第2部

第7章 航空機の制御といったら「モデル・ベース」設計

高度な制御のための運動のモデル化＆定式化

藤原 大悟

ここまでで，STEVAL-DRONE01のハードウェアやファームウェアをカスタマイズするために必要な知識について一通り解説しました．

ここからはPID制御の代わりとして，状態フィードバック制御を題材に，ドローン用の角速度制御器を設計して実装し，飛行実験を行います．

本章では準備として，ドローンの動きのモデル化，運動方程式の計算，システム同定実験によるパラメータ調整など，ドローンの制御系設計を行うにあたっての基本的事項について解説していきます．

モデル化についての基礎知識

● 設計の手始めは入力に対する動きを数式で表す

ドローンの（ドローンに限らず航空機についても）飛行制御系の設計を行う際は，いきなり設計に入るのではなく，まずはドローンが各プロペラのモータへの入力に対してどのように運動するのかを数式で表現する作業を行います．それが終わった後に，立てた数式に基づいて制御系の設計を行うという流れです．

ドローンの運動を表現した数式を数学モデルまたは単にモデルと呼びます．プラモデルのように，実物とは異なるものの，ある側面（この場合はドローンの運動）については，実物の特徴をよく反映したもの，という意味でモデルです．

モデルを作ることをモデリングあるいはモデル化と呼びます．また，モデルに基づいて設計を行うことをモデル・ベース設計と呼びます．

● 航空機の制御設計と言ったら「モデル・ベース」設計

モデル・ベース設計（MBD：Model-Based Design）は，近年多方面でよく耳にするようになりましたが，航空機の飛行制御の世界では昔から当たり前のように行われてきました．モデル・ベース設計の利点は，実物を使わず机上でさまざまな技術検討ができる点ですが，その一方，モデルが実物と乖離（かいり）していると，例えば設計した制御系を実際にドローンに実装したら制御性能が思ったほど出ない，最悪の場合離陸困難など，技術検討が無意味なものになりかねません．モデルと実物との間の差異をモデル化誤差と呼び，これをいかに小さくするかが大事になってきます．

● 本章ではモデル化を頑張る

モデル・ベース飛行制御系設計では，大半の時間をモデリングに費やすことも珍しくありません．ただし，モデル化誤差を完全にゼロにすることは困難です．誤差の存在を認めた上で，どこに/どのような/どの程度の誤差が残ったかを把握して，技術検討に取り組むことを心掛けることが肝心です．

本章では，ドローンのモデリングに焦点を当てます．ここからは数式が多くなります．誌面の都合上，簡略化した形になるので，より深く知りたい方は，国内外の機械/航空/制御関係の学会が発行する論文誌を調べることをお勧めします．

● モデル化の前提

ドローンは，x-z平面とy-z平面に関して対称な形状/質量分布であるとします．トリム飛行状態はホバリングとし，ホバリングおよびその近傍（低速飛行）の運動を考え，機体胴体が受ける空気抵抗（有害抵抗）は無視し得るとします．

ここで，トリム（trim）とは釣り合い，つまり機体胴体に働く力の釣り合いが取れた状態のことを言います．これは，例えばホバリングや等速直線運動のような定常状態を指します．どのような飛行状態をトリム飛行状態とするかは設計者が都度決めることになります．

● ドローンの運動をモデル化するステップ

ドローンの運動をモデル化すると，おおよそ**図1**のようになります．信号の流れは一部の例外を除いて左から右へ向かいます．左から順に，第1段階としてドローンに働く力やトルクの計算，第2段階として運動方程式の計算による加速度/角加速度の算出，第3段階としてそれらの積分による速度/位置および角速度

