^{\kappa}\left(\frac{3}{4}\right)^i \phi_{\kappa-i}. \tag{5.10}$$

すなわち, フィルタの出力 y_κ は, 過去の観測値 $\phi_{\kappa-i}$ に指数的に減衰する重み $(3/4)^i/4$ をかけた, 重み付き平均となる. (直近の観測値の重みが大きく, 過去に遡るほど重みは小さくなる.)

(5.2) 式と同様に, 時刻 t における y_κ の値を $y(t)$ とすると, 図 5.8の赤線となる.

5.2.2　FIR フィルタと IIR フィルタ *

ディジタルフィルタはその構造から, FIR(Finite Impulse Response) フィルタと IIR(Infinite Impulse Response) フィルタに分類できる. 低域通過フィルタ (LPF: Low Pass Filter) だけでなく, 高域通過フィルタ (HPF: High Pass Filter), 帯域通過フィルタ (BPF: Band Pass Filter) やさらに一般的なフィルタも実現できる.

FIR フィルタの出力は, 過去 n 時刻前までの入力の移動平均:

$$y[k] = b_0u[k] + b_1u[k-1] + \cdots + b_nu[k-n] \tag{5.11}$$

で表される. $u[k]$, $y[k]$ の z 変換を $U(z)$, $Y(z)$ とおくと FIR フィルタのパルス伝達関数 ($Y(z) = G(z)U(z)$ となる $G(z)$) は

$$G(z) = b_0 + b_1z^{-1} + \cdots + b_nz^{-n} \tag{5.12}$$

で与えられる. 一方, IIR フィルタの出力は, 過去 n 時刻前までの入力と出力の移動平均:

$$\begin{aligned} y[k] = &-a_1y[k-1] - \cdots - a_ny[k-n] \\ &+b_0u[k] + b_1u[k-1] + \cdots + b_nu[k-n] \end{aligned} \tag{5.13}$$

で表され, そのパルス伝達関数は

$$G(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{5.14}$$

で与えられる. 一般に FIR フィルタの次数 n は IIR フィルタの次数に比べ非常に大きく, 数十から数百という値をとることが多い. 組み込みマイコンでディジタルフィルタを実装する場合などは, IIR フィルタを使うことが多い. 小節 5.2.1で導出した LPF $F_d(z)$ は 1 次の IIR フィルタである.

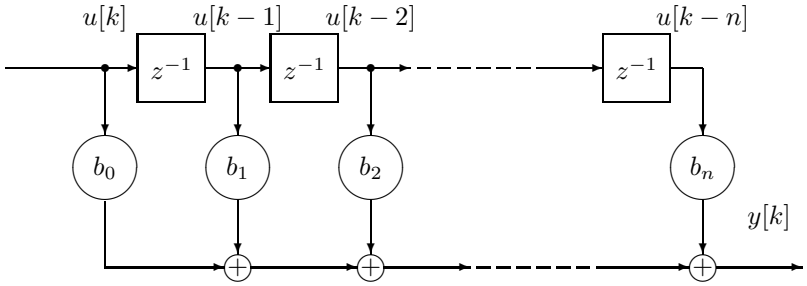

図 5.9　FIR フィルタの回路実現

FIR フィルタの実現を図 5.9に示す. z^{-1} の数 n だけメモリが必要である. 一方, IIR フィルタの実現の仕方は複数 (無限に) あり, ここでは, 直接型 II 構造 (図 5.10) と状態空間表現 (図 5.11) の 2 つを紹介する. 状態空間表現では, $b_i' = b_i - b_0 a_i$ $(i = 1, \ldots, n)$ である. $x[k] = (x_1[k], \ldots, x_n[k])^\top$ を状態変数ベクトルという.

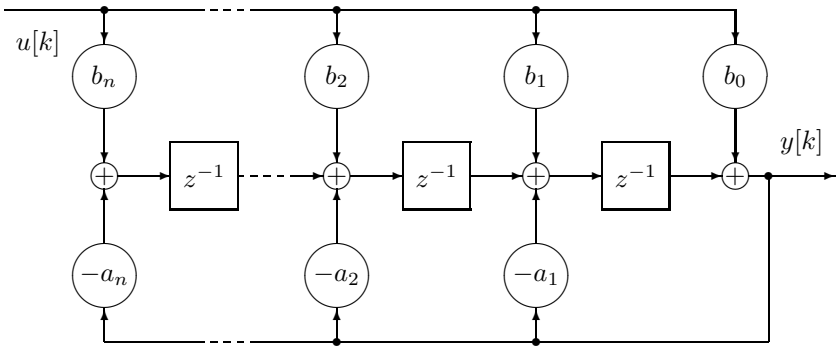

図 5.10　IIR フィルタの回路実現 (直接型 II 構造)

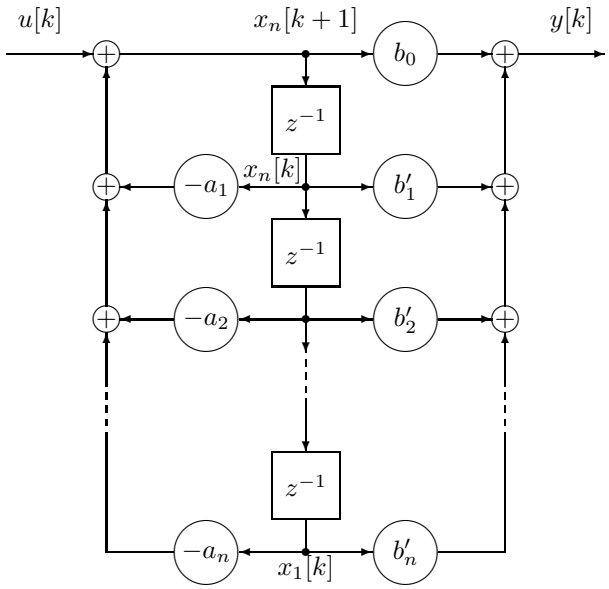

図 5.11　IIR フィルタの回路実現 (状態空間表現)

高次の IIR により LPF, HPF, BPF などを設計する場合, アナログフィルタをもとにディジタルフィルタに変換する方法が一般的である. アナログフィルタの設計法としては, バタワース (Butterworth) フィルタ, チェビシェフ (Chebyshev) フィルタ (I 型, II 型), 楕円フィルタなどがある. チェビシェフフィルタでは, I 型は通過帯域に, II 型は阻止帯域に, 楕円フィルタでは通過帯域・阻止帯域ともにリップルが発生するが, その分, 通過帯域と阻止帯域の境界のゲインの傾きが大きくなっている. 高次にすればするほど境界のゲインの傾きは急峻になる. アナログフィルタの実装では, 次数に応じた回路素子が必要になるため, 高次のフィルタは実装が難しくなる. 一方, ディジタルフィルタでは次数を増やすのは比較的簡単なので, バタワースフィルタがよく用いられる.

