

倒立振子の安定化制御

前田 拓

2017年7月12日

目 次

第 1 章 はじめに	3
1.1 目的	3
1.2 実験装置	3
第 2 章 モデリング	5
2.1 数式モデル	5
2.1.1 状態方程式	5
2.1.2 観測方程式	7
2.2 パラメータの同定	7
2.2.1 パラメータの検証	11
2.2.2 M と f の検証	11
2.2.3 J と c の検証	13
第 3 章 制御系設計	15
3.0.1 特性解析	15
3.0.2 制御システムの構成	16
3.0.3 状態フィードバック F の設計	17
3.0.4 $\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{J}$ の設計	17
3.0.5 制御システムの離散化	17
3.0.6 振り上げ制御と安定化	18
第 4 章 シミュレーション	21
4.0.1 重み行列変更に関するシミュレーション	21
4.0.2 オブザーバの極変更に関するシミュレーション	21
4.0.3 サンプリング周期変更に関するシミュレーション	22
4.0.4 振り上げ制御のシミュレーション	24
第 5 章 実験	25
5.0.1 実験装置	25
5.0.2 安定化制御の実験	25
5.0.3 重み行列	25
5.0.4 オブザーバの極	26
5.0.5 サンプリング周期	26
5.0.6 シミュレーションと実験結果の比較	26
5.0.7 目標値変更における安定化制御に関する考察	26
5.0.8 振り上げ制御	32
第 6 章 おわりに	33
6.0.1 まとめ	33

第7章 付録 プログラム	37
7.1 目標値変更における安定化制御	37
7.2 振り上げ制御	41

図 目 次

1.1 図 1.1: 倒立振子系	3
2.1 図 2.1: モデリングのための力の分解	5
2.2 図 2.2: パラメータ a の決定	8
2.3 図 2.3: 台車のステップ応答	9
2.4 図 2.4: 台車のステップ応答	10
2.5 図 2.5: ステップ応答による M と f の検証	12
2.6 図 2.6: ステップ応答による M と f の検証	12
2.7 図 2.7: 自由振動による J と c の検証	13
3.1 図 3.1: 制御システムの構成	16
4.1 図 4.1: 重み行列による比較(台車位置)	21
4.2 図 4.2: 重み行列による比較(振子角度)	21
4.3 図 4.3: オブザーバの極による比較(台車位置)	22
4.4 図 4.4: オブザーバの極による比較(振子角度)	22
4.5 図 4.5: 台車速度の推定誤差	22
4.6 図 4.6: 振子角速度の推定誤差	22
4.7 図 4.7: サンプリング周期による比較(台車位置)	23
4.8 図 4.8: サンプリング周期による比較(振子角度)	23
4.9 図 4.9: 台車速度の推定誤差	23
4.10 図 4.10: 振子角速度の推定誤差	23
4.11 図 4.11: 台車位置	24
4.12 図 4.12: 振子角度	24
5.1 図 5.1: 本実験で使用した倒立振子	25
5.2 図 5.2: 重み行列による比較(台車位置)	26
5.3 図 5.3: 重み行列による比較(振子角度)	26
5.4 図 5.4: オブザーバの極による比較(台車位置)	26
5.5 図 5.5: オブザーバの極による比較(振子角度)	26
5.6 図 5.6: サンプリング周期による比較(台車位置)	27
5.7 図 5.7: サンプリング周期による比較(振子角度)	27
5.8 図 5.8: パターン 01 の台車位置	27
5.9 図 5.9: パターン 01 の振子角度	27
5.10 図 5.10: パターン 02 の台車位置	28
5.11 図 5.11: パターン 02 の振子角度	28
5.12 図 5.12: パターン 03 の台車位置	28
5.13 図 5.13: パターン 03 の振子角度	28

5.14 図 5.14: パターン 04 の台車位置	28
5.15 図 5.15: パターン 04 の振子角度	28
5.16 図 5.16: パターン 05 の台車位置	29
5.17 図 5.17: パターン 05 の振子角度	29
5.18 図 5.18: パターン 06 の台車位置	29
5.19 図 5.19: パターン 06 の振子角度	29
5.20 図 5.20: パターン 07 の台車位置	29
5.21 図 5.21: パターン 07 の振子角度	29
5.22 図 5.22: パターン 08 の台車位置	30
5.23 図 5.23: パターン 08 の振子角度	30
5.24 図 5.24: パターン 09 の台車位置	30
5.25 図 5.25: パターン 09 の振子角度	30
5.26 図 5.26: パターン 10 の台車位置	30
5.27 図 5.27: パターン 10 の振子角度	30
5.28 図 5.28: パターン 11 の台車位置	31
5.29 図 5.29: パターン 11 の振子角度	31
5.30 図 5.30: パターン 12 の台車位置	31
5.31 図 5.31: パターン 12 の振子角度	31
5.32 図 5.32: 台車位置	32
5.33 図 5.33: 振子角度	32

表 目 次

2.1 表 2.1: 測定した物理パラメータ	11
3.1 表 3.1: A の固有値	15
4.1 表 4.1: 重み行列によるシミュレーションに用いるパラメータの種類	21
4.2 表 4.2: オブザーバの極によるシミュレーションに用いるパラメータの種類	22
4.3 表 4.3: サンプリング周期によるシミュレーションに用いるパラメータの種類	23
4.4 表 4.4: 振り上げ制御のシミュレーションに用いるパラメータの種類	24
4.5 表 4.5: 振り上げ後の安定化制御に用いるパラメータ	24
5.1 表 5.1: シミュレーションと実験の比較に用いるパラメータ	27

第1章 はじめに

1.1 目的

本実験の目的は、倒立振子系を状態空間表現を用いて安定化制御し、線形不変システムを設計することである。具体的に、次のことを目的とする。

- 倒立振子が安定化制御を行っている状態において、外乱による影響で振子が傾いたとき、倒立状態に戻すことができる（不安定平衡点の安定化）。
- 倒立振子系に一定周期のパルス入力を与え、台車を目的の変位へ移動させる。
- 倒立振子が入力なしで静止している状態から、台車を動かすことにより振子を振り上げ、倒立状態にする（振り上げ制御）。

1.2 実験装置

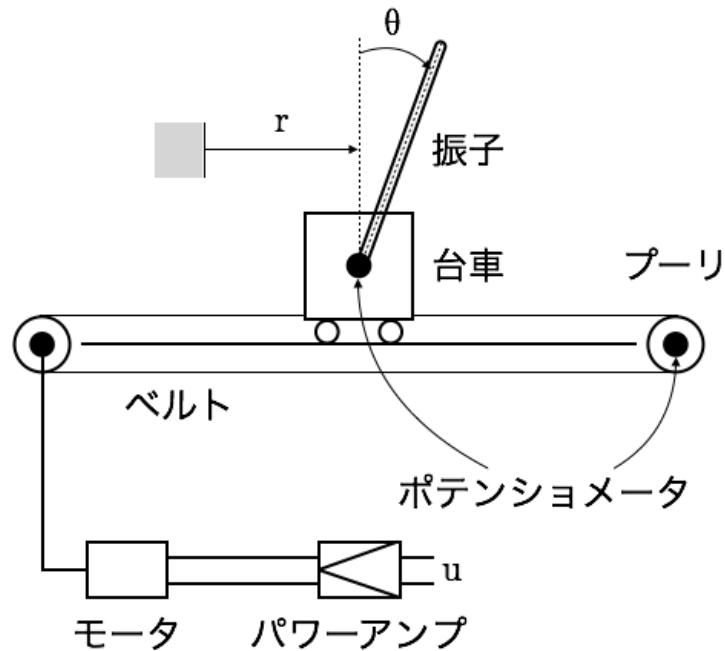


図 1.1: 倒立振子系

図 1.1 は本実験で使用する倒立振子系である。系は、モータ、ベルト、ブーリ系から成り、台車はモータからの入力によりベルト上を水平方向に動くことができる。台車の初期状態からの変位を r とする。また、鉛直方向上向きから時計回りを正の方向として、台車に取り付けられた振子が回転した角度を θ とする。ポテンショメータにより、 r と θ を測定し、入力 u を与える。

第2章 モデリング

2.1 数式モデル

制御器の設計のため、倒立振子系の状態方程式、観測方程式から数式モデルを導出する。

2.1.1 状態方程式

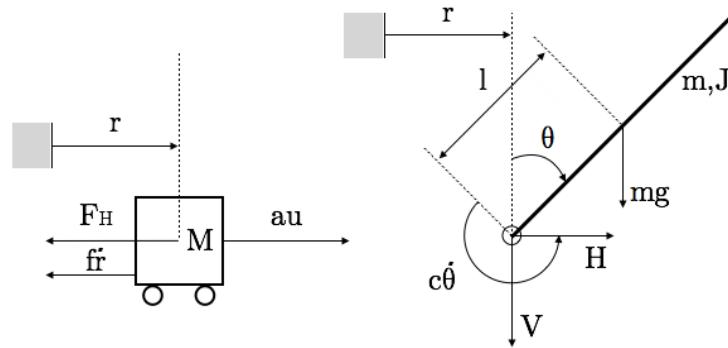


図 2.1: モデリングのための力の分解

図 2.1 から導出した倒立振子系の運動方程式を式 (2.1) から式 (2.4) に示す。

$$M\ddot{r} = au - F_H - f\dot{r} \quad (2.1)$$

$$J\ddot{\theta} = lF_V \sin \theta - lF_H \cos \theta - c\dot{\theta} \quad (2.2)$$

$$m \frac{d^2}{dt^2}(r + l \sin \theta) = F_H \quad (2.3)$$

$$m \frac{d^2}{dt^2}(l \cos \theta) = F_V - mg \quad (2.4)$$

ただし、 M, f は台車の質量と摩擦係数、 m, l, c, J は振子の質量、振子の重心から回転軸までの距離、回転軸摩擦係数、重心周りに働く慣性モーメントである。また、 F_H, F_V は振子が台車から受ける水平効力と垂直抗力である。 F はモータによる台車への駆動力であり、定数 a 、駆動アンプへの入力電圧 u を用いて式 (2.5) で表される。

$$F = au \quad (2.5)$$

ここで、系の状態 x を 4 つの状態変数からなる縦ベクトルとする。すなわち、

$$x = \begin{bmatrix} r \\ \theta \\ \dot{r} \\ \dot{\theta} \end{bmatrix}$$

と定義する. 次に, 式(2.1)から式(2.4)から倒立振子系の非線形方程式を求める. 式(2.1), 式(2.2)から F_H を消去すると, 式(2.6)が得られる.

$$(M+m)\ddot{r} + (ml \cos \theta)\ddot{\theta} = au + (ml \sin \theta)\dot{\theta}^2 - f\dot{r} \quad (2.6)$$

また, 式(2.2), 式(2.3), 式(2.4)から F_H, F_V を消去すると,

$$J\ddot{\theta} = l \left(m \frac{d^2}{dt^2} (l \cos \theta + mg) \right) \sin \theta - l \left(m \frac{d^2}{dt^2} (r + l \sin \theta) \right) \cos \theta$$

となり, 式(2.7)が得られる.

$$ml \cos \theta \ddot{r} + (J + ml^2)\ddot{\theta} = mgl \sin \theta - c\dot{\theta} \quad (2.7)$$

式(2.6), 式(2.7)を行列表現すると,

$$\begin{bmatrix} M+m & ml \cos \theta \\ ml \cos \theta & J + ml^2 \end{bmatrix} \begin{bmatrix} \ddot{r} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -f\dot{r} + ml(\sin \theta)\dot{\theta}^2 + au \\ mgl \sin \theta - c\dot{\theta} \end{bmatrix}$$

$$\begin{bmatrix} \ddot{r} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} M+m & ml \cos \theta \\ ml \cos \theta & J + ml^2 \end{bmatrix}^{-1} \begin{bmatrix} -f\dot{r} + ml(\sin \theta)\dot{\theta}^2 + au \\ mgl \sin \theta - c\dot{\theta} \end{bmatrix}$$

となる. よって,

$$\dot{x} = f(x, u) = \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ K^{-1} \begin{bmatrix} -f\dot{r} + ml(\sin \theta)\dot{\theta}^2 + au \\ mgl \sin \theta - c\dot{\theta} \end{bmatrix} \end{bmatrix}, \quad K = \begin{bmatrix} M+m & ml \cos \theta \\ ml \cos \theta & J + ml^2 \end{bmatrix} \quad (2.8)$$

が得られる. 本実験では, 不安定平衡点 $x = 0$ 近傍で線形化したモデルを採用できる. θ に関して式(2.8)を一次近似すると, $\sin \theta \approx \theta$, $\cos \theta \approx 1$, $\theta^2 \approx 0$ と近似できることから,

$$\dot{x} = \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ K^{-1} \begin{bmatrix} -f\dot{r} + au \\ mgl\theta - c\dot{\theta} \end{bmatrix} \end{bmatrix}, \quad K = \begin{bmatrix} M+m & ml \\ ml & J + ml^2 \end{bmatrix} \quad (2.9)$$

を得る. 線形化された倒立振子系の状態方程式は, 式(2.10)のように表現される.

$$\dot{x} = Ax + Bu \quad (2.10)$$

ここで,

$$A = \begin{bmatrix} O_{2 \times 2} & I_2 \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} O_{2 \times 1} \\ B_2 \end{bmatrix}$$

である. ただし,

$$A_{21} = K^{-1} \begin{bmatrix} 0 & 0 \\ 0 & mgl \end{bmatrix}, \quad A_{22} = K^{-1} \begin{bmatrix} -f & 0 \\ 0 & -c \end{bmatrix}, \quad B_2 = K^{-1} \begin{bmatrix} a \\ 0 \end{bmatrix}$$

である.

