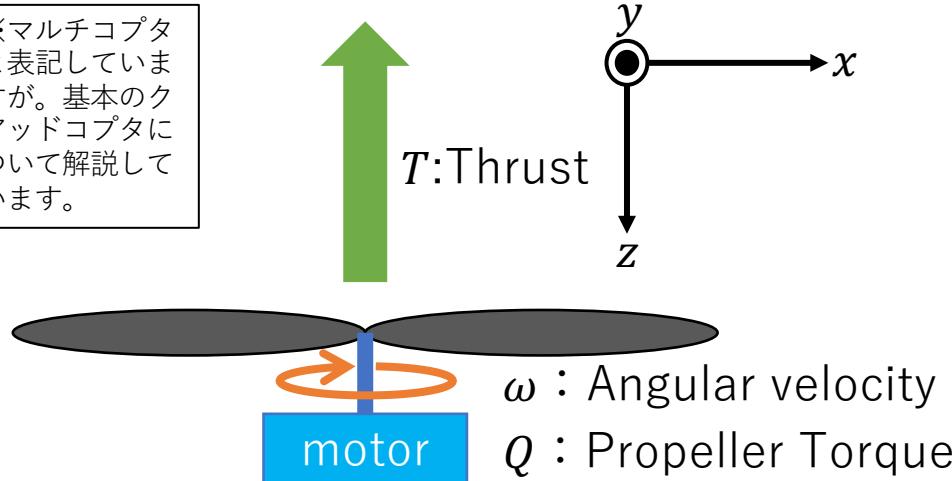


「マルチコプタの運動と制御」基礎のきそ ver 0.22

※マルチコプタと表記していますが。基本のクアッドコプタについて解説しています。



推力・トルクと角速度の関係

$$\begin{aligned} T &= C_T \omega^2 & C_T &:\text{推力係数} \\ Q &= C_Q \omega^2 & C_Q &:\text{トルク係数} \end{aligned} \quad (1)$$

推力とトルクは角速度の2乗に比例します。

モータ+プロペラ系の運動方程式と回路の微分方程式

$$\begin{aligned} J\dot{\omega} + D\omega + C_Q \omega^2 + Q_f &= Ki \\ L\dot{i} + Ri + K\omega &= e \end{aligned}$$

マルチコプタのモータはBLDCモータが一般的ですがブラシ付きモータと見做します。

J :慣性モーメント [kgm^2]

D :動粘性抵抗係数 [Nms/rad]

Q_f :摩擦トルク [Nm]

K :トルク定数・逆起電圧定数 [Nm/A] [Vs/rad]

L :コイルインダクタンス [H]

R :コイル抵抗 [Ω]

e :印加電圧 [V]

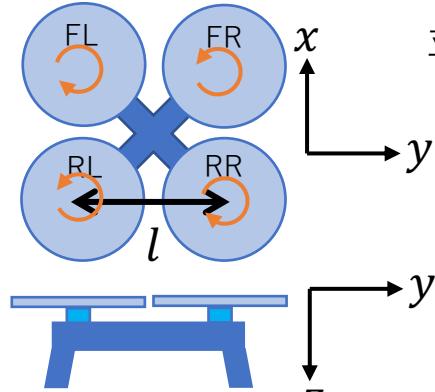
L は非常に小さいので無視すると。

$$i = \frac{e - K\omega}{R} \quad (2)$$

よって、モータ+プロペラ系の回転の運動方程式は次になります

$$J\dot{\omega} + \left(D + \frac{K^2}{R}\right)\omega + C_Q \omega^2 + Q_f = \frac{K}{R}e \quad (3)$$

剛体としての機体の運動方程式



並進運動

$$m(\dot{u} + qw - rv) = -mg \sin \theta \quad (4)$$

$$m(\dot{v} + ru - pw) = mg \cos \theta \sin \phi$$

$$m(\dot{w} + pv - qu) = mg \cos \theta \cos \phi$$

$$-T_{FR} - T_{FL} - T_{RR} - T_{RL}$$

回転運動 (5)

$$I_x \dot{p} + (I_z - I_y)qr = 0.5l(T_{FR} + T_{RR} - T_{FL} - T_{RL})$$

$$I_y \dot{q} + (I_x - I_z)rp = 0.5l(T_{RR} + T_{RL} - T_{FR} - T_{FL})$$

$$I_z \dot{r} + (I_y - I_x)pq = Q_{FR} + Q_{RL} - Q_{FL} - Q_{RR}$$

前後、左右対称なため慣性乗積は0とします。

I_x, I_y, I_z :慣性モーメント [kgm^2]

u, v, w :機体の速度 [m/s]

p, q, r :機体の角速度 [Nm]

m :機体の質量 [kg]

g :重力加速度 [m/s^2]

$T_{FR}, T_{FL}, T_{RR}, T_{RL}$:推力 [N]※下向き正

$Q_{FR}, Q_{FL}, Q_{RR}, Q_{RL}$:トルク [Nm]

ϕ, θ, ψ :オイラー角 [rad]

オイラー角の微分

$$\dot{\phi} = p + q \sin \phi \tan \theta + r \cos \phi \tan \theta$$

$$\dot{\theta} = q \cos \phi - r \sin \phi$$

$$\dot{\psi} = (q \sin \phi + r \cos \phi) / \cos \theta$$

クオータニオンの微分

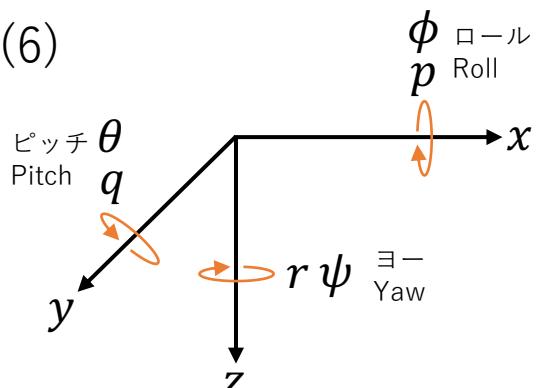
$$\dot{q}_1 = 0.5(rq_2 - qq_3 + pq_4)$$

$$\dot{q}_2 = 0.5(-rq_1 + pq_3 + qq_4)$$

$$\dot{q}_3 = 0.5(qq_1 - pq_2 + rq_4)$$

$$\dot{q}_4 = 0.5(-pq_1 - qq_2 - rq_3)$$

(6)



(7)

慣性座標系→機体座標系の座標変換行列E

(方向余弦行列:DCM) とオイラー角 (8)

$$E = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} = \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \sin \phi \cos \theta \\ \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi & \cos \phi \cos \theta \end{bmatrix}$$

$$\phi = \tan^{-1} E_{23} / E_{33}$$

$$\theta = \tan^{-1} -E_{13} / \sqrt{E_{23}^2 + E_{33}^2} \quad (9)$$

$$\psi = \tan^{-1} E_{12} / E_{11}$$

方向余弦行列とクオータニオン (10)

$$E = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1 q_2 + q_3 q_4) & 2(q_1 q_3 - q_2 q_4) \\ 2(q_1 q_2 - q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 + q_1 q_4) \\ 2(q_1 q_3 + q_2 q_4) & 2(q_2 q_3 - q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}$$

方向余弦行列からクオータニオン (11)

(10)の対角項どうしの和差でクオータニオンを一つ特定できる。その際、0にならないものを選択する。実際は、対角項の全ての組み合わせを試して、求められた物の中から最大のものを選んだのち、他のクオータニオンを以下のように算出する。

先に q_1 を求めた場合

$$q_1 = 0.5\sqrt{1 + E_{11} - E_{22} - E_{33}}$$

$$q_2 = (E_{12} + E_{21}) / 4q_1$$

$$q_3 = (E_{13} + E_{31}) / 4q_1$$

$$q_4 = (E_{23} + E_{32}) / 4q_1$$

先に q_3 を求めた場合

$$q_3 = 0.5\sqrt{1 - E_{11} - E_{22} + E_{33}}$$

$$q_1 = (E_{13} + E_{31}) / 4q_3$$

$$q_2 = (E_{23} + E_{32}) / 4q_3$$

$$q_4 = (E_{12} - E_{21}) / 4q_3$$

先に q_2 を求めた場合

$$q_2 = 0.5\sqrt{1 - E_{11} + E_{22} - E_{33}}$$

$$q_1 = (E_{12} + E_{21}) / 4q_2$$

$$q_3 = (E_{13} + E_{31}) / 4q_2$$

$$q_4 = (E_{23} - E_{32}) / 4q_2$$

先に q_4 を求めた場合

$$q_4 = 0.5\sqrt{1 + E_{11} + E_{22} + E_{33}}$$

$$q_1 = (E_{23} - E_{32}) / 4q_4$$

$$q_2 = (-E_{13} + E_{31}) / 4q_4$$

$$q_3 = (E_{12} + E_{21}) / 4q_4$$

速度の慣性空間への座標変換(オイラー角表現)

$$\begin{aligned} \dot{X}_e &= u \cos \theta \cos \psi + v(\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi) \\ &\quad + w(\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) \\ \dot{Y}_e &= u \cos \theta \sin \psi + v(\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi) \\ &\quad + w(\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) \\ \dot{Z}_e &= -u \sin \theta + v \sin \phi \cos \theta + w \cos \phi \cos \theta \end{aligned} \quad (12)$$

速度の慣性空間への座標変換(クオータニオン表現)

$$\begin{aligned} X_e &= (q_1^2 - q_2^2 - q_3^2 + q_4^2)u + 2(q_1 q_2 - q_3 q_4)v + 2(q_1 q_3 + q_2 q_4)w \\ \dot{Y}_e &= 2(q_1 q_2 + q_3 q_4)u + (-q_1^2 + q_2^2 - q_3^2 + q_4^2)v + 2(q_2 q_3 - q_1 q_4)w \\ \dot{Z}_e &= 2(q_1 q_3 - q_2 q_4)u + 2(q_2 q_3 + q_1 q_4)v + (-q_1^2 - q_2^2 + q_3^2 + q_4^2)w \end{aligned} \quad (13)$$

※(12)(13)は方向余弦行列の逆行列（方向余弦行列の場合転置で済む）と速度ベクトルの積という形で簡単に記述できますが、実際にコーディングするときは展開形式が使いやすいと思い上記のように記述しました。

微小擾乱法による運動方程式等の線形化

本資料では定常状態（平衡状態）の状態量（定常値）に添字0をつけて表現します。マルチコプタのホバリングを想定しているので、多くの定常値は0になります。そうすると、以下のように、各状態量は定常値に微小な変化（擾乱）が増えたものと表現できます。微小な変化には Δ をつけて表記します。

$$\begin{array}{lll} e = e_0 + \Delta e & u = u_0 + \Delta u & p = p_0 + \Delta p \\ \omega = \omega_0 + \Delta \omega & v = v_0 + \Delta v & q = q_0 + \Delta q \\ & w = w_0 + \Delta w & r = r_0 + \Delta r \end{array} \quad \begin{array}{l} \phi = \phi_0 + \Delta \phi \\ \theta = \theta_0 + \Delta \theta \\ \psi = \psi_0 + \Delta \psi \end{array} \quad (14)$$

定常状態では微分値は0となるので各運動方程式から次の釣り合いの方程式が導かれる。

モータ + プロペラ

$$\left(D + \frac{K^2}{R}\right)\omega_0 + C_Q\omega_0^2 + Q_f = \frac{K}{R}e_0$$

機体並進運動

$$\begin{aligned} q_0w_0 - r_0v_0 &= -g \sin \theta_0 \\ r_0u_0 - p_0w_0 &= g \cos \theta_0 \sin \phi_0 \\ m(p_0v_0 - q_0u_0) &= mg \cos \theta_0 \cos \phi_0 \\ &\quad - C_T(\omega_{FR0}^2 + \omega_{FL0}^2 + \omega_{RR0}^2 + \omega_{RL0}^2) \end{aligned}$$

機体回転運動

$$\begin{aligned} (I_z - I_y)q_0r_0 &= 0.5lC_T(\omega_{FR0}^2 + \omega_{RR0}^2 - \omega_{FL0}^2 - \omega_{RL0}^2) \\ (I_x - I_z)r_0p_0 &= 0.5lC_T(\omega_{RR0}^2 + \omega_{RL0}^2 - \omega_{FR0}^2 - \omega_{FL0}^2) \\ (I_y - I_x)p_0q_0 &= C_Q(\omega_{FR0}^2 + \omega_{RL0}^2 - \omega_{FL0}^2 - \omega_{RR0}^2) \end{aligned}$$

オイラー角

$$\begin{aligned} p_0 + q_0 \sin \phi_0 \tan \theta_0 + r_0 \cos \phi_0 \tan \theta_0 &= 0 \\ q_0 \cos \phi_0 - r_0 \sin \phi_0 &= 0 \\ (q_0 \sin \phi_0 + r_0 \cos \phi_0) / \cos \theta_0 &= 0 \end{aligned}$$

ホバリング状態では

$u_0 = v_0 = w_0 = p_0 = q_0 = r_0 = \phi_0 = \theta_0 = \psi_0 = 0$ となり、以上の式から次の関係式が導かれる。

$$\begin{aligned} \omega_{FR0}^2 + \omega_{FL0}^2 + \omega_{RR0}^2 + \omega_{RL0}^2 &= mg/C_T \\ \omega_{FR0}^2 - \omega_{FL0}^2 + \omega_{RR0}^2 - \omega_{RL0}^2 &= 0 \\ -\omega_{FR0}^2 - \omega_{FL0}^2 + \omega_{RR0}^2 + \omega_{RL0}^2 &= 0 \\ \omega_{FR0}^2 - \omega_{FL0}^2 - \omega_{RR0}^2 + \omega_{RL0}^2 &= 0 \end{aligned}$$

よって、これらを連立して、ホバリング時のプロペラ角速度は以下となる。

$$\omega_0 = \omega_{FR0} = \omega_{FL0} = \omega_{RR0} = \omega_{RL0} = \frac{1}{2} \sqrt{\frac{mg}{C_T}}$$

したがって、ホバリング時のモータへの入力電圧は以下となる。

$$e_0 = \frac{DR+K^2}{2K} \sqrt{\frac{mg}{C_T}} + \frac{C_Q R mg}{4 K C_T} + \frac{R}{K} Q_f \quad \text{※この値は全てのモータで同一となる}$$

これまで明らかになった運動方程式等に(14)を代入する。例えばモータ + プロペラであれば、次のようになる。

$$J\dot{\Delta\omega} + \left(D + \frac{K^2}{R}\right)(\omega_0 + \Delta\omega) + C_Q(\omega_0 + \Delta\omega)^2 + Q_f = \frac{K}{R}(e_0 + \Delta e)$$

上式に、釣り合いの式を適用すると

$$J\dot{\Delta\omega} + \left(D + \frac{K^2}{R}\right)\Delta\omega + C_Q(2\omega_0\Delta\omega + \Delta\omega^2) = \frac{K}{R}\Delta e$$

更に、変化量の積や2乗は他の項に比して微小であるので無視すると

$$J\dot{\Delta\omega} + \left(D + \frac{K^2}{R} + 2C_Q\omega_0\right)\Delta\omega = \frac{K}{R}\Delta e$$

※モータ+プロペラは四組ありますが、ここでは1本だけ示します。
必要に応じて場所を示す添字FR,FL,RR,RLをつけ区別します。

以上が、線形化されたモータ+プロペラの運動方程式になります。同様にして他の運動方程式等も線形化します。

並進運動の線形化運動方程式

$$\dot{\Delta u} = -g\Delta\theta$$

$$\dot{\Delta v} = g\Delta\phi$$

$$\dot{\Delta w} = -2C_T\omega_0(\Delta\omega_{FR} + \Delta\omega_{FL} + \Delta\omega_{RR} + \Delta\omega_{RL})/m$$

回転運動の線形化運動方程式

$$I_x\dot{\Delta p} = lC_T\omega_0(\Delta\omega_{FR} + \Delta\omega_{RR} - \Delta\omega_{FL} - \Delta\omega_{RL})$$

$$I_y\dot{\Delta q} = lC_T\omega_0(\Delta\omega_{RR} + \Delta\omega_{RL} - \Delta\omega_{FR} - \Delta\omega_{FL})$$

$$I_z\dot{\Delta r} = 2C_Q\omega_0(\Delta\omega_{FR} + \Delta\omega_{RL} - \Delta\omega_{FL} - \Delta\omega_{RR})$$

オイラー角の線形化微分方程式

$$\dot{\Delta\phi} = \Delta p$$

$$\dot{\Delta\theta} = \Delta q$$

$$\dot{\Delta\psi} = \Delta r$$

位置に関する線形化微分方程式

$$\dot{\Delta X_e} = \Delta u$$

$$\dot{\Delta Y_e} = \Delta v$$

$$\dot{\Delta Z_e} = \Delta w$$



マルチコプタの伝達関数

求めた線形化された運動方程式等を変動量の初期値0でラプラス変換して伝達関数を求めます。通常変数は大文字で表しますが、元の小文字をそのまま使用します。

モータ+プロペラ系

$$\Delta\omega = \frac{K}{JR_s + DR + K^2 + 2RC_Q\omega_0} \Delta e$$

前後運動

$$\Delta u = -\frac{g}{s} \Delta\theta$$

左右運動

$$\Delta v = \frac{g}{s} \Delta\phi$$

上下運動

$$\Delta w = -\frac{2C_T\omega_0/m}{s} (\Delta\omega_{FR} + \Delta\omega_{FL} + \Delta\omega_{RR} + \Delta\omega_{RL})$$

ロールレート

$$\Delta p = \frac{lC_T\omega_0/I_x}{s} (\Delta\omega_{FR} + \Delta\omega_{RR} - \Delta\omega_{FL} - \Delta\omega_{RL})$$

ピッチレート

$$\Delta q = \frac{lC_T\omega_0/I_y}{s} ((\Delta\omega_{RR} + \Delta\omega_{RL} - \Delta\omega_{FR} - \Delta\omega_{FL}))$$

ヨーレート

$$\Delta r = \frac{2C_Q\omega_0/I_z}{s} (\Delta\omega_{FR} + \Delta\omega_{RL} - \Delta\omega_{FL} - \Delta\omega_{RR})$$

さらに運動の伝達モデルにモータ+プロペラ系の伝達モデルを代入して、伝達モデルをまとめていきます。

上下運動

$$\Delta w = -\frac{2C_T\omega_0 K/m}{s(JRs + DR + K^2 + 2RC_Q\omega_0)} (\Delta e_{FR} + \Delta e_{FL} + \Delta e_{RR} + \Delta e_{RL})$$

ロールレート

$$\Delta p = \frac{lC_T\omega_0 K/I_x}{s(JRs + DR + K^2 + 2RC_Q\omega_0)} (\Delta e_{FR} - \Delta e_{FL} + \Delta e_{RR} - \Delta e_{RL})$$

ピッチレート

$$\Delta q = \frac{lC_T\omega_0 K/I_y}{s(JRs + DR + K^2 + 2RC_Q\omega_0)} (-\Delta e_{FR} - \Delta e_{FL} + \Delta e_{RR} + \Delta e_{RL})$$

