エンコーダ・減速機付きダブルモータの ねじれトルク推定にバックラッシュモデル を用いた負荷側角度制御

三重大学工学研究科
電気電子工学専攻

2023(令和 5) 年度修士論文

池 田　遊 斗

目 次

第 1 章

序論

1.1 序論

ねじ締めや部品の取り付けのような環境との接触動作を伴う産業用ロボット，あるいは人との接触を伴う協働ロボットでは，軽量化や力制御性能の向上のためにモータ出力軸や減速機出力軸の先の剛性を低くする場合がある。しかし，機械系の低剛性化により機械共振周波数が低下し，高速動作時に振動しやすくなる問題がある。

減速機付き電磁モータの負荷側角度制御は，駆動側エンコーダを用いる手法と用いない手法に分類できる。駆動側エンコーダを用いない手法として，文献 [1] では負荷側に取り付けられた角度センサのみを用いる手法，文献 [2] [3] [4] では角度センサを用いず各相コイルの誘起電圧からモータ側角度を推定する手法が提案されている。これらの手法は減速機などの駆動側/負荷側間の機械部品のモデル化誤差の影響を受けやすい

駆動側に角度センサを用いる手法は，減速機出力軸の先に負荷側エンコーダを用いる手法と用いない手法に分けられる。文献 [5]-[8] では負荷側エンコーダを用いない手法が提案されている。文献 [5] では駆動側の角度センサのみを用いて負荷側角度を制御する手法が提案されている。文献 [6] [7] [8] では 2 つのモータを用いて 1 つの負荷を制御するノーバックラッシュ駆動が提案されている。しかし，減速機内のモデル化誤差や低剛性部品のねじれに起因する定常的な角度偏差が生じやすい。

負荷側エンコーダを用いる手法は，駆動側エンコーダと負荷側エンコーダの間のねじれ角度を制御器にフィードバック (Feed Back:FB) する手法と FB しない手法に分けられる。文献 [9]-[18] では，ねじれ角度を FB しない手法が提案されている。文献 [9] [10] [11] [12] [13] [14] の手法は，過渡なねじれトルクに起因する軸ねじれ振動が問題となる。文献 [15] では駆動側角度を用いた外乱オブザーバによりねじれトルクを推定し FB している。文献 [16] では駆動側と負荷側の重心を制御することで軸ねじれを抑えている。文献 [17] [18] ではトルクセンサで計測した軸ねじれトルクを FB することで振動を抑制している。しかしながら，ねじれ角度を制御器に FB しないため Backlash(BL) による自励振動の抑制が難しい。

ねじれ角度を FB する手法は，ねじれトルク推定にバックラッシュモデルを用いる手法と用いない手法に分類できる。文献 [19] [20] では低剛性機械部品のねじれトルクの推定の際にバックラッシュモデルを用いない手法が提案されている。この手法はねじれトルクの推定誤差に起因する目標値追従特性の劣化が問題となる。

ねじれトルクの推定に BL モデルを用いる手法は，1 つの負荷を 2 つのモータを用いて制御する手法と 1 つのモータで制御する手法に分類できる。文献 [21] [22] では負荷側角度を 1 つのモータを用いて制御する手法が提案されている。軸ねじれ角度と負荷側角度を独立に制御できないため、目標値追従特性と外乱抑圧特性を両立させることが難しい。

そこで本研究では負荷側角度を 2 つのモータを用いることで目標値追従特性と外乱抑圧特性の両立を試みる。

以下に，本論文の構成を示す。第 2 節では制御対象のモデル化をする。第 2.1 節では負荷側角度制御器の設計を行う。第 2.3 節ではシミュレーションと実験によって提案手法の有効性を示す。最後に第 3 節で結論を述べる。

なお，本論文中の $\widehat{\bigcirc}$ と $\bigcirc_n$ はそれぞれオンライン推定するパラメータと事前同定するパラメータを意味する。

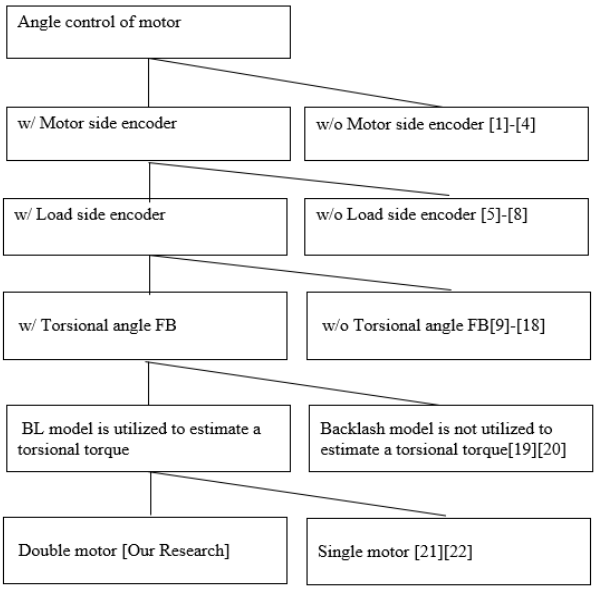


Fig. 1-1: Logic tree diagram of cited references.

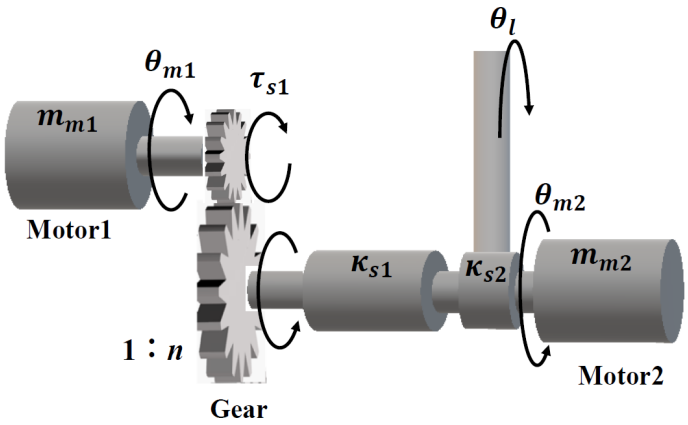


Fig. 2-1: Electromagnetic motor with a reduction gear, motor-side encoder, and load-side encoder.

第 2 章

制御対象のモデリング

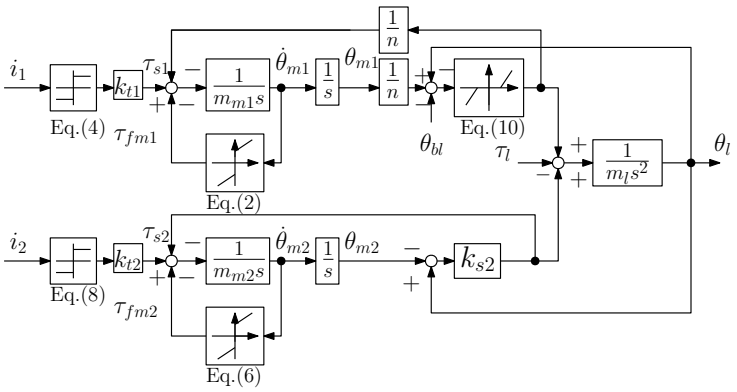


Fig. 2-2: The plant model or the mathematical model of Fig. 1.

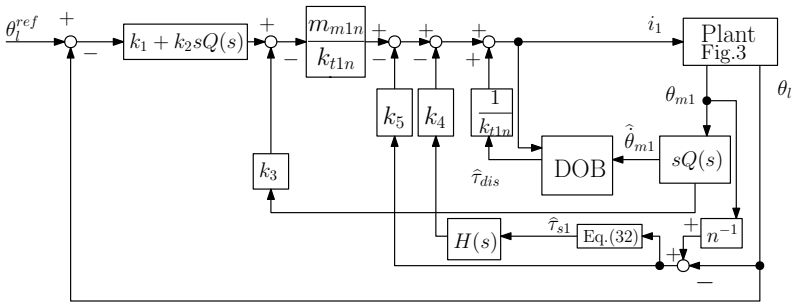


Fig. 2-3: The load-side angle control system of Motor1.

制御対象は減速機，モータ，駆動側/負荷側エンコーダから構成されている。第 2.0.1 節では制御対象の概要を説明し，第 2.0.2 節では減速機入力軸側の動力学を示し，第 2.0.3 節では減速機出力軸側の動力学を示す。

2.0.1　制御対象の概要

制御対象の構造を Fig. 2-1 に示す。ここで，$\theta_{m\circ}$，θ_l，$m_{m\circ}$，m_l，$\kappa_{s\circ}$，n はそれぞれ，減速前のモータ回転軸の角度，減速後の負荷側回転軸の角度，モータ側回転軸の慣性モーメント，負荷側回転軸の慣性モーメント，モータ側回転軸・負荷側回転軸間（以下，非線形バネ）のばね定数，

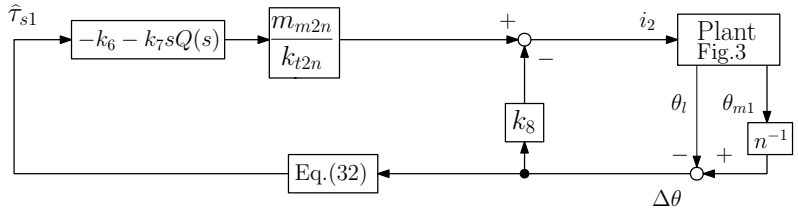


Fig. 2-4: The load-side angle control system of Motor2.

ギア比を表し，$\circ$ は Motor1，Motor2 を表す数字を示す。Fig. 2-2 に制御対象のブロック線図を示す。θ_{m1} と θ_l はエンコーダで計測できるものとする。$i_\circ$，$\dot{\theta}_{m\circ}$，$\dot{\theta}_l$，$\Delta\theta := \theta_{m\circ}/n - \theta_l$，$k_{t\circ}$ はそれぞれ，モータに流れる電流，モータ側回転軸の角速度，負荷側回転軸の角速度，非線形バネのねじれ角度，モータのトルク定数を表す。

2.0.2　モータと減速機

減速前のモータ回転軸周りの運動方程式は (2.1) 式で表す。

$$m_{m1}\ddot{\theta}_{m1} = k_{t1}i_{m1} - \tau_{fm1} - \frac{\tau_{s1}}{n} \tag{2.1}$$

$$\tau_{fm1} = \begin{cases} \frac{\tau^+_{cm1}}{\alpha}\mathrm{sat}(\dot{\theta}_{m1}) + d^+_{m1}\dot{\theta}_{m1} & (0 \le \dot{\theta}_{m1}) \\ \frac{\tau^-_{cm1}}{\alpha}\mathrm{sat}(\dot{\theta}_{m1}) + d^-_{m1}\dot{\theta}_{m1} & (\dot{\theta}_{m1} < 0) \end{cases} \tag{2.2}$$

$$\mathrm{sat}(\dot{\theta}_{\mathrm{m1}}) = \begin{cases} \alpha & (\alpha \le \dot{\theta}_{m1}) \\ \dot{\theta}_{m1} & (-\alpha < \dot{\theta}_{m1} < \alpha) \\ -\alpha & (\dot{\theta}_{m1} \le -\alpha) \end{cases} \tag{2.3}$$

$$\mathrm{sat}(i_1) = \begin{cases} i_{1m} & (i_{1m} \le i_1) \\ i_1 & (-i_{1m} < i_1 < i_{1m}) \\ -i_{1m} & (i_1 \le -i_{1m}) \end{cases} \tag{2.4}$$

$$m_{m2}\ddot{\theta}_{m2} = k_{t2}i_{m2} - \tau_{fm2} - \tau_{s2} \tag{2.5}$$

$$\tau_{fm2} = \begin{cases} \frac{\tau_{cm2}^{+}}{\beta}\mathrm{sat}(\dot{\theta}_{m2}) + d_{m2}^{+}\dot{\theta}_{m2} & (0 \leq \dot{\theta}_{m2}) \\ \frac{\tau_{cm2}^{-}}{\beta}\mathrm{sat}(\dot{\theta}_{m2}) + d_{m2}^{-}\dot{\theta}_{m2} & (\dot{\theta}_{m2} < 0) \end{cases} \tag{2.6}$$

$$\mathrm{sat}(\dot{\theta}_{\mathrm{m2}}) = \begin{cases} \beta & (\beta \leq \dot{\theta}_{m2}) \\ \dot{\theta}_{m2} & (-\beta < \dot{\theta}_{m2} < \beta) \\ -\beta & (\dot{\theta}_{m2} \leq -\beta) \end{cases} \tag{2.7}$$

$$\mathrm{sat}(i_2) = \begin{cases} i_{2m} & (i_{2m} \leq i_2) \\ i_2 & (-i_{2m} < i_2 < i_{2m}) \\ -i_{2m} & (i_2 \leq -i_{2m}) \end{cases} \tag{2.8}$$

ここで，$\tau_{fm\circ}$，$\ddot{\theta}_{m\circ}$，$\tau_{cm\circ}^{\pm}(>0)$，$d_{m\circ}^{\pm}(>0)$，$\alpha(>0)$，$\beta(>0)$，i_{1m}，i_{2m} はそれぞれ，モータおよび減速機の摩擦トルク，モータの角加速度，モータのクーロン摩擦トルク，モータの粘性係数，ゼロ速の閾値，Motor1 の電流飽和の閾値，Motor2 の電流飽和の閾値を表す。モータと減速機の摩擦トルクをモータ側に集約して考えるものとする。

2.0.3　負荷側

負荷側回転軸周りの運動方程式は (2.9) 式で表す。

$$m_l\ddot{\theta}_l = \tau_{s1} - \tau_{s2} - \tau_l \tag{2.9}$$

$$\tau_{s1} = \begin{cases} \kappa_{s1}(\Delta\theta - p_{bl}) + d_{s1}\Delta\dot{\theta} & (p_{bl} \leq \Delta\theta) \\ \kappa_{s1}(\Delta\theta + n_{bl}) + d_{s1}\Delta\dot{\theta} & (\Delta\theta \leq -n_{bl}) \\ 0 + d_{s1}\Delta\dot{\theta} & (-n_{bl} < \Delta\theta < p_{bl}) \end{cases} \tag{2.10}$$

$$\tau_{s2} = \kappa_{s2}(\theta_{m2} - \theta_l) \tag{2.11}$$

$$l_{bl} = p_{bl} + n_{bl} \tag{2.12}$$

ここで，τ_s，$\ddot{\theta}_l$，$p_{bl}(>0)$，$n_{bl}(>0)$，l_{bl} はそれぞれ，非線形バネのねじれトルク，負荷側回転軸の角加速度，正側のバックラッシュ幅，負側のバックラッシュ幅，正負合計のバックラッシュ幅を表す。

2.1　負荷側角度制御系の設計

第 2.1.1 項，第 2.1.2 項，第 2.1.3 項ではそれぞれ，負荷側角度制御系，外乱オブザーバ，モータ角速度制御器を設計する。第 2.1.4 項では提案法と従来法の違いを示す。

2.1.1　負荷側角度制御系

負荷側角度制御系のブロック線図を Fig. 2-3，Fig. 2-4 に示す。負荷側角度制御系は外乱オブザーバ，モータ角速度制御器，ねじれトルク推定器から構成される。負荷側エンコーダにより直接計測可能な θ_l を制御することを目的とする。

2.1.2　外乱オブザーバ (DOB)

推定外乱トルク $\widehat{\tau}_{dis}$ は (2.13) 式で与える。

$$\widehat{\tau}_{dis} = Q(Qk_{t1n}i_{m1} + gm_{m1n}\widehat{\dot{\theta}}_{m1}) - gm_{m1n}\widehat{\dot{\theta}}_{m1} \tag{2.13}$$

$$\widehat{\dot{\theta}}_{m1} = sQ\theta_{m1} \tag{2.14}$$

ここで，$Q(s) := g_1/(g_1 + s)$ はローパスフィルタであり，g_1 はカットオフ角周波数を表す。g_1 は制御周期とエンコーダ分解能から決定する。

2.1.3　モータ角度制御器

Motor1

Motor1 のモータ角度制御器は負荷角度偏差に対する比例・微分制御器，ねじれトルク推定器および DOB で構成する。

$$\begin{aligned} i_1 = {} & ((k_1 + k_2sQ)(\theta_l^{ref} - \theta_l) - k_3\widehat{\dot{\theta}}_{m1})\frac{m_{m1n}}{k_{t1n}} \\ & -k_4H\widehat{\tau}_{s1} - k_5\Delta\theta + \frac{\widehat{\tau}_{dis}}{k_{t1n}} \end{aligned} \tag{2.15}$$

Motor2

Motor2 のモータ角度制御器は $\hat{\tau}_{s1}$ に対する比例・微分制御器で構成する。

$$i_2 \quad = \quad (k_6 + k_7 sQ)\hat{\tau}_{s1}\frac{m_{m2n}}{k_{t2n}} - k_8\Delta\theta \tag{2.16}$$

ここで，k_1 は負荷側角度比例ゲイン，k_2 は負荷側角度微分ゲイン，k_3 は駆動側角速度フィードバックゲイン，k_4 はねじれトルク FB ゲイン，k_5 はねじれ角度 FB ゲイン，k_6 はねじれトルク比例ゲイン，k_7 はねじれトルク微分ゲイン，k_8 はねじれ角度 FB ゲインを表す。

従来法と提案法

従来法 1(文献 (9)**)**(conv1)　(2.17) 式のように，減速機入力軸側のモータのみを用い，ねじれトルクを制御器に FB しない手法を従来法 1 とする。

$$k_4 \quad = \quad 0 \tag{2.17}$$

$$k_5 \quad = \quad 0 \tag{2.18}$$

$$k_6 \quad = \quad 0 \tag{2.19}$$

$$k_7 \quad = \quad 0 \tag{2.20}$$

$$k_8 \quad = \quad 0 \tag{2.21}$$

従来法 2(文献 (22)**)**(conv1)　(2.22) 式のように，減速機入力軸側のモータのみを用い，ねじれトルクを HPF に通し，制御器に FB する手法を従来法 2 とする。

$$k_4 \quad \neq \quad 0 \tag{2.22}$$

$$k_5 \quad \neq \quad 0 \tag{2.23}$$

$$k_6 \quad = \quad 0 \tag{2.24}$$

$$k_7 \quad = \quad 0 \tag{2.25}$$

$$k_8 \quad = \quad 0 \tag{2.26}$$

提案法 (prop)　(2.27) 式のように，減速機入力軸側と出力軸側の両方のモータを用いて負荷側角度を制御する手法を提案法とする。

$$k_4 \neq 0 \tag{2.27}$$

$$k_5 \neq 0 \tag{2.28}$$

$$k_6 \neq 0 \tag{2.29}$$

$$k_7 \neq 0 \tag{2.30}$$

$$k_8 \neq 0 \tag{2.31}$$

2.1.4　ねじれトルク推定器

エンコーダにより計測される θ_l，θ_{m1} より (2.32) 式を用いてねじれトルクを推定する。

$$\widehat{\tau}_{s1} = \begin{cases} \{\kappa_{s1n}(\Delta\theta - p_{bln}) & \\ +d_{s1n}\Delta\widehat{\dot{\theta}}\} & (p_{bln} \leq \Delta\theta) \\ \{\kappa_{s1n}(\Delta\theta + n_{bln}) & \\ +d_{s1n}\Delta\widehat{\dot{\theta}}\} & (\Delta\theta \leq -n_{bln}) \\ (0 + d_{s1n}\Delta\widehat{\dot{\theta}}) & (-n_{bln} < \Delta\theta < p_{bln}) \end{cases} \tag{2.32}$$

$$H(s) = \frac{s}{s + g_2} \tag{2.33}$$

ここで，$\Delta\widehat{\dot{\theta}} = \widehat{\dot{\theta}}_{m1}/n - \widehat{\dot{\theta}}_l$，$g_2$ はハイパスフィルタのカットオフ周波数とする。

2.1.5　制御器ゲインの設計

ここでは，(2.34) 式に基づき k_1，k_2，k_3，k_4，k_5，k_6，k_7，k_8 を最適化する。最適化時は，$m_l = m_{ln}$，$m_m = m_{mn}$，$k_t = k_{tn}$，$\kappa_s = \kappa_{sn}$，$d_m^+ = d_{mn}^+$，$d_l^+ = d_{ln}^+$，$p_{bln} = n_{bln} = \tau_l = 0$ を仮定する。

設計条件

$$\min\gamma \tag{2.34}$$

Table 2.1: Desgined parameter for simulation and experiment.

	k_1	k_2	k_3	k_4	k_5	g_2	k_6	k_7	k_8
conv1	100000	300	60	0	0	∞	0	0	0
conv2	190000	3000	105	1.2	0.07	1.2	0	0	0
prop	190000	3000	105	0	0	0	500	300	0.07

ここで，$\gamma := ||G_{cmd}(j\omega)W(j\omega)||_2 + ||G_{bl}(j\omega)W_b(j\omega)||_2$ とする。G_{cmd} は θ_l^{ref} から $\theta_l^{ref} - \theta_l$ までの閉ループ伝達関数，G_{bl} は θ_{bl} から θ_l までの閉ループ伝達関数，$W(j\omega)$=$10/(j\omega + 10)$，$W_b(j\omega)$=$1/(j\omega + 1)$ とした。また，$\theta_l^{ref} - \theta_l$ から θ_l までの一巡伝達関数のナイキスト軌跡の位相余裕を 50 deg 以上とする制約条件を課すものとする。現段階では，提案法については試行錯誤的に決定したゲインを用いている。

Table 2.2: Physical parameters for simulation and experiment.

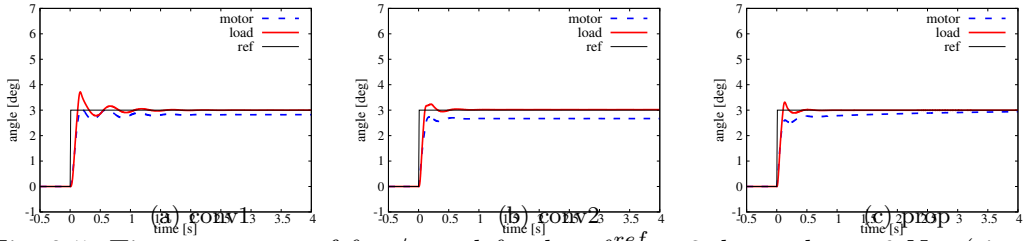
Gear ratio n	100
Spring constant k_{s1n}	0.0524 Nm/deg
Torque constant of motor1 k_{t1n}	0.011 Nm/A
Torque constant of motor2 k_{t2n}	0.0934 Nm/A
Inertia moment of motor1 m_{m1n}	0.0000000731 kgm^2
Inertia moment of motor12 m_{m2n}	0.0000531 kgm^2
Inertia moment of load m_{ln}	0.001 kgm^2
Control period t_c	1 ms
Coulomb friction of motor1 τ^+_{cm1n}	0.00155 Nm
Coulomb friction of motor1 τ^-_{cm1n}	0.00200 Nm
Damping of motor1 d^+_{m1n}	1.48×10^{-7} Nms/deg
Damping of motor1 d^-_{m1n}	2.18×10^{-6} Nms/deg
Threshold value α	286 deg/s
Coulomb friction of motor2 τ_{cm2n+}	0.01 Nm
Coulomb friction of motor2 τ_{cm2n-}	0.01 Nm
Damping of motor2 d_{m2n+}	2.27×10^{-8} Nms/deg
Damping of motor2 d_{m2n-}	2.27×10^{-8} Nms/deg
Backlash width l_{bln}	0.5 deg
Damping of spring d_{s1n}	0.86 Nms/deg
threshold value γ	0.76 A

2.2　シミュレーション

本節では，負荷側角度制御のシミュレーション結果を示すことで第 2.1.3 節で提案したダブルモータシステムの有用性を証明する。

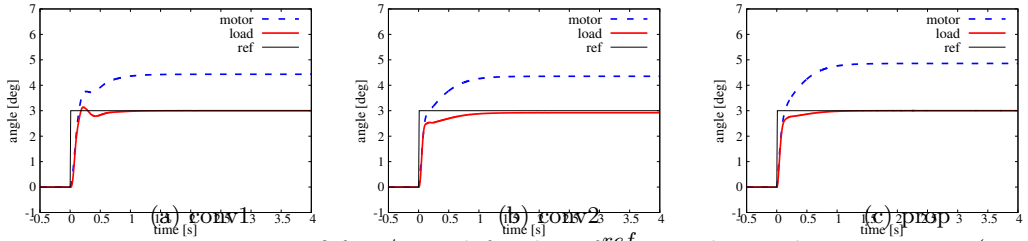
2.2.1　設定

物理パラメータは Table 2.2，設計パラメータは従来法１，従来法２については第 2.1.5 節で最適化したパラメータ，提案法については試行錯誤的に決定したパラメータを用いる。



(a) conv1　(b) conv2　(c) prop

Fig. 2-5: Time responses of θ_{m1}/n and θ_l when $\theta_l^{ref} = 3$ deg and $\tau_l = 0$ Nm (simulation)



(a) conv1　(b) conv2　(c) prop

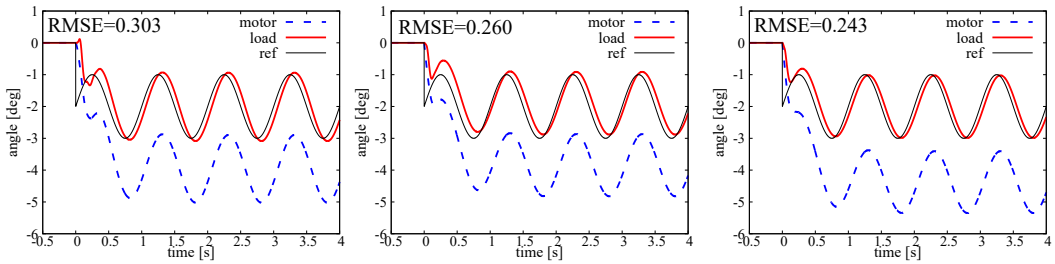
Fig. 2-6: Time responses of θ_{m1}/n and θ_l when $\theta_l^{ref} = 3$ deg and $\tau_l = 0.1$ Nm (simulation)

2.2.2　シミュレーション結果

Fig. 2-5 に θ_l^{ref}=3 deg，$\tau_l = 0$ Nm の場合のシミュレーション結果を示す。Fig. 2-11(a)，(b)，(c) は第 2.1.3 節で説明した従来法と提案法におけるシミュレーション結果である。この結果より，従来法 1 では立ち上がり時にバックラッシュに起因する振動が発生していることが分かる。

Fig. 2-6 に θ_l^{ref}=3 deg，$\tau_l = 0.1$ Nm の場合のシミュレーション結果を示す。この結果より，負荷が加わった場合には，どの手法においても良好な角度制御が達成されていることが分かる。

Fig. 2-7 に θ_l^{ref}=-2+$sin(2\pi t)$ deg，$\tau_l = -0.1$ Nm の場合のシミュレーション結果を示す。図中左上の RMSE は 2 乗平均平方根誤差を示す。この結果より，正弦波の負荷側角度指令値が加わった場合には提案法が最も角度偏差が小さく制御ができていることが分かる。



(a) conv1　(b) conv2　(c) prop

Fig. 2-7: Time responses of θ_{m1}/n and θ_l when $\theta_l^{ref} = sin(2\pi t)$ deg and $\tau_l = -0.1$ Nm(simulation)

2.3　実験

本節では，負荷側角度制御の実験結果を示すことで第 2.1.3 節で提案したダブルモータシステムの有用性を証明する。

2.3.1　設定

実験装置を Fig. 2-10 に示す。駆動側モータおよび減速機としてハーモニックドライブ・システムズ製の RSF-5B-100-E050-C を用い，負荷トルク τ_l を模擬するための電磁モータとしては maxon 製の RE50 を用いる。エンコーダは駆動側に分解能 500 pulse/rev. のものを，負荷側に分解能 54000 pulse/rev. のものを，それぞれカウンタボードで 4 逓倍して用いる。減速機の出力軸と負荷用モータの回転軸の間には低剛性カップリングを挿入した。実験開始前に駆動側ギアをゆっくり動かし，負荷側のギアに触れた地点を初期位置 ($p_{bln} = \theta_m/n = \theta_l = 0$) と設定してから実験を開始する。負荷トルク τ_l は負荷用の電磁モータを電流制御することで調整する。負荷側角度制御の実験で用いる物理パラメータには，第 2.2 節で示した Table 2.2 の値を用いた。

2.3.2　モータ及び減速機の摩擦特性の同定

駆動側モータと減速機で発生する摩擦トルクを同定する。プラントに対して，電流指令値 i_{m1} を (2.35) 式で与えることで，モータ角速度 $\dot{\theta}_m$ を制御する。

$$i_1 = (k_{pv} + \frac{k_{iv}}{s}) \cdot \frac{m_{mn}}{k_{tn}} (\dot{\theta}_{m1}^{ref} - \hat{\dot{\theta}}_{m1}) + \frac{\hat{\tau}_{dis}}{k_{t1n}} \tag{2.35}$$

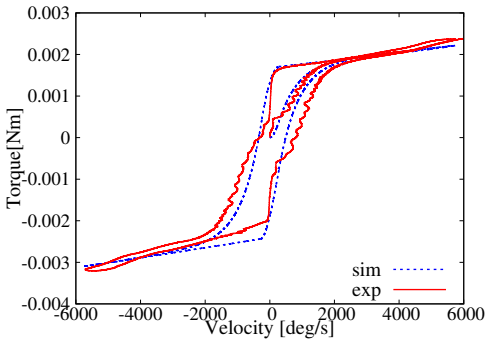


Fig. 2-8: $\widehat{\dot{\theta}}_m$ - $\widehat{\tau}_{dis}$ curve.