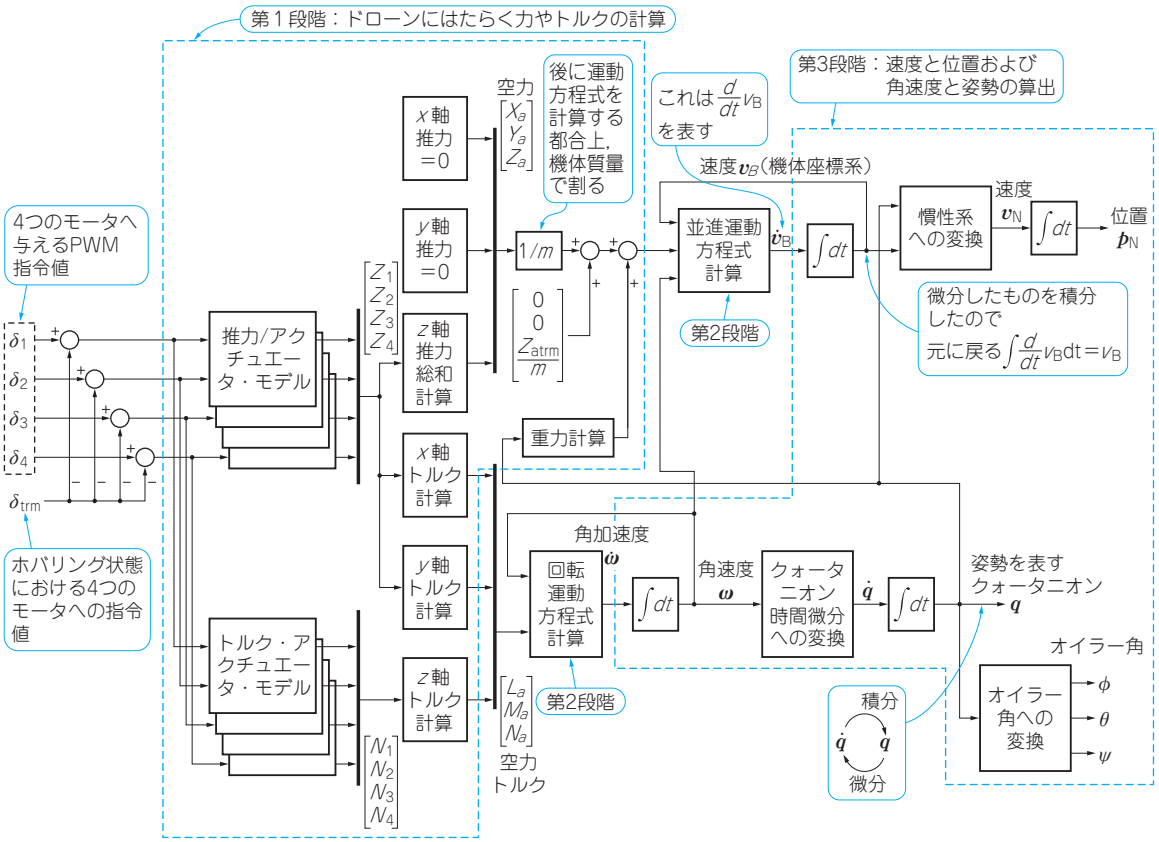

図1　3段階で運動モデルを構築する

/姿勢の算出に分かれます．また，運動には重心の位置変化を表す並進運動と，機体胴体の姿勢変化を表す回転運動の2つがあり，これら2つは互いに影響を及ぼしています．

ステップ1…ドローンに働く力やトルクの計算

● ホバリング状態でのPWM値は一定

ドローンには，プロペラを介して空気から受ける力（空力）と重力という大きく分けて2つの力がかかります．空力については，4つのモータへ与えるPWM指令値δ_1，δ_2，δ_3，δ_4のみで定まるものと仮定します．まずは，これらからPWM指令値のトリム値δ_{trm}を差し引きます．

トリム値とは，トリム飛行状態のときに各変数がとる一定値のことです．従って，δ_{trm}はホバリング飛行状態における4つのモータのPWM指令値となります．なお，トリム飛行状態のとり方や重心位置によっては，4つのモータそれぞれにトリム値が異なることもありますが，今回は4つ全て等しい値δ_{trm} = 950LSBとしました．

● 飛行状態を限定しモデルを単純化

トリム値を差し引くことで，トリム飛行状態近傍のモータやプロペラ，空力の挙動のみ考えればよくなり，問題が単純化されます．本来，空力は大変複雑な挙動を示すものです．

例えば，設計しようとする飛行制御系が高速飛行など機体を激しく動かす運動や，機体質量が大きく変化する状況など，幅広い飛行条件（飛行エンベロープ）までカバーすることが求められるなら，それに見合うようPWM指令値や機体の速度，角速度，機体質量，慣性モーメントの取り得る可能な限り広い値の範囲で挙動がきちんと模擬できるモデルを作るのが望ましいです．ここでは，ホバリング近傍の飛行状態のみをカバーすることを考え，線形な特性とみなし問題を単純化しました．

● アクチュエータのモデリング

上記のことを踏まえ，PWM指令値に対しプロペラがどのような推力やトルクを機体胴体に及ぼすかをモデリングします．

推力については，PWMの変化に対して推力の変化がわずかに遅れることを模擬するため，各プロペラに

ついてゲインK_T[N/LSB]，時定数τ_T[s]の1次遅れ系を使って表現しました．時定数とは，一定値が入力されたときに最終値の63.2%に達するまでの時間で，値が大きいほど応答が遅くなります．先に解説したノイズ・フィルタの1次遅れ系と同じで，折点周波数[Hz]は$1/(2\pi\tau_T)$となります．伝達関数で表すと次の通りです．

$$\frac{\bar{Z}_i(s)}{\Delta\bar{\delta}_i(s)}=\frac{K_T}{\tau_T s+1} \quad \cdots\cdots(1)$$

