

## PID 制御

創造設計第二 TA：寺内直樹, 中村雄大

平成 22 年 11 月 4 日

### 1. はじめに

今回の試作検討ではロータリエンコーダをセンサとしたモーターの PID 制御を学ぶ。本 Tutorial で行う PID 制御の概要を Fig. 1 に示す。モーターおよびロータリエンコーダのキットはこちらで配布するものを用いる。また、これらのモーターの駆動およびロータリエンコーダの利用には Tutorial3 で製作したドライバ基板を使用する。

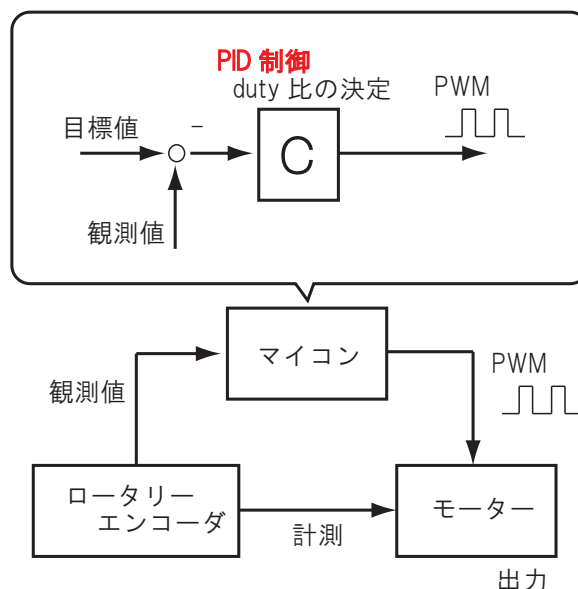


Fig. 1: Tutorial4 で扱う PID 制御の概要

### 2. ロータリエンコーダ

本 Tutorial ではモーターの回転角を測定するのにロータリエンコーダを用いる。ロータリエンコーダは軸が一定角回転するとパルスを出力するため、このパルス数をマイコンのタイマでカウントすることでモーターの回転角を求めることができる。

今回用いるロータリエンコーダは Fig. 2 のような 2 相出力のものである。A 相、B 相の位相のずれた 2 つのパルスの立ち上がりおよび状態を計測することで回転方向と回転角を測定することができる。2 相式ロータリエンコーダの詳細はメカトロニクスラボ・マイコン編を参照する。また、エンコーダの詳しい仕様はデータシートで確認できる。

#### 2.1 カウント方式

パルスのカウント方式には主に 1, 2, 4 通倍カウントがあり、本 Tutorial では 4 通倍カウントを用いる。4 通倍カウントでは、A 相の立ち上がりエッジと立ち下がりエッジおよび B 相の立ち上がりエッジと立ち下がりエッジ

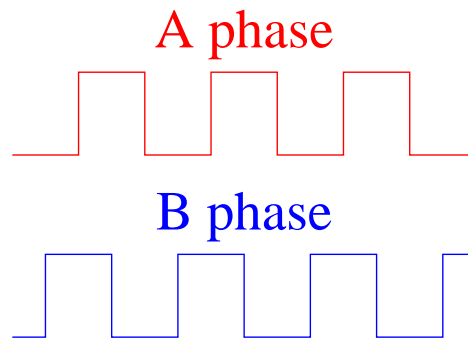


Fig.2: 2相出力信号

Table 1: A相/B相と1,2,4通倍カウントの関係

A相	立ち上がり		立ち下がり		High(1)	Low(0)	High(1)	Low(0)
B相	High(1)	Low(0)	High(1)	Low(0)	立ち上がり		立ち下がり	
回転方向	逆 (-)	正 (+)	正 (+)	逆 (-)	正 (+)	逆 (-)	逆 (-)	正 (+)
1通倍	カウント		×なし		×なし		×なし	
2通倍	カウント		カウント		×なし		×なし	
4通倍	カウント		カウント		カウント		カウント	