ここで，$\dot{\theta}_{m1}^{ref}$ は三角波角速度指令値，$k_{pv} = 1500\ \mathrm{s}^{-1}$，$k_{iv} = 4500\ \mathrm{s}^{-2}$，$g_1 = 2$ rad/s とした。駆動側と負荷側をつなぐカップリングが取り外されている状態で同定試験を行うため，$\tau_s = 0$ が成り立つ。 実験の結果を Fig. 2-8 に示す。ここで，実線はシミュレーション，点線は実験値を表している。このグラフより，シミュレーション値と実験値の特性が近いことから，Equation (12) の摩擦モデルおよび Table 2.2 の中の τ_{cm1n}^{+}，τ_{cm1n}^{-}，d_{m1n}^{+}，d_{m1n}^{-} の妥当性が示された。

2.3.3　非線形バネ特性の同定

Fig. 2-10 の装置を用いて減速前のモータ回転軸と減速後の負荷側回転軸の間の非線形バネ特性を同定する。駆動側角度 θ_m は PD 制御器を用いて (2.36) 式のように制御する。

$$i_1 \quad = \quad (k_{iv} + k_{pv} sQ) \cdot \frac{m_{m1n}}{k_{t1n}} (\theta_{m1}^{ref} - \theta_{m1}) + \frac{\widehat{\tau}_{dis}}{k_{t1n}} \tag{2.36}$$

ここで，$\theta_{m1}^{ref} = 0$ deg，$k_{iv} = 50000\ \mathrm{s}^{-2}$，$k_{pv} = 650\ \mathrm{s}^{-1}$ とした。負荷トルクは $\tau_l = 0.2\sin(t)$ Nm を時刻 $t = 0$ s から $t = 3\pi$ s まで印加した。Fig. 2-9 に非線形バネのねじれ角度 $\Delta\theta$ に対する負荷トルク τ_l の結果を示す。ここで，実線，点線は，それぞれシミュレーション値，実験値を表す。バックラッシュ領域幅 l_{bl}=$p_{bl} + n_{bl}$ は Fig. 2-9 から同定した。また，実験結果を直線近似し，その傾きをばね定数 κ_{s1n} とした。負荷トルクを模擬するためのモータの摩擦特性によってヒステリシス特性が現れている。

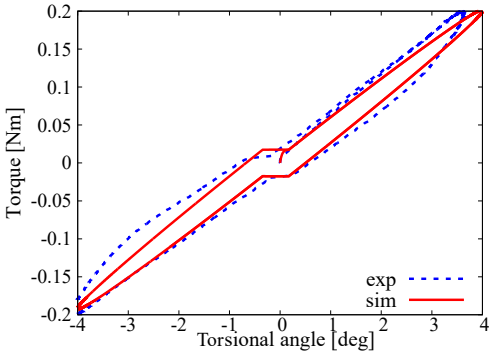


Fig. 2-9: $\Delta\theta$ - $\widehat{\tau_l}$ curve.

2.3.4　実験結果

Fig. 2-11 に θ_l^{ref}=3 deg，$\tau_l = 0$ Nm の場合の実験結果を示す。Fig. 2-11(a)，(b)，(c) は第 2.1.3 節で説明した従来法と提案法における実験結果である。この実験結果より，従来法 1 ではバックラッシュに起因する自励振動が発生しており，従来法 2 と提案法では定常角度偏差が少なく制御ができていることが分かる。

Fig. 2-12 に θ_l^{ref}=3 deg，$\tau_l \neq 0$ Nm の場合の実験結果を示す。本実験では，アームの先端をスポンジに押し当てることにより負荷を加えている。この実験結果より，従来法 1 では制御器ゲインを大きくできないため立ち上がり時間が長くなっていることが分かる。従来法 2 と提案法では，定常角度偏差とオーバーシュートともに小さく，角度制御ができていることが分かる。

Fig. 2-13 に θ_l^{ref}=$-2 + sin(2\pi t)$ deg の場合の実験結果を示す。図中左上の RMSE は 2 乗平均平方根誤差を示す。この結果より，提案手法では他の手法と比較し偏差が少なく，角度制御ができていることが分かる。

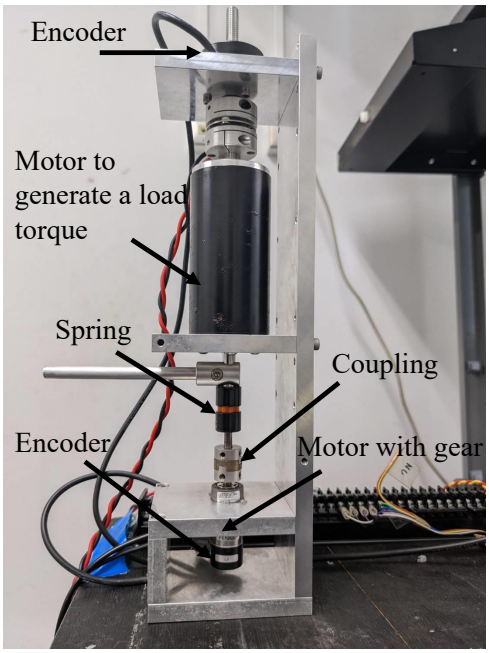
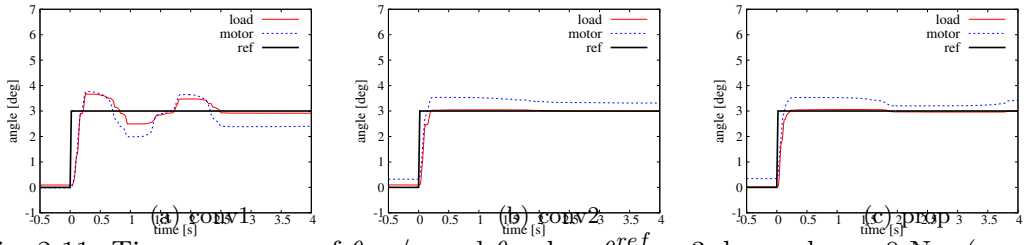
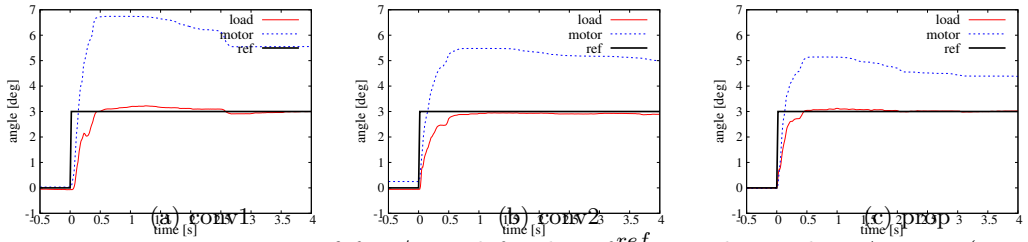


Fig. 2-10: Experimental system that consists of an electric motor, drive-side encoder, load-side encoder, gear reducer, and shaft coupling.



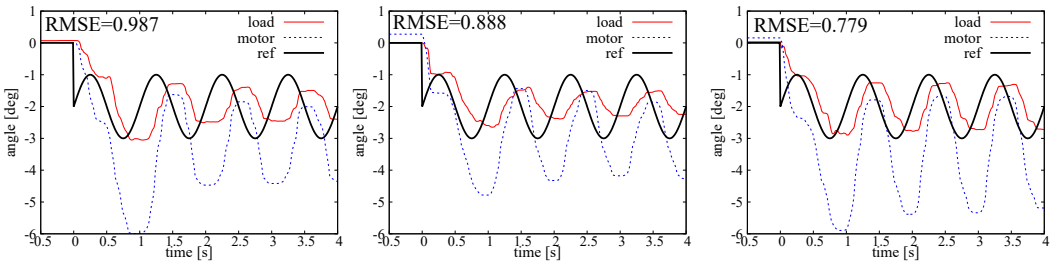
(a) conv 1　(b) conv 2　(c) prop

Fig. 2-11: Time responses of θ_{m1}/n and θ_l when $\theta_l^{ref} = 3$ deg and $\tau_l = 0$ Nm (experiment)



(a) conv1　(b) conv2　(c) prop

Fig. 2-12: Time responses of θ_{m1}/n and θ_l when $\theta_l^{ref} = 3$ deg and $\tau_l \neq 0$ Nm (experiment)



(a) conv1　(b) conv2　(c) prop

Fig. 2-13: Time responses of θ_{m1}/n and θ_l when $\theta_l^{ref} = -2 + sin(2\pi t)$ deg (experiment)

第 3 章

結論

本研究では，モータ/負荷側エンコーダ，低剛性カップリング，減速機を有するダブルモータのための負荷側角度制御器の設計を行った。ねじれトルクを制御器に FB しない従来法 1 や，負荷側角度とねじれトルクを一つのモータで制御する従来法 2 と比較して，定常角度偏差や振動の少ない負荷側角度制御が達成されることが実験により確認された。

今後の課題としては，ゲインの最適化が挙げられる。

Appendix A

電気系の仕様

A.1 概要

本研究で用いた実験システムの全体図を Fig. A-2 に示す。本システムは PC，電源部，減速機付き電磁モータ駆動部，電磁モータ駆動部からなる。PC には，DA ボード（インタフェース製 PCI-3340），カウンタボード（インタフェース製 PCI-6205C）が接続されている。AC 供給用の電源部は，コンセントから AC100V が供給され，ノイズフィルタ（Schaffner 製 FN9222-15-06）およびスイッチを介しスライダック（東京理工舎社製 RSA-10）に接続されている。DC 供給用の電源部は，コンセントから AC100V が供給され，スイッチおよびフィルタを介しスイッチング電源に接続されている。スイッチおよびフィルタは，Fig. A-2 のようになっており，主電源スイッチ，副電源スイッチ，LED，ノイズフィルタ（TDK ラムダ製 RSEN-2016）からなる。スイッチング電源は，モータドライバの最大電流を考慮し，DC24V 用にミスミ製 ESP10-300-24，DC5V 用にコーセル製 R10A-5-N を用いた。主な電気電子機器のメーカー、型式は Table A.1 にまとめた。また，主な電気電子機器の仕様書は Fig. A-1 の System，見積書は Fig. A-1 の quotation に入っている。

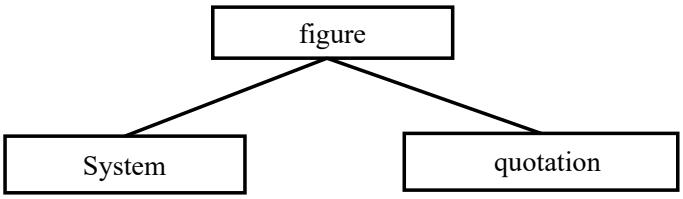


Fig. A-1: 電気系の仕様のディレクトリ

Table A.1: 主な電気電子機器について.

DA ボード	インタフェース	PCI-3340
カウンタボード	インタフェース	PCI-6205C
AC 供給部のノイズフィルタ	Schaffner	FN9222-15-06
DC 供給部のノイズフィルタ	TDK ラムダ	RSEN-2016
スライダック	東京理工舎社	RSA-10
DC24V 用スイッチング電源	ミスミ	ESP10-300-24
DC5V 用スイッチング電源	コーセル	R10A-5-N
負荷用モータ	マクソンモータ	RE50
負荷側エンコーダ	マイクロテックラボラトリー	MEH-28
負荷用モータ駆動ドライバ	サーボテクノ	PMA6
AC サーボアクチュエータ	ハーモニックドライブ	RSF-5B-100-E050-C
減速機	ハーモニックドライブ	CSF
減速機付きモータ駆動ドライバ	ハーモニックドライブ	HA-680

A.2 減速機付き電磁モータ駆動部

駆動部にはハーモニックドライブ社製 AC サーボアクチュエータ RSF-5B-100-E050-C，減速機 CSF および駆動ドライバ HA-680 を使用した。ドライバとエンコーダはハーモニックドライブ社製

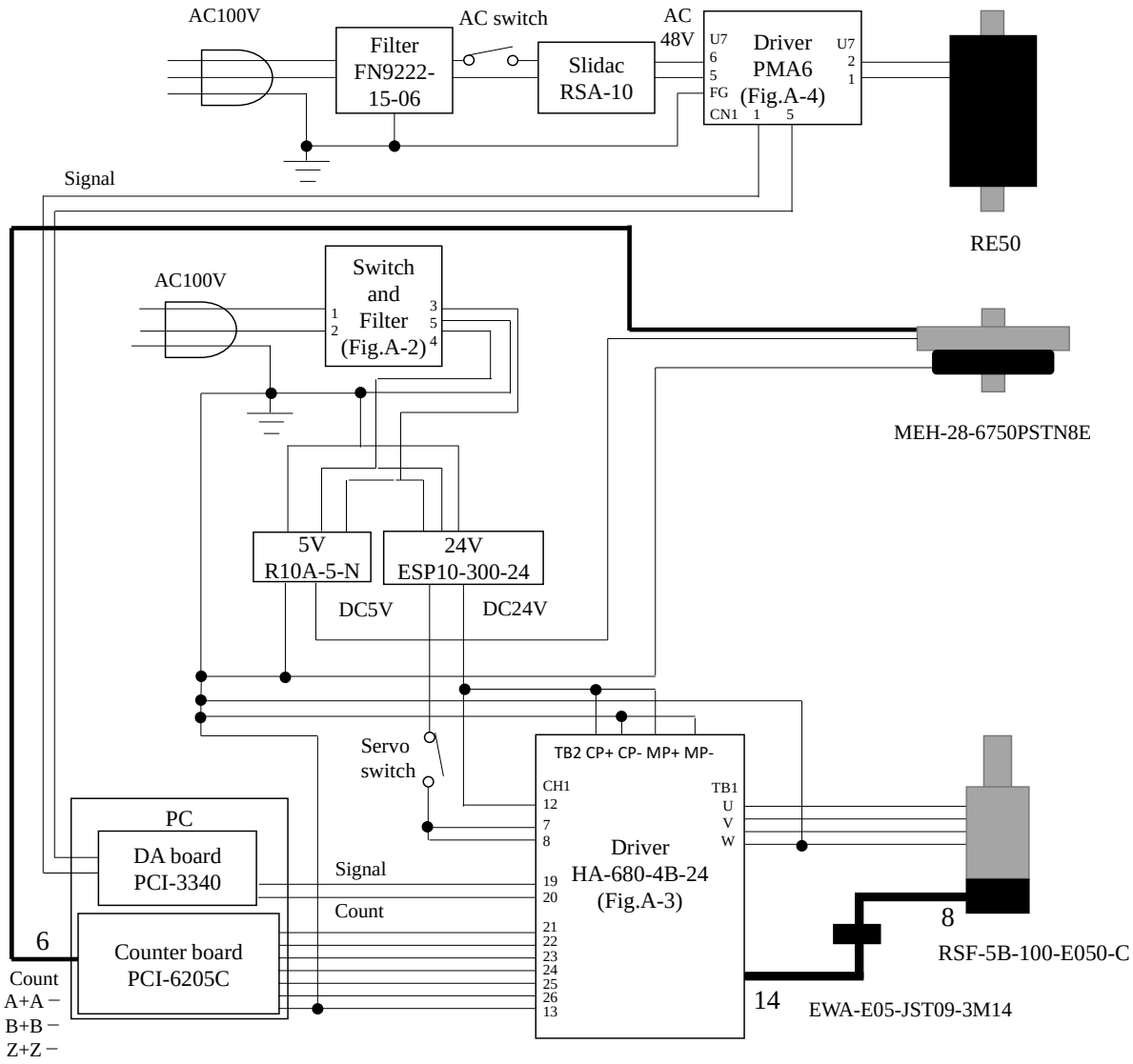


Fig. A-2: Experimental system

中軸ケーブル EWA-E05-JST09-3M14 でつながれている。ドライブ側はラインドライバ方式，エンコーダ側はオープンコレクタ方式であり途中で，中軸ゲーブル内で変換を行っている。HA-680 の接続関係を Fig. A-3 に示す。アクチュエータと減速機の仕様 Table A.2 に示す。このハーモニックドライブ製 RSF には Type ME のエンコーダが取り付けられており 500pulse/rev. の分解能である。この出力は減速機により 1/100 倍に減速されそして 4 逓倍したため減速機の出力軸での実質的な分解能は 200000pulse/rev. である。

Table A.2: Parameters of RSF

Nominal Voltage	24 V
Maximum continuous torque	0.44 mNm
Maximum continuous speed	100 rpm
Torque constant	0.011 Nm/A
Moment inertia	0.731 gcm^2

Table A.3: Parameters of RE50

Nominal Voltage	70 V
Maximum continuous torque	420 mNm
Maximum continuous speed	4620 rpm
Torque constant	93.4 mNm/A
Moment inertia	542 gcm^2

Table A.4: Parameters of PMA6

Rated voltage	88 V
Rated current	6.0 A
Maximum voltage	88 V
Maximum current	15.0 A
Input voltage	AC 16 V∼ 110 V or DC 20 V∼ 150 V
Transformation constant for current control	1.5 A/V

A.3 電磁モータ駆動部

環境の模擬とエンドエフェクタのエンコーダとしてマクソンモータ社製電磁モータ RE50 とマイクロテックラボラトリー社製エンコーダ MEH-28 を使用し，また駆動ドライバとしてサーボテクノ社製 PMA6 を駆動電力源として東京理工舎社製スライダック RSA-10 を使用した。PMA6 の接続関係を Fig. A-4 に示す。それぞれの仕様を Table A.3，Table A.4，Table C.1 に示す。スライダックは PMA6 の安全性を考慮し 48V の電圧を出力するように設定した。

◆ 接続例２　RSF supermini シリーズ

次の例は、「トルク制御」の接続例です。「パラメータ 11：入力機能割当」および、「パラメータ 12：出力機能割当」の設定値が「0」の場合です。お使いのアクチュエータで接続例が異なりますので注意してください。

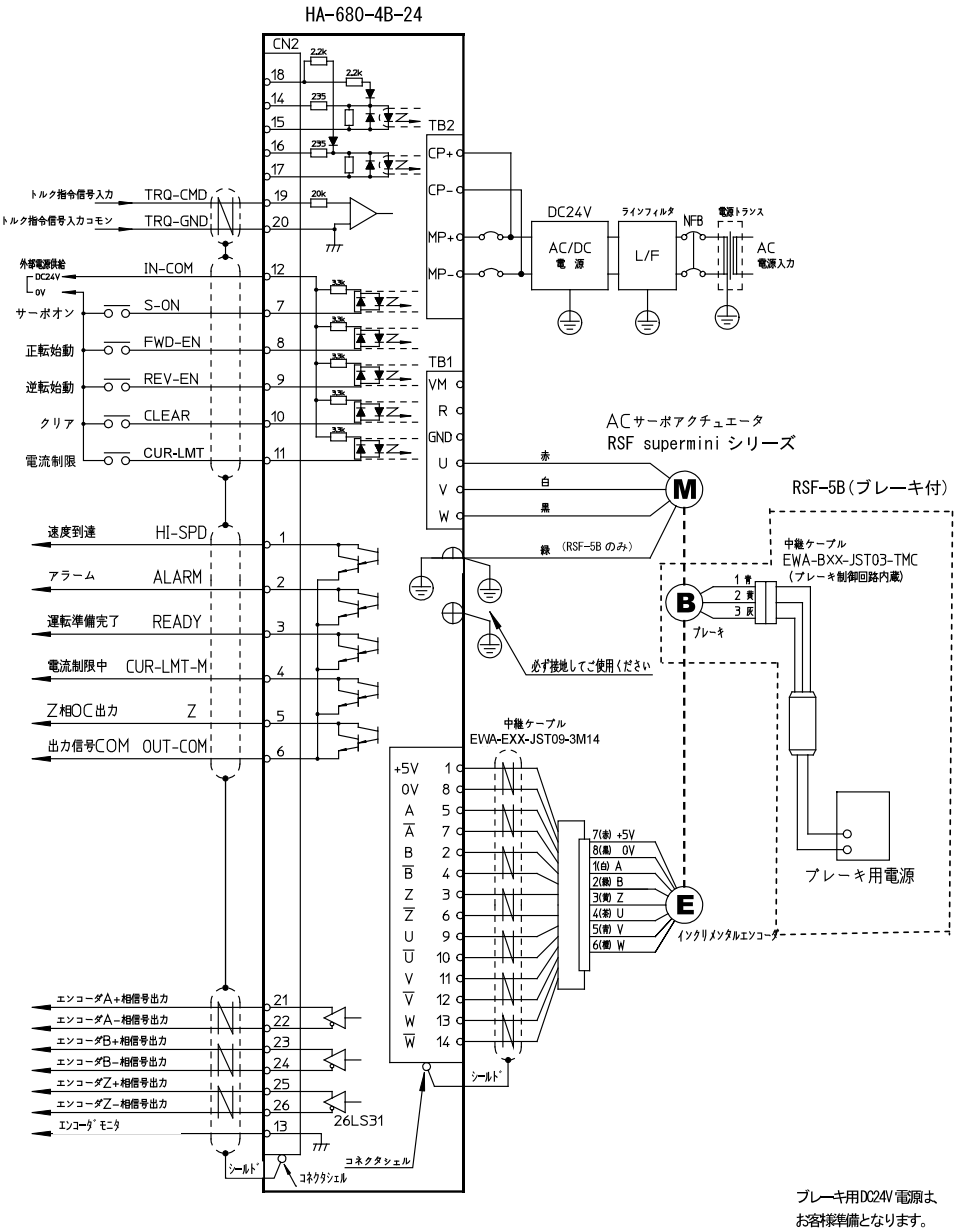
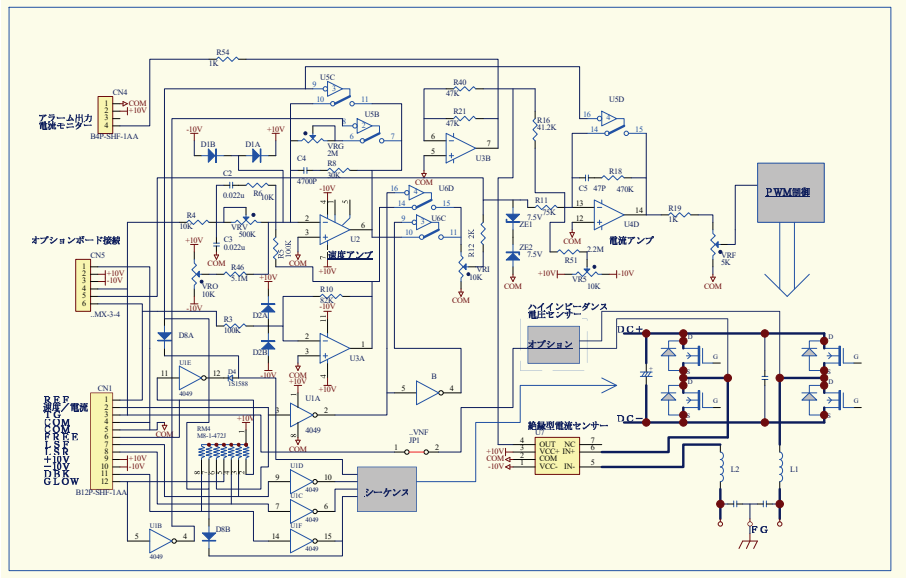


Fig. A-3: HA680

１１．演算回路図



１２．オプション各種

型名	オプション仕様	取付	説明
Ｖ	電圧制御	本体に部品追加	主回路とは、１００ＫΩのﾊｲｲﾝﾋﾟｰﾀﾞﾝｽ抵抗でつながります。タコゼネがなくて、ある程度速度可変したい場合に使用します。又、±出力の可変電源として使用できます。（外部リアクトル追加）
ＦＶＰＡ	エンコーダ フィードバック制御	本体にボード組込	エンコーダフィードバックにより速度制御をする場合に使用します。
ＰＡ	アナログ位置制御	“	ポテンショフィードバックにより、簡単な位置決め制御をする場合に使用します。
	電流制限入力		電流制限値を外部指令０～＋１０Ｖ入力することで変更できます。
	タコゼネモニタ出力		タコゼネ電圧を 1/4 にして出力します。21V→5. 2V
ＰＳ２	回生吸収ユニット	本体外部取付	回生エネルギーが大きくて、ドライブ内部で吸収できない場合に使用します。
ＰＬＬ制御	ＰＬ１（廃止品です 後継機種 LPV220）	本体にボード組込	エンコーダフィードバックＰＬＬ制御 パルス列入力（位置決め制御・精密速度制御）
ＩＳ	入力信号ｱｲｿﾚｰｼｮﾝ	“	ＶＥＬ／ＣＵＲ、ＦＲＥＥ、ＬＳＦ、ＬＳＲ、ＤＢＫ、ＧＬＯＷの各信号入力及びアラーム出力がフォトカプラにより絶縁されています。
＊＊	特別注文	“	仕様をお打ち合わせの上制作します。

Fig. A-4: PMA6

Appendix B

制御系の仕様

第 B.1 節でシミュレーションで用いたソースコードを示す。第 B.2 節で実験で用いたソースコードを示す。第 B.3 節で解析に用いたソースオードを示す。Fig. B-1 が用いたソースコードのディレクトリである。

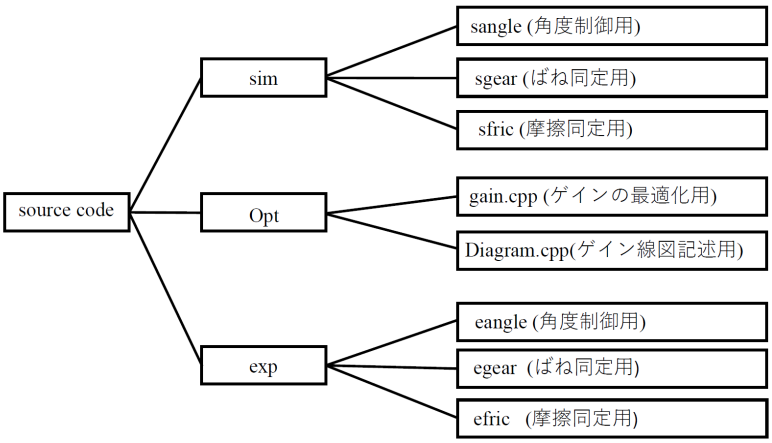


Fig. B-1: ソースコードのディレクトリ

B.1 シミュレーション

ばね定数の同定で用いたシミュレーションプログラムを Fig. B.1.1，駆動側摩擦の同定で用いたシミュレーションプログラムを Fig. B.1.1，負荷側角度制御で用いたシミュレーションプログラムを Fig. B.1.1 に示す。ディレクトリ構造は実験の節でシミュレーションと実験をまとめて示した。