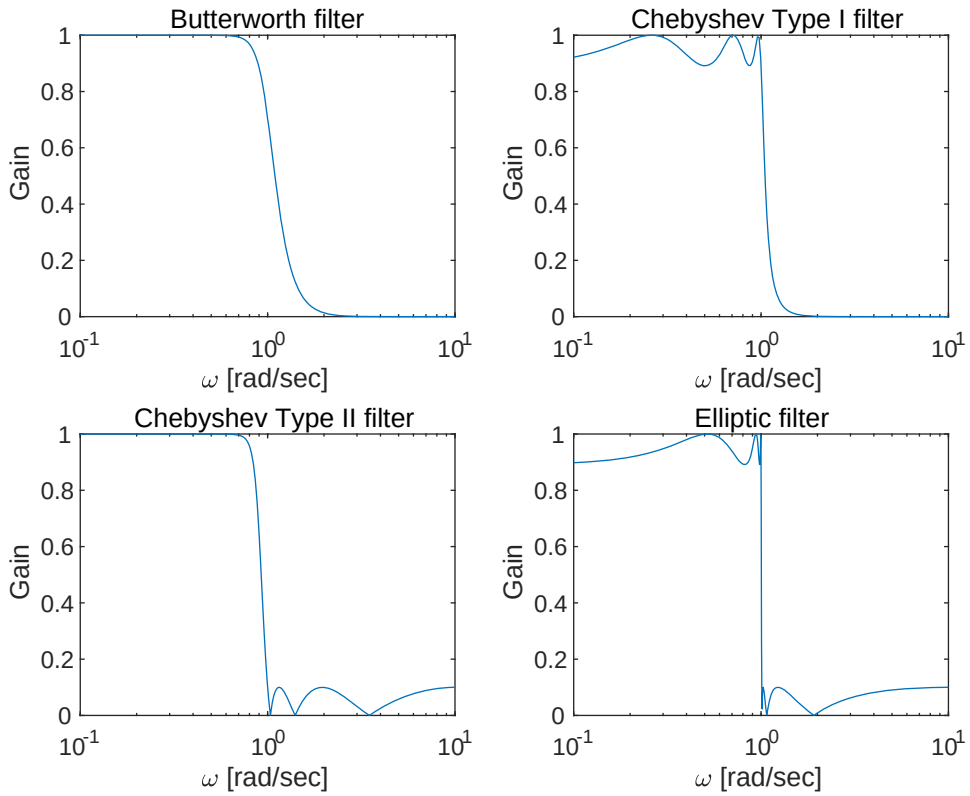

図 5.12　例 6 次の LPF の周波数応答

1 次の LPF

$$F_{d1}(z) = \frac{1}{4 - 3z^{-1}} \tag{5.15}$$

の帯域幅 ($|F_{d1}(e^{2\pi f_c j})| = 1/\sqrt{2}$ となる f_c[Hz]) と同じ帯域幅をもつ 2 次のバタワースフィルタは

$$F_{d2}(z) = \frac{0.0173 + 0.0345z^{-1} + 0.0173z^{-2}}{1 - 1.5955z^{-1} + 0.6646z^{-2}} \tag{5.16}$$

となる. Scilab などの CAD システムを使うと簡単に求めることができる. $|F_{d1}(e^{2\pi f_c j})| = 1/\sqrt{2}$ となる f_c は

$$\frac{1}{2} = \frac{1}{(4 - 3e^{-2\pi f_c j})(4 - 3e^{2\pi f_c j})} = \frac{1}{16 - 12(e^{2\pi f_c j} + e^{-2\pi f_c j}) + 9} = \frac{1}{25 - 24\cos 2\pi f_c}$$

より, $\cos 2\pi f_c = (25 - 2)/24$. Unix の端末ソフト上から,

```
% scilab &
```

と Scilab を立ち上げたのち, Scilab コンソール上で,

```
-->fc = acos((25-2)/24)/2/%pi;
-->Fd2 = iir(2,'lp','butt',fc,[0,0])
```

とすれば, (5.16) を得る. `iir` の詳細は, `help iir` とすればヘルプが出てくる. $F_{d2}(z)$ のゲイン線図を図 5.13に示す. 横軸は正規化された周波数 (Nyquist 周波数が 0.5) である.

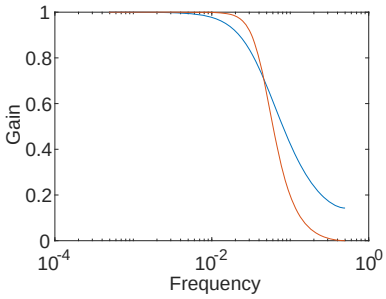

図 5.13　1 次 (青) と 2 次 (赤) の LPF のゲイン

5.3　演習と実験

5.3.1　モータの PWM 制御

PWM(Pulse Width Modulation; パルス幅変調) 制御によりモータの回転速度を調整する実験を行う. PWM 制御の詳細については, 次週, 「第 6 章 フィードバック制御」で勉強する. ここでは, PWM 制御は, モータに印加する平均的な電圧をアナログ的に調整する仕組みだと理解すれば十分であろう. また, マイコンからはモータを駆動するほどの大電流を流すことはできないので, マイコンからの On/Off の指令に従って FET(Field Effect Transistor, 電界効果トランジスタ) を On/Off し, 大電流を制御する必要があることを承知しておくこと.