ここでiはモータ・プロペラの番号($i=1$，2，3，4)，Z_iはプロペラの推力[N]，$\Delta\delta_i$は$(\delta_i-\delta_{trm})$，$\bar{Z}_i(s)$と$\Delta\bar{\delta}_i(s)$はそれぞれZ_iと$\Delta\delta_i$のラプラス変換です．モデル・パラメータは，$K_T=2\times10^{-4}$，$\tau_T=0.1$としました．

トルクについては，筆者が実験データを解析した限り$\Delta\delta_i$に対するトルクN_i[Nm]の挙動に遅れ特性を含める必要はないと判断しました．そこで，次式のように単純なゲインK_Q[Nm/LSB]で模擬しました．モデル・パラメータは，$K_Q=5.5\times10^{-7}$としました．

$$N_i=K_Q\Delta\delta_i \quad \cdots\cdots(2)$$

● プロペラの推力とトルクを機体にかかる力とトルクに変換

ここまでは各プロペラの推力とトルクですが，これらを機体の重心に働く機体軸3軸に関する力とトルクに変換します．推力はz軸方向成分のみで，x軸とy軸の成分を持たないとすれば，空力ベクトル$\boldsymbol{F}_a$は次式となります．

$$\frac{\boldsymbol{F}_a}{m}=\frac{1}{m}\begin{bmatrix}X_a\\Y_a\\Z_a\end{bmatrix}+\begin{bmatrix}0\\0\\\dfrac{Z_{atrm}}{m}\end{bmatrix} \quad \cdots\cdots(3)$$

$$\begin{bmatrix}X_a\\Y_a\\Z_a\end{bmatrix}=\begin{bmatrix}0\\0\\1\end{bmatrix}(Z_1+Z_2+Z_3+Z_4) \quad \cdots\cdots(4)$$

後に運動方程式を計算する都合上，機体質量m[kg]で割って表します．式(3)の右辺第2項のZ_{atrm}は，トリム飛行状態におけるプロペラ推力の合計です．

Z_i($i=1$，2，3，4)は，各プロペラ推力のトリム飛行状態からの変化分であり，機体にはたらく空力を計算するに当たっては，Z_{atrm}を加算する必要があることに注意してください．プロペラ推力が，x軸まわりのトルクL_aとy軸まわりのトルクM_aを発生し，z軸まわりのトルクN_aは，プロペラ・トルクが発生源となります．すなわち次式となります．

$$\begin{bmatrix}L_a\\M_a\\N_a\end{bmatrix}=\begin{bmatrix}l_{army} & & \\ & l_{armx} & \\ & & 0\end{bmatrix}\begin{bmatrix}-1&1&1&-1\\-1&-1&1&1\\0&0&0&0\end{bmatrix}\begin{bmatrix}Z_1\\Z_2\\Z_3\\Z_4\end{bmatrix}$$

$$+\begin{bmatrix}0&0&0&0\\0&0&0&0\\1&-1&1&-1\end{bmatrix}\begin{bmatrix}N_1\\N_2\\N_3\\N_4\end{bmatrix} \quad \cdots\cdots(5)$$

ここで，l_{armx}，l_{army}[m]は隣り合うプロペラ間の距離の1/2であり，それぞれx軸に平行な成分(モータ4⇔1間と2⇔3間)とy軸に平行な成分(モータ1⇔2間と3⇔4間)です．$l_{armx}=l_{armx}=0.0485$としました．N_i($i=1$，2，3，4)は各プロペラトルクの，トリム飛行状態からの変化分です．推力の場合と異なり，トリム値の加算は不要です．いま前提としているトリム飛行状態であるホバリングでは，機体重心周りの空力トルクの総和が0であるためです．

● 重力の計算

重力の方向は，基準座標系(ドローンが地上に着地している状態での機体座標系)のz軸に平行です．後に解説する運動方程式は，機体座標系で数式表現する都合上重力も機体座標系で表現しておく必要があります．3次元空間内のあるベクトルの代数表現を基準座標系と機体座標系の間で相互変換するために用いる行列C_Bを次式に示します．

$$C_B=\begin{bmatrix}q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_3q_0) & 2(q_3q_1+q_2q_0)\\2(q_1q_2+q_3q_0) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_1q_0)\\2(q_3q_1-q_2q_0) & 2(q_2q_3+q_1q_0) & q_0^2-q_1^2-q_2^2+q_3^2\end{bmatrix} \quad \cdots\cdots(6)$$