ヨーレート

$$\Delta r = \frac{2C_Q\omega_0 K/I_z}{s(JRs + DR + K^2 + 2RC_Q\omega_0)} (\Delta e_{FR} - \Delta e_{FL} - \Delta e_{RR} + \Delta e_{RL})$$

伝達関数の分母分子を $DR + K^2 + 2RC_Q\omega_0$ で割り、新たな記号を以下のように定義すると、これらの伝達モデルが少し見やすくなります。

$$\tau = \frac{JR}{DR + K^2 + 2RC_Q\omega_0}$$

$$K_p = \frac{lC_T\omega_0 K}{(DR + K^2 + 2RC_Q\omega_0)I_x}$$

$$K_w = \frac{2C_T\omega_0 K}{(DR + K^2 + 2RC_Q\omega_0)m}$$

$$K_q = \frac{lC_T\omega_0 K}{(DR + K^2 + 2RC_Q\omega_0)I_y}$$

$$K_r = \frac{2C_Q\omega_0 K}{(DR + K^2 + 2RC_Q\omega_0)I_z}$$

以上から、伝達モデルを書き換えると以下になります。

上下運動

$$\Delta w = -\frac{K_w}{s(\tau s + 1)} (\Delta e_{FR} + \Delta e_{FL} + \Delta e_{RR} + \Delta e_{RL})$$

ロールレート

$$\Delta p = \frac{K_p}{s(\tau s + 1)} (\Delta e_{FR} - \Delta e_{FL} + \Delta e_{RR} - \Delta e_{RL})$$

ピッチレート

$$\Delta q = \frac{K_q}{s(\tau s + 1)} (-\Delta e_{FR} - \Delta e_{FL} + \Delta e_{RR} + \Delta e_{RL})$$

ヨーレート

$$\Delta r = \frac{K_r}{s(\tau s + 1)} (\Delta e_{FR} - \Delta e_{FL} - \Delta e_{RR} + \Delta e_{RL})$$

4つの舵面（入力）

以上まで見てきたモデルについては4入力1出力システムに見えますが、以下のように入力として航空機の舵面に見立てた、スロットル、ロール舵、ピッチ舵、ヨー舵を導入すると1入力1出力システムとなり古典制御理論で扱えます。またこれら4つの入力からモータへの入力は一意に決定できます。

$$\text{スロットル} \quad \delta_T = \Delta e_{FR} + \Delta e_{FL} + \Delta e_{RR} + \Delta e_{RL}$$

$$\text{ロール舵} \quad \delta_a = \Delta e_{FR} - \Delta e_{FL} + \Delta e_{RR} - \Delta e_{RL}$$

$$\text{ピッチ舵} \quad \delta_e = -\Delta e_{FR} - \Delta e_{FL} + \Delta e_{RR} + \Delta e_{RL}$$

$$\text{ヨー舵} \quad \delta_r = \Delta e_{FR} - \Delta e_{FL} - \Delta e_{RR} + \Delta e_{RL}$$

※スロットルは舵ではありませんが、話の流れ上舵として話させてもらいます。T,a,e,rの添字記号はスロットル、エルロン、エレベータ、ラダーを表しています

各舵面から各モータ入力算出

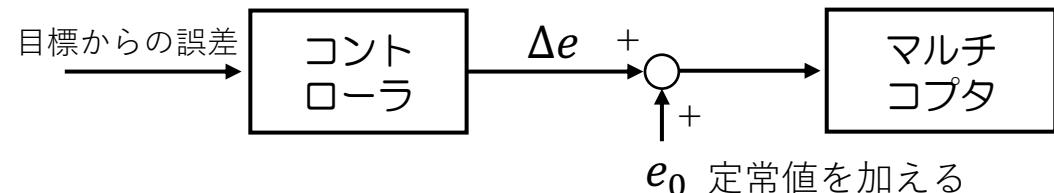
右前モータ $\Delta e_{FR} = (\delta_T + \delta_a - \delta_e + \delta_r)/4$

左前モータ $\Delta e_{FL} = (\delta_T - \delta_a - \delta_e - \delta_r)/4$

右後モータ $\Delta e_{RR} = (\delta_T + \delta_a + \delta_e - \delta_r)/4$

左後モータ $\Delta e_{RL} = (\delta_T - \delta_a + \delta_e + \delta_r)/4$

※これをミキシングと言
います。



マルチコプタの伝達関数（最終）

前後運動

$$\Delta u = -\frac{g}{s} \Delta \theta$$

ロールレート

$$\Delta p = \frac{K_p}{s(\tau s + 1)} \delta_a$$

左右運動

$$\Delta v = \frac{g}{s} \Delta \phi$$

ピッチレート

$$\Delta q = \frac{K_q}{s(\tau s + 1)} \delta_e$$

上下運動

$$\Delta w = -\frac{K_w}{s(\tau s + 1)} \delta_T$$

ヨーレート

$$\Delta r = \frac{K_r}{s(\tau s + 1)} \delta_r$$

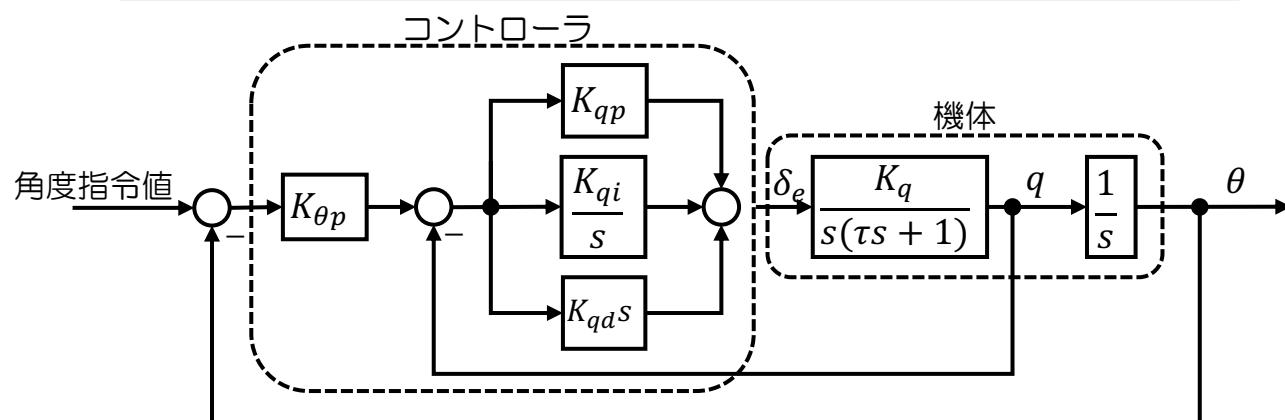
ホバリング状態で線形化しているので、平衡点が違う場合（例えば直進中）などを考える際には並進速度や0角速度は0では無いので線形化された運動方程式の姿も変わり伝達関数の姿も変わってきます。ただし、平衡状態において並進速度のみで角速度が0の場合は角速度に関する伝達関数は上記と同じになります。

制御システム構築の際の注意点

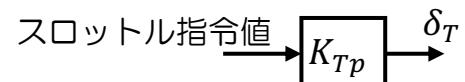
線形化する際に微小擾乱法を用いていますが、以上のモデルで制御系を設計した場合制御される量は変化量になることに注意してください。操作量として得られる量も平衡状態を生み出す定常値からの変化量が算出されます。実際にマルチコプタに入力する際は定常値との和を加える必要があります。通常制御系設計の際に描かれるブロック線図ではこれは示されていないかもしれません。

簡単な姿勢制御システムの一例

姿勢制御システムの簡単な例を以下に示します。インナーループで角速度をPID制御で安定化し、さらにアウターループは比例制御で姿勢角を制御します。以下はピッチ制御のみですが、他の姿勢の制御も基本的には同じです。制御入力は最終的にはミキシングして各モータに分配します。スロットルに関する信号に適切にゲインをかけて δ_T とします。



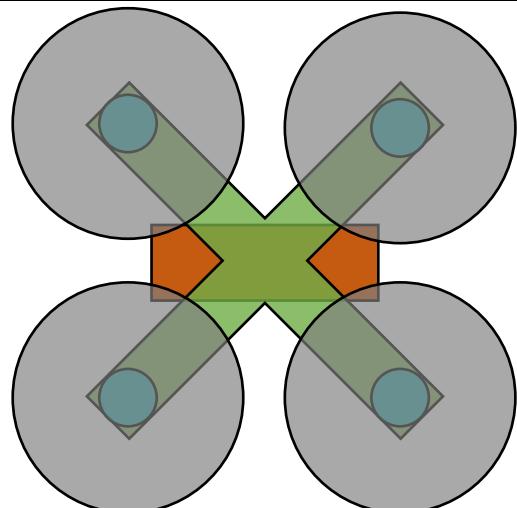
ピッチ角制御系（ロール、ヨーも同様）



スラスト制御系

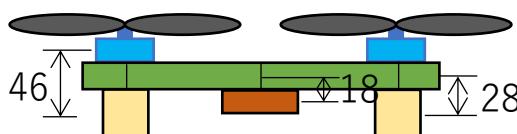
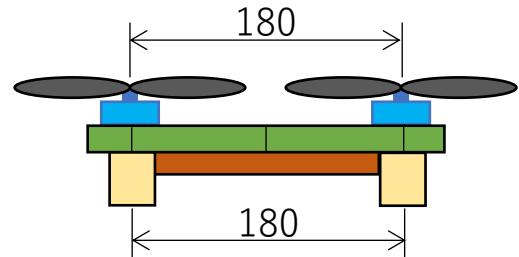
マルチコプタの数値例（ノミナルモデル）

以下に、机上で検討するため、実機を想定して（自分のマルチコプタを測定して）モデルとなる数値例を示したいと思います。



質量

モータ $0.035 \times 4 = 0.140\text{kg}$
プロペラ $0.01 \times 4 = 0.040\text{kg}$
フレーム 0.190kg
脚 $0.02 \times 4 = 0.080\text{kg}$
機体総質量 **0.450kg**
バッテリー 0.260kg
全備重量 **0.710kg**



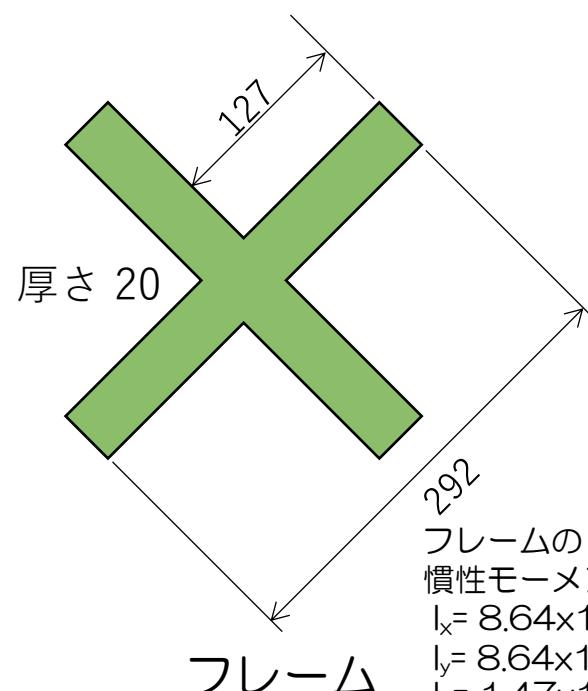
マルチコプタのレイアウト

マルチコプタの慣性モーメント

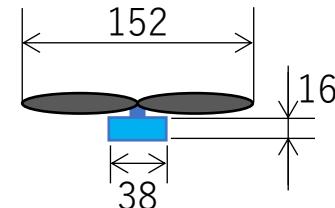
$$\begin{aligned} I_x &= 6.10 \times 10^{-3} \text{ kgm}^2 \\ I_y &= 6.53 \times 10^{-3} \text{ kgm}^2 \\ I_z &= 1.16 \times 10^{-2} \text{ kgm}^2 \end{aligned}$$

※Githubに計算のためのスクリプトを公開しています。

https://github.com/kouhei1970/fundamental_of_multicopter_control/blob/main/Numerical_sample.ipynb



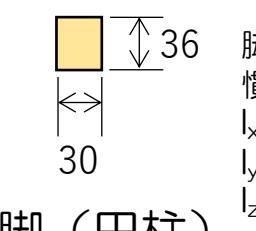
フレーム



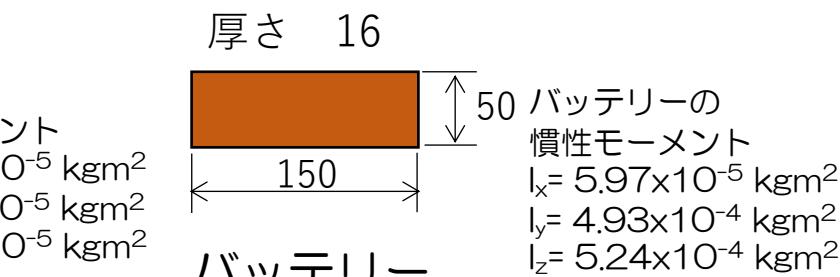
プロペラとモータ

プロペラとモータの慣性モーメント
 $I_x = 5.02 \times 10^{-6} \text{ kgm}^2$
 $I_y = 5.02 \times 10^{-6} \text{ kgm}^2$
 $I_z = 8.12 \times 10^{-6} \text{ kgm}^2$

※プロペラの大きさは無視しています



脚（円柱）



バッテリー

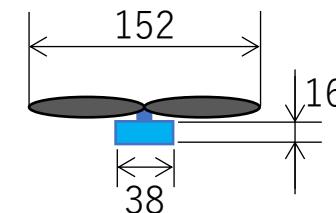
慣性モーメント計算に関するメモ

一様な長方形の慣性モーメント $\frac{1}{12}m(a^2 + b^2)$

一様な円柱の慣性モーメント 円の中心周り $\frac{1}{2}mr^2$

一様な円柱の慣性モーメント 長手方向の中心周り $\frac{1}{4}mr^2 + \frac{1}{12}mL^2$

平行軸の定理 $I = I_{cg} + md^2$



プロペラとモータ

モータのパラメータ

巻線インダクタンス $L : 3.7 \mu\text{H}$
 巷線抵抗 $R : 0.12\Omega$
 トルク定数 $K : 3.28 \times 10^{-3} \text{ Nm/A}$
 動粘性抵抗係数 $D : 0.0$
 摩擦トルク $Q_f : 0.0$

プロペラのパラメータ

推力係数 $C_T : 8.3 \times 10^{-7} \text{ Ns}^2/\text{rad}^2$
 トルク係数 $C_Q : 3.0 \times 10^{-8} \text{ Nms}^2/\text{rad}^2$

モータ+プロペラ系の時定数

$$\tau : 1.93 \times 10^{-2} \text{ s}$$

上下運動のゲイン

$$K_w : 0.524$$

ロール運動のゲイン

$$K_{roll}(K_p) : 10.3$$

ピッチ運動のゲイン

$$K_{pitch}(K_q) : 9.11$$

ヨー運動のゲイン

$$K_{yaw}(K_r) : 2.15$$

以上の数値例から、マルチコプタの伝達関数のパラメータを算出すると左の値が求まります。

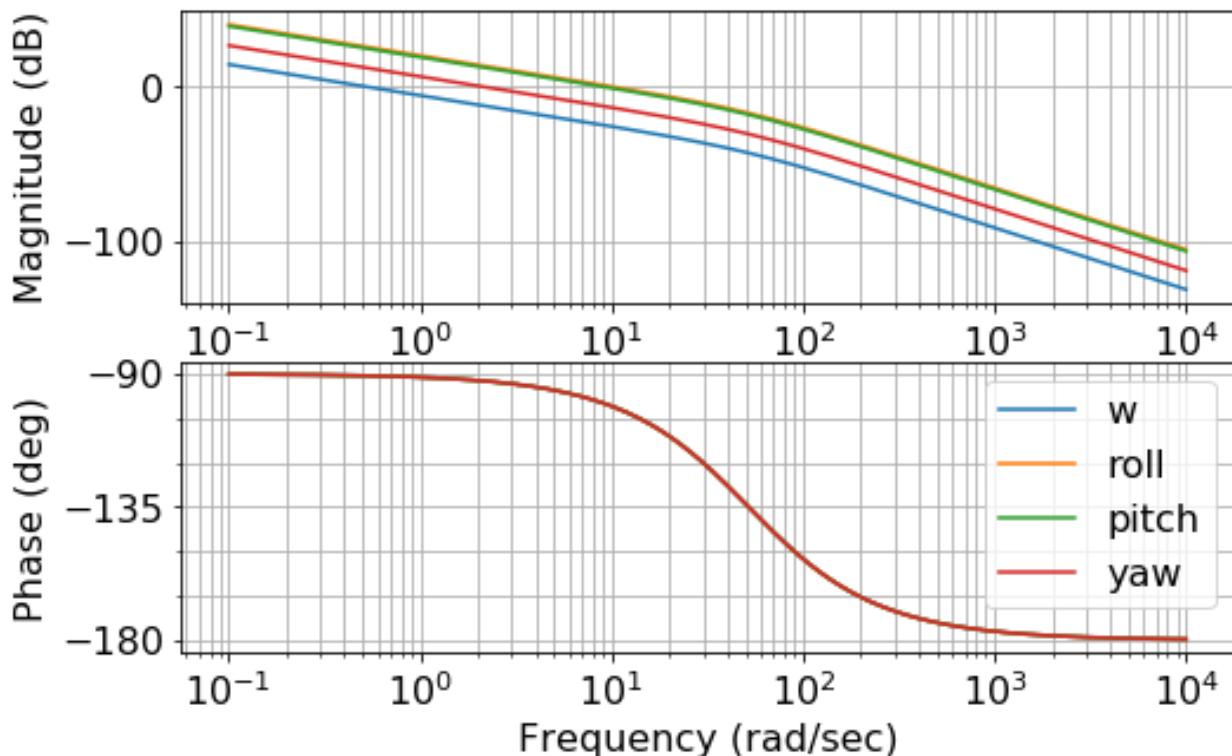
モータとプロペラのトルク応答に関する時定数が約0.02sとなり、その逆数は50Hzなので、少なくともその10倍の周波数500Hzぐらいで制御は回したいところです。