⟨演習 5.1⟩ モータの PWM 制御実験

マイコン信号によるモータの直接駆動は, I/O ポートの電流容量制限を超えるため不可である. それを回避するため, マイコンとモータ間にインターフェース (アンプ) を入れる. ここでは, パワー MOS N チャンネル FET のソース接地型スイッチング回路を使う. ゲートが high/low のときドレイン-ソース間が on/off となる. 以下の手順でマイコンによるモータ駆動の基礎を演習する.

1. 図 5.14の電池以外の回路をブレッドボード上に作成し, マイコンの I/O ポート D6 と GND を接続せよ. 必ず, モータと並列に**コンデンサ** (0.1μF) を入れること. 最後にモータ用の電源 (電池) の**極性**を間違えずにブレッドボードに接続せよ (図 5.16). (FET のソース接地回路 (図 5.14) では, ソースがグラウンド側, ドレインが + 電源側でなければならない.)

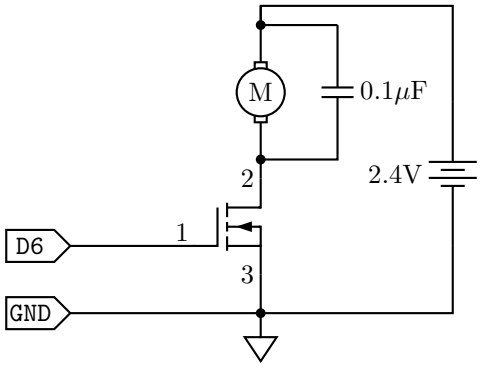

図 5.14　FET を用いたモータドライバ回路

1: ゲート
2: ドレイン
3: ソース

1 2 3

図 5.15　2SK2232 のピン配置

リスト 5.2　プログラム例 1

```
const int D6 = 6;        // PWM 出力ディジタルピンの定義
float duty = 0.2;        // duty 変数の定義. duty 比変更

void setup() {analogWrite( D6 , duty * 255 );}
                         // pinMode() は省略可. PWM 設定が優先される.
void loop() {}
```

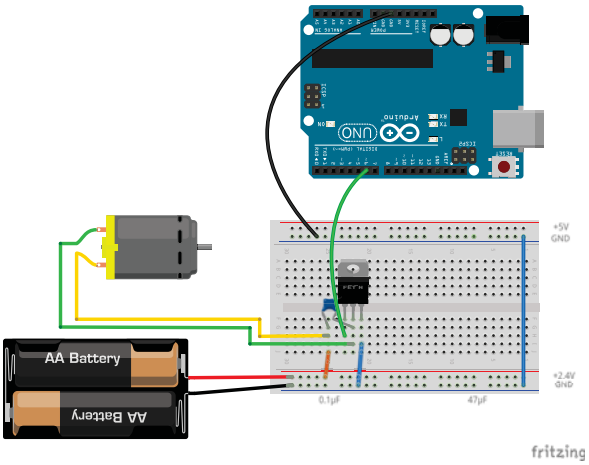

図 5.16　図 5.14の回路のブレッドボード上での実装

2. マイコンの D6 ピンから duty 比 25% および 20% の PWM 信号を加えたとき, モータの速度がどのように変化するか確認せよ (リスト 5.2). モータが回転しなかった場合は, その理由を考察せよ.
3. 4 秒ごとに duty 比を 25% と 20% に交互に変更するプログラムを作成し, その動作を確認せよ. その際, 後の実験のため, MsTimer2 を用いて 40[msec] ごとにタイマ割込みを発生させ, ISR (Interrupt Service Routine) が 100 回呼び出されるごとに duty 比を変更するようにせよ (リスト 5.3).

リスト 5.3　プログラム例 2

```
#include <MsTimer2.h>
const int D6 = 6;                          // PWM 出力ディジタルピンの定義
float dutyH=0.25, dutyL=0.2, duty=dutyL;// duty 変数の定義．duty 比変更
boolean isHigh=true;
int iteration=0;
int Ts=40;                                 // サンプリング周期 [msec]
void control() {                           // タイマ割り込み (MsTimer2) の ISR
  iteration++;
  if (iteration>=100) {                    // 4[sec] ごと
    iteration=0;
    if (isHigh) {
      isHigh=false;
      duty = dutyL;
    } else {
      isHigh=true;
      duty = dutyH;
    }
  }
  analogWrite(D6,duty*255);
}
void setup() {
  analogWrite(D6,0);
  MsTimer2::set(Ts, control);              // 40[msec] ごと
  MsTimer2::start();
}
void loop() {}
```

タイマー割り込みで不具合が生じた場合は，リスト 5.4 を参考に loop() 関数から millis() 関数を使って 40msec ごとに control() を呼び出すようにせよ．

リスト 5.4　プログラム例 2 の修正 (MsTimer2 を使わない場合)

```
//#include <MsTimer2.h>          // コメントアウト or 削除
unsigned long timePrev=0;        // 大域変数の追加
...
void setup() {
  analogWrite(D6,0);
//  MsTimer2::set(Ts, control); // コメントアウト or 削除
//  MsTimer2::start();          // コメントアウト or 削除
}
void loop() {                    // 変更
  unsigned long timeNow=millis();
  if (timeNow-timePrev>=Ts) {
    control();
    timePrev=timeNow;
  }
}
```

5.3.2　モータの回転速度の計測

〈演習 5.2〉モータ回転速度の計測

ホールセンサの出力によって外部割込みを発生させ, 何ミリ秒ごとに割込みが発生しているか計測することにより, モータの回転速度を計測する.

1. 図 5.6のホールセンサ回路を図 5.14のモータドライバ回路に追加してブレッドボード上に作成し, マイコンの入力ポート D2, GND, Vcc を接続せよ (図 5.17). また, 回転板についている磁石がホールセンサの上を通過するようにホールセンサが配置されているか確認せよ.

