

制御の基礎

仙台市地域連携フェロー

熊谷正朗

kumagai@tjcc.tohoku-gakuin.ac.jp

東北学院大学工学部
ロボット開発工学研究室 **RDE**

今回の目的

○ 制御の基礎

テーマ1: 制御の目的と基本

- ・ 制御するとは
- ・ 制御の基本（フィードバック、PID）

テーマ2: 少し特殊な制御

- ・ フィードフォワード
- ・ 非線形制御

テーマ3: 制御の実例

- ・ モータの制御、ロボット制御

制御する

○ 対象を思い通りに動かすこと

◇ 思い通り ~ 現在値 = 目標値

- ・ 現在値: 対象の状態(センサなどで検出)
- ・ 目標値: 対象のあるべき状態

※ 一定値/時々刻々変化する値

※ しかるべき目標値の生成も含む

◇ 方法

- ・ 現在値を変化させうる操作を行う。
- ・ {目標、現在} → 制御方法/制御則 → 操作

制御する

○ 対象を思い通りに動かすこと

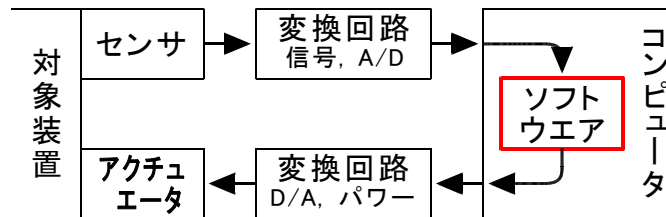
◇ たとえば

- ・ モータの回転角度 / 回転速度
- ・ 電気炉の温度
- ・ 電圧（電源装置など）
- ・ 腕ロボットの手先の位置姿勢
- ・ 道路での車の渋滞
- ・ コンピュータのサーバの負荷バランス

制御する

○ 制御する≡制御ソフトをつくる

- ・コンピュータのソフトウェアで制御を行うことが今は主流。
- ・従来はアナログ制御が主流だった。



制御する

○ 制御しやすいもの

◇操作できる

- ・短絡的に操作が結果に反映される。
 - ・操作を増やしたら結果も増える(減る)。
- ※単調性、線形性、比例すれば一番楽

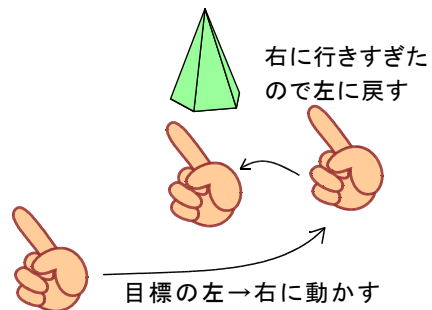
◇計測できる

- ・計測できなければフィードバック制御不可。
- ・制御の性能は計測の性能以下。

制御の基本

○ フィードバック制御

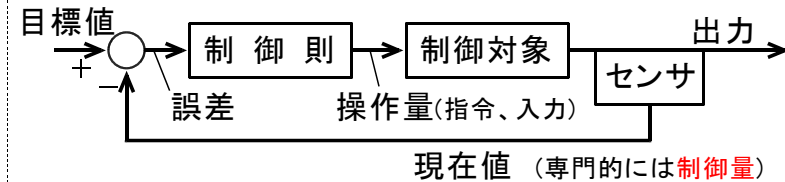
センサで読み取った現在値を、目標に近づけるようにする制御。(○○制御というと、一般にこの一種)



制御の基本

○ フィードバック制御

センサで読み取った現在値を、目標に近づけるようにする制御。(○○制御というと、一般にこの一種)