各ゲインは、質量や各運動の軸に関する慣性モーメントが大きくなると小さくなる値です。

それぞれの値が小さいほど、速度が大きくなるので応答が早いように感じられます。

マルチコプタの速度と角速度のボード線図

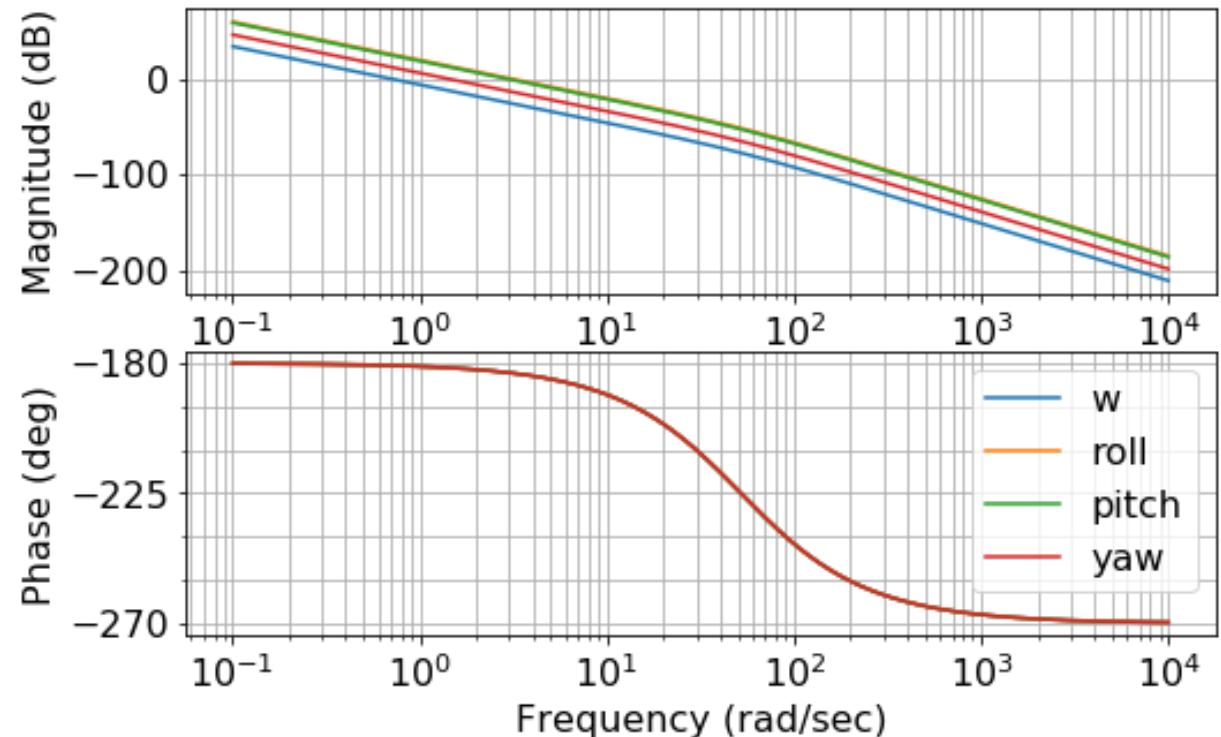
伝達関数のパラメータが決定されたので、それぞれの伝達関数のボード線図を描いてみます。各伝達関数の分母多項式は同一なので、分子は定数ですので、位相曲線は同一のものになっています。ゲインの大きさのみ違うので、ゲイン曲線が上下に平行移動した形です。それぞれのシステムには0の極が存在するので、自ら平衡状態に戻ることはないため、フィードバックによる安定化が必要です。



マルチコプタの速度・角速度系のボード線図

マルチコプタの上下位置と角度のボード線図

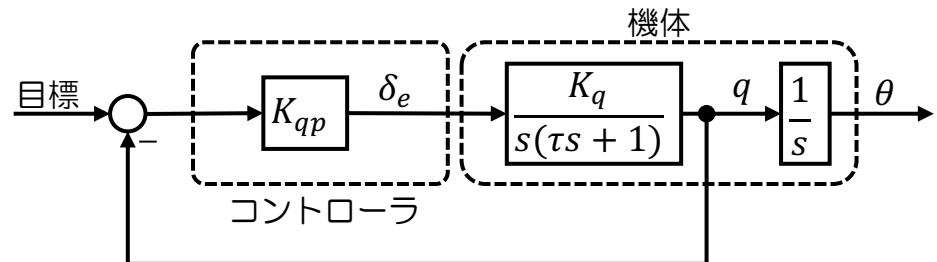
角度に関するボード線図は積分器が二つ入ってくるので、位相余裕は全くなく。単純に角度フィードバックしても、マルチコプタは不安定になってしまいます。PID補償などで、位相を進めてやって、位相余裕を確保して、安定化を図りたいところです。前に示したように、角速度制御ループと角度制御ループの2重ループが有効です。



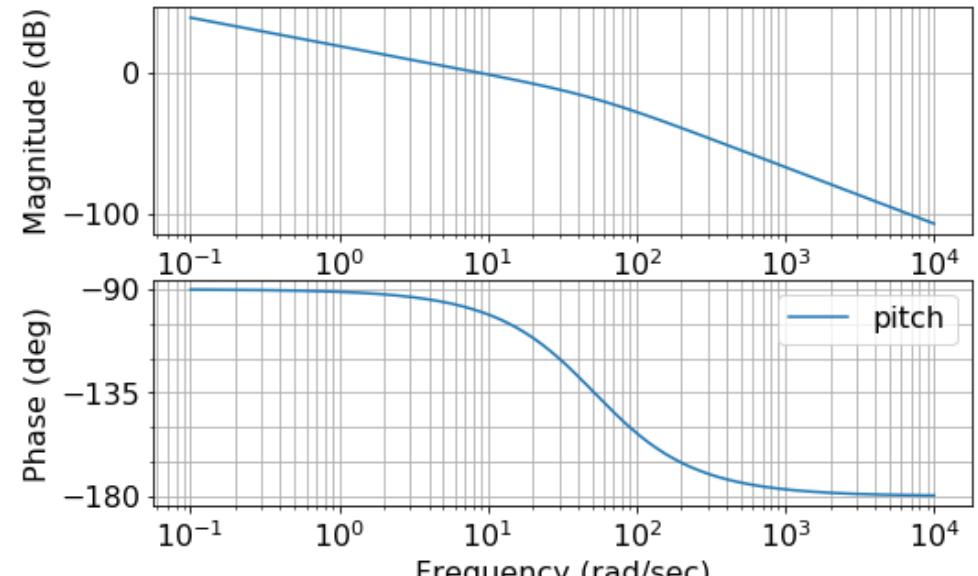
数値例も整ったので、ここから机上制御実験をやっていこうと思います。比例制御からはじめり、PID制御等々について検討と設計のあらましなどについて触れていくと思います。陽にロバスト制御は扱わない予定ですが、平衡点からズレた場合の挙動や無駄時間を考えた場合の挙動などのノミナルモデルとの比較もしてみます。

角速度制御：比例制御のみ

ピッチレート制御系を例にとって、マルチコプタの制御を考え始めてみます。最初は最も単純な比例制御だけを施してみます。後々、問題があることがわかってきますが簡単なものからです。ボード線図を見ると、位相余裕が大変大きく、比例制御だけだとゲイン線図が上に上がるだけなので比例ゲインは相当な大きさまで行けそうです。



比例制御によるピッチレート制御系



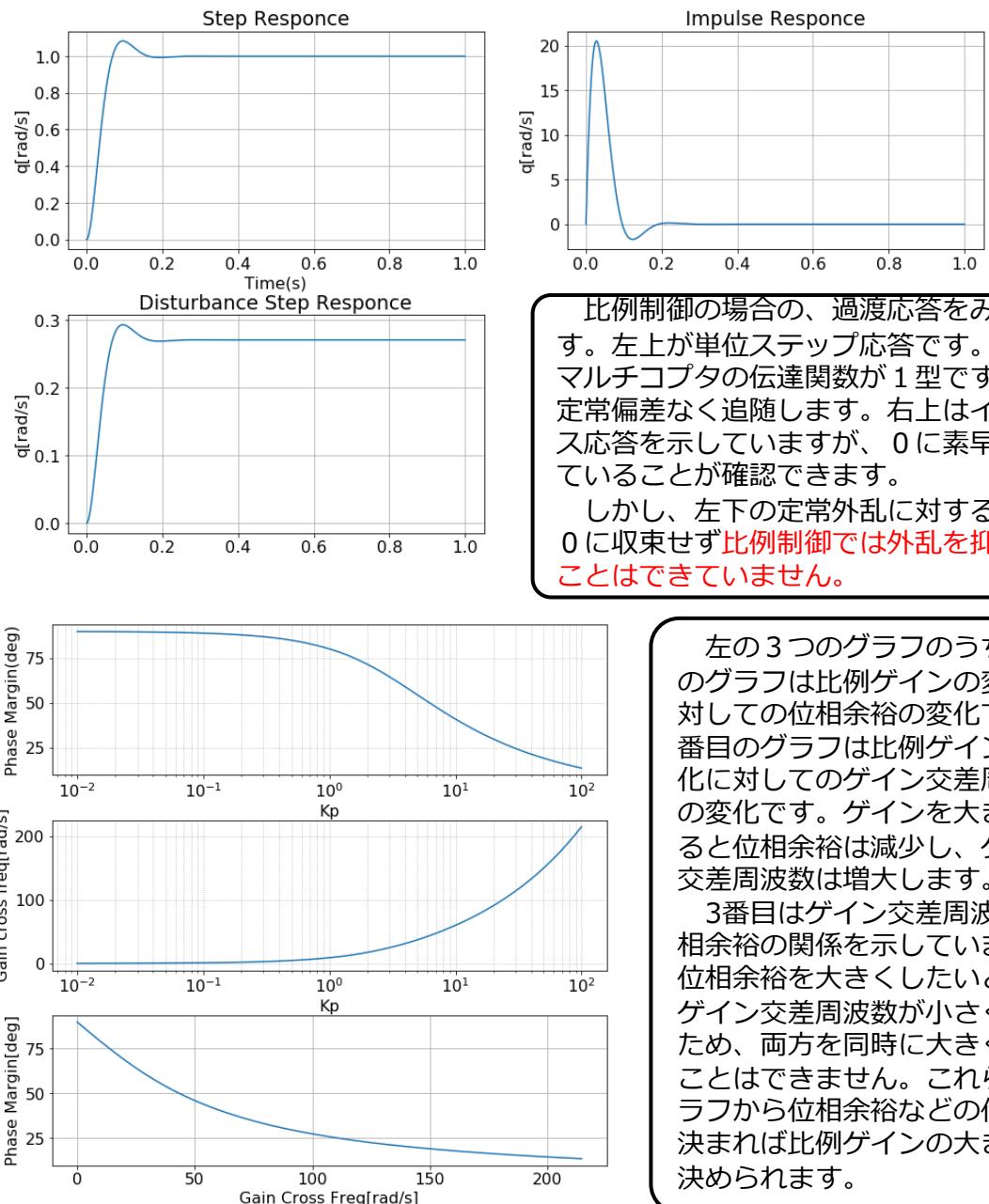
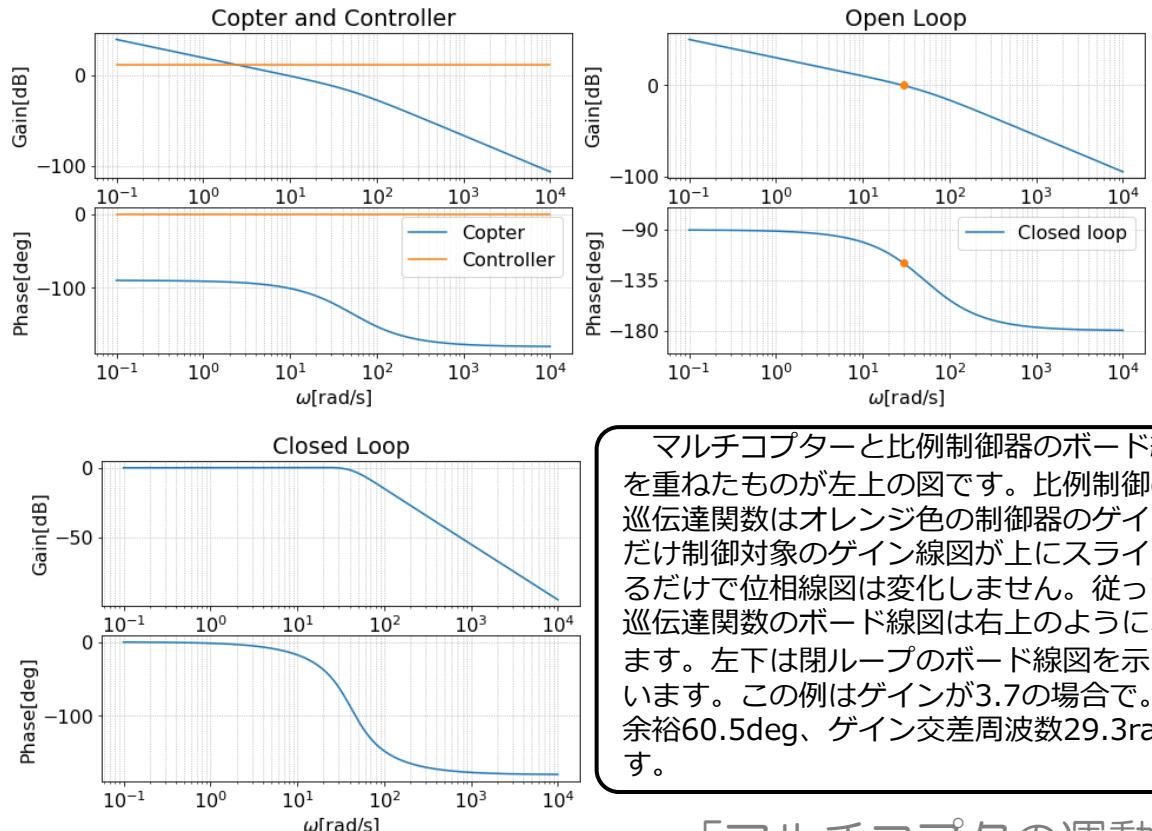
ピッチレート伝達関数のボード線図

比例制御の開ループ伝達関数

$$L(s) = \frac{K_{qp} K_q}{s(\tau s + 1)}$$

以上の様な開ループ伝達関数となります。一見、1型（積分器が一個ある）なので、目標値追随性も良さそうです。比例制御（以下、P制御）だけでも良いかも！と期待させてくれます。まあ、実際はそう簡単には進まないのが残念なところです。

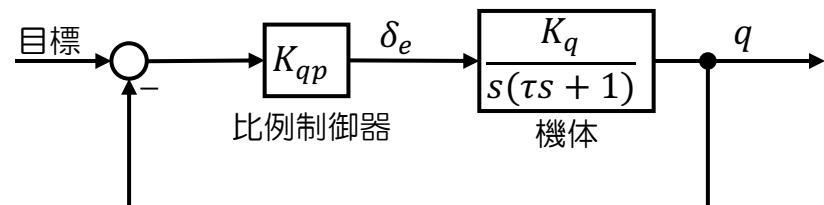
比例制御の設計



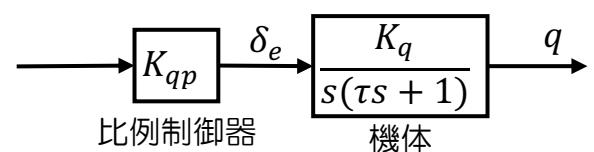
古典制御での設計の勘所

ゲインを調整しながら、時間応答を見て良さそうな値を探しても良いのですが、それだと良い値を見つけるための試行錯誤時間が割と長くなることが多いです。その方法でも、経験を積むと割と勘が働くようになりますが、古典制御では周波数応答の世界であたりをつけてから微修正するというのが良い方法だと思います。

制御対象と制御器を含めた開ループ伝達関数（これを一巡伝達関数と呼びます。）この一巡伝達関数の周波数応答をボード線図で確認して図★の様になる様に形を整えます。これをループ整形と呼びます。各種の制御器とそのゲインの調整で形がどの様に変化するかは法則性があるので、それを基にループ整形します。

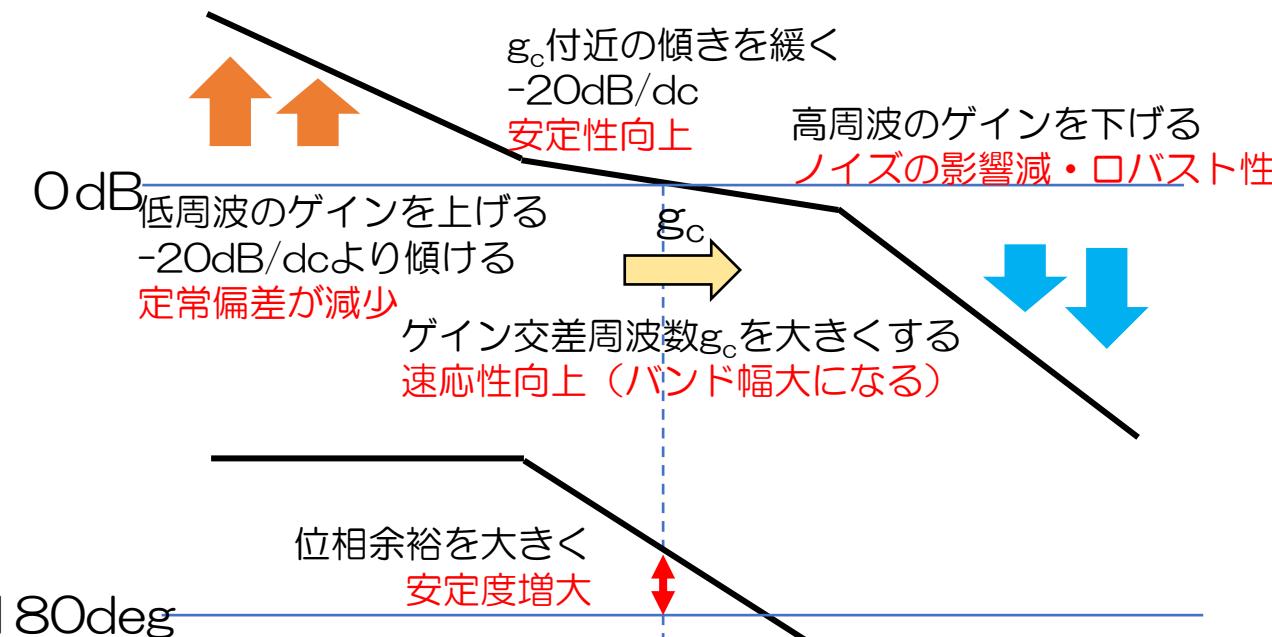


フィードバックを切る



これが一巡伝達関数（開ループ伝達関数）

$$L(s) = \frac{K_{qp} K_q}{s(\tau s + 1)}$$



図★ 望ましいボード線図の形

大雑把にまとめると以下の様になるかと思います

安定性：位相余裕を大きくする

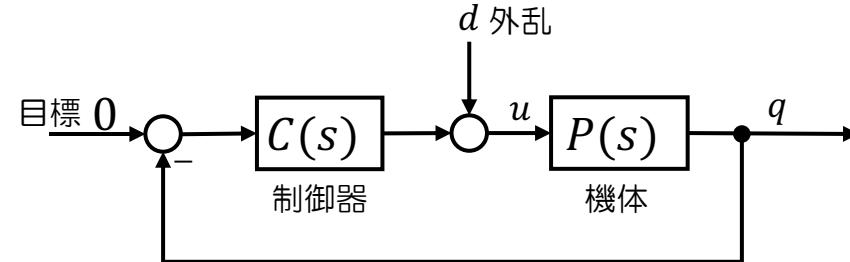
速応性：ゲイン交差周波数を大きくする

定常性：低周波のゲインを上げる

ロバスト性・対ノイズ性：高周波のゲインを下げる

フィードバック制御における外乱の影響

マルチコプタの制御において比例制御では外乱の影響が残る結果となりましたが、フィードバック制御における外乱の影響について簡単にさらいします。



外乱から出力までの伝達関数

$$G_{qd}(s) = \frac{P(s)}{1 + C(s)P(s)}$$

マルチコプタの比例制御の場合

$$\text{コブタ : } P(s) = \frac{K_q}{s(\tau s + 1)}$$

$$\text{比例制御器 : } C(s) = K_{qp}$$

$$G_{qd}(s) = \frac{\frac{K_q}{s(\tau s + 1)}}{1 + \frac{K_{qp}K_q}{s(\tau s + 1)}} = \frac{K_q}{s(\tau s + 1) + K_{qp}K_q}$$

$$G_{qd}(s) = \frac{K_q}{s(\tau s + 1) + K_{qp}K_q}$$

$$q(s) = \frac{K_q}{s(\tau s + 1) + K_{qp}K_q} d(s)$$

外乱が定常外乱の場合

$$q(s) = \frac{K_q}{s(\tau s + 1) + K_{qp}K_q} \frac{1}{s}$$

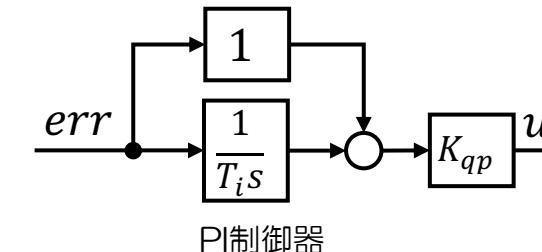
最終値の定理を用いると

$$\begin{aligned} q(\infty) &= \lim_{s \rightarrow 0} s \frac{K_q}{s(\tau s + 1) + K_{qp}K_q} \frac{1}{s} \\ &= \lim_{s \rightarrow 0} \frac{K_q}{s(\tau s + 1) + K_{qp}K_q} \\ &= \frac{1}{K_{qp}} \end{aligned}$$

