仙台市/仙台市産業振興事業団

ロボット博士の基礎からのメカトロニクスセミナー

C09/Rev 1.01

第９回

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

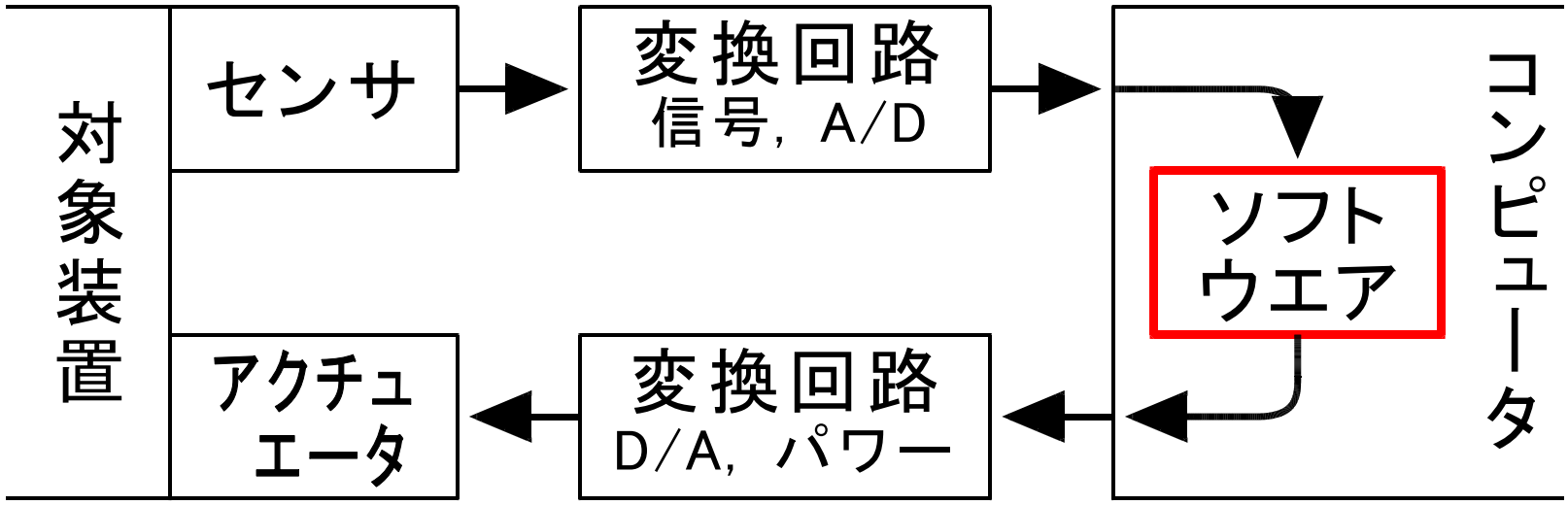
- モータの回転角度 / 回転速度
- 電気炉の温度
- 電圧（電源装置など）
- 腕ロボットの手先の位置姿勢
- 道路での車の渋滞
- コンピュータのサーバの負荷バランス

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エ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

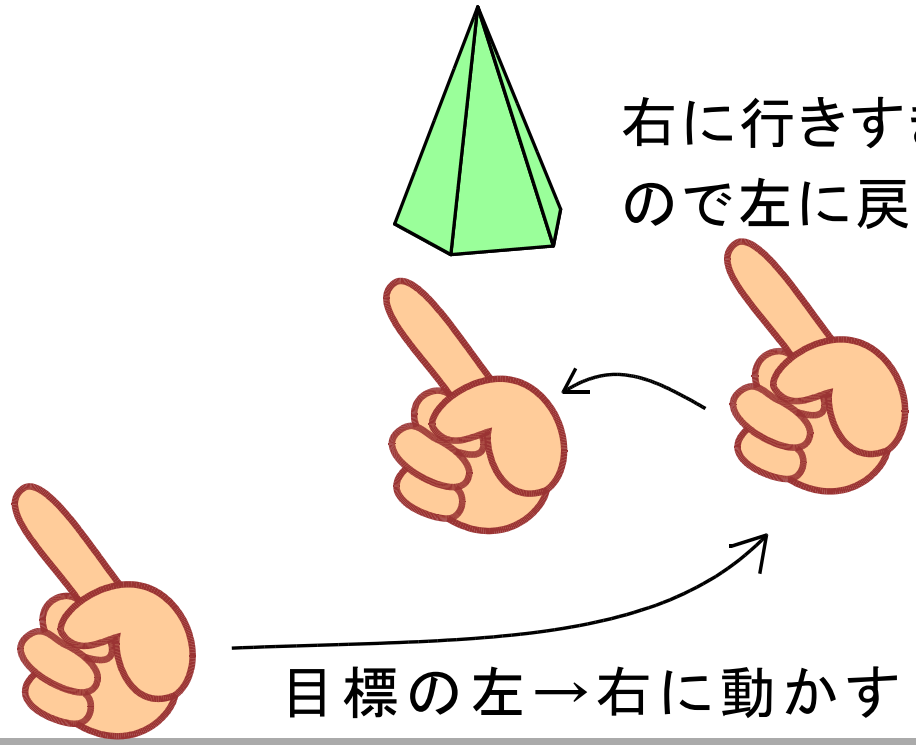
・計測できなければフィードバック制御不可。

・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

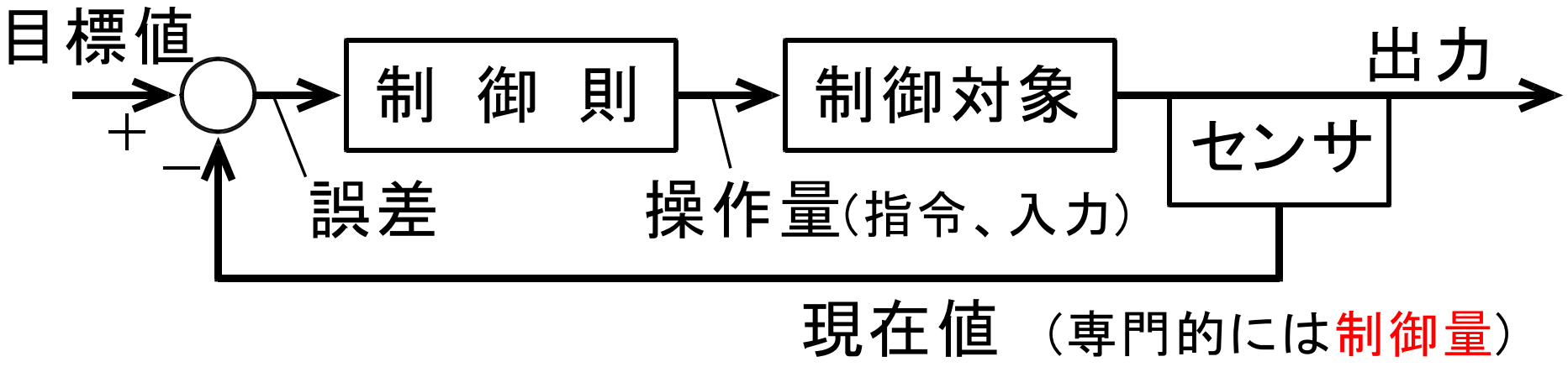
センサで読み取った現在値を、目標に近づけるようにする制御。（○○制御というと、一般にこの一種）