2.1.2 観測方程式

2つの観測出力は,

$$y_1 = c_1 r$$

$$y_2 = c_2 \theta$$

のように表される. ここで, c_1 は変位・電圧変換係数, c_2 は角度・電圧変換係数である. これらからなる縦ベクトル, すなわち出力 y を,

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

と定義すると, 倒立振子系に対する観測方程式を,

$$y = Cx \quad (2.11)$$

と表せる. ただし,

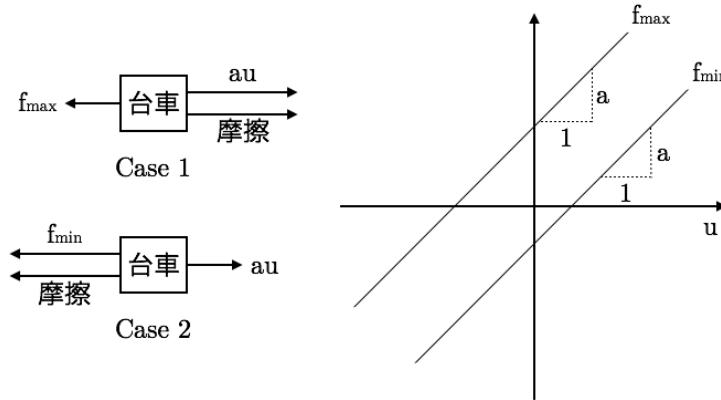
$$N = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}, \quad C = \begin{bmatrix} N & O_{2 \times 2} \end{bmatrix} = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \end{bmatrix}$$

である.

2.2 パラメータの同定

パラメータの同定のため, 以下の手順で測定を行う.

1. 実測できるパラメータ m (振子の質量 [kg]), l (振子の重心から回転軸までの長さ [m]) の測定を行う.
2. 駆動アンプへの入力電圧とモータ駆動力の変換係数 a を求める.
 - (a) 振子を台車から取り外す.
 - (b) モータに一定電圧 u を加え, ばねばかりで引き, 台車が正の方向に動き始めるときの力 ($au +$ 摩擦力) を f_{max} , 負の方向に動き始めるときの力 ($au -$ 摩擦力) を f_{min} とする. 測定は 10[V], 11[V], ..., 15[V] の各電圧に対して行う.
 - (c) f_{max} , f_{min} それぞれの回帰直線の傾きを平均し, その値を $a[N/V]$, すなわち入力電圧 $u[V]$ とモータ駆動力 $f[N]$ の変換係数とする.
3. 台車の変位 r と y_1 の変換係数を $c_1 = 1.0$ [V/m], 振子の角度 θ と y_2 の変換係数を $c_2 = 1.0$ [V/rad] とする.
4. 以下の 2通りの方法で台車系の質量 $M[\text{kg}]$ と摩擦係数 $f[\text{kg}/\text{s}]$ を求める.

図 2.2: パラメータ a の決定

- ステップ応答による方法

振子を台車から取り外したまま、台車のステップ応答を測定する。そのとき、台車の運動方程式は、

$$M\ddot{r} = au - fr \quad (2.12)$$

であり、入力 u から目標値 r までの伝達関数 G は、

$$G(s) = \frac{K}{s(Ts + 1)} \quad (2.13)$$

となる。ただし、

$$K = \frac{a}{f}, \quad T = \frac{M}{f} \quad (2.14)$$

である。初期状態を零ベクトルとすれば、台車のステップ応答は、

$$r(t) = KU_0 \left(T \exp \left(\frac{-t}{T} \right) + t - T \right) \quad (2.15)$$

である。(図 2.3 参照) ただし、 U_0 はステップ入力の $t > 0$ での値である。式 (2.15)において、 $t \rightarrow \infty$ の極限をとると、

$$\lim_{t \rightarrow \infty} r(t) = KU_0(t - T) \quad (2.16)$$

となる。図 2.3 を参考に、式 (2.16) から T, K を求め、

$10[V], 11[V], \dots, 15[V]$ の各電圧をステップ入力として与え、それぞれのステップ応答から求めたパラメータ T, K を平均し、 M, f を決定する。

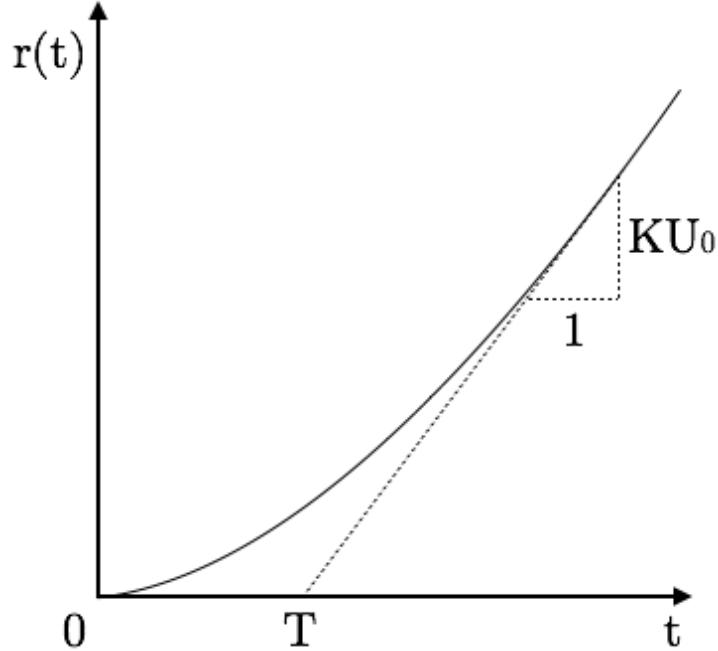


図 2.3: 台車のステップ応答

- フィードバックによる方法

台車の数式モデルである式 (2.12), 出力 $y_1 = c_1 r$ について, フィードバックにより入力を制御する. すなわち,

$$u = k(y_c - y_1) \quad (2.17)$$

とする ($y_c = \text{const}$, $k > 0$). このとき, 閉ループシステムの応答は,

$$M\ddot{y}_1 + fy_1 + c_1 aky_1 = c_1 aky_c \quad (2.18)$$

となる (図 2.4 参照). 一方, 偏差 z を加え, ばねばかりで引き, 台車が正の方向に動き始めるときの力

$$z = y_1 - y_c \quad (2.19)$$

により定義すると, z は以下の式 (2.20) に従う.

$$\ddot{z} + 2\zeta\omega_n\dot{z} + \omega_n^2 z = 0 \quad (2.20)$$

ただし,

$$\zeta = \frac{f}{2\sqrt{c_1 a k M}}, \quad \omega_n = \sqrt{\frac{c_1 a k}{M}} \quad (2.21)$$

である. 式 (2.20) の解は,

$$0 < \zeta < 1$$

のとき減衰振動となり, そのときの解は,

$$z(t) = \frac{z_0}{\sqrt{1 - \zeta^2}} \exp(-\omega_n \zeta t) \sin(\omega_n \sqrt{1 - \zeta^2} t + \phi)$$

ただし,

$$\phi = \tan^{-1} \frac{\sqrt{1 - \zeta^2}}{\zeta}$$

で与えられる. ここで, $z_0 = z(0) = -y_c$ である. いま, T とし, 時刻 t_1 と $t_2 = t_1 + T$ において波形 z の山が隣り合うものとする. このとき, 振幅の減衰比は,

$$\frac{|z_2(t_2)|}{z_2(t_1)} = \exp(-\lambda) \quad (2.22)$$

となる. λ は対数減衰比であり, 次が成り立つ.

$$\lambda = \frac{2\pi\zeta}{\sqrt{1 - \zeta^2}}, \quad T = \frac{2\pi}{\omega_n \sqrt{1 - \zeta^2}} \quad (2.23)$$

従って, 式 (2.21), 式 (2.23) から, パラメータ M, f は,

$$M = \frac{c_1 a k T^2}{4\pi^2 + \lambda^2}, \quad f = \frac{2\lambda M}{T} \quad (2.24)$$

のようにして求まる.

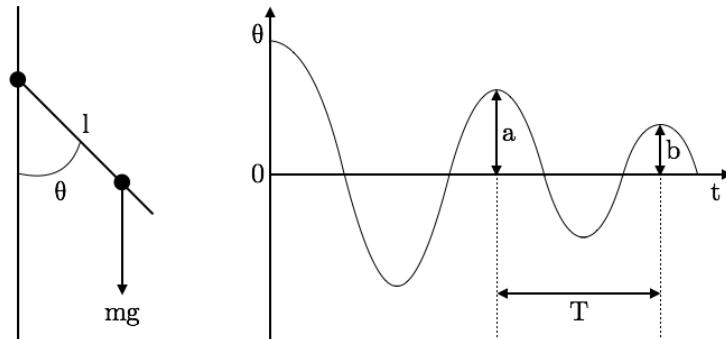


図 2.4: 台車のステップ応答

5. パラメータ J と c の決定

振子を台車に取り付け, 台車を固定したまま振子を自由振動させることにより, パラメータ J, c を測定する. このとき, 振子の運動方程式は,

$$(J + ml^2)\ddot{\theta} = -mgl \sin \theta - c\dot{\theta} \quad (2.25)$$

$$y_2 = c_2 \theta \quad (2.26)$$

で与えられる. θ を微小区間で考えると, 式 (2.25), 式 (2.26) は次のように書ける.

$$\ddot{y}_2 + 2\zeta\omega_n\dot{y}_2 + \omega_n^2 y_2 = 0 \quad (2.27)$$

$$\zeta = \frac{c}{2\sqrt{mgl(J+ml^2)}}, \omega_n = \sqrt{\frac{mgl}{J+ml^2}} \quad (2.28)$$

式(2.27)の解で表される減衰振動の対数減衰率を λ , 周期を T とすると, 式(2.23)が成り立つことから J と c は,

$$J = \frac{mglT^2}{4\pi^2 + \lambda^2} - ml^2, c = \frac{2\lambda(J+ml^2)}{T} \quad (2.29)$$

のように与えられる.

2.2.1 パラメータの検証

測定した物理パラメータを表2.1に示す.

表2.1: 測定した物理パラメータ

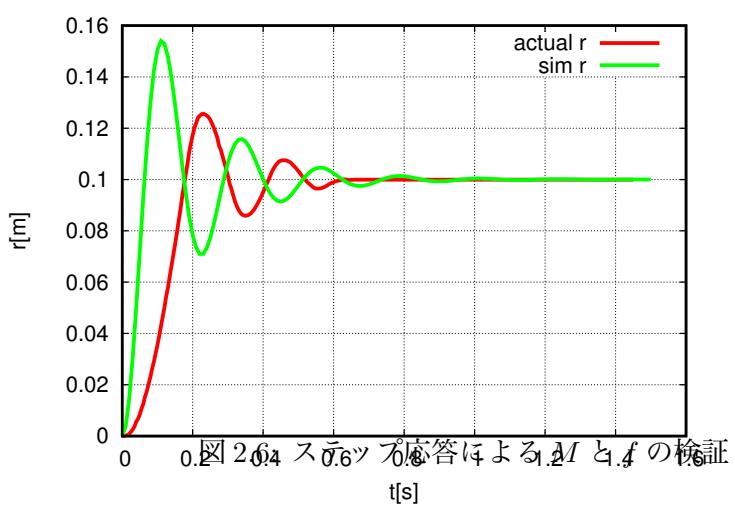
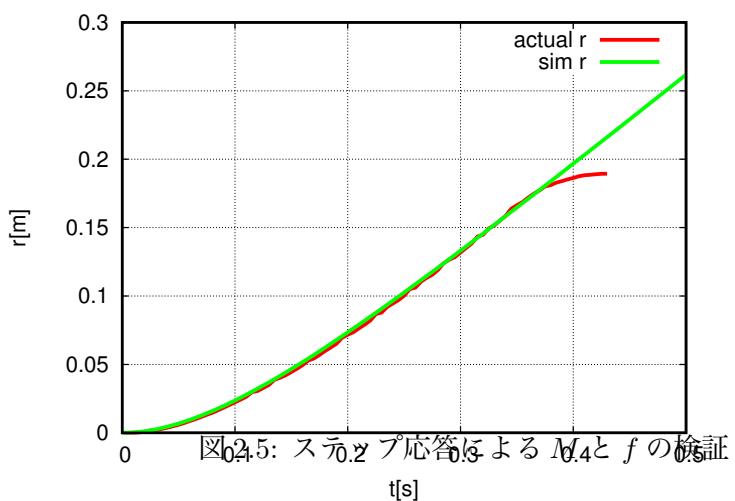
m [kg]	0.038
l [m]	0.12
M (Step) [kg]	1.00
f (Step) [kg/s]	9.67
M (Feedback) [kg]	1.51
f (Feedback) [kg/s]	16.5
J [kgm ²]	3.90E-4
c [kgm ² /s]	9.82E-5
a [N/V]	0.49
c_1 [V/m]	1.0
c_2 [V/rad]	1.0
g [m/s ²]	9.8

次に, 表2.1に示すパラメータの検証を行う.