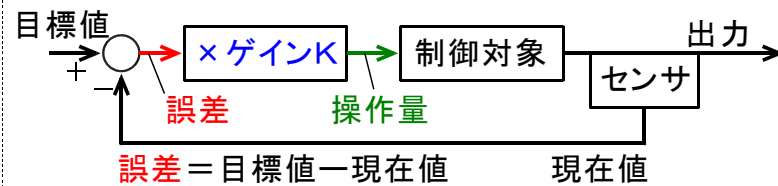


制御の基本

○ 比例制御（P制御）

◇概要

- ・ 誤差に比例した操作を行う
- ・ 例）温度誤差に比例した電力出力
角度誤差に比例したモータ速度

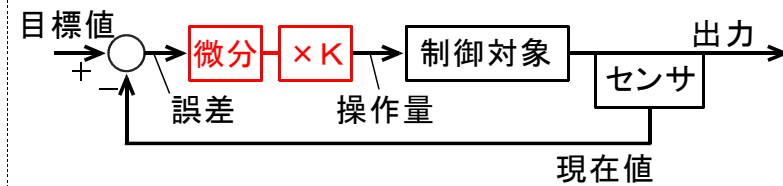


制御の基本

○ 微分制御（D制御）

◇概要

- ・ 誤差の時間変化に比例した操作を行う。
- ・ 誤差が増加しそうなときに歯止め。
- ・ 単独では使わない。（P制御と組み合わせ）

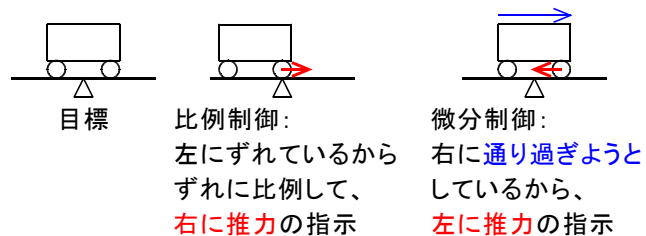


制御の基本

○ 微分制御（D制御）

◇イメージ

- ・ 誤差の時間変化に比例した操作を行う。
- ・ 誤差が増加しそうなときに歯止め。



制御の基本

○ 微分制御（D制御）

◇別の見方

- ・ 誤差の時間変化に比例した操作を行う。

・ 速度を一致させる制御

現在値と目標の速度の差に比例

誤差 = (目標 - 現在)

誤差の時間変化

= (目標の時間変化 - 現在の時間変化)

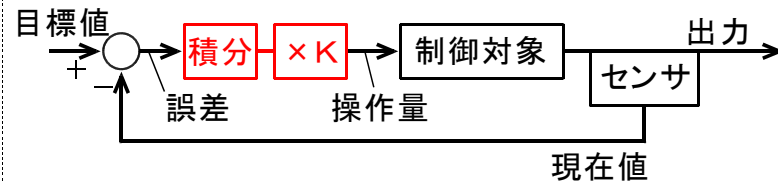
= (目標の速度 - 現在の速度)

制御の基本

○ 積分制御 (I 制御)

◇概要

- ・誤差の積分に比例した操作。
- ・誤差が残っていると徐々に操作を大きく。
- ・例) 上り坂でのアクセル



P制御・D制御・I制御

○ P制御

◇基本はP制御

- ・対象によってはPのみもOK。
- 例) モータの角度をモータの速度で操作。

◇P制御のみではNGな場合:

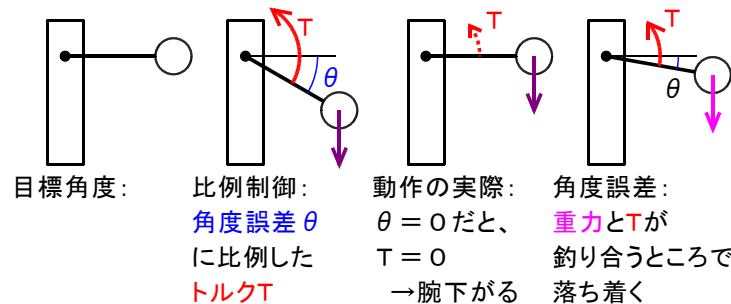
- ・誤差がなくなる。
- ・振動(行き過ぎ)を起こす。

P制御・D制御・I制御

○ PI制御

◇P制御だと誤差が残る事例

- ・腕の角度を力で操作

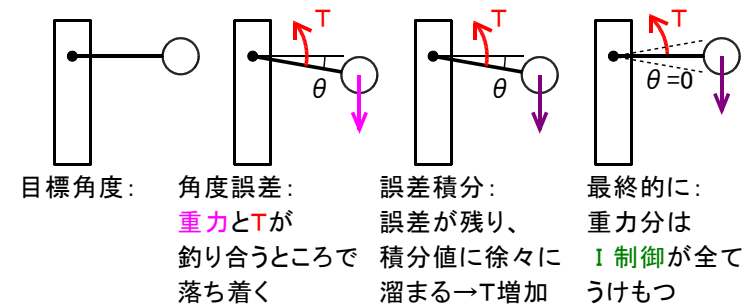


P制御・D制御・I制御

○ PI制御

◇PI制御による解決 (操作 = P制御 + I制御)

- ・腕の角度を力で操作

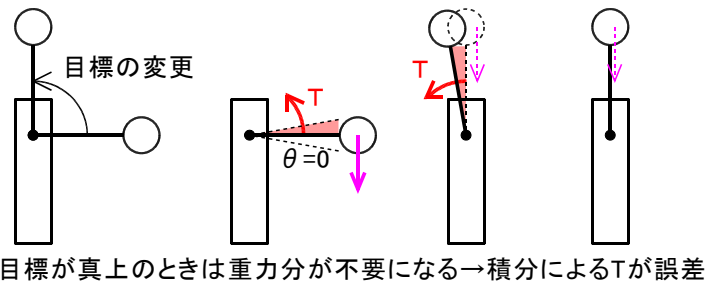


P制御・D制御・I制御

○ PI制御

◇PI制御の弱点＝応答の遅れ

- ・誤差の積分が解消するまでは誤差になる

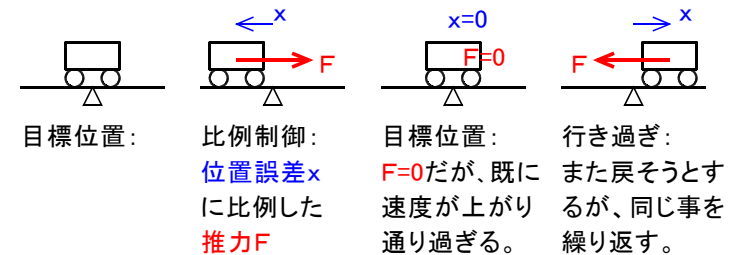


P制御・D制御・I制御

○ PD制御

◇P制御だと行き過ぎ/発振が起こる事例

- ・力操作で物体の位置を制御（実はバネと同じ）



P制御・D制御・I制御

○ PD制御

◇PD制御による改善

- ・機械的にはダンパーを入れることに対応
- ・D制御は「速度を一致させるようにする」
- ・目標が一定値の場合＝速度ゼロ
 - 対象の速度をゼロにする＝ブレーキ的
 - 振動が収まる
- ・目標が急に変化した場合（速度が急にでた）
 - 速度差を埋めるように出力

P制御・D制御・I制御

○ PD制御

◇PD制御の弱点＝センサノイズの影響

- ・高周波数のノイズの「時間変化」は大きい
 - 例) 1秒(1Hz)で1V変化 → 1[V/s]
 - 10 μs(100kHz)で1mV → 100[V/s]
- ＝「対象の変化」ではなく、ノイズにより過敏に反応する危険性
- ・センサ分解能や処理周期の影響も受ける。
 - (後述) →第7回 信号処理の基礎

P制御・D制御・I制御

○ PID制御

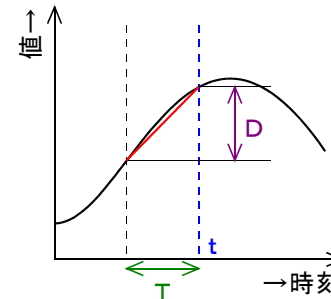
◇三種の組み合わせ

- ・ P制御：主たる誤差の低減
- ・ I制御：誤差が残らないようにする
- ・ D制御：振動の低減
ブレーキや反応性向上
- ・ それぞれに「どの程度の比率(ゲイン)で」
比例させるかは、種々の方法で調整
※ 勘、挙動の観察、限界感度法など