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

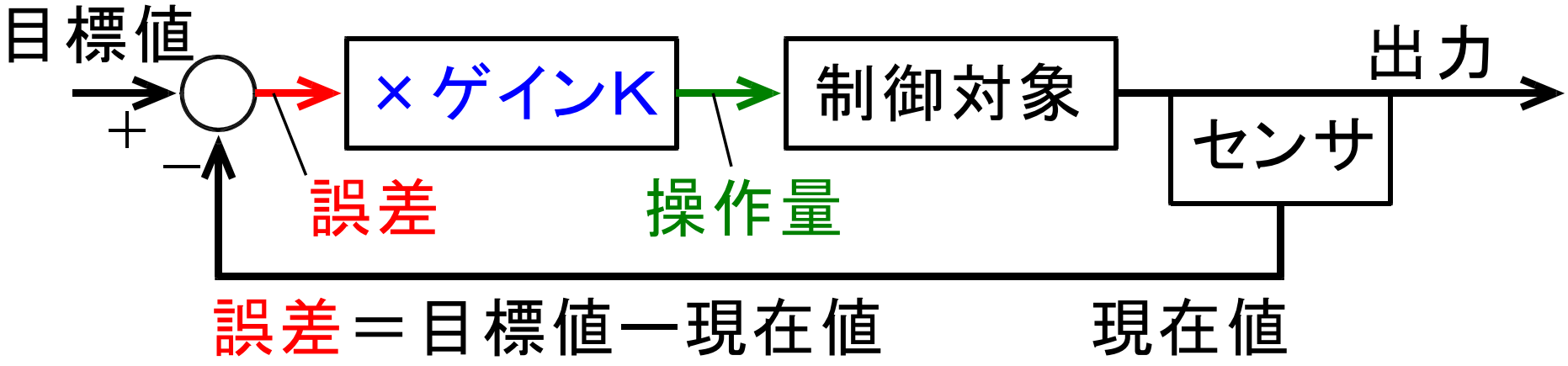
○ 比例制御（P制御）

◇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

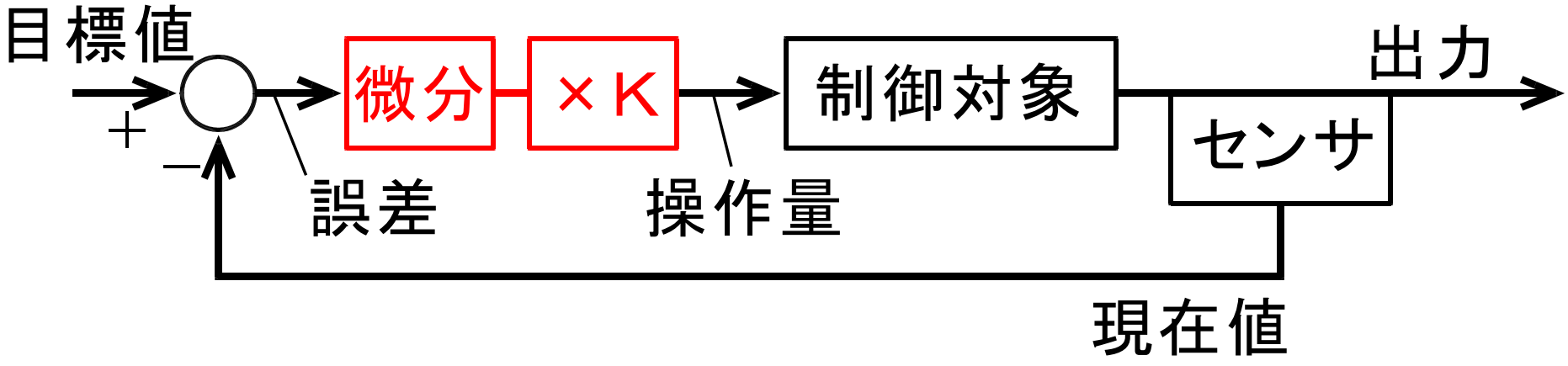


制御の基本

○ 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）

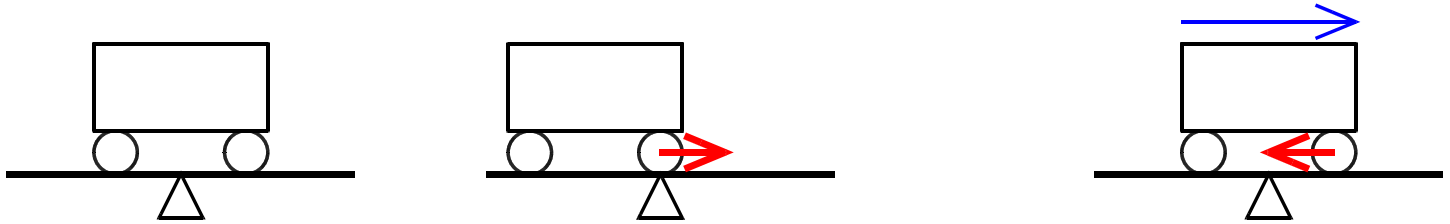


制御の基本

○ 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。



目標

比例制御：
左にずれているから
ずれに比例して、
右に推力の指示

微分制御：
右に通り過ぎようと
しているから、
左に推力の指示

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＝(目標ー現在)

  誤差の時間変化

  ＝(目標の時間変化ー現在の時間変化)

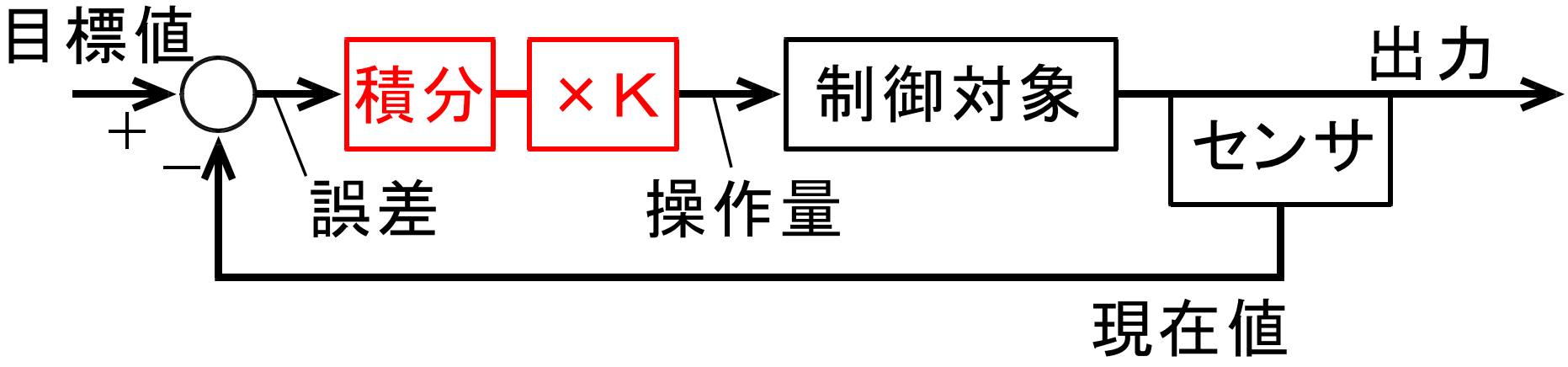
  ＝(目標の速度ー現在の速度)

