

東洋大学大学院工学研究科機能システム専攻

マイクロメカトロニクス・制御特論 講義資料

2010 年度
山川聡子

目次

0. はじめに	1
1. 運動方程式	5
2. 古典制御におけるシステムの表現（伝達関数）	11
3. 現代制御におけるシステムの表現（状態空間表現）	14
4. システムの安定性	17
5. システムの極と時間応答	20
6. 古典制御と周波数領域での解析	23
7. 現代制御における状態フィードバック制御	28
8. サーボ系	35
9. ロボットアームの制御	40
授業の課題(レポート)	43
付録	45

マイクロメカトロニクス・制御特論

0. はじめに

制御工学は、ロボットや機械を希望通りに動かすために不可欠な知識である。動的システムの特徴を捉え、解析し、動かす制御工学の考え方は、メカトロニクスに限らず、経済、生体活動などにも広く応用できる。

近年、企業においても制御系設計の支援ツールとしてコンピュータのソフトウェアを用いることが多くなっている。そこで、本講義では、自動車産業などでも良く用いられている MATLAB、制御系設計に有効な数式処理ソフトである Mathematica を用いた演習を取り入れて、制御対象のモデリングと制御系の解析法、および代表的な制御系設計法について講義を行う。講義の終わりには、2リンクのロボットアームを対象として実際に制御則の設計を行う。

本講義では、重要なポイントに絞った説明を行い、定理の証明や数学的な説明は極力省略する。したがって、古典制御の基礎的な知識、および、物理と行列の基礎的な知識は、各自復習のうえ、受講してほしい。

授業の概要

スケジュール

0. 授業の概要、制御系設計支援ソフトの紹介
1. モデリング（運動方程式）
2. システムの表現1（伝達関数）
3. システムの表現2（状態空間表現）
4. システムの時間応答と安定性1（システムの安定性）
5. システムの時間応答と安定性2（システムの時間応答と極）
6. 制御系設計1（古典制御と周波数領域での解析）
7. 制御系設計2（現代制御における状態フィードバック制御）
8. 制御系設計3（サーボ系）
9. ロボットアームの制御
- 10-14. コントローラ設計（演習）

※受講者の状況に応じて説明内容や進度を調整する。

評価法

2回程度のレポート課題を出します。内容によって評価します。

参考文献例

- ランダウ=リフシッツ：力学，東京図書(1974)... 訳本．剛体運動の基本.
増淵，川田：システムのモデリングと非線形制御，コロナ社(1996)... 様々なシステム例あり
北川，掘込，小川：自動制御工学，森北出版(2001)... 大学のテキストレベルの古典制御
吉川：古典制御論，昭晃堂(2004)... 上記よりも詳しい古典制御の本
美多，小郷著：システム制御理論入門，実教出版(1979)... ポイントをおさえた現代制御の本
児玉，須田著：システム制御のためのマトリクス理論，コロナ社(1978)... 詳しい行列の本

制御系設計支援ソフト

制御を人の手で行うのではなく、機械や電気回路、コンピュータなどを用いて自動的に行うことを自動制御という。自動制御は、18世紀の産業革命の時代、J.Wattの蒸気機関の回転数を一定に保つガバナという装置から始まったといわれている。その後、安定性などが数学的に解析され、制御工学といわれる分野が発達した。19世紀の中頃には、BodeやNyquistなどによって、周波数領域での解析が行われ、古典制御と呼ばれる解析手法が成熟した。1950年頃になると、制御対象が複雑になり、多入力多出力システムを制御するための現代制御論が登場した。KalmanやLuenbergerらによって示された現代制御は、最適性を取り入れた体系的な理論である。しかし、システム表現やコントローラ設計に行列とその方程式を用いること、制御対象のモデルを必要とすることなどから、経験的な古典制御に比べて産業分野での普及は滞っていた。1980年代になると、ポスト現代制御として、 H_∞ 制御といわれる制御法が登場した。 H_∞ 制御は、外乱やモデル化誤差に対してもシステム性能を保証するロバスト性を考えた制御法である。これは、古典制御の周波数領域での解析と、現代制御の多入力多出力を取り扱うための定式化など、二つの方法の性質を合わせ持っている。しかし、コントローラ設計には、現代制御と同等、もしくはそれ以上の数学的知識を必要とした。

ところで、1980年頃に、MathematicaやMATLABといわれる数式処理ソフトが登場した。特に、1984年に製品化されたMATLABでは制御系設計用のアプリケーションが多数提供された。このアプリケーションのおかげで、古典、現代制御のみならず、ロバスト制御などの複雑な制御系設計における繰り返し計算なども、非常に簡単に行うことができるようになった。また、DSP(Digital signal processor)とのインターフェースも提供されたため、シミュレーションから実験までが手軽にできる環境ができた。その結果、多くの企業が利用し、 H_∞ 制御は、自動車や航空機関連の企業でも実際に利用されるようになった。

以下では、MathematicaとMATLABという二つの数式処理ソフトについて説明する。

0. 数式処理ソフト

数式処理ソフトは、その名の通り、数式を取り扱うことができるソフトウェアである。一般のコンピュータ言語との違いを有理数の取り扱いの例で示そう。

問い： $\frac{1}{6} \times 6$ はいくつか？

もちろん、答えは1である。さて、この計算を最も手近なコンピュータである電卓で行ってみよう。まず、 $1/6$ を計算すると、答えは0.1666666667と表示される。この時点で、 $1/6$ ではない。しかし、これに6をかけると1になる。この調子で、つぎのような計算をしてみよう。

$$\underbrace{\frac{1}{6} \times \frac{1}{6} \times \cdots \times \frac{1}{6} \times \frac{1}{6} \times \frac{1}{6}}_{n \text{ 回}} \times \underbrace{6 \times 6 \times \cdots \times 6 \times 6 \times 6}_{n \text{ 回}}$$

もちろん、自然数 n がいくつであっても厳密解は1である。電卓で計算した場合、 $n=3$ 回程度なら、1に戻るであろう。もっと n を大きくしたらどうなるか？これは、電卓の性能を測るのに良い実験でもある。性能の悪い電卓だと $n=10$ 程度で答えが0.999999となり、1にならない。

これは、電卓やコンピュータの計算丸め誤差に起因する。コンピュータは、0と1の間を有限の数字で分割して取り扱うため、この誤差が生じてしまう。

これに対して、数式処理ソフトは、 $1 \div 6$ の答えを $\frac{1}{6}$ と返すことができるソフトである。つまり、 $1/6$ を $1 \div 6$ を計算せずに、1を6で割った値と記憶している。同じように $1/a$ といったように、未知変数を用いた計算も可能である。さらに、数式処理ソフトの多くは様々な数学公式のライブラリを持っており、微分積分、方程式の求解なども可能である。

このような数式処理ソフトは、1960年代後半から研究されてきており、以下で紹介するMathematicaやMATLAB以外にも、ReduceやMapleなど、古くから多くのソフトが開発されている。

1. MATLAB

matrix Laboratoryの略である。

行列、ベクトル計算、グラフなどのライブラリを持つ技術計算言語（環境）である。DSP用コードを作成するコンパイラなどが用意されており、コントローラ設計、シミュレーションから実装までを行うことができる。そのため、企業におけるコントローラ設計でも利用されている。

1970年後半 Cleve Moler(ニューメキシコ大)が開発。

1984年 John N. Littleが商品化 Mathworks社
↑制御工学が専門... 制御分野で普及

2. Mathematica

多様な関数を持つ数式処理ソフトであり、ユーザーが作成したパッケージといわれる関数も多く公開されている。パッケージを集めたアプリケーションソフトも販売されている。MATLABが実装まで視野に入れた展開をしているのに対し、Mathematicaは大学や研究機関向けに豊富なアルゴリズムを取り揃えているイメージが強い。

1981年 Stephen Wolframが前身のSymbolic manipulation programを商業リリース
↑理論物理、複雑系が専門(カリフォルニア工科大)

