

現代制御理論と EPICSを使った 倒立振子制御

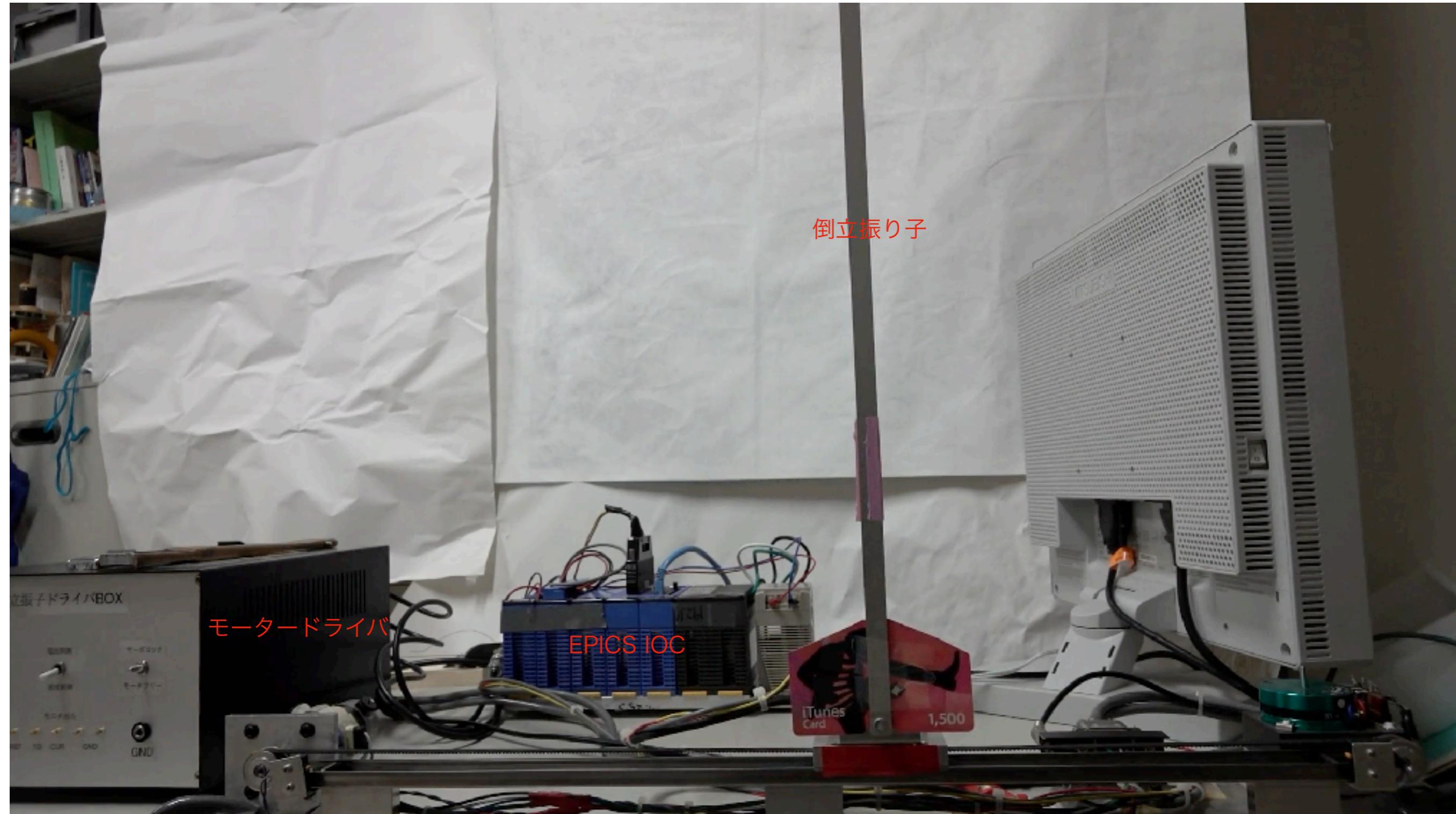
EPICS 講習会2022

2023.3.3

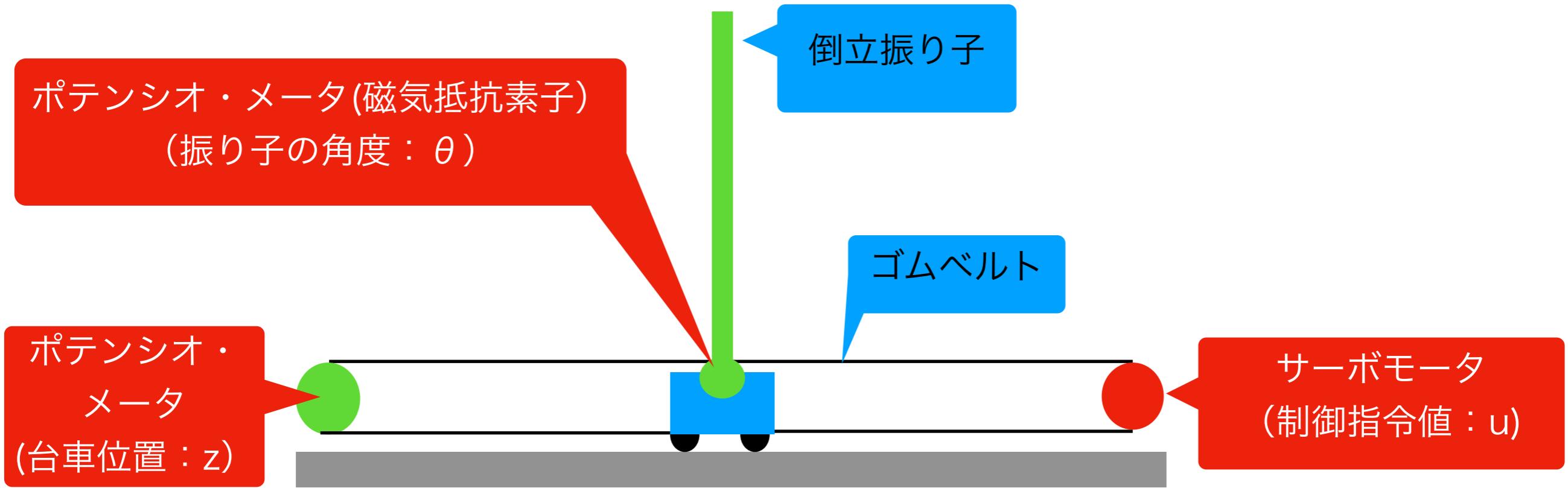
KEK 加速器制御 山本 昇

倒立振り子装置の動作

倒れても、また立ち上ります。



倒立振り子装置の概要



制御：

振り子の角度(θ)と台車の位置(z)を入力に、
制御指令値(u)を適切に与えることで、
振り子の倒立($\theta=0$)と
台車の静止($z=\text{一定}$)を維持する。

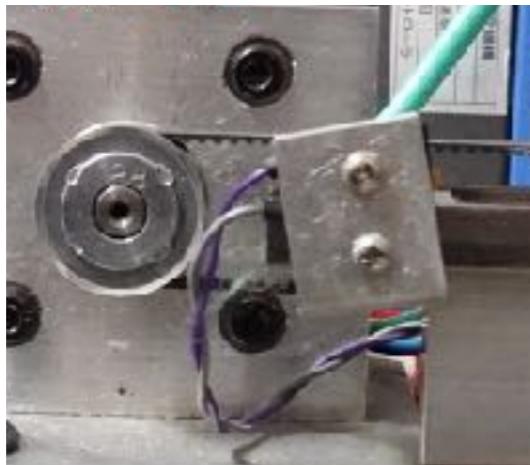


倒立振り子装置

モータドライバ



台車駆動
モータ



リミットスイッチ
(左右一対)