2.2.2 M と f の検証

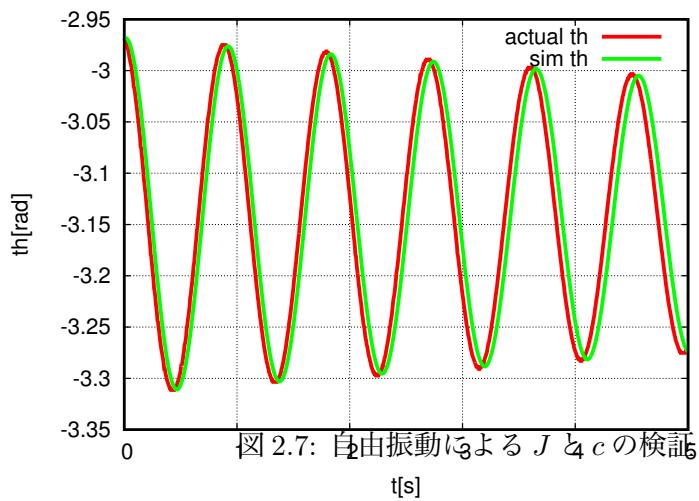
図2.5に, 入力電圧13.0[V]のときの台車系のステップ応答とそのシミュレーションを示す. また, 図2.6に台車系のフィードバック応答とそのシミュレーションを示す.

図2.6では, シミュレーション結果と実験結果が不一致なため, 不適切なパラメータである. 一方, 図2.5では, 直線近似を行うために十分な一致が見られるため, ステップ応答により測定した $M = 1.00$ と $f = 9.67$ を用いて実験を行う.



2.2.3 J と c の検証

図 2.7 に振子の自由振動の応答とそのシミュレーションを示す。実験とシミュレーションの波形を比較すると、初期位相が異なっているが、振幅と周期が近い値をとっているので測定した $J = 3.90\text{E-}4$ と $c = 9.82\text{E-}5$ を実験に用いる。



第3章 制御系設計

3.0.1 特性解析

第2章で物理パラメータの決定と数式モデルの導出を行った。ここでは、決定したパラメータから、式(2.9)、式(2.10)の特性解析を行う。倒立振子の線形モデルの状態空間表現は以下のようになる。

$$A, B, C \text{ の値} \quad (3.1)$$

まず、行列 A の安定判別を行う。 A のすべての固有値に関して、実部が負であれば A は安定性行列である。すなわち、 A の固有値 $\lambda_1, \lambda_2, \dots$ に対し、

$$\operatorname{Re}[\lambda_i] < 0$$

が成立すればよい。MATLAB で A の固有値を算出した結果を表3.1に示す。

表 3.1: A の固有値

固有値	$\operatorname{Re}[\lambda_i]$
$0 + 0i$	0
$-8.89 \times 10^{-2} + 6.93i$	-8.89×10^{-2}
$-8.89 \times 10^{-2} - 6.93i$	-8.89×10^{-2}
$-9.45 + 0i$	-9.45

表3.1から、倒立振子系のシステムは不安定である。次に、システムの可制御性、可観測性を判別する。可制御性は、 n をシステムの次数（倒立振子系のシステムの次数は $n = 4$ ）に対し、可制御性行列、

$$U_C = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}$$

のランクがシステムの次数と等しければ満たされる。また、可観測性行列は、

$$U_o = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

であり、 U_o のランクがシステムの次数と等しければ可観測となる。式(3.1)において、MATLAB を用いて U_c, U_o のランクを計算した結果、以下のようになった。

$$\operatorname{rank}[U_c] = 4, \operatorname{rank}[U_o] = 4$$

以上から、倒立振子系のシステムは不安定であり、可制御かつ可観測である。

3.0.2 制御システムの構成

図3.1に、倒立振子系に対する制御システムの構成を示す。

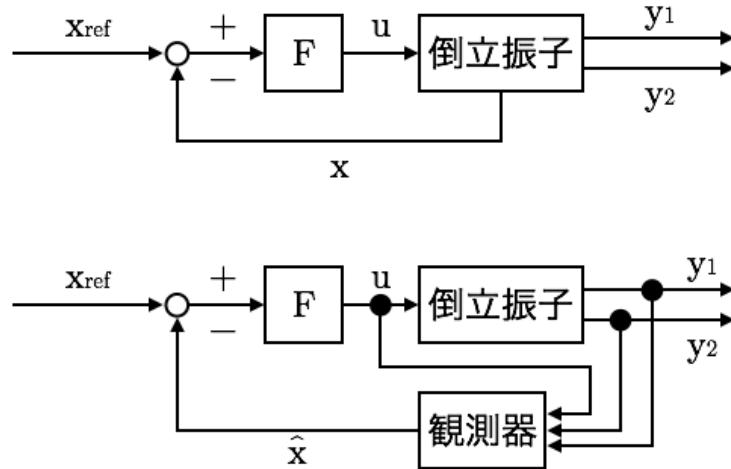


図3.1: 制御システムの構成

制御系のフィードバックによる入力は、

$$u = -F(x - x_{ref})$$

ただし、

$$x_{ref} = \begin{bmatrix} y_c \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

であり、\$y_c\$ は指定された台車位置を表す。本実験では、\$\dot{r}, \dot{\theta}\$ の検出器は用いず、状態 \$x\$ は推定値 \$\hat{x}\$ において、

$$x = \lim_{t \rightarrow \infty} \hat{x} \quad (3.2)$$

とする。また、式(3.2)を満足する最小次元オブザーバ、

$$\hat{z} = \hat{A}z + \hat{B}y + \hat{J}u \quad (3.3)$$

$$\hat{x} = \hat{C}z + \hat{D}y \quad (3.4)$$

を用いる。ここで、オブザーバの次数は2であり、設計すべき制御システムは以下のようになる。

- \$F(1 \times 4)\$
- \$\hat{A}(2 \times 2), \hat{B}(2 \times 2), \hat{J}(2 \times 1), \hat{C}(4 \times 2), \hat{D}(4 \times 2)\$

3.0.3 状態フィードバック F の設計

F は, システムを安定化する状態フィードバック,

$$u = -Fx \quad (3.5)$$

を満たすように求める. 式 (3.5) を LQ 問題として解くため, 2 次形式評価関数,

$$J = \int_0^\infty (x^T Q x + R u^2) dt \quad (3.6)$$

$$Q = \text{diag}(q_1^2, q_2^2, q_3^2, q_4^2), R = 1 \quad (3.7)$$

を考える. ここで, $\text{diag}(\dots)$ は, 各引数を固有値にもつ対角行列を表す. これは,

$$J = \int_0^\infty \left(q_1^2 r^2 + q_2^2 \theta^2 + q_3^2 \dot{r}^2 + q_4^2 \dot{\theta}^2 + u^2 \right) dt \quad (3.8)$$

と表せるから, q_1, q_2, q_3, q_4 はそれぞれ, 台車位置 r , 振子角度 θ , 台車速度 \dot{r} , 振子角速度 $\dot{\theta}$ に対する重み係数である. 式 (3.6), 式 (3.7) を最小にする F は, リッカチ方程式,

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

の解 ($P > 0$) に対し,

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

で与えられる.

3.0.4 $\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{J}$ の設計

式 (3.3), 式 (3.4) が式 (3.2) を満足するための十分条件は, ある行列 $U(1 \times 4)$ が存在して,

$$\begin{aligned} UA &= \hat{A}U + \hat{B}C \\ UB &= \hat{J} \\ I &= \hat{C}U + \hat{D}C \end{aligned}$$

であり, かつ \hat{A} が安定性行列であることである. これらの条件を満足するオブザーバを設計するため, Gopinath の方法を用いる.

3.0.5 制御システムの離散化

式 (3.5), 式 (3.3), 式 (3.4) で示した u, \hat{z}, \hat{x} は, 連続時間上で設計される制御器である. そこで, 計算機で制御器を表現するためこれらの離散時間化を行う. 各サンプリングの間隔を Δ とすると,

$$\begin{aligned} u[k] &= -F\hat{x}[k] \\ z[k+1] &= \hat{A}_d z[k] + \hat{B}_d y[k] + \hat{J}_d u[k] \\ \hat{x}[k] &= \hat{C}_d z[k] + \hat{D}_d y[k] \end{aligned}$$

と表せる. ただし, $k = 0, 1, \dots$ において,

$$x_{ref} = \begin{bmatrix} \hat{A}_d & \begin{bmatrix} \hat{B}_d & \hat{J}_d \end{bmatrix} \\ 0 & I_3 \end{bmatrix} = \exp \Delta \begin{bmatrix} \hat{A} & \begin{bmatrix} \hat{B} & \hat{J} \end{bmatrix} \\ 0 & I_3 \end{bmatrix}$$

である。

3.0.6 振り上げ制御と安定化

台車と振子の運動方程式は、式(2.6)、式(2.7)から、

$$\begin{aligned} (M+m)\ddot{r} + (ml \cos \theta)\ddot{\theta} &= au + (ml \sin \theta)\dot{\theta}^2 - f\dot{r} \\ ml \cos \theta \ddot{r} + (J+ml^2)\ddot{\theta} &= mgl \sin \theta - c\dot{\theta} \end{aligned}$$

で与えられる。振子が鉛直上向きのときを基準とする振子の力学的エネルギーは、

$$E = \frac{1}{2} (J + ml^2) \dot{\theta}^2 + mgl (\cos \theta - 1) \quad (3.9)$$

であり、力学的エネルギーの時間微分は、

$$\frac{dE}{dt} = (J + ml) \theta \ddot{\theta} - mgl \dot{\theta} \sin \theta \quad (3.10)$$

となる。振り上げ処理を行うため、入力 u を次のように計算する。

$$u = \frac{1}{a} \left(f\dot{r} - ml \sin \theta \dot{\theta}^2 + ml \cos \theta \ddot{\theta} + (M+m)v \right) \quad (3.11)$$

$$v = -\frac{c\dot{\theta}}{ml \cos \theta} + k(E - E_0) \text{sign}(\dot{\theta} \cos \theta) \quad (3.12)$$

sign は符号関数であり、引数が 0 のとき 0 を、正のとき 1 を、負のとき -1 を返す。式(2.6)に式(3.11)を代入すると、

$$\ddot{r} = v \quad (3.13)$$

を得る。式(3.13)、式(3.12)を式(2.7)に代入すると、

$$(J + ml^2)\ddot{\theta} = mgl \sin \theta - ml \cos \theta (k(E - E_0) \text{sign}(\dot{\theta} \cos \theta)) \quad (3.14)$$

を得る。さらに、これを式(3.10)に代入すると、

$$\begin{aligned} \frac{dE}{dt} &= -ml \dot{\theta} \cos \theta (k(E - E_0) \text{sign}(\dot{\theta} \cos \theta)) \\ &= -mlk(E - E_0) \text{sign}(\dot{\theta} \cos \theta) \dot{\theta} (\cos \theta) \end{aligned}$$

となる。リアブノフ関数として、

$$V = \frac{(E - E_0)^2}{2} \quad (3.15)$$

を考えると、 V の時間微分、

$$\begin{aligned}\frac{dV}{dt} &= (E - E_0) \frac{dE}{dt} \\ &= -mlk(E - E_0)^2 \text{sign}(\dot{\theta} \cos \theta) \dot{\theta} (\cos \theta) \leq 0\end{aligned}\tag{3.16}$$

これより, $\dot{\theta} \cos \theta \neq 0$ のとき, V は減少して 0 に収束し, E は E_0 に収束する. 実際の制御では, 台車の加速度目標 v を制限し,

$$u = \frac{1}{a} \left(f\dot{r} - ml \sin \theta \dot{\theta}^2 + ml \cos \theta \ddot{\theta} + (M + m)v \right)$$

$$v = -\frac{c\dot{\theta}}{ml \cos \theta} + \text{sat}_{ng}(k(E - E_0)\text{sign}(\dot{\theta} \cos \theta))$$

とする. ただし, sat_{ng} は最小値 $-ng$, 最大値 ng の飽和関数である. また, n は重力加速度と台車の加速度の比である. さらに, k を大きくすればより早く E が E_0 に収束する.