をカウントして回転角度を測定する。さらに各エッジを検知したときのもう一方の相の電位レベルを計測することで回転方向を判別する。本 Tutorial で用いる 202A100A というエンコーダは 1 回転で 100 パルス出力するので、その分解能は 1 通倍カウントの分解能  $360/100 = 3.6[\text{deg}]$  の 4 倍の  $0.9[\text{deg}]$  となる。Table 1 に 1, 2, および 4 通倍カウントとエッジの関係をまとめる。

## 2.2 マイコンとの接続

4 通倍カウントを実装するためには、外部割り込みにより A 相 B 相それぞれに対し立ち上がり、立ち下がりエッジの検出が必須である。しかし、本授業で使用しているマイコンには立ち上がりおよび立ち下がりエッジ両方に対し割り込みを発生させるポートは存在しない。したがって A 相の立ち上がりエッジと立ち下がりエッジおよび B 相の立ち上がりエッジと立ち下がりエッジの検出を 4 つのポートで行う。本 Tutorial では Table 2 に示すポートでそれぞれのエッジを検出する。

Table 2: 4 通倍カウントで用いるポート

A 相立ち上がり	P50/WKP0
A 相立ち下がり	P51/WKP1
B 相立ち上がり	P52/WKP2
B 相立ち下がり	P53/WKP3

Tutorial3 で製作したドライバ基板にはエンコーダと接続するコネクタが付属している。ロータリエンコーダのピン配置を Fig.3 に示す。また、マイコン側のピン配置を Fig.4 に示す。A 相、B 相の信号がそれぞれ Table 2 に対応するポートへと繋がっていることを確認すること。

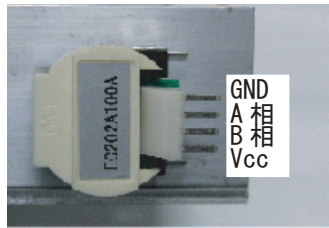


Fig.3: ロータリーエンコーダのピン配置

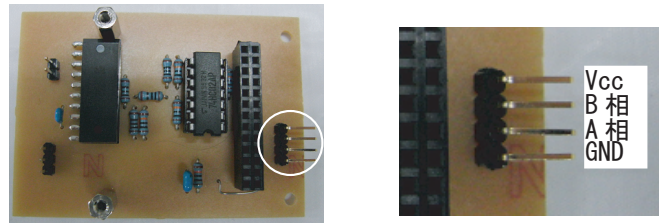


Fig.4: マイコンのコネクタの様子

マイコンにモーターおよびエンコーダを接続するとマイコンのコネクタの様子は Fig.5 のようになり, 全体としては Fig.6 のようになる.

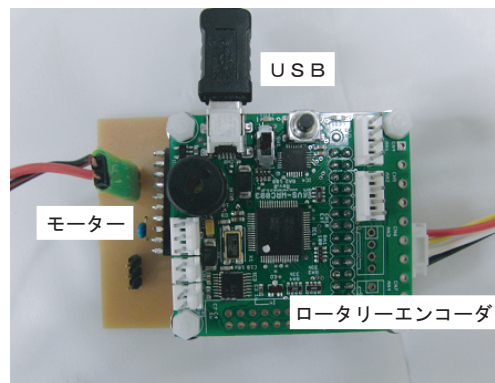


Fig.5: マイコンのコネクタの様子

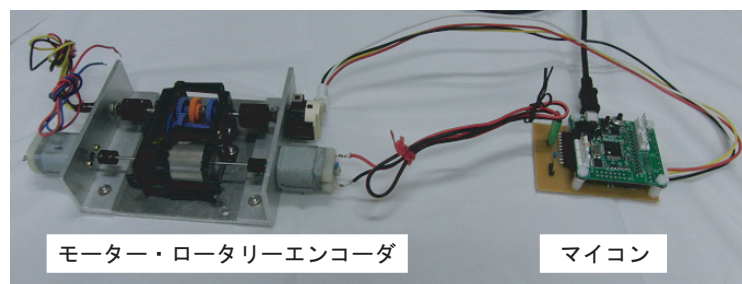


Fig.6: マイコンとモーター・ロータリーエンコーダを接続した様子

なお, Fig.6 ではメス-メスコネクタを用いてマイコンとモーターを繋いでいるが, 今回はこのコネクタを配布していないので IC クリップを用いて接続する.