C_Bの右から機体座標系で表現された3次列ベクトルをかけると，基準座標系で表現した3次列ベクトルが得られます．C_Bの転置行列，C_B^Tの右から基準座標系で表現された3次列ベクトルを掛けると，機体座標系で表現した3次列ベクトルが得られます．重力は，重力加速度の大きさをg[m/s^2]として，基準座標系上では，

$$\begin{bmatrix}0\\0\\-mg\end{bmatrix} \quad \cdots\cdots(7)$$

なので，機体座標系で表現すると次式となります．

$$-g\begin{bmatrix}2(q_3q_1-q_2q_0)\\2(q_2q_3+q_1q_0)\\q_0^2-q_1^2-q_2^2+q_3^2\end{bmatrix} \quad \cdots\cdots(8)$$

なお，機体質量mで割ってあります．これが**図1**の重力計算の中で行う計算です．

ステップ2…運動方程式の計算による加速度/角加速度の算出

● 回転の運動方程式

運動方程式の計算は，機体に加わる力とトルクから，機体の速度/角速度の時間微分を算出するものです．すなわち，力を運動へ変換する処理と言えます．

高校や大学で習う基礎の力学で扱われていて，なじみがあるかと思います．

ただし，ドローンを含め航空機の運動方程式はやや特殊です．力学の授業では，おそらく運動方程式を基準座標系（慣性系）で表現することが多いのですが，航空機の場合は並進／回転ともに機体座標系（動座標系）で表現するので，式の書き方が異なります．以下，機体座標系原点は重心に一致しているものとし，回転の運動方程式は次式となります．

$$\dot{\omega} = -J^{-1}\left\{\omega\times(J\omega)\right\}+J^{-1}N_{\text{total}} \quad \cdots\cdots(9)$$

$$J=\begin{bmatrix} J_{xx} & J_{xy} & J_{zx} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{zx} & J_{yz} & J_{zz} \end{bmatrix} \quad \cdots\cdots(10)$$

$$\boldsymbol{\omega}=\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad \cdots\cdots(11)$$

$$\boldsymbol{N}_{\text{total}}=\begin{bmatrix} L_{\text{a}} \\ M_{\text{a}} \\ N_{\text{a}} \end{bmatrix} \quad \cdots\cdots(12)$$

ここで，演算子「×」はベクトルの外積，Jは慣性行列，J^{-1}はJの逆行列，$\boldsymbol{\omega}$は機体3軸まわりの角速度ベクトル，$\boldsymbol{N}_{\text{total}}$は機体に働く重心周りのトルク・ベクトルです．慣性行列Jの成分のうち，対角成分J_{xx}，J_{yy}，J_{zz}は慣性モーメント，それ以外の成分J_{xy}，J_{yz}，J_{zx}は慣性乗積と呼びます．いずれも単位は[kg m^2]です．機体はx-z平面とy-z平面に関して対称と仮定しているので，$J_{xx}=J_{yy}$および$J_{xy}=J_{yz}=J_{zx}=0$となります．値は$J_{xx}=J_{yy}=1.0\times10^{-4}$，$J_{zz}=2.0\times10^{-4}$としました．

● 並進運動の運動方程式

並進運動とは，重心が移動する運動のことです．運動方程式は，次式となります．

$$\dot{\boldsymbol{v}}_{\text{B}} = -\boldsymbol{\omega}\times\boldsymbol{v}_{\text{B}}+\frac{1}{m}\boldsymbol{F}_{\text{total}} \quad \cdots\cdots(13)$$

$$\boldsymbol{v}_{\text{B}}=\begin{bmatrix} v_{x\text{B}} \\ v_{y\text{B}} \\ v_{z\text{B}} \end{bmatrix} \quad \cdots\cdots(14)$$

$$\frac{1}{m}\boldsymbol{F}_{\text{total}}=\frac{\boldsymbol{F}_{\text{a}}}{m}-g\begin{bmatrix} 2(q_3q_1-q_2q_0) \\ 2(q_2q_3+q_1q_0) \\ q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \quad \cdots\cdots(15)$$

ここで，$\boldsymbol{v}_{\text{B}}$は機体3軸の速度ベクトル，$\boldsymbol{F}_{\text{total}}$は機体重心に働く力ベクトルです．また，$\dot{\boldsymbol{v}}_{\text{B}}$は$\boldsymbol{v}_{\text{B}}$の時間微分で，機体座標系で表現された機体速度を時間微分したものです．基準座標系で表現された機体速度の時間微分（加速度）を，機体座標系に変換したものではないことに注意してください．機体質量m[kg]は，$m=0.0762$としました．