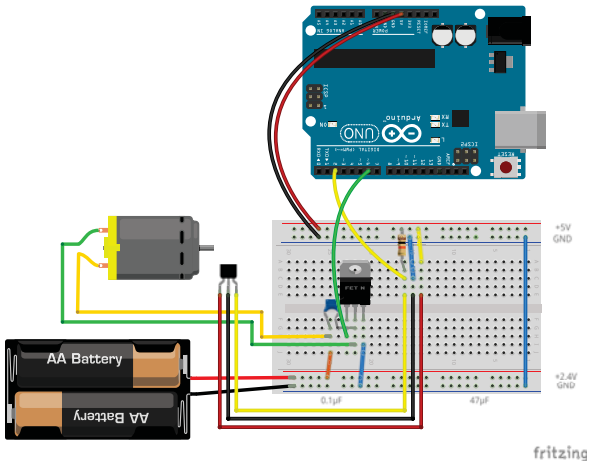

図 5.17　図 5.14と図 5.6の回路のブレッドボード上での実装

2. D2 ピンを外部割込みポートに指定し, 前回の割り込みが発生した時刻 T_p[msec] と今回の割り込みが発生した時刻 T[msec] との差から, モータの回転速度 ϕ[rpm] を計算するプログラムを作成せよ. ϕ は次式で与えられる:

$$\phi[\text{rotation/min}] = \frac{1000[\text{msec/sec}] \times 60[\text{sec/min}]}{(T - T_p)[\text{msec/rotation}]} \tag{5.17}$$

また, 求めた回転速度を 40[msec] ごとにシリアル通信にて確認せよ. (リスト 5.1参照) リスト 5.1 では通信速度を 115200bps に指定している. シリアルモニタの通信速度もそれに合わせること.

3. 「3.1 可視化実験 (1):Processing と Arduino との連携」で行った通信方式 1 に倣って, リスト 5.1 を修正して, rpm 値と duty 比をバイナリデータとして送信するプログラムを作成せよ. また, それらの値を Processing スケッチ `rpmmon.pde` を用いて監視せよ. `rpmmon.pde` 中の `setup()` 内で `port` を設定しているところで`"COM6"` は各自の環境に合わせて適宜設定し直すこと. また, Arduino からのデータ送信はつぎの条件を満たすこと.
   - 通信速度は 115200bps とする.
   - rpm 値と 100 倍した duty 比を `unsigned int` にキャストしてから送信すること.

```
unsigned int x = (unsigned int)motrpm;
unsigned int y = (unsigned int)(100.0*duty);
```

   注. float 型変数 `duty` には現在の duty 比 $\in [0, 1]$ が代入されているものとする.

- 2 つ組データの先頭を表す識別子を 'H' とし, 'H'xy を 40[msec] ごとに送信する.

```
Serial.write('H');
Serial.write(x>>8);
Serial.write(x&255);
Serial.write(y>>8);
Serial.write(y&255);
```

4. 観測したデータを保存 (SAVE DATA をクリック) し, 出力された csv ファイルを scilab を用いてグラフ化せよ. 端末から "scilab &" と入力すると scilab が立ち上がる. Scilab コンソールからつぎのようにして図をプロット&保存できる. (データファイルの名前 "data17....csv" は適宜変更せよ.)

```
-->X = csvRead('data170407134403.csv');          // データを読み込む.
-->T = X(:,1)*0.04;                              // サンプリング周期 0.04[sec]
-->Y = X(:,2);                                   // モータ回転速度 [rpm]
-->U = X(:,3);                                   // duty 比 [%]
-->figure(1);                                    // 図 1(ID 番号 1) を作成
-->plot(T,Y,'r.-');                              // 図 1 に赤いドットと直線で
-->xs2eps(1,"fig1.eps");                         // 図 1 を eps ファイル"fig1.eps"
                                                 // として保存
-->figure(2);                                    // 図 2 を作成
-->plot(T,U,'b-');                               // 図 2 に青い直線で
-->a=get("current_axes");
-->a.data_bounds = [0,0;40,50];                  // axis を設定
-->xs2eps(2,"fig2.eps");                         // 図 2 を eps ファイルとして保存
```

Scilab 命令をスクリプトファイルとして保存し, スクリプトとして実行することもできる. (Scilab コンソールから "help exec" として調べよ.)

Scilab の使い方などは, コンソールから "help **コマンド名**" で調べるか, Scilab オンラインヘルプで調べることができる. また, Scilab の使い方をまとめた web ページがたくさんあるので, 各自で調べること.

5.3.3　フィルタ処理

ロータリエンコーダの量子化誤差などを軽減するため, 第 5.2.1節で説明したローパスフィルタにより観測値を平滑化する.

⟨課題 5.1⟩ ローパスフィルタによる平滑化と伝達関数の推定

1. rpm 値を 1 次の LPF $\dfrac{1}{4-3z^{-1}}$ に通し, 平滑化するように, 演習 5.2-3 のプログラムの ISR(`countUp()`) を修正せよ. κ 回目の割り込みが発生した時刻を T_κ とすると, 平滑化後の rpm 値 y_κ は次式となる.

$$y_\kappa = \frac{1}{4}\left\{3y_{\kappa-1} + \frac{1000\times 60}{T_\kappa - T_{\kappa-1}}\right\} \tag{5.18}$$

   注. y_κ, T_κ は配列を宣言する必要はない. y_κ に関しては, `float` 型変数 (例えば, `motrpmf`) として宣言し, 毎回上書きしていけばよい. また, T_κ は一つ前の値だけを保持しておけばよい.
2. Processing スケッチ `rpmmon.pde` を用いて duty 比, モータ回転速度, LPF による平滑化後の値の 3 つを同時に観測せよ. (`rpmmon.pde` は 4 チャンネルまで同時に計測することができ, チャンネル数は自動で認識する.) また, グラフを Scilab を使って表示・保存せよ. (Scilab ですでにグラフィックウィンドウを開いている場合は, `close` コマンドで一旦閉じておいてから表示すること.)
   平滑化処理の有無でどのような違いがみられるか?
3. 下の「ゲインと時定数の読み取り」を参考に, 表示したグラフからゲインと時定数を読み取り, モータの伝達関数を推定せよ. 拡大したグラフも作成 (axis を適宜設定) し, 図 5.18 を参考に読み取った値をマジェンタの線で示せ.
4. 下の「Scilab によるシミュレーション」を参考に, 求めた伝達関数の応答を計算し, 課題 5.1-3 で描画したグラフに重ねて描け.

