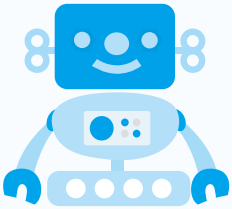
06

制御工学から見たソフトウェア
―ロボット製作における制御とソフトウェア―



三輪昌史（徳島大学大学院ソシオテクノサイエンス研究部）

基

組込み教材用ロボットのライブラリには前進関数 Forward() などが用意されている．そのため“プログラミング言語の文法を学んでいれば容易に組込みシステムを開発できる”という錯覚に陥りやすい．しかし実際に Forward() 関数を用いると，ロボットは 1 メートルも進まないうちに曲がってしまう．これは，ライブラリ関数は単にモータに対する回転指令を送るだけで，実際の回転角度などについて関知しないことが原因である．対策として何度も実験を繰り返してロボットの癖を調べてプログラムに組み込むなど，制御工学の知識なしに何とか真っ直ぐ進むように細工すると，可読性が低くメインテナンスが困難なプログラムになる．そもそもこのように作成したシステムは一見動作していても不安定であり，劣化・故障する危険さえある．本稿では，組込みシステム開発に携わるのに必要な制御工学の基礎について紹介する．

制御とは，“ある目的に適合するように，対象となっているものに所要の操作を加えること”[1] と定義されている．たとえば私たちが無意識に行う，歩くなどの動作も制御である．制御の目的としては，大きく 2 つがある．1 つ目は，不安定な対象を安定にし，使えるようにすることである．もう 1 つは応答の改善である．制御での安定・不安定について説明する．図 -1(a) のボールは，何か外力が加わると転がり落ちてしまう．一方，図 -1(b) ではボールは外力によって移動するが，外力がなくなると元の位置に戻る．よって図 -1(a) は不安定を，図 -1(b) は安定を示している．図 -1(a) に盛り土をしたのが図 -1(c) になるが，この場合ボールが盛り土を超えない限り，図 -1(b) のようにボールは元の位置に戻ることができる．盛り土により範囲内では安定になったわけである．この盛り土が制御で，“不安定な系を安定にした”，ということになる．また，図 -1(d) では盛り土の傾斜が大きくなっており，図 -1(c) の場合よりもボールが早く戻るようになっている．これが“応答の向上”に相当する．

数学的な安定と不安定の定義は次のようになる．“入力がないときには平衡状態にあるシステムに対し初期条件を与えた場合，時間の経過とともに平衡状態に復帰すれば安定，しなければ不安定である”．この定義に基づくと，安定かどうかはシステムの微分方程式，または伝達関数で判別できる．本稿では比較的簡単な伝達関数を扱う古典制御理論の考えに基づいた説明を行う．伝達関数は，システムの入出力の関係をラプラス変換で表したもので，出力のラプラス変換を入力のラプラス変換で割ったものである．

$$G(s)=\frac{Y(s)}{U(s)} \tag{1}$$

ここで，$G(s)$ は伝達関数，$Y(s)$ は出力のラプラス変換，$U(s)$ は入力のラプラス変換である．図 -2 に示す，

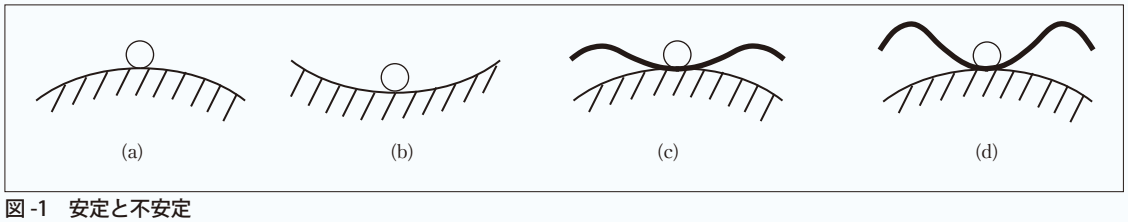
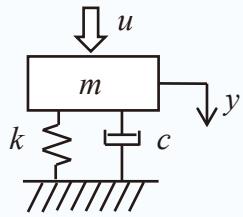