B.1.1 ソースコード

ばね特性の同定

List B.1: c ファイル

```
1  //ギアの伝達誤差無しモデル
2
3  #include <stdio.h>
4  #include <math.h>
5  #include <string.h>
6
7  #define MAX(a,b) ((a)>(b)? (a):(b))   //最大値計算マクロ
8  #define MIN(a,b) ((a)>(b)? (b):(a))   //最小値計算マクロ
9  #define CLIP(a,b,c) (MIN(MAX(a,b),c)) //範囲内に丸めるマクロ
10
11
12 #define Mmn 0.0000000731      //モータ慣性力のノミナル値
13 #define Mm 0.0000000731       //モータ慣性力
14 #define Kt 0.011              //モータのトルク定数
15 #define Ktn 0.011                  //モータのトルク定数のノミナル値
16 #define tc 0.001                   //サンプリング値
17 #define Kv 700.0  //微分ゲイン
18 #define Kp 50000.0  //比例ゲイン
19 #define G 100.0    //カットオフ周波数
20 #define Kg 80.0           //ばね定数
21 #define N 100.0    //ギア比
22 #define Ml 0.0000531//負荷側の慣性
23 #define lbl 0.005                  //初期位置変えるときばね剛性の式も変える
24 #define rbl −0.002
25 #define tau_cm 0.00165  //駆動側静止摩擦
26 #define dm 0.00001 //駆動側粘性摩擦
27 #define taul_cm 0.00001 //負荷側静止摩擦
28 #define Dl 0.03  //負荷側粘性摩擦
29 #define sbl 0.00
30 #define A 0.3  //閾値
31 double x_cmd = 0.0;       //入力
32 double T = 0.0;
33 double taull;
34
35 struct joint {
36     double i_ref;    //電流指令値
37   double Xm;     //モータ位置
38   double Xm_dot;  //モータ速度
39   double Xm_hat;  //擬似微分のモータ速度
40   double Xm_dot_hat; //擬似微分のモータ加速度
41   double Xl;       //負荷位置
42   double Xl_dot;    //負荷速度
43   double tau_s;   //ねじれトルク
44   double tau_m;  //モータトルク
45   double tau_dis_hat; //外乱トルク
```

```
46    double integral_tau_dis_hat;
47    double im;      //モータ電流
48    double imr;
49    double imr_hat;
50    double tau_coul;
51    double deltax;
52    double dx_dot;
53    double dx_hat;
54    double dx;
55    double taul_coul;
56    double tau_l;
57    double deltax_dot;
58  };
59
60
61
62  void joint_init (struct joint *j) {
63      j->i_ref = 0.0;
64    j->Xm = 0.0;
65    j->Xm_dot = 0.0;
66    j->Xm_hat = 0.0;
67    j->Xm_dot_hat = 0.0;
68    j->Xl = 0.0;
69    j->Xl_dot = 0.0;
70      j->tau_s = 0.0;
71    j->tau_m = 0.0;
72    j->tau_dis_hat = 0.0;
73    j->integral_tau_dis_hat = 0.0;
74    j->im = 0.0;
75    j->imr = 0.0;
76    j->imr_hat = 0.0;
77    j->tau_coul = 0.0;
78    j->deltax = 0.0;
79    j->dx_dot = 0.0;
80    j->dx_hat = 0.0;
81    j->dx = 0.0;
82    j->taul_coul = 0.0;
83    j->tau_l = 0.0;
84    j->deltax_dot = 0.0;
85  };
86
87  double sat(double x)
88  {
89    if(x>1.0) return 1.0;
90    else if(x<-1.0) return -1.0;
91    else return x;
92  }
93
94  void dob(struct joint *j) {
95    j->imr_hat += (j->imr - j->imr_hat) * G * tc;
96    j->integral_tau_dis_hat += (Ktn * (j->imr_hat) + G * Mm * j->Xm_dot_hat - j->integral_tau_dis_hat) * G * tc;
97    j->tau_dis_hat = j->integral_tau_dis_hat - G * Mm * j->Xm_dot_hat;
98  }
99
100 int main (void){
101     FILE *date;
102     date = fopen("data.dat", "w");
103   struct joint joint1;
104    joint_init (&joint1);
105   for(double t = 0.0; t < 12.56; t += tc) {
106     joint1.deltax_dot = (x_cmd - joint1.Xm_hat) * G;
107     joint1.deltax += (joint1.deltax_dot) * tc;
108     joint1.im = (Kp * (x_cmd - joint1.Xm) + Kv * joint1.deltax_dot)*Mm/Ktn + joint1.tau_dis_hat / Ktn;
109     joint1.imr = CLIP(-0.76, joint1.im, 0.76);
```

```
    joint1.Xm_dot_hat = (joint1.Xm − joint1.Xm_hat) * G;
    joint1.Xm_hat += joint1.Xm_dot_hat * tc;
    dob(&joint1);
//////////////*機械モデル*////////////////////////////////////////////////////////////////////////////////////////////////////
    if(joint1.Xm/N−joint1.Xl>=rbl && joint1.Xm/N−joint1.Xl<=lbl) {
      joint1.tau_m = joint1.imr * Kt − joint1.Xm_dot * dm;
      joint1.tau_coul = tau_cm*sat((Mmn * joint1.Xm_dot / tc + joint1.tau_m) / tau_cm); //同上
        joint1.Xm_dot += ((joint1.tau_m − joint1.tau_coul) * tc) / Mmn;            //同上
    } else if(joint1.Xm/N−joint1.Xl>rbl || joint1.Xm/N−joint1.Xl<lbl) {
      joint1.tau_m = joint1.imr * Kt − joint1.Xm_dot * dm − joint1.tau_s/N;
      joint1.tau_coul = tau_cm*sat((Mmn * joint1.Xm_dot / tc + joint1.tau_m) / tau_cm); //同上
        joint1.Xm_dot += ((joint1.tau_m − joint1.tau_coul) * tc) / Mmn;            //同上
    }                                                //単位飽和関数の使用 菊植先生の論文使用 実際のモータ出力は理論式からモータのダンパとねじれトルクを引いた数値


    joint1.Xm += joint1.Xm_dot * tc;
      joint1.dx_dot=((joint1.Xm/N−joint1.Xl)−joint1.dx_hat)*G;
    joint1.dx_hat+=joint1.dx_dot*tc;
    joint1.dx = joint1.Xm/N−joint1.Xl;// − 0.0015*sin(0.5*t);

    if(joint1.dx>=rbl && joint1.dx<=lbl) {
      joint1.tau_s = (joint1.dx)*0.0;//+joint1.dx_dot*Dl;
    } else if(joint1.dx>lbl) {
      joint1.tau_s = Kg*(joint1.dx−lbl);//+1.0*joint1.dx_dot;
    } else if(joint1.dx<rbl) {
      joint1.tau_s = Kg*(joint1.dx−rbl);//+1.0*joint1.dx_dot;
    }
    //joint1.Xl_dot += ((joint1.tau_s−joint1.Xl_dot*Dl)/Ml)*tc;
    //joint1.Xl_dot += ((joint1.tau_s)/Ml)*tc;
    taull = A*sin(t);
    joint1.tau_l = joint1.tau_s − joint1.Xl_dot * Dl − taull;
    joint1.taul_coul = taul_cm*sat((Ml * joint1.Xl_dot / tc + joint1.tau_l) / taul_cm); //同上
        joint1.Xl_dot += ((joint1.tau_l − joint1.taul_coul) * tc) / Ml;             //同上
    joint1.Xl += joint1.Xl_dot*tc;
// ///////////////////////////////////////////////////////////////////////////////////////////////////////////////

    fprintf(date, "%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", t, x_cmd, joint1.Xm_dot_hat, joint1.tau_dis_hat, taull,
          joint1.Xm/N−joint1.Xl, joint1.Xl, joint1.Xm/N);
  }
}
//ばね特性のグラフは横軸joint1.Xm/N−joint1.Xl 縦軸 taull で確認できる。
//実験とのばね特性の比較は Kg、lbl を変化させながらシミュレーションと一致させる。
```

駆動側摩擦の同定

List B.2: c ファイル

```
//ギアの伝達誤差無しモデル

#include <stdio.h>
#include <math.h>
#include <string.h>

#define MAX(a,b) ((a)>(b)? (a):(b))   //最大値計算マクロ
#define MIN(a,b) ((a)>(b)? (b):(a))   //最小値計算マクロ
#define CLIP(a,b,c) (MIN(MAX(a,b),c)) //範囲内に丸めるマクロ
```

```
10
11
12 #define Mmn 0.0000000731      //モータ慣性力のノミナル値
13 #define Mm 0.0000000731       //モータ慣性力
14 #define Kt 0.011              //モータのトルク定数
15 #define Ktn 0.011                  //モータのトルク定数のノミナル値
16 #define tc 0.0002                   //サンプリング値
17 #define Kv 5000.0   //積分ゲイン
18 #define Kp 1200.0   //比例ゲイン
19 #define G 2.0    //カットオフ周波数
20 #define Kg 80.0   //ギアのばね定数
21 #define N 100.0 //ギア比
22 #define Ml 0.0000531 //負荷側の慣性
23 #define lbl 0.000                  //初期位置変えるときばね剛性の式も変える
24 #define rbl −0.008
25 //#define tau_cm 0.00165
26 //#define dm 0.00001
27 #define taul_cm 0.0037
28 #define Dl 0.00001
29 #define sbl 0.00
30 double v_cmd = 0.0;        //入力
31 double T = 0.0;
32 double dm;
33 double tau_cm;
34
35 struct joint {
36     double i_ref;    //電流指令値
37   double Xm;     //モータ位置
38   double Xm_dot;  //モータ速度
39   double Xm_hat;  //擬似微分のモータ速度
40   double Xm_dot_hat; //擬似微分のモータ加速度
41   double Xl;       //負荷位置
42   double Xl_dot;    //負荷速度
43   double tau_s;   //ねじれトルク
44   double tau_m;  //モータトルク
45   double tau_dis_hat; //外乱トルク
46   double integral_tau_dis_hat;
47   double im;       //モータ電流
48   double imr;
49   double imr_hat;
50   double tau_coul;
51   double deltax;
52   double dx_dot;
53   double dx_hat;
54   double dx;
55   double taul_coul;
56   double tau_l;
57 };
58
59
60
61 void joint_init (struct joint *j) {
62     j−>i_ref = 0.0;
63   j−>Xm = 0.0;
64   j−>Xm_dot = 0.0;
65   j−>Xm_hat = 0.0;
66   j−>Xm_dot_hat = 0.0;
67   j−>Xl = 0.0;
68   j−>Xl_dot = 0.0;
69     j−>tau_s = 0.0;
70   j−>tau_m = 0.0;
71   j−>tau_dis_hat = 0.0;
72   j−>integral_tau_dis_hat = 0.0;
73   j−>im = 0.0;
```

```
74    j->imr = 0.0;
75    j->imr_hat = 0.0;
76    j->tau_coul = 0.0;
77    j->deltax = 0.0;
78    j->dx_dot = 0.0;
79    j->dx_hat = 0.0;
80    j->dx = 0.0;
81    j->taul_coul = 0.0;
82    j->tau_l = 0.0;
83  };
84
85  double sat(double x)
86  {
87     if(x>1.0) return 1.0;
88    else if(x<-1.0) return -1.0;
89    else return x;
90  }
91
92  void dob(struct joint *j) {
93    j->imr_hat += (j->imr - j->imr_hat) * G * tc;
94    j->integral_tau_dis_hat += (Ktn * (j->imr_hat) + G * Mm * j->Xm_dot_hat - j->integral_tau_dis_hat) * G * tc;
95    j->tau_dis_hat = j->integral_tau_dis_hat - G * Mm * j->Xm_dot_hat;
96  }
97
98  int main (void){
99       FILE *date;
100      date = fopen("data.dat", "w");
101    struct joint  joint1 ;
102     joint_init (&joint1);
103    for(double t = 0.0; t < 50.0; t += tc) {
104
105       if(t<10.0) {
106         v_cmd=10.0*t;
107         tau_cm = 0.00165;
108         dm = 0.845* 1e-5;
109       } else if(t>=10.0 && t<20.0) {
110                 v_cmd=-10.0*t+200.0;
111         tau_cm = 0.00165;
112         dm = 0.845* 1e-5;
113       } else if(t>=20.0 && t<30.0) {
114                 v_cmd=-10.0*t+200.0;
115         tau_cm = 0.00223;
116         dm = 1.14* 1e-5;
117           } else if(t>=30.0 && t<40.0) {
118                 v_cmd=10.0*t-400.0;
119                 tau_cm = 0.00223;
120         dm = 1.14* 1e-5;
121           } else if(t>=40.0 && t<50.0) {
122         v_cmd=10.0*t-400.0;
123         tau_cm = 0.00165;
124         dm = 0.845* 1e-5;
125       }
126       joint1 .deltax += (v_cmd - joint1.Xm_dot_hat) * tc;
127       joint1 .im = (Kp * (v_cmd - joint1.Xm_dot_hat) + Kv * joint1.deltax)*Mm/Ktn;
128       joint1 .imr = CLIP(-0.76, joint1.im, 0.76);
129       joint1 .Xm_dot_hat = (joint1.Xm - joint1.Xm_hat) * G;
130       joint1 .Xm_hat += joint1.Xm_dot_hat * tc;
131       dob(&joint1);
132 //////////////*機械モデ
            ル*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

133       if( joint1 .Xm/N-joint1.Xl>=rbl && joint1.Xm/N-joint1.Xl<=lbl) {
134         joint1 .tau_m = joint1.imr * Kt - joint1.Xm_dot * dm;
135         joint1 .tau_coul = tau_cm*sat((Mmn * joint1.Xm_dot / tc + joint1.tau_m) / tau_cm); //同上
```

```
        joint1.Xm_dot += ((joint1.tau_m - joint1.tau_coul) * tc) / Mmn;          //同上
    } else if(joint1.Xm/N-joint1.Xl>rbl || joint1.Xm/N-joint1.Xl<lbl) {
      joint1.tau_m = joint1.imr * Kt - joint1.Xm_dot * dm - joint1.tau_s/N;
      joint1.tau_coul = tau_cm*sat((Mmn * joint1.Xm_dot / tc + joint1.tau_m) / tau_cm); //同上
        joint1.Xm_dot += ((joint1.tau_m - joint1.tau_coul) * tc) / Mmn;          //同上
    }                                                   //単位飽和関数の使用　菊植先生の論文使用　実際のモータ出力は理論式からモータのダンパとねじれトルクを引いた数値


    joint1.Xm += joint1.Xm_dot * tc;
      joint1.dx_dot=((joint1.Xm/N-joint1.Xl)-joint1.dx_hat)*G;
    joint1.dx_hat+=joint1.dx_dot*tc;
    joint1.dx = joint1.Xm/N-joint1.Xl - 0.00*sin(0.5*t);

    if(joint1.dx>=rbl && joint1.dx<=lbl) {
      joint1.tau_s = (joint1.dx)*0.0;//+joint1.dx_dot*Dl;
    } else if(joint1.dx>lbl) {
      joint1.tau_s = Kg*(joint1.dx-lbl);//+1.0*joint1.dx_dot;
    } else if(joint1.dx<rbl) {
      joint1.tau_s = Kg*(joint1.dx-rbl);//+1.0*joint1.dx_dot;
    }
    //joint1.Xl_dot += ((joint1.tau_s-joint1.Xl_dot*Dl)/Ml)*tc;
    //joint1.Xl_dot += ((joint1.tau_s)/Ml)*tc;
    joint1.tau_l = joint1.tau_s  - joint1.Xl_dot * Dl;
    joint1.taul_coul = taul_cm*sat((Ml * joint1.Xl_dot / tc + joint1.tau_l) / taul_cm); //同上
        joint1.Xl_dot += ((joint1.tau_l - joint1.taul_coul) * tc) / Ml;          //同上
    joint1.Xl += joint1.Xl_dot*tc;
// ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    fprintf(date, "%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", t, v_cmd, joint1.Xm_dot_hat, joint1.tau_dis_hat, joint1.imr_hat,  joint1.Xm/N-joint1.Xl, joint1.Xl, joint1.Xm/N);
  }
}

//摩擦特性のグラフは横軸joint1.Xm_dot_hat 縦軸 joint1.tau_dis_hat で見ることができる。
//実験から摩擦特性のパラメータを求めるときには tau_cm,taul_cm,Dl,dm の値を変えながらシミュレーションと実験値を一致させる。
```

負荷角度制御

List B.3: c ファイル

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define MAX(a,b) ((a)>(b)? (a):(b))   //最大値計算マクロ
#define MIN(a,b) ((a)>(b)? (b):(a))   //最小値計算マクロ
#define CLIP(a,b,c) (MIN(MAX(a,b),c)) //範囲内に丸めるマクロ


#define Mmn 0.0000000731      //モータ慣性力のノミナル値
#define Mm 0.0000000731       //モータ慣性力
#define Mm2 0.0000531
#define Kt 0.011              //モータのトルク定数
#define Ktn 0.011                      //モータのトルク定数のノミナル値
#define Ktl 0.0934
```

```
18 #define tc 0.001                    //サンプリング値
19 #define G 300.0
20 #define g2 42.0
21 #define GDOB 300.0
22 #define N 100.0
23 #define Nn 100.0
24 #define Ml 0.001//0.0007//0.0000531
25 ////////////////////////////////////////////////////////////
26 #define pbl 0.00//0.0010136 0.012635          //初期位置変えるときばね剛性の式も変える −0.009048 0.000254
27 #define nbl −0.009//−0.001064 //足したら 0.0020776
28 #define pbl2 0.00
29 #define nbl2 −0.009
30 #define A 1 //(dgree)
31 #define propZ//提案法と従来法を入れ替えられる
32 #define LDOB
33 #define s//s は正弦波、hは方形波 step はステップ
34 #define BL
35 #define kr 0.0// 5deg conv1 1.3
36 #define Cn 0.0//2.0//0.1
37 #define Cn2 0.0//0.1 0.05
38 #define k7 0.07
39
40 double Kg=4.0; //80/8/0.8
41 double Kgn=4.0; //プラントのばね定数
42 double ks2=80.0;
43 double kp=190000.0;//190000
44 double kd=3000.0;//3000
45 double k1=105.0; //105
46
47 double kpl=500.0;//1000.0;//1000;
48 double kdl=300.0;//700.0;//700;
49 double kpp = 10000.0;
50 double kdd = 20.0;
51 ////////////////////////////////////////////////////////////
52 double dm;
53 double tau_cm;
54 #define taul_cm 0.01
55 #define Dl 0.0000013
56 #define De 0.001
57 #define PLSm 2000.0
58 #define PLSl 216000.0
59 long long int cnt_m=0, cnt_l=0;
60 #define pv 15.0
61 #define ptau_cm 0.00165
62 #define pdm 1.345∗ 1e−5
63 #define ntau_cm 0.00223
64 #define ndm 1.845∗ 1e−5
65 #define tau_cl 0.01
66 #define dl 0.0000013
67 double nv=0.0;
68 #define vm 4.0
69 #define hvm 4.0
70 #define vl 1.0
71 #define Tmax 0.42
72
73 static double num=0.0, RMSE=0.0, RMSE_hat=0.0;
74 double e_x = 0.0;
75 double e_x_dot = 0.0;
76 double e_x_hat = 0.0;
77 double x_cmd = 0.0;
78 double x_cmd2 = 0.0;
79 double hipass_il = 0.0;
80
81 struct joint {
```

```
82      double i_ref;    //電流指令値
83    double Xm;     //モータ位置
84    double Xm_dot;  //モータ速度
85    double Xm_hat;  //擬似微分のモータ速度
86    double Xm_dot_hat; //擬似微分のモータ加速度
87    double Xl;       //負荷位置
88    double Xl_dot;    //負荷速度
89    double tau_s;   //ねじれトルク
90    double tau_m;  //モータトルク
91    double tau_dis_hat; //外乱トルク
92    double integral_tau_dis_hat;
93    double im;       //モータ電流
94    double imr;      //モータ電流リミッタ
95    double imr_hat;  //モータ電流推定値
96    double tau_coul; //駆動側の摩擦
97    double deltax;
98    double dx_dot;    //ねじれ角速度
99    double dx_hat;    //ねじれ角度推定値
100   double dx;       //ねじれ角度
101   double taul_coul; //負荷側の摩擦
102   double tau_l;   //負荷トルク
103   double hdelta_X;
104   double integral_tau_l_hat;//負荷トルク推定値
105   double Xl_dot_hat; //負荷角速度推定値
106   double Xl_hat;    //負荷角度推定値
107   double tau_e_hat;
108   double tau_e;
109   double pre_Xm_dot;
110   double tau_cmd_hat;
111   double Dq_ff, Dw_ff, iDw_ff, ltau_cmd, lltau_cmd, tau_error, itau_error, Dw_fb, w_mcmd, dw_mcmd, idw_mcmd,
              lw_mcmd;
112   double tau_tor_hat;
113   double imre_hat;
114   double integral_tau_e_hat;
115   double Xll, Xmm;//駆動側/負荷側角度
116   double hat_tau_cm;
117   double tau_s_hat;//ねじれトルク推定値
118   double tau_m_hat;//モータ入力トルク推定値
119   double tau_coul_hat;//駆動側摩擦推定値
120   double e_taul;
121   double Tz, Vb, Vc,fricm,integral_tau_s_hat2,fricm2, dll ;//仮想的なばねによる反力の導出
122   double tau_s_hat2,tau_s_hat3,tau_s_hat4,tau_s_hat_dot;//ねじれトルク推定値
123   double integral_taul_dis_hat , tau_l_hat ,tau_s_hatg;
124   double il, ilr , tl ;
125   //twin
126   double tau_m2,Xm2_dot,Xm2,dx2;
127
128 };
129
130
131
132 void joint_init (struct joint *j) {
133     j->i_ref = 0.0;
134   j->Xm = 0.0;
135   j->Xm_dot = 0.0;
136   j->Xm_hat = 0.0;
137   j->Xm_dot_hat = 0.0;
138   j->Xl = 0.0;
139   j->Xl_dot = 0.0;
140     j->tau_s = 0.0;
141   j->tau_m = 0.0;
142   j->tau_dis_hat = 0.0;
143   j->integral_tau_dis_hat = 0.0;
144   j->im = 0.0;
```

```
145     j->imr = 0.0;
146     j->imr_hat = 0.0;
147     j->tau_coul = 0.0;
148     j->deltax = 0.0;
149     j->dx_dot = 0.0;
150     j->dx_hat = 0.0;
151     j->dx = 0.0;
152     j->taul_coul = 0.0;
153     j->tau_l = 0.0;
154     j->hdelta_X= 0.0;
155     j->integral_tau_l_hat= 0.0;
156     j->Xl_dot_hat= 0.0;
157     j->Xl_hat= 0.0;
158     j->tau_e_hat= 0.0;
159     j->tau_e= 0.0;
160     j->pre_Xm_dot=0.0;
161     j->tau_cmd_hat = 0.0;
162     j->Dq_ff=0, j->Dw_ff=0, j->iDw_ff=0, j->ltau_cmd=0, j->lltau_cmd=0, j->tau_error=0, j->itau_error=0, j->
                Dw_fb=0, j->w_mcmd=0, j->dw_mcmd=0, j->idw_mcmd=0, j->lw_mcmd=0;
163     j->tau_tor_hat=0.0;
164     j->imre_hat=0.0;
165     j->integral_tau_e_hat=0.0;
166     j->Xmm=0.0, j->Xll=0.0;
167     j->hat_tau_cm=0.0;
168     j->tau_s_hat=0.0;
169     j->tau_m_hat=0.0;
170     j->tau_coul_hat=0.0;
171     j->e_taul=0.0;
172     j->Tz=0.0;
173     j->Vb=0.0, j->Vc=0.0;
174     j->fricm=0.0;
175     j->integral_tau_s_hat2=0.0;
176     j->fricm2=0.0;
177     j->dll=0.0;
178       j->tau_s_hat2=0.0;
179     j->tau_s_hat3=0.0;
180     j->tau_s_hat4=0.0;
181     j->tau_s_hat_dot = 0.0;
182     j->tau_l_hat = 0.0;
183     j->tau_s_hatg = 0.0;
184     j->integral_taul_dis_hat = 0.0;
185     j->il = 0.0, j->ilr = 0.0, j->tl = 0.0;
186
187     //twin
188     j->tau_m2 = 0.0;
189     j->Xm2_dot = 0.0;
190     j->Xm2 = 0.0;
191     j->dx2 = 0.0;
192
193 };
194
195 double sat(double x)
196 {
197     if(x>vm) return vm;
198     else if(x<-vm) return -vm;
199     else return x;
200 }
201
202 double sat1(double x)
203 {
204     if(x>vl) return 1.0;
205     else if(x<-vl) return -1.0;
206     else return x/vl;
207 }
```

```
208
209 double satm(double x)
210 {
211   if(x>hvm) return hvm;
212   else if(x<−hvm) return −hvm;
213   else return x;
214 }
215 ////////////////////DOB/////////////////////////////////////
216 void dob(struct joint ∗j) {// 卒論 (3.1)式
217   j−>imr_hat += (j−>imr − j−>imr_hat) ∗ GDOB ∗ tc;
218   j−>integral_tau_dis_hat += (Ktn ∗ (j−>imr_hat) + GDOB ∗ Mm ∗ j−>Xm_dot_hat − j−>integral_tau_dis_hat) ∗
            GDOB ∗ tc;
219   j−>tau_dis_hat = j−>integral_tau_dis_hat − GDOB ∗ Mm ∗ j−>Xm_dot_hat;
220
221 #ifdef LDOB
222     j−>tau_s_hatg += (j−>ilr−j−>tau_s_hatg)∗GDOB∗tc;
223   j−>integral_taul_dis_hat += (j−>tau_s_hatg ∗ Ktl + GDOB ∗ Mm2 ∗ j−>Xl_dot_hat − j−>integral_taul_dis_hat) ∗
            GDOB ∗ tc;
224   j−>tau_l_hat = (j−>integral_taul_dis_hat − GDOB ∗ Mm2 ∗ j−>Xl_dot_hat);
225   #endif
226
227
228
229 ////////////////ねじれトルクの推定/////////////////////////////////////////////////////////////////////
230   #ifdef conv1//従来法 2 卒論 (3.5)式
231   if(j−>Xm_dot_hat>=0.0){
232     tau_cm=ptau_cm;
233     dm=pdm;
234   } else if(j−>Xm_dot_hat<0.0){
235     tau_cm=ntau_cm;
236     dm=ndm;
237   }
238   j−>tau_m_hat = j−>imr_hat ∗ Kt;
239   j−>tau_coul_hat = tau_cm/hvm∗satm((Mmn ∗ j−>Xm_dot_hat / tc + j−>tau_m_hat) / tau_cm);
240   j−>fricm = j−>tau_coul_hat + 1.0∗dm ∗ j−>Xm_dot_hat;
241   j−>fricm2 += (j−>fricm−j−>fricm2) ∗ G ∗ tc;
242   //j−>fricm2 = j−>fricm;
243   j−>tau_s_hat=(j−>tau_dis_hat − j−>fricm2)∗N;
244   #endif
245 }
246
247 #ifdef propZ//提案法 卒論 (3.10)式
248 void wblz(struct joint ∗j) {
249   if(j−>dx>nbl2 && j−>dx<pbl2) {
250       j−>tau_s_hat = ((j−>dx)∗0.0+0.015∗j−>Vc);//∗(tanh(10∗(j−>dx−pbl2−nbl2)∗(j−>dx−pbl2−nbl2)));
251     } else if(j−>dx>=pbl2) {
252       j−>tau_s_hat = (Kgn∗(j−>dx−pbl2)+0.015∗j−>Vc);//∗tanh(16000∗((j−>dx)−pbl2)∗((j−>dx)−pbl2));
253     } else if(j−>dx<=nbl2) {
254       j−>tau_s_hat = (Kgn∗(j−>dx−nbl2)+0.015∗j−>Vc);//∗(tanh(16000∗(j−>dx−nbl2)∗(j−>dx−nbl2)));
255     }
256     //ねじれトルク微分
257     j−>tau_s_hat4 += j−>tau_s_hat_dot ∗ tc;
258     j−>tau_s_hat_dot = (j−>tau_s_hat − j−>tau_s_hat4)∗G;
259     //ハイパス
260     j−>tau_s_hat3 += j−>tau_s_hat2 ∗ g2 ∗ tc;
261     j−>tau_s_hat2 = j−>tau_s_hat − j−>tau_s_hat3;
262
263 }
264
265 #endif
266
267 #ifdef conv2//従来法 3 卒論 (3.8)式
268 void wobl(struct joint ∗j) {
269   j−>tau_s_hat = Kgn∗(j−>dx−nbl);//+0.1∗j−>Vc;
```

```
270     j->tau_s_hat3 += j->tau_s_hat2 * g * tc;
271     j->tau_s_hat2 += (j->tau_s_hat-j->tau_s_hat2) * g * tc;
272 }
273 #endif
274
275
276 ////////////////Bl 振動抑制/////////////////////////////////////////////////////////////////////////
277 #ifdef BL//卒論 (3.11)式
278 void wz(struct joint *j) {
279   j->Vc=((j->Xm/N-j->Xl)-j->Vb)*G;
280   j->Vb+=j->Vc*tc;
281   j->Tz +=(Cn*(j->Xm/N-j->Xl - pbl) - j->Tz) * G * tc;
282   j->Tz = (j->Xm/N-j->Xl - pbl) * Cn;
283 }
284 #endif
285
286 #ifdef BLv//卒論 (3.11)式
287 void wb(struct joint *j) {
288   j->Vc=((j->Xm/N-j->Xl)-j->Vb)*G;
289   j->Vb+=j->Vc*tc;
290   j->Tz =Cn * j->Vc;
291 }
292 #endif
293 ////////////////////////////////////////////////////////////////////////////////////////////////
294
295 int main (void){
296   FILE *date;
297   struct joint  joint1 ;
298    joint_init (&joint1);
299   date = fopen("data.dat", "w");
300   nv=ntau_cm/(ndm-(pdm*pv+ptau_cm)/pv);
301     for(double t = -0.5; t < 4.0; t += tc) {
302 //////////////*制御
              器*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

303     //エンコーダ
304     cnt_m = (long long)(joint1.Xmm*PLSm/(2*M_PI));
305     joint1 .Xm = (double)(cnt_m*2*M_PI/PLSm);
306     cnt_l  = (long long)(joint1.Xll*PLSl/(2*M_PI));
307     joint1 .Xl = (double)(cnt_l*2*M_PI/PLSl);
308     //角速度推定, オブザーバ
309     joint1 .Xm_dot_hat = (joint1.Xm - joint1.Xm_hat) * G;
310     joint1 .Xm_hat += joint1.Xm_dot_hat * tc;
311       joint1 .Xl_dot_hat = (joint1 .Xl - joint1 .Xl_hat) * G;
312     joint1 .Xl_hat += joint1.Xl_dot_hat * tc;
313     dob(&joint1);
314     #ifdef prop
315     wbl(&joint1);
316         #endif
317     #ifdef propZ
318     wblz(&joint1);
319         #endif
320     #ifdef conv2
321     wobl(&joint1);
322         #endif
323     #ifdef BL
324     wz(&joint1);
325         #endif
326     #ifdef BLv
327     wb(&joint1);
328         #endif
329     #ifdef step//ステップ指令値
330     if  (t  < 0) {
331       joint1 .tau_s_hat2= 0;
```