第4章 シミュレーション

目標値変更、振り上げ制御のシミュレーションを行い、重み行列、オブザーバの極、サンプリング周期、パラメータ k, n を変更し、それぞれの変更による性能の違いを確かめる。

4.0.1 重み行列変更に関するシミュレーション

目標値変更における安定化制御で、表 4.1 に示すパラメータを用いてシミュレーションを行う。

表 4.1: 重み行列によるシミュレーションに用いるパラメータの種類

パターン	重み行列 Q	オブザーバの極 P	サンプリング周期 $dt[s]$
パターン 1	$Q_1(1E6, 1E5, 1, 1)$	$P_1((-23,0), (-23,0))$	$dt_1:0.005$
パターン 2	$Q_2(1E5, 1E6, 1, 1)$	$P_2((-23,0), (-23,0))$	$dt_2:0.005$
パターン 3	$Q_3(1E6, 1E6, 1, 1)$	$P_3((-23,0), (-23,0))$	$dt_3:0.005$

表 4.1 に従ってシミュレーションを行った結果を図 4.1、図 4.2 に示す。

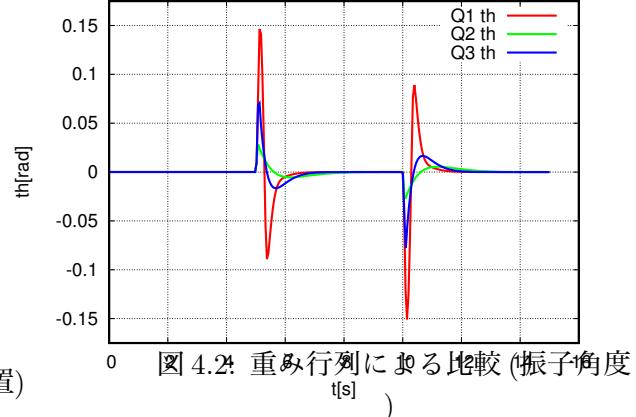
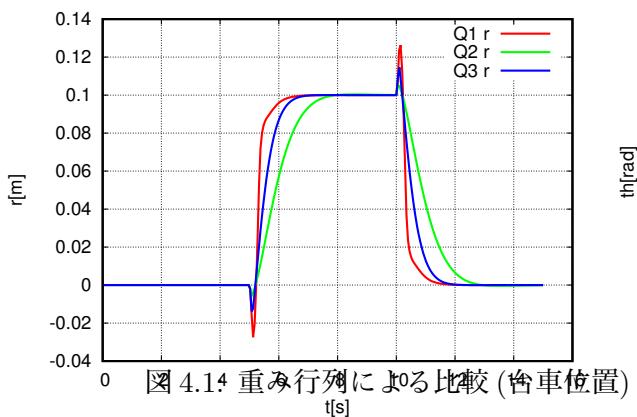


図 4.1 から、最も重みを大きくしたパターン 1 をみると、たしかに台車の目標位置への収束は速くなっているが、台車の位置を有線する一方で振子の角度の振幅は大きくなっている。一方で、振子角度の重みを最も大きくしたパターン 2 では、振子角度の振幅は小さく抑えられていると同時に、台車位置の目標値への収束が 3 パターンの中で最も遅くなっている。

4.0.2 オブザーバの極変更に関するシミュレーション

目標値変更における安定化制御で、表 4.2 に示すパラメータを用いてシミュレーションを行う。

表 4.2: オブザーバの極によるシミュレーションに用いるパラメータの種類

パターン	重み行列 Q	オブザーバの極 P	サンプリング周期 $dt[s]$
パターン 1	$Q_1(1E6, 1E5, 1, 1)$	$P_1((-23,0), (-23,0))$	$dt_1:0.005$
パターン 2	$Q_2(1E6, 1E5, 1, 1)$	$P_2((-50,0), (-50,0))$	$dt_2:0.005$

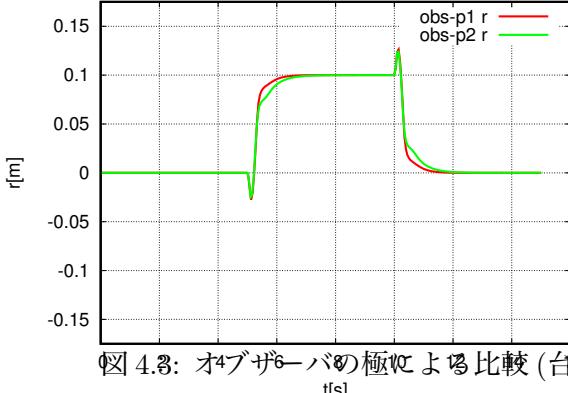


図 4.3: オブザーバの極による比較(台車位置)

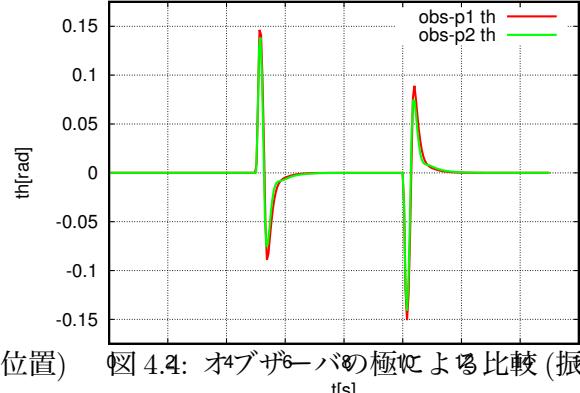


図 4.4: オブザーバの極による比較(振子角度)

表 4.2 に従ってシミュレーションを行った結果を図 4.3, 図 4.4 に示す。

台車位置, 振子角度だけでは変化が表れにくいので, それぞれのオブザーバの推定誤差を用いて比較を行う。図 4.5 に台車速度に関する推定誤差を, 図 4.6 に振子角速度に関する推定誤差を示す。

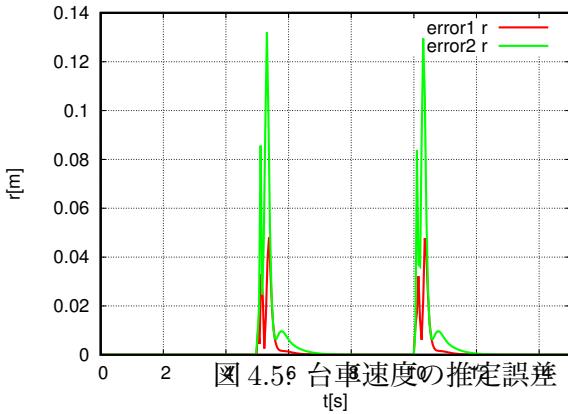


図 4.5: 台車速度の推定誤差

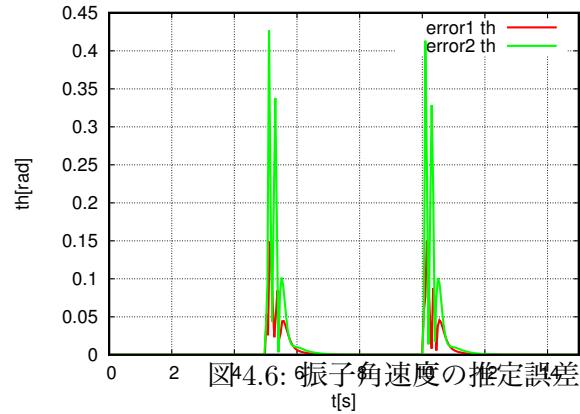


図 4.6: 振子角速度の推定誤差

台車速度, 振子角度の推定誤差とともに, オブザーバの極が負の方向に虚軸から遠いほど推定誤差が大きくなっている。よって, このシミュレーション結果からはオブザーバの極は虚軸に近い極配置が好ましいと言える。実際には, 極の絶対値がより大きいほど推定誤差は小さくなるが, 本実験で行ったシミュレーションのパラメータでは既に極の絶対値が十分に大きく, オブザーバの極配置以外の要因により推定誤差が大きくなつたと考察できる。

4.0.3 サンプリング周期変更に関するシミュレーション

目標値変更における安定化制御で, 表 4.3 に示すパラメータを用いてシミュレーションを行う。

表 4.3: サンプリング周期によるシミュレーションに用いるパラメータの種類

パターン	重み行列 Q	オブザーバの極 P	サンプリング周期 $dt[s]$
パターン 1	$Q_1(1E6, 1E5, 1, 1)$	$P_1((-23,0), (-23,0))$	$dt_1:0.005$
パターン 2	$Q_2(1E6, 1E5, 1, 1)$	$P_2((-23,0), (-23,0))$	$dt_2:0.01$

表 4.3 に従ってシミュレーションを行った結果を図 4.7, 図 4.8 に示す.

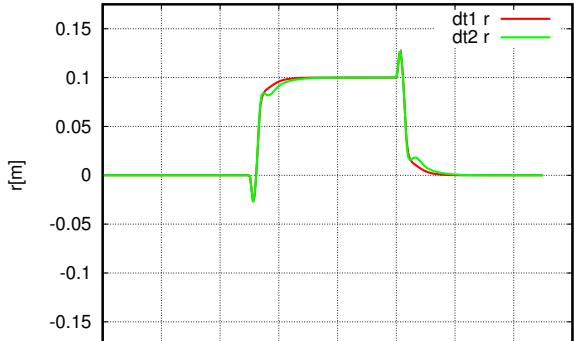


図 4.7: サンプリング周期による比較 (台車位置)
t[s]

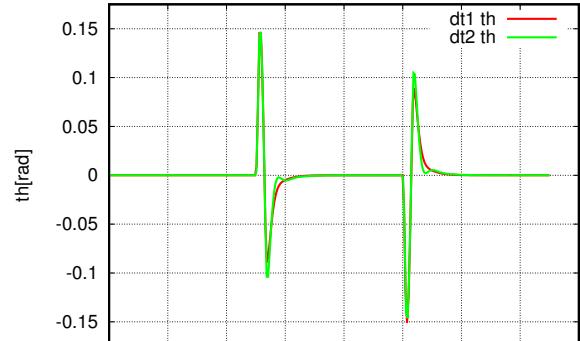


図 4.8: サンプリング周期による比較 (振子角度)
t[s]

オブザーバの極によるシミュレーション同様, 台車位置, 振子角度だけでは変化が表れにくいので, それぞれのオブザーバの推定誤差を用いて比較を行う. 図 4.9 に台車速度に関する推定誤差を, 図 4.10 に振子角速度に関する推定誤差を示す.

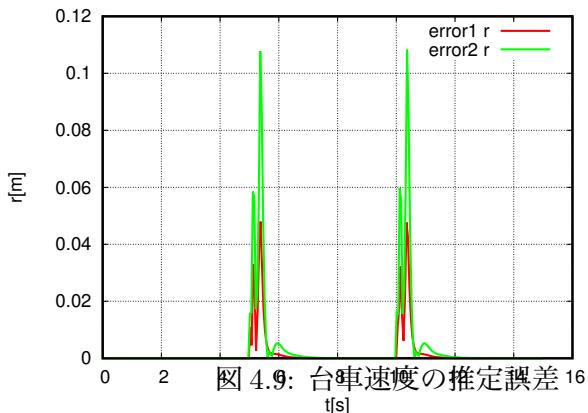


図 4.9: 台車速度の推定誤差
t[s]

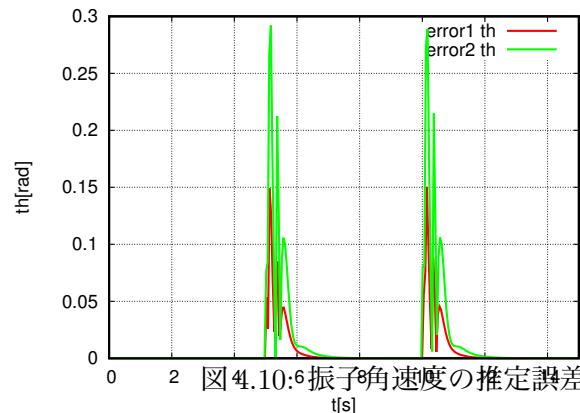


図 4.10: 振子角速度の推定誤差
t[s]

図 4.9, 図 4.10 から, サンプリング周期が短いほど推定誤差が小さくなっている. サンプリング周期を大きくすると, 計測するデータの間隔が大きくなってしまうため, 実際の値への追従が遅れる. よって, このシミュレーションでは意図した結果が得られたと言える.