1988年 Mathematica発売. Wolfram research社

演習「数式処理ソフト Mathematica を使ってみよう」

Mathematica のカーネル

- ・起動後の最初の演算では「Kernel」を読み込みに少し時間がかかる。（作業領域の確保）
- ・計算したい式を入力，その後「Shift」＋「Enter」で実行。
- ・一度読み込んだら，その「Kernel」を shut down するまでは変数の値は記憶される。

厳密値と近似値

- ・「1」や「-3」は整数として取り扱われ，厳密解が返されます。「1.」や「-3.00」は実数として取り扱われ，近似値が返される。
- ・厳密な値を近似値にするとき → $N[x, n]$ で， n 桁までの x の近似値。
- ・厳密値で計算すると時間がかかる。

簡単な関数

- ・関数は大文字から始まる．引数は[]でくくる。
- ・平方根： $\text{Sqrt}[x]$
三角関数： $\text{Sin}[x]$ ， $\text{Cos}[x]$ ， $\text{Tan}[x]$ ， $\text{ArcSin}[x]$ など（角度は radian）
対数： $\text{Log}[x]$ （自然対数）
指数関数： $\text{Exp}[x]$
- ・微分： $\text{D}[a*t^2+1]/t, t]$
不定積分： $\text{Integrate}[a*t^2+1, t]$
- ・ラプラス変換： $\text{LaplaceTransform}[e^{-t}, t, s]$ ，
逆ラプラス変換： $\text{InverseLaplaceTransform}[s+1, s, t]$
- ・方程式を解く： $\text{Solve}[5x^2+1=0, x]$

1. 次の計算を実行してみましょう。

2^{100} , $2.^{100}$, $N[2^{100}, 2]$

$\text{Pi}/2$, $\text{Pi}/2.$, $N[\text{Pi}/2]$ （ただし，Pi は円周率 π を表す）

$I*I$, I^3 , $(1+I)^2$ （ただし，I は虚数単位を表す）

2. 方程式を解いてみましょう。

$\text{Solve}[x^3+x^2+1=0, x]$

$\text{Solve}[x^5+x^2+1=0, x]$ （5次以上は解の公式がないので，解が出ません）

$\text{NSolve}[x^5+x^2+1=0, x]$ （5次以上でも，数値解は得られます）

$\text{DSolve}[x'[t]+x[t]=1, x[t], t]$ （微分方程式の一般解）

$\text{DSolve}[\{x'[t]+x[t]=1, x[0]=0\}, x[t], t]$ （初期値がある場合）

モデリング

1. 運動方程式

良い制御を行うためには、制御対象の運動学的、力学的特徴を理解しておく必要がある。この特徴は、客観的な表現を用いて記述し、他者と共有できるようにすることが望ましい。これは、その後の問題点の原因究明や制御法の改善にも役立つだろう。

工学的には、制御対象の運動の特徴は微分方程式を用いて表現されることが多い。その代表的なものとして運動方程式(*equation of motion*)があげられる。運動方程式として、最もポピュラーな表現は、ニュートンの第二法則(*Newton's second law*)であろう。質点の質量と加速度をかけたものは、質点に加わる力の総和に等しいというものである。質量を m 、時刻 t での位置を $x(t)$ 、質点にかかる力の総和を $F(t)$ とすれば、

$$m\ddot{x}(t) = F(t) \tag{1-1}$$

で表わされる。

さて、複数の質点間の相互作用がある場合など、系(=システム)が複雑になると、質点にかかる力 $F(t)$ が容易に導出できないことがある。特に、回転運動と並進運動が混ざっているときなどは、導出が煩雑になることが多い。このような場合に、ラグランジュ方程式(*Lagrange's equations*)を用いると、運動方程式が比較的容易に得られる。

そこで、本章ではラグランジュ方程式について説明する。なお、ニュートンの運動方程式とラグランジュ方程式から得られる運動方程式は、同じ系の運動を表わしており、等価である。

1-1 一般化座標 (*generalized coordinate*)

ある系の位置をすべて決めるために必要な独立な量を自由度(*degree of freedom*)と言う。例えば、3次元空間に1つの質点が浮いている系ならば、自由度は3である。また、系の位置を決めるのに十分な任意の量を一般化座標という。たとえば、3次元空間に固定した直交座標系 O -xyz の座標は、前述した系の一般化座標のひとつである。一般化座標は無数の組み合わせが選択できるので、対象とする問題で扱いやすい座標を選べばよい。

一般化座標の値をすべて与えれば、ある時刻での系の位置(=座標)は確定する。しかし、この時刻以降に、位置がどのように変化するかは定まらない。力学的な問題においては、ある時刻での位置と速度がすべて与えられると、その時刻以降の位置変化が予測できる。これは、運動方程式を解く際に、初期位置と初期速度が与えられれば、解が一意に決まることで理解できるだろう。

1-2 最小作用の原理 (*principle of least action, Hamilton's principle*)

系の位置を表わす一般化座標を $\mathbf{q} = (q_1, q_2, \dots, q_n)$ 、その導関数(一般化速度)を $\dot{\mathbf{q}} = (\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n)$ とおく。力学系の運動は、1-1節で述べたように \mathbf{q} と $\dot{\mathbf{q}}$ で決まるので、系の運動はある関数 $L(\mathbf{q}, \dot{\mathbf{q}}, t)$ で特徴付けられる。この関数 L をラグランジアン *Lagrangian* という。

ここで、時刻 t_1 で位置が q_1 にあり、時刻 t_2 で q_2 になるとする。このとき、系は、

$$S = \int_{t_1}^{t_2} L(q, \dot{q}, t) dt \quad (1-2)$$

が最小の値をとるように変化する。これを**最小作用の原理**といい、古典力学の基礎的な原理である。

最小作用の原理を満たすための q と \dot{q} の条件を求めてみよう。

まず、 $q=q(t)$ が作用 S を最小にする解だとしよう。すなわち、 q が $q(t)$ から少しでもずれると、 S は大きくなる。 q_1 から q_2 に変化する途中で $q(t)$ から微小な値 $\delta q(t)$ だけずれたとする。このとき、作用 S は、

$$\delta S = \int_{t_1}^{t_2} L(q + \delta q, \dot{q} + \delta \dot{q}, t) dt - \int_{t_1}^{t_2} L(q, \dot{q}, t) dt \quad (1-3)$$

だけ増加する。第一項目を $\delta q(t)=\delta \dot{q}(t)=0$ の周りでテーラー展開すると、

$$\begin{aligned} \delta S &= \int_{t_1}^{t_2} \left(L(q, \dot{q}, t) + \frac{\partial L(q, \dot{q}, t)}{\partial q} \delta q + \frac{\partial L(q, \dot{q}, t)}{\partial \dot{q}} \delta \dot{q} + \dots \right) dt - \int_{t_1}^{t_2} L(q, \dot{q}, t) dt \\ &= \int_{t_1}^{t_2} \left(\frac{\partial L(q, \dot{q}, t)}{\partial q} \delta q + \frac{\partial L(q, \dot{q}, t)}{\partial \dot{q}} \delta \dot{q} \right) dt + O(\delta^2) \end{aligned} \quad (1-4)$$

である。 $O(\delta^2)$ は、 $\delta q(t)$ と $\delta \dot{q}(t)$ の 2 次以上の項であり、第一項目より十分小さいので無視する。 S が $q(t)$ で極小になるためには、 $\delta S=0$ が必要である。この条件は、

$$\int_{t_1}^{t_2} \left(\frac{\partial L}{\partial q} \delta q + \frac{\partial L}{\partial \dot{q}} \delta \dot{q} \right) dt = 0 \quad (1-5)$$

となる。ここで、 $L(q, \dot{q}, t)$ は単に L と表わしている。ところで、(1-5)の左辺第二項目は、部分積分をすると、

$$\int_{t_1}^{t_2} \frac{\partial L}{\partial \dot{q}} \delta \dot{q} dt = \left[\frac{\partial L}{\partial \dot{q}} \delta q \right]_{t_1}^{t_2} - \int_{t_1}^{t_2} \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) \delta q dt \quad (1-6)$$