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：

・誤差がなくならない。

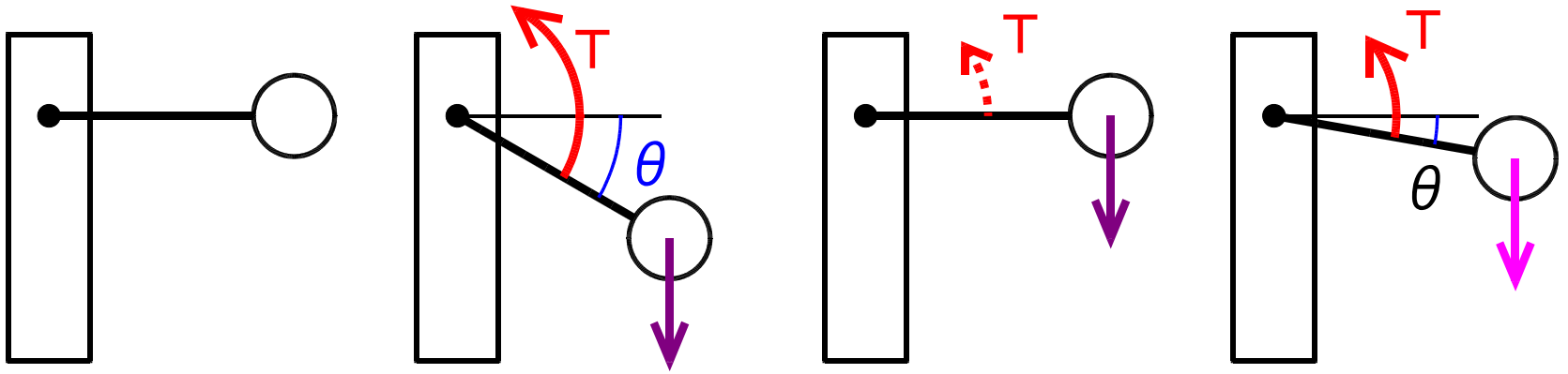
・振動(行き過ぎ)を起こす。

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



目標角度：

比例制御：
角度誤差θ
に比例した
トルクT

動作の実際：
θ＝０だと、
Ｔ＝０
→腕下がる

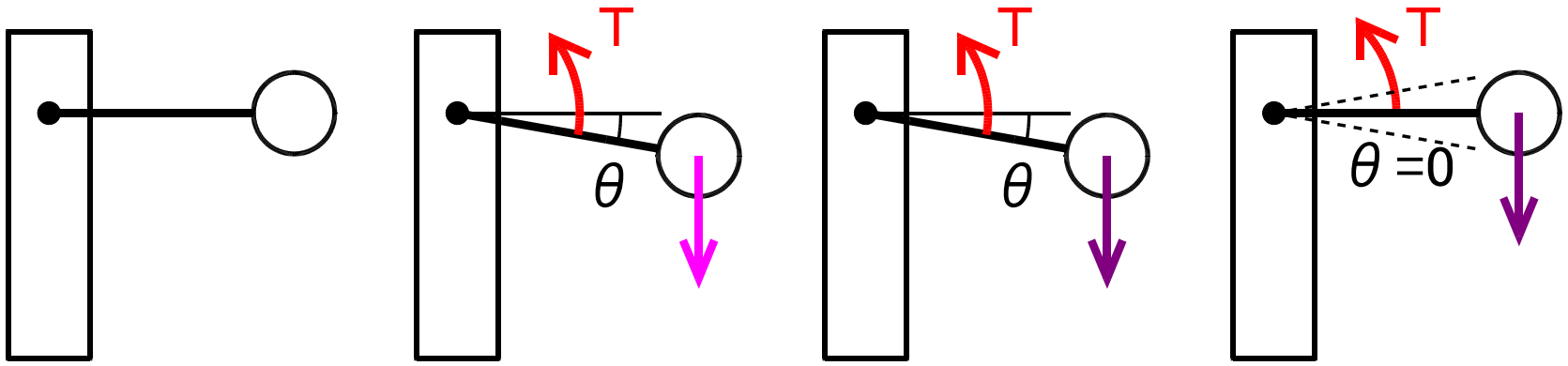
角度誤差：
重力とTが
釣り合うところで
落ち着く

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



目標角度：

角度誤差：
重力とTが
釣り合うところで
落ち着く

誤差積分：
誤差が残り、
積分値に徐々に
溜まる→T増加

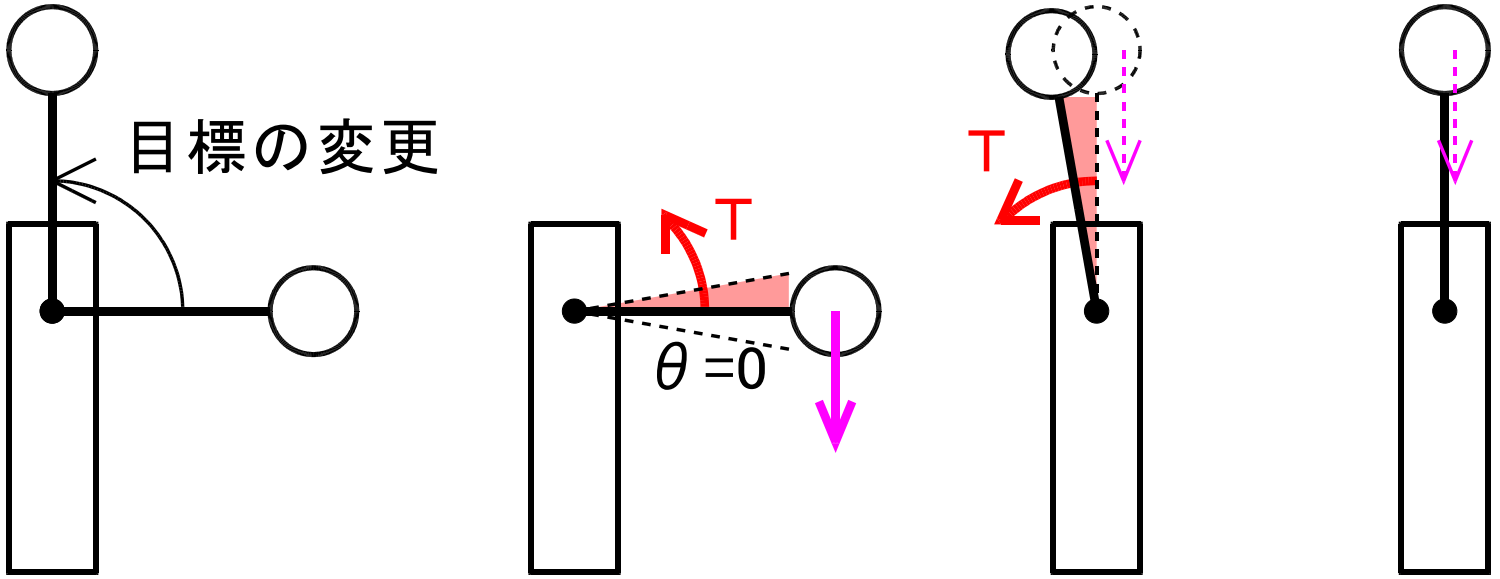
最終的に：
重力分は
I 制御が全て
うけもつ

P制御・D制御・I制御

○ PI制御

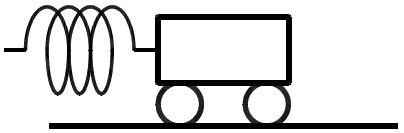
◇PI制御の弱点＝応答の遅れ

・誤差の積分が解消するまでは誤差になる



目標が真上のときは重力分が不要になる→積分によるTが誤差

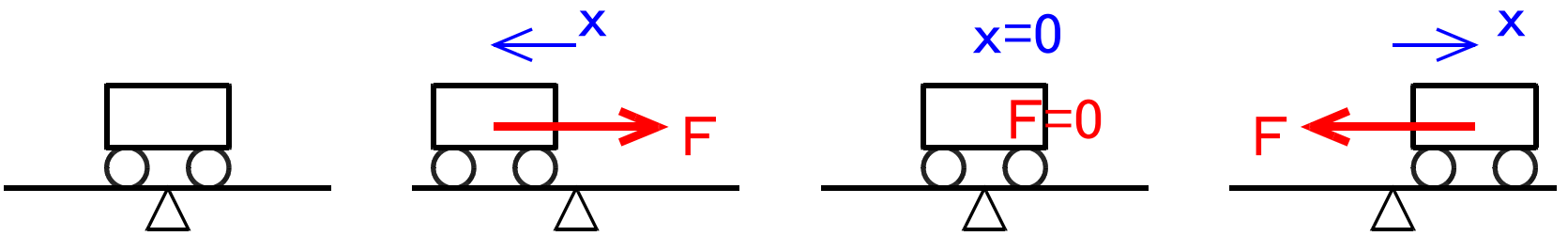
P制御・D制御・I制御



○ 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）



目標位置：

比例制御：
位置誤差x
に比例した
推力F

目標位置：
F=0だが、既に
速度が上がり
通り過ぎる。

行き過ぎ：
また戻そうとす
るが、同じ事を
繰り返す。

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]

10μs(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

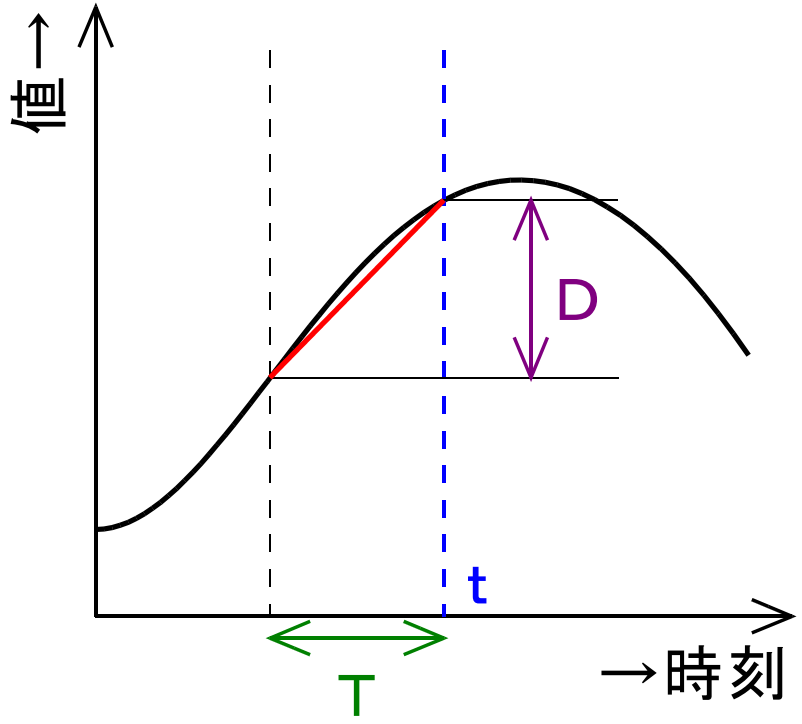
- Ｐ制御：主たる誤差の低減
- Ｉ制御：誤差が残らないようにする
- Ｄ制御：振動の低減
  ブレーキや反応性向上