```
332     }
333   if (t>=0) {
334     x_cmd2 = M_PI*A/180.0;
335     } else {
336       x_cmd2 = 0;
337       joint1.tau_s_hat=0.0;
338       joint1.Tz=0.0;
339     }
340     if (t <= 0.01) {
341       x_cmd = x_cmd2 / 0.01 * t;
342     } else if (t >= 0.01 && t <= 4) {
343       x_cmd = x_cmd2;
344     }
345     #endif
346     #ifdef h//方形波指令値
347     if (t < 0) {
348       joint1.tau_s_hat2= 0;
349     }
350   if (t>=0) {
351     x_cmd2 = M_PI*A/180.0;
352     } else {
353       x_cmd2 = 0;
354       joint1.tau_s_hat=0.0;
355       joint1.Tz=0.0;
356     }
357     if (t <= 0.01) {
358       x_cmd = x_cmd2 / 0.01 * t;
359     } else if (t >= 0.01 && t <= 2) {
360       x_cmd = x_cmd2;
361     } else if (t >= 2 && t <= 2.01) {
362       x_cmd = -x_cmd2 / 0.01 * (t - 2.00) + x_cmd2;
363     } else {
364       x_cmd = 0;
365     }
366     #endif
367     #ifdef s//正弦波指令値
368     if (t<=0) {
369       x_cmd = 0;
370       joint1.tau_s_hat=0.0;
371       joint1.Tz=0.0;
372     } else {
373     x_cmd = -2*M_PI*A/180.0 + M_PI*A/180.0*sin(2*3.1415*t);
374     }
375     #endif
376
377     e_x = x_cmd - joint1.Xl;
378     e_x_dot = (e_x - e_x_hat) * G;
379     e_x_hat += e_x_dot * tc;
380
381
382     joint1.im = Mm/Ktn*(kp*e_x+kd*e_x_dot-k1*joint1.Xm_dot_hat)-kr*(joint1.tau_s_hat2)-joint1.Tz+(joint1.
                tau_dis_hat)/Ktn; //駆動側モータ入力電流
                 卒論 (3.3)式
383     joint1.imr = CLIP(-0.76, joint1.im,0.76); //リミッタ
384
385     //ねじれ角度 FB
386     //joint1.il = -Mm2/Ktl*(kpl*-Mm2/Ktl*(kpl*(joint1.Xm/N - joint1.Xl) + kdl*(joint1.Vc));// - joint1.tau_l_hat
                /Ktl;
387
388     //BL モデル
389     joint1.il = -Mm2/Ktl*(kpp*e_x+kdd*e_x_dot+kpl*joint1.tau_s_hat + kdl * joint1.tau_s_hat_dot) - k7*(joint1.Xm/
                N - joint1.Xl);
390     //joint1.il = -Mm2/Ktl*(kpl*joint1.tau_s_hat + kdl * joint1.tau_s_hat_dot) - k7*(joint1.Xm/N - joint1.Xl);
391     // hipass_il += (joint1.il - hipass_il) * g2 * tc;
```

```
392     joint1 . ilr  = CLIP(−0.76, joint1.il ,0.76);
393 //
            ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
394
395
396 //////////////*機械モデ
            ル*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
397
398      //Motor1
399     if(joint1 .Xmm/N−joint1.Xll>=nbl && joint1.Xmm/N−joint1.Xll<=pbl) {//駆動側の摩擦特性 卒論 (2.2)式
400       if(joint1 .Xm_dot>=0.0){
401         tau_cm=ptau_cm;
402         dm=pdm;
403       } else if(joint1 .Xm_dot<0.0){
404         tau_cm=ntau_cm;
405         dm=ndm;
406       }
407       joint1 .tau_m = joint1.imr ∗ Kt;
408       joint1 .tau_coul = tau_cm/vm∗sat((Mmn ∗ joint1.Xm_dot / tc + joint1.tau_m) / tau_cm); //同上
409         joint1 .Xm_dot += ((joint1.tau_m − joint1.Xm_dot ∗ dm− joint1.tau_coul ) ∗ tc) / Mmn;
410     } else if(joint1 .Xmm/N−joint1.Xll<nbl || joint1.Xmm/N−joint1.Xll>pbl) {
411       if(joint1 .Xm_dot>=0.0){
412         tau_cm=ptau_cm;
413         dm=pdm;
414       } else if(joint1 .Xm_dot<0.0){
415         tau_cm=ntau_cm;
416         dm=ndm;
417       }
418       joint1 .tau_m = joint1.imr ∗ Kt ;
419       joint1 .tau_coul = tau_cm/vm∗sat((Mmn ∗ joint1.Xm_dot / tc + joint1.tau_m) / tau_cm); //同上
420         joint1 .Xm_dot += ((joint1.tau_m− joint1.Xm_dot ∗ dm − joint1.tau_s/N− joint1.tau_coul) ∗ tc) / Mmn;      //
                同
                上
421     }
422
423
424     joint1 .Xmm += joint1.Xm_dot ∗ tc;
425       joint1 .dx_dot=((joint1.Xmm/N−joint1.Xll)−joint1.dx_hat)∗G;
426     joint1 .dx_hat+=joint1.dx_dot∗tc;
427     joint1 .dx = joint1.Xmm/N−joint1.Xll;// − 0.0015∗sin(0.5∗t);
428
429     //Motor2
430     joint1 .tau_m2 = joint1. ilr  ∗ Ktl;
431     joint1 . taul_coul  = taul_cm∗sat1((Mm2 ∗ joint1.Xm2_dot / tc + joint1.tau_m2) / taul_cm);
432     joint1 .Xm2_dot += ((joint1.tau_m2− joint1.Xm2_dot ∗ Dl − joint1.tl− joint1.taul_coul) ∗ tc) / Mm2;
433     joint1 .Xm2 += joint1.Xm2_dot ∗ tc;
434     joint1 .dx2 = joint1.Xm2 − joint1.Xl;
435     joint1 . tl  = ks2 ∗ joint1 .dx2;
436
437
438
439     if (t <= 2 && t >= 0) {//負荷トルク指令値
440       joint1 . dll= −0.1∗tanh(2∗t);
441     }
442     if (t >= 2) {
443       joint1 . dll= −0.1;//∗tanh(30∗t);//0.01∗tanh(20∗t)−0.005∗tanh(20∗t);
444     }
445
446     if(joint1 .dx>=nbl && joint1.dx<=pbl) {//バックラッシュを含んだばね特性 卒論 (2.5)式
447       joint1 .tau_s  = (joint1 .dx)∗0.0+0.015∗joint1.dx_dot;
448     } else if(joint1 .dx>pbl) {
449       joint1 .tau_s  = Kgn∗(joint1.dx−pbl)+0.015∗joint1.dx_dot;
```

```
    } else if(joint1.dx<nbl) {
      joint1.tau_s = Kgn*(joint1.dx-nbl)+0.015*joint1.dx_dot;
    }
    joint1.tau_l = joint1.tau_s - joint1.dll - joint1.tl;
  //同上 卒論 (2.6)式
        joint1.Xl_dot += ((joint1.tau_l - joint1.taul_coul) * tc) / Ml;              //同上
    joint1.Xll += joint1.Xl_dot*tc;
//  ////////////////////////////////////////////////////////////////////////////////////////////////////////////

     fprintf(date, "%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", t/*1*/, x_cmd/M_PI*180.0 /*2*/, joint1.Xll/M_PI*180.0 /*3*/, joint1.tau_s/*4*/, joint1.tau_s_hat2/*5*/, joint1.tau_s_hat/*6*/,(joint1.Xmm/N-pbl)/M_PI*180.0/*7*/, joint1.ilr/*8*/,joint1.Xm2/M_PI*180.0/*9*/,joint1.dx2 * ks2/*10*/,joint1.imr/*11*/);
    //fprintf(date, "%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", t/*1*/, Mm*(kp*e_x+kd*e_x_dot)/*2*/, Mm*(k1*joint1.Xm_dot_hat) /*3*/, kr*(joint1.tau_s_hat2)*Ktn/*4*/, joint1.Tz*Ktn/*5*/, (joint1.tau_dis_hat)/*6*/,-Mm2*(kpl*joint1.tau_s_hat + kdl * joint1.tau_s_hat_dot)/*7*/, k7*(joint1.Xm/N - joint1.Xl)*Ktl/*8*/,joint1.tau_l_hat/*9*/,joint1.Xl_dot * Dl + joint1.dll + joint1.ilr/*10*/);
  // plot"data.dat" using 1:2 with lines linewidth 7 title"(k_1+k_2s)e","data.dat" using 1:3 with lines linewidth 7 title"k_3s{/Symbol q}_m","data.dat" using 1:6 with lines linewidth 7 title"{/Symbol t}_{dis}","data.dat" using 1:7 with lines linewidth 7 title"(k_6+k_7s){/Symbol t}_s","data.dat" using 1:8 with lines linewidth 7 title"k_8{/Symbol D}{/Symbol q}"
  //plot"data.dat" using 1:2 with lines linewidth 6 title"(k_1+k_2s)e_x","data.dat" using 1:3 with lines linewidth 6 dt 3 title"k_3s{/Symbol q}_{m1}","data.dat" using 1:6 with lines linewidth 6 title"{/Symbol t}_{dis}","data.dat" using 1:7 with lines linewidth 6 dt 3 title"(k_6+k_7s){/Symbol t}","data.dat" using 1:8 with lines linewidth 6 dt 3 title"k_8({/Symbol q}_{m1}/n-{/Symbol q}_l)"
        //plot"data.dat" using 1:2 with lines linewidth 6 title"(k_1+k_2s)e_x","data.dat" using 1:3 with lines linewidth 6 dt 3 title"k_3s{/Symbol q}_{m1}","data.dat" using 1:6 with lines linewidth 6 dt 3 title"{/Symbol t}_{dis}","data.dat" using 1:4 with lines linewidth 6 title"k_4{/Symbol t}_{s1}","data.dat" using 1:5 with lines linewidth 6 dt 3 title"k_5({/Symbol q}_{m1}/n-{/Symbol q}_l)"
}
}
//////////////////////////////シミュレーションで変更するパラメータについて////////////////////////////////////////
//負荷角度指令値を変更する場合は27行目の「#define h」で方形波,「 #define s」で正弦波にでき,25行目のA の値を変更することで振幅を変更できる。
//従来法, 提案法の切り替えは 26行目の「#define conv1」で従来法1, 29 行目の[k4=1.64]で従来法2,「#define conv2」かつ[k4=1.64]で従来法3,「#define prop」かつ[k4=1.64]かつ [Cn=0.6]で提案法にできる
//制御ゲインの変更は 34,35行目から変更可能
//ばね定数を変えたい場合は 32,33行目を変更することで可能
//モータの摩擦特性が変わった場合は 40〜52行目のパラメータを変更する。
```

B.2　実験

Fig. B.2.1, Fig. B.2.2, Fig. B.2.3 に実験で用いたソースコードを示す。

B.2.1　ばね特性同定のソースコード

Gear/user.h

List B.4: h ファイル

```
//実験モード
//#define USMtendon //USM 腱駆動機構の実験
//#define USMsea //USM の SEA の実験
#define HDsea  //ハーモニックドライブの SEA の実験
```

```
5  //#define TEST  //電圧指令直接入力テスト
6
7  //各種定数
8  #define ST  0.001   //control period
9
10 //カウンタボード (24bit)
11 #define COUNT_OVER 0x1000000
12 #define COUNT_MAX 0xffffff //24bit
13 #define Low_limit 216000  //low value limit for encoder
14 #define Max_limit (0xffffff−216000) //max value limit for encoder
15
16 //DA ボード(16bit)
17 //レンジ共通
18 #define BIT0V  32768  //0V のときのビット
19 //5V レンジ用
20 #define BIT5V  65535  //5V のときのビット
21 #define BIT3V2 53738  //3.2V のときのビット
22 #define BIT_5  6553   //1V あたりのビット
23 //10V レンジ用
24 #define BIT10V 65535  //10V のときのビット
25 #define BIT_10 3277   //1V あたりのビット
26
27 //駆動側モータ
28 #define Imax  0.76    //駆動側モータの定格電流[A]
29 #define I_Vhd  0.0798   //駆動側モータドライバの指令電圧から電流への変換係数[A/V]
30 #define Ktm  0.011  //モータのトルク定数[Nm/A]
31 #define N  100.0  //ギア比
32 #define Mm  0.0000000731 //モータの慣性モーメント [kgmˆ2]
33 #define Dm  0.0000124 //モータの粘性摩擦係数 [Nms/rad]
34 #define Tcm  0.0015 //モータのクーロン摩擦トルク [Nm]
35 //負荷モータ (maxon)
36 #define Tmax  0.42    //負荷モータの定格トルク[Nm]
37 #define Ktl   0.0934      //負荷モータトルク定数[Nm/A]
38 #define I_Vemm  1.5     //負荷モータドライバの指令電圧から電流への変換係数[A/V]
39
40 #define ONE_SEC_IN_NANO 1000000000
41
42 //物理定数
43 #define Ks  4.4   //ねじりばね定数[Nm/rad]
44 #define Ml  0.0000542 //慣性モーメント [kgmˆ2]
45 #define Ke  4.0   //環境弾性[Nm/rad]
46
47 //制御器パラメータ
48 #define DT  0.001   //制御周期[s]
49 #define PLSm  2000.0   //モータ側エンコーダ分解能[P/R]
50 #define PLSl  216000.0  //負荷側エンコーダ分解能[P/R]
51 #define G  400.0  //カットオフ周波数[rad/s]
52 //#define Vcmdd 100.0  //速度指令値 [rad/s] 10毎に 10˜100までマイナス方向も
53 #define Kp  50000.0   //モータ P ゲイン
54 #define Kd  700.0
55 #define qcmd 0.0
56
57
58
59 struct motor{
60   unsigned long prev_count;
61   int over_24bit;
62 };
```

Gear/user.c

List B.5: c ファイル

```
1     //負荷モータへの指令
2     /*if(q_l>0.0){
3       t_l = Ke*q_l;
4       if(t_l>Tmax) t_l=Tmax;
5       if(t_l<−Tmax) t_l=−Tmax;
6     }else t_l=0.0;*/
7     t_l=0.3*sin(T);//負荷トルク指令値
8     if(t_l>Tmax) {
9     t_l=Tmax;
10    }
11    if(t_l<−Tmax) {
12    t_l=−Tmax;
13    }
14    i_emm = t_l/Ktl; //電磁モータに流したい電流を算出
15    v_emm = i_emm/I_Vemm; //ドライバに送る指令電圧を算出
16    T += 0.001;
17 //ばね特性のグラフは横軸ねじれ角度 q_m/N−q_l 縦軸負荷トルク t_l で確認できる
```

Gear/Makefile

List B.6: make ファイル

```
1  CC      = gcc −lrt
2  CFLAGS = −O0 −I/usr/src/linux −I/usr/realtime/include −I/usr/local/include/opencv
3  LDFLAGS = −L/usr/lib −L/root/OpenCV−2.3.0/lib
4  LDLIBS = −lgpg6204 −lpthread −lgpg3300
5
6  all: user
7
8  user: user.o
9    $(CC) −o user $(LDFLAGS) user.o $(LDLIBS)
10
11 user.o: user.c user.h
12   $(CC) −o user.o $(CFLAGS) −Wall −c user.c
13
14 clean:
15   rm −f *.o user
```

B.2.2 駆動側摩擦同定のソースコード

Friction/user.h

List B.7: h ファイル

```
1  //実験モード
2  //#define USMtendon //USM 腱駆動機構の実験
3  //#define USMsea //USM の SEA の実験
4  #define HDsea //ハーモニックドライブの SEA の実験
5  //#define TEST //電圧指令直接入力テスト
6
7  //各種定数
8  #define ST 0.0002 //control period
9
10 //カウンタボード (24bit)
11 #define COUNT_OVER 0x1000000
12 #define COUNT_MAX 0xffffff //24bit
13 #define Low_limit 216000 //low value limit for encoder
14 #define Max_limit (0xffffff−216000) //max value limit for encoder
15
16 //DA ボード(16bit)
17 //レンジ共通
18 #define BIT0V 32768 //0V のときのビット
19 //5V レンジ用
20 #define BIT5V 65535 //5V のときのビット
21 #define BIT3V2 53738 //3.2V のときのビット
22 #define BIT_5 6553 //1V あたりのビット
23 //10V レンジ用
24 #define BIT10V 65535 //10V のときのビット
25 #define BIT_10 3277 //1V あたりのビット
26
27 //駆動側モータ
28 #define Imax 0.76 //駆動側モータの定格電流[A]
29 #define I_Vhd 0.0798 //駆動側モータドライバの指令電圧から電流への変換係数[A/V]
30 #define Ktm 0.011 //モータのトルク定数[Nm/A]
31 #define N 100.0 //ギア比
32 #define Mm 0.0000000731 //モータの慣性モーメント [kgmˆ2]
33 #define Dm 0.0000124 //モータの粘性摩擦係数 [Nms/rad]
34 #define Tcm 0.0015 //モータのクーロン摩擦トルク [Nm]
35 //負荷モータ (maxon)
36 #define Tmax 0.42 //負荷モータの定格トルク[Nm]
37 #define Ktl 0.0934 //負荷モータトルク定数[Nm/A]
38 #define I_Vemm 1.5 //負荷モータドライバの指令電圧から電流への変換係数[A/V]
39
40 #define ONE_SEC_IN_NANO 1000000000
41
42 //物理定数
43 #define Ks 4.4 //ねじりばね定数[Nm/rad]
44 #define Ml 0.0000542 //慣性モーメント [kgmˆ2]
45 #define Ke 4.0 //環境弾性[Nm/rad]
46
47 //制御器パラメータ
48 #define DT 0.0002 //制御周期[s]
49 #define PLSm 2000.0 //モータ側エンコーダ分解能[P/R]
50 #define PLSl 216000.0 //負荷側エンコーダ分解能[P/R]
51 #define G 2.0 //カットオフ周波数[rad/s]
52 //#define Vcmdd 100.0 //速度指令値 [rad/s] 10毎に 10˜100までマイナス方向も
53 #define Kp 1200.0 //モータ P ゲイン
54 #define Ki 5000.0
55 #define Kpp 50.0 //モータ P ゲイン
56 #define Kii 50.0
57 #define qcmd 180.0
58
59
```

```
60
61 struct motor{
62   unsigned long prev_count;
63   int over_24bit;
64 };
```

Friction/user.c

List B.8: c ファイル

```
1       //DOB
2       li_m = LPF(i_m, &fol_li_m, G);
3       iht_dis = li_m*Ktm + G * Mm * w_m;
4       lht_dis = LPF(iht_dis, &fol_lht_dis, G);
5       ht_dis = lht_dis - G * Mm * w_m;
6       //lw_m =LPF(w_m, &fol_lw_m, G);
7       //ht_dis = lht_dis - G * Mm * lw_m;
8       //モータに流れる電流
9       if(T<10.0) {
10        Vcmd=10.0*T;
11      } else if(T>=10.0 && T<30.0) {
12              Vcmd=-10.0*T+200.0;
13        } else if(T>=30.0) {
14        Vcmd=10.0*T-400.0;
15      }
16      T+=0.0002;
17      dX += (Vcmd - w_m) * DT;
18      i_m = (Kp*(Vcmd-w_m)+Ki*dX)*Mm/Ktm;
19 //摩擦特性のグラフは横軸 w_m 縦軸 ht_dis で見ることができる
```

Friction/Makefile

List B.9: make ファイル

```
1  CC      = gcc -lrt
2  CFLAGS = -O0 -I/usr/src/linux -I/usr/realtime/include -I/usr/local/include/opencv
3  LDFLAGS = -L/usr/lib -L/root/OpenCV-2.3.0/lib
4  LDLIBS = -lgpg6204 -lpthread -lgpg3300
5
6  all: user
7
8  user: user.o
9    $(CC) -o user $(LDFLAGS) user.o $(LDLIBS)
10
11 user.o: user.c user.h
12   $(CC) -o user.o $(CFLAGS) -Wall -c user.c
13
14 clean:
15   rm -f *.o user
```

B.2.3 負荷角度制御のソースコード

Angle/user.c

List B.10: c ファイル

```
/************************************************************
  user.c

  Processor: 2
  OS: LINUX 2.6.32
  Structure^^c2^^a1 §
    soft  real−time −> user commnad (main)
              −> packet−receiving (thread1)
    hard real−time −> packet−sending and motion control (thread2)

  User interface for control module in user space
  Bilateral Control

 ************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/types.h>
#include "user.h"
#define FILE_NAME "file.dat"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>
#include <fcntl.h>
#include <sys/file.h>
#include <signal.h>
#include <errno.h>
#define ECHOMAX 255
#include <pthread.h>
#include <sched.h>
#include <time.h>
#include <math.h>

#include "system.h"
#include "matrix.h"
#include "controller.h"
#include "parallel_dynamics.h"
#include "Command.h"
#include "filter.h"

#include "PCI−3120.h"
#include "PCI−3340.h"
#include "PCI−6205C.h"
//#include "peripheral.h"

#include <iostream>

//従来法1、従来法2はすべてコメントアウト、提案法は doublemotor のみコメントアウトを外す
//#define loadtorque
#define doublemotor
//#define doublemotorconv

#define prop

//command//ここで負荷側角度指令値の波形を変更する
//#define h
#define sin
//#define step

using std::cout;
```

```
63  using std::endl;
64
65  unsigned long int cnt_self=0;
66  unsigned long int cnt_self_old=0;
67  unsigned long int cnt_other=0;
68  int user_input=0;
69  double t=0.0;
70  long int delay=0;
71  int nRet=0;
72
73  int  interval_est =100;
74  int  queue_first_net2usr =0;
75  int  queue_last_net2usr=0;
76  int  flag [2];
77
78  struct timeval s_time = {0}, e_time = {0}, dtime = {0};//^^c5^^93^^c3^^a8^^c3^^8d^^c3^^bd^^c2^^bb^^c3^^be^^c5^^
        bd^^c3^^96^^c2^^b7 ×^^c3^^82¬^^c3^^8d^^c3^^91
79
80
81  FILE *debug_fp;
82  double bat_vol = 0;
83
84  pthread_mutex_t mutex;
85
86  void Vout_convert(const double min,const double max,const double i_ref[6],double V_out[6]){
87    V_out[7] = 1*CLIP(-10.0,i_ref[0]/0.0798, 10.0);
88    V_out[1] = 1*CLIP(-0.5,i_ref[1]/1.5, 0.5);
89    //V_out[theta_3] = -1*CLIP(min,i_ref[theta_3], max);
90    //V_out[theta_4] = 1*CLIP(min,i_ref[theta_4], max);
91    //V_out[theta_5] = -1*CLIP(min,i_ref[theta_5], max);
92    //V_out[theta_6] = 1*CLIP(min,i_ref[theta_6], max);
93  }
94
95  void DieWithError(char *errorMessage)
96  {
97    perror(errorMessage);
98    exit (1);
99  }
100
101
102 void sigcatch(int sig){
103   if(sig==SIGALRM){
104     printf("catch_SIGALRM_and_exit.\n");
105     exit (1);
106   }
107 }
108
109
110 //////////Packet Receiving//////////
111 void *thread1 (void *arg1){
112
113   return 0;
114 }
115
116
117 //////////Packet Sending and Motion Control//////////
118 void *thread2 (void *arg2){
119
120   //#define USE_BDOB
121   //#define USE_RFOB
122
123   struct timespec NextTime = {0};
124   struct timeval StartTime = {0};
125   struct timeval StartTimePrev = {0};
```

```
126  struct timeval ActPeriodTime = {0};
127  struct timespec PeriodTime = {0};
128  struct timeval EndTime = {0};
129  struct timeval CompTime = {0};
130  struct timespec TimeInWait = {0};
131  struct timespec PreventStuck = {0};
132  struct timeval c_time = {0}, c_time_prev = {0}, c_period = {0};//^^c3^^80^^c2^^a9^^c5^^be^^c3^^a6^^c5^^92^^c3^^
             be^^c5^^bd^^c3^^bc^^c2^^b7 ×^^c3^^82¬
133  struct timeval stime = {0}, etime = {0}, dtime = {0};//^^c2^^b7 ×^^c3^^82¬^^c3^^8d^^c3^^91
134  double m_time = 0;
135  double c_time_usec = 0;
136  int cnt = 0;
137  double time = 0;//t^^e2^^82^^ac^^c3^^88^^c3^^86±^^e2^^82^^ac^^c5^^be
138
139  ///// parallel_control ///////
140  int i,j;
141  int motor_out =0; //¥^^c3^^a2^^c2^^a1^^c5^^92¥^^c2^^bf^^c5^^93^^c3^^90^^c3^^8e^^c3^^8f^^c2^^b5^^c3^^b6^^c2
             ^^b2^^c3^^84
142  int error = 0;
143  double V_out[6]={0};
144  double V_out_Zero[6]={0}; //output zero voltage
145  double ad_voltage[16]={0};
146  static long count[6]={0};
147
148  //response vector
149  const double p_0[6]={0.0, 0.0, 0.147570, 0.0*d2r, 0.0*d2r, 0.0*d2r};
150  double p[6]={p_0[x], p_0[y],p_0[z], p_0[ roll ], p_0[pitch ], p_0[yaw]};
151  double dp[6]={0};
152
153  double theta[6]={(45.0*d2r),(135.0*d2r),(135.0*d2r),(0.0*d2r),(0.0*d2r),(45.0*d2r)};
154  double theta_prev[6]={(45.0*d2r),(135.0*d2r),(135.0*d2r),(0.0*d2r),(0.0*d2r),(45.0*d2r)};
155  double theta_n[6]={(0.0*d2r),(135.0*d2r),(135.0*d2r),(0.0*d2r),(0.0*d2r),(45.0*d2r)};
156  double hat_theta[6]={(45.0*d2r),(135.0*d2r),(135.0*d2r),(0.0*d2r),(0.0*d2r),(45.0*d2r)};
157
158  double dtheta[6]={0},dtheta_n[6]={0};
159  double dtheta_j[6]={0};
160
161
162
163  double F_ref[6]={0},i_ref[6]={0}, i_ref_i [6]={0};
164
165  //2dof parameter
166  double theta2[2] = {0};
167  double theta3[2] = {0};
168  double dtheta2[2] = {0};
169  double dtheta3[2] = {0};
170
171        //single
172         double Tmax = 0.76;
173         double t_l = 0.0;
174         double Ktl = 0.0934;
175         double Ml = 0.0000542;
176
177         double tau_s_hat = 0.0;
178         double tau_s_hatd = 0.0;
179         double tau_s_hath = 0.0;
180    //制御器ゲインの変更
181         double k1 = 190000.0;//100000 190000 250000
182         double k2 = 3000.0;//300 3000 900
183         double k3 = 105.0;//60 105 150
184         double k4 = 0.0;//1.2 0.6
185         double k5 = 0.0;//0.07 0.0
186
187         double k6 = 500.0;//500.0;
```

```
188         double k7 = 0.0;//300.0;//300.0;
189         double k8 = 0.07;//0.07;
190
191         double k9 = 3000.0;//3000.0
192         double k10 = 150.0;//150.0
193         double k11 = 10.0;//10.0
194
195         double k12 = 10000;
196         double k13 = 20;
197
198
199         double g2 = 0.0;//42.0 0.0
200         double delta = 0.0;
201         double ql_ref = 0.0;
202         double ql_ref2 = 1.0 * 3.141592/180;//負荷側角度指令値の振幅を変更できる。
203         double Mm = 0.0000000731;
204         double iht_dis = 0.0;
205         double lht_dis = 0.0;
206         double ht_dis = 0.0;
207         double w_m = 0.0;
208         double w_m2 = 0.0;
209         double w_l = 0.0;
210         double w_l2 = 0.0;
211         double G = 300.0;
212         double ktm = 0.011;
213         double q_m = 0.0;
214         double q_l = 0.0;
215         double n = 100.0;
216         double li_m = 0.0;
217         double z = 0.0;
218         double qcmd = 0.0;
219         double dX = 0.0;
220         double dX2 = 0.0;
221         double kp = 3000.0;
222         double ki = 100.0;
223         double pbl = 0.0;
224         double nbl = 0.0;
225         double lbl = 0.009;
226         double de = 0.0;
227         double de2 = 0.0;
228         double qt = 0.0;
229         double re = 0.0;
230         double ql_error = 0.0;
231         double dx = 0.0;
232         double dx2 = 0.0;
233         double delta3 = 0.0;
234         double delta2 = 0.0;
235         double ks = 3.0;
236         double dn = 0.02;
237         double tau_s_hat2 = 0.0;
238         double tau_s_hat3 = 0.0;
239         double times = 0.0;
240         double e_p = 0.0;
241         double e_d = 0.0;
242         double e_d2 = 0.0;
243 ARCS::PCI3120 ADboard1( 0x2120, ARCS::PCI3120::RANGE_U_10V);
244 ARCS::PCI3340 DAboard2( 0x2000);
245
246 DAboard2.SetVoltage(V_out_Zero);
247 //ARCS::PCI6205C CNTboard1(0x20e0, 0x20c0, 0x20a0, 0x2080,6,true);
248 ARCS::PCI6205C CNTboard2(0x2080, 0x2060, 0x2040, 0x2020,6,true);
249 Dynamics parallel(theta,p_0);
250 using namespace ARCS;
251
```