外乱の影響が残る

マルチコプタの比例積分（PI）制御の場合

制御器に積分器が入るPI制御の場合の外乱の影響



$$\text{PI制御器 : } C(s) = K_{qp} \left(1 + \frac{1}{T_i s} \right)$$

比例ゲイン : K_{qp}

積分時間 : T_i

外乱から出力までの応答

$$q(s) = \frac{K_q s}{T_i s^2 (\tau s + 1) + K_{qp} K_q (\tau s + 1)} d(s)$$

外乱が定常外乱の場合

$$\begin{aligned} q(s) &= \frac{K_q s}{T_i s^2 (\tau s + 1) + K_{qp} K_q (\tau s + 1)} \frac{1}{s} \\ &= \frac{K_q}{T_i s^2 (\tau s + 1) + K_{qp} K_q (\tau s + 1)} \end{aligned}$$

最終値の定理を用いると

$$\begin{aligned} q(\infty) &= \lim_{s \rightarrow 0} s \frac{K_q}{T_i s^2 (\tau s + 1) + K_{qp} K_q (\tau s + 1)} \\ &= 0 \end{aligned}$$

PI制御では外乱の影響は0となる

一見すると、マルチコプタの伝達関数には積分器が含まれているので、定常外乱をも抑制してくれる勘違いしそうですが、比例制御ではステップ入力には定常偏差なく追随してくれますが、定常外乱を0にする効果はありません。ただし、比例ゲインを大きくすることで外乱の影響を小さくする事はできますが、ハイゲインにすることにより制御入力が飽和したりノイズの影響が大きくなったりと限界があります。

そこで、この様に制御器に積分器を含ませることによって、外乱を抑制する効果が現れることがわかります。

角速度制御：比例積分（PI）制御

PI制御の開ループ伝達関数

$$L(s) = \frac{K_{qp} K_q (T_i s + 1)}{T_i s^2 (\tau s + 1)}$$

比例ゲイン : K_{qp}
積分時間 : T_i

PI制御の設計

安定の条件を調べる

一巡伝達関数（開ループ伝達関数） $L(s)$ から、閉ループの伝達関数 $W(s)$ を求め、閉ループの特性方程式を求め、フルビツツの安定判別法から、安定の条件が明らかになるか見てみます。

閉ループ伝達関数

$$W(s) = \frac{L(s)}{1 + L(s)} = \frac{K_{qp} K_q (T_i s + 1)}{T_i \tau s^3 + T_i s^2 + K_{qp} K_q T_i s + K_{qp} K_q}$$

特性方程式

$$T_i \tau s^3 + T_i s^2 + K_{qp} K_q T_i s + K_{qp} K_q = 0$$

フルビツツ行列

$$\begin{bmatrix} T_i & K_{qp} K_q & 0 \\ T_i \tau & K_{qp} K_q T_i & 0 \\ 0 & T_i & K_{qp} K_q \end{bmatrix}$$

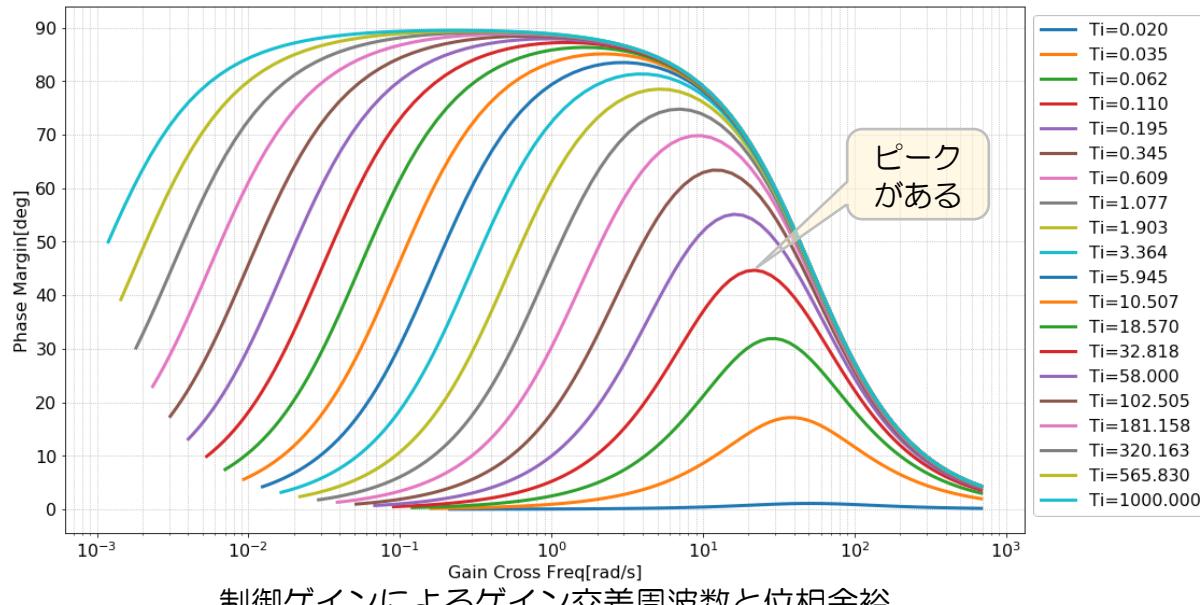
フルビツツの安定判別法を適用していくと、特性方程式の各係数は全て正でなければならないのですが、それは満たされます。次にフルビツツ行列の小行列式を求めていくと次の安定の条件についての結論が得られます。

安定の条件: $T_i > \tau$

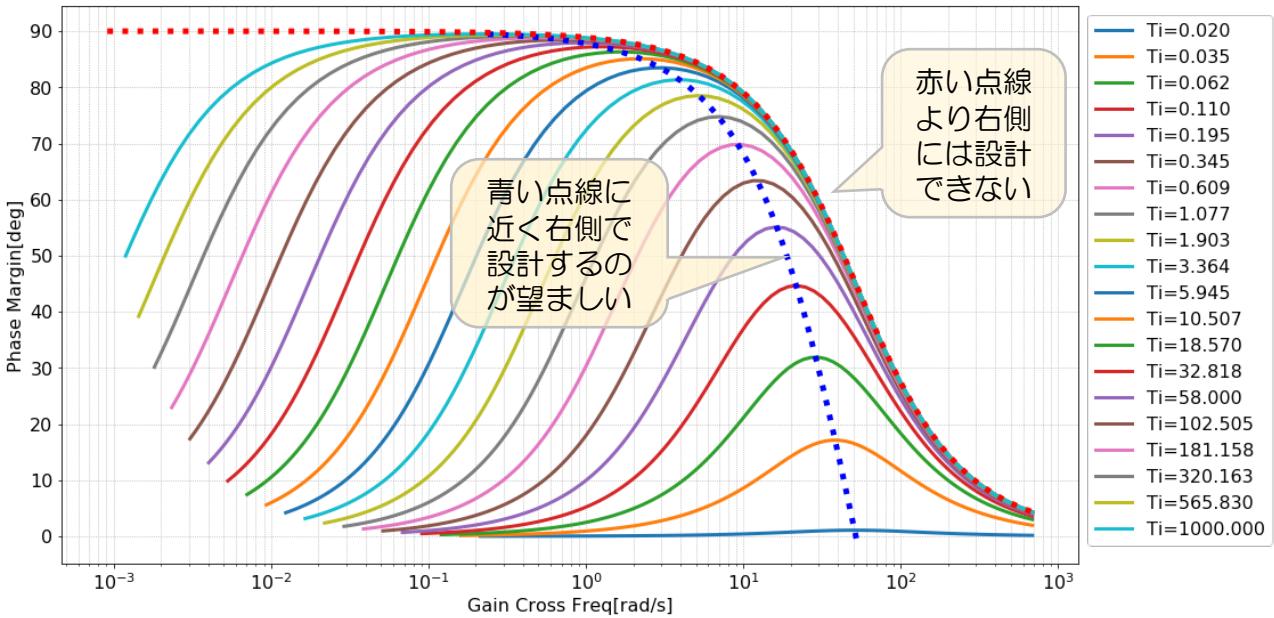
PI制御の積分時間はモータ+プロペラ系の時定数より大きくなければならない

積分時間・比例ゲインとゲイン交差周波数と位相余裕

総当たり的で手計算では不可能ですが、積分ゲインを決めて、比例ゲインを変えていくとゲイン交差周波数と位相余裕がどの様になるかみていきます。ゲイン交差周波数と位相余裕がどちらも大きくなると速応性と安定性が増大しますが、どちらも同時に確実に大きくするには難しく、当てずっぽうではかなりくたびれるので、実際にグラフ化して確認します



積分時間を固定して、比例ゲインを変えていく、その時のゲイン交差周波数と位相余裕を横軸ゲイン交差周波数、縦軸位相余裕でプロットすると、位相余裕の値にピークがある曲線が得られます。ざっくりいうと、ゲイン交差周波数と位相余裕はどちらも大きくとりたいのでピークのところが位相余裕の最適値と言うことになります。ピークのゲイン交差周波数と位相余裕を連ねた線が、将に制御設計の分水嶺と言えそうです。



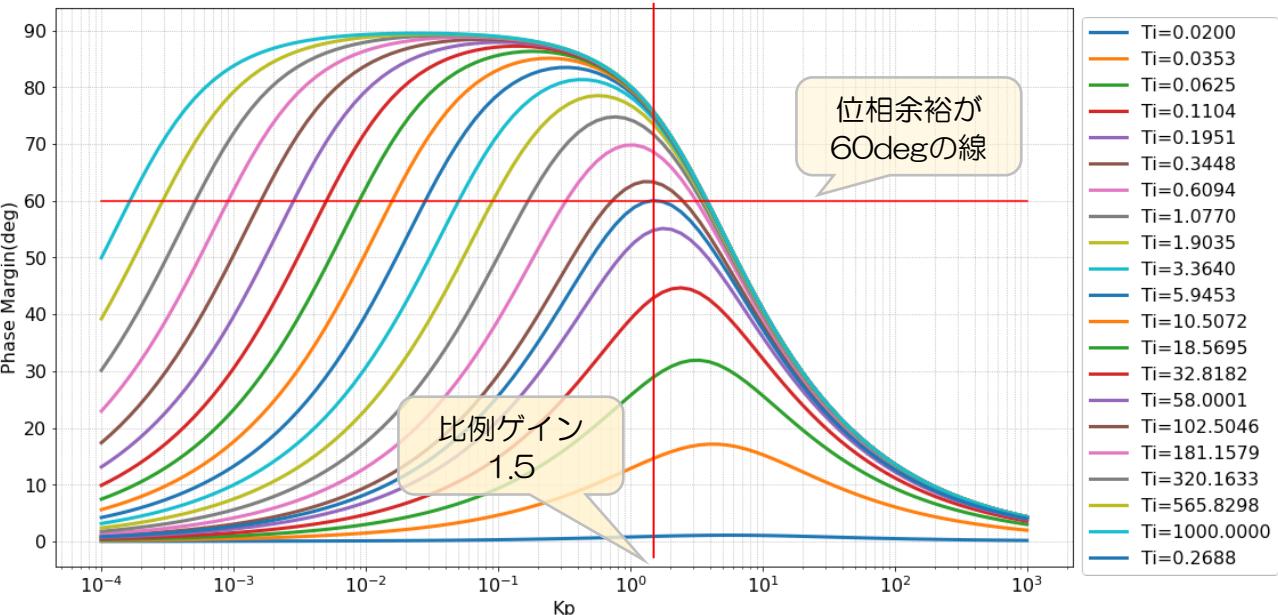
図A ゲイン交差周波数と位相余裕
位相余裕の最適線と比例制御の場合を重ねた図

図Aにおいて、各積分時間における位相余裕のピークを連ねた曲線が**太い青い点線**です。また比例制御においてゲインを変化させた際に、ゲイン交差周波数と位相余裕は**太い赤い点線**の上を移動します。

PI制御において、青い点線よりも左側は、速応性も安定性ももっとよくできる点があることを意味するので選択する事は合理的ではありません。一方の、青い点線よりも右側は安定性が小さくなることを許容して速応性を上げるということになりそうです。また、赤い点線に近いところでは比例制御に性質が似たものとなるためわざわざPI制御をかけて定常性や外乱抑制を期待しているのに、その効果はあまり無いことになりそちらも選択する合理的な理由はありません。

結論的には青い点線に近くかつその右側の領域で、ゲイン交差周波数と位相余裕を満足する比例ゲインと積分時間を見つけることになります。

制御器のゲインの決定



図B 横軸が比例ゲイン縦軸が位相余裕のグラフ
制御器の比例ゲインと積分時間を決定してみましょう。

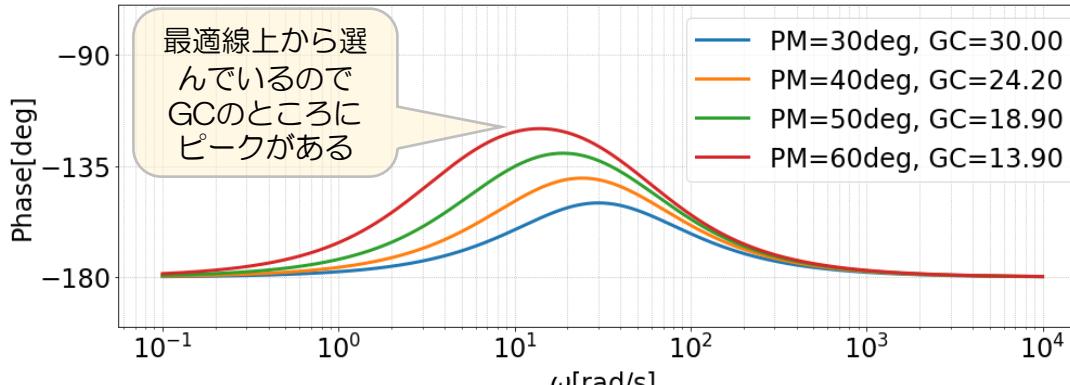
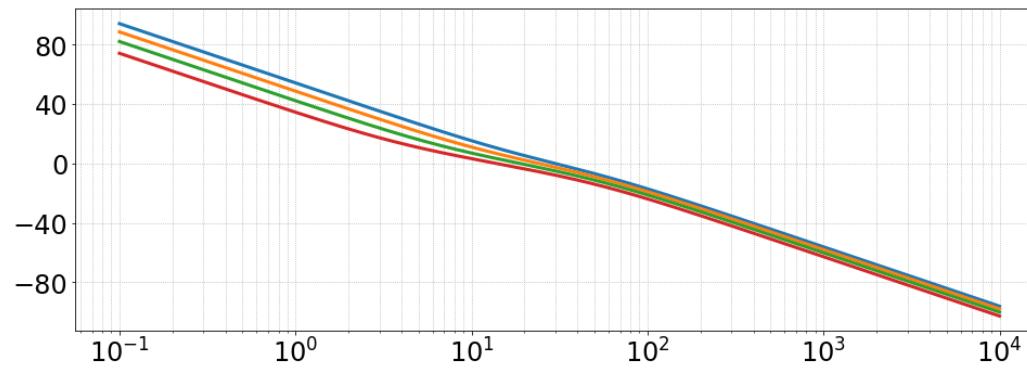
図Aから青い点線の上からゲイン交差周波数と位相余裕の設計値を選びます。ここでは位相余裕を60degとします。そうすると青い点線からゲイン交差周波数は13.9rad/sとグラフから読み取れます。また、積分ゲインは0.27s程度とも分かれます。次に、図Bから位相余裕60degの時の比例ゲインを読み取ります。図Bは図Aと同じ様に見えますが、横軸は比例ゲインになっています。縦軸は図Aと同じ位相余裕です。図Bからは比例ゲインが1.5程度である事が読み取れます。