ゲインと時定数の読み取り

図 5.7, 5.8 からゲインと時定数を読み取るにはつぎのようにする. 図 5.8 の最初のステップを拡大したものを図 5.18に示す. 入力である Duty 比のステップ信号の高さ (high と low の差) は, $0.25-0.2=0.05$ である. これに対して出力は各ステップごとに高さがまちまちではあるが, 2.8 秒から 10 秒の間の応答を見ると, おおよそ, $3190-1940=1250$ である. よってゲインはその比 $K=1250/0.05=25000$ となる. 時定数 T は, 目標値の $1-e^{-1}\sim 0.632$ 倍に到達するまでの時間なので, low から high に変わるときに $1940+1250\times 0.632=2730$ に達するまでの時間を測ればよい. 最初のステップより, $3.36-2.84=0.52\sim 0.5$, すなわち, 約 0.5 秒である. よって, 伝達関数は,

$$G(s) = \frac{K}{1+Ts} = \frac{25000}{1+0.5s} \tag{5.19}$$

となる. 図 5.8中の $\hat{y}(t)$(–·–) は, 図 5.7の入力 $u(t)$ を (5.19) 式の LPF $G(s)$ に通したときの出力である:

$$\hat{y}(t) = G(s)u(t). \tag{5.20}$$

出力のフィルタ値 $y(t)$(—) とほぼ同じ値を取っており, 大まかな動きをとらえているといえよう.

Scilab によるシミュレーション

伝達関数 $G(s)$ の時間応答を Scilab でシミュレーションするにはつぎのようにする. リスト 5.5のファイルを (使い慣れた) テキストエディタで作成し, 例えば, `kadai5-1-3.sci` という名前で保存する. Scilab コンソール上から `cd` や `pwd` コマンドを使って Scilab のカレントディレクトリをファイルを保存したディレクトリに変更する. カレントディレクトリを変更後, Scilab のコンソール上から

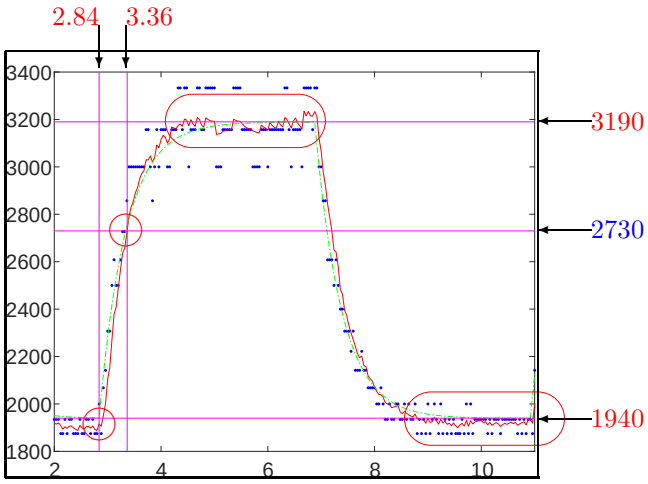

図 5.18　モータの回転速度の拡大図,
横軸:t[sec], 縦軸:$\phi(t)$ [rpm](···), $y(t)$ [rpm](—), $\hat{y}(t)$[rpm](-·-)

リスト 5.5　Scilab によるシミュレーション

```
X = csvRead('data170407134403.csv');     // データを読み込む.
T = X(:,1)*0.04;                         // サンプリング周期 0.04[sec]
U = X(:,2);                              // duty 比 [%]
Y = X(:,3);                              // モータ回転速度 [rpm]
Yf = X(:,4);                             // 平滑化されたモータ回転速度 [rpm]
N = size(U,1);                           // データ数
K=25000;                                 // ゲイン
Ts=0.5;                                  // 時定数
uave = U'*ones(N,1)/N;                   // U の平均
yave = Y'*ones(N,1)/N;                   // Y の平均
U1 = U-ones(N,1)*uave;                   // オフセット処理
s=poly(0,'s');                           // s を Laplace 変換の s として使うおまじない
Gs = syslin('c',K,1+Ts*s);               // 伝達関数 Gs = K/(1+Ts s) の定義
Yhat1 = csim(U1'/100,T',Gs);             // 線形システムのシミュレーション(時間応答)
Yhat=Yhat1'+ones(N,1)*yave;              // オフセット処理
plot(T,Y,'b.');
plot(T,Yf,'r-');
plot(T,Yhat,'g:');                       // 描画
```

```
--> exec kadai5-1-3.sci
```

と実行すればよい. 一般に, 線形モデルが有効なのは動作点近傍だけなので, 入力の平均値を引いて原点近傍に平行移動し, シミュレーション結果には出力の平均値をたしている点に注意せよ.

Scilab には SciNotes という組込み Scilab テキストエディタも存在する. 組込みエディタを使う場合は, `help scinotes` で使い方を調べよ.

⟨課題 5.2⟩ ローパスフィルタの帯域幅

1. 式 (5.7) の α が 2, 4, 8 の場合の応答を同時に観測し, Scilab を用いてグラフに表示せよ.
2. $\alpha = 2, 4, 8$ の場合について, 立ち上がりの速度, 定常状態における雑音の影響などを比較せよ.

⟨発展課題 5.1⟩ 2 次のバタワースフィルタ

1. 式 (5.16) に示す 2 次のバタワースフィルタをプログラム上で実装せよ.
2. 1 次の LPF($\alpha = 4$) と 2 次のバタワースフィルタの出力を比較せよ.