となる。運動の最初と最後では与えられた位置にいるので、 $\delta q(t_1)=\delta q(t_2)=0$ であり、右辺第一項目は 0 になる。(1-6)を(1-5)に代入すると、

$$\int_{t_1}^{t_2} \left(\frac{\partial L}{\partial q} - \frac{d}{dt} \frac{\partial L}{\partial \dot{q}} \right) \delta q dt = 0 \quad (1-7)$$

となる。任意の変位 $\delta q(t)$ に対して(1-7)が成立する、つまり、 $q(t)$ が作用 S を最小にする条件は、

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0 \quad (1-8)$$

と得られる。この微分方程式(1-8)がラグランジュ方程式である。

1-3 ラグランジアン (Lagrangian)

ラグランジアン L が決まれば、それを微分することで運動方程式が得られる。では、ラグランジアンはどのように求めるのだろうか？

まず、何の力も働いていない様な3次元空間で拘束されずに運動している一つの質点を考えてみよう。基準となる場所や時間を変えても、運動の性質は変わらない。このような系を表わすラグランジアンは、位置と時間に依存しないので、 $L(\dot{q})$ である。さらに、速度の向きが逆でも、運動の性質は変わらないから、 $L(\dot{q}^2)$ と書き表せる。詳しい説明は参考文献¹⁾を参照してほしいが、結局、このときのラグランジアンは、定数 $\times \dot{q}^2$ となる。この定数として、一般には質量 m を用いて $m/2$ を選ぶ。複数の質点が独立に運動している場合は、これを足し合わせ、

$$L = \sum_i \frac{1}{2} m_i \dot{q}_i^2 \quad (1-9)$$

となる。すなわち、ラグランジアンは運動エネルギーの総和と等しい。

さて、つぎに、例えば、質点間に働く引力のように、場に相互作用が存在している場合を考えよう。このとき、ラグランジアンはこの相互作用の分だけ修正される。

$$L = \sum_i \frac{1}{2} m_i \dot{q}_i^2 - U(\mathbf{q}) \quad (1-10)$$

修正分の U は、質点の位置だけに依存するものであり、ポテンシャルエネルギーと呼ばれる。ポテンシャルエネルギーには、位置エネルギーやバネの弾性エネルギーなどがある。

これを一般的に書き直すと、力学系のラグランジアンは、運動エネルギー T とポテンシャル U を用いて、

$$L = T - U \quad (1-11)$$

となる。

さて、(1-11)式において、ラグランジアンは運動エネルギーとポテンシャルの差になっており、力学的エネルギーの総和とは異なる。なぜ、ラグランジアンはエネルギーの引き算なのか？と、不思議に思うかもしれない。そこで、上記の定義から、力学的エネルギー保存則を導出することで、この式が正しいことを確認してみよう。

ラグランジアンの全微分は、 \mathbf{q} と $\dot{\mathbf{q}}$ の偏微分を用いて、

$$\frac{dL}{dt} = \frac{\partial L}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} + \frac{\partial L}{\partial \dot{\mathbf{q}}} \frac{d\dot{\mathbf{q}}}{dt} \quad (1-12)$$

と書き表せる²⁾。右辺第一項目に(1-8)を代入すると、

$$\begin{aligned} \frac{dL}{dt} &= \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) \frac{d\mathbf{q}}{dt} + \frac{\partial L}{\partial \dot{\mathbf{q}}} \frac{d\dot{\mathbf{q}}}{dt} \\ &= \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \frac{d\mathbf{q}}{dt} \right) \end{aligned} \quad (1-13)$$

となる。さらに、(1-13)の両辺をまとめると、

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \frac{d\mathbf{q}}{dt} - L \right) = 0 \quad (1-14)$$

である。これは、

$$\frac{\partial L}{\partial \dot{\mathbf{q}}} \frac{d\mathbf{q}}{dt} - L = \text{const.} \quad (1-15)$$

を表わしている。ところで、ラグランジアンが(1-10)で表わされる場合、

$$\frac{\partial L}{\partial \dot{\mathbf{q}}} \frac{d\mathbf{q}}{dt} = 2 \sum_i \left(\frac{1}{2} m_i \dot{q}_i \frac{dq_i}{dt} \right) (= 2T) \quad (1-16)$$

である。(1-11)と(1-16)を(1-15)に代入すると、

$$T + U = \text{const.} \quad (1-17)$$

となり、力学的エネルギーの保存則が導かれた。

1-4 ラグランジュの運動方程式

前節までは、外部とのエネルギーのやりとりがない保存系(孤立系)のラグランジュの運動方程式について説明した。摩擦などによって系のエネルギーが散逸する場合や、外力が加えられる場合に対しては、より一般的に、ラグランジュの運動方程式が、

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} + \frac{\partial D}{\partial \dot{\mathbf{q}}} = \mathbf{F} \quad (1-18)$$

で与えられる³⁾。ここで、 D は摩擦などの散逸エネルギーである。減衰係数 c_i のダンパの場合は、

$$D = \sum_i \frac{1}{2} c_i \dot{q}_i^2 \quad (1-19)$$

で計算される。 \mathbf{F} は一般化座標の方向に働く一般化力(外力やトルク)である。

(1-18)を一般化座標 $\mathbf{q}=(q_1, q_2, \dots, q_n)$ の各要素ごとに計算すると、加速度と位置・速度や力との関係を表わす運動方程式が得られる。

まとめ：ラグランジュの運動方程式

一般化座標を $\mathbf{q}=(q_1, q_2, \dots, q_n)$ とする。系の運動エネルギーの総和が T 、ポテンシャルエネルギーの総和が U 、散逸エネルギーの総和が D 、一般化力が $\mathbf{F}=(F_1, F_2, \dots, F_n)$ であるとき、運動方程式は、

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = F_i \quad (i=1, 2, \dots, n)$$

で得られる。ただし、 $L=T-U$ である。

1-5 質点の運動方程式

質量 m の物体に外力 F が加えられたときの鉛直方向の運動を考えよう．重力加速度は g とし、前節までの説明にしたがって、ラグランジュの運動方程式を求める．

地表面から測った質点 m の位置を x とする．運動エネルギーは、

$$L = \frac{1}{2} m \dot{x}^2 \quad (1-20)$$

である．ポテンシャルエネルギーは、

$$U = mgx \quad (1-21)$$

である．したがって、ラグランジアンは、

$$L = \frac{1}{2} m \dot{x}^2 - mgx \quad (1-22)$$

となり、

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = m \ddot{x} \quad (1-23)$$

$$\frac{\partial L}{\partial x} = -mg \quad (1-24)$$

より、ラグランジュの運動方程式は、

$$m \ddot{x} + mg + 0 = F \quad (1-25)$$

と得られる．なお、このとき、ニュートンの運動方程式(1-1)は、

$$m \ddot{x} = -mg + F \quad (1-26)$$

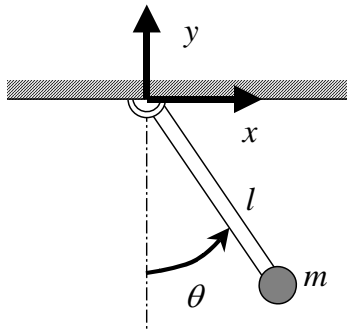
であり、明らかにラグランジアンを用いて得られた運動方程式(1-25)と同じである．

参考文献：

1. ランダウリフシッツ：力学，東京図書(1974)
3. 小寺，長谷川：工学系学生のための常微分方程式，森北出版(1996)
2. 増淵，川田：システムのモデリングと非線形制御，コロナ社(1996)

演習1 「運動方程式をたてる」

1. 運動方程式



図のように、長さ l で天井に取り付けられた質量 m の振り子がある。振り子は、平面内で左右に触れるとする。振り子の運動を表わす運動方程式を、下記の方法で求めなさい。なお、棒の質量は無視してよい。

1. ニュートンの運動方程式
2. ラグランジュの運動方程式

なお、図中の x, y, θ は参考として書いた例であり、座標系の取り方は自由とする。

2. Mathematica で微分をしてみましょう。

Mathematica のコマンド

微分 : $D[f(t), t]$... $f(t)$ を t で微分する
 関数 : $x[t]$ x は t の関数である
 関数の微分 : $x'[t]$ x の t による一階微分