## 2.3 割り込みによるパルスのカウント

前節のようにポートを割り当てた場合、エッジが検出できるよう各ポートを次のように設定する。

```
/*エンコーダ用の外部割り込みの設定*/
/*P50 で立ち上がりエッジを検出*/
IEGR2.BIT.WPEG0 = 1; //WKP0:立ち上がりエッジ検出 (立ち下がりエッジ検出であれば 0)
IO.PMR5.BIT.WKP0 = 1; // P50 を入力端子に設定
IWPR.BIT.IWPF0 = 0; //WKP0 の割り込み要求フラグをクリア
IENR1.BIT.IENWP = 1; //WKP0~WKP5 の割り込み許可
```

上の例では P50 の設定を行っているが、P51 ~ P53 でも同じように設定を行う。また、割り込みによる処理を INT.WKP(void) に記述する必要がある。

ロータリーエンコーダによるエッジ検知に関しては、encoder.c の Four\_Count() に次のように記述されている。

```
/*gEncCnt : エッジ検出のカウント*/

/*A 相の立ち上がり処理*/
//立ち上がりエッジを検出すれば"1"
if( IWPR.BIT.IWPF0 == 1 ){
    IWPR.BIT.IWPF0 = 0; //割り込みフラグクリア
    //B 相の状態が"Low"なら正転
    if( IO.PDR5.BIT.B2 == 0 ){
        /*正転時の処理*/
        gEncCnt++;
    }else{
        /* 逆転時の処理*/
        gEncCnt--;
    }//if( IO.PDR5.BIT.B2)
}//if( IWPR.BIT.IWPF0)

/*以下 A 相の立ち下がり、B 相の立ち上がりおよび B 相の立ち下がりに関して記述する。*/
```

このようにして、gEncCnt の増減を見れば正転、逆転の判断ができる。さらに前述した通り、本 Tutorial で用意したエンコーダでは 1 回転で 100 パルス出力し、A 相・B 相の立ち上がり・立ち下がりエッジを検出する。そのため、1 周正転させた場合には gEncCnt は 400 増加することになる。これを用いれば、回転角を求めたり、回転の速度を求めたりすることができる。例えば 1sec で gEncCnt の値が 100 増加していれば、その 1sec での平均速度は  $2\pi \cdot (100/400) = \pi/2$  [rad/s] と算出できる。