デジタル制御とPID

○ 積分と微分の計算（信号処理の基礎より）

◇微分

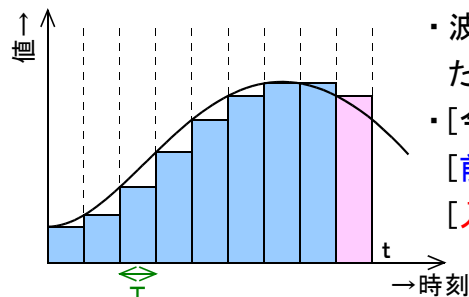


- ・ ある時刻 t で間隔 T の間の変化 D を求める。
- ・ D/T をその瞬間の微分値として使う。

デジタル制御とPID

○ 積分と微分の計算（信号処理の基礎より）

◇積分



- ・ 波形と横軸で囲まれた面積。短冊の和。
- ・ [今回の積分値] =
[前回の積分値] +
[入力値] $\times T$

デジタル制御とPID

○ PID制御のプログラム

◇時間間隔 T ごとに以下の処理を実行

$$\text{誤差 } e = \text{目標値 } r - \text{現在値 } y$$

$$\text{誤差積分 } e_i = e_i + e \times T$$

$$\text{誤差微分 } e_d = (e - \text{前回誤差 } e_l) / T$$
$$e_l = e$$

$$\begin{aligned} \text{操作 } u = & \text{比例ゲイン } K_P \times e \\ & + \text{積分ゲイン } K_I \times e_i \\ & + \text{微分ゲイン } K_D \times e_d \end{aligned}$$

デジタル制御とPID

○ デジタル制御の留意点

◇ I 制御

- ・ I 制御で誤差が取り切れないと、積分値がどこまでも大きくなる → 制御不能。

例) 機械的なトラブル

操作の最大値を超えて対応できない

- ・ 対策:

操作が飽和しない程度に積分値の上限下限を設定しておく。

デジタル制御とPID

○ デジタル制御の留意点

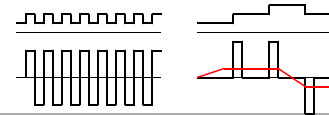
◇ D 制御

- ・ ノイズの問題 (前述)
- ・ 微分値の突発化 (→ 第7回信号処理p17)

- ・ 対策:

デジタルローパスフィルタでノイズ低減など

※ただし、フィルタが強すぎると微分の機能がなくなる

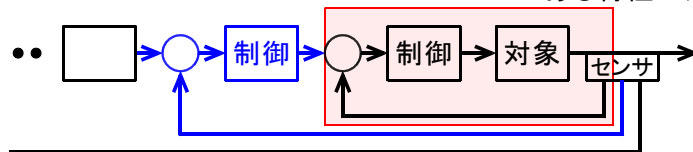


制御の多重化

○ 入れ子の制御

◇ FB制御の輪を多重化する

制御こみで
ある特性の対象



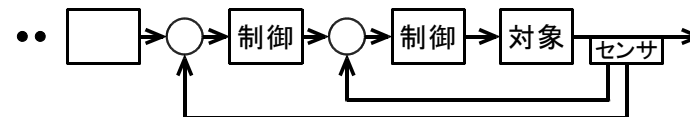
- ・ [内側の制御]を一つの対象と見て、その外に制御を追加する。
- ・ さらにその外に重ねて....。

制御の多重化

○ 入れ子の制御

◇ 留意点

- ・ 制御を切り分け/安定化させやすい。
- ・ 内側の制御は、それを利用する制御に対して、十分に応答が速い必要あり。
(俗に10倍程度)
- ・ トータルでは応答速度が落ちやすい。