図 -1　安定と不安定



u：外力(入力)
y：位置(出力)
m：質量
k：ばね定数
c：減衰係数

$$m\frac{d^2y}{dt^2}+c\frac{dy}{dt}+ky=u$$

$$ms^2Y(s)+csY(s)+kY(s)=U(s) \longrightarrow G(s)=\frac{Y(s)}{U(s)}=\frac{1}{ms^2+cs+k}$$

図 -2　ばね振動系とその伝達関数

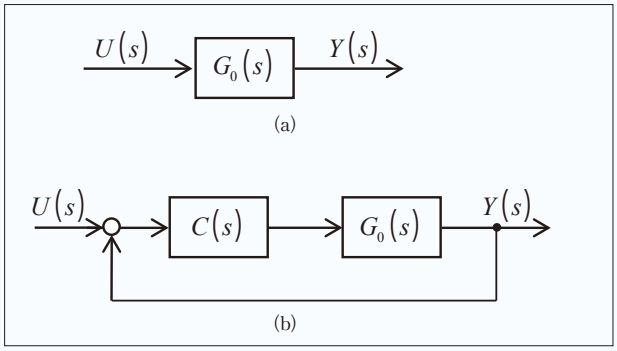


図 -3　伝達関数のブロック線図での表現

外力 u により質量の位置 y を制御しているばね振動系では，伝達関数は質量 m，ばね定数 k，減衰係数 c で決まる系の特性を表している．

系に入力がある場合，つまりロボットなどの制御系が動作している場合の安定性では，“入力が有界であるとき，出力も有界であれば安定である”という定義の入出力安定を用いる．この入出力安定が成立するのは制御系の伝達関数が安定な場合である．系の伝達関数を数学的に解析することで，安定が判別できる．

さて制御ではフィードバックなどを用いて系の伝達関数を改善することができる．図 -3(a) は伝達関数 $G_0(s)$ のブロック線図を，図 -3(b) は $G_0(s)$ にコントローラ（制御側）$C(s)$ をフィードバックで加えた系のブロック線図を示す．図 -3(b) のブロック線図を伝達関数で記述すると，

$$G(s)=\frac{C(s)G_0(s)}{1+C(s)G_0(s)} \tag{2}$$

となる．ここで，$G_0(s)=\dfrac{A(s)}{B(s)}$ だとすると，

$$G(s)=\frac{C(s)A(s)/B(s)}{1+C(s)A(s)/B(s)}=\frac{C(s)A(s)}{B(s)+C(s)A(s)} \tag{3}$$

となる．伝達関数の分母を 0 とした式を特性方程式という．この特性方程式の解（根）のすべてにおいて実部が負であれば，その伝達関数の系は安定であることが数学的に判明しており，特性方程式の解で安定判別が行える．図 -3(b) の例では，フィードバックを加えることで特性方程式が $B(s)$ から $B(s)+C(s)A(s)$ に変更された．つまり，制御を加えるということは伝達関数を変更することである．

$C(s)$ には PID 制御がよく使用されている．この制御方法では目標値と出力の誤差を用いて系への入力を決定する．誤差の大きさに比例定数（ゲイン）をかけた値を入力とすると，誤差が小さければ小さい入力，大きければ大きい入力となる．この入力により系は誤差を減らすように応答する．これを P 制御（比例制御）という．P は Proportional の頭文字である．同様に誤差の積分に比例した入力を与える制御を I 制御（積分制御）という．I は Integral の頭文字である．I 制御により外部的な要因（外乱）による，P 制御では対応できない定常偏差をキャンセルできる．P 制御ではゲインを大きくすると応答が早くなるが，大きすぎると行き過ぎ（オーバーシュート）や振動が始まる．これは応答が早すぎることが原因であり，同時に誤差の変化も早くなっている．そこで誤差の速度（微分）に比例した入力をブレーキとして導入することで行き過ぎや振動を抑えることができる．これを D 制御（微分制御）という．D は Differentiation の頭文字である．このように，PID の各制御では誤差やその積分，微分に対してゲインを設定することで伝達関数を改善する．制御対象の伝達関数 $G_0(s)$ が分かっていれば，$C(s)$ のゲインを決めることで系全体の伝達関数 $G(s)$ を改善できる．