- 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

ディジタル制御とPID

○ 積分と微分の計算（信号処理の基礎より）

◇微分

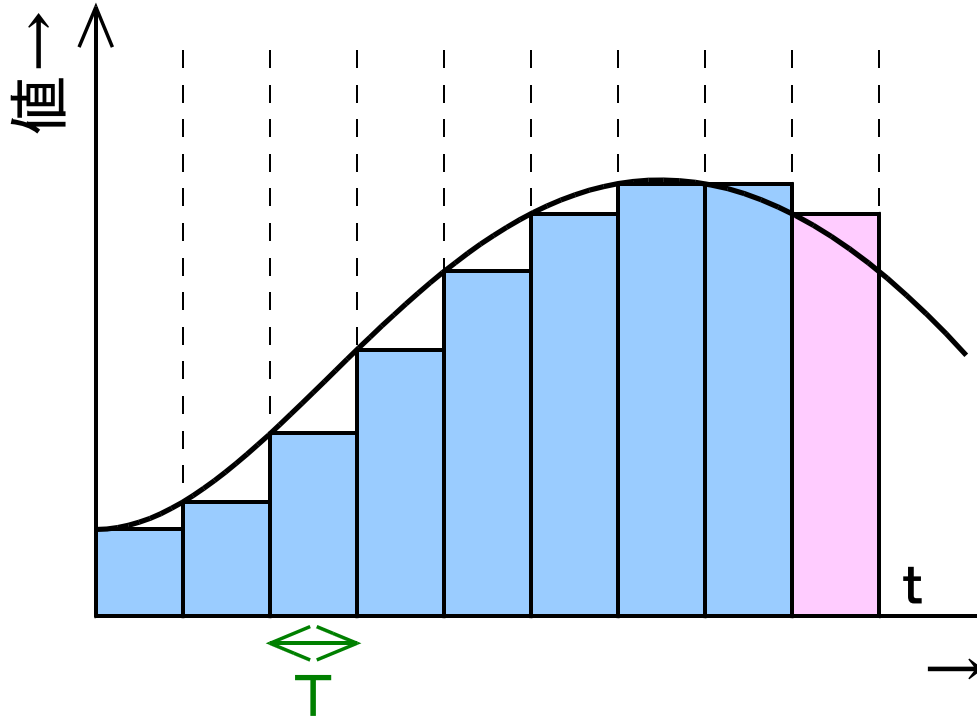


- ある時刻tで間隔Tの間の変化Dを求める。
- D/Tをその瞬間の微分値として使う。

ディジタル制御と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]×T

ディジタル制御と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

誤差e ＝ 目標値r － 現在値y

誤差積分ei ＝ ei ＋ e×T

誤差微分ed ＝（e － 前回誤差el)/T

el ＝ e

操作u＝ 比例ゲインKP × e

＋ 積分ゲインKI × ei

＋ 微分ゲインKD × ed

ディジタル制御とPID

○ ディ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。

ディジタル制御とPID

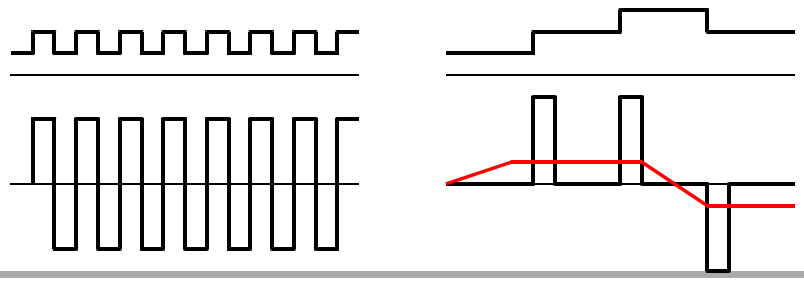
○ ディジタル制御の留意点

◇ 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

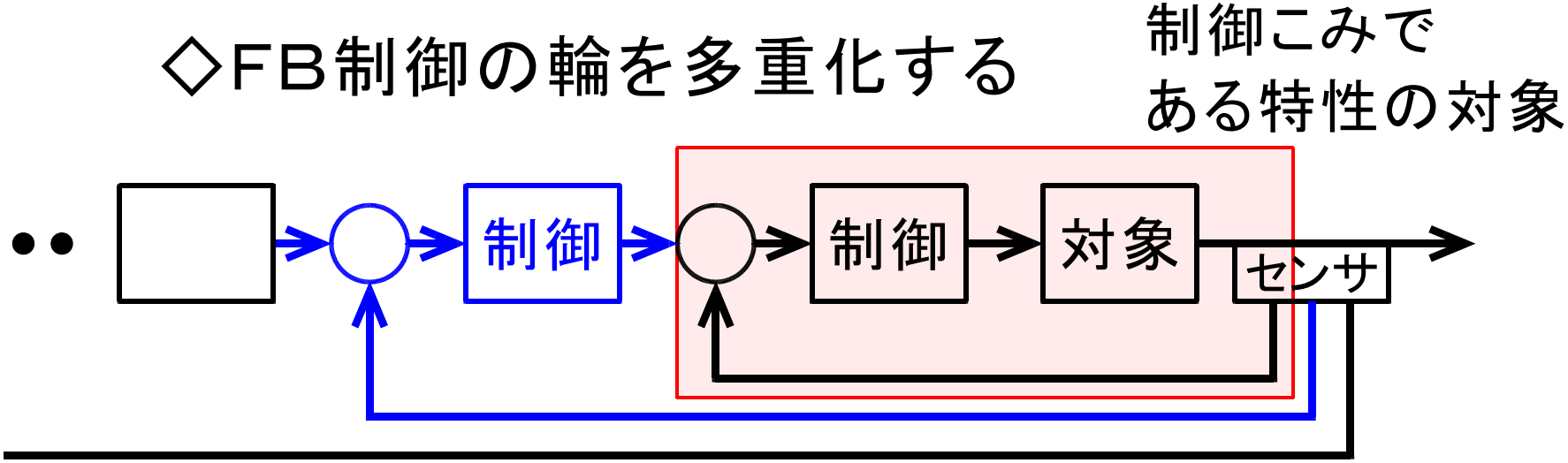
  ※ただし、フィルタが強すぎると微分の機能がなくなる



制御の多重化

○ 入れ子の制御

◇FB制御の輪を多重化する



・[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。

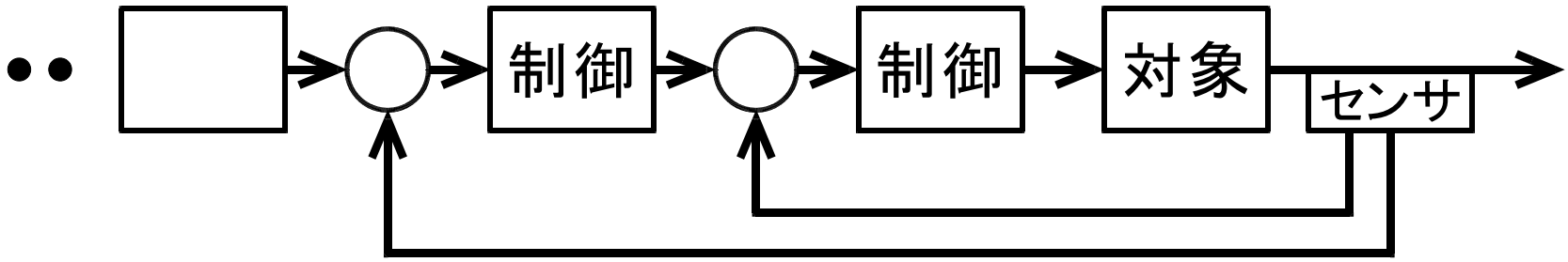
・さらにその外に重ねて....。

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

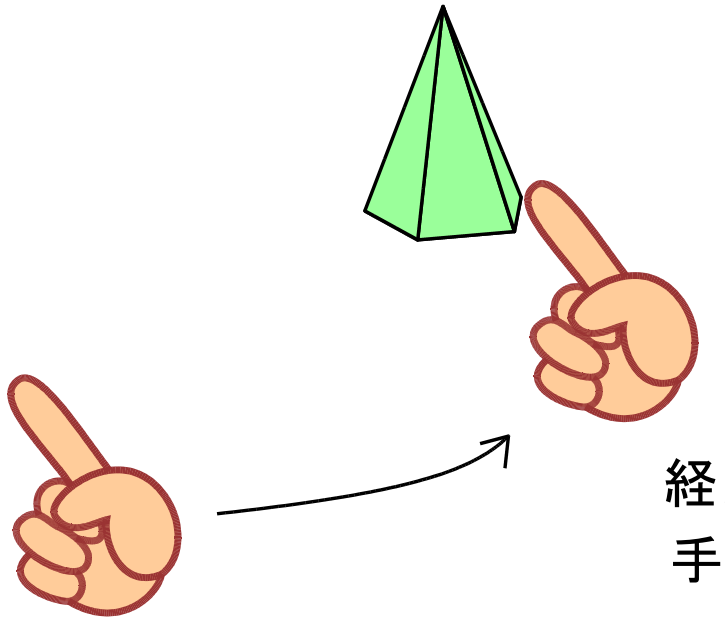
フィードフォワード制御

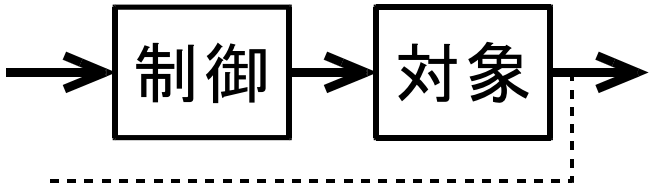
○ フィードフォワードの概念

◇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)