2-1. 次の関数を t で微分してみよう。

$$t^2, e^t, \log_e t, \frac{1}{t+1}, \sin(t), x(t), \frac{1}{2} m v(t)^2 \quad (m \text{ は定数})$$

2-2. $L = \frac{1}{2} m (\dot{x} + \dot{y})^2$ を \dot{x}, \dot{y} でそれぞれ偏微分してみよう。さらにこの答えを時間で微分してみよう。

システムの表現

2. 古典制御におけるシステムの表現（伝達関数）

古典制御における制御対象は、1入力1出力(Single Input Single Output, SISO)の時不変で連続な線形システムである。時不変なシステムとは、質量や粘性係数などのシステムパラメータが時間によって変化しないシステムである。一方、連続システムであるとは、入力や出力といった信号が連続時間的に変化することを表す。線形システムでは、入力が2倍になれば、出力も2倍になるという線形関係が成立する。

連続時間システムの状態や応答の変化の様子は、多くの場合、微分方程式で表すことができる。

例：ばねダンパ系

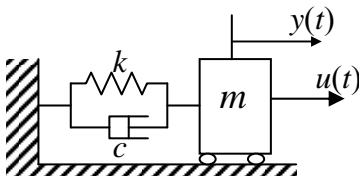


図 2-1 ばねダンパ系

質量が $m[\text{kg}]$ の物体が壁にばね定数 k のばね、粘性係数 c のダッシュポットで取り付けられている。このとき、物体に力 $u(t)$ を加えると、物体のつり合いからの位置 $y(t)$ は、運動方程式、

$$m\ddot{y}(t) + c\dot{y}(t) + ky(t) = u(t) \quad (2-1)$$

にしたがって変化する。

2-1 ラプラス変換 (Laplace transformation)

システムの状態や応答の変化を表す微分方程式を解けば、応答や状態の様子を知ることができる。複雑なシステムを表すモデルは、高階の微分方程式になったり、連立微分方程式になったりする。一般に、これらの微分方程式は解くのが大変である。そこで、ラプラス変換を用いる方法がある。ラプラス変換は、微分方程式を代数方程式に変換するため、解析やコントローラ設計が容易になるという利点もある。

ラプラス変換は、時間関数 $f(t)$ を複素関数 $F(s)$ に変換する。その定義は、

$$F(s) = \int_0^{\infty} f(t)e^{-st} dt \quad (= \mathcal{L}(f(t))) \quad (2-2)$$

である。変換記号として、 $\mathcal{L}(\cdot)$ を用いる。この定義に従うと、たとえば、関数の微分は、

$$\mathcal{L}\left(\frac{df(t)}{dt}\right) = sF(s) - f(0) \quad (2-3)$$

となる。

2-2 伝達関数 (transfer function)

ラプラス変換を用いて運動方程式(2-1)を変換してみよう．簡単な例として， $\dot{x}(0) = x(0) = 0$ として考えよう． $Y(s) = \mathcal{L}(y(t))$ ， $U(s) = \mathcal{L}(u(t))$ とすると，

$$ms^2Y(s) + csY(s) + kY(s) = U(s) \quad (2-4)$$

となる．微分演算子がなくなり， $U(s)$ と $Y(s)$ の関係が代数方程式で表されていることがわかる．

システムの入力と出力をラプラス変換した関数の比 $Y(s)/U(s)$ をシステムの伝達関数と呼ぶ．ただし，伝達関数では，システムの信号の初期値をすべて 0 で考える．

たとえば，システム(2-1)の伝達関数は，(2-4)式より，

$$\frac{Y(s)}{U(s)} = \frac{1}{(ms^2 + cs + k)} \quad (2-5)$$

となる．

2-3 ブロック線図 (block diagram)

システムは，複数の構成要素(サブシステム)で構成されていることが多い．このとき，サブシステム間での信号の伝達を表すのに，ブロック線図がよく用いられる．

ブロック線図では，システムは四角で囲まれ，信号は矢印で表される．

例：フィードバック制御系

左図で表されるシステムは，出力 $Y(s)$ が入力信号に影響していることから，フィードバック制御系と呼ばれる．

このシステムにおいて，外部入力 $R(s)$ から出力 $Y(s)$ への伝達関数はどうなるだろうか？

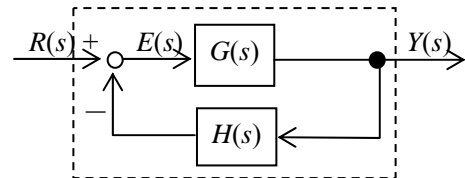


図 2-2 フィードバック制御系

ブロック線図から，次の関係が読み取れる．

$$Y(s) = G(s)E(s) \quad (2-6)$$

$$E(s) = R(s) - H(s)Y(s) \quad (2-7)$$

この代数方程式を連立させて $E(s)$ を消去すると，

$$\frac{Y(s)}{R(s)} = \frac{G(s)}{1 + H(s)G(s)} \quad (2-8)$$

という伝達関数が得られる．

演習2 「ラプラス変換，逆ラプラス変換をおこなう」（Mathematica の場合）

1. ラプラス変換してみましょう。

ステップ関数(3, a =一定値), ランプ関数 ($t, 3t$),

三角関数 ($\sin 2t, \cos wt$), 指数関数 (e^t, e^{-at}), その他 ($te^{-at}, e^{-at} \sin 2t$)

注: e^{-at} は, Exp[-at]と入力する. 掛け算は, * を使うか, 空白をあければよい.

2. 逆ラプラス変換してみましょう。

1, $\frac{1}{s}$, $\frac{K}{s+T}$, $\frac{w}{s+w^2}$, $\frac{1}{s^2+s+1}$ など

注: s^2 は s^2 と入力する. また, Mathematica では大文字と小文字は区別される.

3. 現代制御におけるシステムの表現（状態空間表現）

3-1 伝達関数から状態方程式へ

古典制御で扱うシステムの入力は1つであり，出力も1つであった．また，システムの入出力比を表す伝達関数にもとづいて解析・設計を行うため，システム内部の状態は考慮していない．

しかし，宇宙開発を始めとする技術の発展の中で，複数の入力や出力を持つ複雑なシステムの制御問題を考える必要が生じた．現代制御論は，多入出力システムを取り扱うのに適した理論である．現代制御論では，システムのモデルを状態方程式(*state equation*)で表す．状態方程式とは，システムの状態を定義し，それぞれの状態の時間変化を表す一階の微分方程式を連立させたモデルである．状態方程式を用いることで，システムの状態を考えた解析・設計を行うことができる．

例：伝達関数から状態方程式へ

伝達関数が，

$$\frac{Y(s)}{U(s)} = \frac{1}{(ms^2 + cs + k)} \quad (2-5)$$

で表される2次系を考えよう．(2-5)式を逆ラプラス変換すれば，

$$\ddot{y}(t) + a\dot{y}(t) + by(t) = u(t) \quad (2-1)$$

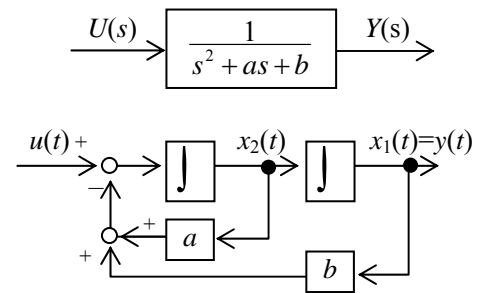


図 3-1 伝達関数と状態空間表現

という2階の微分方程式に戻る．ここで，つぎのように状態ベクトル \mathbf{x} を定義しよう．なお，以下では， $x(t)$ が時間関数であることが明らかな場合は，単に x と表記する．

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y \\ \dot{y} \end{pmatrix} \quad (3-1)$$

このとき，(2-1)式はつぎのように書き直すことができる．

$$\dot{x}_2 + \frac{c}{m}x_2 + \frac{k}{m}x_1 = \frac{1}{m}u \quad (3-2)$$

また，明らかに，

$$\dot{x}_1 = x_2 \quad (3-3)$$

であるから，(3-2)(3-3)式をあわせて行列表現を用いると，

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{x} \end{aligned} \quad (3-4)$$