今回の目的

○ 制御の基礎

テーマ1: 制御の目的と基本

- ・ 制御するとは
- ・ 制御の基本（フィードバック、PID）

テーマ2: 少し特殊な制御

- ・ フィードフォワード
- ・ 非線形制御

テーマ3: 制御の実例

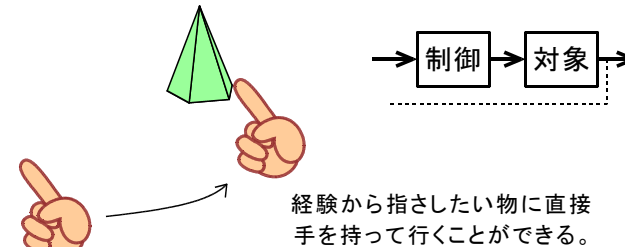
- ・ モータの制御、ロボット制御

フィードフォワード制御

○ フィードフォワードの概念

◇フィードバックしない制御

- ・ 直に操作量を決める。
- ・ センサなし。(注:「センサレス」は意味が異なる)



フィードフォワード制御

○ フィードフォワードの特徴

◇速い

- ・ 操作→センサの検出を待たない。
- ・ 目標に寄せていく時間がかからない。

◇柔軟性

- ・ FBでは対応に困る**対象の癖**を
あらかじめ補正することができる。

◇精度は悪い

- ・ 少しの状況変化でも誤差が生じやすい。

フィードフォワード制御

○ フィードフォワードを使う

◇対象の特性

- ・ こう操作したら動く、という関係が必須。
- ・ 対象の原理/モデル解析による、
入出力特性の測定、学習

◇特性に応じた操作を生成

- ・ 「操作→結果」の逆 × 望む結果

◇フィードバックと併用

- ・ FFだけでは誤差が避けられない。

フィードフォワード制御

○ フィードフォワードの例

◇ステッピングモータ

- ・電流の切り替え回数だけ回る。

◇マニピュレータ、脚歩行ロボット

- ・関節の角度制御はフィードバック。
 - ・手先、脚先の位置に対する関節角度はフィードフォワード。
- (内部計算がFB的な場合あり)

フィードフォワード制御

○ フィードフォワードの例

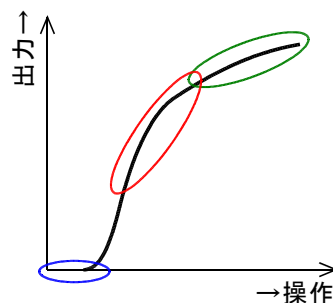
◇自動車の運転 / FBからFFへ

- ・初めて乗った頃:
車の動きを見ながらハンドルを回す。
速度を見ながらペダル操作を考える。
↓
- ・慣れると(特に通い慣れた道):
感覚的にハンドル、ペダル操作。
ギアの変速も半ばパターン化。

非線形制御

○ 線形ではない(≡比例しない)対象

◇操作→結果が非線形

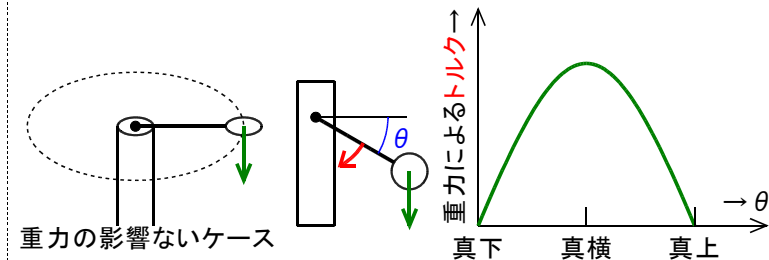


- ・ **比例しない。**
 - ・ **頭打ち**になってくる。
 - ・ 少しの操作では出力が上がらない。
(**不感帯**)
- ※下がってくる、は別の問題が生じる