経験から指さしたい物に直接
手を持って行くことができる。

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）:

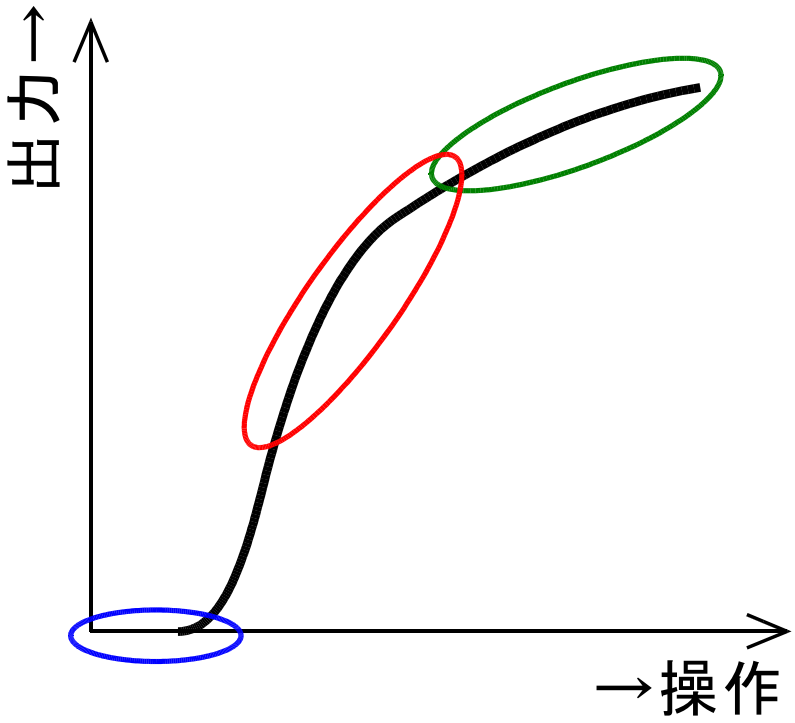
感覚的にハンドル、ペダル操作。

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



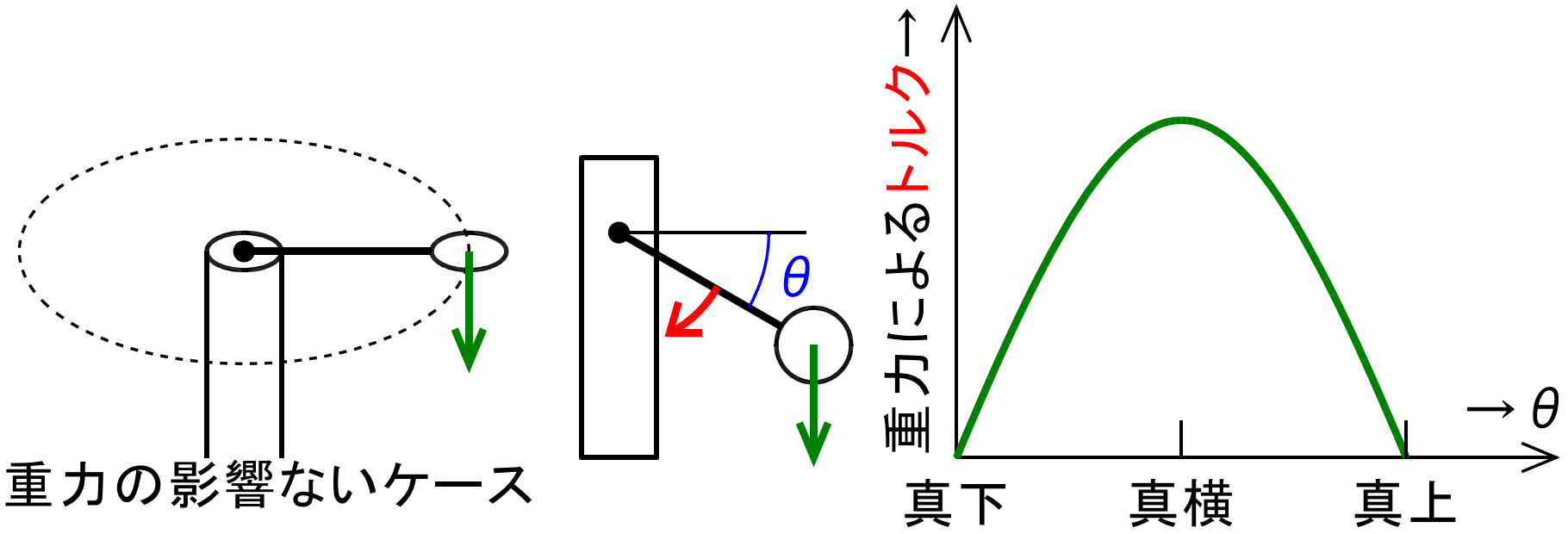
- 比例しない。
- 頭打ちになってくる。
- 少しの操作では
  出力が上がらない。
  (不感帯)