逆に言えば，望ましい応答をする伝達関数 $G(s)$

図-4　クアッドコプタ

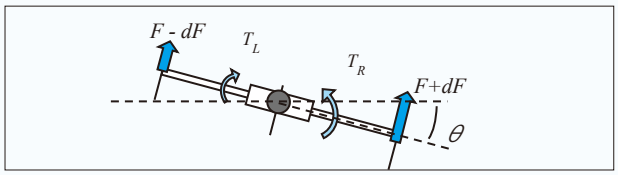


図-5　ロール軸回りの回転モデル

をまず決定し，式(2)からこれを実現する $C(s)$ を逆算できる．これが制御系の設計である．

次に例として4個のサーボモータで駆動されるプロペラを持つクアッドコプタ（図-4）を考える．このサーボモータは目標速度にモータの回転速度が追従するモータである．4つのプロペラが同じ速度で回転し，同じ大きさの推力が発生することで機体は水平に維持される．実際には各部品ごとの個体差，風などの外乱によって，クアッドコプタの姿勢は水平からわずかに傾く．機体姿勢を水平に維持するには，機体姿勢に関する制御系を構成する．そのためには機体姿勢を検出するセンサが必要である．

図-5はクアッドコプタのロール軸回りの姿勢に関するモデルである．図中では左右のサーボモータの推力差 $2dF$ に起因する差動トルク $T=T_R-T_L$ によって機体は左に回転し，水平状態への復帰を試みている．姿勢に関する微分方程式は次式となる．

$$I\frac{d^2\theta}{dt^2}=T \tag{4}$$

ここで，I はクアッドコプタのロール軸回りの慣性モーメント，θ はクアッドコプタの傾斜角度である．これを伝達関数 $G_0(s)$ にすると次式となる．

$$G_0(s)=\frac{\Theta(s)}{T(s)}=\frac{1}{Is^2} \tag{5}$$

できあがった伝達関数にトルク T を発生させるサーボモータの伝達関数，機体の傾きを検出するセンサ，サーボモータへの入力を決定するコントローラを加えて姿勢制御系を構成する．この系のブロック線図を図-6に示す．θ_{in} は目標角度，$U(s)$ が機体への入力，$G_{Multi}(s)$ はサーボモータも含めたマルチコプタのロール軸に関する伝達関数，θ_{in} は出力である機体の角度である．適切な差動トルク T を発生させる入力 $U(s)$ を決定するコントローラ $C(s)$ を設計することで，外乱等で機体が傾いても自動的に水平を維持し，安全な飛行ができるようになる．

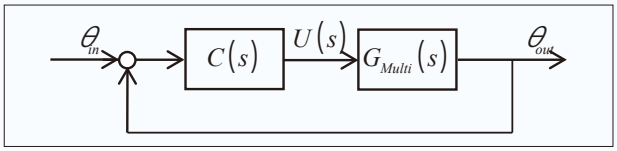


図-6　姿勢制御系のブロック線図例

さて，$C(s)$ によって伝達関数を改良できることを説明した．実際に制御系を構成するのは制御対象と，出力を検出するセンサと，$C(s)$ である．制御対象とセンサは機械である．$C(s)$ は，昔は機構（ワットのガバナー）や電気機械回路（紫電改の自動空戦フラップ）などハードウェアで実装されていたが，現在ではマイコンとソフトウェアの組込み技術で実装される．目標値と出力値の情報を用い，$C(s)$ の動作をソフトウェアで実装することで，制御を実現している．

本稿では比較的簡単な伝達関数を扱う古典制御理論の考えに基づいた説明を行った．制御では古典制御理論のほかに，離散時間で設計するディジタル制御，状態方程式でシステムを記述する現代制御理論や，システムの数学モデルの不確かさの影響を抑制する H^∞ 制御理論など，さまざまな制御理論が報告されている．それぞれに特徴がある制御理論であるが，実際に使用するためには，対象に合わせた数学モデルの構築と実装が重要である．実装ができなければ，組込み制御は成り立たない．制御理論に基づいた組込み実装が重要である．

参考文献
1) 添田　喬，中溝高好：自動制御の講義と演習，日進出版(2007).

(2014年10月3日受付)

三輪昌史　miw@tokushima-u.ac.jp
徳島大学大学院ソシオテクノサイエンス研究部准教授，博士（工学），無人航空機に関する研究に従事．