### 3. PID 制御

PID 制御とは制御手法の 1 つであり、産業界では主な制御手法として広く用いられている。「PID」は

- 比例：Proportional
- 積分：Integral
- 微分：Derivative

それぞれの頭文字をとったものであり、その名の通りこれら 3 つの動作を組み合わせた制御手法である。

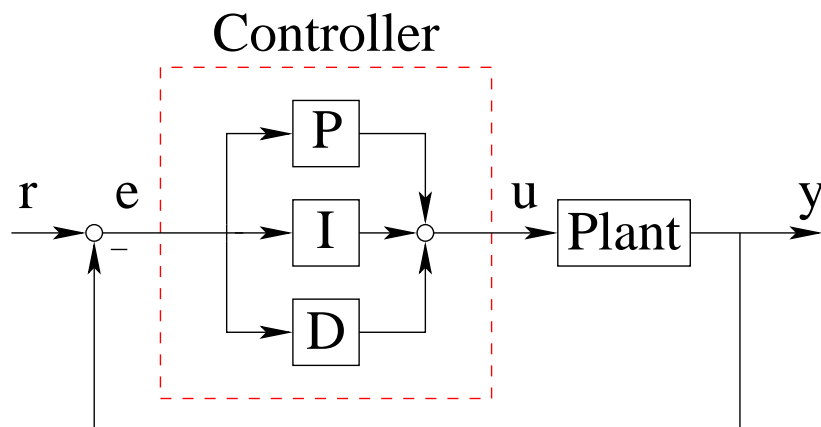


Fig. 7: PID 制御のブロック線図

具体的には Fig. 7 のように目標値  $r$  と出力  $y$  との偏差  $e$ 、その積分、そしてその微分に係数を掛けて足したものを制御入力とする。式で表すと以下のようになる。

$$u = K_P e + K_I \int e dt + K_D \dot{e} \quad (1)$$

それぞれの係数  $K_P$ ,  $K_I$ ,  $K_D$  は比例、積分、微分動作に対するゲインであり、これらの値を調節することで所望の出力を実現する。例えば、位置の目標値  $x_d$  と観測された位置  $x$  を用いて  $e = x_d - x$  として適切な入力を求めれば位置  $x$  を制御することができ、速度の目標値  $v_d$  と観測された速度  $v$  を用いて  $e = v_d - v$  として適切な入力を求めれば速度  $v$  を制御することができる。

各節では、比例、積分、微分の項がそれぞれどのように出力に対し影響を与えるのかを復習し、次章で課題を通じマイコンを用いて実装する。

#### 3.1 P 制御

P 制御は最も単純な制御方法である。入力  $u$  は

$$u = K_P e \quad (2)$$

のように表される。すなわち、「出力が目標からずれていれば入力を大きくして、目標に近づいたら入力を小さくする」。しかし、これだけでは必ずしも所望の動作は得られない可能性がある。例えば、

- 定常偏差が残る。
- 収束スピードを早くしようとすると振動的になる。

という問題がある。

そこで、P 制御を基本として積分動作や微分動作を加えて所望の動作の実現を目指す。

#### 3.2 PI 制御

P 制御に積分動作を加えたものである。入力  $u$  は

$$u = K_P e + K_I \int e dt \quad (3)$$

のように表される．P制御では，今回扱うモーターのように摩擦があるような場合には偏差が小さくなり入力uが小さくなると入力uが摩擦より小さくなってしまい，そのため，モーターが動かなくなり偏差が残ってしまう（定常偏差）．この時，偏差を積分していれば時間の経過と共に徐々に入力uが大きくなり偏差をなくすることができる．このように積分動作には定常偏差を取り除く効果がある．

### 3.3 PD 制御

P制御に微分動作を加えたものである．入力uは

$$u = K_P e + K_D \dot{e} \quad (4)$$

のように表される．P制御に速度の偏差の項が加わっているため，より速く動かそうとする．その結果，応答性が高まり外乱に強くなる．ただし，微分項のゲイン  $K_D$  を大きくしすぎるとオーバーシュートが発生しやすくなるので注意が必要である．

### 3.4 PID 制御

P制御に積分動作，微分動作を加えたものである．積分動作，微分動作両方の特徴を活かすことができるが，それぞれの項に対する係数のチューニングをしっかりと行うことが必須である．これらパラメータの設計方法としては限界感度法やステップ応答法などが主な方法として広く使われている．