※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。

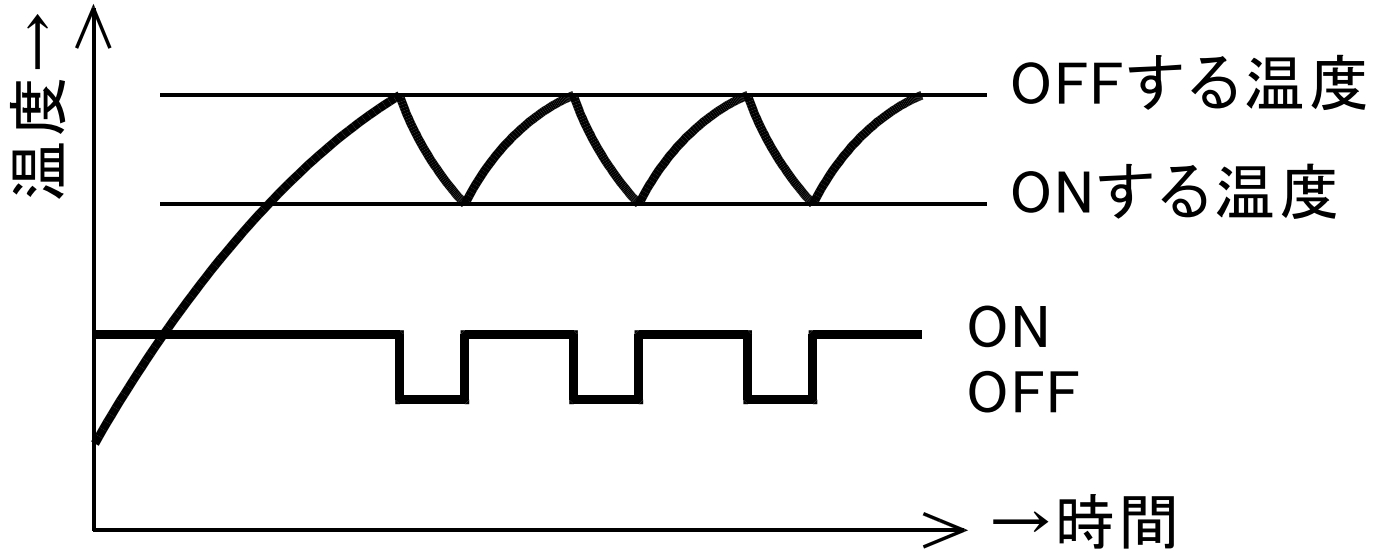
非線形制御

○ 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

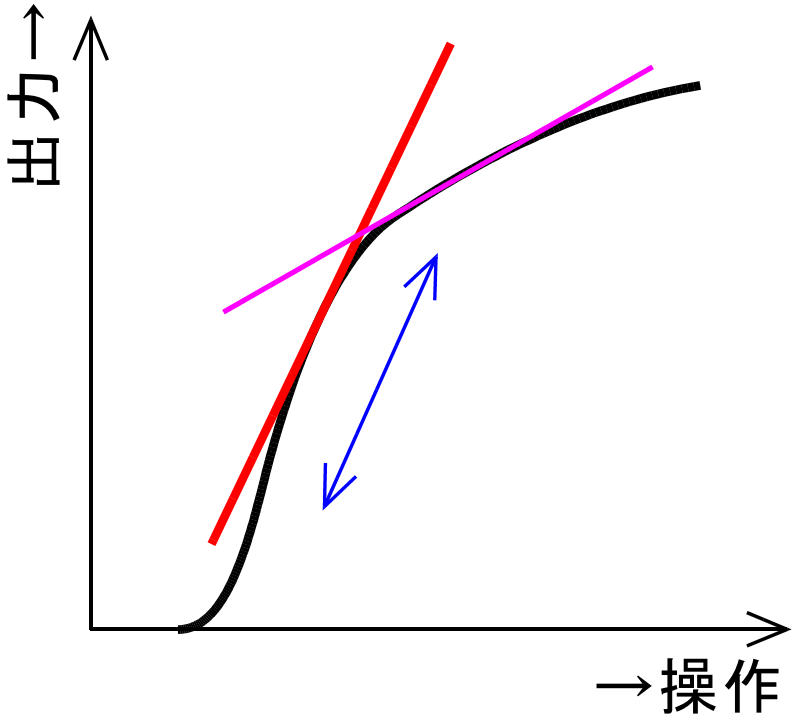
例）昔のコタツなど



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

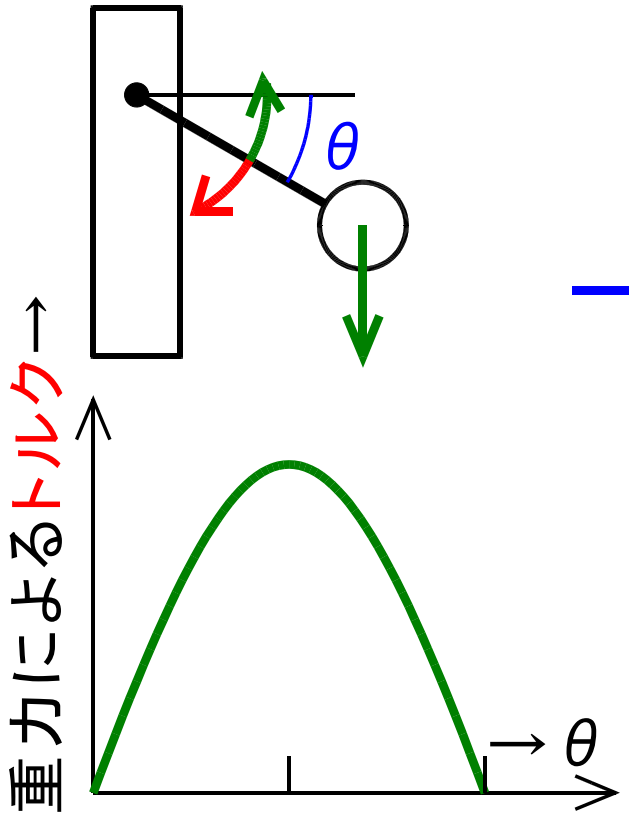


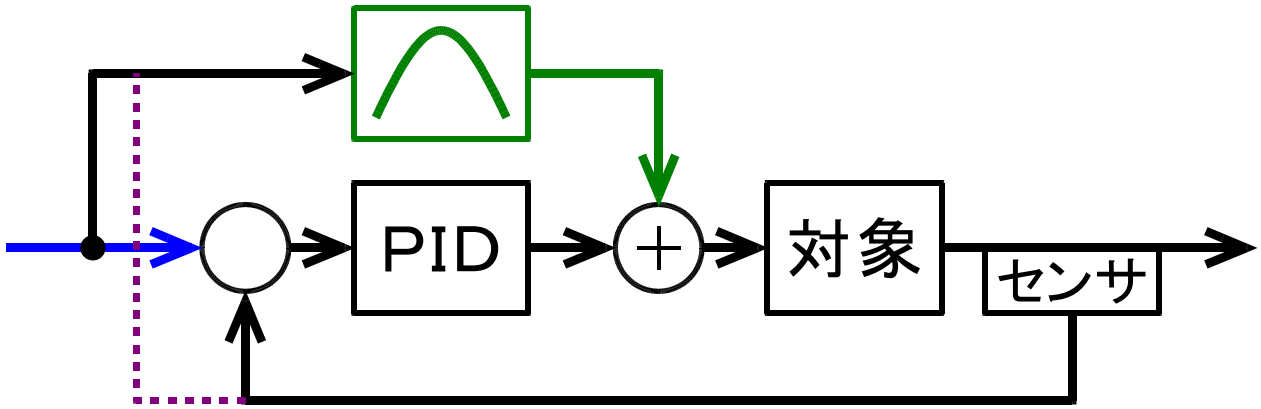
- 接線を求める。
  (計算、実験的)
- 極端に変化なければ
  ある程度の範囲で
  ほぼ一致する。
- 複数に分割もあり。