ステップ3…速度／位置および角速度／姿勢の算出

● 機体の速度と位置の算出

機体座標系での速度$\boldsymbol{v}_{\text{B}}$は，時間微分である$\dot{\boldsymbol{v}}_{B}$を時間で1階積分すれば得られます．位置は基準座標系で算出します．行列C_{B}を用いて$\boldsymbol{v}_{\text{B}}$を基準座標系での表現に変換してから積分すれば基準座標系上での位置$\boldsymbol{p}_{N}$が得られます．以上より，次式が成り立ちます．

$$\boldsymbol{v}_{\text{B}}=\int\dot{\boldsymbol{v}}_{\text{B}}dt \quad \cdots\cdots(16)$$

$$\boldsymbol{p}_{\text{N}}=\begin{bmatrix} p_{x\text{N}} \\ p_{y\text{N}} \\ p_{z\text{N}} \end{bmatrix}=\int C_{\text{B}}\boldsymbol{v}_{\text{B}}dt \quad \cdots\cdots(17)$$

● 機体の角速度と姿勢の算出

機体座標系での角速度$\boldsymbol{\omega}$は，時間微分である$\dot{\boldsymbol{\omega}}$を時間で1階積分すれば得られます．次に姿勢は，まずクォータニオンで算出します．AHRSのところで解説したように，角速度$\boldsymbol{\omega}$をクォータニオンの時間微分$\dot{\boldsymbol{q}}$に変換し，時間で1階積分すれば，姿勢を表すクォータニオンが得られます．以上より，次式が成り立ちます．

$$\dot{\boldsymbol{q}}=\frac{1}{2}\boldsymbol{q}\otimes\boldsymbol{\omega}=\frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad \cdots\cdots(18)$$

$$\boldsymbol{q}=\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}=\int\dot{\boldsymbol{q}}dt \quad \cdots\cdots(19)$$

クォータニオンからオイラー角ϕ, θへの変換方法はAHRSのところで解説したので参考にしてください．

システム同定でパラメータをチューニング

機体の質量や慣性モーメント，寸法など上で解説したモデルには幾つかのパラメータがあります．ここでは，これらを定める方法について解説します．

● 計測可能な量は計測値を用いる

質量や寸法といった，計測器を用いて容易に計測できるものは，計測した値を用います．ST-DRONEのモデルでは，m，l_{armx}，l_{army}の3つが該当します．値は前述した通りです．

● 計測が困難なものは解析計算から暫定値を用いる

直接計測は難しいものの，解析計算である程度値が分かるものは，まずは計算した値で仮設定し，この後に述べるシステム同定の過程において必要であれば調

整を行います．ST-DRONEのモデルでは，慣性モーメントJ_{xx}，J_{yy}，J_{zz}が該当します．

筆者は次のようにして計算しました．胴体のフレームについては，ST社のウェブ・ページから3次元CADの図面データをダウンロードできるので，この図面データを用いて胴体フレームの慣性モーメントを推定しました．

次に，フレームに取り付ける各部品の重量と取り付け位置を測り，各部品の慣性モーメントを胴体中心まわりに換算して積算しました．なお，Li-Po電池とFCU基板は，密度均一な正方形の板と仮定して慣性モーメントを算出しました．解析計算により，$J_{yy}=1.10\times10^{-4}$，$J_{zz}=2.08\times10^{-4}$[kg m^2]が得られ，システム同定を経て前述した値にしました．

● 計測も解析計算も困難…そんなときは「システム同定」で値を得る

モデル・パラメータの中には計測も解析計算も困難なものがあります．ST-DRONEのモデルでは，アクチュエータや空力の特性に関係するK_T，τ_T，K_Qが該当します．

この場合は，システム同定により値を得ます．実物のドローンを飛行させ，操縦かんをさまざまに操作して，そのときの操作量や機体の加速度/角速度を同時に記録する実験（システム同定実験）を行います．そして，実験時と同様の条件でモデルのシミュレーションを行い，シミュレーション結果と飛行実験結果ができる限り一致するようにパラメータの値を調整していき，値を決定します．

● システム同定しやすいドローンの操作方法

よく行われる操縦方法は，低周波から高周波までさまざまな周波数で加振し，かつ振幅を大きめに操作する方法です．ただし，墜落しないように十分注意しながら操縦します．詳しく知りたい方は，システム同定理論を学んでください．リファレンス・デザインのFCUのソースコードのままでは機体をあまり大きく加振できなかったので，`rc.h`の`PITCH_MAX_DEG`と`ROLL_MAX_DEG`を`20`から`60`へ，`YAW_MAX_DEG`の数値部分を`120.0`から`360.0`へ，それぞれ変更して，操縦かん操作に対して機体の動きが大きくなるようにしました．