振り子検出
光スイッチ



振り子角度検出
ポテンシオメータ



台車位置検出用
ポテンシオメータ

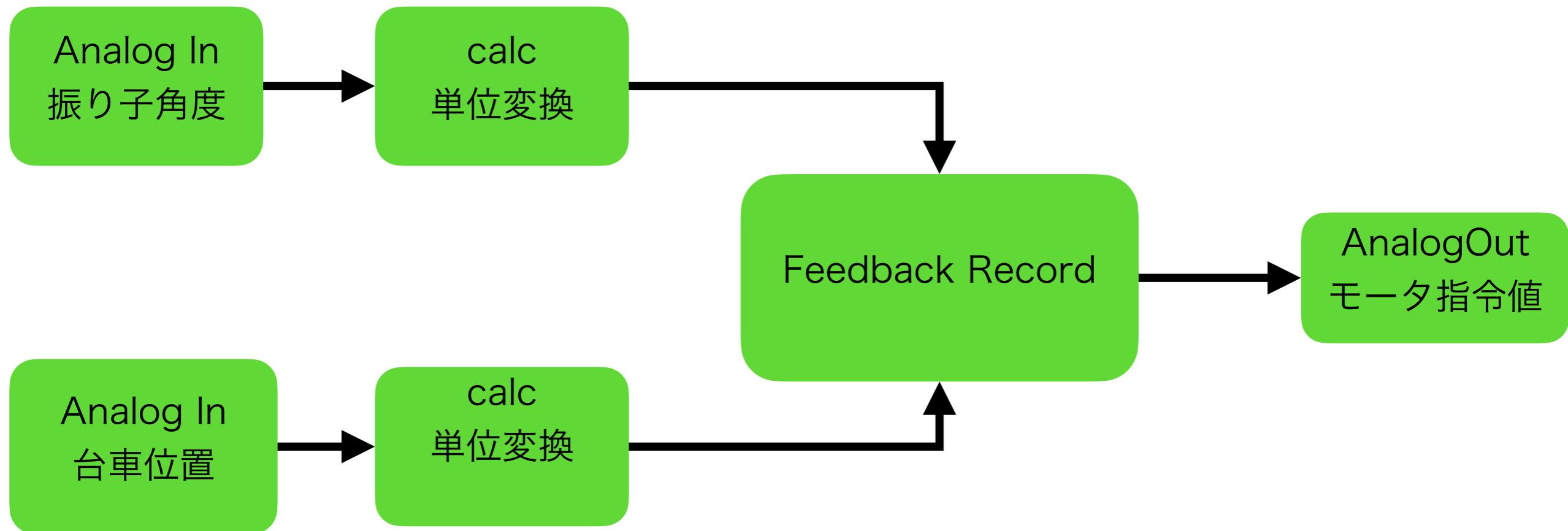
制御装置

PLCベースのEPICS IOC

- ハードウェア
 - Yokogawa PLC(FA-M3/e-RT)
 - CPU: RP-61 (Linux based プログラマブルコントローラ)
 - IO modules : ADC, DAC, DI
 - モータードライバ ボックス
- ソフトウェア
 - EPICS DB :
 - 入出力 : FA-M3用のデバイスサポート
 - カスタムレコード : 倒立振り子制御専用のレコード
 - EPICS sequencer: 後述

EPICS DB

Feedback Record



Feedback レコードは、

振り子の角度と、台車の位置の現在値と、過去の履歴から、

現在の振り子の角速度と速度を推定し、

それらの値に基づいて台車を駆動するモータの指令値を出力する。

EPICS IOC

sequencer

IOCでは以下のsequencerが動作している。

- ・台車がレール端に到達した場合に、
- ・倒立フィードバックを停止し、
- ・台車を中心位置に戻し、
- ・振り子を振りあげて、
- ・倒立状態を復旧する。

制御則: 状態推定器付き
状態フィードバック

振り子装置の運動方程式と制御則

線形化された運動方程式 + 速度フィードバック付きモータドライバ

$$\frac{d}{dt} \begin{pmatrix} z \\ \theta \\ \dot{z} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 0, & 0, & 1, & 0 \\ 0, & 0, & 0, & 1 \\ 0, & 0, & -\eta, & 0 \\ 0, & \frac{mgl}{(I+ml^2)}, & \frac{-\eta ml}{(I+ml^2)}, & \frac{-\mu_\theta}{(I+ml^2)} \end{pmatrix} \begin{pmatrix} z \\ \theta \\ \dot{z} \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \xi \\ \frac{ml\xi}{(I+ml^2)} \end{pmatrix} u$$

状態変数 : X

$$X = \begin{pmatrix} z \\ \theta \\ \dot{z} \\ \dot{\theta} \end{pmatrix}$$

観測量 : y

$$y = (z, \theta) = \begin{pmatrix} 1, 0, 0, 0 \\ 0, 1, 0, 0 \end{pmatrix} \begin{pmatrix} z \\ \theta \\ \dot{z} \\ \dot{\theta} \end{pmatrix}$$

状態フィードバック

状態方程式

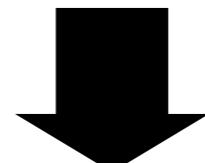
$$\frac{dX}{dt} = AX + Bu$$

$$+$$

制御則：

状態フィードバック

$$Y = CX + Du$$



$$\frac{dX}{dt} = (A - BK) X$$

$A - BK$ の全ての固有値の実部が負の時

$X \rightarrow 0$ as $t \rightarrow \infty$

$$Y = (C - DK) X$$

制御可?