PID制御のイメージ図を Fig. 8 に載せる．

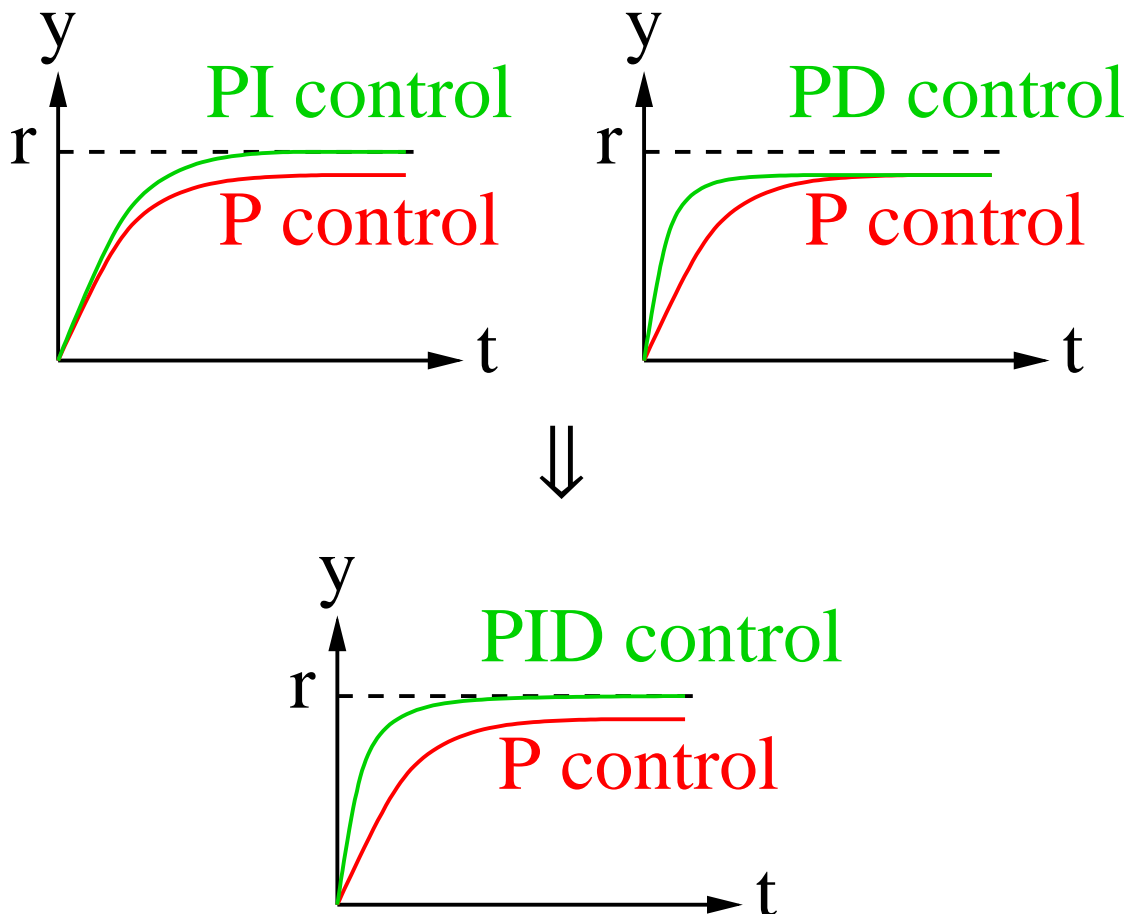


Fig. 8: PID 制御

### 3.5 目標軌道

PID 制御はある目標値に追従させるための制御方法である。十分早い時間で目標値に追従させることができるパラメータが見つかった場合、目標値にダイナミクスを加えた目標軌道に追従させることができる。例えば、速度の目標軌道を  $v = \sin t$  と設定し、PID 制御により速度をこれに追従させれば、滑らかに加速と減速を繰り返す動きが実現できる。また、状態遷移に伴い目標軌道を変化させれば、A 区間では加速、B 区間では減速といった動きが可能である。



注意

目標位置に短時間で到達させようとしたり速度の目標軌道を大きくしたりすると、目標速度軌道がモーターの出せる最大速度を越えてしまい、目標軌道に追従できなくなるので注意が必要である。