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



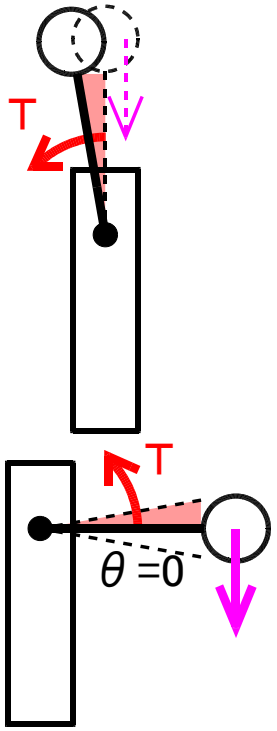


- 操作量＝通常のPID＋重力保証
- 目標角度からの算出が一般的。
- センサから得た角度も有用。

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

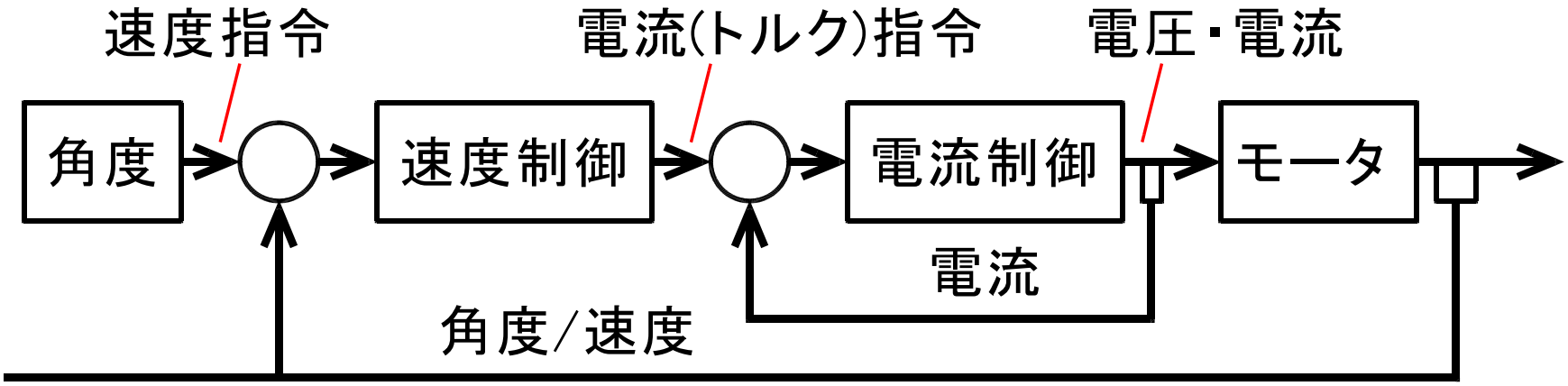
・モータの制御、ロボット制御

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
運動の関係がすっきりする。

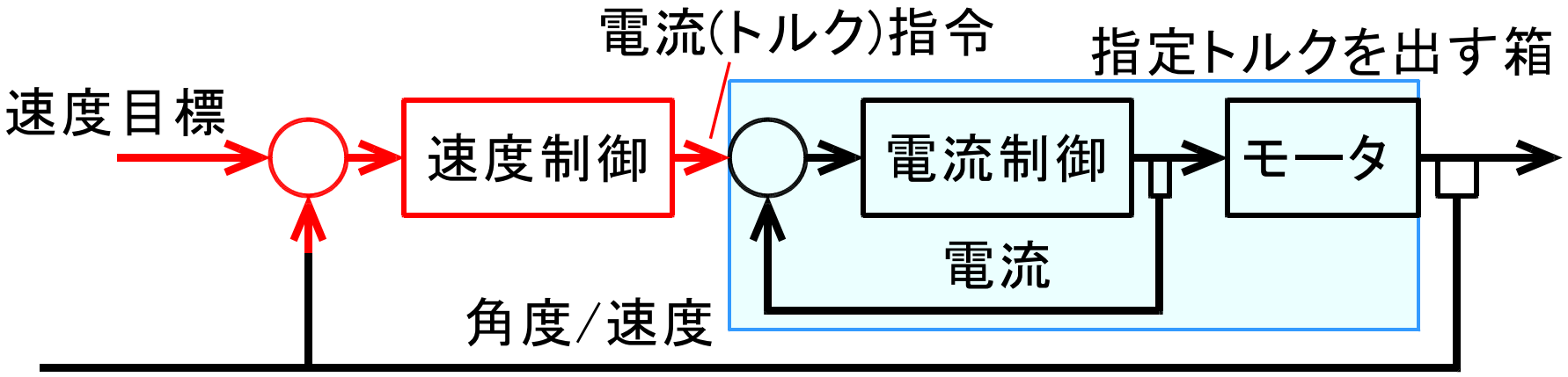


ロボット・メカトロ制御の実例

○ 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)