が得られる．(3-4)式を状態空間表現という．一般に，状態方程式は係数行列に A, B, C を用いて，

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x}\end{aligned}\tag{3-5}$$

と書き表す。状態数が n ，入力数が m ，出力数が p である場合，係数行列のサイズは， \mathbf{A} は $n \times n$ ， \mathbf{B} は $n \times m$ ， \mathbf{C} は $p \times n$ となる。

3-2 状態方程式と伝達関数行列

今度は，逆に状態方程式(3-5)から，伝達関数行列を求めてみよう。スカラー関数のときと同様に，(3-7)式の両辺をラプラス変換すると，

$$\begin{aligned}s\mathbf{X}(s) &= \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s) \\ \mathbf{Y}(s) &= \mathbf{C}\mathbf{X}(s)\end{aligned}\tag{3-6}$$

となる。ただし， $\mathbf{X}(s)$ は， $\mathbf{x}(t)$ の各要素をラプラス変換したベクトルである。 \mathbf{A} が行列であることに注意して，(3-6)式をまとめると，

$$\mathbf{Y}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)\tag{3-7}$$

となる。ここで， \mathbf{I} は， \mathbf{A} と同じサイズの単位行列である。 $\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$ は伝達関数行列と呼ばれる。

なお，2-1 節の例で示したように，伝達関数行列から状態方程式を導くことを，実現(*realization*)という。一つの伝達関数を実現するための状態の選び方は一意ではなく，得られる状態方程式は無数に存在する。

演習3 「状態方程式」(Mathematica または MATLAB)

1. 次のシステムの状態方程式を求めなさい.

$$1) Y(s) = \frac{1}{s^3 + s^2 + s + 2} U(s)$$

$$2) Y(s) = \frac{s+1}{s^2 + s + 1} U(s)$$

2. 1で計算した状態方程式から, $G(s) = C(sI - A)^{-1}B$ を計算して, 元の伝達関数が求められることを確かめなさい.

3. 行列の指数関数について, 次の計算を試みましょう. ただし, $A = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}$ とします.

1) e^{At} を計算しましょう. また, $t=0$ とすると, いくつになりますか?

2) e^{At} と $I + \frac{At}{1!} + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots + \frac{(At)^n}{n!} + \dots$ を比較しなさい. ただし, $t=0.1$ とします.

誤差が5%以下になるのは, n がいくつのときでしょうか?

3) $(e^{At})^{-1}$ と e^{-At} が等しいことを確かめなさい.

4) $\frac{d}{dt} e^{At}$ と Ae^{At} が等しいことを確かめなさい.

5) e^{At} と $(sI - A)^{-1}$ の逆ラプラス変換した関数が等しいことを確かめなさい.

システムの時間応答と安定性

4. システムの安定性(stability)

一般に、制御系に要求される最も基本的な性質は安定なことである。システムが不安定であると、状態や出力の値が発散してしまう。これは、振動が徐々に大きくなっていく現象や可動範囲の超過などを表し、システムの故障や事故を引き起こす。

「安定」の直感的なイメージは、外部からの入力が無くなったときに、システムの状態がある一定値(=平衡点)に落ち着くということである。たとえば、やじろべえを指で弾くと振動するが、しばらく時間が経つと再び最初のバランスをとった姿勢に戻る。これが安定である。

安定性の厳密な定義は、複数存在する。入出力安定性、内部安定性、リアプノフ安定性などである。ただし、古典制御で対象としている1入力1出力の線形システムでは、これらの定義はいずれも等価な条件となり、以下で述べる極の値に帰着する。

例：一次系のインパルス応答

図4-1で表される一次系のインパルス応答を調べよう。

インパルス入力をラプラス変換した $U(s)=1$ を代入してから、逆ラプラス変換すると、出力(応答)は、 $y(t) = be^{-at}$ となる。この定常値は、

$$\lim_{t \rightarrow \infty} y(t) = 0 \quad , a > 0$$

$$\lim_{t \rightarrow \infty} y(t) = \infty \quad , a < 0$$

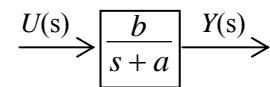


図4-1 一次系

である。 a の符号が負のとき、出力が無限大に発散することがわかる。つまり、伝達関数の分母の係数 a の値によって、システムの安定性が決まる。

4-1 特性方程式(characteristic equation)と極 (pole)

実際に時間応答を計算してシステムの安定性を確認するのは手間がかかる。前述の例が示すように、システムの安定性は伝達関数の分母によって決まる。これに注目すると、時間応答を計算することなく、安定性を判定できる。

伝達関数の分母をシステムの特性多項式と呼ぶ。そして、特性多項式=0とした s の方程式を、システムの特性方程式と呼ぶ。この特性方程式の解をシステムの極と呼ぶ。極はシステムの安定性や挙動を知るために重要な値であり、伝達関数の次数と同じだけ存在する。

極を用いると、システムが安定であるための必要十分条件は、

$$\text{システムが安定} \Leftrightarrow \text{全ての極の実部が負}$$

と表される。特に実部が0のときは、安定限界と呼ぶこともある。

例：一次系の安定性

前節の例において，伝達関数 $G(s) = \frac{b}{s+a}$ の特性方程式は $s+a=0$ である．よって，極は $s=-a$ である． $a>0$ ならば，極は負であり，システムは安定となる．これは，インパルス応答を調べた結果と一致している．

4-2 状態空間表現における安定性

3章で示したように，一般的な状態方程式，

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \quad (3-5)$$

の伝達関数行列は，

$$Y(s) = C(sI - A)^{-1}BU(s) \quad (3-7)$$

である．この伝達関数行列の特性方程式は，

$$|sI - A| = 0 \quad (4-1)$$

である．したがって，この特性方程式の解，すなわちシステムの極のすべての実部が負であれば，システム(3-5)は安定となる．

ところで，(4-1)の解は，行列 A の固有値と等しい．このことから，システム(3-5)が安定であるための必要十分条件は， A の固有値の全ての実部が負である，と言い換えることができる．

4-3 安定判別法(stability criterion)

システムの安定性を調べるには，全ての極の実部の符号を調べればよい．しかし，5次以上の方程式の一般解が存在しないことなどから，特性方程式の次数があがると，極を求めることは難しくなる．そこで，方程式を解くことなく，安定性を調べる方法として，ラウスの安定判別法やフルビッツの判別法がある．これらは制御のテキストなどには必ず掲載されている手法であるので，ここでは省略する．

一方，近年では，数式処理ソフトウェアの発達により，方程式の数値解を求めることが容易になった．したがって，高次のシステムでも直接，極を計算して安定性を調べる事が可能である．

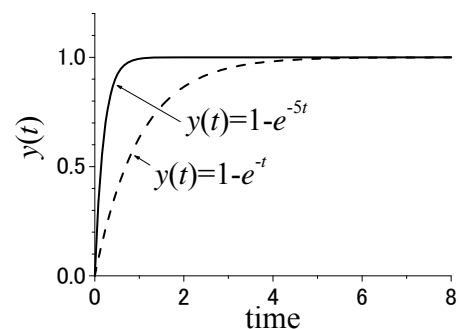


図 5-1 一次系のステップ応答

演習4 「システムの極と時間応答をみる」（Mathematica の場合）

有理関数の分母

Denominator[$a(s)/b(s)$]. . . 有理関数の分母 $b(s)$ を取り出す

方程式を解く

Solve[$b(s)=0, s$]. . . s の方程式 $b(s)=0$ を s について解く

NSolve[$b(s)=0, s$]. . . s の方程式 $b(s)=0$ を s について数値的に解く

FindRoot[$b(s)=0, \{s, s_0\}$]. . . s の方程式 $b(s)=0$ について $s=s_0$ の近傍から解を探索する
(一つの解を見つけるだけ. 一度に全ての解は得られない)

実部と虚部

Re[x]. . . 変数 x の実数部分を取り出す

Im[x]. . . 変数 x の虚数部分を取り出す

行列

Det[A]. . . A の行列式を計算する

Inverse[A]. . . A の逆行列を計算する

IdentityMatrix[n]. . . $n \times n$ の単位行列

1. 極を求めてみましょう.

$$\frac{1}{s+2}, \frac{1}{s^2+2}, \frac{1}{s^2+s+2}, \frac{1}{s^3+s^2+s+2}, \frac{1}{s^4+s^3+s^2+s+2}, \frac{1}{s^5+s^4+s^3+s^2+s+2},$$

$$\frac{1}{s+a}, \frac{1}{s^2+as+b}$$

2. 安定判別

特性方程式が $s^3+s^2+s-1=0$ の場合, 安定か不安定か? (極の実部の符号を調べなさい)

特性方程式が $s^5+s^4+s^3+s^2+s+1=0$ の場合はどうか?