```
252   //start_force();//^^c3^^8e^^c3^^8f¥^^c2^^bb¥^^c3^^b3¥^^c2^^b5^^c3^^82¬^^c3^^84^^c3^^aa^^c2^^b3^^c2^^ab^^
          c2^^bb^^c3^^8f
253   FILE *data;
254   data= fopen("data.dat","w");
255
256   gettimeofday(&c_time_prev, NULL);//^^c5^^93^^c3^^a9^^c2^^b2^^c3^^b3^^e2^^82^^ac^^c3^^80^^e2^^82^^ac ± prev
          ^^e2^^82^^ac^^c3^^8e^^c2^^bb^^c3^^be^^c5^^bd^^c3^^96^^c5^^92^^c3^^a8^^c3^^86^^c3^^80
257
258   clock_gettime(CLOCK_MONOTONIC, &NextTime);
259
260
261   while(user_input == 0){
262
263     gettimeofday(&StartTime, NULL);
264     timersub(&StartTime, &StartTimePrev, &ActPeriodTime);
265     StartTimePrev = StartTime;
266
267   time = t;
268
269   motor_out = 1; //^^c5^^93^^c3^^90^^c3^^8e^^c3^^8f^^c2^^b5^^c3^^b6^^c2^^b2^^c3^^84
270
271   //ADboard1.ConvStart();                    // AD^^c3^^8a^^c3^^91^^c5^^bd^^c2^^b9^^c2^^b3^^c2^^ab^^c2^^bb^^c3
          ^^8f
272   //ADboard1.WaitBusy();                     // AD^^c3^^8a^^c3^^91^^c5^^bd^^c2^^b9^^e2^^82^^ac ¬^^c5^^bd°^^c3
          ^^8e^^c2^^bb^^e2^^82^^ac^^c2^^b9^^e2^^82^^ac^^c3^^ab^^e2^^82^^ac^^c3^^9e^^e2^^82^^ac^^c3^^87^^c3
          ^^82^^c3^^94^^c2^^b5^^c2^^a1(¥^^c3^^96¥^^c3^^ad¥^^c3^^83¥^^c2^^ad¥^^c3^^b3¥°^^c3^^86° ^^c2^^
          ba^^c3^^ae)
273   //ADboard1.GetVoltage(6,ad_voltage);
274   CNTboard2.ZpulseClear(false);  //Z^^c3^^81^^c3^^98^^c2^^bb^^c3^^88^^c3^^8d^^c3^^91^^c2^^b5^^c3^^b6^^c2^^b2
          ^^c3^^84^^e2^^82^^ac^^c2^^b7^^e2^^82^^ac^^c3^^8a^^e2^^82^^ac^^e2^^82^^ac^^c2^^a1^^c2^^a1true:^^c2
          ^^bb^^c3^^88^^c3^^8d^^c3^^91^^c2^^a1^^c2^^a1false:^^c3^^89^^c3^^94^^c2^^bb^^c3^^88^^c3^^8d^^c3^^91
275   CNTboard2.GetCount(count);   //¥^^c5^^a1¥^^c3^^b3¥^^c2^^b3^^c2^^a1^^c5^^92¥^^c3^^80¥^^c3^^91¥^^c3^^
          ab¥^^c2^^b9^^e2^^82^^ac^^c3^^b2^^c5^^92^^c3^^a8^^c3^^86^^c3^^80
276
277 ////////////////////////////////////////////////////////////////////////////////////////////////////
278   //^^c3^^80^^c2^^a9^^c5^^be^^c3^^a6^^c2^^b3^^c2^^ab^^c2^^bb^^c3^^8f//
279   //^^c2^^b7 ×^^c3^^82¬^^c2^^b3^^c2^^ab^^c2^^bb^^c3^^8f//
280   gettimeofday(&s_time, NULL);//^^c2^^b7 ×^^c3^^82¬¥^^c2^^b9¥^^c2^^bf^^c2^^a1^^c5^^92¥^^c3^^88
281   theta_n[theta_1] = count[0]*motor_pulse;//motor side angle
282   theta_n[theta_2] = count[theta_2]*encoder_pulse;
283   theta_n[theta_3] = -count[theta_3]*encoder_pulse;
284   theta_n[theta_4] = -count[theta_4]*encoder_pulse;
285   theta_n[theta_5] = count[theta_5]*encoder_pulse;
286   theta_n[theta_6] = -count[1]*encoder_pulse;//load side angle
287    diff(DOF,gdiff,theta_n,dtheta,theta);
288         q_m = count[0]*motor_pulse;
289         q_l = -count[1]*encoder_pulse;
290         w_m = (q_m - w_m2) * G;
291         w_m2 += w_m * ST;
292         w_l = (q_l - w_l2) * G;
293         w_l2 += w_l * ST;
294
295 ////////////////////////////////////////angle controll program
296 ////////////////////////////////////
297   theta2[0]=theta[0];
298   theta2[1]=theta[1];
299   theta3[0]=theta[3];
300   theta3[1]=theta[2];
301   dtheta2[0]=dtheta[0];
302   dtheta2[1]=dtheta[1];
303   dtheta3[0]=dtheta[3];
304   dtheta3[1]=dtheta[2];
305
306
```

```
#ifdef loadtorque
    if (t >= 31 && t <= 33) {
        t_l = −0.02∗tanh(20∗(t−31));
    } else if (t>= 33 && t<= 35) {
        t_l= −0.02∗tanh(20∗(t−31));
    } else {
        t_l = 0.0;
     }

    if( t_l>Tmax) {
  t_l=Tmax;
     }
    if( t_l<−Tmax) {
  t_l=−Tmax;
     }

    i_ref [1] = t_l/Ktl;

#endif

#ifdef doublemotor
    if (t >= 31 && t <= 35) {
        i_ref [1] = (−k12 ∗ ql_error − k13 ∗ dx−k6 ∗ tau_s_hat − k7 ∗ tau_s_hatd) ∗ Ml/Ktl − k8 ∗ (delta − pbl);
        if ( i_ref [1] > 0.76) {
            i_ref [1] = 0.76;
        } else if ( i_ref [1] <−0.76) {
            i_ref [1] = −0.76;
         }
    }
#endif

#ifdef doublemotor
    if (t >= 31 && t <= 35) {
        e_p = ql_error − nbl;//
        e_d = G ∗ (e_p − e_d2);
        e_d2 += e_d ∗ ST;
        // i_ref [1] = (−k9 ∗ e_p − k10 ∗ e_d + k11 ∗ q_l) ∗ Ml/Ktl;
        if ( i_ref [1] > 0.76) {
            i_ref [1] = 0.76;
        } else if ( i_ref [1] <−0.76) {
            i_ref [1] = −0.76;
         }
    }
#endif

//////motor side///////////////
#ifdef step
    if (t>= 31.00 && t <= 31.02) {
        ql_ref = ql_ref2 / 0.02 ∗ (t−31.00);
    } else if (t >= 31.02 && t <= 40.00) {
        ql_ref = ql_ref2;
    } else {
        ql_ref = 3.141592/180 ∗ 0.0;
    }
#endif

#ifdef h
    if (t>= 31.00 && t <= 31.02) {
        ql_ref = ql_ref2 / 0.02 ∗ (t−31.00);
    } else if (t >= 31.02 && t <= 33.00) {
        ql_ref = ql_ref2;
    } else if (t >= 33.00 && t <= 33.02) {
        ql_ref = −ql_ref2 / 0.02 ∗ (t − 33.00) + ql_ref2;
    } else {
```

```
371         ql_ref  = 3.141592/180 * 0.0;
372     }
373 #endif
374
375 #ifdef sin
376     if (t>=31.00) {
377         ql_ref  = −2.0 * 3.141592/180 + ql_ref2 * cos(1.0*3.141592*1*times − 3.141592/2);
378          times += ST;
379     } else {
380         ql_ref  = 0.0;
381     }
382 #endif
383
384 //BL
385
386     if (t <= 30) {
387        li_m += (i_ref[0] − li_m) * G * ST;
388   iht_dis  = li_m*ktm + G * Mm * w_m;
389         lht_dis  += (iht_dis − lht_dis) * G * ST;
390   ht_dis = lht_dis − G * Mm * w_m;
391         if (z == 0 && q_l <= 0.004) {
392       qcmd = 0.08*t;
393       dX = (qcmd − dX2) * G;
394       dX2 += dX *0.001;
395        i_ref [0]  = (kp*(qcmd − q_m)+ki*dX)*Mm/ktm+ht_dis/ktm;
396         }
397         if (q_l >= 0.004 && z == 0) {
398             if (pbl == 0.0) {
399         de = q_m / n − q_l;
400         z = 1.0;
401         qt = t;
402                 printf("end_Phase1\n");
403             }
404         }
405         if (z == 1.0 && q_l > −0.004) {
406         qcmd = −0.18 *(t−qt) + 0.1 * qt;
407         dX = (qcmd − dX2) * G;
408         dX2 += dX * 0.001;
409          i_ref [0]  = (kp*(qcmd − q_m)+ki*dX)*Mm/ktm+ht_dis/ktm;
410         }
411         if (z == 1.0 && q_l <= −0.004) {
412         de2 = q_m/n − q_l;
413         //pbl = 0;
414         pbl = (lbl + de + de2) / 2;
415         nbl = pbl − lbl;
416         z =2.0;
417         qt = t;
418               printf("end_Phase2\n");
419         }
420         if (z == 2.0 && q_l <= 0) {
421         qcmd = 0.005*(t − qt);
422         dX = (qcmd − dX2) * G;
423         dX2 += dX * 0.001;
424               i_ref [0]  = (kp*(qcmd − q_m)+ki*dX)*Mm/ktm+ht_dis/ktm;
425         } else if (q_l >= 0 && z == 2.0) {
426          i_ref [0]  = 0;
427         z = 3;
428         re = 1.0;
429             printf("end_BL\n");
430         }
431    }
432 #ifdef prop
433     if (t > 30 && re == 1.0) {
434         if ((q_m/n − pbl− q_l) > 0) {
```

```
435         tau_s_hat =(ks * (q_m/n - q_l - pbl) + dn * delta2);
436          }else if ((q_m/n-q_l - pbl)< -lbl) {
437         tau_s_hat =(ks * (q_m/n - q_l - nbl) + dn * delta2);
438     } else {
439         tau_s_hat =(0.0 + dn * delta2);
440     }
441         tau_s_hat2 += tau_s_hath * g2 * ST;
442         tau_s_hath = tau_s_hat - tau_s_hat2;
443               tau_s_hatd = (tau_s_hat - tau_s_hat3) * G;
444               tau_s_hat3 += tau_s_hatd * ST;
445   }
446
447 #endif
448
449      if (t >= 31) {
450          //DOB
451          li_m += (i_ref[0] - li_m) * G * ST;
452     iht_dis = li_m*ktm + G * Mm * w_m;
453          lht_dis += (iht_dis - lht_dis) * G * ST;
454     ht_dis = lht_dis - G * Mm * w_m;
455          ql_error = ql_ref - q_l;
456          dx = (ql_error - dx2)*G;
457          dx2 += dx * ST;
458          delta = q_m/n-q_l;
459          delta2 = (delta - delta3) * G;
460          delta3 += delta2 *ST;
461          i_ref [0] = (k1 * ql_error + k2 * dx - k3 *w_m) * Mm/ktm + ht_dis/ktm -k4 * tau_s_hath - k5 * (delta - pbl
                    );
462      }
463  //i_ref [0] = 0;
464
465   Vout_convert(-10.0,10.0, i_ref ,V_out);
466   DAboard2.SetVoltage(V_out);
467   gettimeofday(&e_time, NULL);//^^c2^^b7 ×^^c3^^82¬^^c5^^93^^c2^^aa^^c3^^8e^^c2^^bb
468   //^^c3^^80^^c2^^a9^^c5^^be^^c3^^a6^^c5^^93^^c2^^aa^^c3^^8e^^c2^^bb//
469 //
          ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

470   gettimeofday(&c_time, NULL);//^^c2^^bb^^c3^^be^^c5^^bd^^c3^^96^^c5^^92^^c3^^a8^^c3^^86^^c3^^80
471   timersub( &c_time, &c_time_prev, &c_period);
472   c_time_prev = c_time;
473   c_time_usec = c_period.tv_usec;
474   m_time = (c_time.tv_sec + c_time.tv_usec * 1e-6);//^^c5^^b8^^c2^^ae^^c2^^bf^^c3^^b4^^c3^^8a^^c3^^91^^c5^^bd^^
              c2^^b9
475
476   cnt++;
477   timersub(&e_time, &s_time, &dtime);
478
479
480 //printf
481    if( cnt_self %(int)(0.1/ST)==0){
482    printf("time:%.2f_", t );
483    //printf("e:%d ", error ); printf(" ");
484    //for(i=0;i<3;i++){printf("%.1f ", 100*p_cmd[i]);}for(i=3;i<DOF;i++){printf("%.1f ", p_cmd[i]/d2r);}printf(" ");
485    //printf("p :"); for(i=0;i<3;i++){printf("%.1f ", 100*p[i]);} for(i=3;i<DOF;i++){printf("%.3f ", p[i]/
486          printf("%.3f_%.3f_%.3f_%.5f_%.3f_%.3f_", ql_ref/d2r,q_m/n/d2r,q_l/d2r,pbl,tau_s_hat,tau_s_hath);
487          printf("%.3f_%.3f", i_ref [0], i_ref [1]);
488
489    printf("\n");
490      }
491
492      NextTime.tv_nsec = NextTime.tv_nsec + PeriodTime.tv_nsec;
493      PeriodTime.tv_nsec = ST*ONE_SEC_IN_NANO; //100us
```

```
494     if(NextTime.tv_nsec >= ONE_SEC_IN_NANO){ // tv_nsec^^e2^^82^^ac ¬ 1^^c3^^89^^c3^^83^^e2^^82^^ac^^c3^^
                b2^^c3^^84¶^^e2^^82^^ac^^c5^^a1^^e2^^82^^ac^^c2^^bf^^e2^^82^^ac^^c3^^a9^^c2^^a1^^e2^^82^^ac
495       NextTime.tv_nsec −= ONE_SEC_IN_NANO; // tv_nsec^^e2^^82^^ac^^c2^^ab^^e2^^82^^ac^^c3^^a91^^c3^^89^^c3
                ^^83^^c3^^8a ¬^^e2^^82^^ac^^c3^^8e¥^^c3^^8a¥^^c3^^8e^^c3^^89^^c3^^83^^c2^^bf^^c3^^b4^^e2
                ^^82^^ac^^c3^^b2°^^c3^^ba^^e2^^82^^ac^^e2^^82^^ac^^e2^^82^^ac^^c3^^86^^c2^^a1^^e2^^82^^ac
496           NextTime.tv_sec++;                    // ^^e2^^82^^ac^^c5^^93^^e2^^82^^ac^^c3^^8e^^c3^^82^^c3^^a5^^e2
                ^^82^^ac^^c3^^af^^e2^^82^^ac^^c3^^aa^^e2^^82^^ac^^c3^^8btv_sec^^e2^^82^^ac^^c3^^8b1^^c3
                ^^89^^c3^^83^^c3^^8a ¬^^c3^^84^^c3^^89^^c2^^b2^^c3^^83
497 }
498
499     gettimeofday(&EndTime, NULL);              // ^^c5^^93^^c2^^aa^^c3^^8e^^c2^^bb^^c2^^bb^^c3^^be^^c2^^b9^^c3
                ^^af^^e2^^82^^ac^^c3^^8e^^c5^^92^^c3^^a8^^c3^^86^^c3^^80
500     timersub(&EndTime, &StartTime, &CompTime); // ^^c5^^b8^^c3^^83^^c3^^88^^c3^^b1^^c2^^bb^^c3^^be^^c5^^bd
                ^^c3^^96^^e2^^82^^ac^^c3^^b2^^c2^^b7 ×^^c2^^bb^^c2^^bb(timeval^^c2^^b9^^c5^^93^^c3^^82^^e2
                ^^82^^ac^^c3^^82^^c3^^8e^^e2^^82^^ac^^c3^^8f^^c3^^83±^^c5^^93^^c3^^a3^^e2^^82^^ac^^c3^^8b^^c5^^
                be^^c2^^ba^^c2^^bb^^c2^^bb^^e2^^82^^ac^^c3^^87^^e2^^82^^ac^^c2^^ad^^e2^^82^^ac^^c3^^8a^^e2
                ^^82^^ac^^e2^^82^^ac^^e2^^82^^ac^^c2^^b3^^e2^^82^^ac^^c3^^88^^e2^^82^^ac^^c3^^8b^^c3^^83^^c3^^
                ad°^^c3^^95)
501
502 //fprintf
503     if( cnt_self %(int)(0.01/ST)==0){
504     fprintf (data, "%f␣%f␣", t, c_time_usec);   //1, 2
505           fprintf (data, "%.3f␣%.3f␣%.3f␣%.3f␣%.3f␣",ql_ref/d2r,q_m/n/d2r,q_l/d2r,tau_s_hat,tau_s_hath);
506           fprintf (data, "%.3f␣%.3f", i_ref [0], i_ref [1]);
507
508     if(CompTime.tv_usec > ST * 1e6){
509     fprintf (data, "␣!␣");
510     }
511     fprintf (data, "\n");
512     }
513
514     pthread_mutex_lock(&mutex);
515     cnt_other=net2usr.data[0];
516      cnt_self_old =net2usr.data[1];
517     delay=cnt_self−cnt_self_old ;
518      cnt_self ++;
519     t=cnt_self*ST;
520     pthread_mutex_unlock(&mutex);
521
522       while(user_input==0){
523         clock_gettime(CLOCK_MONOTONIC, &TimeInWait); // ^^c5^^be^^c5^^93^^c2^^ba^^c3^^9f^^c2^^bb^^c3^^be^^
                  c2^^b9^^c3^^af^^e2^^82^^ac^^c3^^8e^^c5^^92^^c3^^a8^^c3^^86^^c3^^80
524         if(NextTime.tv_sec <= TimeInWait.tv_sec && NextTime.tv_nsec <= TimeInWait.tv_nsec) break;
525         pthread_testcancel ();     // ¥^^c2^^b9¥^^c3^^ac¥^^c3^^83¥^^c3^^89^^e2^^82^^ac^^c3^^8e¥^^c2^^ad¥^^c3^^
                  a3¥^^c3^^b3¥^^c2^^bb¥^^c3^^ab^^e2^^82^^ac ¬^^c3^^81÷^^e2^^82^^ac^^c3^^a9^^e2^^82^^ac^^c3
                  ^^ac^^e2^^82^^ac^^c3^^86^^e2^^82^^ac^^e2^^82^^ac^^e2^^82^^ac^^c3^^ab^^e2^^82^^ac^^c2^^ab^^c3
                  ^^87° ^^e2^^82^^ac^^c3^^8e^^e2^^82^^ac^^c2^^bf^^e2^^82^^ac^^c3^^a1^^c2^^b3^^c3^^8e^^c3^^87§
526         asm("nop");
527       }
528       clock_nanosleep(CLOCK_MONOTONIC, 0, &PreventStuck, NULL);
529     }
530     DAboard2.SetVoltage(V_out_Zero);
531      fclose (data);
532     //end_force (); //^^c3^^8e^^c3^^8f¥^^c2^^bb¥^^c3^^b3¥^^c2^^b5^^c5^^93^^c2^^aa^^c3^^8e^^c2^^bb
533     return 0;
534 }
535
536 //////////Main//////////
537 int main (void){
538
539     struct timespec PreventStuck = {0};
540      printf("Main_is_created.\n");
541     debug_fp = fopen("debug_data.dat","w");
```

```
542   //^^c3^^84^^c3^^89^^c2^^b2^^c3^^83^^c2^^a1^^c2^^a1DA^^e2^^82^^ac^^c3^^8e¥^^c3^^87¥^^c5^^be¥^^c2^^bf¥^^
                c3^^ab^^c5^^93^^c3^^90^^c3^^8e^^c3^^8f¥^^c3^^9d^^c2^^a1^^c5^^92¥^^c3^^88^^e2^^82^^ac^^c3^^8e^^c5
                ^^93^^c3^^a9^^c5^^bd^^c3^^bc^^c5^^b8^^c3^^b5^^c3^^82^^c3^^96^^e2^^82^^ac^^c3^^b20^^e2^^82^^ac^^
                c3^^8b^^c2^^b7^^c3^^a8^^c3^^84^^c3^^aa^^e2^^82^^ac^^c2^^b9^^e2^^82^^ac^^c3^^ab
543   usleep(500000);//^^c3^^83^^c3^^99±^^c3^^a4 500000
544
545   //DaClose(nDevice_da);
546   pthread_mutex_init(&mutex, NULL);
547   if(SIG_ERR==signal(SIGALRM, sigcatch)){
548     printf(" failed to set signal handler.\n");
549   }
550
551   struct sched_param ThreadParam;
552   ThreadParam.sched_priority = sched_get_priority_max(SCHED_FIFO);
553   pthread_t th1;
554   pthread_t th2;
555   pthread_create( &th1, NULL, thread1, NULL ); //for packet sending
556   pthread_create( &th2, NULL, thread2, NULL ); //for controller
557   pthread_setschedparam(th1, SCHED_FIFO, &ThreadParam);
558   pthread_setschedparam(th2, SCHED_FIFO, &ThreadParam);
559   pthread_setschedparam(pthread_self(), SCHED_FIFO, &ThreadParam);
560   printf("Thread1−2 are created.\n");
561
562   while (user_input == 0){
563     scanf("%d", &user_input);
564     clock_nanosleep(CLOCK_MONOTONIC, 0, &PreventStuck, NULL);
565   }
566
567   pthread_join(th1,NULL);
568   pthread_join(th2,NULL);
569   printf("Thread1−2 are completed.\n");
570   pthread_mutex_destroy(&mutex);
571
572   printf("BATTERY VOLTAGE(AT START TIME):%3.1fV\n",bat_vol);
573
574   printf("Main is completed.\n");
575   fclose (debug_fp);
576
577   return 0;
578 }
```

Angle/Makefile

List B.11: make ファイル

```
1  CC      = gcc −lrt
2  CFLAGS = −O0 −I/usr/src/linux −I/usr/realtime/include −I/usr/local/include/opencv
3  LDFLAGS = −L/usr/lib −L/root/OpenCV−2.3.0/lib
4  LDLIBS = −lgpg6204 −lpthread −lgpg3300
5
6  all : user
7
8  user: user.o
9    $(CC) −o user $(LDFLAGS) user.o $(LDLIBS)
10
11 user.o: user.c user.h
12   $(CC) −o user.o $(CFLAGS) −Wall −c user.c
13
14 clean:
15   rm −f ∗.o user
```

B.3 解析

B.3.1 最適化のアルゴリズム

Fig. B.3.2，Fig. B.3.3 に解析で用いたソースコードを示す。また，最適化のアルゴリズムを以下に示す。

1. 「gain.cpp」のプログラム内で設計変数，伝達関数を記述する。
2. プログラムを実行すると，定義した設計変数の値が変化していき，開ループ伝達関数の位相余裕の条件を満たしていた時に，目標値追従特性に関する感度関数と外乱抑圧特性に関する感度関数の H2 ノルムを計算する。
3. プログラムが終了したときに H2 ノルムが最小の時の設計パラメータを最適化パラメータとする。

B.3.2 ゲインの最適化

Opt/gain.cpp

List B.12: c ファイル

```
/*
c++で実行､最適化プログラム
*/
#include <math.h>
#include <iostream>
#include <fstream>
#include <complex>
#include <vector>
#include <time.h>

using namespace std;

//#define GAIN//ゲイン余裕から計算
#define PHASE//位相余裕から計算


bool stability ();

vector<double> k;
vector<double> optimalPara;
/////////////////設計者が定義/////////////////////////////////////////////
//ここに求めたい条件を書く !!!
constexpr double minPara = 2;//求めたい制御器パラメータの最小値. 0.0 にはしないこと
constexpr double maxPara = 160.0;//求めたい制御器パラメータの最大値
constexpr int intervalPara = 0.1;//10^n~ 2*10^n までの制御器パラメータの計算間隔. 10^n~10^(n+1)間で適用
constexpr int intervalFreq = 1000;//10^n~ 2*10^n までの周波数の計算間隔. 10^n~10^(n+1)間で適用
constexpr double minFreq = 0.1;//求めたい周波数領域の最小値 [rad/s]
```

```
28 constexpr double maxFreq = 1000.0;//求めたい周波数領域の最大値 [rad/s]
29 constexpr double roomRef = −50.0;//確保したい余裕
30 constexpr double omomi = 10.0;
31 constexpr double omomie = 1.0;
32 constexpr double tau_r = 0.44;
33
34 //求めたい制御器パラメータ関係
35 constexpr double parameters = 1;//求めたい制御パラメータの個数 k[0]:P ゲイン、k[1]:D ゲイン、k[2]:AmFB ゲイン,k[3]:ねじれトルクFB ゲイン、k[4]:HPF のカットオフ周波数,k[5]:ねじれ角度FB ゲイン
36 // !!! 以下に定数を定 義!!!
37 constexpr double d_e = 1.0e−03;
38 constexpr double d_l = 1.3e−06;
39 constexpr double d_m = 2.387∗ 1e−5;
40 constexpr double g = 300.0;
41 constexpr double k_s = 3.0;
42 constexpr double wc = 1.0;
43 constexpr double we = 1.0;
44 constexpr double k_t = 0.011;
45 constexpr double k_e = −8.0;
46 constexpr double M_l = 0.0007;
47 constexpr double M_m = 7.31e−08;
48 constexpr double n = 100.0;
49 constexpr double k_pv = 0.0;
50 constexpr double m = 0.0;
51 constexpr double lbl = 0.015;
52
53 double normsumm = 0.0;
54 double normsum = 0.0;
55 double normbl = 0.0;
56 double normcmd = 0.0;
57 double angle = 0.0;
58 double gainf = 0.0;
59 double d = 0.0;
60 double a1 = 100.0;
61 double a2 = 10.0;
62 double a3 = 0.1;
63 double a4 = 0.04;
64 double a5 = 2.0;
65
66
67 // !!! 以下で各システムを定義す る!!! 定義だけでい い!!!
68 //関数の中で毎回定義すると時間がかかる.
69   complex<double> A;
70   complex<double> B;
71   complex<double> C;
72   complex<double> D;
73   complex<double> E;
74   complex<double> F;
75   complex<double> G;
76   complex<double> H;
77   complex<double> I;
78   complex<double> J;
79   complex<double> K;
80   complex<double> L;
81   complex<double> M;
82   complex<double> N;
83   complex<double> O;
84   complex<double> P;
85   complex<double> Q;
86   complex<double> R;
87   complex<double> S;
88   complex<double> T;
89   complex<double> U;
90   complex<double> V;
```

```
91    complex<double> W;
92    complex<double> X;
93    complex<double> Y;
94    complex<double> Z;
95    complex<double> Wcmd;
96    complex<double> Wbl;
97    complex<double> We;
98    complex<double> Ze;
99    complex<double> AA;
100   complex<double> AB;
101   complex<double> AC;
102 ////////////////////////////////////////////////////////
103
104 int freqRes(double *normsummf,double *normsumf,double *normblf,double *normcmdf, double *anglef, double *
              gainlef) {
105
106   //ラウス数列を用いた安定判別
107   /*#ifdef CLOSED
108     if ( stability ()) {
109     } else {
110       return 0;
111     }
112   #endif*/
113
114   //以下計算用パラメータ
115   double gain = 0.0;
116   double phase = 0.0;
117   double gainRec = 0.0;
118   double phaseRec = 0.0;
119   complex<double> s(0.0, 0.0);//現在計算する周波数
120   complex<double> closedTransfer(0.0, 0.0);//閉ループ伝達関数
121   complex<double> loopTransfer(0.0, 0.0);//一巡伝達関数
122
123 /////////////////設計者が定義/////////////////////////////////////////
124   //目標値追従特性
125   complex<double> Gcmd(0.0, 0.0);
126   //外乱抑圧特性
127   complex<double> Gbl(0.0, 0.0);
128   //バックドライバビリティ
129   complex<double> Gt(0.0, 0.0);
130   //電流に対する感度関数
131   //complex<double> Gi(0.0, 0.0);
132 ////////////////////////////////////////////////////////
133
134 /////////////////設計者が定義/////////////////////////////////////////
135   //最適化で使用するパラメータ//
136   double gcf = 0.0;//Gain crossover frequency. 現在のゲイン交差周波数
137   double gcfRec = 0.0;//周波数応答の計算済みで最大のゲイン交差周波数. 周波数での応答更新用
138   static double gcfRoomRec = 0.0;//周波数応答計算後に記録させるゲイン交差周波数. パラメータ変更後の応答更新用
139   double room = 0;//実際の余裕. 位相かゲインかは#define で決める.
140   double phaseVariable = 0.0;//位相計算補正用変数
141   double variableIntervalFreq = minFreq / intervalFreq;//計算間隔の変数
142   int intervalFreqCount = 0;//周波数ごとの計算間隔を変更する変数
143     static double normRec = 1000;
144     double normSqSumbl = 0.0;
145   double normSqSumcmd = 0.0;
146   //double normSqSume = 0.0;
147   //double normSqSumi = 0.0;
148     double Normcmd = 0.0;
149   double mNormcmd = 0.0;
150   double Normbl = 0.0;
151   double mNormbl = 0.0;
152   //double Norme = 0.0;
153   //double mNorme = 0.0;
```