フィードバックゲインは存在するのか？

そもそも、使える情報は、

制御値 u と測定値 Y だけ

制御には 状態変数 X が必要

状態フィードバック

状態方程式の形式的な解

$$\frac{dX(t)}{dt} = AX(t) + Bu(t)$$

を積分型で書くと、

$$X(t) = e^{At}X(0) + \int_{-\infty}^t d\tau e^{A(t-\tau)}Bu(\tau)$$

$$= e^{At}X(0) + \int_{-\infty}^0 d\tau e^{-A\tau}Bu(t+\tau)$$

状態フィードバック

可制御性

状態 X の次元が n あるとき、

$$[B, AB, A^2B, \dots, A^{n-1}B]$$

のランクが n の時、系は”可制御”と呼ばれる。

ランクが n 以下ということは、

$u(t)$ を変えて、変更できない状態空間の方向が存在するということ

系が可制御であるとき、

$A - BK$ のすべての固有値の実部が負となる K が存在する。

ちょっと横道： 現代制御と古典制御

状態空間表示と伝達関数表示

ちょっと横道

状態空間表示：現代制御理論

伝達関数表示：

$$\begin{cases} \frac{dX}{dt} = Ax + Bu \\ Y = CX + Du \end{cases} \xrightarrow{\text{ss2tf}} Y(s) = \frac{b_M s^M + b_{M-1} s^{M-1} \dots + b_1 s + b_0}{s^N + a_{N-1} s^{N-1} + \dots + a_1 s + a_0} U(s)$$
$$\xleftarrow{\text{tf2ss}}$$

$$Y(s) = [C(sI - A)^{-1} B + D] U(s)$$

- ・プロパーなシステム($M \leq N$)では
状態空間表示と伝達関数表示を相互に変換可能。
- ・それぞれに tf2ss/ss2tf といった関数が利用できる。

状態空間表示と伝達関数表示

SISOシステムでは、つぎの二つの伝達関数は同じになる。

$$Y(s) = \left[C(sI - A)^{-1} B + D \right] U(s) \quad Y(s) = \left[B^T (sI - A^T)^{-1} C^T + D \right] U(s)$$

$$\begin{cases} \frac{dX}{dt} = Ax + Bu \\ Y = CX + Du \end{cases}$$

可制御正規型



$$\begin{cases} \frac{dX}{dt} = A^T x + C^T u \\ Y = B^T X + Du \end{cases}$$

可観測正規系

つまり、プロパーな伝達関数を持つシステムは、
状態空間表示でみると、可制御かつ可観測なシステムに対応している。

本道に帰ります

最適レギュレータ法

フィードバックゲイン K を求める

目標積分：これを最小化する K を使う。

$$I = \int_{t_0}^{t_f} dt \{x^T Q x + u^T R u\}$$

フィードバックゲイン : K

$$K = R^{-1} B^T P$$

代数的リカッチ方程式 (Riccati algebraic equation)