3. 極とステップ応答

二次系 $\frac{1}{s^2+as+b}$ の極は, a と b の値によって, 異なる2つの実数解, 重解, 共役複素解のいずれかになる. 定数 a, b をつぎのように変えて, それぞれのステップ応答のグラフを描いて確認しなさい.

実数解 : $a=1.5, b=0.5$

重解 : $a=2, b=1$

共役複素解 : $a=1, b=1$

4. $\dot{x} = Ax$ というシステムの係数行列が,

$$A = \begin{pmatrix} 0 & 1 & 3 & -1 \\ 0 & -1 & -2 & 2 \\ -1 & 0 & -2 & -1 \\ -1 & 0 & -3 & 0 \end{pmatrix}$$

であるとき, 極を求めなさい. ($|sI - A| = 0$ を解いて求める)

5. システムの極と時間応答

極の実部の符号によって安定性が決まるが、極の絶対値によって、システムの過渡応答が変わる。まず、例を見てみよう。

例：一次系の過渡応答

伝達関数 $G(s) = \frac{1}{s+a}$ の極は、 $s=-a$ であり、ステップ応答は、 $y(t) = 1 - e^{-at}$ である。 $a=1, 5$ のときのステップ応答のグラフを図 5-1 に示す。 $a=5$ 、すなわち、極が $s=-5$ であるときのほうが、極が -1 のときよりも速く定常値 1 に収束する。

2 次以上のシステムでは、極に複素数を含むことがある。極を複素平面にプロットしたときの位置と、過渡応答の特徴との関係をまとめておこう。

まず、複素平面の左半平面にプロットされる極は、実部が負であるので安定な極である。逆に右半平面は不安定である。極が安定である場合、上記の例で示したように、複素平面の左側にあるほど（ $-\infty$ に近づくほど）、収束は速い。また、複素数の極は、必ず共役複素数として現れる。共役複素数は、実軸対称にプロットされる。実軸から離れるほど、過渡応答は振動的になる。安定限界である純虚数の極は、虚軸上にプロットされる。このときはステップ応答はある振幅で振動を続ける。

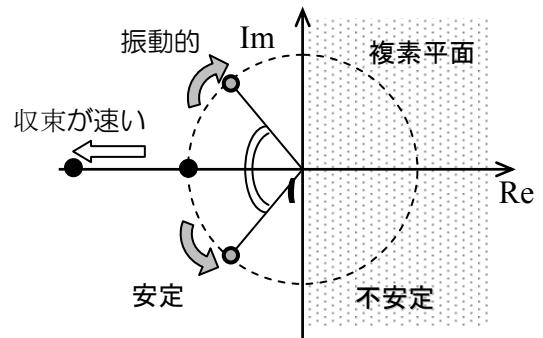


図 5-2 複素平面での極の配置

さらに高次のシステムとして、状態方程式(3-5)で表わされる一般的な線形システムを考えよう。状態方程式は、状態ベクトル \mathbf{x} についての一階の微分方程式であるので、状態方程式(3-5)の解 $\mathbf{x}(t)$ はつぎのように得られる。

$$\mathbf{x}(t) = e^{At} \mathbf{x}(0) + \int_0^t e^{A(t-\tau)} \mathbf{B}u(\tau) d\tau \quad (5-1)$$

ただし、 $\mathbf{x}(0)$ は状態の初期値であり、 e^{At} はつぎの式で定義される指数関数行列である。

$$e^{At} = I + \frac{At}{1!} + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots + \frac{(At)^n}{n!} + \dots \quad (5-2)$$

このシステムの初期値応答を見てみよう。状態方程式(3-5)において、初期値 $\mathbf{x}(0)$ 、入力 $\mathbf{u}(t)=0$ としてラプラス変換すると、

$$\begin{aligned} s\mathbf{X}(s) - \mathbf{x}(0) &= A\mathbf{X}(s) \\ \mathbf{X}(s) &= (sI - A)^{-1} \mathbf{x}(0) \end{aligned} \quad (5-3)$$

となる。ここで、システムの n 個の極を $p_1 \cdots p_n$ とおく。すなわち、 $|sI - A| = (s - p_1)(s - p_2) \cdots (s - p_n)$ である。このとき、 $(sI - A)^{-1}$ の全ての要素は、分子の係数がいかなる値であっても、

$$\frac{a_{ij}}{(s-p_1)(s-p_2)\cdots(s-p_n)} = \frac{c_{ij1}}{(s-p_1)} + \frac{c_{ij2}}{(s-p_2)} + \cdots + \frac{c_{ijn}}{(s-p_n)} \quad (5-4)$$

と部分分数展開できる。(5-3)式の右辺は、 $(sI-A)^{-1}$ に係数ベクトル $\mathbf{x}(0)$ をかけたものなので、 $\mathbf{X}(s)$ は(5-4)の線形和となる。したがって、 $\mathbf{X}(s)$ を逆ラプラス変換して得られる $\mathbf{x}(t)$ は、(5-4)を逆ラプラス変換して得られる $e^{p_1 t}$ 、 $e^{p_2 t}$ 、 \cdots 、 $e^{p_n t}$ の線形和になる。この $e^{p_1 t}$ 、 $e^{p_2 t}$ 、 \cdots 、 $e^{p_n t}$ をシステム固有のモード(*mode*)と呼ぶ。モードは、極の値で決まることから、システムの応答は極の値に大きく左右されることがわかる。個々のモードのふるまいは、図5-2で示した傾向に従う。たとえば、一つ(一対)でも原点に近い極があると、そのモードのためにシステムの収束が遅くなる。

演習5 「システムの時間応答をみる」（MATLAB/Simulink の場合）

1. 演習4の3「極とステップ応答」を MATLAB を用いてグラフを描いてみましょう.

制御系設計

6. 古典制御と周波数領域での解析

6-1 フィードバック制御による安定化

全ての極の実部が負であれば、システムは安定である。不安定な極を持つシステムに対して、何らかの制御を行って極を安定な値に変更することを安定化(*stabilization*)という。フィードバック制御による安定化制御の例をみてみよう。

例 6-1：一次系のフィードバック制御

図 6-1 の上図で表される一次系、

$$\frac{Y(s)}{R(s)} = \frac{1}{s-0.5} \quad (6-1)$$

の極は+0.5 であり、不安定である。このシステムに対して、図 6-1 の下図のようにフィードバック制御をほどこす。すると、外部入力 $R(s)$ から出力 $Y(s)$ までの伝達関数は、

$$\frac{Y(s)}{R(s)} = \frac{1}{s-0.5+K} \quad (6-2)$$

となる。フィードバック系の極は $0.5-K$ であるから、 $K>0.5$ とすれば制御系は安定となる。

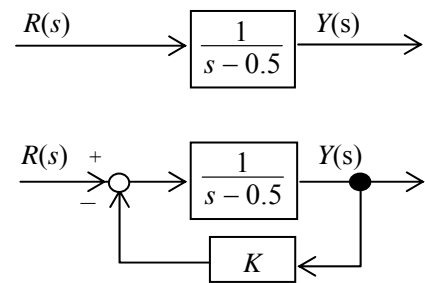


図 6-1 フィードバック制御系 1

6-2 定常応答の改善

制御の目的はいくつかあるが、安定化のつぎに与えられる主な要求の一つに、目標値追従がある。システムの入力 $R(s)$ を目標値 $R(s)$ に一致させるための制御系を目標値追従制御系、もしくはサーボ系(*servo system*)という。このとき、出力と目標値との差を偏差(*error*)と呼ぶ。一般に、制御対象が持つ動特性のため、制御開始時から偏差を完全に 0 にすることは難しい。そこで、まず、十分時間が経ったときの偏差 (= 定常偏差, *steady-state error*) を 0 にすることが第一の目標となる。