非線形制御

○ 線形ではない(≡比例しない)対象

◇状態に応じて非線形に余分な負担がある



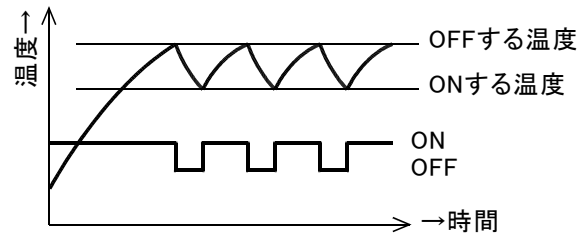
- ・ 腕の角度に応じて、重力で下げるトルク。
- ・ 角度と質量から見積は可能。

非線形制御

○ 線形ではない(≡比例しない)対象

◇操作そのものが非線形 例) オンオフ制御

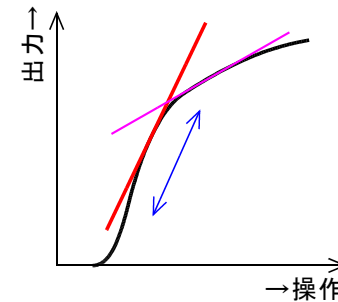
- ・対象によってはそれなりに制御できる
例) 昔のコタツなど



非線形制御

○ 線形化

◇特性の非線形さを部分的に線形とみる

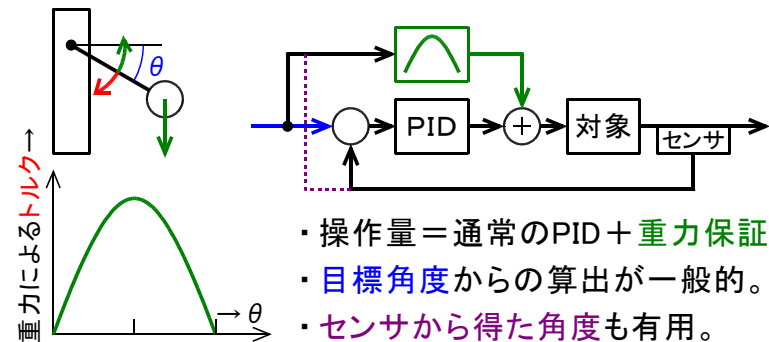


- ・接線を求める。
(計算、実験的)
- ・極端に変化なければある程度の範囲でほぼ一致する。
- ・複数に分割もあり。

非線形制御

○ 非線形補償(FFの一種)

◇非線形に発生する負担をあらかじめ補助



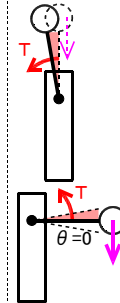
- ・操作量＝通常のPID＋重力保証
- ・目標角度からの算出が一般的。
- ・センサから得た角度も有用。

非線形制御

○ 非線形補償(FFの一種)

◇非線形補償と積分(I)制御

- ・非線形補償はI制御の「誤差を生じる」問題は起きにくい。
- ・非線形補償の計算値と実際の差があるため、非線形補償だけではI制御は不要にはならない。
- ・I制御を減らせる点で、制御の速度を上げやすい。



現代制御理論

○ モデル(特性数式)と数学を駆使した制御

◇古典制御

- ・PID制御などは古典制御と呼ばれる。
- ・**伝達関数**: **周波数特性**で制御理論。
- ・数学的にはラプラス変換。

◇現代制御

- ・最適制御、 H^∞ 制御ほか。
- ・**状態方程式**: **時間特性**(誤差の時間積分など)。
- ・数学的には行列、ベクトルが主。

現代制御理論

○ モデルがあれば高性能

◇モデルと評価関数

- ・線形微分方程式で表される**モデル**。
- ・どの項目を「重く/軽く」**評価**したいか
例) 位置誤差を低減したい、
加速を押さえたい、省エネしたい
を決める数値。
- ・間接的に対象の状態を**推定**する手法。

→ **必ず安定に動作**する制御パラメータ

現代制御理論

○ 適用の難しさ

◇モデル化できればOK

= **モデル化できないとNG**

- ・ガタなど数式化しにくい要素
- ・対象の特性データ

◇あくまで個人的な経験で言えば....