$$PA + A^T P - PBR^{-1}B^T P + Q = 0$$

P は正定値対称な解

状態推定器

状態方程式

$$\frac{dX}{dt} = Ax + Bu$$

$$Y = CX + Du$$

に対して、状態の推定値 \hat{X} を次の式で計算してみる、

$$\frac{d\hat{X}}{dt} = A\hat{X} + Bu + H(Y - C\hat{X} - Du)$$

$e \equiv X - \hat{X}$ を考えると、

H を選択することで、

$$\frac{de}{dt} = (A - HC)e$$

$$\lim_{t \rightarrow \infty} e = 0$$

状態推定器のゲイン

状態推定器のゲインは、適当な仮定のもとで、次の
リカッチ方程式

$$AP_s + P_s A^T - P_s C^T V^{-1} C P_s + W = 0$$

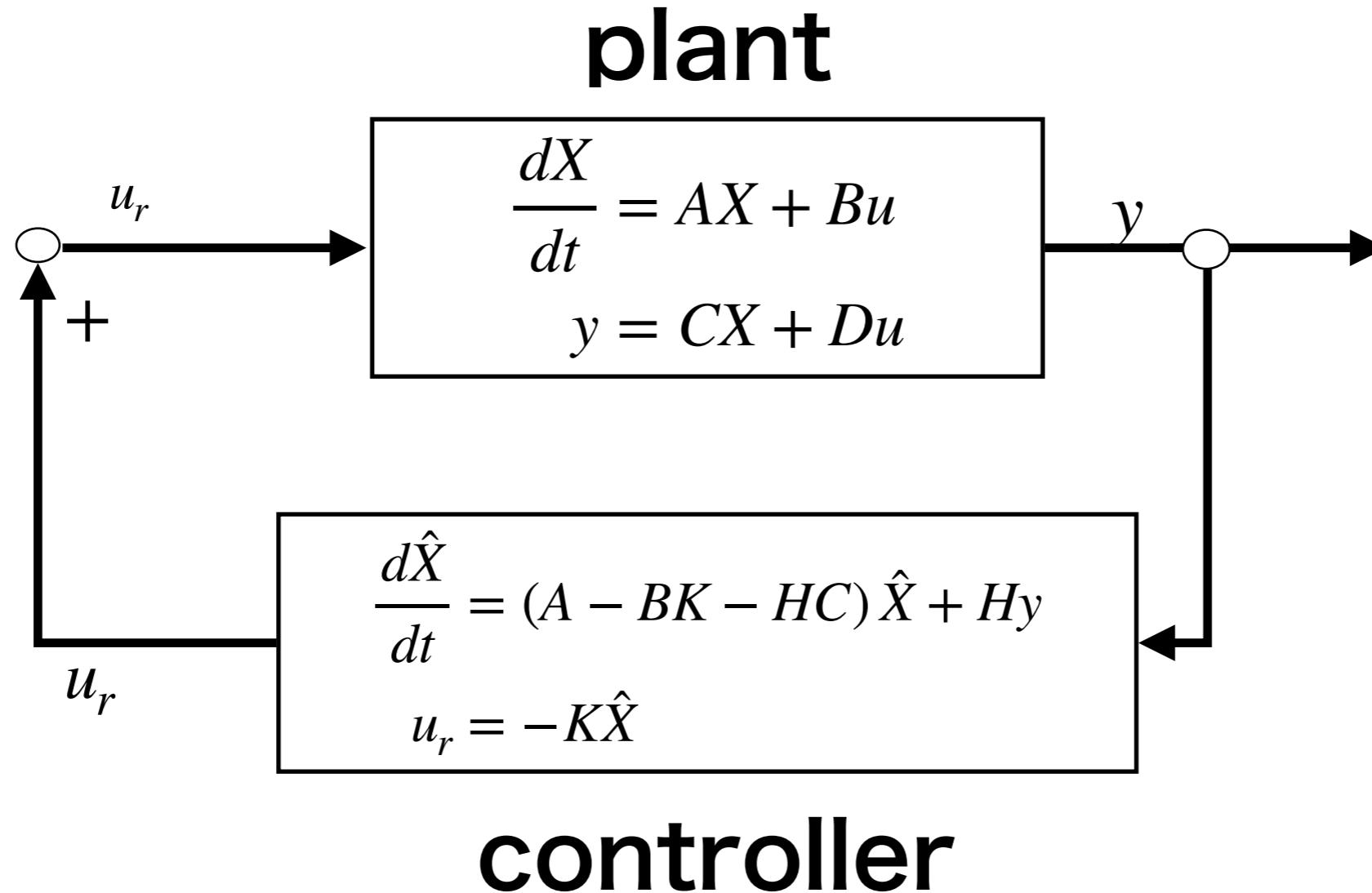
の 正定対称な解 P_s を用いて、

$$H = P_s C^T V^{-1}$$

としてよい。

Feedback systemのダイアグラム

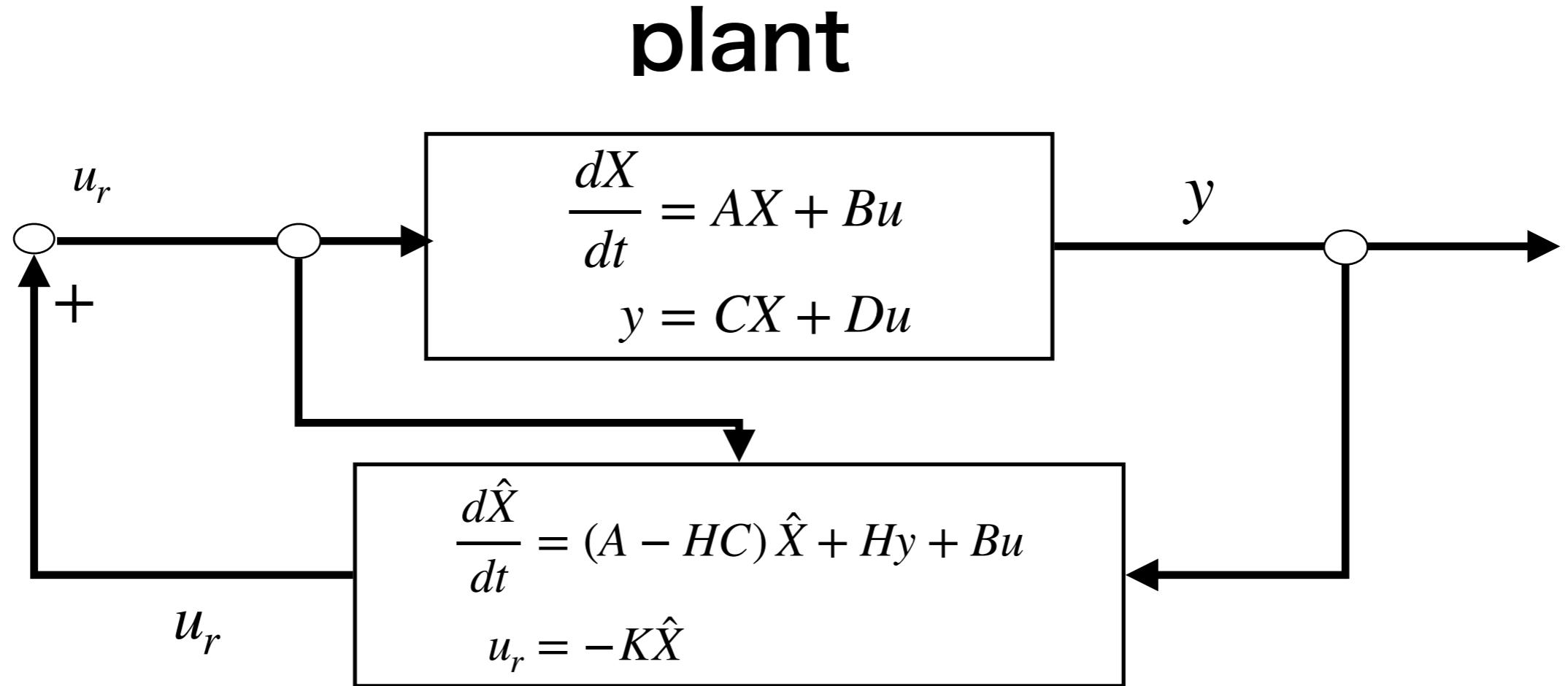
コントローラは対象システム(plant)の出力から、状態量を推定、推定状態量に基づき、制御量を決定する。