```
154   //double Normi = 0.0;
155   //double mNormi = 0.0;
156   double NormSum = 0.0;
157   double Wsbl = 0.0;
158   //double Wi = 0.0;
159   double NormSum1 = 0.0;
160   double normsumm = 0.0;
161 ////////////////////////////////////////////////////////
162
163   for (double freq = minFreq; freq <= maxFreq; freq += variableIntervalFreq) {
164     if (intervalFreqCount == intervalFreq * 9) {//カウントが一杯になると計算間隔を変更
165       variableIntervalFreq  *= 10;
166       intervalFreqCount = 1;//ここは 1が正解
167     } else {
168       intervalFreqCount++;
169     }
170
171     phaseVariable = phase;
172     s = complex<double> (0.01, freq);//s = 0.01 + jw の計算. 不安定極を避けるため +0.01してある.
173
174 ////////////////設計者が定義/////////////////////////////////////////
175     // !!! 以下伝達関数の計 算!!! 頑張って書 く!!!
176         k[1]  = 3000;
177     k[2]  = 105;
178     k[3]  = 1.2;
179     k[4]  = 190000;
180     k[5]  = 0.07;
181       Q = g/(g+s);
182     H = s/(k[4]+s)+(k[5]);//prop
183     Ze = k_e + d_e*s;//*Q;
184     W = complex<double> (1.0, 0.0)*complex<double> (omomi, 0.0) / (s + complex<double> (omomi, 0.0));
185     We = complex<double> (omomie, 0.0) / (s + complex<double> (omomie, 0.0));
186     A = k_s/(n*(k_s+d_l*s+M_l*s*s));
187     M = k_t/(M_m*s*s+d_m*s+k_s/(n*n)−A*k_s/n);
188     X = k_t*Q*Q/M+g*M_m*s*Q−g*M_m*s*Q;
189     B = M/(complex<double> (1.0, 0.0)−X*M/k_t);
190     C = B/(complex<double> (1.0, 0.0)+B*k_s*H*(complex<double> (1.0, 0.0)/n−A)*(k[3]));//k4
191         //C = B/(complex<double> (1.0, 0.0)+B*k_s*H*(complex<double> (1.0, 0.0)/n−A)*complex<double> (0.0,
                0.0));
192     D = M_m*C/k_t/(complex<double> (1.0, 0.0)+M_m*C*(k[2])*s*Q/k_t);
193     F = M_m/k_t*(k[0]+(k[1])*s*Q)/(M_m*(k[2])*s*Q/k_t/A+(k[3])*H*k_s*(complex<double> (1.0, 0.0)/n/A−complex
              <double> (1.0, 0.0))−X/k_t/A+complex<double> (1.0, 0.0)/A/M);//k4
194         //F = M_m/k_t*(k[0]+k[1]*s*Q)/(M_m*k[2]*s*Q/k_t/A+complex<double> (0.0, 0.0)*H*k_s*(complex<double>
                (1.0, 0.0)/n/A−complex<double> (1.0, 0.0))−X/k_t/A+complex<double> (1.0, 0.0)/A/M);
195         I = complex<double> (1.0, 0.0)/(M_l*s*s+d_l*s+k_s+F*k_s/n/A);
196     Y = M_m*s*s+d_m*s+k_s/(n*n);
197     U = (k_t*X−k_t*(k[3])*k_s*H/n−(k[2])*s*Q*M_m−Y)/(−k_t*(k[3])*k_s*H+M_m*(k[0]+(k[1])*s*Q)−k_s/n);//k4*2
198         //U = (k_t*X−k_t*complex<double> (0.0, 0.0)*k_s*H/n−k[2]*s*Q*M_m−Y)/(−k_t*complex<double> (0.0,
                0.0)*k_s*H+M_m*(k[0]+k[1]*s*Q)−k_s/n);
199     Z = complex<double> (1.0, 0.0)/(M_l*s*s+d_l*s+k_s−k_s/(n*U));
200     J = Z*Ze/(Z*Ze−complex<double> (1.0, 0.0));
201     K = M/(complex<double> (1.0, 0.0)−M*X/k_t);
202     L = n*(M_l*s*s+d_l*s+k_s)/k_s;
203     N = L*(M_m*s*s+d_m*s)+k_s*L/(n*n)−k_s/n;
204     P = (k_t*N+k_s/n)/(M_m*s*s+d_m*s−k_t*L+k_s/(n*n));
205     R = (k_s/n)/(M_m*s*s+d_m*s−k_t*L+k_s/(n*n));
206     O = (M_m*s*s+d_m*s+k_s/(n*n)−k_t*L)/(k_t*N+k_s/n);
207     S = −Ze/(M_l*s*s+d_l*s−Ze+k_s−k_s/n/O);
208     T = (k_s*R/n−k_s)/(M_l*s*s+d_l*s+k_s−k_s*P/n);
209     V = (g*M_m*s*Q*(Q−complex<double> (1.0, 0.0))/k_t−M_m*(k[2])*s*Q/k_t−(k[3])*k_s*H/n)/(complex<double
              > (1.0, 0.0)−Q*Q);//k4
210         //V = (g*M_m*s*Q*(Q−complex<double> (1.0, 0.0))/k_t−M_m*k[2]*s*Q/k_t−complex<double> (0.0, 0.0)*k_s
                *H/n)/(complex<double> (1.0, 0.0)−Q*Q);
211     AA = ((k[3])*k_s*H−M_m/k_t*(k[0]+(k[1])*s*Q))/(complex<double> (1.0, 0.0)−Q*Q);
```

```
212         //AA = (complex<double> (0.0, 0.0)*k_s*H-M_m/k_t*(k[0]+k[1]*s*Q))/(complex<double> (1.0, 0.0)-Q*Q);
213     Wbl = -lbl/(tau_r/k_e);
214     AC = (-(k[1]+k[2]*s*Q)*M_m/k_t+k[3]*k_s*H)/(complex<double> (1.0, 0.0)-Q*Q);
215     AB = (k_t*AC+k_s/n)/(M_m*s*s+d_m*s+k_s/n/n-k_t*V);
216         loopTransfer = F;
217         Gcmd = complex<double> (1.0, 0.0)/(complex<double> (1.0, 0.0)+F);
218         Gbl = (n*(M_m*s*s+d_m*s)-k_t*V*n)/(k_t*(V*L+AA)-N); // xbl to xl
219     Gt = Ze/(k_s*AB/n-k_s+Ze-(M_l*s*s+d_l*s));
220 ////////////////////////////////////////////////////////
221
222     //ゲイン・位相の計算
223     try {
224       gain = 20 * log10(abs(loopTransfer));
225     }
226     catch (char * str) {
227       gain = 0;
228     }
229     phase = arg(loopTransfer) * 180 / M_PI;
230     //位相余裕・ゲイン交差周波数の計算
231     if (abs(phase - phaseVariable) > 180) {//位相計算値の修正用
232       phase -= 360;
233     }
234     if (gain > 0.0) {
235       gcfRec = freq;
236       phaseRec = phase + 180;
237     }
238
239     if (phaseRec < 0) {
240       return 0;
241     }
242     //ゲイン余裕の計算
243     if (phase > -180) {
244       gainRec = - gain;
245     }
246
247 ////////////////設計者が定義/////////////////////////////////////////
248     //ノルムの二乗和を計算. Cmd は直達項を避けるため, 重みの役割を含んだW をかける
249     normSqSumcmd += norm(Gcmd*W) * variableIntervalFreq;
250         normSqSumbl += norm(Gbl*We) * variableIntervalFreq;
251         //normSqSume += norm(Ge*W) * variableIntervalFreq;
252         //normSqSumi += norm(Gi*W*Wi) * variableIntervalFreq;
253   }
254
255   Normcmd = sqrt(normSqSumcmd / M_PI);
256   Normbl = sqrt(normSqSumbl / M_PI);
257   //Norme = sqrt(normSqSume / M_PI);
258   //Normi = sqrt(normSqSumi / M_PI);
259   NormSum = Normcmd + Normbl;
260   normsumm = normRec;
261   //出力したパラメータ（ゲイン）以外は、ポインタ関数で定義する//
262   *anglef = phaseRec;
263   *gainlef = gainRec;
264   *normsumf = NormSum;
265   *normsummf = normsumm;
266   *normblf = Normbl;
267   *normcmdf = Normcmd;
268 ////////////////////////////////////////////////////////
269   #ifdef PHASE
270   if (phaseRec >= roomRef){
271     //d = 1.0;
272     if(NormSum <= normRec) {
273   //if (phaseRec >= roomRef) {
274     for (int l = 0; l < k.size(); l++) {
275       optimalPara[l] = k[l];
```

```
276       }
277
278       normRec = NormSum;
279       mNormbl = Normbl;
280       //mNorme = Norme;
281       mNormcmd = Normcmd;
282       //mNormi = Normi;
283       gainf = phaseRec;
284     }
285       }
286     #endif
287     #ifdef GAIN
288     if (gainRec >= roomRef && NormSum <= normRec) {
289         //d = 1.0;
290       for (int l = 0; l < k.size(); l++) {
291         optimalPara[l] = k[l];
292       }
293
294       normRec = NormSum;
295       mNormbl = Normbl;
296       //mNorme = Norme;
297       mNormcmd = Normcmd;
298       //mNormi = Normi;
299     }
300     #endif
301
302   return 0;
303 }
304
305 int loopFun(int loopCount) {//任意の制御器パラメータ数に対応するための再帰関数
306       FILE* pFile;
307       pFile = fopen("data2.dat", "a");
308     //double variableIntervalPara = minPara / intervalPara;//制御器の計算間隔の変数
309
310     double variableIntervalPara = 0.2;//制御器の計算間隔の変数
311     int intervalParaCount = 0;//計算間隔を変更するための変数
312
313     if (loopCount > 0) {
314       loopCount--;
315
316       for (double i = minPara; i <= maxPara; i += variableIntervalPara) {
317       /*if (i == 0.001) {
318         variableIntervalPara = 0.0099;
319       } else if (i >= 0.1 && i <=2.0) {
320         variableIntervalPara = 0.1;
321       } else if (i >= 2.0 && i <=100) {
322         variableIntervalPara = 10;
323       } else if(i >= 100 && i <= 6000) {
324         variableIntervalPara = 400;
325       } else {
326         variableIntervalPara = 40000;
327       }
328               k[loopCount] = i;*/
329         if (loopCount == 0) {
330           k[loopCount] = i;//i;
331         } else if (loopCount == 1) {
332           k[loopCount] = 300;//i/100-1000;//初期値を 0にするために定数を引く
333               } else if (loopCount == 2) {
334           k[loopCount] = 60;//i/2000-50;
335               } else if (loopCount == 3) {
336           k[loopCount] = 0.0;//i/50000-2;
337               } else if (loopCount == 4) {
338           k[loopCount] = 0;//i/5000-20;
339               } else if (loopCount == 5) {
```

```
340           k[loopCount] = 0.0;//i/5000000−0.02;
341         }
342         loopFun(loopCount);
343         if (intervalParaCount == intervalPara * 9 && variableIntervalPara<100.0) {//カウントが一杯になると計算間隔を変
                更 1000以上は 100ずつ増える
344           variableIntervalPara *= 10;
345           intervalParaCount = 1;//ここは 1が正解
346         } else {
347           intervalParaCount++;
348         }
349
350
351       }
352
353     } else {
354
355       freqRes(&normsumm,&normsum,&normbl,&normcmd,&angle,&gainf);
356
357     }
358
359 ////////////////設計者が定義/////////////////////////////////////////
360     //if (angle < roomRef) {//制約条件でのゲインを出力
361     if (d == 0.0) {
362       d = 0.0;
363     fprintf(pFile, "%f\t%f\t%f\t%f\t%e\t%e\t%e\t%e\t%e\t%e\t%f%e\t\n",normsumm,normsum,normbl,normcmd,k[0],
              k[1],k[2],k[3],k[4],k[5]*3.0,gainf);
364     fclose(pFile);
365     }
366     //
367 ////////////////////////////////////////////////////////
368     return 0;
369 }
370
371 int main() {
372     clock_t start = clock();
373     //求めたいデータ数の調整
374     k.resize(parameters);
375     optimalPara.resize(parameters);
376
377     //繰り返し計算
378     loopFun(parameters);
379
380
381     //
382
383 ////////////////設計者が確認,cygwin の画面に出力/////////////////////////////////////////
384     //最適パラメータの表示
385     if (optimalPara[0] == 0.0) {
386       printf("Such_parameters_are_not_exist!!\n");
387       printf("You_must_relax_the_phase_restriction!!");
388     } else {
389       for (int i = 0; i < optimalPara.size(); i++) {
390       printf("%e\t", optimalPara[i]);
391     }//*/
392 ////////////////////////////////////////////////////////
393     }
394     clock_t end = clock();
395       std::cout << "Elapsed_Time:" << (double)(end − start) << "[ms]\n";
396     return 0;
397 }
398 /*bool stability () {
399     int denoSize = sizeof(denoPara) / sizeof(*denoPara);//ラウス数列の行数
400     bool kakunou = true;//ラウス数列の 1, 2行目に値を入れるための変数
401     int kakunouJun = denoSize − 1;//同上
```

```
402   int row = 0;//ラウス行列の列数
403   //分母の係数が負なら不安定
404   for (int i = 0; i < denoSize; i++) {
405     if (denoPara[i] < 0) {
406       return false;
407     }
408   }
409
410   //伝達関数の分母からラウス表の配列を確保
411   if (denoSize % 2 == 0) {
412     row = denoSize / 2;
413   } else {
414     row = (int) denoSize / 2 + 1;
415   }
416   vector<vector<double>> routh(denoSize, vector<double> (row));
417
418   //ラウス数列の 1, 2行目に値を格納
419   for (int i = row; i > 0; kakunouJun--) {
420     if (kakunou) {
421       routh[0][row - i] = denoPara[kakunouJun];
422       kakunou = false;
423     } else {
424       routh[1][row - i] = denoPara[kakunouJun];
425       kakunou = true;
426       i--;
427     }
428   }
429   //ラウス表の計算
430   for (int i = 2; i < routh.size(); i++) {
431     for (int j = 0; j < routh.at(0).size(); j++) {
432       routh[i][j] = (routh[i - 1][0] * routh[i - 2][j + 1] - routh[i - 2][0] * routh[i - 1][j + 1]) / routh[i - 1][0];
433     }
434   }
435   //ラウス数列による安定性の判定
436   for (int i = 0; i < routh.size(); i++) {
437     if (routh[i][0] < 0.0) {
438       return false;
439     }
440   }
441   /*//*/ラウス表を表示
442   for (int i = 0; i < routh.size(); i++) {
443     for (int j = 0; j < routh.at(0).size(); j++) {
444       printf("%lf ", routh[i][j]);
445     }
446     printf("\n");
447   }//*//*
448
449   return true;
450 }*/
```

B.3.3 ゲイン線図やナイキスト線図による解析

Opt/diagram.cpp

List B.13: c ファイル

```
1  /*
2  c++で実行、ボード線図を出力するプログラム
3  */
4  #include <math.h>
```

```
5  #include <iostream>
6  #include <fstream>
7  #include <complex>
8  #include <vector>
9  #include <time.h>
10
11 using namespace std;
12 #define CONV
13 //#define PROP1
14 //#define PROP2
15
16
17 int main() {
18   clock_t start = clock();
19
20   FILE* freq_data;
21
22
23   constexpr int intervalFreq = 1000.0;//10^n~ 2*10^n までの周波数の計算間隔. 10^n~10^(n+1)間で適用
24   constexpr double minFreq = 0.1;//求めたい周波数領域の最小値 [rad/s]
25   constexpr double maxFreq = 1000.0;//求めたい周波数領域の最大値 [rad/s]
26
27 /////////////////設計者が定義/////////////////////////////////////////
28     //求めたい制御器パラメータ関係//
29   // !!! 以下に定数を定 義!!!
30 constexpr double d_e = 1.0e-03;//1.0e-03;
31 constexpr double d_l = 1.3e-06;
32 constexpr double d_m = 2.387* 1e-5;
33 constexpr double g = 300.0;
34 constexpr double k_s = 3.0;
35 constexpr double wc = 1.0;
36 constexpr double we = 1.0;
37 constexpr double k_t = 0.011;
38 constexpr double k_e = -8.0;
39 constexpr double M_l = 0.0007;//0.000035;//0.0007
40 constexpr double M_m = 7.31e-08;
41 constexpr double n = 100.0;
42 constexpr double k_pv = 0.0;
43 constexpr double m = 0.0;
44 constexpr double lbl = 0.015;
45 constexpr double omomi = 10.0;
46 constexpr double omomie = 1.0;
47 constexpr double tau_r = 0.44;
48
49 double normsumm = 0.0;
50 double normsum = 0.0;
51 double normbl = 0.0;
52 double normcmd = 0.0;
53 double angle = 0.0;
54 double gainf = 0.0;
55 double d = 0.0;
56
57 ////////////////////////////////////////////////////////////
58
59 /////////////////設計者が定義/////////////////////////////////////////
60     //制御器ゲイン//
61   #ifdef CONV
62     constexpr double k[6] = {190000, 3000, 105, 1.2, 2000, 0.07};//剛性、トルク比例ゲイン、モータ角速度比
           例ゲイン
63     freq_data = fopen("prop.dat", "w");
64   #elif defined PROP1
65     constexpr double k[3] = {80.000000e+00, 6.00000e+02, 8.000000e+02};
66     freq_data = fopen("ks80.dat", "w");
67   #elif defined PROP2
```

```
68      constexpr double k[3] = {4.000000e−01, 3.60000e+03,      2.800000e+03};
69      freq_data = fopen("ks04.dat", "w");
70    #endif
71  ////////////////////////////////////////////////////////
72
73  ////////////////設計者が定義////////////////////////////////////////
74      //伝達関数を求めるための中間変数//
75    // !!! 以下で各システムを定義す る!!! 定義だけでい い!!!
76    //関数の中で毎回定義すると時間がかかる.
77    complex<double> A;
78    complex<double> B;
79    complex<double> C;
80    complex<double> D;
81    complex<double> E;
82    complex<double> F;
83    complex<double> G;
84    complex<double> H;
85    complex<double> I;
86    complex<double> J;
87    complex<double> K;
88    complex<double> L;
89    complex<double> M;
90    complex<double> N;
91    complex<double> O;
92    complex<double> P;
93    complex<double> Q;
94    complex<double> R;
95    complex<double> S;
96    complex<double> T;
97    complex<double> U;
98    complex<double> V;
99    complex<double> W;
100   complex<double> X;
101   complex<double> Y;
102   complex<double> Z;
103   complex<double> AA;
104   complex<double> Wcmd;
105   complex<double> Wbl;
106   complex<double> We;
107   complex<double> Ze;
108   complex<double> AB;
109   complex<double> AC;
110
111   complex<double> plant;
112   complex<double> plant2;
113   complex<double> DOB;
114   complex<double> Gdelta;
115 ////////////////////////////////////////////////////////
116
117 ////////////////設計者が定義////////////////////////////////////////
118   //以上に求めたい条件を書く///////
119   //求めたいデータ数の調整
120   //以下計算用パラメータ
121   complex<double> s(0.0, 0.0);//現在計算する周波数
122   complex<double> closedTransfer(0.0, 0.0);//閉ループ伝達関数
123   complex<double> loopTransfer(0.0, 0.0);//一巡伝達関数
124
125   //目標値追従特性
126   complex<double> Gcmd(0.0, 0.0);
127   //目標値追従特性に関する感度関数
128   complex<double> Gcmde(0.0, 0.0);
129   //外乱抑圧特性
130   complex<double> Gt(0.0, 0.0);
131   //外乱抑圧特性
```

```
132   complex<double> Gbl(0.0, 0.0);
133   //トルク飽和抑圧特性
134   //complex<double> Gt(0.0, 0.0);
135 ////////////////////////////////////////////////////////
136
137 ////////////////設計者が定義/////////////////////////////////////////
138     //ボード線図の導出に使う変数//
139   double phaseVariable = 0.0;//位相計算補正用変数
140   double variableIntervalFreq = minFreq / intervalFreq;//計算間隔の変数
141   double gainbl = 0.0;
142   double gainCmd = 0.0;
143   double gainCmde = 0.0;
144   double gaine = 0.0;
145   double gaint = 0.0;
146   double gainQ = 0.0;
147   double gaindelta = 0.0;
148   double realAxis = 0.0;
149   double imagAxis = 0.0;
150   double Wi =0.0;
151   int intervalFreqCount = 0;//周波数ごとの計算間隔を変更する変数
152
153 ////////////////////////////////////////////////////////
154
155
156   for (double freq = minFreq; freq <= maxFreq; freq += variableIntervalFreq) {
157     if (intervalFreqCount == intervalFreq * 9) {//カウントが一杯になると計算間隔を変更
158       variableIntervalFreq  *= 10;
159       intervalFreqCount = 1;//ここは 1が正解
160     } else {
161       intervalFreqCount++;
162     }
163   //phaseVariable = phase;
164
165     s = complex<double> (0.01, freq);//s = 0.01 + jw の計算. 不安定極を避けるため +0.01してある.
166
167 ////////////////設計者が定義/////////////////////////////////////////
168     // !!! 以下伝達関数の計 算!!! 頑張って書 く!!!
169       //Q = g/(g+s);
170       Q = g/(g+s);
171     H = s/(k[4]+s)+(k[5]);//prop
172         //H = s/(complex<double> (0.0, 0.0)+s)+k[3]/100000;//conv2
173           //H = k[3]/100000;//conv1
174     Ze = k_e + d_e*s;//*Q;
175     W = complex<double> (omomi, 0.0) / (s + complex<double> (omomi, 0.0));
176     We = complex<double> (omomie, 0.0) / (s + complex<double> (omomie, 0.0));
177     A = k_s/(n*(k_s+d_l*s+M_l*s*s));
178     M = k_t/(M_m*s*s+d_m*s+k_s/(n*n)-A*k_s/n);
179     X = k_t*Q*Q/M+g*M_m*s*Q-g*M_m*s*Q;
180     B = M/(complex<double> (1.0, 0.0)-X*M/k_t);
181     C = B/(complex<double> (1.0, 0.0)+B*k_s*H*(complex<double> (1.0, 0.0)/n-A)*(k[3]));//k4
182         //C = B/(complex<double> (1.0, 0.0)+B*k_s*H*(complex<double> (1.0, 0.0)/n-A)*complex<double> (0.0,
                  0.0));
183     D = M_m*C/k_t/(complex<double> (1.0, 0.0)+M_m*C*(k[2])*s*Q/k_t);
184     F = M_m/k_t*(k[0]+(k[1])*s*Q)/(M_m*(k[2])*s*Q/k_t/A+(k[3])*H*k_s*(complex<double> (1.0, 0.0)/n/A-complex
              <double> (1.0, 0.0))-X/k_t/A+complex<double> (1.0, 0.0)/A/M);//k4
185         //F = M_m/k_t*(k[0]+k[1]*s*Q)/(M_m*k[2]*s*Q/k_t/A+complex<double> (0.0, 0.0)*H*k_s*(complex<double>
                    (1.0, 0.0)/n/A-complex<double> (1.0, 0.0))-X/k_t/A+complex<double> (1.0, 0.0)/A/M);
186         I = complex<double> (1.0, 0.0)/(M_l*s*s+d_l*s+k_s+F*k_s/n/A);
187     Y = M_m*s*s+d_m*s+k_s/(n*n);
188     U = (k_t*X-k_t*(k[3])*k_s*H/n-(k[2])*s*Q*M_m-Y)/(-k_t*(k[3])*k_s*H+M_m*(k[0]+(k[1])*s*Q)-k_s/n);//k4*2
189         //U = (k_t*X-k_t*complex<double> (0.0, 0.0)*k_s*H/n-k[2]*s*Q*M_m-Y)/(-k_t*complex<double> (0.0,
                  0.0)*k_s*H+M_m*(k[0]+k[1]*s*Q)-k_s/n);
190     Z = complex<double> (1.0, 0.0)/(M_l*s*s+d_l*s+k_s-k_s/(n*U));
191     J = Z*Ze/(Z*Ze-complex<double> (1.0, 0.0));
```