例 6-2：一次系のフィードバック制御の定常偏差

図 6-1 の例において、目標入力 $R(s)$ が大きさ r のステップ関数であるとき、定常偏差は最終値の定理より、

$$\begin{aligned}
 \lim_{t \rightarrow \infty} e(t) &= \lim_{t \rightarrow \infty} (r(t) - y(t)) \\
 &= \lim_{s \rightarrow 0} s(R(s) - Y(s)) \\
 &= \lim_{s \rightarrow 0} s \left(\frac{s-1.5+K}{s-0.5+K} \right) R(s) \\
 &= \frac{-1.5+K}{-0.5+K} r
 \end{aligned} \tag{6-3}$$

となる。したがって、 $K=1.5$ とすれば、定常偏差を 0 にすることができる。

6-3 過渡応答の改善

多くの場合、応答はできるだけ早く定常状態に落ち着くことが望ましい。前述の例では、 K を大きくすると、極が $-\infty$ に近づくので、応答を速くすることができる。しかし、応答を速くするためにゲインを大きくしようとすると、定常偏差が 0 にできない。

そこで、PID 制御を行うことを考えよう。PID 制御(*Proportional, Integral and Derivative control*)とは、偏差の比例、積分、微分信号をフィードバックに用いる制御である。すなわち、図

6-2 において、コントローラ $C(s)$ に、

$$\text{比例制御 } C(s) = K_P \tag{6-4}$$

$$\text{積分制御 } C(s) = K_I / s \tag{6-5}$$

$$\text{微分制御 } C(s) = K_D s, \tag{6-6}$$

もしくはこれらの組み合わせを用いる。

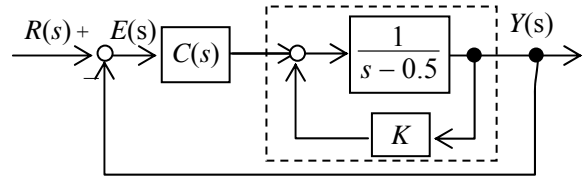


図 6-2 フィードバック制御系 2

例 6-3：一次系のフィードバック制御の PID 制御

図 6-2 のシステムで、 $K=1.5$ とする。目標入力 $R(s)$ が大きさ r のステップ関数であるとき、コントローラ $C(s)$ として、比例、積分、微分を用いると、それぞれ、定常偏差 $E(s)$ は、

$$\text{比例制御 } C(s) = K_P$$

$$\begin{aligned}
 \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{1}{1 + K_P \frac{1}{s+1}} \frac{r}{s} \\
 &= \lim_{s \rightarrow 0} \frac{s+1}{s+1+K_P} r \\
 &= \frac{1}{1+K_P} r
 \end{aligned} \tag{6-7}$$

$$\text{積分制御 } C(s) = K_I / s$$

$$\begin{aligned}
 \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{1}{1 + \frac{K_I}{s} \frac{1}{s+1}} \frac{r}{s} \\
 &= \lim_{s \rightarrow 0} \frac{s(s+1)}{s^2 + s + K_I} r \\
 &= 0
 \end{aligned} \tag{6-8}$$

微分制御 $C(s) = K_D s$

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{1}{1 + K_D s} \frac{1}{s} r \\ &= \lim_{s \rightarrow 0} \frac{s+1}{(1 + K_D)s+1} r \\ &= r \end{aligned} \tag{6-9}$$

となる。積分制御を行うことでゲインの値にかかわらず、定常偏差が 0 にできる。また、一般に、比例や微分要素を用いることで、速応性(目標値に速く到達する性質)が向上する。それぞれのゲインの調整法は、極配置や経験的な Ziegler-Nichols の方法などがある。

6-4 周波数応答とボード線図

システムに正弦波状入力を加わると、入力と同じ角周波数の正弦波状の出力が生じる。正弦波状入力に対するシステムの定常応答を周波数応答(*frequency response*)と言う。周波数応答は、入出力信号の振幅比(*gain*)と位相ずれ(*phase*)で表される。このゲインや位相は、入力の角周波数 ω の関数になる。これらは、伝達関数 $G(s)$ の s に $j\omega$ を代入した周波数伝達関数 $G(j\omega)$ を用いて、

ゲイン： $20 \log_{10} |G(j\omega)|$ [dB]

位相： $\angle G(j\omega)$ [°]

で計算できる。横軸に角周波数 ω の対数を取り、縦軸にゲイン、位相をプロットしたグラフは、ボード線図(*Bode diagram, Bode plot*)と呼ばれ、周波数領域での解析に用いられる。

6-5 位相余裕, ゲイン余裕

図 6-3 下図のような閉ループ系(*closed loop system*)の安定性は、図 6-3 上図の開ループ系(*open loop system*)のボード線図から判断できることが知られている。なお、上図は、下図の一巡伝達関数とも呼ばれる。

ボード線図において、ゲインが 0[dB]になる周波数を、ゲイン交点角周波数と呼ぶ。この角周波数のときに、位相が -180° より何度上にあるかを位相余裕(*phase margin*)という(図 6-4)。一方、位相が -180° になる周波数を位相交点角周波数と呼ぶ。この角周波数のときに、ゲインが 0dB よりどれだけ下にあるかをゲイン余裕(*gain margin*)という。位相余裕、ゲイン余裕が正であることが、閉ループ系が安定であるための必要十分条件である。位相余裕、ゲイン余裕が大きいほど、モデル化誤差などがあってもシステムの安定性が保たれることになる。誤差や外乱があっても安定性が保つ性質は、ロバスト性(*robustness*)と呼ばれ、制御系に要求される性能の一つである。ただし、余裕を持たせすぎると、速応性など、他の仕様を損なうことがある。

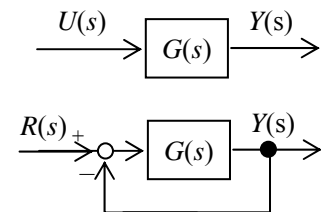


図 6-3 開ループ系と閉ループ

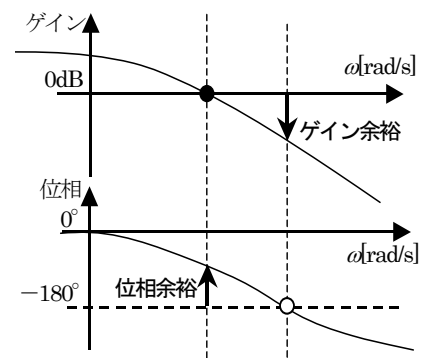


図 6-4 ゲイン余裕, 位相余裕

6-6 位相進み補償，位相遅れ補償

図 6-5 のフィードバック制御系のコントローラとして，

$$C(s) = K \frac{T_N s + 1}{T_D s + 1} \quad (6-10)$$

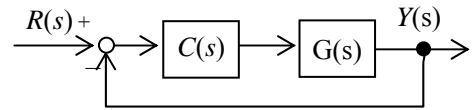


図 6-5 フィードバック制御系

を考えよう．このコントローラは，分母と分子の係数の関係によって，

$T_N > T_D \dots$ 位相進み補償回路

$T_N < T_D \dots$ 位相遅れ補償回路

と呼ばれている．このコントローラを用いることで，ゲイン余裕，位相余裕が改善できる．

演習6 古典制御（MATLAB/Simulink）

1. 例 6-1 において，入力 $r(t)$ を大きさ 1 のステップ関数として，ゲイン K の値によって，応答 $y(t)$ がどう変わるかを調べてみましょう。
また，偏差 $r(t) - y(t)$ のグラフも描いて，定常偏差を比べてみましょう。
2. 例 6-3 において，ゲイン K_p , K_I , K_D を変化させて，応答の違いを比較しましょう。
3. 図 6-2 のコントローラを

$$C(s) = K_p + K_I \frac{1}{s} + P_D s$$

としたとき，入力 $R(s)$ から $Y(s)$ までの伝達関数を計算しなさい。ただし， $K=1.5$ とします。
また， K_p , K_I , K_D を変化させて，定常偏差を 0 にして，できるだけ速く定常値に収束するように，ゲインを調整してみましょう。そのときのシステムの極も求めて，応答と極の関係を考察しましょう。