● システム同定実験からパラメータを調整

▶プロペラ用パラメータK_T，τ_T

K_T（ゲイン），τ_T（時定数）については，スロットル操縦かんを加振しモータのPWM指令値とz軸加速度を記録する実験を行ってシステム同定しました．その際のデータを**図2**に示します．上のグラフは加速度（プロペラ推力合計を機体質量で割った値．重力加速度を含む），下のグラフは同定実験時の各モータのPWM指令値です．スロットルを低周波数から高周波数で上下に大きく動かしています．

上のグラフの2本の線は，破線が同定実験時に計測された値で，実線が同定実験時と同様のPWM値をモデルへ入力しシミュレーションして得られた値です．実線が破線にできるだけ近くなるようにK_T，τ_Tの値を調整しました．

▶トルク用パラメータK_Q

K_Q（ゲイン）については，ラダー操縦かんを加振し，ヨーの角速度目標値と角速度を記録する実験を行ってシステム同定しました．モデルの方にもヨー角速度のPID制御を付けた状態でシミュレーションし，パラメータ調整を行いました．その際のデータを**図3**に示します．グラフの3本の線は，破線が同定実験時に計測されたヨー角速度ω_zの値で，実線が同定実験時と同様のヨー角速度目標値をモデルへ入力しシミュレーションして得られたヨー角速度ω_zの値，1点鎖線が同定実験時にラダー操縦かん操作によって生成したヨー角速度目標値です．実線が破線にできるだけ近くなるようにK_Qの値を調整しました．

● 姿勢変化に関するシステム同定精度の確認

エルロン操縦かんを加振し，ロールの角速度目標値と角速度を記録する実験を行いました．姿勢変化は，4つのプロペラの推力の変化により行うため，関係するモデル・パラメータはK_T，τ_Tとなり，新たなパラメータはありません．念のため姿勢変化運動がきちんとモデリングされているかどうか確認することが目的です．

データを**図4**に示します．グラフの3本の線の意味は，上記のヨー角速度の場合と同様で，ヨーがロールに変わります．この結果から，7～9sの高周波で加振しているところで同定実験の方が同定結果よりも大きな角速度振幅となっているという違いが見られますが，それ以外は，振幅，周波数，時間遅れについておおむね合っていて，モデリングがきちんと行えていることが分かります．

● システム同定実験に合わせてFCUのソースコードを変更した

システム同定実験を実施するにあたり，FCUのソースコードに手を加えています．`rc.h`の変更箇所はすでに述べた通りです．`main.c`と`debug.c`の変更箇所は**リスト1**と**リスト2**の通りです．