```
192     K = M/(complex<double> (1.0, 0.0)−M∗X/k_t);
193     L = n∗(M_l∗s∗s+d_l∗s+k_s)/k_s;
194     N = L∗(M_m∗s∗s+d_m∗s)+k_s∗L/(n∗n)−k_s/n;
195     P = (k_t∗N+k_s/n)/(M_m∗s∗s+d_m∗s−k_t∗L+k_s/(n∗n));
196     R = (k_s/n)/(M_m∗s∗s+d_m∗s−k_t∗L+k_s/(n∗n));
197     O = (M_m∗s∗s+d_m∗s+k_s/(n∗n)−k_t∗L)/(k_t∗N+k_s/n);
198     S = −Ze/(M_l∗s∗s+d_l∗s−Ze+k_s−k_s/n/O);
199     T = (k_s∗R/n−k_s)/(M_l∗s∗s+d_l∗s+k_s−k_s∗P/n);
200     V = (g∗M_m∗s∗Q∗(Q−complex<double> (1.0, 0.0))/k_t−M_m∗(k[2])∗s∗Q/k_t−(k[3])∗k_s∗H/n)/(complex<double
            > (1.0, 0.0)−Q∗Q);//k4
201         //V = (g∗M_m∗s∗Q∗(Q−complex<double> (1.0, 0.0))/k_t−M_m∗k[2]∗s∗Q/k_t−complex<double> (0.0, 0.0)∗k_s
                ∗H/n)/(complex<double> (1.0, 0.0)−Q∗Q);
202     AA = ((k[3])∗k_s∗H−M_m/k_t∗(k[0]+(k[1])∗s∗Q))/(complex<double> (1.0, 0.0)−Q∗Q);
203         //AA = (complex<double> (0.0, 0.0)∗k_s∗H−M_m/k_t∗(k[0]+k[1]∗s∗Q))/(complex<double> (1.0, 0.0)−Q∗Q);
204     AC = (−(k[1]+k[2]∗s∗Q)∗M_m/k_t+k[3]∗k_s∗H)/(complex<double> (1.0, 0.0)−Q∗Q);
205     AB = (k_t∗AC+k_s/n)/(M_m∗s∗s+d_m∗s+k_s/n/n−k_t∗V);
206     Wbl = −lbl/(tau_r/k_e);
207         loopTransfer = F;
208         Gcmd = complex<double> (1.0, 0.0)/(complex<double> (1.0, 0.0)+F);
209         Gbl = (n∗(M_m∗s∗s+d_m∗s)−k_t∗V∗n)/(k_t∗(V∗L+AA)−N); // xbl to xl
210     Gdelta = complex<double> (1.0, 0.0) + (M_l∗s∗s+d_l∗s)/k_s∗Gbl;
211     Gt = Ze/(k_s∗AB/n−k_s+Ze−(M_l∗s∗s+d_l∗s));
212
213     plant = k_t∗k_s/n/((M_l∗s∗s+d_l∗s+k_s)∗(M_m∗s∗s+d_m∗s+k_s/(n∗n))−k_s∗k_s/(n∗n));
214     plant2 = (k_t+k_s∗plant/n)/(M_m∗s∗s+d_m∗s+k_s/(n∗n));//電流指令から駆動側角度
215     DOB = complex<double> (1.0, 0.0)/(complex<double> (1.0, 0.0)−((Q−complex<double> (1.0, 0.0))∗g∗M_m∗s∗
                Q∗plant2/k_t+Q∗Q));
216     //loopTransfer = M_m/k_t∗(k[0]+k[1]∗s∗Q)∗plant∗DOB;//負荷側角度FB PD
217     //loopTransfer = M_m/k_t∗DOB/(complex<double> (1.0, 0.0)+k[2]∗s∗Q∗plant2∗M_m/k_t∗DOB)∗(k[0]+k[1]∗s∗Q)∗
                plant2/n;//∗DOB;
218
219 ////////////////////////////////////////////////////////
220
221 ////////////////設計者が定義//////////////////////////////////////////
222     //ゲイン・位相の計算
223
224
225     //gainCmd = 20 ∗ log10(abs(Gcmd∗W));
226     //gainbl = 20 ∗ log10(abs(Gbl∗We∗Wbl));
227     gainQ = 20 ∗ log10(abs(loopTransfer));
228     gainCmd = 20 ∗ log10(abs(Gcmd∗W));
229     gainbl = 20 ∗ log10(abs(Gbl∗We));
230     gaindelta = 20 ∗ log10(abs(Gdelta));
231 ////////////////////////////////////////////////////////
232
233     realAxis = loopTransfer.real();
234     imagAxis = loopTransfer.imag();
235
236 ////////////////設計者が定義//////////////////////////////////////////
237     //出力//
238   fprintf(freq_data, "%e\t%e\t%e\t%e\t%e\t%e\t%e\n", freq, gainQ, gainCmd, gainbl, gaindelta, realAxis, imagAxis);
239 ////////////////////////////////////////////////////////
240
241   }
242   clock_t end = clock();
243     std::cout << "Elapsed_Time:" << (double)(end − start) << "[ms]\n";
244   return 0;
245   fclose(freq_data);
246 }
```

Appendix C

機械系の仕様

C.1 概要

USM-エンドエフェクタ間に用いる弾性カップリング (ミスミ ZG-8, MSF-16, MJT-20-Bl)3通り購入した。また，負荷試験用に HD-電磁モータ間を接続する高剛性のカップリング (ミスミ MCOG17) も購入した。それぞれのカップリングの仕様を Table C.2 に示す。

ドライバの設定は PSF-520 というソフトウェアと HDM-RS232C という通信ケーブルを用いることにより変更ができる。

このマクソン製 RE50 には Type ME のエンコーダが取り付けられており 54000pulse/rev. の分解能である。この出力を 4 逓倍したため実質的な分解能は 216000pulse/rev. である。

Table C.1: Parameters of RSA-10

Output current	10 A
Output capacity	1.0 KVA

AC サーボアクチュエータ（RSF-5B-100-E050-C）と DC モータ (DC-motor-RE50-GB-200W-KL-2WE) の軸はそれぞれ直径 5mm と 8mm であり，カップリングを用いてそれぞれを連結する。

加工図面は onedrive 上の「figure」→「Structure」，見積書は「figure」→「quotation」に入っている。

Table C.2: Specification of each coupling

Model	Torsion spring constant [Nm/rad]	Maximum torque [Nm]
ZG-8 5-8	0.48	1.0
MSF-16 5-8	4.4	1.0
MJT-20-BL 5-8	16.0	3.6
MCOG17 5-8	1,000	5.0

C.2 加工図面

実験機の機構の設計図を Fig. C-1–Fig. C-7 に示す。Fig. C-1–Fig. C-7 の部品は SolidWorks で設計し，ミスミ FA 特注サービスで発注した。

これら構成部品と組立図の SLD ファイル（SolidWorks のみで編集・閲覧可），STEP ファイル（フリーソフト Fusion360 等でも編集・閲覧可）を \\YGSERVER\common\experimental_equipments\実機製作 (ハードウェア関連)\ マニピュレータ班実機 (SEA)\CAD にアップロードした。また，見積書については，YGSERVER\common\experimental_equipments\ 実機製作 (ハードウェア関連)\ マニピュレータ班実機 (SEA)\ 見積書にアップロードした。SolidWorks による組立図のキャプチャ画像を Fig. C-8 に示す。

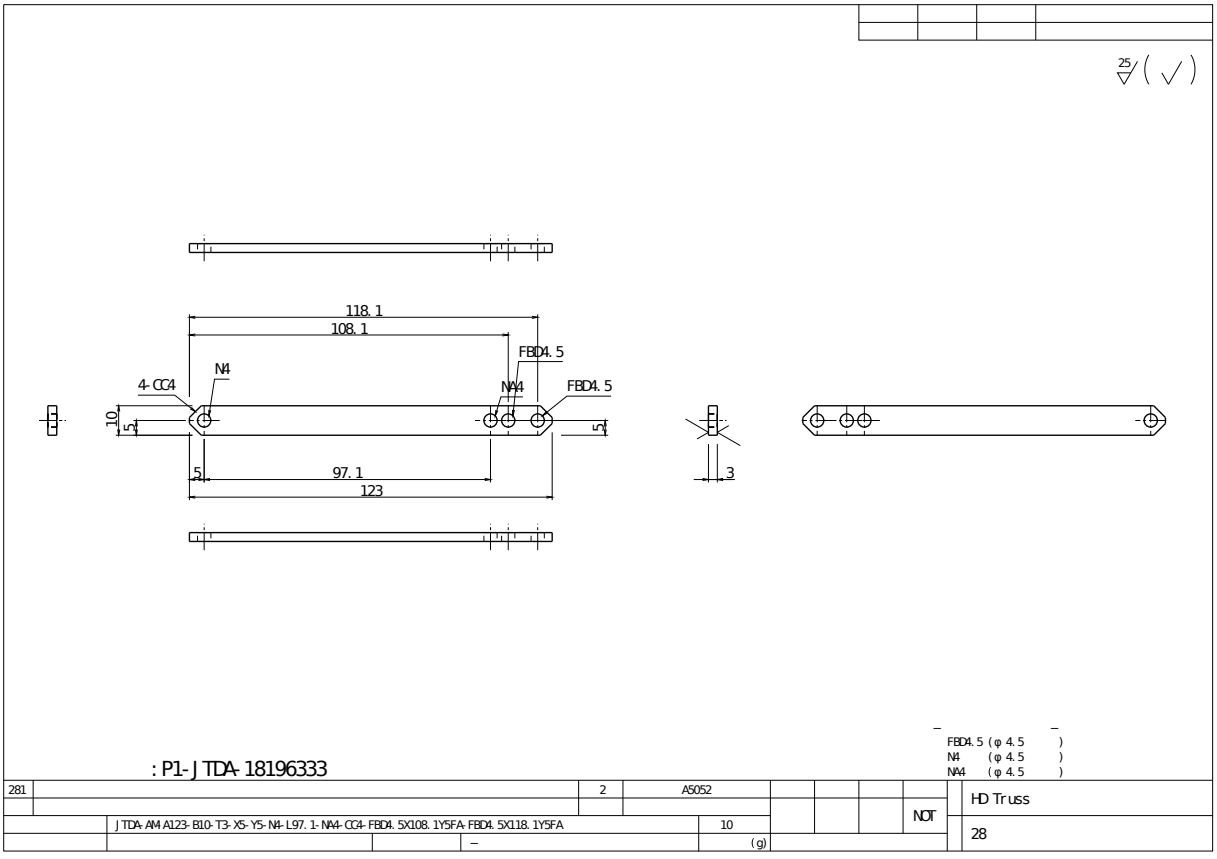


Fig. C-1: Reinforcement

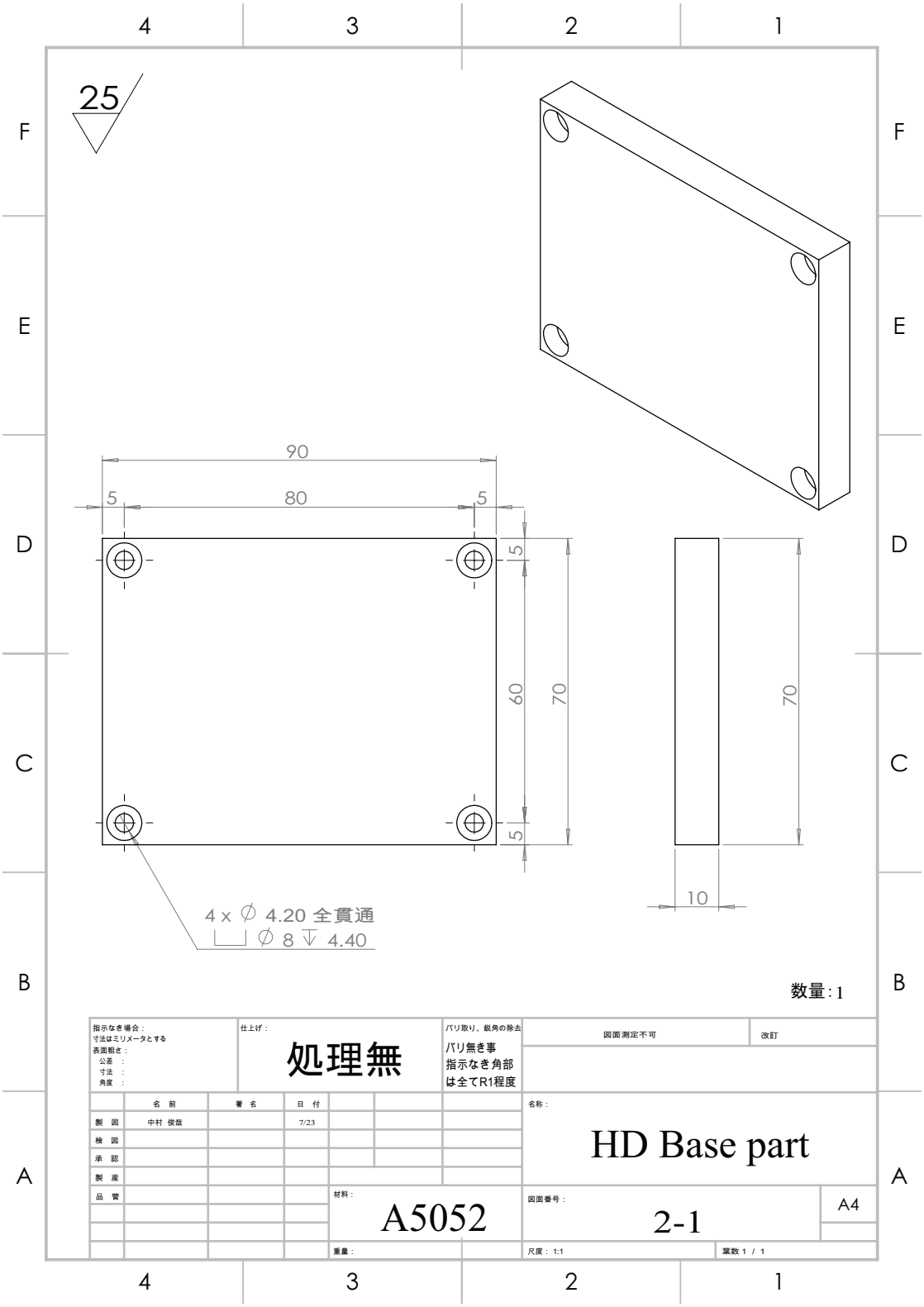


Fig. C-2: Base part

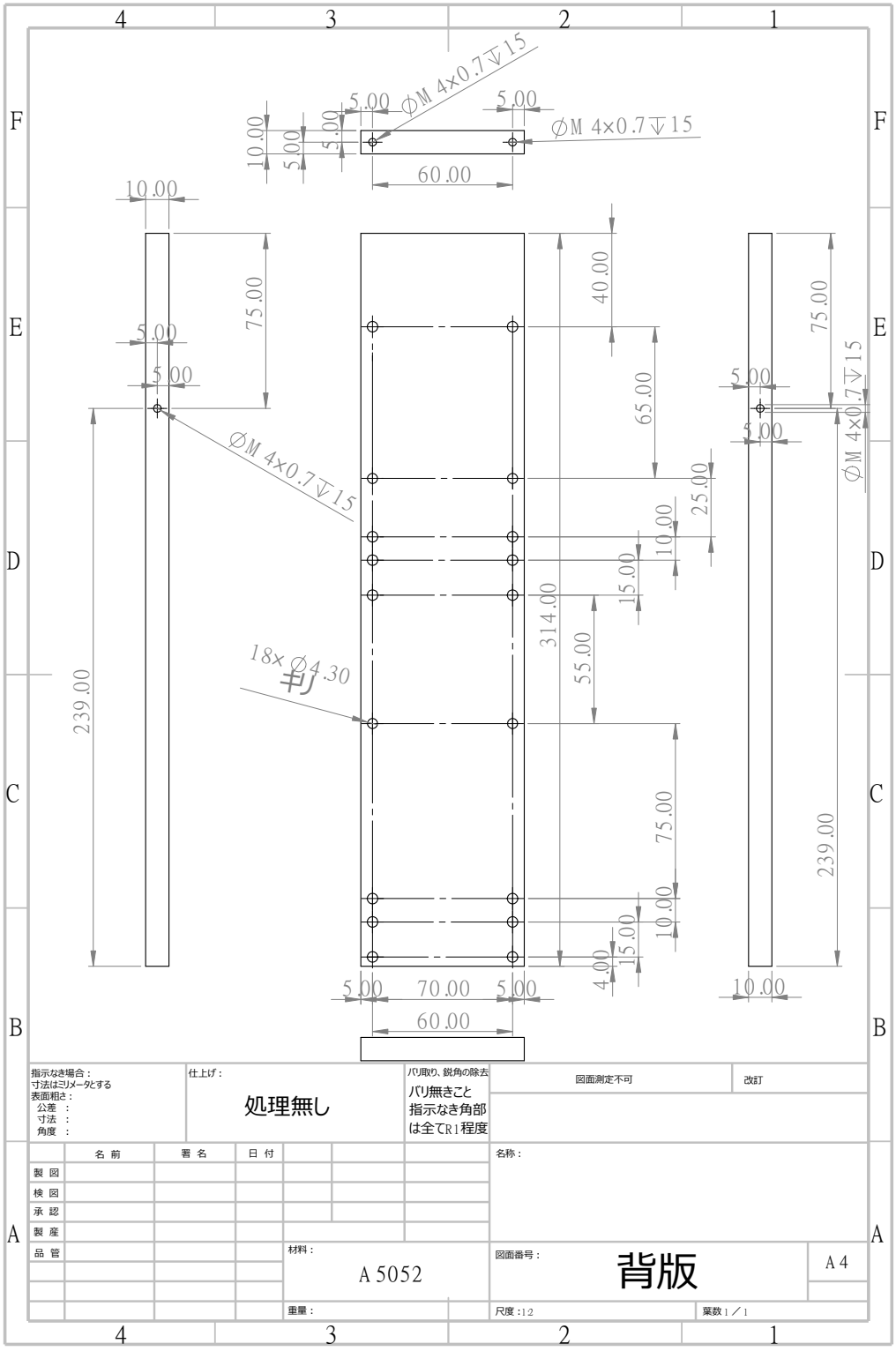


Fig. C-3: Load post

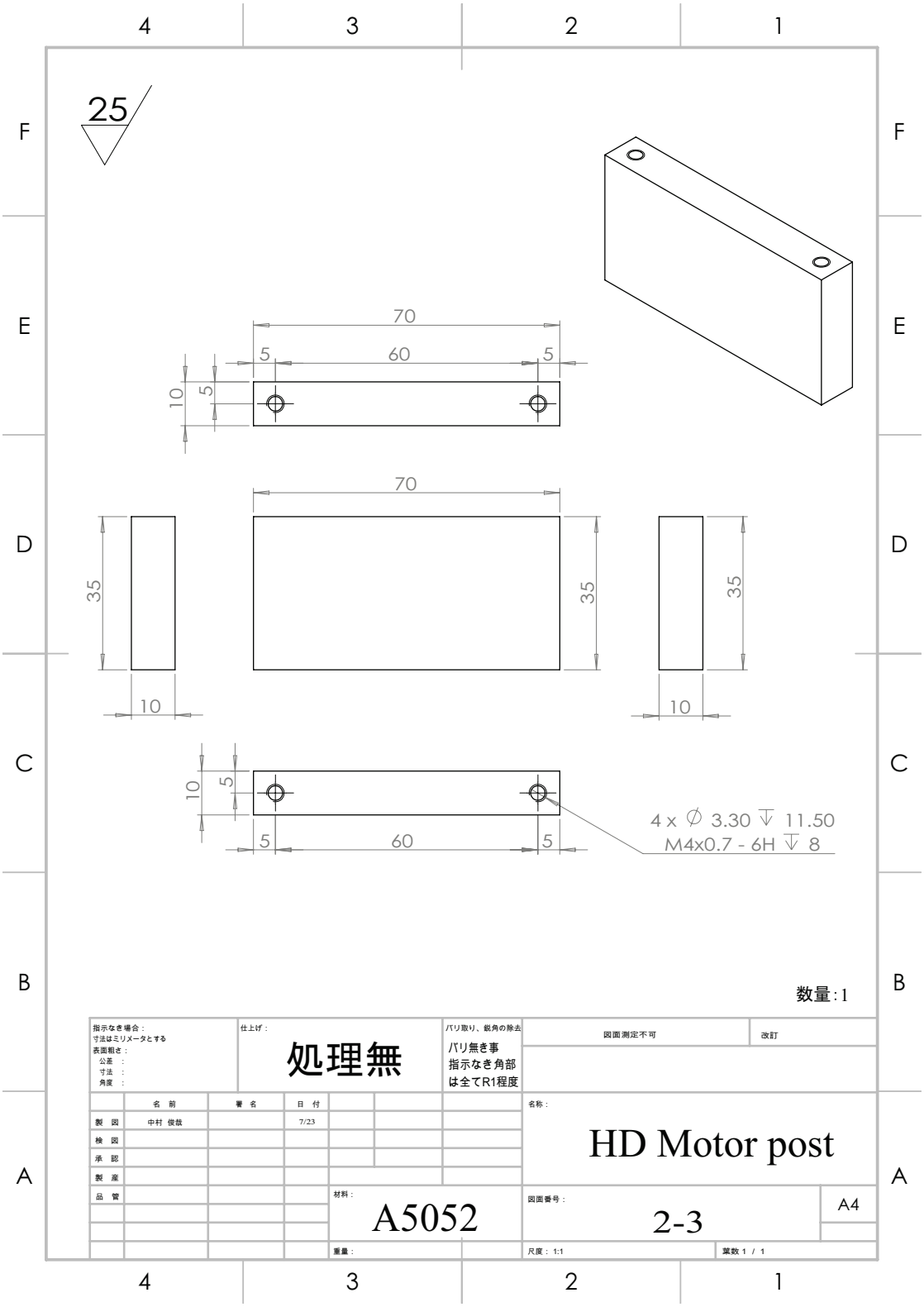


Fig. C-4: Motor post

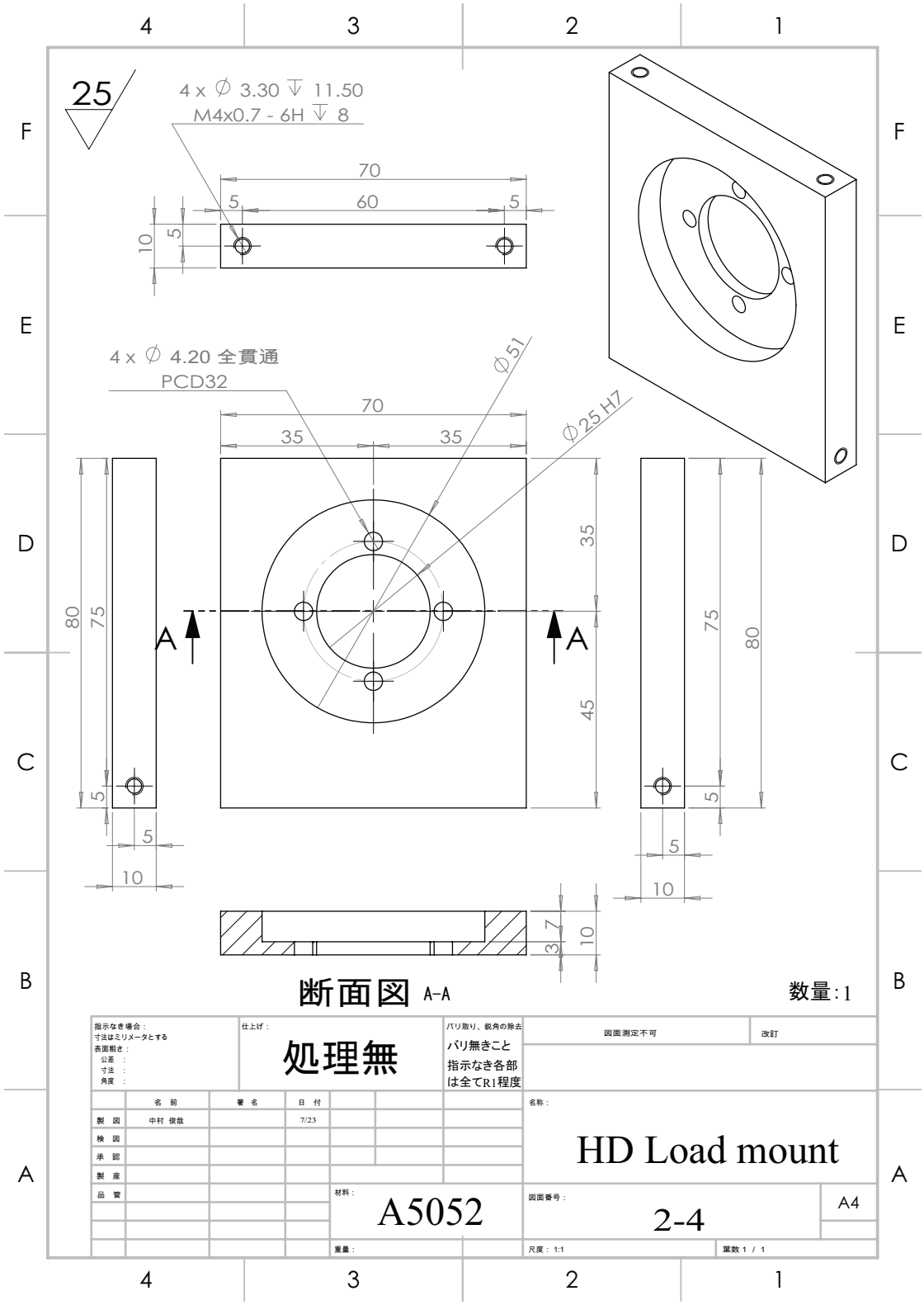


Fig. C-5: Load mount

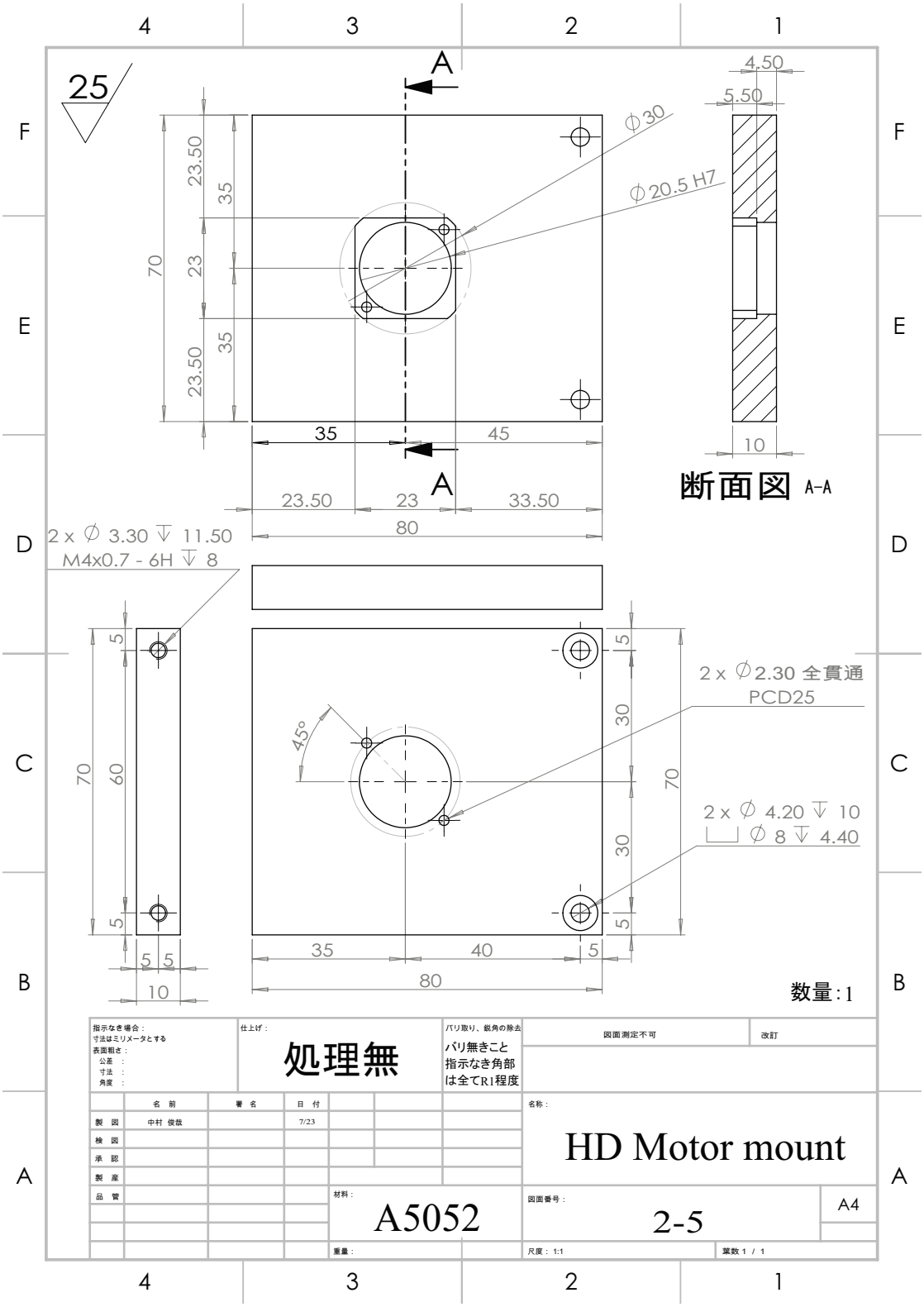


Fig. C-6: Motor mount

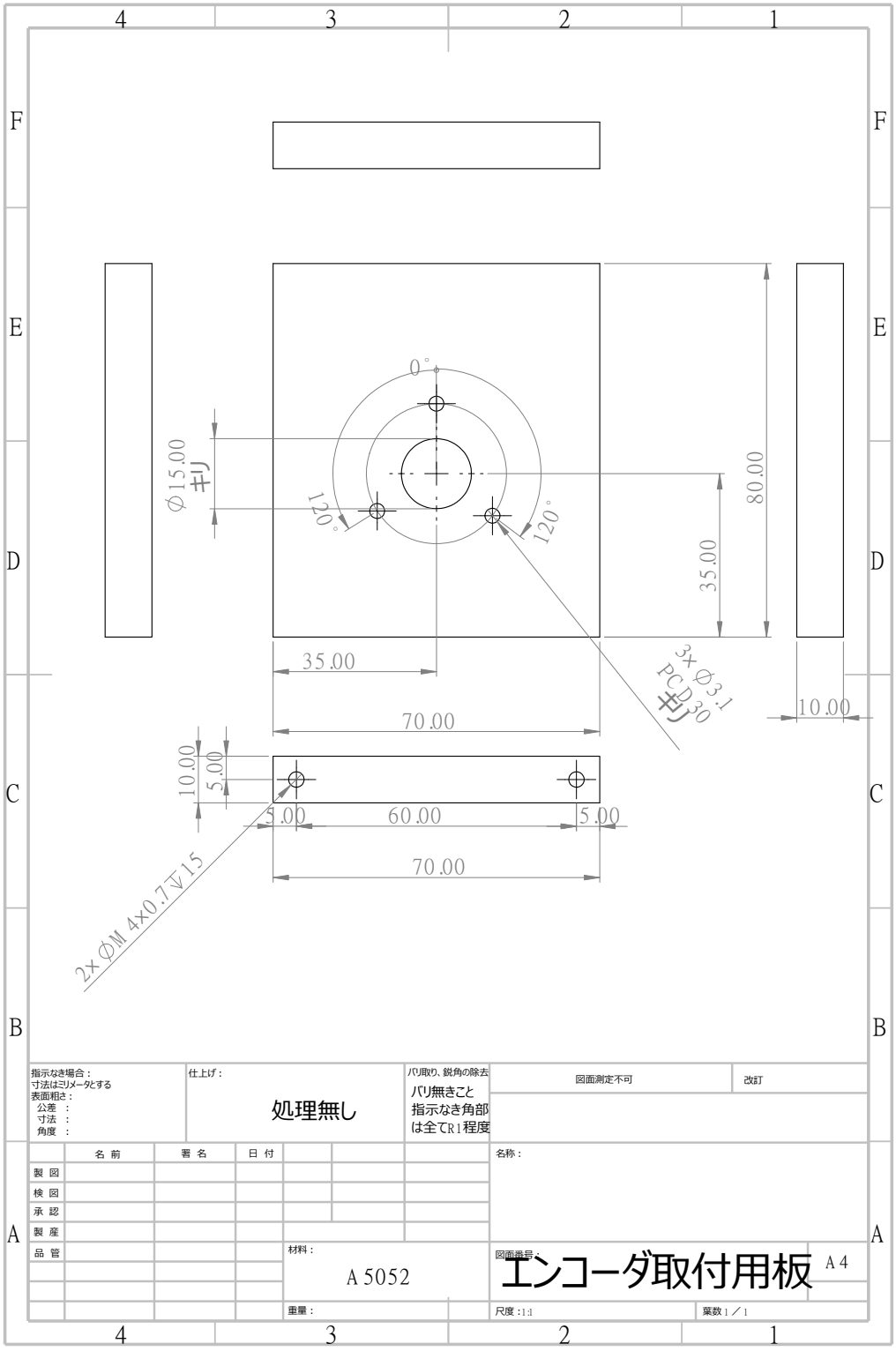


Fig. C-7: Encoder mount

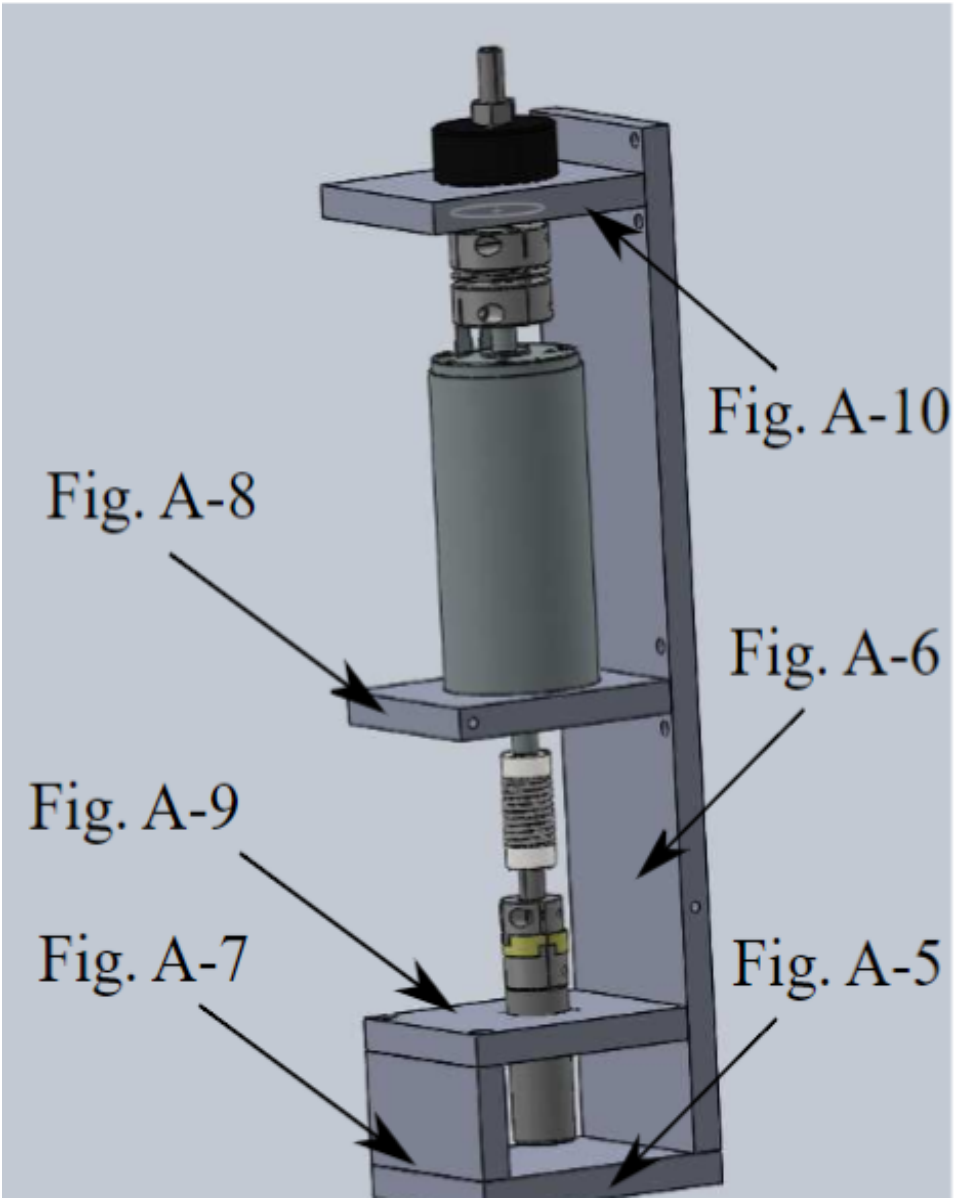


Fig. C-8: Image capture of SolidWorks

Appendix D

動作手順

D.1　シミュレーション

D.1.1　ばね特性の同定

1. cygwin で [gcc user.c] → [./a] と入力しシミュレーションデータを [data.dat] として出力する。
2. gnuplot で実験により同定したばね特性のグラフとシミュレーションでのばね特性のグラフを同時に重ね合わせて表示する。(シミュレーションでは横軸 joint1.Xm/N ー joint1.Xl，縦軸 taull とする)
3. 実験のグラフとシミュレーションのグラフが一致するようにシミュレーション上のパラメータ (Kg: ばね定数，lbl:BL 幅，ばねの粘性係数) を変化させる。(以降，実験とシミュレーションが一致するまで繰り返す)

Table D.1: シミュレーション後生成される data.dat の変数と変数名の関係

t	x＿cmd	joint1.Xm＿dot＿hat	joint1.tau＿dis＿hat	taull	joint1.Xm/N ー joint1.Xl	joint1.Xl	joint1.Xm/N
時間	駆動側角速度指令値	駆動側角速度推定値	外乱トルク推定値	負荷トルク	ねじれ角度	負荷角度	駆動側角度

D.1.2　駆動側摩擦の同定

1. cygwin で [gcc user.c] → [./a] と入力しシミュレーションデータを [data.dat] として出力する。

2. gnuplot で実験により同定した摩擦特性のグラフとシミュレーションによる摩擦特性のグラフを同時に重ね合わせて表示する。(シミュレーションでは横軸 joint1.Xm __ dot __ hat，縦軸 joint1.tau __ dis __ hat とする。

3. 実験のグラフとシミュレーションのグラフが一致するようにシミュレーション上のパラメータ (τ_{cm}: 静止摩擦係数，d_m: 粘性摩擦係数，A：閾値) を変化させる。(以降，実験とシミュレーションが一致するまで繰り返す)

Table D.2: シミュレーション後生成される data.dat の変数と変数名の関係

t	x __ cmd	joint1.Xm __ dot __ hat	joint1.tau __ dis __ hat	taull	joint1.Xm/N − joint1.Xl	joint1.Xl	joint1.Xm/N
時間	駆動側角速度指令値	駆動側角速度推定値	外乱トルク推定値	負荷トルク	ねじれ角度	負荷角度	駆動側角度

D.1.3 負荷側角度制御

1. user.c を開き，設計パラメータ (k1: 負荷角度偏差比例ゲイン，k2: 負荷角度偏差微分ゲイン，k3: 駆動側角速度比例ゲイン，k4: ねじれトルク FB ゲイン，k5: ねじれ角度 FB ゲイン, k5: ねじれ角度 FB ゲイン, k6: ねじれトルク比例ゲイン, k7: ねじれトルク微分ゲイン, k8: ねじれ角度 FB ゲイン, g2: ハイパスフィルタのカットオフ周波数) を編集する。

2. 従来法１，2 の検証をしたい場合は [il=0] とし，提案法の場合は [il=0] のコメントアウトを外す

3. 負荷角度指令値を方形波とする場合は [# define h]，正弦波とする場合は [# define s] とする。

4. cygwin で [gcc user.c] → [./a] と入力しシミュレーションデータを [data.dat] として出力する。

5. gnuplot で横軸が時間 t，縦軸が負荷角度 joint1.Xll/M __ PI × 180.0 としてグラフを確認できる。

Table D.3: シミュレーション後生成される data.dat の変数と変数名の関係

t	x_{cmd}/M __ PI × 180.0	joint1.Xll/M __ PI	joint1.tau __ s	joint1.tau __ s __ hat
時間	負荷角度指令値	負荷角度	ねじれトルク	ねじれトルク推定値
(joint1.Xmm/N-pbl)/M __ PI*180.0	joint1.ilr	joint1.dx	joint1.dll	joint1.imr
駆動側角度	負荷側モータへの電流指令	ねじれ角度	負荷トルク	モータ電流

D.2 実験

D.2.1 共通動作

次のような手順を踏んで実機実験を行う。

（実験開始）

1. 負荷側のスライダックを 0V から 48V に設定する。(Fig. D-1 中)

2. 主電源（制御ボックスに設置）を入れる。(Fig. D-1 中)

3. 副電源（制御ボックスから伸びている）を入れる。(Fig. D-1 中)

4. RE50 の電源を入れる。(Fig. D-1 中)

5. RSF-5B-100-E050-C のスイッチ入れる。（サーボオン）(Fig. D-1 中)

6. 「./user」→ Enter キーでプログラム実行。

（実験終了）

1. 「1」→ Enter キーでプログラム停止。data.dat が生成される。

2. RSF-5B-100-E050-C のスイッチオフ。（サーボオフ）(Fig. D-1 中)

3. RE50 の電源オフ。(Fig. D-1 中)

4. 副電源オフ。(Fig. D-1 中)

5. 主電源オフ。(Fig. D-1 中)

6. 負荷側のスライダックを 48V から 0V に設定する。(Fig. D-1 中)

※ 1.〜3. の順に注意。（電源オフの前にプログラムを停止すると電磁モータが逆方向にマックスで回転。）

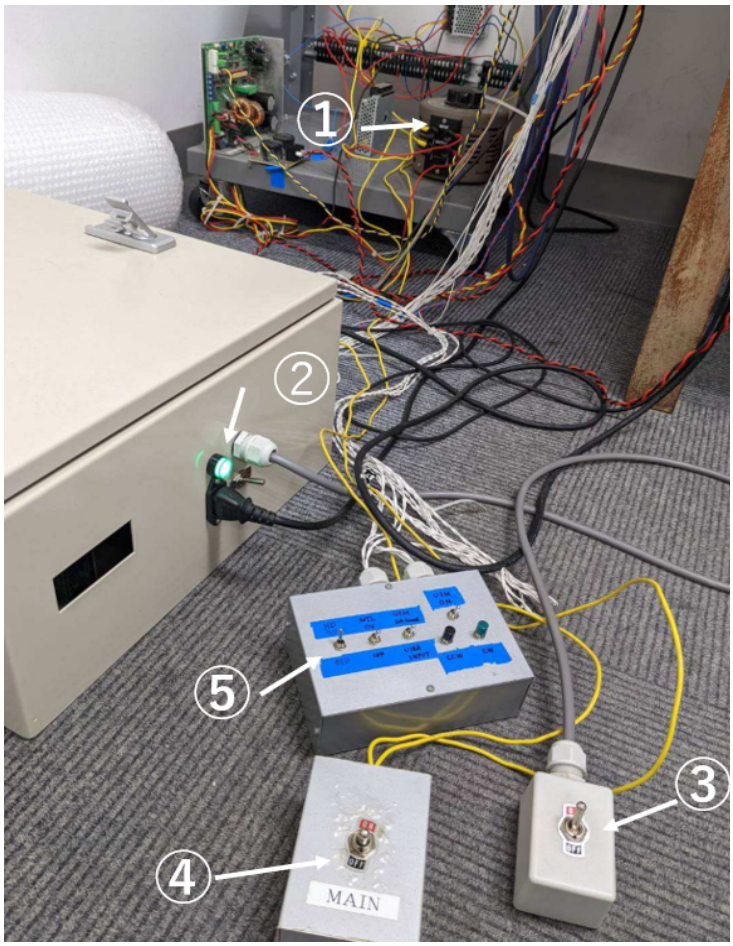


Fig. D-1: 実験開始時操作手順

D.2.2 駆動側の摩擦同定

駆動側モータの摩擦同定時は Fig. D-3 のような状態で実験を行う。パソコンを立ち上げてからは以下のような手順で実験を行う。実験時のパソコンの画面を Fig. D-2 に示す。

1. PC を起動する。
2. user にログインする。(pw：cs1305)
3. ディレクトリを指定。(cd \home \user \ikeda \friction)
4. 「user.h」と「user.c」を開く。(「gedit で開く」のほうが見やすい。)
5. 「user.h」にて制御ゲイン（K_p，K_i) の調整ができる。
   プログラムを編集後保存
6. プログラムが入ったフォルダ内で右クリック→「端末の中に開く」
7. 「make」→ Enter キーでコンパイル
8. 「./user」→ Enter キーでプログラム実行。

駆動側モータが 1 往復したら停止する。実験終了時に「data」のファイルの中に「data.dat」が生成される。実験結果は，gnuplot を開き「plot”data.dat” using 4:7 with lines」と入力することで確認できる。

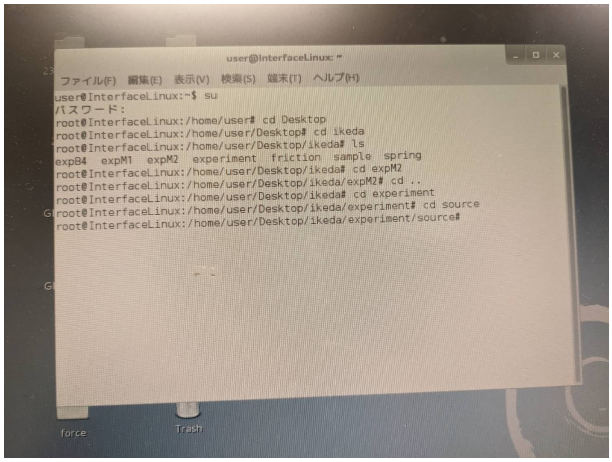


Fig. D-2: 実験時のパソコンの画面

Fig. D-3: 摩擦同定時の実験機

D.2.3 ばね特性の同定

ばね特性の同定時は Fig. D-4 のような状態で実験を行う。特性の同定をしたいばねによて実験機背面のねじの位置を変える。ばねを取り付ける際は六角レンチを使用する。

1. PC を起動する。
2. user にログインする。(pw：cs1305)
3. ディレクトリを指定。(cd \home \user \ikeda \spring)
4. 「user.h」と「user.c」を開く。(「gedit で開く」のほうが見やすい。)
5. 「user.c」にて負荷トルク指令値 τ_l の変更ができる。
   プログラムを編集後保存
6. プログラムが入ったフォルダ内で右クリック→「端末の中に開く」
7. 「make」→ Enter キーでコンパイル
8. 「./user」→ Enter キーでプログラム実行。

実験終了時に「data」のファイルの中に「data.dat」が生成される。実験結果は，gnuplot を開き「plot”data.dat” using 6:8 with lines」と入力することで確認できる。

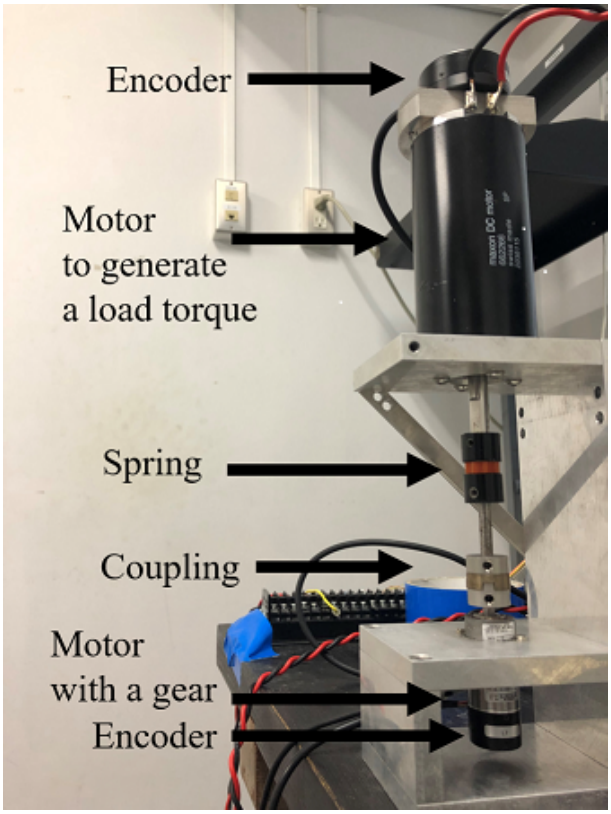


Fig. D-4: ばね特性同定時の実験機

D.2.4 角度制御

負荷側角度制御時は Fig. D-5 のような状態で実験を行う。エンドエフェクタを取り付ける際は六角レンチを使用し，負荷側の慣性を上げるためにエンドエフェクタ先端におもりをつける。

1. PC を起動する。
2. user にログインする。(pw：user)
3. ディレクトリを指定。(cd \home \user \Desktop \ikeda \experiment \source)
4. 「user.cpp」を開く。(「gedit で開く」のほうが見やすい。)
5. 「user.cpp」で従来法と提案法の切りかえや制御器ゲイン (k_1〜k_8，g_2) などの設計パラメータの調整ができる。 プログラムを編集後保存
6. アクセサリから端末を開いて、「source」のディレクトリに入る

7. 「make」→ Enter キーでコンパイル

8. 「./user」→ Enter キーでプログラム実行。

実験終了時に「data」のファイルの中に「data.dat」が生成される。実験結果は，gnuplot を開き「plot”data.dat” using ($1-31):3 with lines」と入力することで確認できる。「data.dat」中の 1 列目は時間 t，3 列目は負荷側角度 θ_l を表しており，プログラム実行から 31 秒間はバックラッシュ幅の同定を行っているためその時間を t から引いている。 ねじれトルク FB ゲイン K の値は小さい値から実行していく（不安定化する恐れがあるため)。

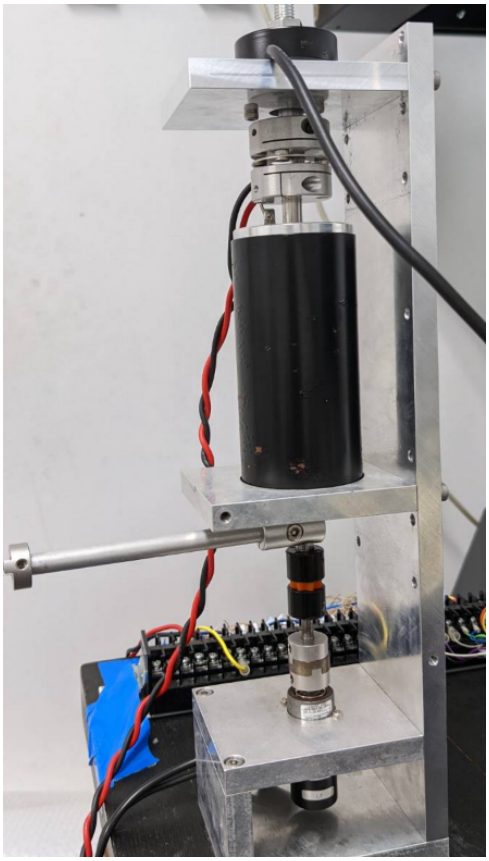


Fig. D-5: 負荷角度制御時の実験機

D.3 解析

1. gain.cpp を開き，最適化したい設計変数の数を「求めたい制御器パラメータ関係」とコメントされているところで決定し、最適化したいパラメータを定義する。

2. 設計パラメータを定義したところから「ゲイン・位相の計算」とコメントされているところの間で制御系全体の伝達関数，評価したい感度関数を記述する。

3. 「constexpr double minPara」で求めたい制御器パラメータの最小値,「constexpr double maxPara」で求めたい制御器パラメータの最大値を決定する。(最小値と最大値を設計変数によって変更したい場合は [k[loopCount]] と記載されている所で調整する。)