状態推定器の状態の更新には、自身の出力計算値 u_r を使っている。

Feedback systemのダイアグラム

コントローラは対象システム(plant)の出力から、状態量を推定、推定状態量に基づき、制御量を決定する。



controller(Observer)

Controllerで指令値 u を読み込んで、状態推定器の更新を行う。

Feedback record

controller

$$\begin{aligned}\frac{d\hat{X}}{dt} &= (A - HC)\hat{X} + Hy + Bu \\ u_r &= -K\hat{X}\end{aligned}$$

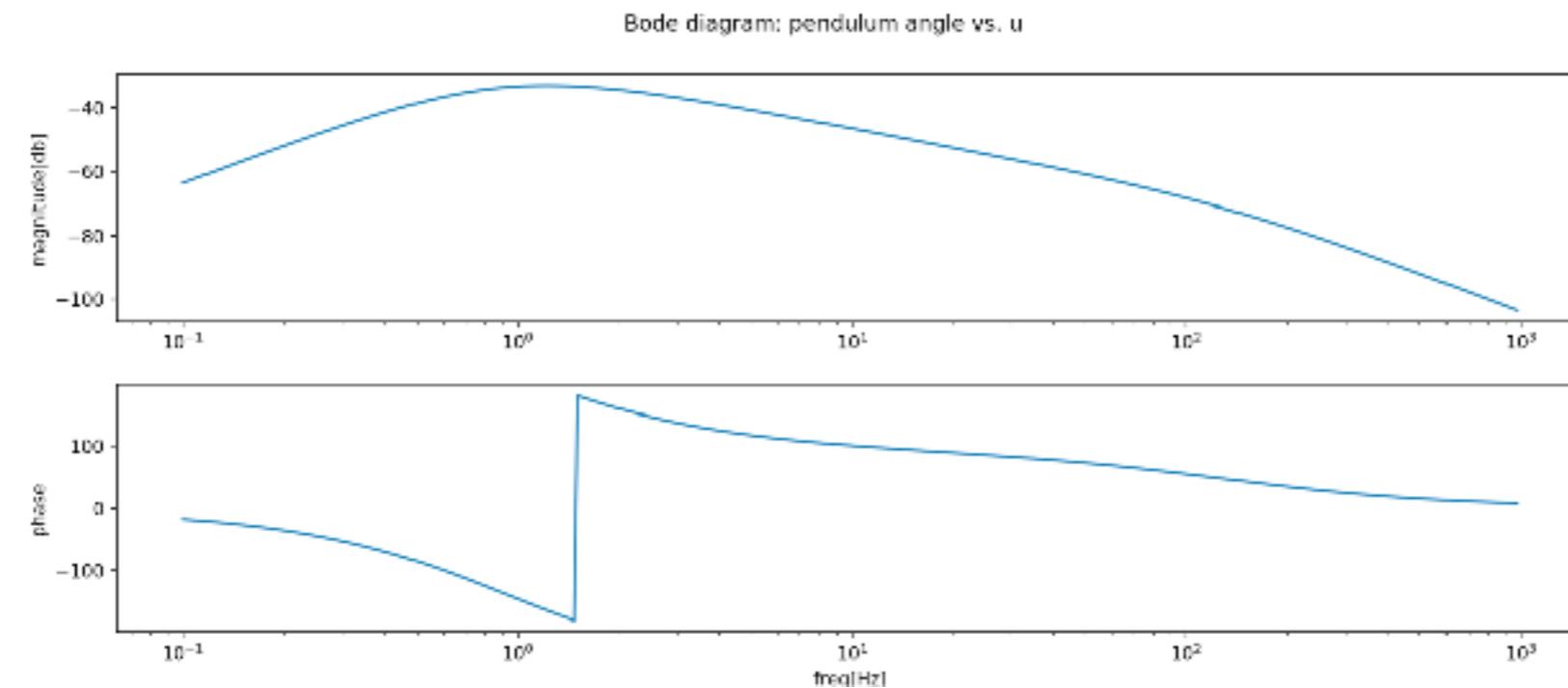
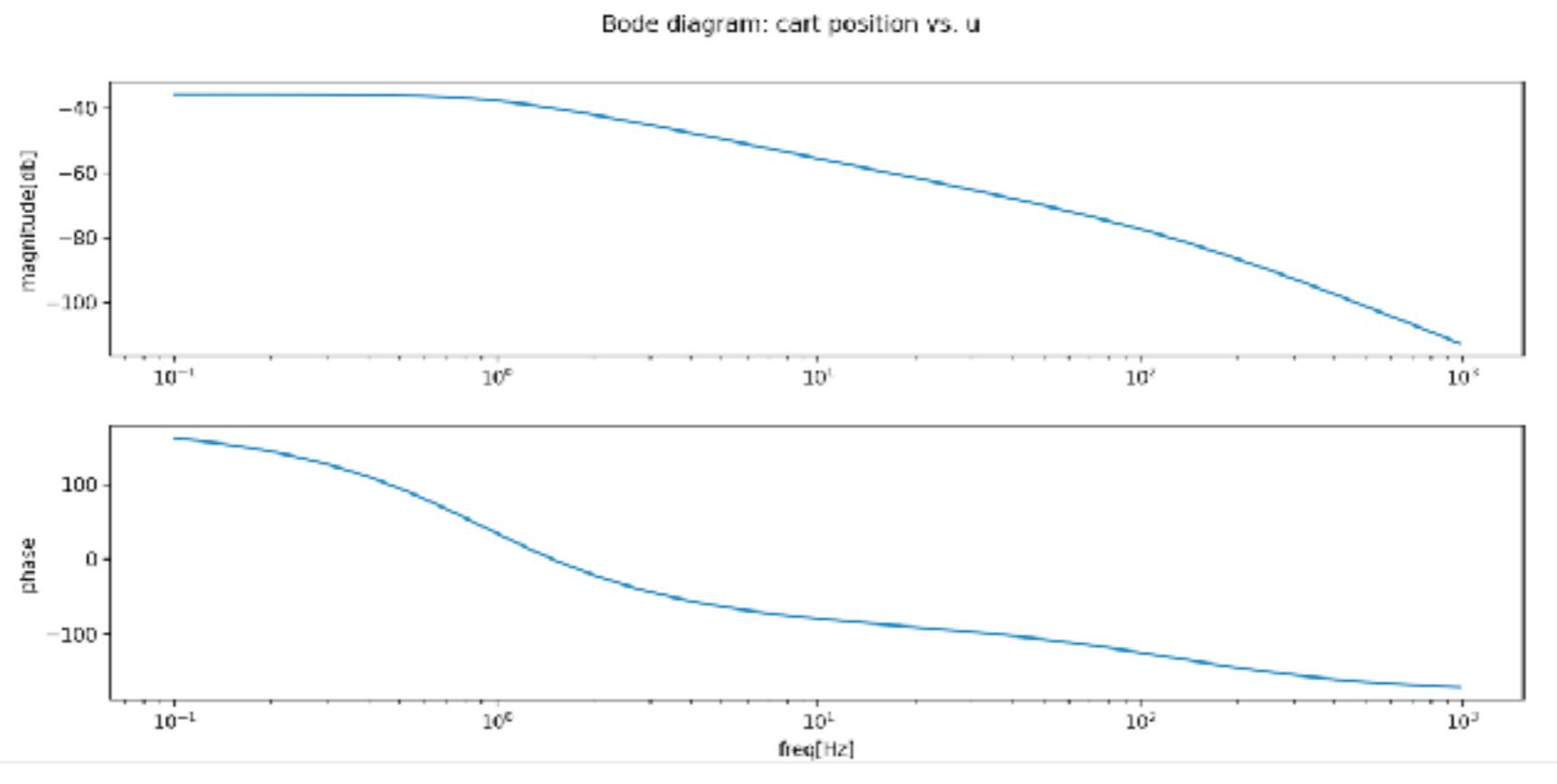
Controllerのアルゴリズムは、観測値 $y = \begin{pmatrix} \theta \\ z \end{pmatrix}$ と制御値 u から状態を推定し、次の制御値 u_r を計算する。実際の計算では、離散化したシステムをつかう

$$\begin{aligned}\hat{X}_{n+1} &= (A - HC)_d \hat{X}_n + H_d y_n + B_d u_n \\ u_{n+1} &= -K_d \hat{X}_{n+1}\end{aligned}$$