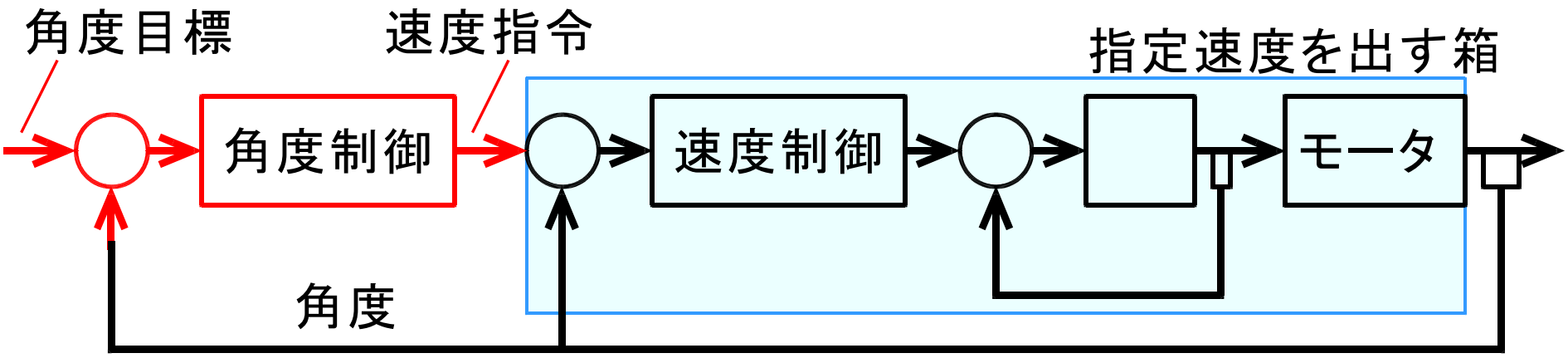


ロボット・メカトロ制御の実例

○ 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

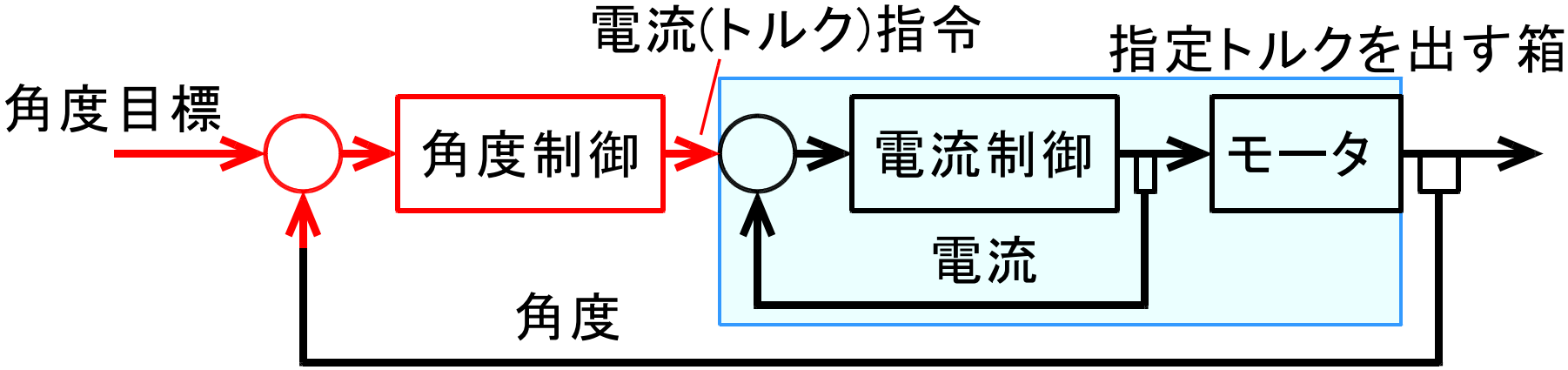
○ モータの制御

◇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。



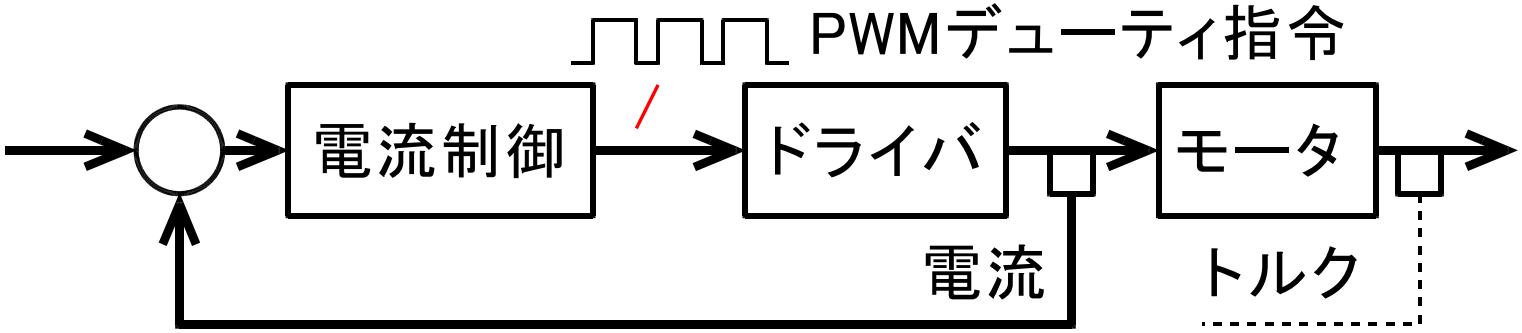
ロボット・メカトロ制御の実例

○ 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。



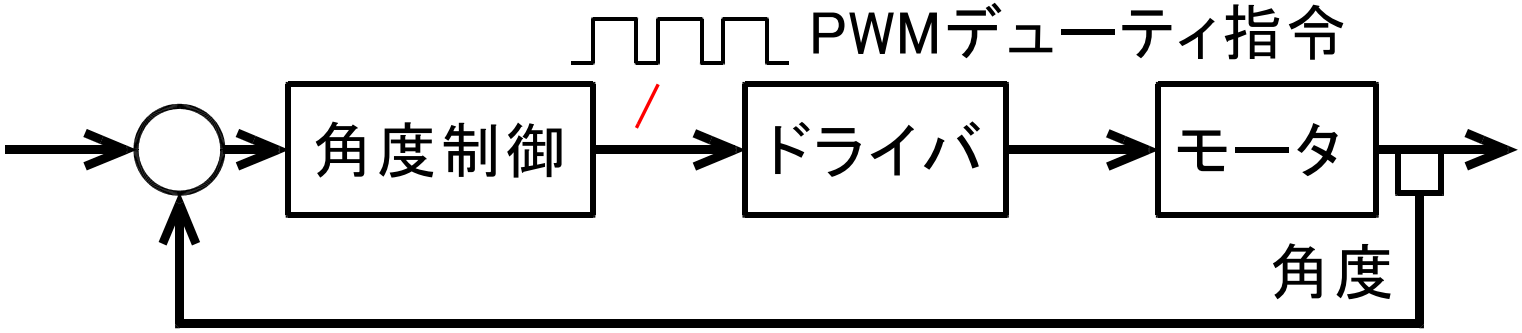
ロボット・メカトロ制御の実例

○ 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。



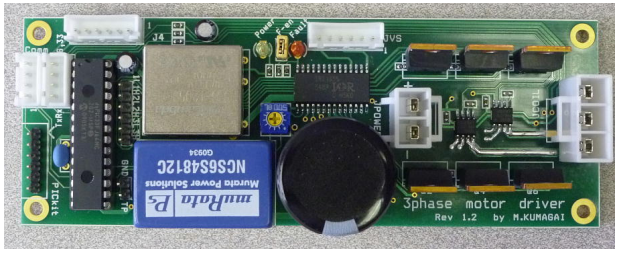
ロボット・メカトロ制御の実例

○ 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。

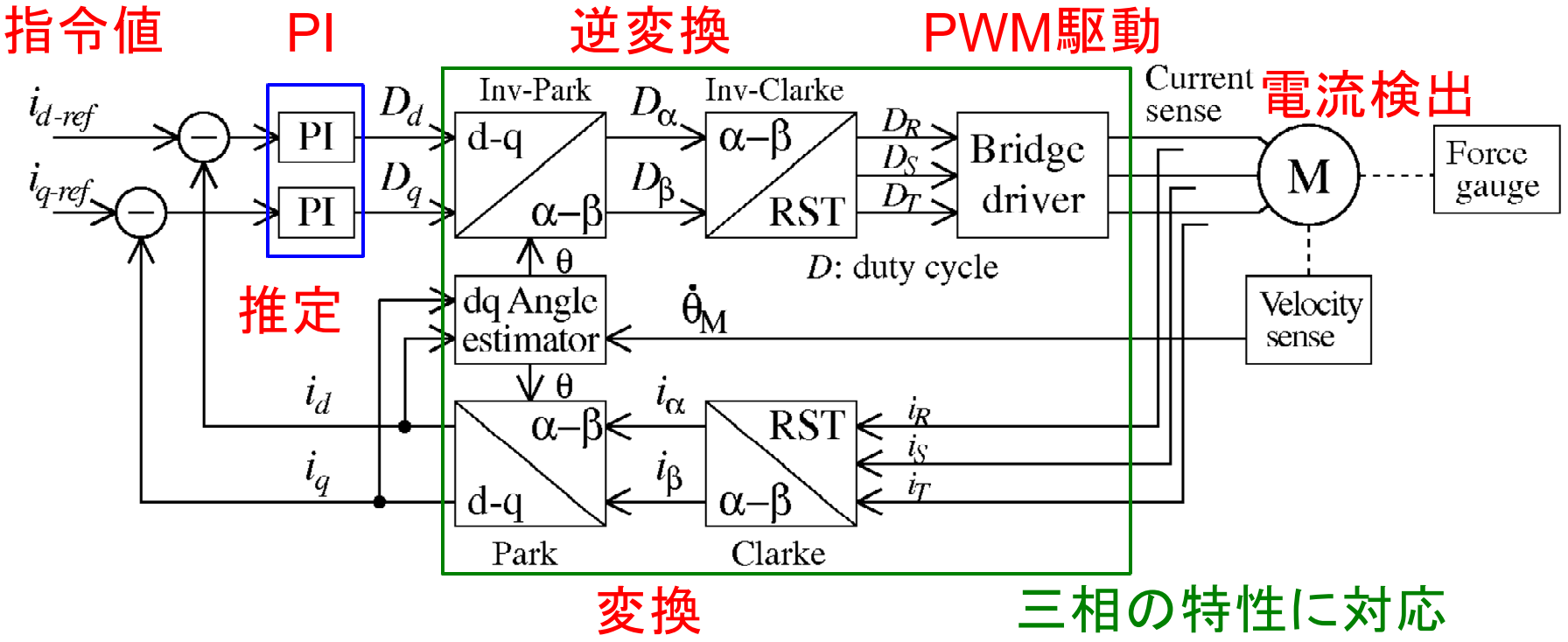
ロボット・メカトロ制御の実例



○ 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

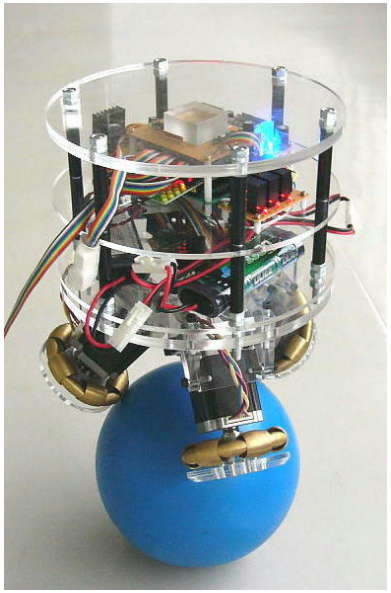


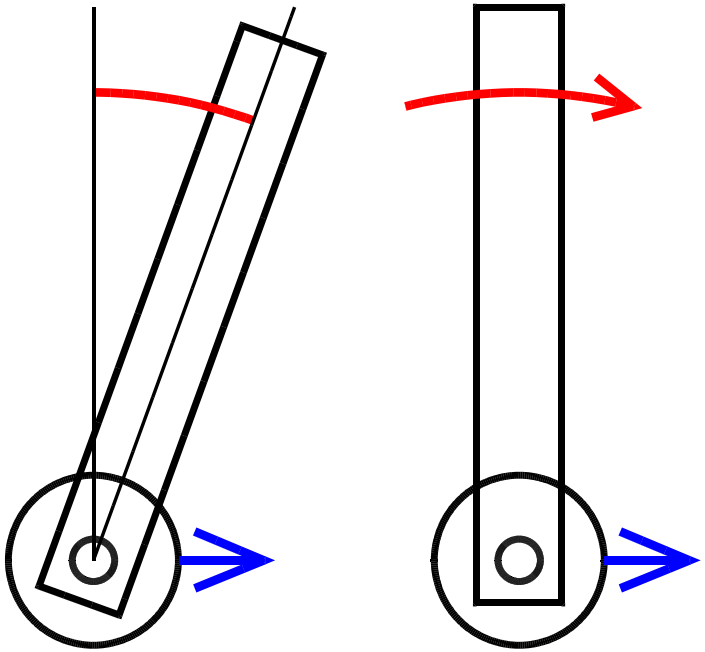
ロボット・メカトロ制御の実例

○ 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





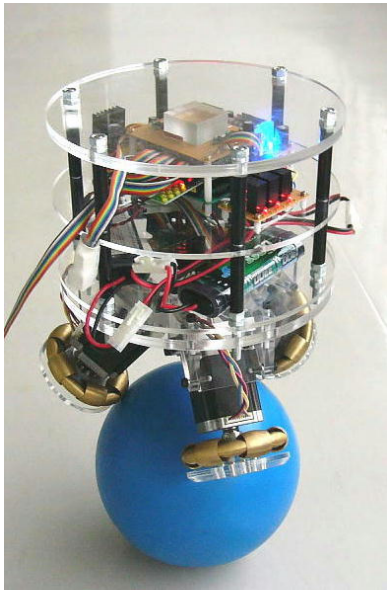
ロボット・メカトロ制御の実例

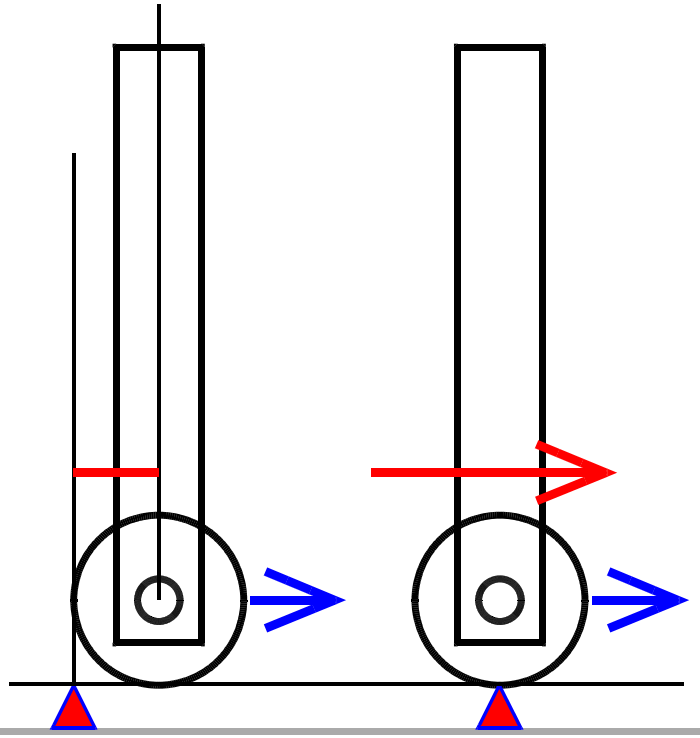
○ 倒立振子(バランスの制御)

◇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。