4.0.4 振り上げ制御のシミュレーション

振り上げシミュレーションに用いたパラメータを表4.4に示す。ただし、振子が安定化制御に移行した際に用いるパラメータは表4.5のパラメータを用いることとする。また、台車位置、振子角度の制限として、台車位置はベルト中心から両方向に0.09[m]、振子角度は $[-\pi, \pi]$ [rad]の範囲の値を取るものとする。

表4.4: 振り上げ制御のシミュレーションに用いるパラメータの種類

パターン	k	n
パターン 1	1.0E3	0.31
パターン 2	1.0E4	0.31
パターン 3	1.0E5	0.31

表4.5: 振り上げ後の安定化制御に用いるパラメータ

重み行列 Q	オブザーバの極 P	サンプリング周期 dt [s]
$Q(1E6, 1E5, 1, 1)$	$P((-23,0), (-23,0))$	$dt:0.005$

表4.4を用いて振り上げ制御のシミュレーションを行った結果を図4.11, 図4.12に示す。

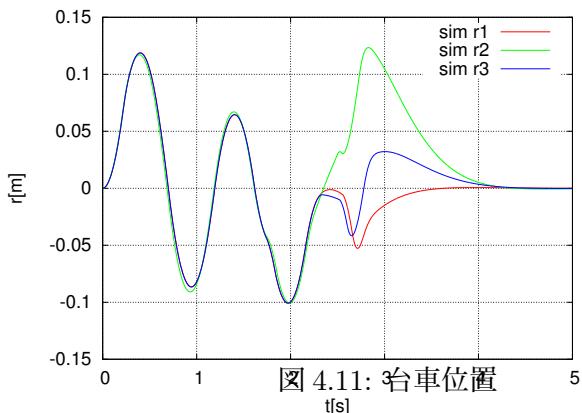


図 4.11: 台車位置

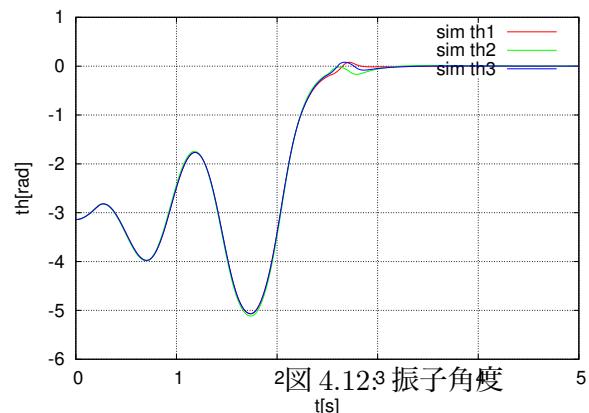


図 4.12: 振子角度

パラメータ k の値を大きくすることで、目標とする振子の運動エネルギーへの収束が速くなることが予想されたが、本実験のシミュレーションでは k の値に関わらず安定化制御に移行するまでの時間に変化はなかった。ただし、安定化制御に移行する際の台車位置はそれぞれ異なっていたため、パラメータ k の値によっては台車が制限された移動範囲外にはみ出してしまった可能性がある。

第5章 実験

5.0.1 実験装置

本実験で用いる実験装置を図 5.1 に示す。この実験装置は第 2 章で示した実験装置と同様の機能を有する。

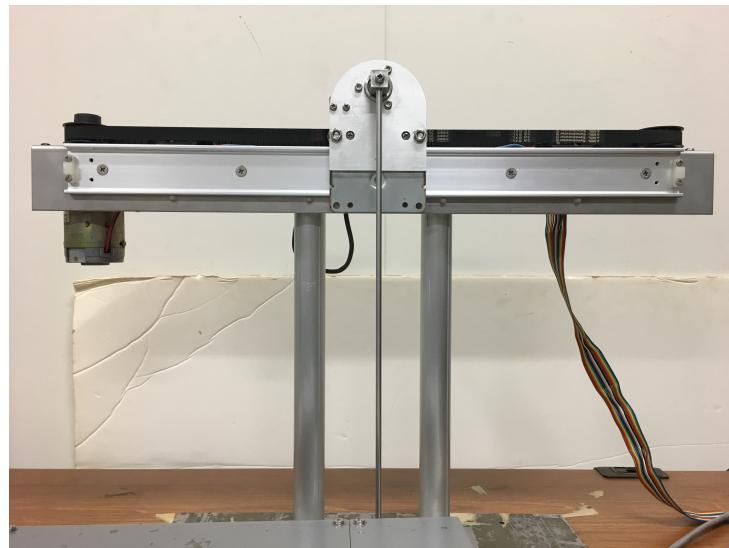


図 5.1: 本実験で使用した倒立振子

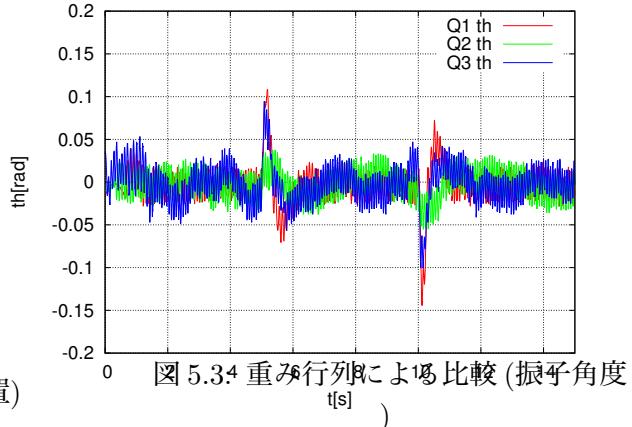
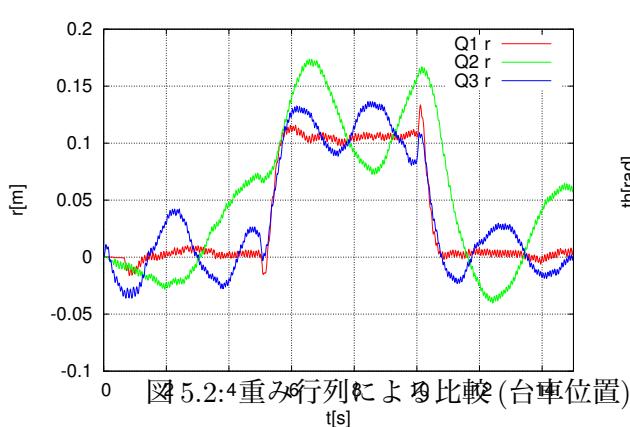
5.0.2 安定化制御の実験

表 4.1 から表 4.3 までのパラメータを用いて、目標値変更における安定化制御を行う。振子が鉛直上向きの状態から安定化制御を開始し、5 秒ごとに台車の目標値を $0.0 \rightarrow 0.5 \rightarrow 0.0[m]$ と交互に切り替える。

5.0.3 重み行列

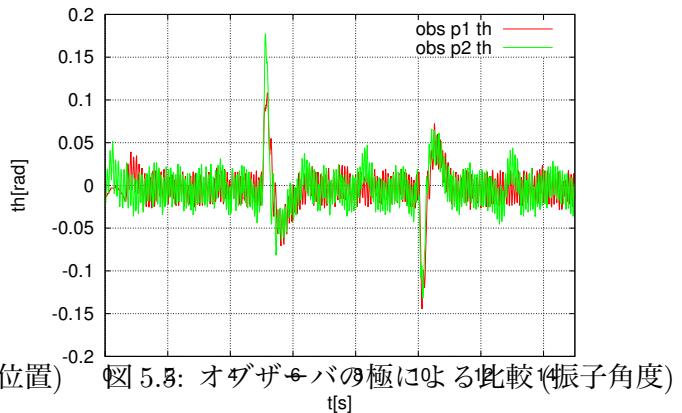
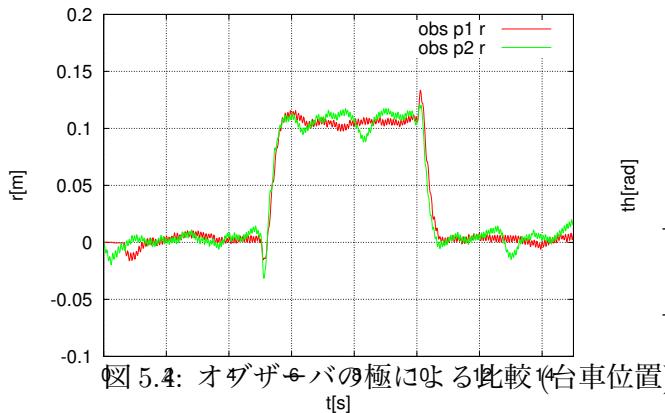
重み行列の変化による実験結果を図 5.2、図 5.3 に示す。

台車位置の重みを最も大きくしたパターン 1 の波形に着目すると、3 パターンの中で最も速く台車位置が目標値に収束している。一方、振子角度の重みを最も大きくしたパターン 2 の台車位置は、目標値に収束する前に目標値が変更され、常に振動的な応答となっている。振子角度についてもシミュレーションの場合と同様のことが言える。