5.4　考察とレポート

報告は, グラフやプログラムを添付するだけでなく, 結果や考察を必ず日本語または英語で説明すること.

[レポート 5.1] LPF の効果と伝達関数

1. 課題 5.1の目的は何か? 100 字以上で答えよ.
2. 課題 5.1の結果を報告せよ. その際, モータの回転速度, duty 比などのデータを, `Scilab` を用いてグラフにすること.
3. 平滑化処理の有無でどのような違いがみられたか? 報告せよ.
4. 制御入力 (duty 比) とモータの回転速度の関係から, モータの時定数と定常ゲインを読み取り, 対象の伝達関数を求めよ. また, 読み取りに使用した拡大グラフを貼り付けよ.
5. 求めた伝達関数のステップ応答 $\hat{y}(t)$ もグラフに表示せよ.
6. 求めた伝達関数をもとに, 次回の課題 6.1 で使うフィードバックゲイン K_P, K_I を求めよ. (求め方は, 第 6.3 節を参照のこと.)

[レポート 5.2] LPF の時定数の影響

1. 課題 5.2の目的は何か? 50 字以上で答えよ.
2. 課題 5.2の結果を報告せよ. その際, モータの回転速度, duty 比などのデータを, `Scilab` を用いてグラフにすること.
3. $\alpha = 2, 4, 8$ の場合で, 立ち上がりの際の遅れや, 定常状態における雑音の影響などを比較せよ. 必要ならば拡大したグラフを作成せよ.

[発展課題レポート 5.1] 2 次のバタワースフィルタ

1. 発展課題 5.1の目的は何か? 50 字以上で答えよ.
2. 発展課題 5.1で, 2 次のバタワースフィルタをプログラミング上でどのように実装したか報告せよ.
3. 1 次のフィルタ ($\alpha = 4$) と 2 次のバタワースフィルタの出力を `Scilab` を用いてグラフにせよ.
4. 1 次のフィルタと 2 次のバタワースフィルタを比較・考察せよ.

表 5.1　使用部品

部品名	規　格	個数	備考
Arduino Uno		1	
USB ケーブル	A オス-B オス 1.5m A-B	1	
ブレッドボード	EIC-801	1	
ブレッドボード・ジャンパーワイヤ	EIC-J-L	1	
モータ・ホールセンサセット	FA-130RA[1] DRV5023A[3] 回転板 ネオジム磁石 ($\phi 3 \times 4$mm)	1	新規貸出 次々回回収
FET	2SK2232[4]	1	新規貸出 次々回回収
白色超高輝度 LED		1	新規貸出 次々回回収 次回使用
コンデンサ	積層セラミック, 0.1μF	1	新規貸出 次々回回収
小型クリップ付コード		2	新規貸出 次々回回収
電池ボックス	2 個用	1	新規貸出 次々回回収
電池	NiH 充電池	2	新規貸出 本日回収
抵抗	10kΩ	1	

参考文献

[1] MABUCHI MOTOR. *FA-130RA Metal-brush motors*, 3 2005.

[2] Arduino Team. Arduino reference. `https://www.arduino.cc/en/Reference/HomePage`.

[3] Texas Instruments. *DRV5023 Digital-Switch Hall Effect Sensor*, 5 2016.

[4] 東芝. **東芝電界効果トランジスタシリコン *N* チャネル *MOS* 形** *2SK2232*, 11 2006.

[5] 荒木光彦. **ディジタル制御理論入門**. 朝倉書店, 1991.

[6] 足立修一. **ディジタル信号とシステム**. 東京電機大学出版局, 2002.

A z 変換とパルス伝達関数

離散時間システムを扱う場合, z 変換とパルス伝達関数を使うと便利である. これらは, 連続時間システムにおけるラプラス変換と伝達関数に相当する. 本付録では, これらを簡単にまとめる. また, サンプラーとホールド回路についても簡単に紹介する. 詳しくは, [5, 6] などを参考にすること.

A.1 z 変換

離散時間信号 $\{x[k]\}_{k=-\infty}^{\infty}$ を考える. ここで, k は離散的な時刻を表す. 例えば, 連続時間信号 $x(t)$ をサンプル時刻 t_k でサンプリングする場合, $x[k] = x(t_k)$ である. また, サンプリング周期 T_s で等間隔にサンプリングする場合は, $t_k = kT_s$ である.

離散時間信号 $\{x[k]\}$ の z 変換は

$$X(z) = \sum_{k=-\infty}^{\infty} x[k] z^{-k} \tag{5.21}$$

で与えられ,

$$X(z) = \mathcal{Z}\{x[k]\} \tag{5.22}$$

と書く. $x[k]$ は $X(z)$ をローラン展開したときの z^{-k} の係数とみなすことができるので, 留数の定理より, z 変換の逆変換 (逆 z 変換) は,

$$x[n] = \mathcal{Z}^{-1}\{X(z)\} = \frac{1}{2\pi j} \oint X(z) z^{n-1} dz \tag{5.23}$$

で与えられる. ここで, $z = 0$ のまわりの一周積分の $1/(2\pi j)$ 倍は $1/z$ の係数となるので, (5.21) 式を (5.23) に代入すると, 上の積分は, $k = n$ の係数 $x[n]$ となるのが理解されよう. 時系列データを扱う場合, ラプラス変換と同様, 時刻が非負の範囲を考えることが多いので, 片側 z 変換:

$$X(z) = \sum_{k=0}^{\infty} x[k] z^{-k} \tag{5.24}$$

を使うことも多い.