設計値が以下の様に決まりました。

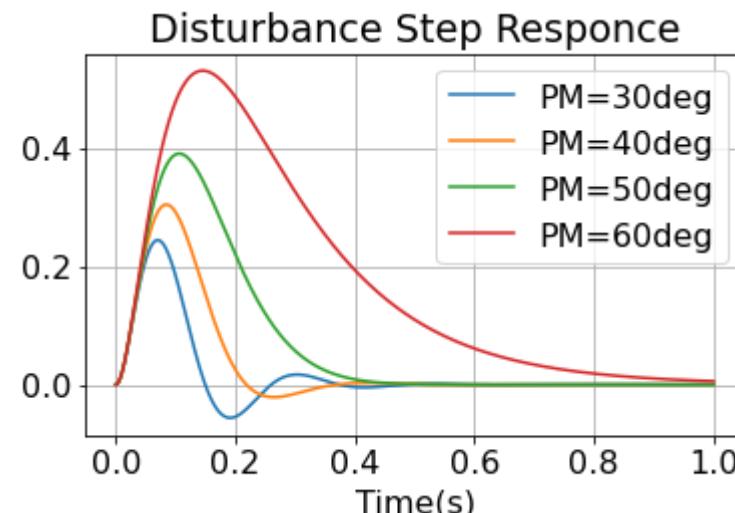
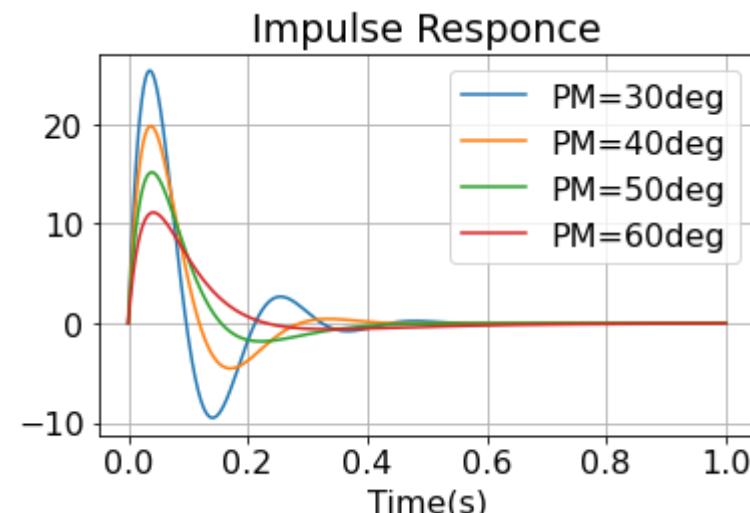
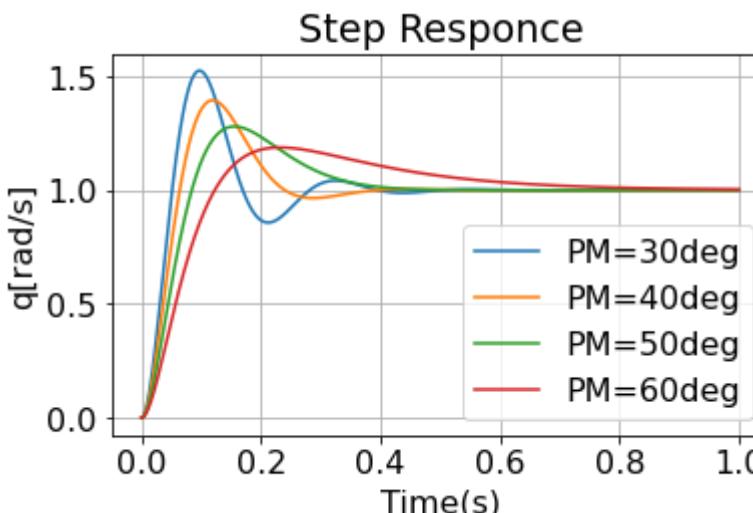
ゲイン交差周波数 : 13.9rad/s
位相余裕 : 60deg
比例ゲイン : 1.5
積分時間 : 0.27s

図Aと図Bの描画について

図Aや図Bについては積分時間を決めて比例ゲインを決めるごとにゲイン交差周波数と位相余裕は求める事ができます。比例ゲインを少しずつ変えると図Aや図Bの1本の曲線が描けます。積分時間を見て、数本の曲線を引くと最適曲線（青い点線）も見えてくるので、グラフから設計値を見出せる様になります。この方法は逆計算や求解アルゴリズムがいらない方法です。



異なるゲインの開ループの周波数特性の比較



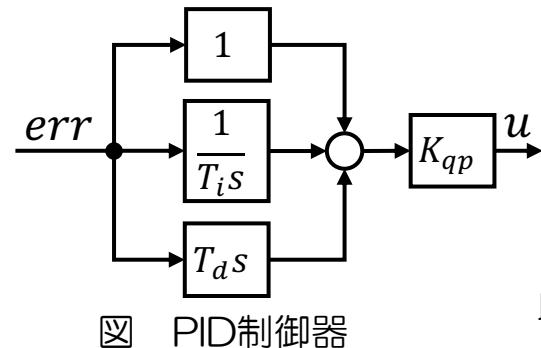
以上から、位相余裕（PM）を30deg、40deg、50deg、60degと決めた際の最適線上の対応するゲイン交差周波数を求め、それに対応する比例ゲインと積分時間を決定し、それぞれの場合の開ループ伝達関数の周波数特性と過渡応答を比較のため重ねてグラフにしてみました。減衰の速さや、オーバーシュートの大きさなどから、PMが40degから50deg程度が妥当なところだと思われます。

位相余裕が大きくなるとオーバーシュートが減少しますが、ゲイン交差周波数は減少し速応性が失われているのが過渡応答から判ります。

また、比例制御と比較すると、外乱を抑制できている事が確認できます。外乱抑制効果は積分時間が小さいほど効果があります。

	位相余裕 deg	ゲイン交差周波数 rad/s	比例ゲイン	積分時間 s
ケース1	30	30	3.3	0.058
ケース2	40	24.2	2.7	0.090
ケース3	50	18.9	2.1	0.15
ケース4	60	13.9	1.5	0.27

角速度制御：PID制御



PID制御器をブロック線図で表現すると図の様になります。この図から伝達関数を求めてみると次の式になります。

$$C(s) = \frac{K_{qp}(T_d s^2 + s + 1/T_i)}{s}$$

比例ゲイン: K_{qp} 微分時間: T_d
積分時間: T_i

開ループ伝達関数

$$L(s) = \frac{K_q K_{qp} (T_d s^2 + s + 1/T_i)}{s^2 (\tau s + 1)}$$

閉ループ伝達関数

$$W(s) = \frac{L(s)}{1 + L(s)} = \frac{K_q K_{qp} (T_d s^2 + s + 1/T_i)}{\tau s^3 + (1 + K_q K_{qp} K_d) s^2 + K_{qp} K_q s + K_{qp} K_q / T_i}$$

安定の条件を調べる

特性方程式

$$\tau s^3 + (1 + K_q K_{qp} K_d) s^2 + K_{qp} K_q s + K_{qp} K_q / T_i = 0$$

フルビツツ行列

$$\begin{bmatrix} 1 + K_q K_{qp} T_d & K_{qp} K_q / T_i & 0 \\ \tau & K_q K_{qp} & 0 \\ 0 & 1 + K_q K_{qp} T_d & K_{qp} K_q / T_i \end{bmatrix}$$

フルビツツの安定条件を求めるときの様になります

安定の条件:

$$T_d > \frac{\tau - T_i}{K_q K_{qp} T_i}$$

各ゲインは正の値となるので、 $T_i > \tau$ の場合はどんな微分時間をとってもマルチコプタの角速度制御系は安定となります。PI制御では $T_i > \tau$ としなければ安定化できませんでしたが、PID制御では $T_i < \tau$ でも T_d が安定条件を満たせば、安定化できることになります。

前述のPI制御において位相余裕を40degにした場合のゲイン設定をそのまま用いて、微分時間を0（微分制御なし）、0.001、0.01、0.1にして計算をしてみた結果を以下に示します。

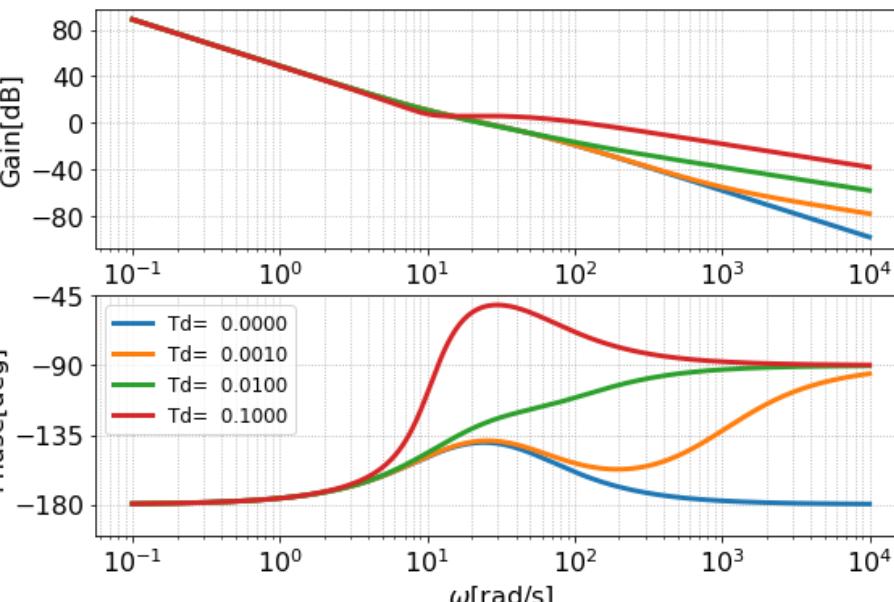
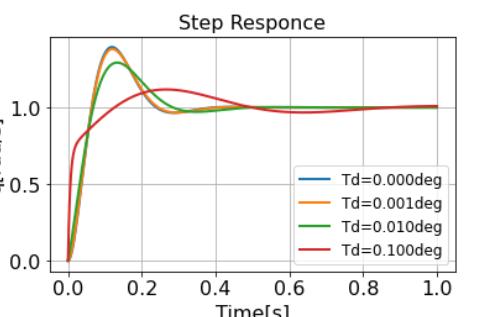
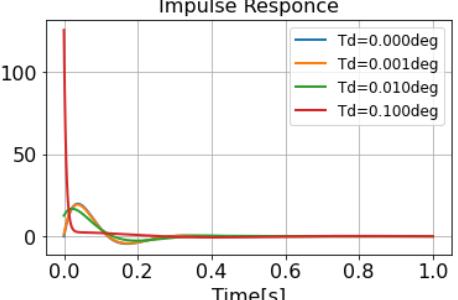


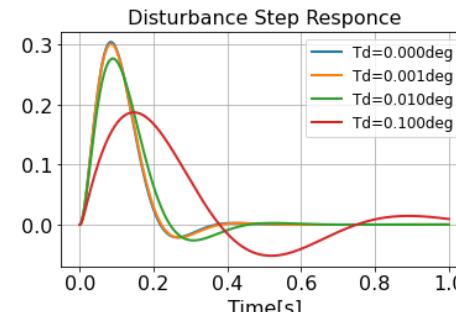
図 PID制御における一巡伝達関数の周波数特性（微分時間変化時）



Step Response



Impulse Response



Disturbance Step Response

図 PID制御における過渡応答（微分時間変化時）

位相余裕とゲイン交差周波数を使ったPID制御系設計

位相余裕（Phase margin）とゲイン交差周波数（Gain crossover frequency）に基づいてマルチコプタのPID制御系の設計を見ていきたいと思います。

位相余裕というのはボード線図で見るとゲイン曲線が0dBを横切る周波数の時の位相が-180degまであとどのくらい余裕があるかを表したものです。

位相余裕とゲイン交差周波数のおさらい

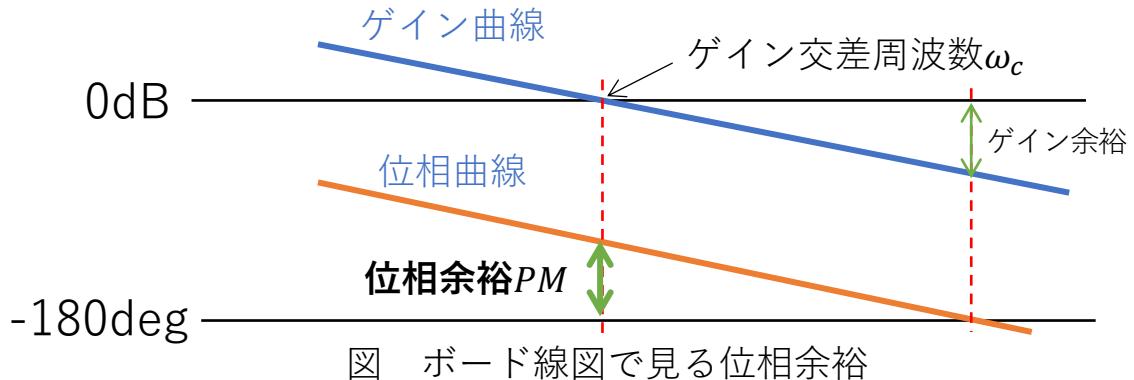


図 ボード線図で見る位相余裕

$$\text{PID制御の一巡伝達関数(開ループ伝達関数)} \\ L(s) = \frac{K_q K_{qp} (T_d s^2 + s + 1/T_i)}{s^2 (\tau s + 1)}$$

一巡伝達関数の周波数伝達関数

$$L(j\omega) = \Delta K_{qp} (u + vj) \quad \text{ここで}$$

$$u = (T_d - \tau) \omega^2 - 1/T_i \quad v = \{(1/T_i - T_d \omega^2) \tau - 1\} \omega$$

$$\Delta = \frac{K_q}{\omega^2 (1 + \omega^2 \tau^2)} \quad \text{となります。}$$

※伝達関数に $j\omega$ を代入して整理するのは結構大変ですが、するとこうなります。

一巡伝達関数のゲイン

$$|L(j\omega)| = \Delta K_{qp} \sqrt{u^2 + v^2}$$

一巡伝達関数の位相

$$\angle L(j\omega) = \tan^{-1} v/u$$

※比例ゲイン K_{qp} については後でそれについて解きたいと思っているのでわざと出しています。

位相余裕はゲイン曲線がボード線図の0dbを横切る周波数の位相を求めるところまで、まずゲインが1（ボード線図はゲインの20logをとったものなので0dBはゲイン1です。）の時の周波数を ω_c とします。

位相余裕はを PM とすると、一巡伝達関数の位相は $PM - \pi$ に等しいことになります

$$\angle L(j\omega_c) = \tan^{-1} v/u = PM - \pi \longrightarrow v/u = \tan(PM - \pi)$$

ここで $\alpha = \tan(PM - \pi)$ としておくと $v = u\alpha$

この式に u, v を入れ直して整理すると次の関係が得られます

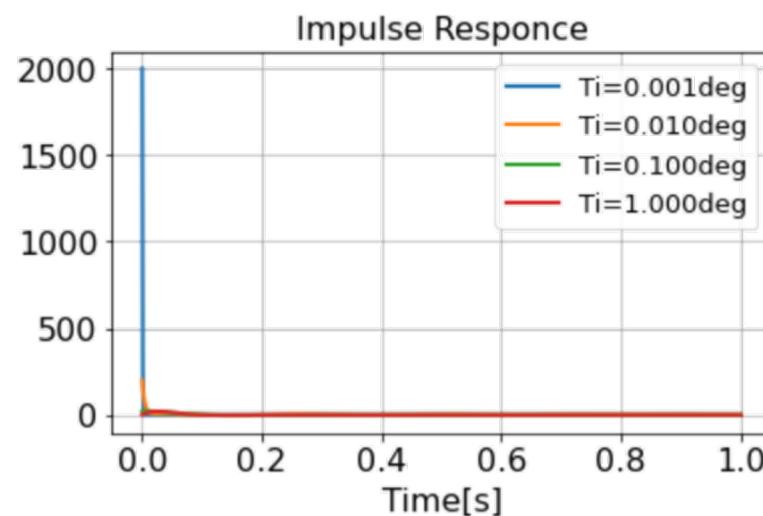
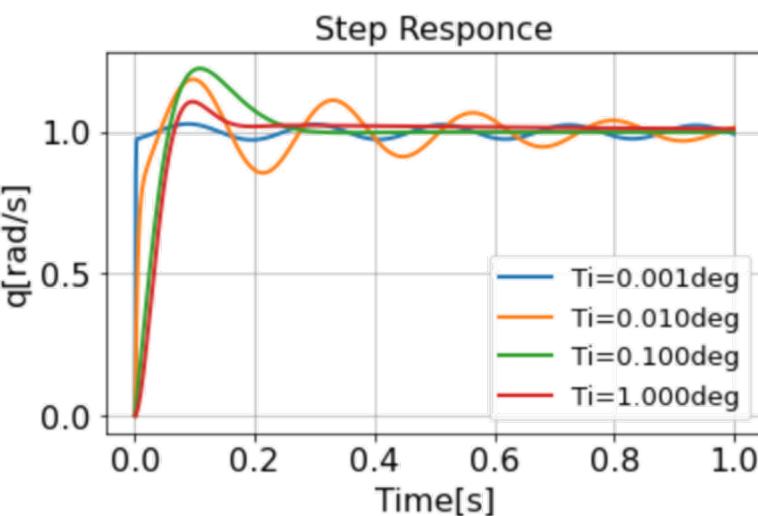
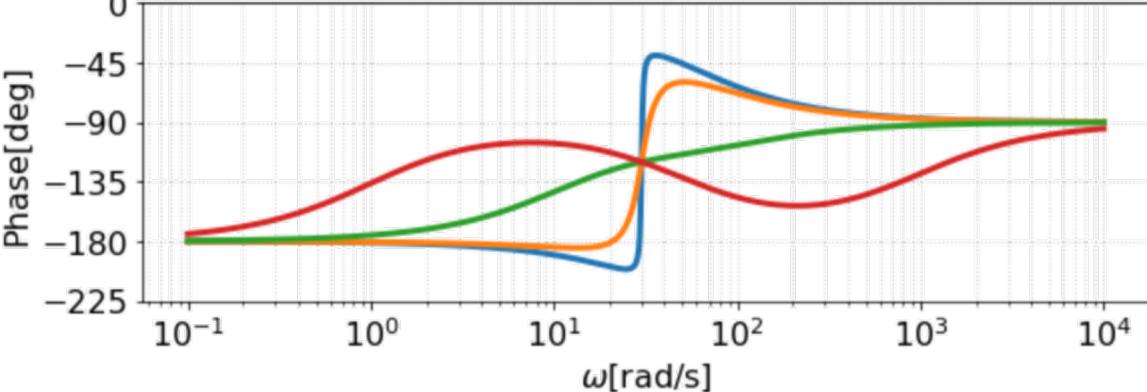
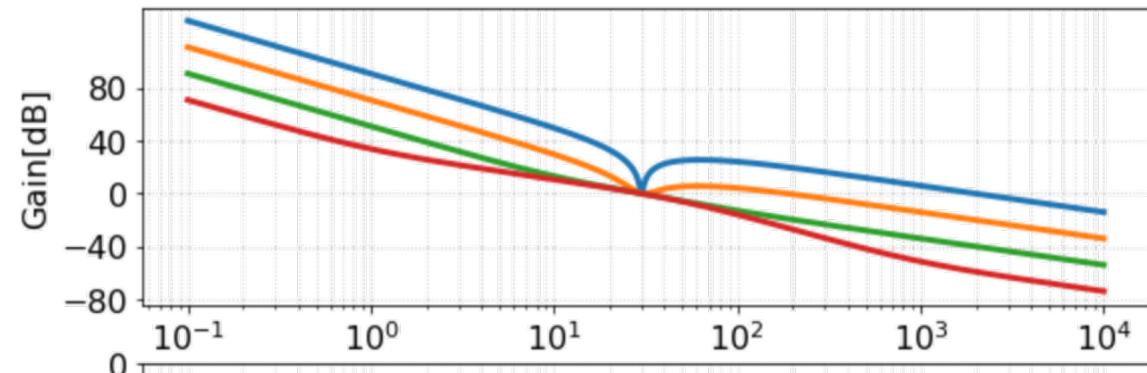
$$T_d = \frac{(\omega_c \tau + \alpha)/T_i + \omega_c^2 \tau \alpha - \omega_c}{\omega_c^2 (\omega_c \tau + \alpha)}$$

$$\text{ゲインについては } |L(j\omega_c)| = \Delta K_{qp} \sqrt{u^2 + v^2} = 1$$

この式に u, v を入れ直して比例ゲイン K_{qp} について整理すると次の関係が得られます

$$K_{qp} = \frac{\omega_c (\omega_c \tau + \alpha)}{K_q \sqrt{1 + \alpha^2}}$$

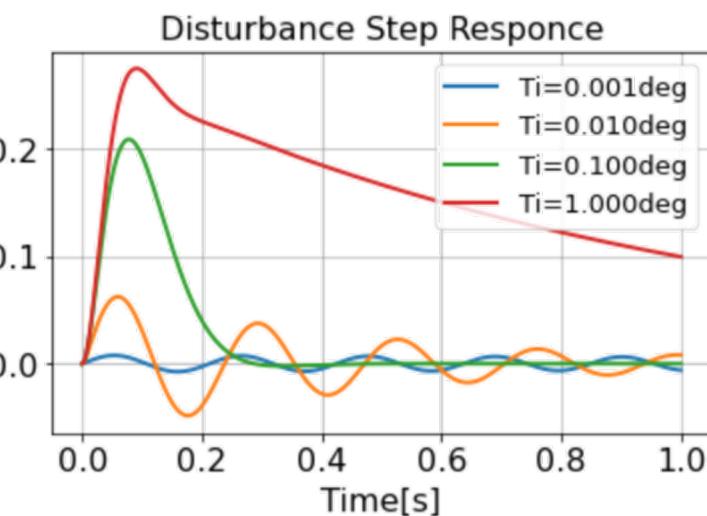
以上から、設計者が位相余裕とゲイン交差周波数を決めると、そこから比例ゲインが一意に決められる事がわかります。またそれとは別に、積分時間を決めるとき微分時間も決まります。これは逆でもいいですが、定常特性や外乱応答などに積分時間が大きな影響を与えるので積分時間を決めて応答を確かめる方が良いと思われます。