- ・モデル化失敗でうまくいったことがない。
- ・PIDを感覚的に決めたほうが早かった。
→無理に挑戦する必要はとりあえず無し？

今回の目的

○ 制御の基礎

テーマ1: 制御の目的と基本

- ・制御するとは
- ・制御の基本 (フィードバック、PID)

テーマ2: 少し特殊な制御

- ・フィードフォワード
- ・非線形制御

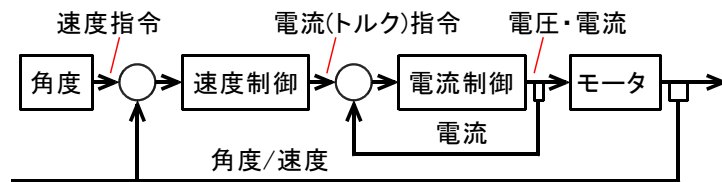
テーマ3: 制御の実例

- ・モータの制御、ロボット制御

ロボット・メカトロ制御の実例

○ モータの制御

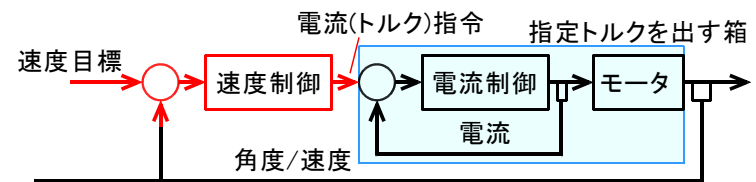
- ◇トルク(電流)制御 (←速度) ← 角度制御
- ・機械の力学特性的に「トルク・力」と運動の関係がはっきりする。



ロボット・メカトロ制御の実例

○ モータの制御

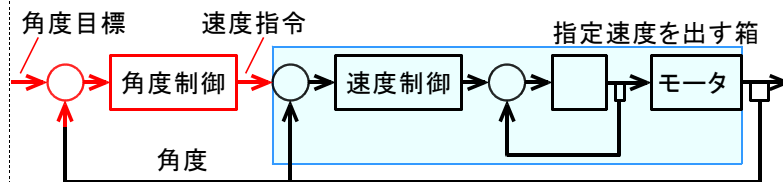
- ◇速度制御→トルク操作
- ・P制御のみでもOKな場合あり。
 - ・PI(定常的に負荷のある場合)、PD(応答向上)



ロボット・メカトロ制御の実例

○ モータの制御

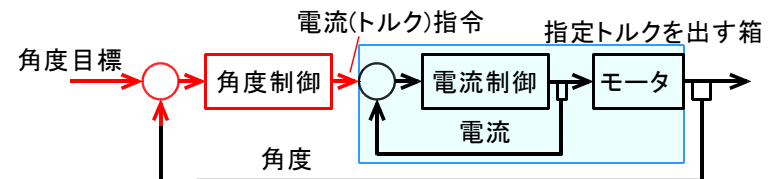
- ◇角度(位置)制御→速度操作
- ・P制御のみで原理的に問題なし。
 - ・PD(応答向上)



ロボット・メカトロ制御の実例

○ モータの制御

- ◇角度(位置)制御→トルク操作
- ・少なくともPD制御が必要、PID。
 - [位置誤差→力]はバネと同じ関係。(前述)
 - ・ゲイン調整は速度FB時より若干難。

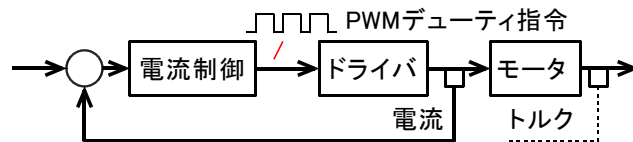


ロボット・メカトロ制御の実例

○ モータの制御

◇トルク(電流)制御→電圧(PWM)指令

- ・モータの**トルク/力**の制御は、モータの**電流**を制御することと同等。
- ・モータの**起電力**(回転速度に比例して電圧発生)に対応するためPI制御が必須。