5.0.4 オブザーバの極

オブザーバの極の変化に関する実験結果を図 5.4, 図 5.5 に示す。



シミュレーションの場合と同様, 台車位置と振子角度に大きな変化は見られなかった。

5.0.5 サンプリング周期

サンプリング周期の変化に関する実験結果を図 5.6, 図 5.7 に示す。

シミュレーションの場合と同様, 台車位置と振子角度に大きな変化は見られなかった。

5.0.6 シミュレーションと実験結果の比較

表 5.1 をもとに, シミュレーションと実験結果を比較した図を, 図 5.8 から図 5.31 に示す。

5.0.7 目標値変更における安定化制御に関する考察

- 重み行列

振子角度よりも台車位置に重みを置いたパターン 01 からパターン 03 と, 振子角度と台車位

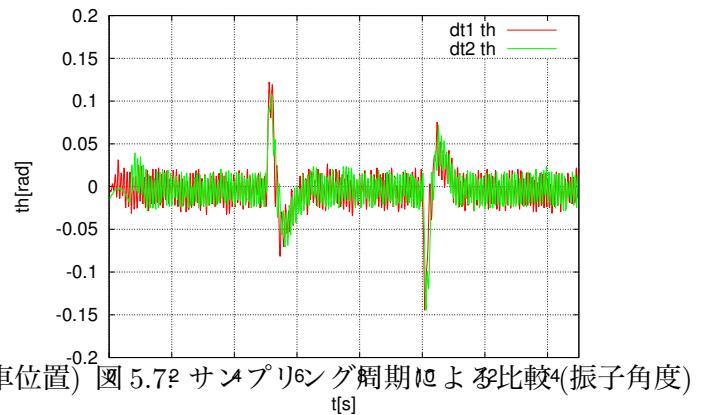
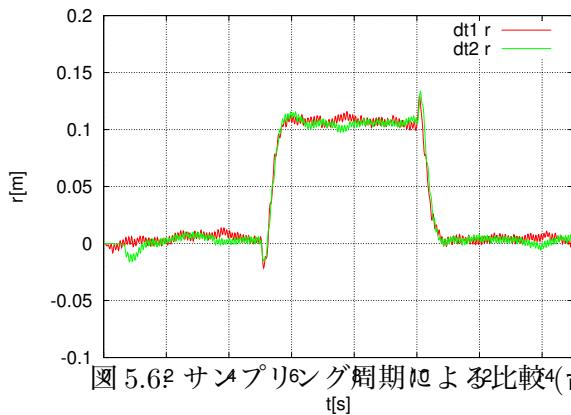
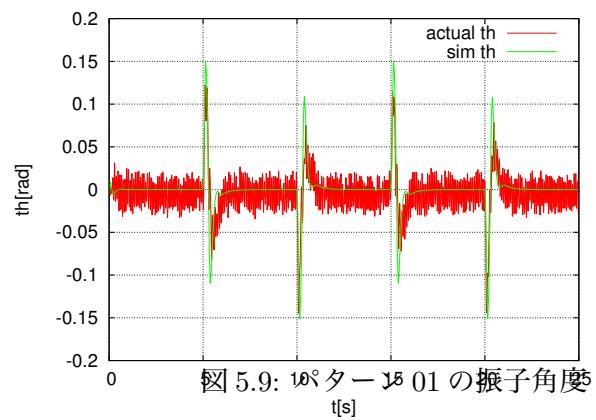
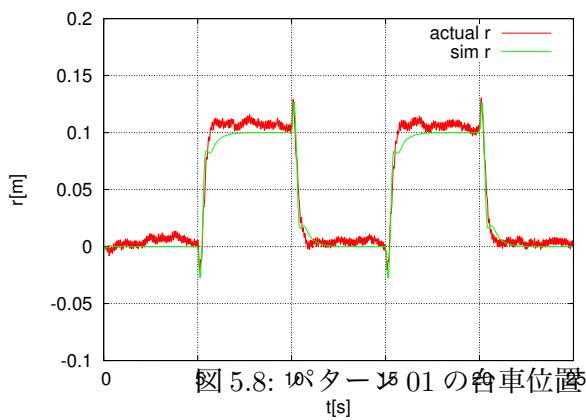
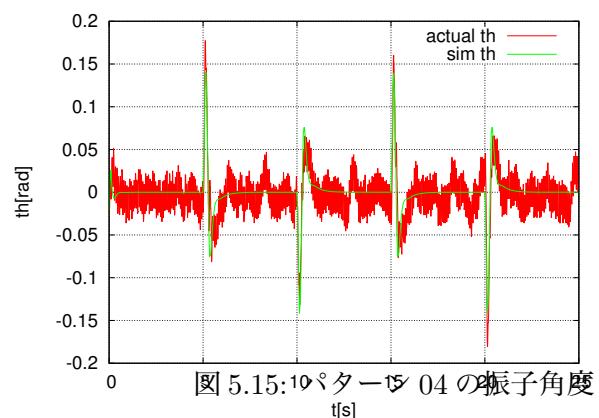
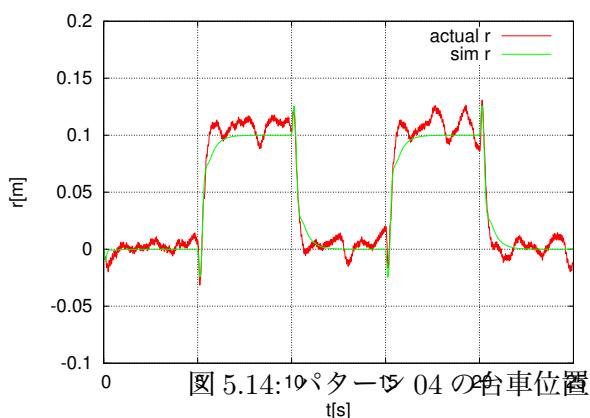
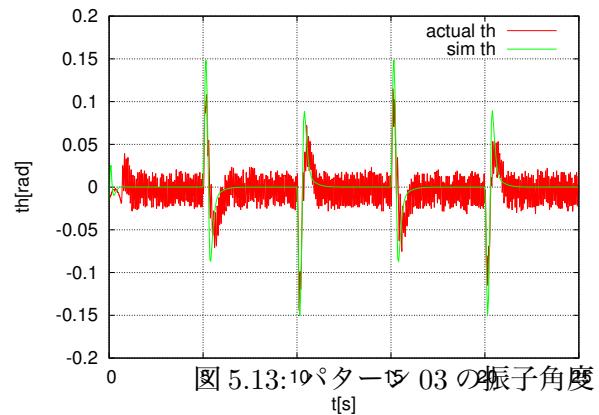
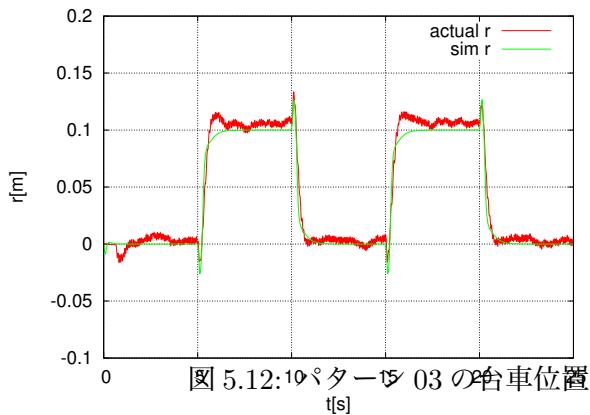
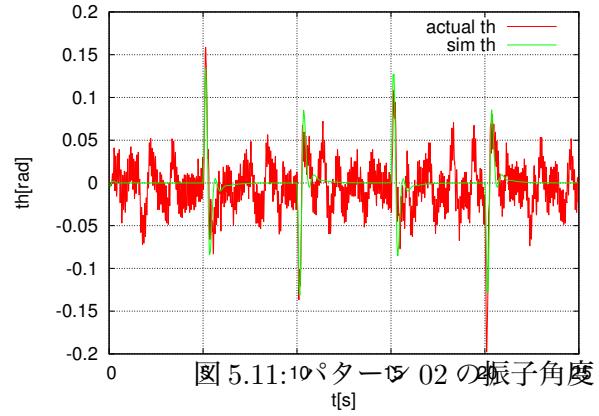
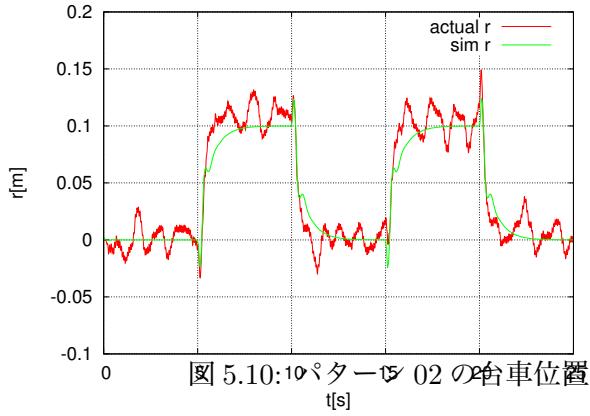
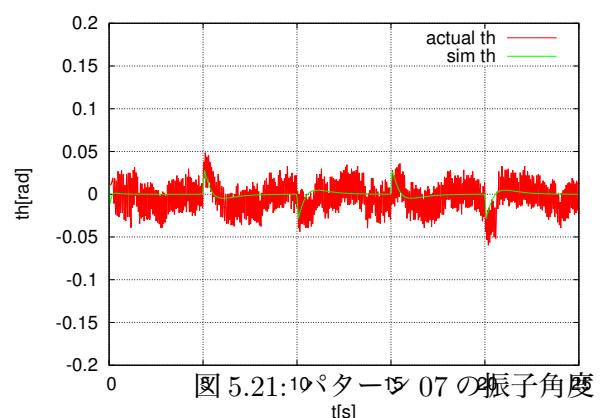
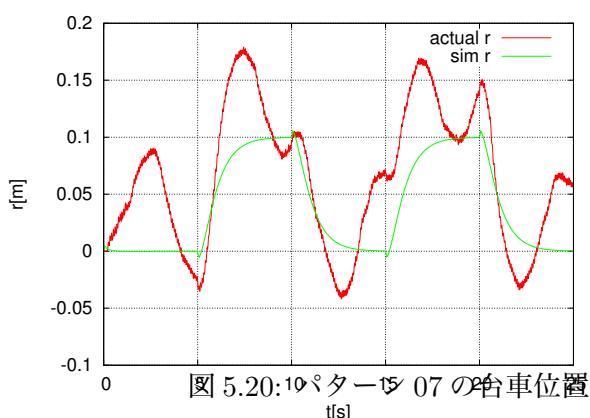
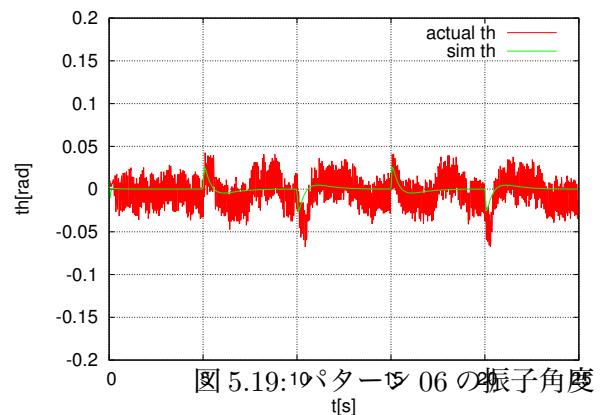
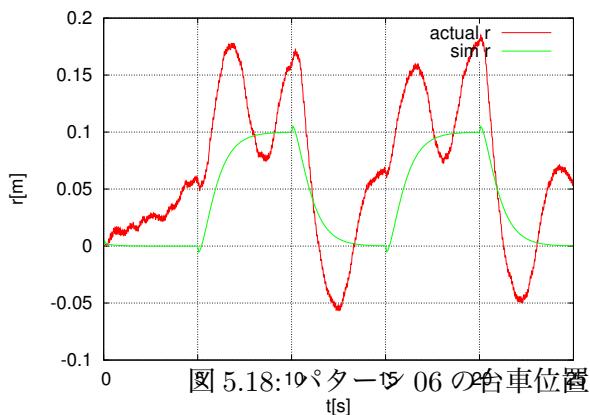
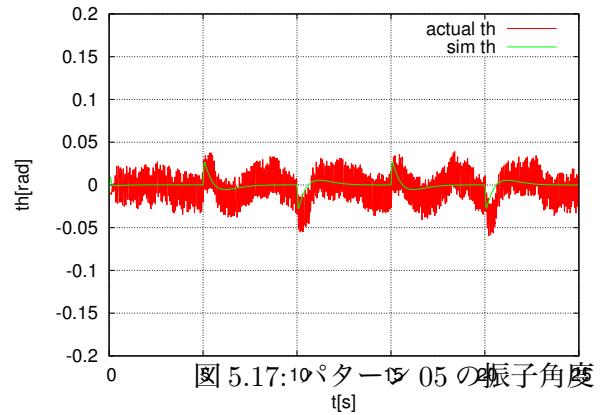
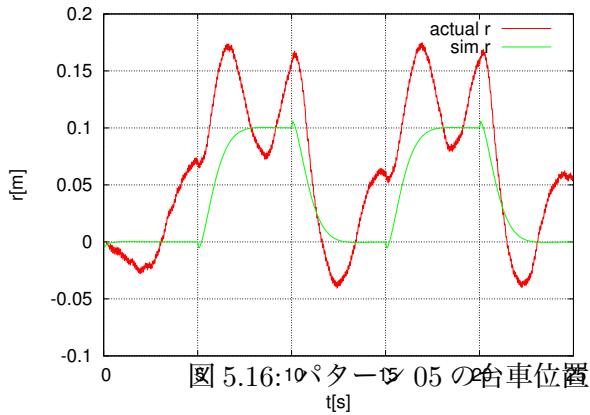


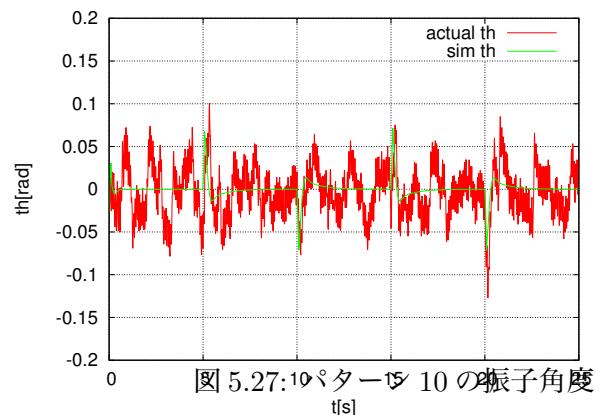
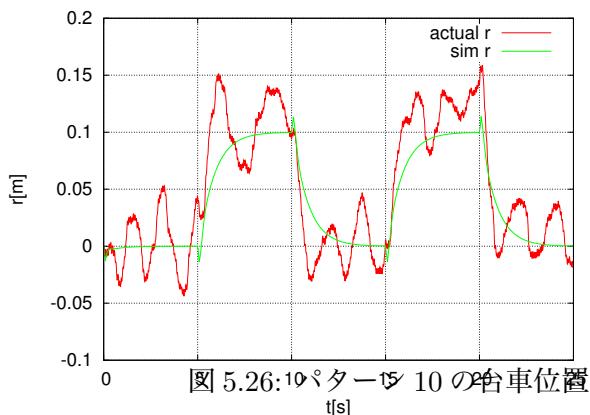
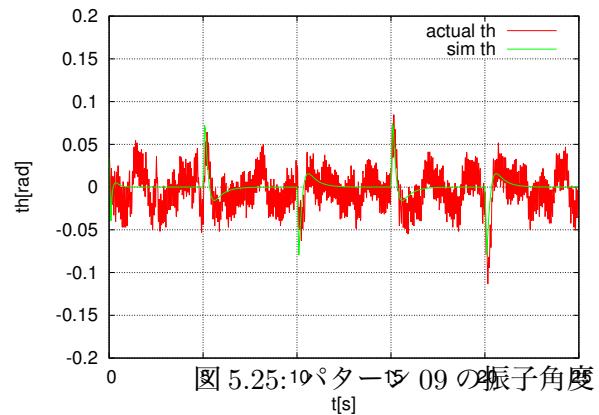
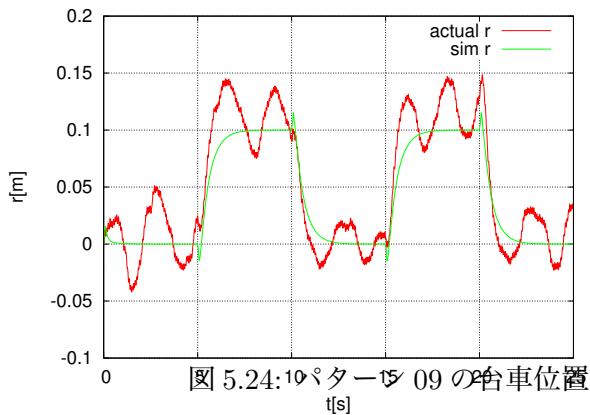
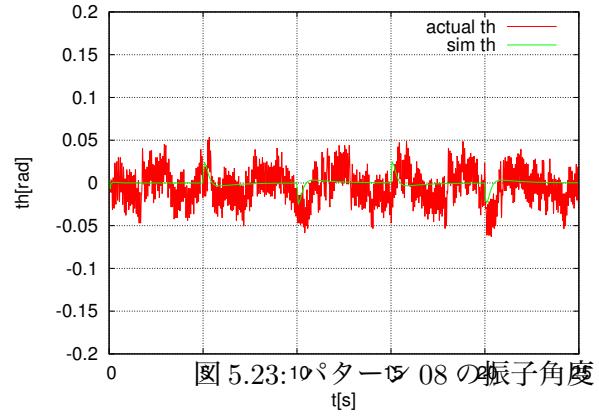
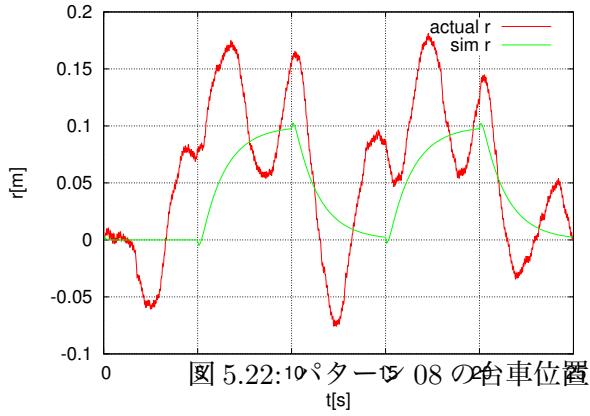
表 5.1: シミュレーションと実験の比較に用いるパラメータ