ラプラス変換と同様, z 変換にもいくつかの重要な性質がある. 線形性:

$$\mathcal{Z}\{ax[n] + by[n]\} = a\mathcal{Z}\{x[n]\} + b\mathcal{Z}\{y[n]\} \tag{5.25}$$

は自明であろう. $zX(z)$ の逆 z 変換は

$$\mathcal{Z}^{-1}\{zX(z)\} = \frac{1}{2\pi j} \oint X(z) z^{(n+1)-1} dz = x[n+1], \tag{5.26}$$

$z^{-1}X(z)$ の逆 z 変換は

$$\mathcal{Z}^{-1}\{z^{-1}X(z)\} = \frac{1}{2\pi j} \oint X(z) z^{(n-1)-1} dz = x[n-1] \tag{5.27}$$

なので, z は進み作用素, z^{-1} は遅れ作用素とみなすことができる. 以上より, 時間軸の推移律:

$$\mathcal{Z}\{x[n-k]\} = z^{-k}\mathcal{Z}\{x[n]\} \tag{5.28}$$

を得る. 離散時間信号の畳込み

$$z[n] = (x * y)[n] = \sum_{k=-\infty}^{\infty} x[k]\, y[n-k] \tag{5.29}$$

の両辺を z 変換すると,

$$\mathcal{Z}\{z[n]\} = \sum_{n=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x[k]\,y[n-k]z^{-n} \tag{5.30}$$

$$= \sum_{n=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x[k]z^{-k}\,y[n-k]z^{-(n-k)} \tag{5.31}$$

$$= \sum_{n=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x[k]z^{-k}\sum_{n=-\infty}^{\infty} y[n-k]z^{-(n-k)} = \mathcal{Z}\{x[n]\}\mathcal{Z}\{y[n]\} \tag{5.32}$$

よって, 畳込みの公式:

$$\mathcal{Z}\{(x*y)[n]\} = \mathcal{Z}\{x[n]\}\,\mathcal{Z}\{y[n]\} \tag{5.33}$$

を得る. この他, 周波数軸の推移律, 時間軸の反転, z 領域における微分, 初期値定理, 最終値定理, などがあるがここでは割愛する. また, z 変換の収束領域に関する議論は省略した.

cf. インパルス列のラプラス変換

連続時間信号 $x(t)$ を時刻 kT_s でサンプリングしたものを $x[k] = x(kT_s)$ とおき,

$$x^*(t) = x(t)\sum_{k=-\infty}^{\infty}\delta(t-kT_s) = \sum_{k=-\infty}^{\infty} x[k]\delta(t-kT_s) \tag{5.34}$$

とする. $x^*(t)$ は, 時刻 kT_s $(k = \cdots -1, 0, 1, \cdots)$ 以外で 0, kT_s では, δ 関数の $x[k]$ 倍となるインパルス列である. これを (両側) ラプラス変換すると,

$$X^*(s) = \int_{-\infty}^{\infty} x^*(t)e^{-st}\,dt = \sum_{k=-\infty}^{\infty} x[k]e^{-skT_s}. \tag{5.35}$$

このとき, $X^*(s)$ を $z = e^{sT_s}$ の関数だと思って $X^*(z)$ と書くと, $X^*(z)$ は $x[k]$ の z 変換になっている. すなわち, $x[k]$ の z 変換は, インパルス列 $x^*(t)$ のラプラス変換に他ならない. この結果は, 実際のディジタル信号がインパルス列であると言っているのではなく, そのように考えると, ラプラス変換との整合性があると言っているのである. $x[k]$ と $x^*(t)$ は同じ離散時間信号の別の表現である. $x[k]$ を数列表現, $x^*(t)$ をインパルス列表現という.

A.2　パルス伝達関数

システムの入力と出力をそれぞれ $u[k]$, $y[k]$, それらの z 変換を $U(z) = \mathcal{Z}\{u[k]\}$, $Y(z) = \mathcal{Z}\{y[k]\}$ とする. システムの入出力の z 変換の比:

$$G(z) = \frac{Y(z)}{U(z)} \tag{5.36}$$

をパルス伝達関数という.

つぎの差分方程式で表されるシステム:

$$y[k] + a_1y[k-1] + \cdots + a_{k_a}y[k-n_a] = b_0u[k] + b_1u[k-1] + \cdots + b_{k_b}u[k-n_b] \tag{5.37}$$

の場合, 両辺を z 変換すると,

$$A(z)Y(z) = B(z)U(z) \tag{5.38}$$

ここで,

$$A(z) = 1 + a_1z^{-1} + \cdots + a_{n_a}z^{-n_a}, \tag{5.39}$$

$$B(z) = b_0 + b_1z^{-1} + \cdots + b_{n_b}z^{-n_b}. \tag{5.40}$$

よって, パルス伝達関数は,

$$G(z) = \frac{Y(z)}{U(z)} = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}} \tag{5.41}$$

となる. $G(z)$ を $1/z = 0$ $(z = \infty)$ のまわりでローラン展開したものを

$$G(z) = \sum_{k=-\infty}^{\infty} g[k] z^{-k} \tag{5.42}$$

とおく. $g[k]$ はインパルス応答と呼ばれる. 畳込みの公式より,

$$y[n] = \sum_{k=-\infty}^{\infty} g[n-k]\, u[k] = \sum_{k=-\infty}^{\infty} g[k]\, u[n-k] \tag{5.43}$$

となる. システム (5.37) は因果的な (出力は未来の入力に依存しない) システムである. この場合, $k < 0$ に対して $g[k] = 0$ となる. すなわち,

$$G(z) = \sum_{k=0}^{\infty} g[k] z^{-k}, \tag{5.44}$$

であり, 入出力の畳込み表現は,

$$y[n] = \sum_{k=0}^{\infty} g[k]\, u[n-k] \tag{5.45}$$

となる.

cf. パルス伝達関数 (5.41) のインパルス応答は $k < 0$ に対して $g[k] = 0$ となることを各自確認せよ.

A.3 サンプラーとホールド回路

実際の物理量をセンサなどで観測したり, アクチュエータなどを介して操作する場合, 離散時間信号と連続時間信号との間の変換が必要になる. サンプラーとは, AD 変換器をモデル化したもので, その動作は (5.34) 式で与えられる. ホールド回路とは, 離散時間信号を連続時間信号に変換する回路のことで, DA 変換器がその代表例である. $u[k]$ の値をサンプリング周期の間保持して

$$u(t) = u[k] \quad \text{for} \quad kT_s \le t < (k+1)T_s \tag{5.46}$$

とする 0 次ホールド (ZOH: Zero-Order Hold) や, 傾き $(u[k] - u[k-1])/T_s$ まで考慮する 1 次ホールド, さらに高次のホールド回路も考えられるが, 一般には 0 次ホールドが用いられる. 各信号の様子を図 5.19に示す.