ここでは、python/controlモジュールのsample_system関数を使って離散化を行った。

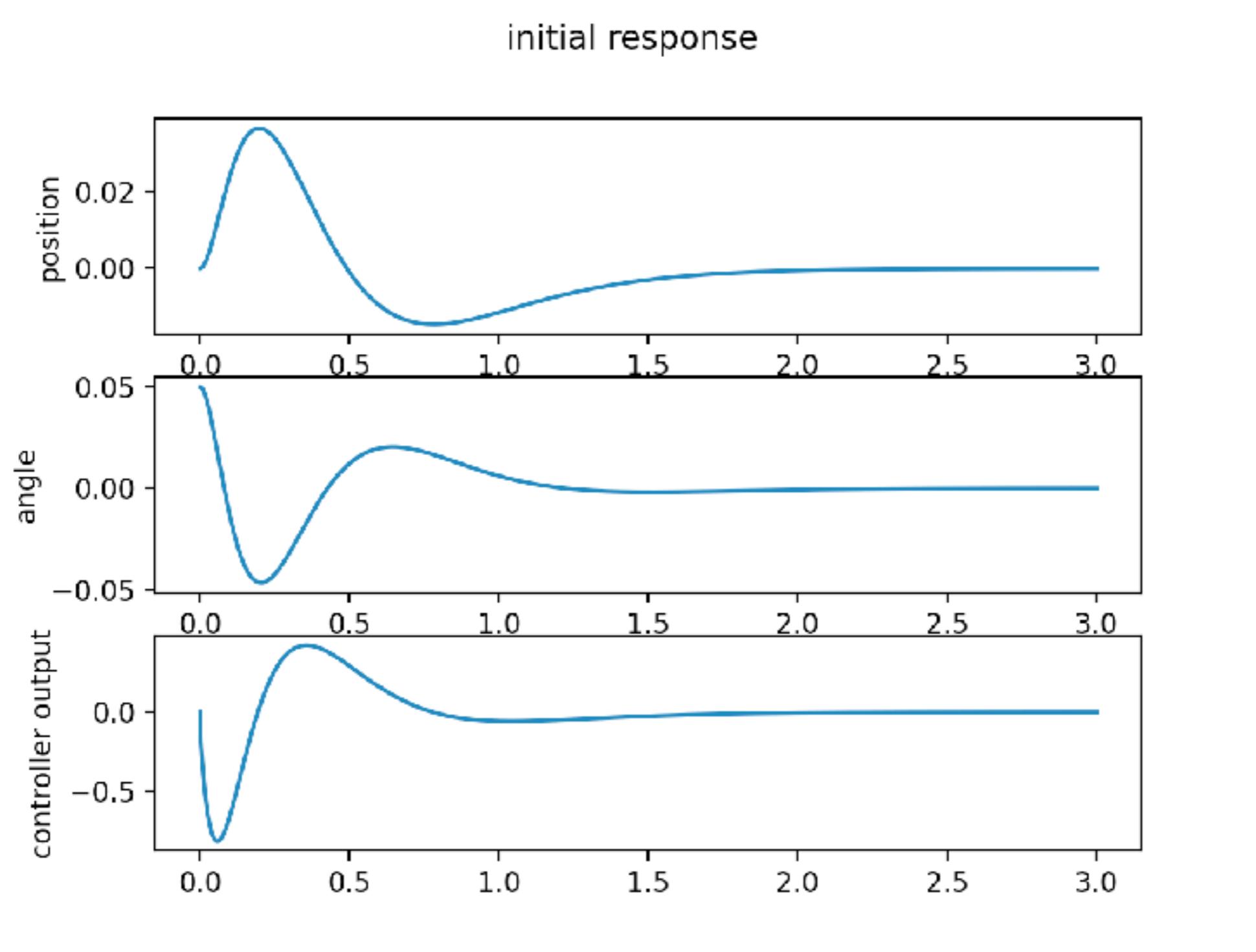
Bode線図

Python matplotlibを使って作成。



システム応答の一例

傾いた振り子を直立させる



まとめ

Feedbackレコード :

測定可能な θ, z, u の履歴を 状態推定値 \hat{X} として保持し、

それに基づいて、次の制御値を計算する。

パラメータは、モデルを仮定して決められている。

システムの調整は最適化の条件(Q, R, V, W)を変更しておこなう。-> 分かり易い。

制御の履歴を基に、これらのパラメータを動的に最適化していくシステムも
考えるられる。

あるいは、計算アルゴリズムを適當なニューラルネットで表現できれば、
そのニューラルネットで動的な学習を行うことも可能ではないだろうか。