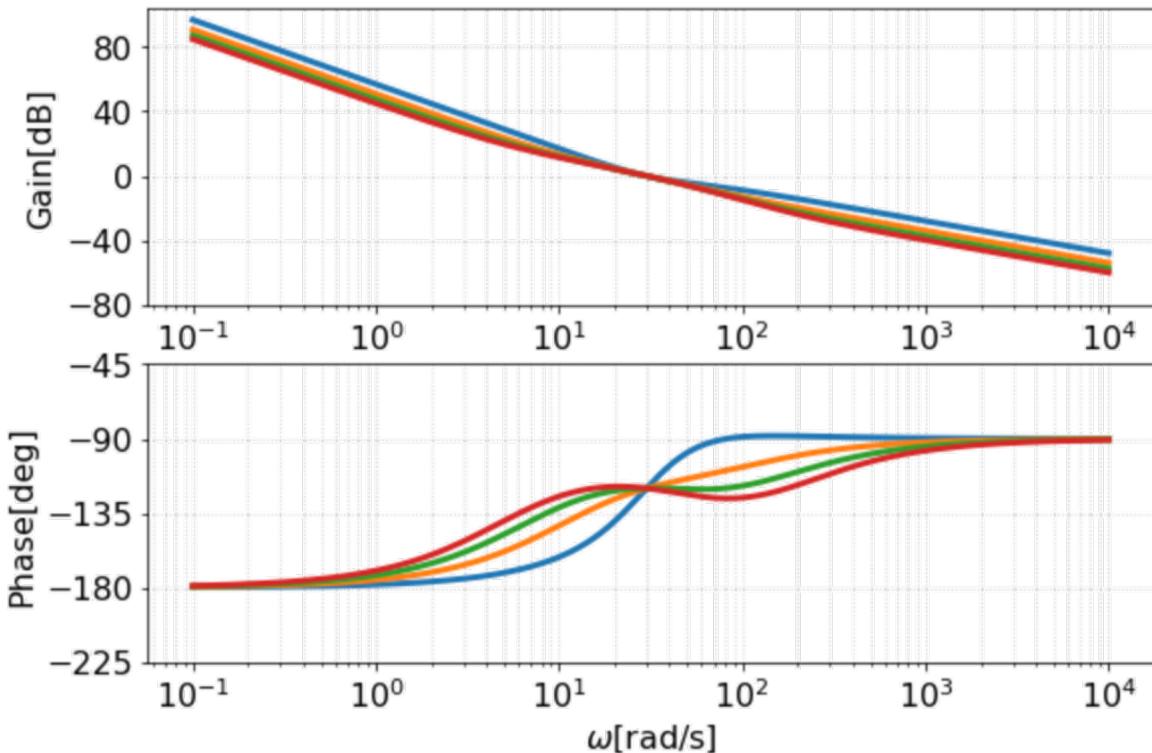


設計者が位相余裕とゲイン交差周波数を決めるとき、PID制御において比例ゲインの値が決まる事がわかったので、具体的に位相余裕を60deg、ゲイン交差周波数を30rad/sに決めて比例ゲインを求めるとき、約3.8となりました。あとは積分ゲインを自由に決められるので、いくつかの値に変えて、周波数応答、過渡応答について調べてみます。

積分ゲインが小さいほどオーバシュートが小さく、外乱の抑制効果も高いけれども、振動的になる。また位相曲線がS字の場合急激な入力にインパルス的な出力ができる。位相曲線がZ字の場合はインパルス的な出力は抑えられる。滑らかに右上がりの位相曲線が理想的です。

位相余裕 60 deg	比例ゲイン	積分時間	微分時間
ゲイン交差周波数 30rad/s	3.8042	0.001	1.1111
		0.01	0.1111
		0.1	0.0111
		1.0	0.0011

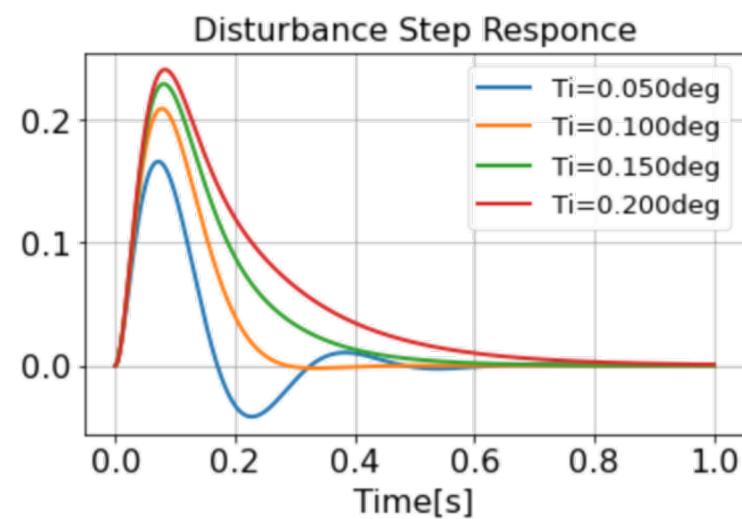
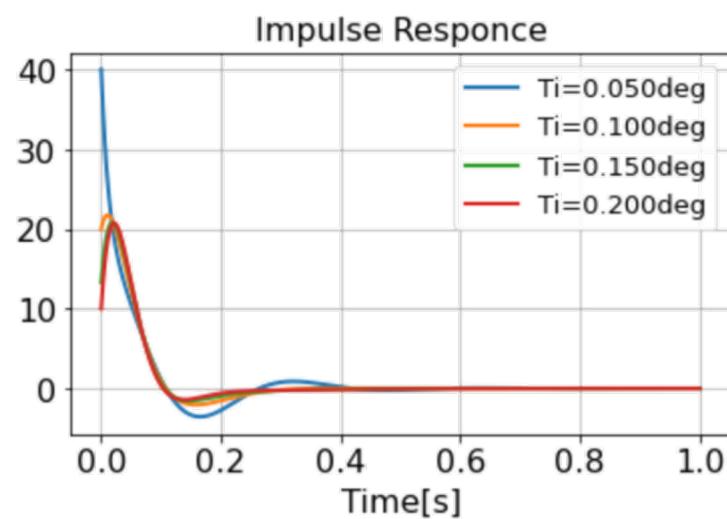
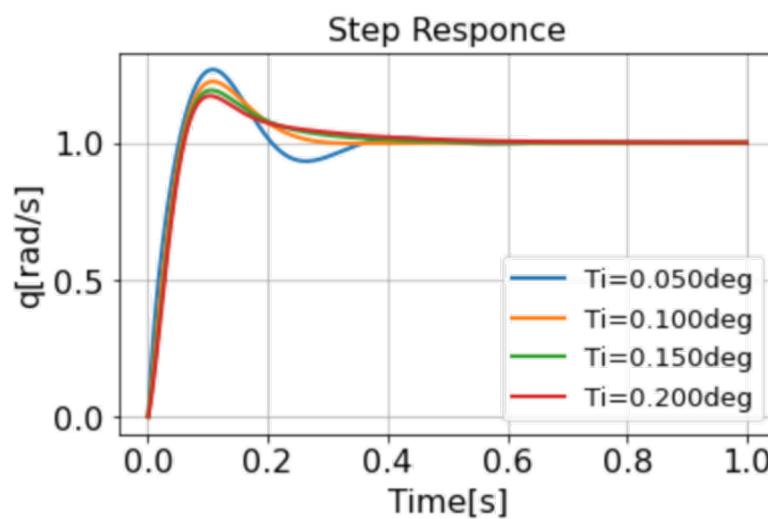


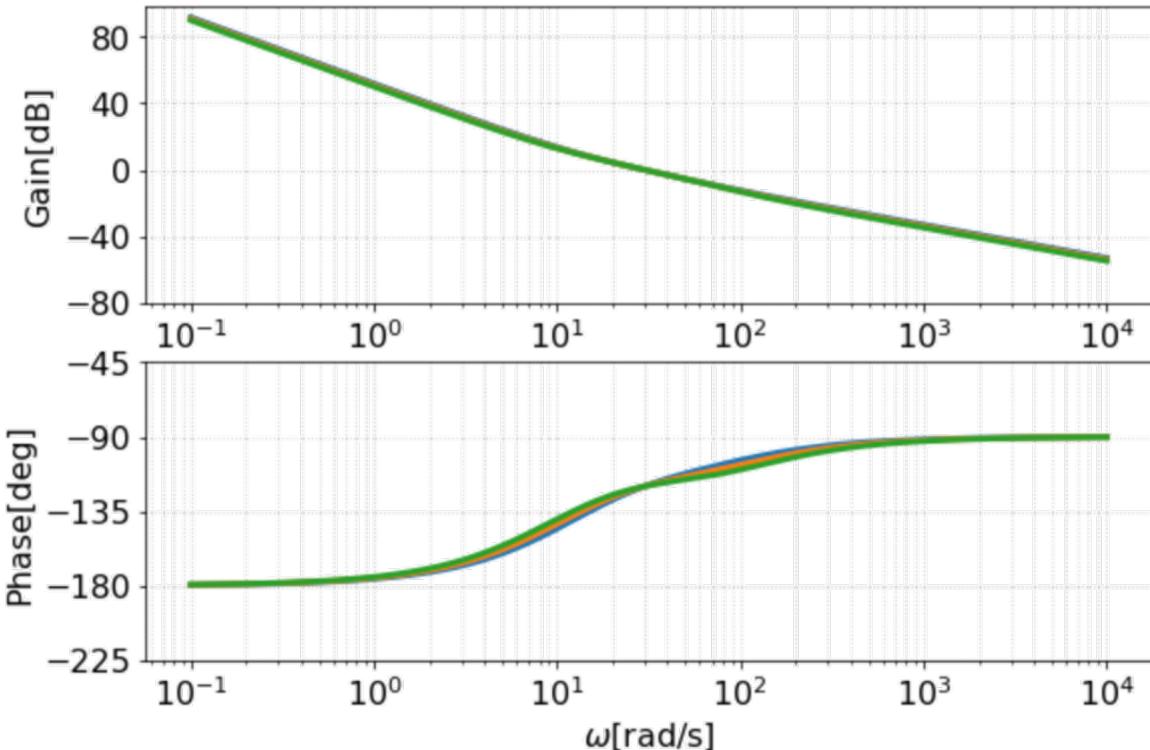


Ti= 0.0500
Ti= 0.1000
Ti= 0.1500
Ti= 0.2000

前ページの実験で、積分時間が0.1付近が理想的な事が見てきましたので、0.1の周りの値を見てみることにしました。0.1付近を境にして大きくなるにつれ、位相曲線のカーブがS字からZ字に変化します。S字の場合はインパルス的な応答が出ていますが、Z字の場合が望ましい応答になる事がここでも、確認できます。

位相余裕	比例ゲイン	積分時間	微分時間
60 deg	0.05	0.0222	
ゲイン交差周波数	0.1	0.0111	
30rad/s	0.15	0.0074	
	0.2	0.0056	

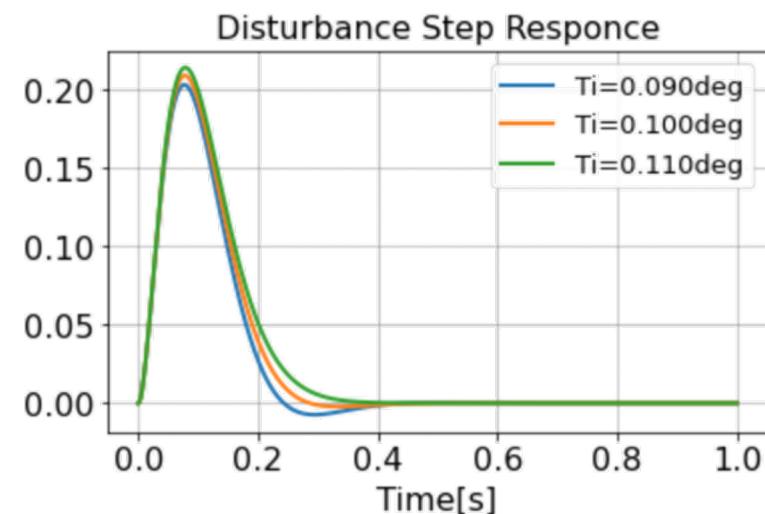
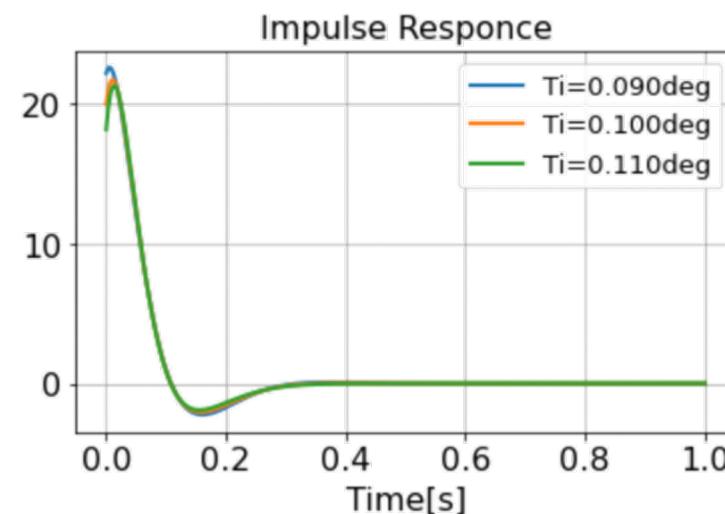
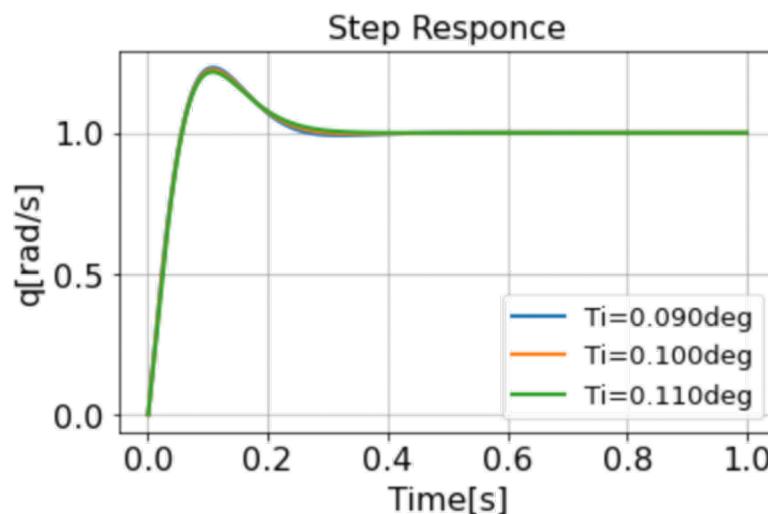


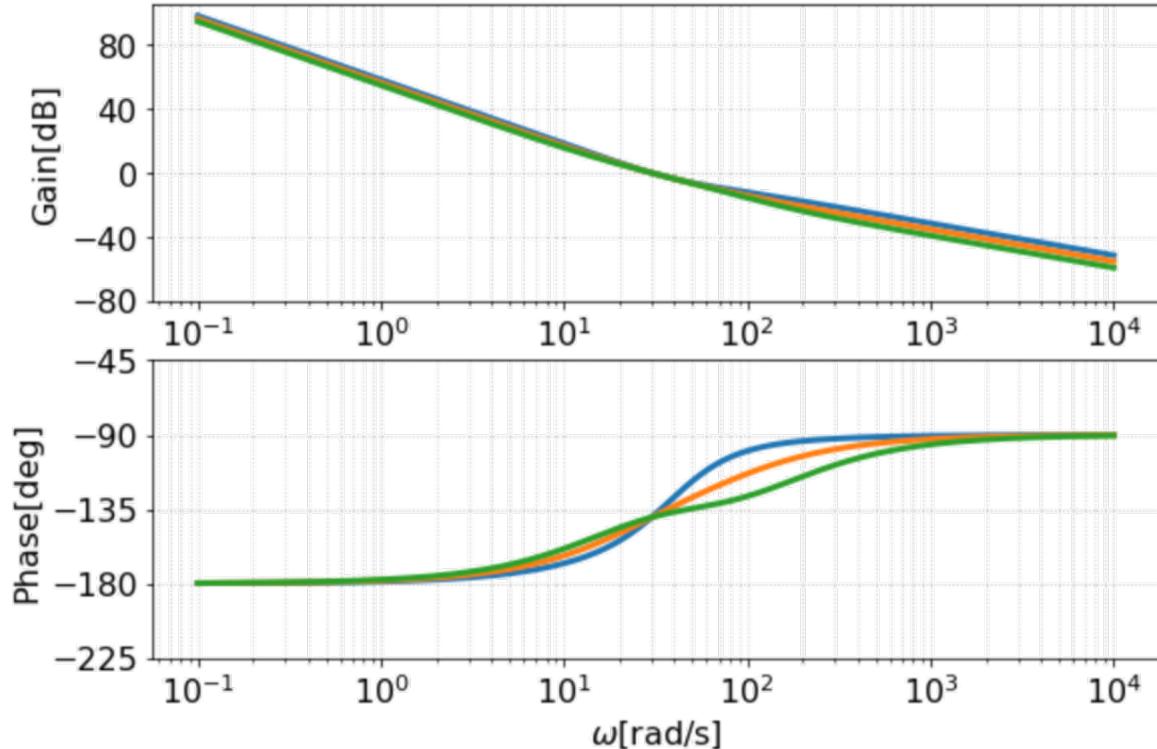


$T_i = 0.0900$
 $T_i = 0.1000$
 $T_i = 0.1100$

さらに、0.1付近で絞り込んでいきます。偶然ですが積分ゲインが0.1が今回の場合は一番良い結果が得られそうです。
今回は追及しませんが、ゲイン交差周波数で位相曲線は変曲点となっていますが理想に近い場合は変曲点を持たない曲線になっている様に見えます。（今後、確認したいと思います。）