注意

「PWM の duty 比が 100 % に達するとブザーが鳴る」等の処理を加え、モーターの出せる最大速度を超えたことがすぐに認識できるようにする

## 4. 課題 4

### 課題 4. 1

配布したモーターとエンコーダの動作確認をし、シリアル通信によって Hterm に文字列を表示させる。

1. 補足資料を参考にして「Hello, World!」を Hterm に表示させる。
2. 3.0[sec] ごとに正転、逆転を繰り返すプログラムを作成し、モーターが駆動することを確認する。
3. 割り込み関数内のコメントを外してエンコーダのパルス出力の検出を有効にし、0.25[sec] ごとに gEncCnt の値を Hterm または Excel のシートに表示する。duty の値は適当に定める（例えば duty = 50000）。時刻、回転方向 (dir)、カウント数 (gEncCnt) をタブ区切りで送信すると良い。

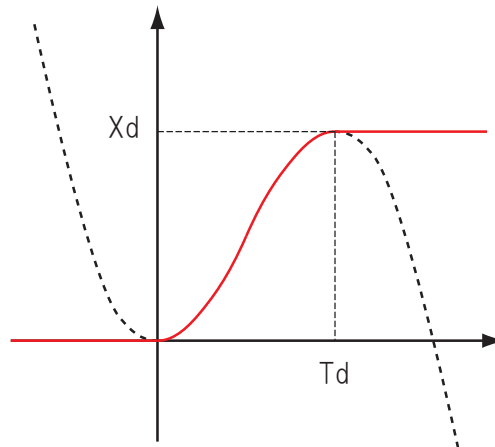
### 課題 4. 2

目標位置に到達するまで滑らかな加減速を行う目標軌道を生成し、出力がその軌道に追従するよう PID 制御を行う。

1. Fig.9 のように三次関数の極小値から極大値までの曲線を用いた目標位置軌道を求める。
  - 目標位置  $x_d$ 、目標到達時間を  $T_d$  と置いて条件を満たす三次関数を求める。
  - 時刻  $t$  における目標値を求める関数 Caluc.target\_x に、求めた三次関数を反映させる。
2. PID 制御に必要な  $e$ ,  $e_i$ ,  $e_d$  を（偏差、偏差の和、偏差の差）を求め、制御入力を算出するプログラムを完成させる。
3. 設計した入力でモーターを制御し、目標軌跡と観測軌跡を比較する。
4. 係数のチューニングを行う。

### 課題 4. 3

1. 今までの Tutorial の中の課題で、終わっていないものがあればそれを終わらす
2. 「センサ入力 状態遷移 状態に対応した出力」の一連の流れを確認する  
実際にレールの上を走らせてみる



**Fig. 9:** 滑らかな加減速を行う目標軌道

#### PID 制御で用いる主な関数

- shisaku04
  - メイン関数が記述されている
    - main
      - メイン関数
    - Buzzer, BuzzerON, BuzzerOFF
      - Tutorial1 で使用したものと同じ. ブザーの設定, ON, OF
    - INT\_WKP
      - エンコーダのパルスのエッジ検出のための割り込み関数
    - INT\_TimerB1
      - 5ms ごとに割り込みされる関数
- motor.c
  - Tutorial3 で使用したものと同じ内容でモータ駆動に関する内容が記述されている.
    - MotorInit
      - ポート関連の初期化
    - Motor
      - モータの駆動用の関数
- encoder.c
  - EncInit
    - エンコーダの利用に伴うポートの設定
  - Four\_Count
    - エンコーダからのパルスのエッジを検出し、カウントする
- PID.c
  - Calcu\_target\_x
    - 各時刻における目標軌道上の値を求める
  - PIDctrl
    - PID 制御によりモーターへの入力である PWM の符号付きパルス幅を求める
  - sci.c, nosprintf.c
    - 補足資料を参照する



ポートの割り当て状況

ポート	機能
P50	A 相立ち上がり検出
P51	A 相立ち下がり検出
P52	B 相立ち上がり検出
P53	B 相立ち下がり検出
P63	外付外部モータ駆動用の PWM を出力
P64	LED1 の点灯
P65	LED2 の点灯
P70	モータの回転方向を決める信号を出力