4. cygwin で [g++ gain.cpp] → [./a] と入力しシミュレーションデータを [data.dat] として出力する。

5. 「data.dat」の一列目が，そこまでの最小の H2 ノルム (変数名は γ) を表し，プログラムが終了した時点で H2 ノルムが最小の時の行のゲインを最適化ゲインとして用いる。

6. 「Diagram.cpp」を開き，上記と同様に制御系全体の伝達関数を記述する。

7. 「gain.cpp」のプログラムで最適化した設計ゲインを「制御器ゲイン」とコメントされている下で定義する。

8. cygwin で [g++ Diagram.cpp] → [./a] と入力しシミュレーションデータを [data.dat] として出力する。

9. 横軸 frec(data.dat の 1 列目)，縦軸 gainCmd(data.dat の 3 列目) で目標値追従特性に関する感度関数のゲイン線図，横軸 frec(data.dat の 1 列目)，縦軸 gainbl(data.dat の 4 列目) で外乱抑圧特性に関する感度関数のゲイン線図，横軸 realAxis(data.dat の 6 列目)，縦軸 imagAxis(data.dat の 7 列目) でナイキスト線図を記述する。

参考文献

[1] 山田翔太, 藤本博志, 堀洋一, “高分解能エンコーダの適用による駆動側情報を用いない 2 慣性系の制振制御法”, 電気学会論文誌D（産業応用部門誌）, Vol. 135, No. 3, pp. 212–219, 2015.

[2] Zhou, Xingwei, Bo Zhou, and Lan Yang, “Position sensorless control for doubly salient electro‐magnetic motor based on line‐to‐line voltage”, *IET Electric Power Applications*, Vol. 12, pp. 81–90, 2018.

[3] Stijn Derammelaere, Bram Vervisch, Jasper De Viaene, and Kurt Stockman, “Sensorless load angle control for two-phase hybrid stepper motors”, *Mechatronics*, Vol. 43, pp. 6–17, 2017.

[4] Weifeng Liu, Huizhen Wang, Yongjie Wang, Lan Xiao, and Zelei Huang, “Position sensorless control for doubly salient electro-magnetic motor based on the terminal voltage”, *IET Electric Power Applications*, Vol. 13, pp. 2070–2078, 2019.

[5] Aravind M.A., Dinesh N.S., and K. Rajanna, “Application of EMPC for precise position control of DC-motor system with Backlash”, *Control Engineering Practice*, Vol. 100, p. 104422, 2020.

[6] Xuemei Ren, Dongwu Li, Guofa Sun, and Wei Zhao, “ESO-BASED ADAPTIVE ROBUST CONTROL OF DUAL MOTOR DRIVING SERVO SYSTEM”, *Asian Journal of Control*, Vol. 18, No. 6, pp. 2358–2365, 2016.

[7] 前川明寛, 小久保賢人, “2 モータ式ノーバックラッシュ駆動制御システム（モータ粘性抵抗差の

影響に関する解析及び実験）”, 日本機械学会論文集, Vol. 88, No. 913, pp. 22–00156–22–00156, 2022.

[8] 鄭聖熹, 奥野耕平, “ダブルモータ駆動方式における出力監視型安全関連部の設計”, 設計工学, Vol. 51, No. 12, pp. 867–874, 2016.

[9] Tao Tang, Sisi Chen, Xuanlin Huang, Tao Yang, and Bo Qi, “Combining Load and Motor Encoders to Compensate Nonlinear Disturbances for High Precision Tracking Control of Gear-Driven Gimbal”, *Sensors*, Vol. 18, No. 3, 2018.

[10] Shen Zhao and Zhiqiang Gao, “AN ACTIVE DISTURBANCE REJECTION BASED APPROACH TO VIBRATION SUPPRESSION IN TWO-INERTIA SYSTEMS”, *Asian Journal of Control*, Vol. 15, No. 2, pp. 350–362, 2012.

[11] Wei Zhao, Xuemei Ren, and Linwei Li, “SYNCHRONIZATION AND TRACKING CONTROL FOR DUAL-MOTOR DRIVING SERVO SYSTEMS WITH FRICTION COMPENSATION”, *Asian Journal of Control*, Vol. 21, No. 2, pp. 674–685, 2019.

[12] Tran Vu Trung and Makoto Iwasaki, “Robust 2-DoF Controller Design using H∞ Synthesis for Flexible Robots”, *IEEJ Journal of Industry Applications*, Vol. 8, No. 4, pp. 623–631, 2019.

[13] Rodolfo Haber Guerra, Ramn Quiza, Alberto Villalonga, Javier Arenas, and Fernando Castao, “Digital Twin-Based Optimization for Ultraprecision Motion Systems With Backlash and Friction”, *IEEE Access*, Vol. 7, pp. 93462–93472, 2019.

[14] 内田豊一, 伊藤彰人, 大島達也, 古屋信幸, “複数サーボモータの協調制御による精密位置決め法”, 日本機械学会論文集C編, Vol. 77, No. 778, pp. 2280–2289, 2011.

[15] Kenta Tsuda, Tomoki Sakuma, Kodai Umeda, Sho Sakaino, and Toshiaki Tsuji, “Resonance-suppression Control for Electro-hydrostatic Actuator as Two-inertia System”, *IEEJ Journal of Industry Applications*, Vol. 6, No. 5, pp. 320–327, 2017.

[16] Motonobu Aoki, Hiroshi Fujimoto, Yoichi Hori, and Taro Takahashi, “Robust resonance suppression control based on self resonance cancellation disturbance observer and application to humanoid robot”, *IEEJ Transactions on Industry Applications*, Vol. 134, No. 4, pp. 376–383, 2014.

[17] Yusuke Kawai, Yuki Yokokura, Kiyoshi Ohishi, and Toshimasa Miyazaki, “Smooth Human Interaction Control using Torsion Torque Controller and Motor-side Normalization Compensator Focusing on Back-forward Drivability”, *IEEJ Journal of Industry Applications*, Vol. 8, No. 2, pp. 322–333, 2019.

[18] Rastko R. Selmic and Frank L. Lewis, “Neural net backlash compensation with Hebbian tuning using dynamic inversion”, *Automatica*, Vol. 37, No. 8, pp. 1269–1277, 2001.

[19] Kenta Araake, Sho Sakaino, and Toshiaki Tsuji, “Design of Resonance Ratio Control with Relative Position Information for Two-inertia System”, *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 648–653, 2019.

[20] Baofang Wang, Makoto Iwasaki, and Jinpeng Yu, “Finite-Time Command-Filtered Backstepping Control for Dual-Motor Servo Systems With LuGre Friction”, *IEEE Transactions on Industrial Informatics*, Vol. 19, No. 5, pp. 6376–6386, 2023.

[21] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, and Satoshi Komada, “Design of a Torsion Torque Estimator that Includes a Backlash Model for a Load-Side Angle Control System that Consists of a Motor, a Reduction Gear, a Spring, and Motor/Load-Side Encoders”, *Proceedings of the 48th Annual Conference of the IEEE Industrial Electronics Society*, 2022.

[22] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, and Satoshi Komada, “Angle Controller for Motor with Reduction Gear that Utilizes Torsional Torque Estimator Including Backlash Model and High-Pass Filter”, *IEEJ Journal of Industry Applications*, Vol. 13, No. 3, 2024.

謝辞

本研究の遂行にあたり，すばらしい研究環境を提供していただき，終始温かい御指導と御助言を賜りました矢代 大祐 助教に心から御礼申し上げます。また，中間発表会，卒論発表会で多くの助言を頂きました駒田 諭 教授，弓場井 一裕 教授，山村 直紀 准教授，小山 昌人 助教に深く感謝いたします。

本研究に関して多大な助言と手厚い始動をしていただいた 近藤 大智 先輩に深く感謝いたします。また，マニピュレータ班として研究を手伝ってくださった 新崎 拓海 先輩， 魚谷 尚志 先輩，高橋 翼 先輩， 野場 匡太郎 君， 大窪 飛雄馬 君， 岡野 凌大 君に深く感謝いたします。

一年を通してモーションコントロール分野の 伴藤 信一郎 君， 西村 仁志 君，には大変お世話になりました。この場を借りて御礼申し上げます。

そして，研究室に配属されてからの 3 年間，苦楽を共にした同期の井 寛人 君のおかげで切磋琢磨し研究に励むことができましたので，感謝の意を表したいと思います。

最後に，6 年間何不自由なく大学生活を送らせて頂いた両親に心より感謝いたします。

令和 6 年 2 月 08 日

研究業績

国内会議論文

[1] 池田遊斗, 矢代大祐, 弓場井一裕, 駒田諭, “駆動側/負荷側エンコーダを有する減速機付き電磁モータのねじれトルク推定にバックラッシュモデルを用いた負荷側角度制御”, 機械学会ロボティクス・メカトロニクス講演会講演論文集, 1A1-G04, Hokkaido, June. 1st-4th, 2022.

[2] 池田遊斗, 矢代大祐, 弓場井一裕, 駒田諭, ”駆動側/負荷側エンコーダを有する減速機付き電磁モータのバックラッシュ幅測定に計測誤差モデルを用いた負荷側角度制御”, 電気・電子・情報関係学会東海支部連合大会講演論文集, F5-3, Online, Aug. 29th30th, 2022.

[3] 池田遊斗, 矢代大祐, 弓場井一裕, 駒田諭, “駆動側/負荷側エンコーダを有する減速機付き電磁モータのねじれトルク推定にバックラッシュモデルと HPF を用いた負荷側角度制御”, 自動制御連合講演会論文集, 2C1-3,Utsunomiya, November, 12 th-13th, 2022

[4] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, Satoshi Komada,”Load Side Angle Control Using High-pass filter with Backlash Model for Torsional Torque Estimation of Electromagnetic Motor with Reduction Gear with Drive and Load Side Encoders” Proceedings of International Symposium for Sustainability by Engineering at MIU,Dec.1st,2022.

[5] 池田遊斗, 矢代大祐, 弓場井一裕, 駒田諭, ”減速機付き電磁モータのねじれトルク推定にバックラッシュモデルとハイパスフィルタを用いた負荷側角度制御器の慣性変動に対する検証”, 電気・電子・情報関係学会東海支部連合大会講演論文集, A5-3, Aichi, Aug. 28th29th, 2023

[6] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, Satoshi Komada,”Load Side Angle Control

by Twin Drive System Using Torsional Angle of Electromagnetic Motor with Reducer with Drive/Load Side Encoder”Proceedings of the International Symposium for Sustainability by Engineering at MIU,Sep.7th,2023.

[7] 池田遊斗, 矢代大祐, 弓場井一裕, 駒田諭, “駆動側/負荷側エンコーダを有する減速機付きダブルモータのねじれトルク推定にバックラッシュモデルを用いた負荷側角度制御”, 自動制御連合講演会論文集, 2D2-5, Sendai, October, 7 th-8th, 2023

[8] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, Satoshi Komada, “Design of Load Side Angle Controller Using Backlash Model for Torsional Torque Estimation of Dual Motor System with Reduction Gear” Proceedings of the International Symposium for Sustainability by Engineering at MIU,Nov.29th,2023.

査読付き国際会議論文

[9] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, Satoshi Komada,“Design of a Torsion Torque Estimator that Includes a Backlash Model for a Load-Side Angle Control System that Consists of a Motor, a Reduction Gear, a Spring, and Motor/Load-Side Encoders,” Proceedings of the 48th Annual Conference of the IEEE Industrial Electronics Society, IECON22-000229, Belgium, Oct.17th20th, 2022

[10] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, and Satoshi Komada, ”Load Side Angle Control Using HPF with Backlash Model for Torsional Torque Estimation of Electromagnetic Motor with Reduction Gear with Drive and Load Side Encoders”, Proceedings of the 9th IEEJ International Workshop on Sensing, Actuation, Motion, Control and Optimization, China, Mar. 24th-26th, 2023

[11] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, Satoshi Komada, “Load-side angle control using backlash model and High-pass filter for torsional torque estimation of Geared Motor

with Drive/Load Side Encoder”, Proceedings of the 48th Annual Conference of the IEEE Industrial Electronics Society, IECON23-000463, Singapore, Oct.16th19th, 2023

[12] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, and Satoshi Komada, ”Load Side Angle Control Using Backlash Model for Torsional Torque Estimation of Double Motor with Reduction Gear”, Proceedings of the 10th IEEJ International Workshop on Sensing, Actuation, Motion, Control and Optimization, Japan, Mar. 2nd-4th, 2024

査読付き原著論文

[13] 近藤大智, 矢代大祐, 池田遊斗, 弓場井一裕, 駒田諭, ”モータ/負荷側エンコーダ及び減速機付きモータを用いたトルク制御系のばね定数最適化”, 電気学会論文誌産業応用部門誌, Vol.143-D, No.1, Jan. 2023.

[14] Yuto Ikeda, Daisuke Yashiro, Kazuhiro Yubai, and Satoshi Komada, ” Angle Controller for Motor with Reduction Gear that utilizes Torsional Torque Estimator including Backlash Model and High-Pass Filter”, IEEJ Journal of Industry Applications, Vol. 13, No. 3,2024(掲載予定)