`main.c`は，システム同定に必要なデータをPCへ送信するための変更です．上下加速度（推力）モデルのシステム同定実験時は，時刻を示すカウンタとz軸

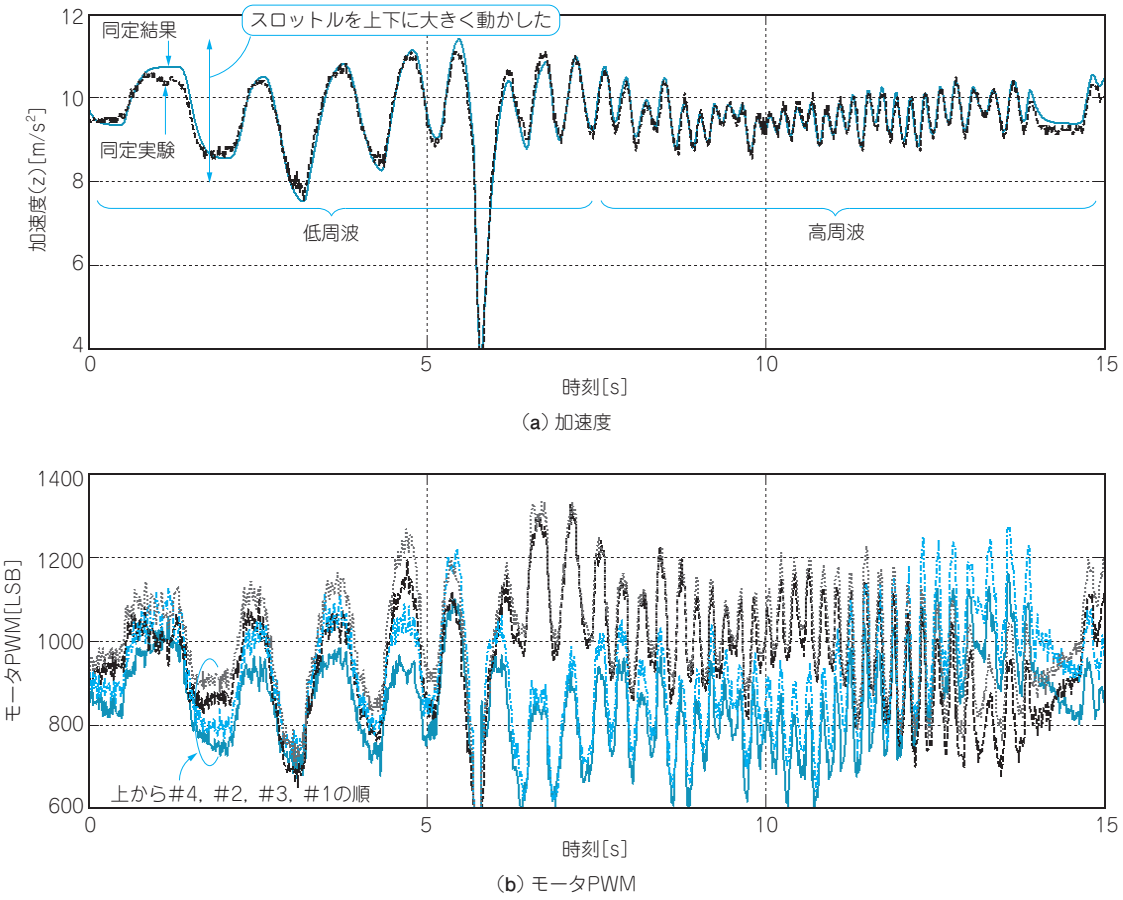

図2　システム同定実験の結果その1…上下加速度（推力）モデル

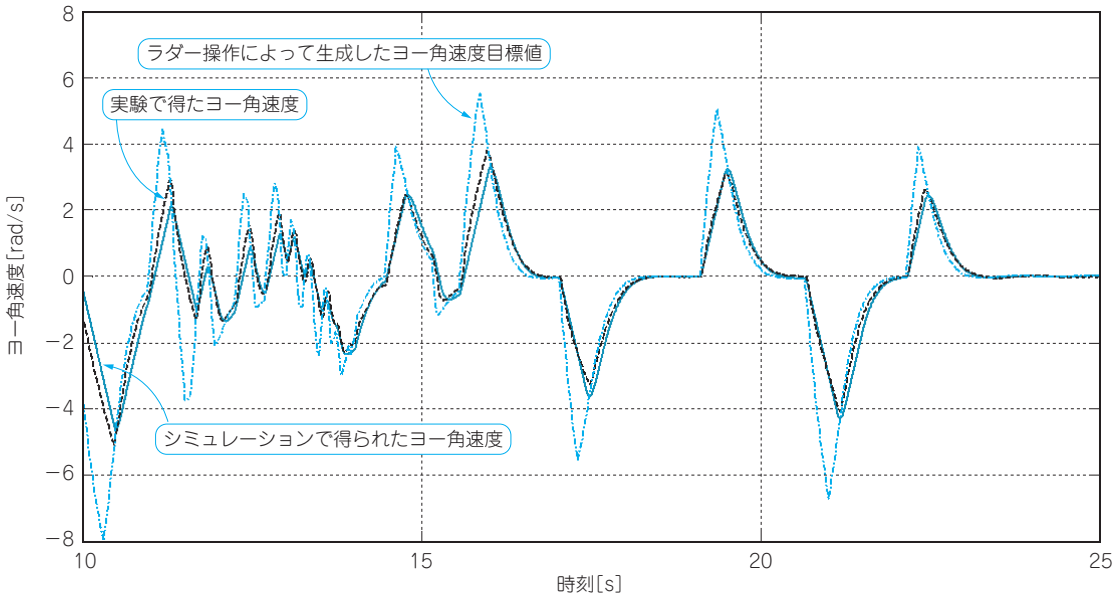

図3　システム同定実験の結果その2…ヨー角速度（トルク）モデル

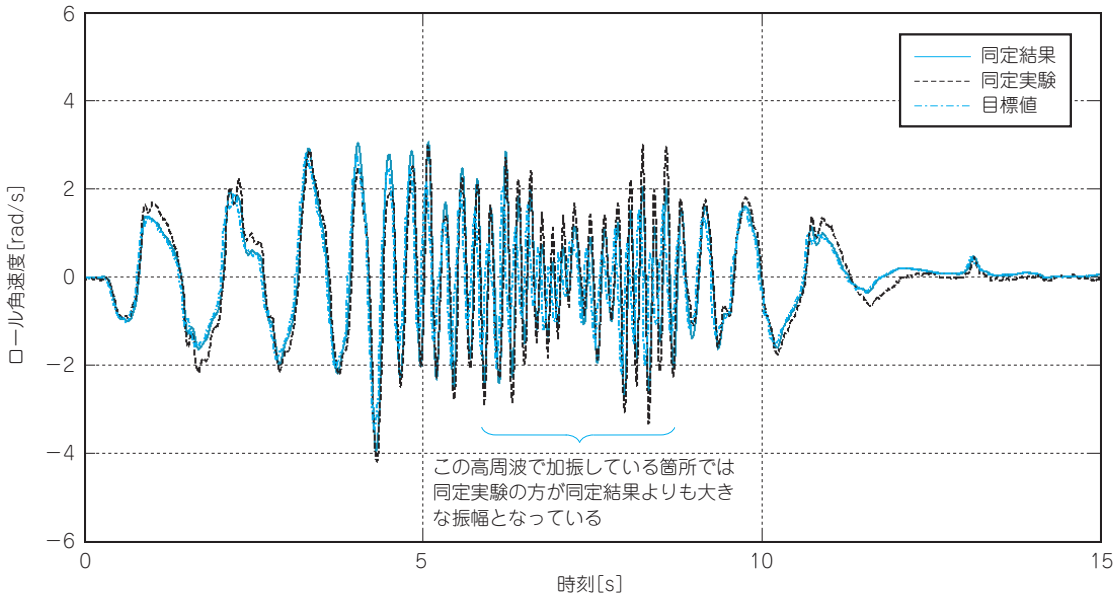

図4　システム同定実験の結果その3…ロール角速度モデル

リスト1　システム同定実験用に**main.c**を変更した

```
uint32_t cnt_usr_1 = 0;
      ⋮
   if ((count1 % 2) == 0)
   PRINTF("%d %d %d %d %d %d %d %d %d\n",
    (int)(cnt_usr_1 / 10),
    (int)(acc_fil_int.AXIS_Z / 10),
    (int)(gyro_fil_int.AXIS_X / 1000),
                 (int)(gyro_fil_int.AXIS_Y / 1000),
                 (int)(gyro_fil_int.AXIS_Z / 1000),
    ((int)motor_pwm.motor1_pwm) / 2,
                   ((int)motor_pwm.motor2_pwm) / 2,
                   ((int)motor_pwm.motor3_pwm) / 2,
    ((int)motor_pwm.motor4_pwm) / 2);
      ⋮
```

元の155行目に挿入：カウンタ

【上下加速度モデルのシステム同定実験時】
元の403行目に挿入：XBeeによるフライトデータのダウンリンク

```
   if ((count1 % 2) == 0)
 PRINTF("%d %d %d %d %d %d %d %d %d\n",
 (int)(cnt_usr_1 / 10),
 (int)(acc_fil_int.AXIS_Z / 10),
 (int)(gyro_fil_int.AXIS_X / 1000),
        (int)(gyro_fil_int.AXIS_Y / 1000),
        (int)(gyro_fil_int.AXIS_Z / 1000),