位相余裕	比例ゲイン	積分時間	微分時間
60 deg	3.8042	0.09	0.0124
ゲイン交差周波数		0.1	0.0111
30rad/s		0.11	0.0101



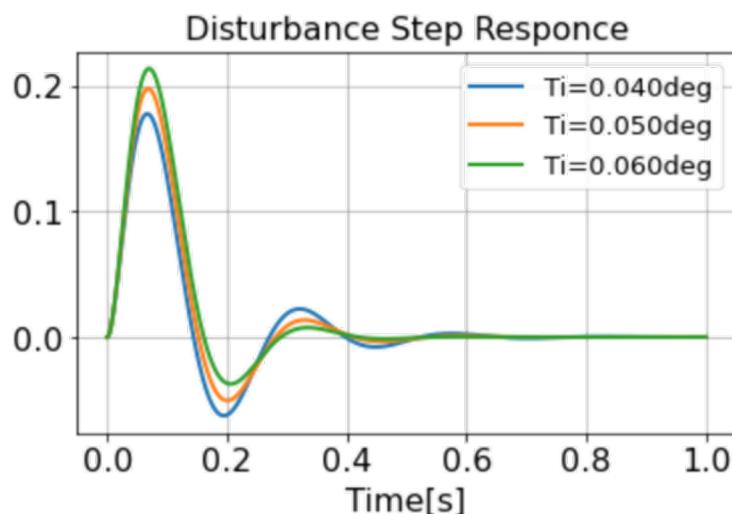
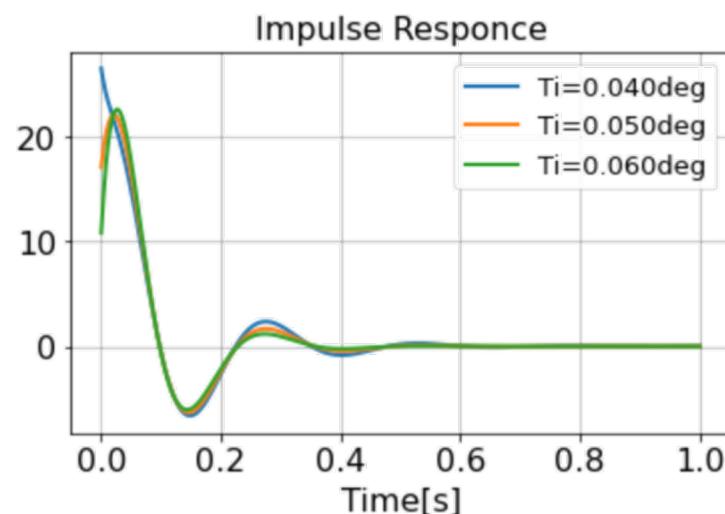
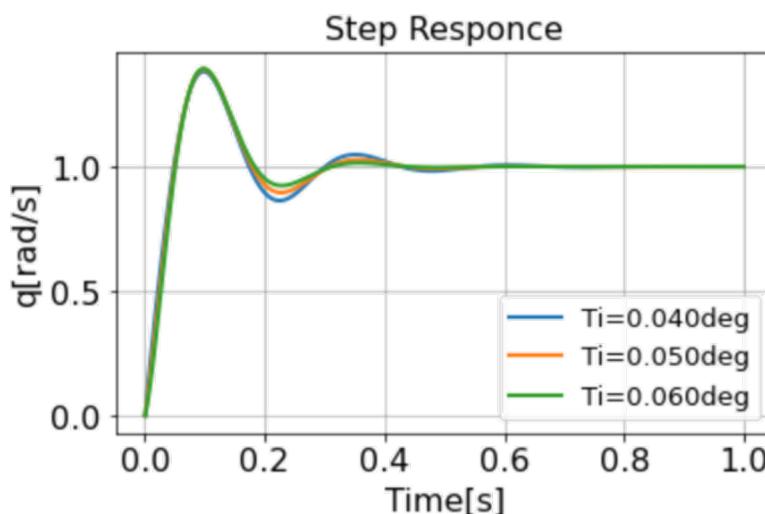


Ti = 0.0400
Ti = 0.0500
Ti = 0.0600

最後に、ゲイン交差周波数は30rad/sのまで、位相余裕を20deg小さくして40degの場合についてみてみました。この場合も比例ゲインは自動的に決まり3.5752となりました。積分時間は0.05付近が最適の様です。（周波数応答と時間応答を見た試行錯誤によります。）位相余裕を減らすと、振動的になっているのがわかると思います。

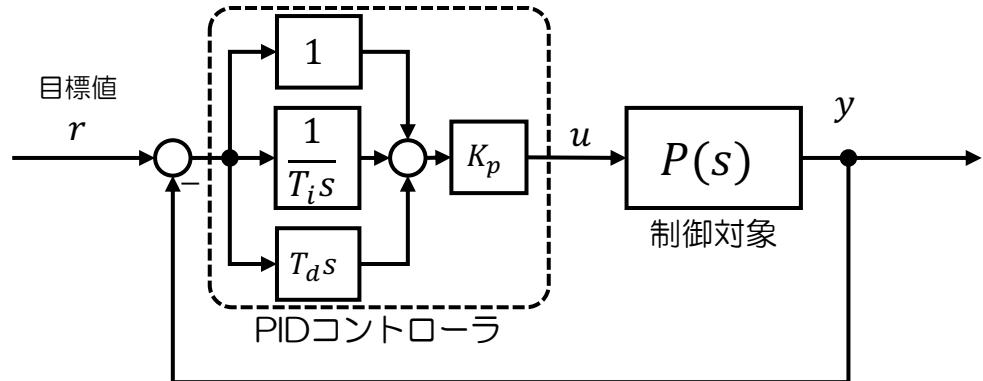
角速度制御については、今のところここまでにしたいと思います。PID制御の設計については理想的なモデルを用意して、低次の項からマッチングしていく部分的モデルマッチングという設計手法があります。設計の見通しが良くて優れた手法だと思っています。もう少ししたら、その話題も追加したいと思います。

位相余裕	比例ゲイン	積分時間	微分時間
40 deg	3.5752	0.04	0.0157
ゲイン交差周波数		0.05	0.0101
30rad/s		0.06	0.0064



位相余裕とゲイン交差周波数を満たすPIDゲイン決定の手順

角度制御の話題に移る前にPID制御での各ゲインの決定について手順をまとめておこうと思います。制御対象が任意の場合で成り立つはずです。あくまで我流ですのであまり参考にはならないかもしれません。昨今ではMATLABにもpidtuneコマンドがありもっと優秀ですので、そちらを使うのがお得かもしれません。



PIDコントローラの記述

$$C(s) = \frac{K_p(T_d s^2 + s + 1/T_i)}{s}$$

PIDコントローラの記法としてはこの書き方は良いと思います。これで積分時間と微分時間だけで位相曲線が決定してしまいます。

周波数伝達関数で考える

$$\begin{aligned} C(j\omega) &= \frac{K_p(-T_d \omega^2 + \omega j + 1/T_i)}{\omega j} \\ &= K_p - K_p(1/T_i - T_d \omega^2)j/\omega \\ &= \alpha + \beta j \end{aligned}$$

伝達関数のsに $j\omega$ を代入したものを周波数伝達関数と呼びます。代入して整理していくと、結果は複素数になります。制御では虚数単位はjです。

今後の式の展開のために文字を置き換えておきます

ここで

$$\alpha = K_p \quad \beta = -K_p(1/T_i - T_d \omega^2)/\omega$$

制御対象の周波数伝達関数

$$P(j\omega) = u + vj$$

制御対象は具体的に語られていませんが周波数伝達関数とは複素数のことなので任意の制御対象をこの様に書き表すことができます。実部と虚部の値は周波数 ω の関数です。

一巡伝達関数の周波数伝達関数

$$\begin{aligned} P(j\omega)C(j\omega) &= u\alpha - v\beta + (u\beta + v\alpha)j \\ &= \zeta + \xi j \end{aligned}$$

ここで

$$\zeta = u\alpha - v\beta \quad \xi = u\beta + v\alpha$$

ゲインと位相

$$\begin{aligned} |P(j\omega)C(j\omega)| &= \sqrt{\zeta^2 + \xi^2} \text{ (ゲイン)} \\ \angle P(j\omega)C(j\omega) &= \tan^{-1} \xi / \zeta \text{ (位相)} \end{aligned}$$

一巡伝達関数は制御対象とコントローラの伝達関数の積なので整理するとこの様になります。ここでも文字の置き換えをします。

一巡伝達関数のゲインと位相は周波数伝達関数の実部と虚部を用いてこの様になります。

位相余裕とゲイン交差周波数

位相余裕 : PM ゲイン交差周波数 : ω_c その時の位相 : ϕ_m

$$\phi_m = PM - \pi \text{ とすると}$$

$$\sqrt{\zeta^2 + \xi^2} = 1 \quad (g)$$

$$\tan \phi_m = \xi / \zeta \quad (p)$$

$$\zeta \sqrt{1 + \tan^2 \phi_m} = 1 \quad (g')$$

$$\xi = \zeta \tan \phi_m \quad (p')$$

ゲイン交差周波数 ω_c ではゲインは1なのでこの式が成り立ちます。

ゲイン余裕PMの時の位相 ϕ_m はなので、位相に関してもこの式が成り立ちます。

(p)式を用いて(g)式を書き換えると、この様に書けます。

(p)式を変形しました

(p')式にこれまで置き換えたものを入れ戻していく、 T_d について解きます。

積分時間と微分時間の関係

$$T_d = \frac{1}{\omega_c^2 T_i} - \frac{v - u \tan \phi_m}{\omega_c(u + v \tan \phi_m)} \quad (d)$$

ここで $\phi_m = PM - \pi$

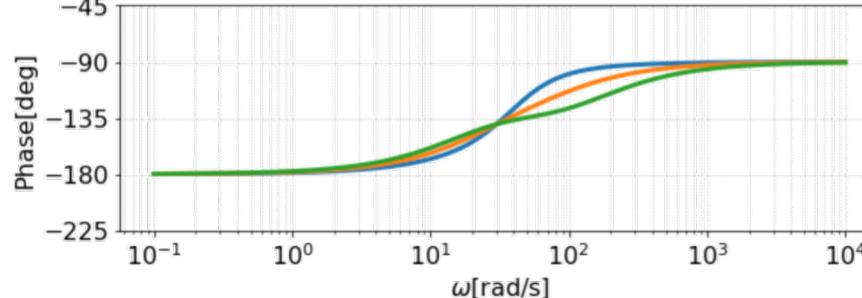
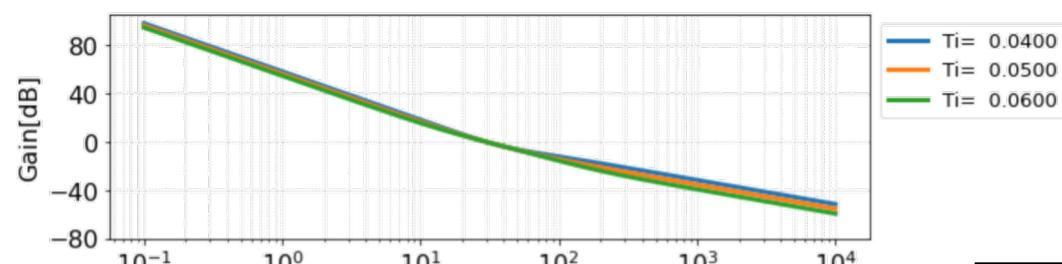
位相余裕、ゲイン交差周波数を設計仕様として決定して、整理していくと積分時間と微分時間の関係式が導かれます。この両者に対して比例ゲインは独立しています。

(g')式にこれまで置き換えたものを入れ戻していく、さらに (d) 式を用いると、積分時間と微分時間が消えます。

比例ゲイン

$$K_p = \frac{u \cos \phi_m + v \sin \phi_m}{u^2 + v^2} \quad (f)$$

全てを代入して整理していくとこの様に比例ゲインを決定する式が導かれます。

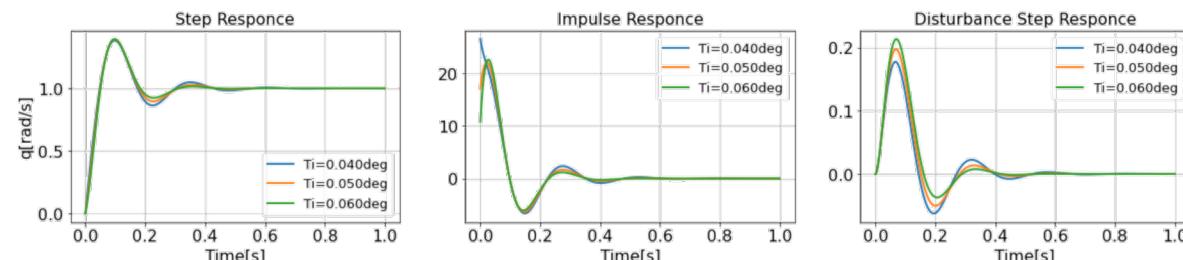


こんな風に周波数応答や過渡応答を確認

【まとめ】PIDゲイン決定の手順

- ① 位相余裕 PM とゲイン交差周波数 ω_c を決める
- ② 位相余裕 PM の時の位相 ϕ_m を計算する
- ③ 制御対象の周波数伝達関数を求める
- ④ 制御対象の周波数伝達関数からの ω_c の時の実部 u の値と虚部 v の値を求める
- ⑤ (f) 式から比例ゲインを求める
- ⑥ 積分時間を適当に決めて (d) 式から微分時間を求める
- ⑦ 求めたゲインで周波数応答、ステップ応答、インパルス応答、外乱からの応答等を見る。良くなければ⑥にもどる。良ければ終了

※実機での調整がこの後あります



PIDゲイン決定のPythonコード (PIDtune)

```
import numpy as np
import control.matlab as matlab
import matplotlib.pyplot as plt
```

これらのモジュールが必要なのでインストールをしてください。

```
#制御対象作成(python-control使用)
tau=0.02 #制御対象の時定数
K=10.0 #制御対象のゲイン
Plant=matlab.tf([K],[tau,1,0])
```

制御対象のパラメータを設定します。例としてマルチコプタのピッチレート系です。

#①位相余裕PMとゲイン交差周波数 ω_c を決める

```
PM=30*np.pi/180
omega_c=30
```

伝達関数表記でシステムオブジェクトを作ります。

設計者が決めた任意の位相余裕、ゲイン交差周波数を設定します。

#②位相余裕PMの時の位相 ϕ_m を計算する

```
phi_m=PM-np.pi
```

#③制御対象の周波数伝達関数を求める

```
#pythonではz=1+2*1jの様に複素数が使えます
```

```
Plant_freq=K/(omega_c*1j)/(tau*1j*omega_c+1)
```

pythonでは虚数単位1jを使って複素数を直接扱えます。制御対象の周波数伝達関数（複素数）を計算します。

#④制御対象の周波数伝達関数からの ω_c の時の実部uの値と虚部vの値を求める

```
u=Plant_freq.real
v=Plant_freq.imag
```

虚数オブジェクトPlantのメンバーである実部(real)と虚部(imag)にアクセスしてu,vとします。

#⑤ (f) 式から比例ゲインを求める

```
Kp=(u*np.cos(phi_m)+v*np.sin(phi_m))/(u**2+v**2)
```

比例ゲインゲット！

#⑥積分時間を適当に決めて (d) 式から微分時間求める

```
Tis=np.array([0.03, 0.04, 0.05])
```

```
Tds=1/omega_c**2/Tis - (v-u*np.tan(phi_m))/omega_c/(u+v*np.tan(phi_m))
```

微分時間ゲット！

```
#⑦求めたゲインで周波数応答、ステップ応答、インパルス応答、外乱からの応答等を見る。
```

#良くなればTiの設定を変えて本スクリプトを再実行する

```
Kps=np.ones(len(Tis))*Kp
```

print('積分時間はこの範囲で設定してください')

```
if (v-u*np.tan(phi_m))/omega_c/(u+v*np.tan(phi_m))>0:
```

```
    print('0 < Ti < [:f]'.format((u+v*np.tan(phi_m))/omega_c/(v-
```

```
u*np.tan(phi_m))))
```

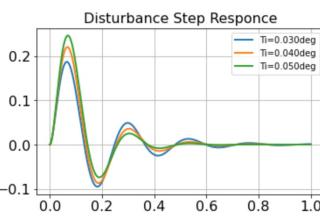
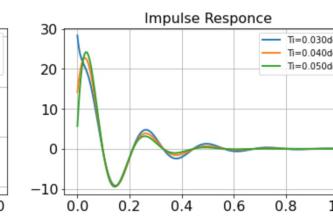
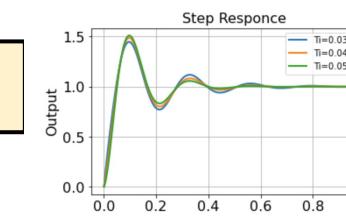
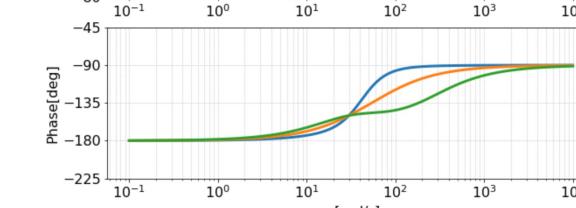
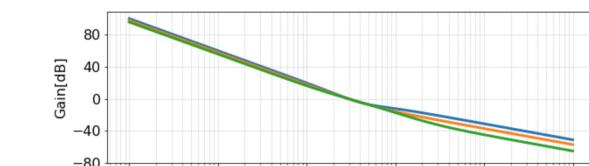
```
else:
```

```
    print('Ti>0¥n')
```

```
fig1=plt.figure(figsize=(10,7))
```

```
fig2=plt.figure(figsize=(20,3.5))
```