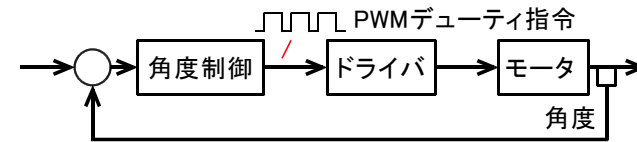


ロボット・メカトロ制御の実例

○ モータの制御

◇速度、位置→電圧(PWM)操作

- ・簡易的によく使われるが、モータの起電力に対処する分だけ要注意 (PI,PID)。
- ・デューティ比は概ね速度に関係する。



ロボット・メカトロ制御の実例

○ モータの制御

◇制御モードの切り替え

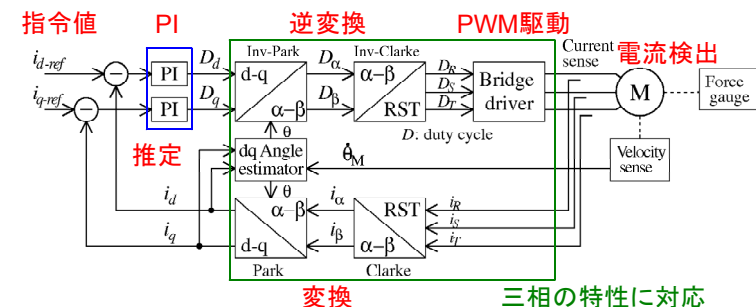
- ・用途に応じてトルク/速度/角度を切り替え
- ・角度制御: 特定の位置決め
- ・速度制御: コンスタントな回転
※角度制御で目標値を連続変化させるより楽
- ・トルク制御: 押しつけ、限界性能出力など
※角度/速度FBだと誤差の蓄積の対処が必要
- ・角度/速度制御+トルクリミットなどもあり。

ロボット・メカトロ制御の実例

○ モータの制御

◇三相モータのベクトル制御

- ・PI制御の回りに複数の変換処理

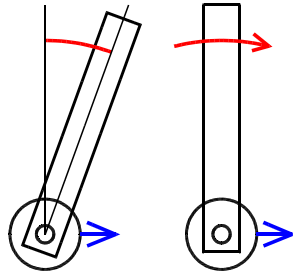
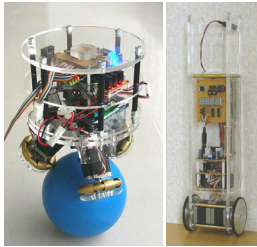


ロボット・メカトロ制御の実例

○ 倒立振子(バランスの制御)

◇姿勢角度の制御

- ・姿勢角度のPD制御→トルク/加速の操作

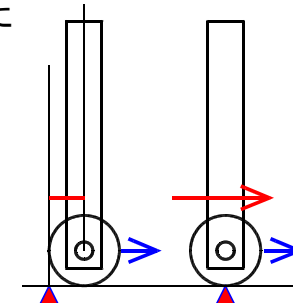
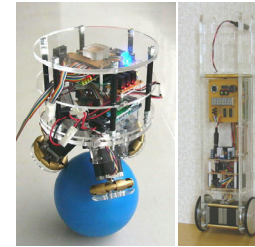


ロボット・メカトロ制御の実例

○ 倒立振子(バランスの制御)

◇位置の制御

- ・位置のPD制御を追加 → 加速度、トルク「遠ざかる方向」に



まとめ

○ 制御の基礎

- ・**制御**は、機械/メカトロに限らず、様々な対象を「思い通りに動かす」ための手法。
- ・主な制御に**フィードバック制御**があり、センサで計測した対象の状態を目標と比較して、一致するように操作する。
- ・代表的なものに**PID制御**～**比例/積分/微分制御**がある。Pを基本に、I/Dの性質を考えた組み合わせが必要。

まとめ

○ 少し高度な制御

- ・制御対象の性質が悪い場合、対象に応じた細工で制御性が改善する場合が多い。
- ・**線形化**や**非線形補償**(特に重力補償)は、比較的容易に**効果を得やすい**。
- ・具体的な対象を制御する場合は、何を使って(操作)、何を制御したいかを考え、適切な制御方法を選定する。