        (int)(pid.x_s1 * 57.29578F), (int)(pid.y_s1
        * 57.29578F), (int)(pid.z_s1 * 57.29578F));
      ⋮
  // PRINTF("%d\t%d\t%d\t%d\t%f\t%f\t%f\t%f\t%f\n",
     gELE, gAIL, gRUD, gTHR, motor_pwm.motor1_pwm,
     euler_ahrs.thx * 57.3, euler_ahrs.thy * 57.3,
        euler_rc.thx * 57.3, euler_rc.thy * 57.3);
      ⋮
++cnt_usr_1; /* 800Hz */
```

【角速度モデルのシステム同定実験時】
【制御実験時】
元の403行目に挿入：XBeeによるフライトデータのダウンリンク

元の432行目をコメントアウト

元の903行目に挿入：カウンタのインクリメント

リスト2　システム同定実験用に**debug.c**を変更した

```
static char temp[255];
      ⋮
//    HAL_UART_Transmit(&huart1, (uint8_t *)str,
                                  len, 1000);
      ⋮
HAL_UART_Transmit_IT(&huart1, (uint8_t *)str,
                                  len);
```

元の14行目：`static`を追加

元の26行目をコメントアウト

元の27行目に挿入：UART送信を割り込み送信にする

加速度，3軸の角速度，4つのモータのPWM指令値を送信するようにしました．角速度モデルのシステム同定実験時と，後述する制御系設計後の制御実験時は，時刻を示すカウンタとz軸加速度，3軸の角速度とそれぞれの目標値を送信するようにしました．データは，約80Hzで送信します．できれば姿勢や姿勢目標値など，もっと多くのデータをより高いレートで送信したいのですが，無線のスループットの制約により必要最小限にしています．

`debug.c`については，`PRINTF()`関数を呼び出した際に実行されるUARTによるデータ送信を，割り込みを使って送信する方式にするための変更を実施しています．割り込みを使うことで，`PRINTF()`関数呼び出し時のメイン・ループの遅延を防ぐようにしました．

ふじわら・だいご