～以降省略～



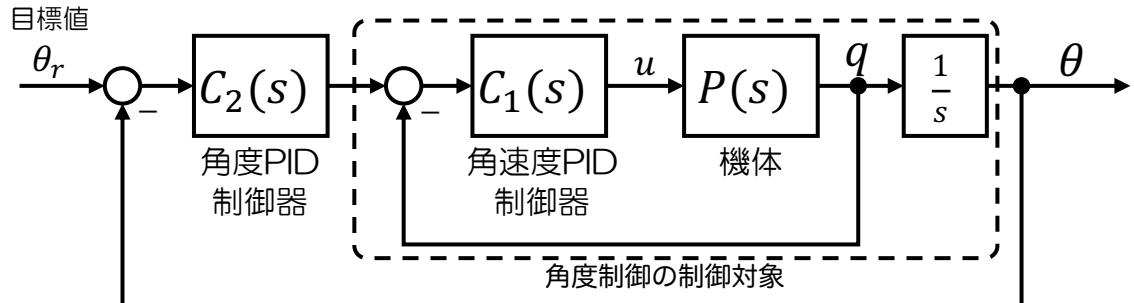
位相余裕とゲイン交差周波数の値により積分時間の設定可能範囲は制限されます。ここでその範囲を表示します。積分時間が範囲外になっている場合は手動で訂正してください。

このコードで前ページの手順が実行できます。赤字の部分が制御対象のパラメータ、伝達関数、制御仕様である位相余裕、ゲイン交差周波数、そして調整パラメータである積分時間です。それぞれ書き換えてください。

実行すると左の様に周波数応答と過渡応答が表示されますので、良くなるまで制御仕様または積分時間を調整してください。コードは以下にあります

https://github.com/kouhei1970/fundamental_of_multicopter_control

マルチコプタの角度制御系の設計



マルチコプタの角度制御系

前の角速度制御系の数値例を元にして、位相余裕とゲイン交差周波数を望ましいもにする角度制御系PID制御器を設計してみます。設計手順は前のページと同じ手順です。

角速度制御系の設計値

位相余裕 60 deg	比例ゲイン 3.8042	積分時間 0.1	微分時間 0.0111
ゲイン交差周波数 30rad/s			

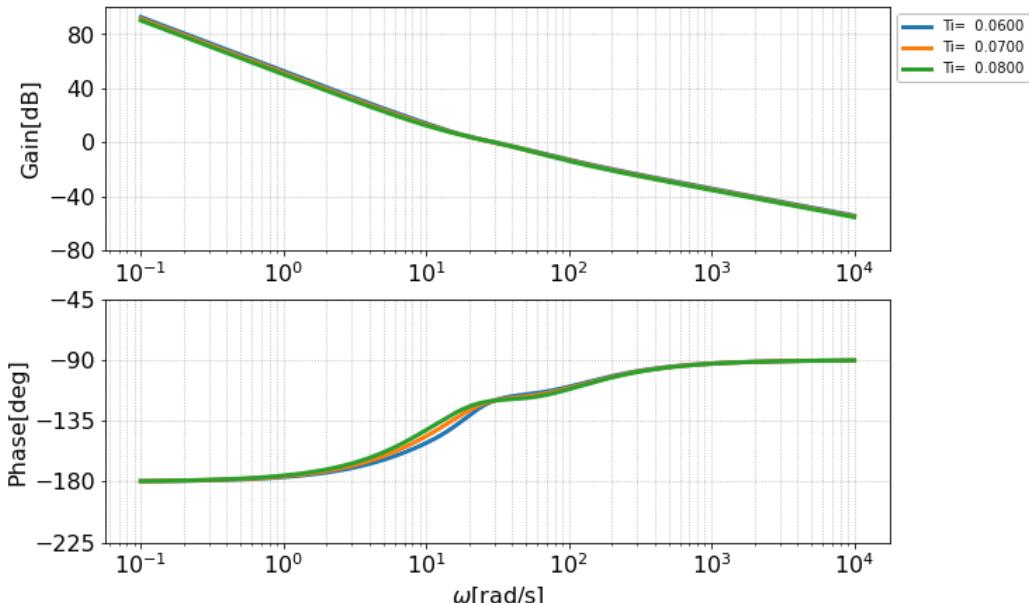
角度制御の制御対象の伝達関数

$$\frac{0.3847s^2 + 34.66s + 346.6}{0.0193s^4 + 1.385s^3 + 34.66s^2 + 346.6}$$

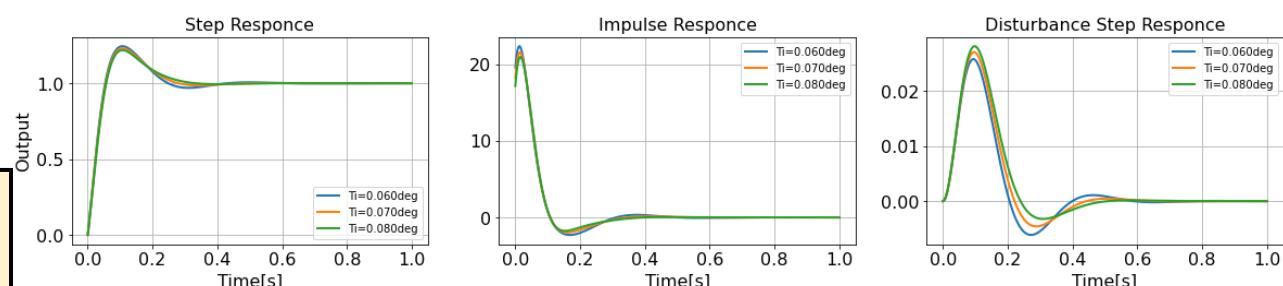
ここまでくると解析的に伝達関数を記述するのは大変になってくるので数値解で失礼です。

角度制御系の設計値

位相余裕 60 deg	比例ゲイン 25.9369	積分時間 0.07	微分時間 0.0352
ゲイン交差周波数 30rad/s			



角度制御系一巡伝達関数のボード線図



角度制御系の過渡応答

少々、試行錯誤を繰り返していますが、最終的に絞り込んだ際の比較結果が、上の図です。最終的に制御入力が大きくなりすぎないかなど気になりますがPID制御では制御器がインプロバーなため解析的には制御入力を見ることができないので、ここはひとまず終了とします。

マルチコプタのシミュレーション

$$\dot{x} = f(t, x, u)$$

対象の動作を表す微分方程式

常微分方程式の解法：4次ルンゲ・クッタ法

$$\begin{aligned} k_1 &= hf(t, x, u) \\ k_2 &= hf(t + 0.5h, x + 0.5k_1, u) \\ k_3 &= hf(t + 0.5h, x + 0.5k_2, u) \\ k_4 &= hf(t + h, x + k_3, u) \\ x &= (k_1 + 2k_2 + 2k_3 + k_4)/6 \\ t &= t + h \end{aligned}$$

x : 状態量 t : 時刻 h : 刻み幅 u : 入力等

上記の計算により刻み幅分の時刻進んだ時刻の状態 x を今の x, t, u から求めていくもので、逐次それらの値が求められます。初期値がなければ計算を始められないで、初期値を与える必要があります。

状態量は複数あっても構いません。また、高階の微分方程式の場合、変数の置き換えによって一階の連立微分方程式に書き換えられるので、高階の常微分方程式でも解くことができます。

スッティフ（硬い、やっかいな）なシステムの場合

例えばマルチコプタのシミュレーションのためにモータのインダクタンスの影響を無視しないとしましょう。そうするとモータの回路方程式について電流の挙動も解く必要がありますが、電流の挙動は機体の運動に対して時定数が極めて小さくなります。この様にシステムに変化の著しく違う状態が混在していると（この様なシステムをスティッフといいます。）数値計算的な不安定が発生します。解決には刻み幅を小さくすることですが、計算時間がかかるのが問題です。

Pythonでの実装例

```
import matplotlib.pyplot as plt
```

```
def rk4(func, t, h, x, *p):
```

$k_1 = h * func(t, x, *p)$

$k_2 = h * func(t + 0.5 * h, x + 0.5 * k_1, *p)$

$k_3 = h * func(t + 0.5 * h, x + 0.5 * k_2, *p)$

$k_4 = h * func(t + h, x + k_3, *p)$

$x = x + (k_1 + 2 * k_2 + 2 * k_3 + k_4) / 6$

return x

ルンゲ・クッタ法を一度だけ計算する関数

```
def xdot(t, x, u):
```

$a = -10$

$b = 1$

return $a * x + b * u$

微分方程式から導関数を定義した関数

```
# ここからメイン
```

$h = 0.01$

$t = 0$

$x = 0$

$u = 1$

fintime = 1

$T = []$

$X = []$

```
for _ in range(int(fintime/h)):
```

$x = rk4(xdot, t, h, x, u)$

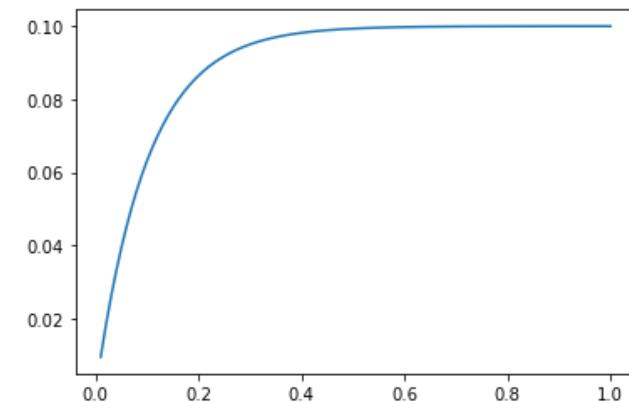
$t = t + h$

$T.append(t)$

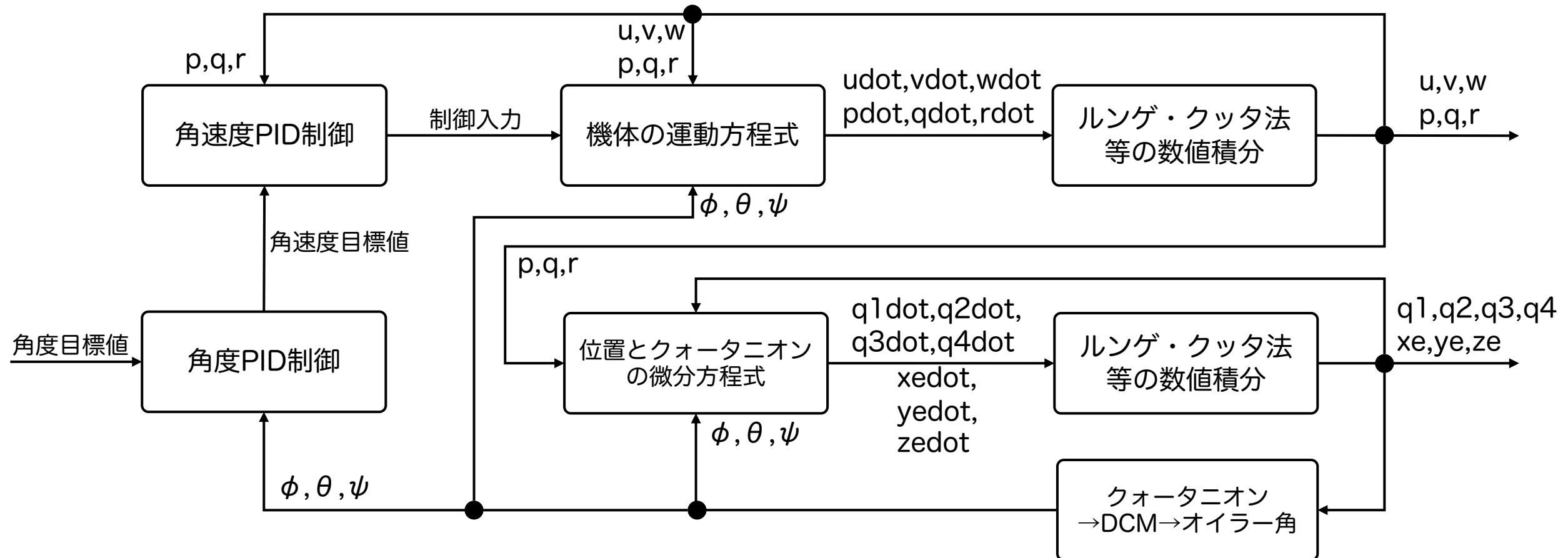
$X.append(x)$

```
plt.plot(T, X)
```

```
plt.show()
```



こんな感じの計算結果が出ます



PID制御の基本的な実装

誤差

$$e_k = y_{ref_k} - y_k$$

数値積分（積分器）

$$s_k = s_{k-1} + e_k h$$

数値微分

$$de_k = (e_k - e_{k-1})/h$$

PID制御の操作量

$$u_k = K_p(e_k + s_k/T_i + T_d de_k)$$

積分器のオーバーフロー対策

積分器に関しては、実際にマイコン等で実装する際には変数のオーバーフローを考慮しておく必要があります。短い制御周期の中で何度も積算していくと場合によっては積分器に割り当てられている変数はオーバーフローしてしまう可能性が高いです。オーバーフローが起こった時に正負が逆転してしまい、制御中に異常な動作に陥り危険です。何があるか分からないので、積分器はある値以上または以下にならない様に値を制限する工夫でオーバーフローを防止してください。

操作量リミッタとワインドアップ

特に微分項が入っている場合に、操作量が大きくなりすぎて、実際の制御対象に加えて良い制限を超える場合があります。その様に過大な操作量が制御対象に加わることを防ぐために、操作量を制限します。その制限をリミッタと呼びます。例えば、目標値が大きく誤差が大きくなると比例項だけでも操作量がリミッタに引っかかることがあります。その場合、制御対象は操作量が限度目一杯で最大の能力で動作することになります。そして目標値に近づく間は積分項の大きさが溜まり続けます。この場合積分項は本来ではあり得ない大きさまで大きくなります。そして、制御対象が目標値を超えると、積分器は現象に転じます。

PID制御は微分項があるのでそれ自体はインプロバー（分母の次数より分子の次数が高い。）なので実現は難しいのですが、微分を左の様に差分で近似することで、実装することはできます。積分の部分はこの例では、区分求積法と同じで誤差の高さで幅が h の細長い長方形を逐次足していくことで実装します。比例項と積分項及び微分項を足し合わせたものがPID制御の算出する操作量（制御入力）となります。

積分項が減少し、本来ありうる値付近になるまで不自然な動きが続きます。この様な現象をワインドアップと言います。巻いて置いてある長いロープの端に取り付けられているものを、ロープを手繰り寄せることで手元に引き寄せる際に、ずいぶん余計な量のロープを巻き取ることになると思いますが、制御のワインドアップはそのことを連想させます。

積分器を持たないPID制御の実装

一般的なPID制御の操作量の微分をもとめ、操作量の微分を数値微分と置き換えることで、PID制御式が積分項を持たない形式にすることができます。これを位置の微分が速度であることのアナロジーで速度型PIDと言います。また、これまでに示したPID制御式は位置型PIDとも呼ばれています。積分器を持たなくて良いことからワインドアップ対策などする必要がなく良いのですが、制御系が非最小位相系になることも知られています。実装の際にはさまざまところで速度型が多用されているとも聞きますが、非最小位相系になってしまうところが気になり、私は採用を躊躇います。

速度型PID制御の操作量

$$u_k = u_{k-1} + K_p(e_k - e_{k-1} + he_k/T_i + T_d\{e_k - 2e_{k-1} + e_{k-2}\}/h)$$

(位置型) PID制御と速度型PID制御の応答の比較



以下にシミュレーションで用いたPID制御の実装例を示します。実際の組み込み系ではCやC++等の高速な言語で置き換える必要があります。

```
#位置型PID制御の実装例
class pid:
    def __init__(self, kp, ti, td,limit_upper=100.0,limit_lower=-100.0,limitter=False):
        self.Sum=0.0
        self.olderr=0.0
        self.oldtime=0.0
        self.limit_upper= limit_upper
        self.limit_lower= limit_lower
        self.limiterFlag=True
        self.Kp=kp
        self.Ti=ti
        self.Td=td
        self.u=0.0

    def controller(self, y, ref, t):
        err=ref-y
        period=t-self.oldtime
        self.Sum=self.Sum+err*period
        #print(period)
        if period==0.0:
            errdot=0.0
        else:
            errdot=(err-self.olderr)/period

        self.u=self.Kp*(err + self.Sum/self.Ti + self.Td*errdot)
        #リミッター
        if self.limiterFlag==True:
            if self.u>self.limit_upper:
                self.u=self.limit_upper
            elif self.u<self.limit_lower:
                self.u=self.limit_lower

        self.oldtime=t
        self.olderr=err

    return self.u
```

```
class vpid:
    def __init__(self, kp, ti, td,limit_upper=100.0,limit_lower=-100.0,limitter=False):
        self.Sum=0.0
        self.olderr1=0.0
        self.olderr2=0.0
        self.oldtime=0.0
        self.limit_upper= limit_upper
        self.limit_lower= limit_lower
        self.limiterFlag=True
        self.Kp=kp
        self.Ti=ti
        self.Td=td
        self.u=0.0

    def controller(self, y, ref, t):
        err=ref-y
        period=t-self.oldtime
        self.Sum=self.Sum+err*period
        #errdot=(err-self.olderr)/period
        if period==0.0:
            self.u=self.u¥
            + self.Kp*(err-self.olderr1 + period*err/self.Ti)
        else:
            self.u=self.u¥
            + self.Kp*(err-self.olderr1 + period*err/self.Ti¥
            + self.Td*(err-2*self.olderr1+self.olderr2)/period)

        #リミッター
        if self.limiterFlag==True:
            if self.u>self.limit_upper:
                self.u=self.limit_upper
            elif self.u<self.limit_lower:
                self.u=self.limit_lower

        self.oldtime=t
        self.olderr2=self.olderr1
        self.olderr1=err

    return self.u
```

基本的なマルチコプタ制御プログラムを作り上げるまでの学習過程

学習カリキュラム

- ① マイコンの開発環境のインストール
- ② 開発環境を用いたビルド工程の習得
- ③ Lチカプログラムのビルドとマイコンへの書き込み
- ④ PWM信号生成方法の習得
- ⑤ UART通信でのコンソールとの通信方法の習得
- ⑥ インターバルタイマー割り込み方法の習得
- ⑦ I2C及びSPI通信方法の習得
- ⑧ ジャイロ・距離計等の外部ペリフェラルと通信して所望のデータを受け取る方法の習得
- ⑨ 送信（操縦装置）・受信装置から指令信号をマイコンが受け取る方法を習得
- ⑩ 送受信装置からの信号に基づきPWM信号のDutyを変更しモータの速度を可変させる方法の習得
- ⑪ スロットル・ロール・ピッチ・ヨー指令値から4つのモータへの回転指令値を生成（ミキシング）する方法の習得
- ⑫ 角速度・加速度（・地磁気）情報から姿勢の推定計算方法を習得
- ⑬ フィルター（EKF・相補フィルタ等）について習得

- ⑭ PID制御の実装方法を習得
- ⑮ PIDによる角速度制御ループの実装を習得
- ⑯ PIDによる角度制御ループの実装を習得
- ⑰ PIDゲインのチューニングについてモデルに基づく方法について学習
- ⑱ 実機のフライトテストにて調整
- ⑲ 現代制御理論での制御へ・・・

番外編

- 使いやすいエディタを見つける
- ソースコードの分割ビルド方法の習得
- 便利な統合環境の導入を検討
- 各言語に習熟
- プログラミング作法の勉強
- 勉強は無限につづく・・

※C言語の基礎的な知識を既に有しているものとしますが、基礎でOK



今は、これが精一杯



つづく