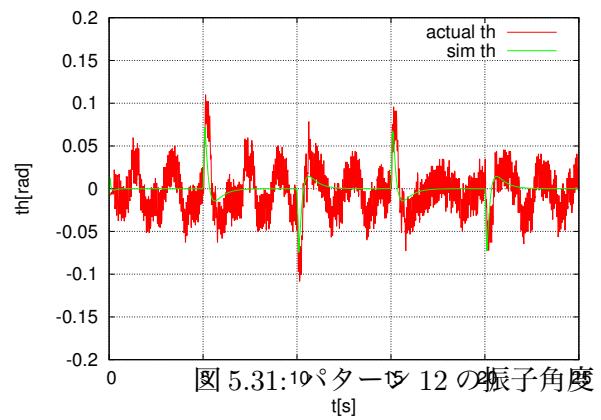
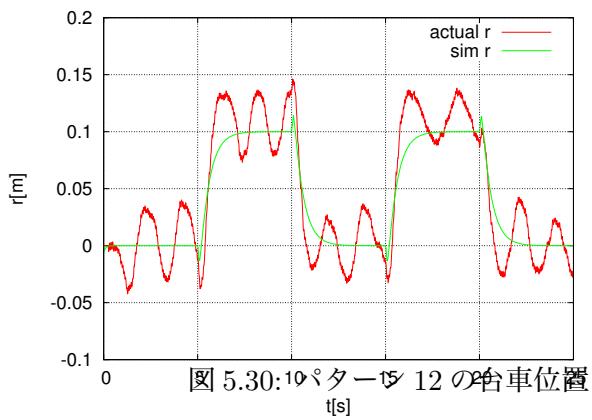
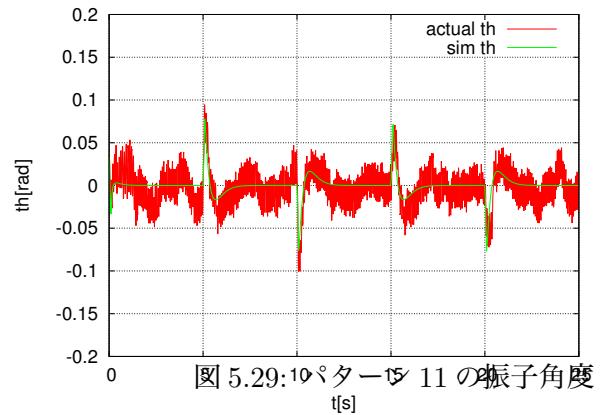
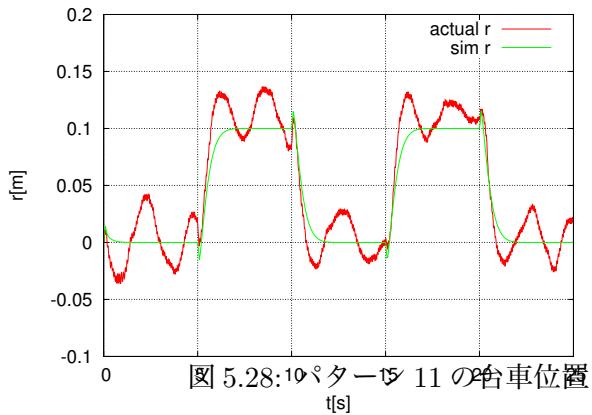
パターン	重み行列 Q	オブザーバの極 P	サンプリング周期 dt
パターン 01	diag(1.0E6, 1.0E5, 1, 1)	(-23, -23)	0.01
パターン 02	diag(1.0E6, 1.0E5, 1, 1)	(-50, -50)	0.01
パターン 03	diag(1.0E6, 1.0E5, 1, 1)	(-23, -23)	0.005
パターン 04	diag(1.0E6, 1.0E5, 1, 1)	(-50, -50)	0.005
パターン 05	diag(1.0E5, 1.0E6, 1, 1)	(-23, -23)	0.005
パターン 06	diag(1.0E5, 1.0E6, 1, 1)	(-50, -50)	0.005
パターン 07	diag(1.0E5, 1.0E6, 1, 1)	(-23, -23)	0.01
パターン 08	diag(1.0E5, 1.0E6, 1, 1)	(-50, -50)	0.01
パターン 09	diag(1.0E6, 1.0E6, 1, 1)	(-23, -23)	0.01
パターン 10	diag(1.0E6, 1.0E6, 1, 1)	(-50, -50)	0.01
パターン 11	diag(1.0E6, 1.0E6, 1, 1)	(-23, -23)	0.005
パターン 12	diag(1.0E6, 1.0E6, 1, 1)	(-50, -50)	0.005











置に同等の重みを置いたパターン 09 からパターン 12 までを比較すると、台車位置に重みを置いた方が目標値への収束が早いことがわかる。逆に、振子角度に重みを置いたパターン 03 とパターン 05 では、パターン 05 の方が振子角度の振動が抑えられていることがわかる。よって、重み行列の各成分に対応した変位の応答を制御することができた。

- オブザーバ

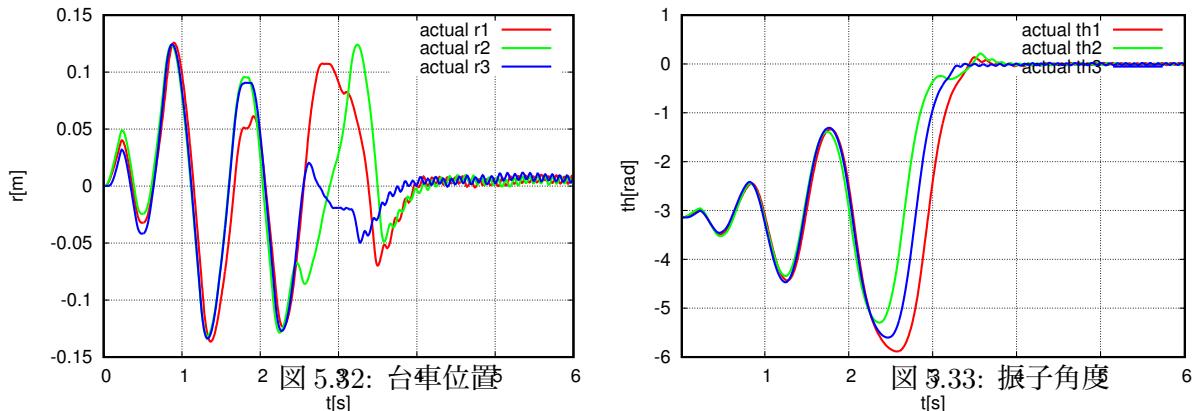
図 5.8 と図 5.10 を比較すると、オブザーバの極の実部が実軸負の方向に原点から離れているほうが、目標値への収束が早いと考えられる。しかし、今回行った実験の結果では、目標値への収束の速さに差異はなく、推定誤差も極が虚軸から遠いほど大きくなっていた。シミュレーション、実験で用いたオブザーバの極はすべて十分に大きく、オブザーバ以外の要因によって推定誤差、収束速度に変化が表れたと考えられる。

- サンプリング周期

サンプリング周期が小さいほど、短い間隔で状態のフィードバックをかけることができるため、より正確なシミュレーション、実験を行うことができる。すなわち、目標値への収束も早くなると言える。図 5.11、図 5.15 を比較すると、サンプリング周期が小さい図 5.15 の方が振動が抑えられ目標値へ早く収束しているため、理論通りの結果が得られたと言える。

5.0.8 振り上げ制御

振り上げ制御の実験には、表 4.4 のパラメータを用いる。ただし、振子が十分に振り上がり、安定化制御に移行した後に用いるパラメータは表 4.5 の値を用いる。以上のパラメータを用いた振り上げ制御の実験結果を図 5.32、図 5.33 に示す。



振り上げ制御の実験では、励振後にパラメータ k の影響が表れている。本実験では、振子角度が $|\theta| < 20^\circ \approx 0.35[\text{rad}]$ となった場合に安定化制御へ移行するように設定している。図 5.33 を見ると、 $|\theta| < 0.35$ となり安定化に移行した順は、パターン 2 が最も速く、パターン 1 が最も遅い結果になっている。実験の場合も、シミュレーション同様に k の値の影響はあまり表れていない。

第6章 おわりに

6.0.1 まとめ

今回の実験を通して、制御系のモデリングから制御器の設計、設計した制御器を用いたシミュレーション、実験を行い、制御の一連の流れを体験することができた。また、制御器を表現するための方法として、Java、MATLAB、Simulink の 3 通りの手段を用い、制御系 CAD の特徴や、実際にプログラムを書くこととの違いを理解することができた。シミュレーションでは意図した通りに動作するが、実験結果は異なる、あるいは逆の挙動を示すこともあります。シミュレーションと実験は異なることを実感した。本実験を通して理解した古賀研究室のプロダクトの特徴、制御の経験をこれから的研究に役立てたい。

関連図書

- [1] B. 倒立振子の安定化制御, 古賀雅伸
- [2] 制御工学実験第3 倒立振子の安定化制御, 古賀雅伸

第7章 付録 プログラム

7.1 目標値変更における安定化制御

xref.mm

```

1 #define LOGMAX 5000
2
3 Real ref;
4 Integer cmd, count;
5 Real smtime, yc, kc;
6 Matrix u, y;
7 Array data;
8
9 /*---- append ----*/
10 Matrix Ahd, Bhd, Chd, Dhd, Jhd, F;
11 Matrix z;
12
13 // センサとアクチュエータ関連の変数 (hardware.mmで使用される)
14 Matrix mp_data, PtoMR;
15
16 // メイン関数
17 Func void main()
18 {
19     void para_init(), var_init();
20     void on_task(), break_task(), off_task_loop();
21     void machine_ready(), machine_stop(), data_save();
22
23     para_init();           // パラメータの初期化
24     var_init();            // 変数の初期化
25     machine_ready();      // 実験装置の準備
26
27     rtSetClock(smtime);   // サンプリング周期の設定
28     rtSetTask(on_task()); // オンライン関数の設定(制御)
29     rtSetBreak(break_task()); // 割り込みキーに対応する関数の設定
30
31     rtStart();             // リアルタイム制御開始
32     off_task_loop();       // オフライン関数
33     rtStop();              // リアルタイム制御終了
34