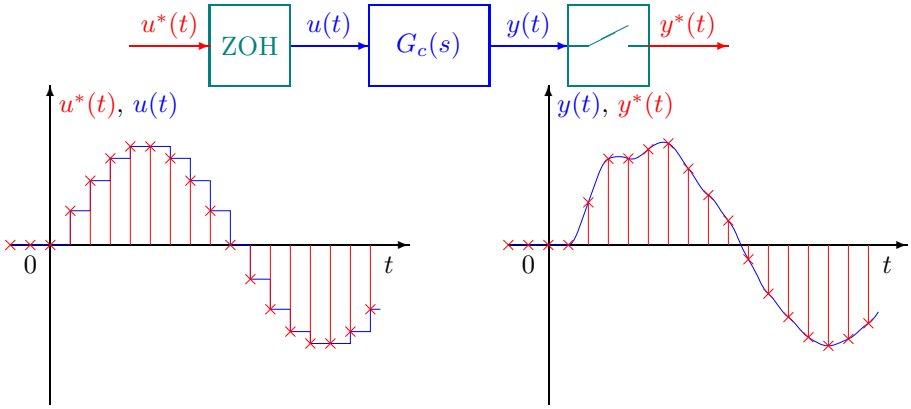

図 5.19 離散時間信号と連続時間信号の変換

インパルス列表現 $u^*(t)$ から (5.46) 式の $u(t)$ を出力する ZOH の伝達関数は,

$$H_c(s) = \frac{1 - e^{-sT_s}}{s} = T_s \left(1 - \frac{1}{2}T_s s + \frac{1}{3!}T_s^2 s^2 - \cdots\right) \tag{5.47}$$

となる. $u^*(t)$ を積分器 $1/s$ に通すと各サンプル時刻 kT_s でステップの高さの差が $u[k]$ となる階段状の関数となる. 一方, e^{-sT_s} は時間 T_s だけ信号を遅らせるむだ時間なので, $H(s)$ の出力は, 現在の積分値から T_s だけ前の積分値を差し引いた, 高さが $u[k]$ の階段状の関数 (5.46) となる.

サンプラーの前後の信号のラプラス変換はつぎのとおりである. 単位インパルス列を $\delta^*(t)$ と書く. $\delta^*(t)$ は周期 T_s の周期関数なので, フーリエ級数展開すると,

$$\delta^*(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s) = \frac{1}{T_s} \sum_{\mu=-\infty}^{\infty} e^{-j\mu\omega_s t}. \tag{5.48}$$

ここで, $\omega_s = 2\pi/T_s$ はサンプリング角周波数である. これより, $y^*(t) = y(t)\delta^*(t)$ をラプラス変換すると, $y(+0) = 0$ のとき,

$$Y^*(s) = \int_{-\infty}^{\infty} \frac{1}{T_s} \sum_{\mu=-\infty}^{\infty} y(t) e^{-(s+j\mu\omega_s)t}\, dt = \frac{1}{T_s} \sum_{\mu=-\infty}^{\infty} Y(s + j\mu\omega_s) \tag{5.49}$$

となる. サンプリングされたインパルス列信号のスペクトル $Y^*(j\omega)$ は連続時間信号のスペクトル $Y(j\omega)$ を ω_s ずつ平行移動して加え合わせたものとなっている. したがって, $Y(j\omega)$ がナイキスト角周波数 $\omega_s/2$ 以上の周波数成分を持たない場合は, $Y^*(j\omega)$ は $Y(j\omega)$ のスペクトル波形を繰り返したものになる (図 5.20上と中). したがって, サンプル値からもとの連続時間信号の周波数スペクトルを再現することができる. 一方, $Y(j\omega)$ がナイキスト角周波数 $\omega_s/2$ 以上の周波数成分を持つ場合, いわゆるエイリアジングが発生し, インパルス列に含まれる情報だけを使って連続時間信号の周波数スペクトルを再現することはできない (図 5.20上と下). これは, 染谷-Shannon のサンプリング定理と呼ばれる重要な結果である.

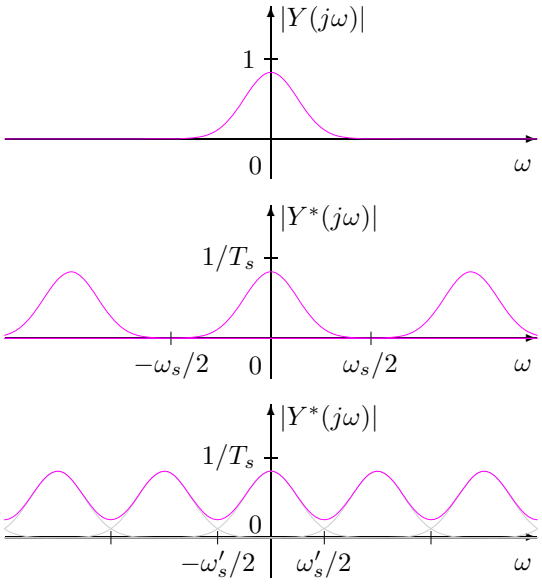

図 5.20　連続時間信号 $y(t)$ とそのインパルス列 $y^*(t)$ のスペクトル

連続時間システムの伝達関数を $G_c(s)$ とすると, インパルス列 $u^*(t)$ から $y(t)$ までの伝達関数は ZOH の伝達関数との積 $H_c(s)G_c(s)$ となる. したがって, パルス伝達関数 $G(z)$ のインパルス応答 $g[h]$ は $H_c(s)G_c(s)/T_s$

のインパルス応答をサンプリングしたものと等しく, $G_c(s)$ のインパルス応答をサンプリングしたものではない点に注意が必要である.