```

```
35 machine_stop();           // 実験装置を停止
36
37 data_save();              // データを保存する
38 }
39
40 // パラメータの初期化
41 Func void para_init()
42 {
43     read Ahd, Bhd, Chd, Dhd, Jhd, F <- "para.mx";
44 }
45
46 // 変数の初期化
47 Func void var_init()
48 {
49     smtime = 0.005;          // サンプリング周期 [s]
50     cmd = 0;                // 制御出力を抑制
51     count = 0;              // ロギングデータの数
52     data = Z(4,LOGMAX);    // ロギングデータを保存する場所
53     z = [0 0]';             // 入力 [v]
54     ref = 0.0;
55 }
56
57 // オンライン関数
58 Func void on_task()
59 {
60     Matrix xh, xref;
61     Matrix sensor();
62     void actuator();
63
64     y = sensor();           // センサから入力
65
66     xh = Chd*z + Dhd*y;
67
68     if (rem(count * smtime, 5) == 0 && count != 0){
69         ref = abs(ref - 0.1);
70     }
71
72     xref = [ref 0 0 0]';
73     u = F * (xref - xh);
74     z = Ahd*z + Bhd*y + Jhd*u;
75
76     // リハーサル中でなければ
77     if (cmd == 1 && ! rtIsRehearsal()) {
78         actuator(u(1));      // アクチュエータへ出力
79     }
```



```

125     cmd = 1;
126     break;
127     /* 'R' */ case 0x52:
128     /* 'r' */ case 0x72:
129         gotoxy(5, 16);
130         print "台車の目標値 [m] : ";
131         read ref;
132         gotoxy(5, 16);
133         printf("                                     ");
134         break;
135     default:
136         break;
137     }
138 }
139 } while ( ! end_flag ); // If end_flag != 0, END
140 }
141
142 // 割り込みキーに対応する関数
143 Func void break_task()
144 {
145     void machine_stop();
146
147     rtStop();
148     machine_stop(); // 実験装置停止
149 }
150
151 // 実験装置の準備
152 Func void machine_ready()
153 {
154     void sensor_init(), actuator_init();
155
156     sensor_init();           // センサの初期化
157     actuator_init();        // アクチュエータの初期化
158
159     gotoxy(5,5);
160     printf("台車の初期位置 : レールの中央");
161     gotoxy(5,6);
162     printf("振子の初期位置 : 真下");
163     gotoxy(5,9);
164     pause "台車と振子を初期位置に移動し, リターンキーを入力して下さい。";
165     gotoxy(5,9);
166     printf("                                     ");
167     gotoxy(5,9);
168     pause "リターンキーを入力すると, 制御が開始されます。";

```

```

169     clear;
170 }
171
172 // 実験装置停止
173 Func void machine_stop()
174 {
175     void actuator_stop();
176
177     actuator_stop();
178 }
179
180 // データのファイルへの保存
181 Func void data_save()
182 {
183     String filename;
184     Array TT;
185
186     filename = "";
187     read filename;
188
189     if (count > 1) {
190         TT = [0:count-1]*smtime;
191         print [[TT][data(:,1:count)]] >> filename + ".mat";
192     }
193 }
```

7.2 振り上げ制御

swingup.mm

```

1 #define LOGMAX 5000
2
3 Integer cmd, count;
4 Real smtime;
5 Matrix u, y;
6 Array data;
7 Integer isSwinging;
8 Real r, theta, dr, dt, ddt;
9 Real sign_dtct, sat_ng;
10 Real v;
11 Real firstInputTime, secondInputTime, stopTime;
12
13
14 // センサとアクチュエータ関連の変数 (hardware.mmで使用される)
```

```
15 Matrix mp_data, PtoMR;  
16  
17 //  
18 //パラメータ、オブザーバの宣言開始  
19 //  
20 Matrix Ahd,Bhd,Chd,Dhd,Jhd,F; //パラメータ  
21 Matrix z; //オブザーバ  
22 Matrix xh, xref;  
23 Real r_max, theta_min; //角度と入力可能範囲の制限  
24 Real pre_r, pre_theta, pre_dt; //過去の位置、角度、角速度  
25 Real l,M,m,a,f,g,c,J,n,k,E0,E; //各定数  
26 //  
27 //ここまで  
28 //  
29  
30  
31 // メイン関数  
32 Func void main()  
{  
    void para_init(), var_init();  
    void on_task(), break_task(), off_task_loop();  
    void machine_ready(), machine_stop(), data_save();  
  
    para_init(); // パラメータの初期化  
    var_init(); // 変数の初期化  
  
    read n;  
    read k;  
  
    machine_ready(); // 実験装置の準備  
  
    rtSetClock(smtime); // サンプリング周期の設定  
    rtSetTask(on_task); // オンライン関数の設定(制御)  
    rtSetBreak(break_task); // 割り込みキーに対応する関数の設定  
  
    rtStart(); // リアルタイム制御開始  
    off_task_loop(); // オフライン関数  
    rtStop(); // リアルタイム制御終了  
  
    machine_stop(); // 実験装置を停止  
  
    data_save(); // データを保存する  
}  
// パラメータの初期化
```

```
60 Func void para_init()
61 {
62     read Ahd,Bhd,Chd,Dhd,Jhd,F <- "para.mx";
63 }
64
65 // 変数の初期化
66 Func void var_init()
67 {
68     smtime = 0.005;           // サンプリング周期 [s]
69     cmd = 0;                 // 制御出力を抑制
70     count = 0;               // ロギングデータの数
71     data = Z(5,LOGMAX);    // ロギングデータを保存する場所
72
73     u = [0.0];               // 入力 [v]
74     z = [0 0]';              // オブザーバの初期状態
75
76     r_max = 0.09;
77     theta_min = 20.0;
78
79     a = 0.49;
80     M = 1.001;
81     l = 0.12;
82     m = 0.038;
83     g = 9.8;
84     J = 3.9E-4;
85     f = 9.67;
86     c = 9.82E-5;
87     E0 = 0.0;
88
89     k = 1E5;
90     n = 0.25;
91
92     r = 0;
93     theta = -PI;
94     dr = 0.0;
95     dt = 0.0;
96     ddt = 0.0;
97     pre_r = 0.0;
98     pre_theta = -180.0/180.0*PI;
99     pre_dt = 0.0;
100
101    E = 0.0;
102    isSwinging = 1;
103    firstInputTime = 2.0E-1;
104    secondInputTime = 4.0E-1;
```

```
105     stopTime = 5E-1;
106 }
107
108
109
110 Func Integer sign(sig)
111 Real sig;
112 {
113     if (sig == 0){
114         return 0;
115     }
116     else{
117         return Integer(sig/abs(sig));
118     }
119 }
120
121
122 Func Real AngleWrapper(angle)
123 Real angle;
124 {
125
126     //角が180度を超えた場合、負の領域に
127     while(PI < 3*angle){
128         angle = angle - 2*PI;
129     }
130     //角が-180度を超えた場合、正の領域に
131     while(angle < -3*PI){
132         angle = angle + 2*PI;
133     }
134     return angle;
135
136 }
137
138 Func Matrix InputLimiter(u)
139 Matrix u;
140 {
141     if (u(1) < -15.0){
142         u(1) = -15.0;
143     }
144     else if (15.0 < u(1)){
145         u(1) = 15.0;
146     }
147     return u;
148 }
```

```

150 Func void DefVoltage()
151 {
152     /* --- calculate input voltage --- */
153
154     // 励振運動を必要とする場合
155     /* --- vibration block --- */
156     if (count*smtime <= firstInputTime){
157         u = [8.0];
158     }
159     else if (count*smtime <= secondInputTime){
160         u = [-8.0];
161     }
162     else if (count*smtime <= stopTime){
163         u = [0.0];
164     }
165     /* --- vibration block END --- */
166
167     // 励振後の運動
168     /* --- swing up start --- */
169     else{
170         // 振子の角度が安定化を始める角度にある && 台車の位置が加速可能
171         // 域内
172         if (abs(theta) <= (theta_min/180*PI) && abs(r) < r_max){
173             isSwinging = 0;
174         }
175
176         /* --- swing up process --- */
177         if (isSwinging == 1) {
178             // 力学的エネルギー
179             E = (J+m*l*l)*dt*dt/2 + m*g*l*(cos(theta) - 1);
180
181             // sign(dt*cos(theta))
182             sign_dtct = sign(dt*cos(theta));
183
184             // sat_ngの計算
185             sat_ng = max(-n*g, min(n*g, k*(E - E0)*sign_dtct));
186
187             // 台車の加速目標vの計算
188             v = -c*dt/(m*l*cos(theta)) + sat_ng;
189
190             // 入力値uの計算
191             u = [(f*dr - m*l*dt*dt*sin(theta) + m*l*ddt*cos(theta) + (M
192                 + m)*v)/(a)];
193
194             // 台車の位置が加速可能域外 && 進行方向が台車がはみ出す方向

```

```
193     if (r_max < abs(r) && 0 < r*u(1,1)){  
194         u = [0];  
195     }  
196     // 安定化に移行する場合  
197     else{  
198         u = [F*(xref - xh)];  
199     }  
200 }  
201 /* --- swing up process END --- */  
202  
203 // 入力電圧を制限  
204 u = InputLimiter(u);  
205  
206 /* --- calculate input voltage END --- */  
207 }  
208  
209 // オンライン関数  
210 Func void on_task()  
211 {  
212     Matrix sensor();  
213     void actuator();  
214  
215     // センサから入力  
216     y = sensor();  
217     xref = [0.0 0.0 0.0 0.0]';  
218  
219     // 角度の補正  
220     y(2) = AngleWrapper(y(2));  
221  
222     // 状態の推定値  
223     xh = Chd*z + Dhd*y;  
224  
225  
226     r = y(1);  
227     theta = y(2);  
228  
229     /* --- 直前の情報 --- */  
230     dr = (r - pre_r)/smtime;  
231     dt = (theta - pre_theta)/smtime;  
232     ddt = (dt - pre_dt)/smtime;  
233     pre_r = r;  
234     pre_theta = theta;  
235     pre_dt = dt;  
236     /* --- 直前の情報 END --- */  
237 }
```

```
238 isSwinging = 1;
239
240 // 入力する電圧の決定
241 DefVoltage();
242
243 // オブザーバの状態の更新
244 z = Ahd*z + Bhd*y + Jhd*u;
245
246 // リハーサル中でなければ
247 if (cmd == 1 && ! rtIsRehearsal()) {
248     // アクチュエータへ出力
249     actuator(u(1));
250 }
251
252 // データのロギング
253 if (cmd == 1 && count < LOGMAX) {
254     count++;
255     data(1:1, count) = u;
256     data(2:2, count) = [y(1)];
257     if (isSwinging && y(2) > 0){
258         data(3:3, count) = [y(2) - 2*PI];
259     }
260     else{
261         data(3:3, count) = [y(2)];
262     }
263     data(4:4, count) = [E]';
264     data(5:5, count) = [isSwinging];
265 }
266 }
267
268 // オフライン関数
269 Func void off_task_loop()
270 {
271     Integer end_flag;
272
273     end_flag = 0;
274
275     gotoxy(5, 6);
276     printf(" 'c': アクチュエータへ出力開始");
277     gotoxy(5, 7);
278     printf("ESC: アクチュエータへ出力停止");
279
280     do {
281         gotoxy(5, 11);
282         printf(" 台車位置 = %8.4f [m], 振子角度 = %8.4f [deg]",
```

```
283     y(1), y(2)/PI*180);
284     gotoxy(5, 12);
285     printf("入力 = %10.4f [N]" , u(1));
286
287     gotoxy(5, 13);
288     printf("運動エネルギー = %10.4f [N]" , E);
289
290     gotoxy(5, 15);
291     printf("データ数 = %4d, 時間 = %7.3f [s]" , count , count*smtime
292         );
293     if (rtIsTimeOut()) {
294         gotoxy(5, 19);
295         warning("\n時間切れ !\n");
296         break;
297     }
298
299     if (kbhit()) {
300         switch (getch()) {
301             case 0x1b: /* ESC */
302                 end_flag = 1;
303                 break;
304             /* 'c' */ case 0x43: // アクチュエータへ出力開始
305             /* 'C' */ case 0x63: // If 'c' or 'C' is
306                         // pressed, start motor
307                 cmd = 1;
308                 break;
309             default:
310                 break;
311         }
312     } while ( ! end_flag); // If end_flag != 0, END
313 }
314
315 // 割り込みキーに対応する関数
316 Func void break_task()
317 {
318     void machine_stop();
319
320     rtStop();
321     machine_stop(); // 実験装置停止
322 }
323
324 // 実験装置の準備
325 Func void machine_ready()
326 {
```

```
327 void sensor_init(), actuator_init();  
328  
329 // センサの初期化  
330 sensor_init();  
331 // アクチュエータの初期化  
332 actuator_init();  
333  
334 gotoxy(5,5);  
335 printf("台車の初期位置 : レールの中央");  
336 gotoxy(5,6);  
337 printf("振子の初期位置 : 真下");  
338 gotoxy(5,9);  
339 pause "台車と振子を初期位置に移動し, リターンキーを入力して下さい  
。";  
340 gotoxy(5,9);  
341 printf("");  
342 gotoxy(5,9);  
343 pause "リターンキーを入力すると, 制御が開始されます。";  
344 clear;  
345 }  
346  
347 // 実験装置停止  
348 Func void machine_stop()  
349 {  
350     void actuator_stop();  
351  
352     actuator_stop();  
353 }  
354  
355 // データのファイルへの保存  
356 Func void data_save()  
357 {  
358     String filename;  
359     Array TT;  
360  
361     filename = "";  
362     read filename;  
363  
364     if (count > 1) {  
365         TT = [0:count-1]*smtime;  
366         print [[TT][data(:,1:count)]] >> filename + ".mat";  
367     }  
368 }
```