ボード線図の書き方（control system toolbox の利用）：
 コマンドラインで行います。伝達関数を作る関数を用います。
 $g1 = tf([a_1, a_0], [b_1, b_0])$... 伝達関数 $g1 = (a_1s + a_0) / (b_1s + b_0)$ を作る
 $bodeplot(g1)$... ボード線図を描く
 $margin(g1)$... ゲイン余裕，位相余裕，ゲイン角周波数，位相角周波数を求め，ボード線図にプロットする
 $pole(g1)$... 伝達関数 $g1$ の極を求める
m ファイルにまとめる：
 コマンドラインで複数の命令を繰り返し行う場合は，**m** ファイルを作成すると便利です。
 「ファイル」→「新規作成」→「M-ファイル」とすると，新規画面が出ます。
 コマンドを書き込んで，「デバッグ」→「実行」で実行します。
 結果は，コマンドラインに表示されます。

4. ボード線図を描いてみましょう。

$$\frac{1}{s}, s, \frac{1}{s+1}, (s+1), \frac{1}{s^2+s+1}, \frac{s+1}{s^2+s+1}, \frac{1}{s^3+s^2+s+2}$$

5. T_N , T_D の値をそれぞれ，0.1, 0.5, 1 に変えて，ボード線図を描き，位相進み，位相遅れの意味を考えてみましょう。

$$C(s) = \frac{T_N s + 1}{T_D s + 1}$$

$T_D \backslash T_N$	0.1	0.5	1.0
0.1			
0.5			
1.0			

7. 現代制御における状態フィードバック制御

7-1 可制御性・可観測性

現代制御では、内部状態を考えることから、可制御性・可観測性といった概念が登場する。可制御性(*controllability*)は、全ての状態を制御できるかどうかを表す性質である。「任意の初期値 $\mathbf{x}(0)$ に対し、有限時間 T で、 $\mathbf{x}(T)=0$ を満たす入力 $\mathbf{u}(t)$ が存在する」とき、システムは可制御であるという。一方、可観測性(*observability*)は、入力と出力から、全ての状態が特定できるかを表す性質である。「ある有限時間 T の間の入力 $\mathbf{u}(t)$ と出力 $\mathbf{y}(t)$ から、初期状態 $\mathbf{x}(0)$ が唯一決まる」とき、システムは可観測であるという。 $\mathbf{x}(0)$ が得られれば、 $\mathbf{x}(t)$ は、(5-1)式より得られる。

例：不可制御，不可観測なシステム

$$\begin{aligned}\dot{\mathbf{x}} &= \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u \\ y &= (1 \quad -1) \mathbf{x}\end{aligned}\tag{7-1}$$

を考えよう。ただし、 $\mathbf{x} = (x_1, x_2)^T$ とする。それぞれの状態について分けてみると、

$$\begin{aligned}\dot{x}_1 &= x_1 + x_2 + u \\ \dot{x}_2 &= 2x_2\end{aligned}\tag{7-2}$$

となり、どのような入力 $u(t)$ を用いても、 x_2 を操作できないことがわかる。したがって不可制御なシステムである。

一方、出力 $y(t)$ を調べてみよう。ここでは、 $u(t)=0$ の場合を考える。

$$y = x_1 - x_2\tag{7-3}$$

出力 $y(t)$ からは、状態の差しかわからない。そこで、出力信号を微分してみよう。

$$\begin{aligned}\dot{y} &= \dot{x}_1 - \dot{x}_2 \\ &= (x_1 + x_2) - 2x_2 \\ &= x_1 - x_2\end{aligned}\tag{7-4}$$

微分した信号も、やはり、 $x_1 - x_2$ という状態の差しかわからない。もう一度微分をしても $x_1 - x_2$ になる。すなわち、このシステムは、入力(この例では 0)と出力 $y(t)$ から、内部状態の差はわかるが、 x_1, x_2 を特定することができない。したがって、不可観測である。

システムの可制御性、可観測性は、状態方程式の係数行列から判断できる。システムの状態数が n である場合、可制御、可観測の必要十分条件はつぎのようになることが知られている。

$$\text{システムが可制御} \Leftrightarrow \text{rank}[B, AB, A^2B, \dots, A^{n-1}B] = n\tag{7-5}$$

$$\Leftrightarrow \text{rank}[sI - A \mid B] = n, \text{ for } \forall s \in C\tag{7-6}$$

$$\text{システムが可観測} \Leftrightarrow \text{rank} \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} = n \quad (7-7)$$

$$\Leftrightarrow \text{rank} \begin{bmatrix} sI - A \\ C \end{bmatrix} = n, \text{ for } \forall s \in C \quad (7-8)$$

である。ただし、 $\forall s \in C$ は、全ての複素数を表す。また、(7-5)式の右辺の行列を可制御性行列 (*controllability matrix*)、(7-7)式の右辺の行列を可観測性行列 (*observability matrix*) という。

7-2 レギュレータ

レギュレータ (調整器, *regulator*) とは、外乱などによって平衡点からずれた状態をすみやかに平衡点に戻すためのフィードバック制御システム (*feedback control system*) である。ここでは線形の状態フィードバック制御のうち、極配置法と最適レギュレータについて説明する。

7-2-1 極配置

状態数が n 、入力数が m の可制御なシステムを考えよう。

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (7-9)$$

今、全ての状態は直接測定可能であるとする。システムの状態 $\mathbf{x}(t)$ を入力 $\mathbf{u}(t)$ の調整に用いる制御法は状態フィードバック (*state feedback*) と呼ばれる。特に、線形の状態フィードバックは、

$$\mathbf{u} = \mathbf{K}\mathbf{x} \quad (7-10)$$

と表される。ここで、 \mathbf{K} は $m \times n$ 行列であり、フィードバックゲイン行列などと呼ばれる定数行列である。

さて、状態フィードバックを用いた場合、閉ループ系 (*closed loop system*) の状態方程式は、

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{BK})\mathbf{x} \quad (7-11)$$

となる。この閉ループ系の特性方程式は、

$$|sI - (\mathbf{A} + \mathbf{BK})| = 0 \quad (7-12)$$

である。したがって、フィードバックゲイン行列 \mathbf{K} の選び方によって、システムの極が変わることがわかる。ここで、状態フィードバックの重要な性質として、次のことが知られている。

「可制御なシステムの全ての極は、状態フィードバックによって任意に設定できる」

システムの極が指定された値 p_1, p_2, \dots, p_n になるようにフィードバックゲイン \mathbf{K} を決定することを極配置という。

例：極配置

つぎの状態方程式で表されるシステムを考えよう．

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u \quad (7-13)$$

このシステムの極は $1 \pm \sqrt{2}$ なので，不安定なシステムである．このシステムの極を -1 に配置することを考えよう．状態フィードバック，

$$u = (k_1 \quad k_2) \mathbf{x} \quad (7-14)$$

を用いると閉ループ系は，

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} (k_1 \quad k_2) \mathbf{x} \\ &= \begin{pmatrix} 0 & 1 \\ 1+k_1 & 2+k_2 \end{pmatrix} \mathbf{x} \end{aligned} \quad (7-15)$$

となる．このシステムの特性格方程式は，

$$\begin{aligned} \left| sI - \begin{pmatrix} 0 & 1 \\ 1+k_1 & 2+k_2 \end{pmatrix} \right| &= 0 \\ s^2 - (2+k_2)s - (1+k_1) &= 0 \end{aligned} \quad (7-16)$$

である．極が -1 になるためには，特性格方程式が

$$s^2 + 2s + 1 = 0 \quad (7-17)$$

となればよい．したがって，(7-16)(7-17)式を， s についての恒等式として解いて，ゲイン k_1 ， k_2 を求めれば，極配置のためのフィードバック則がつぎのように得られる．

$$u = (-2 \quad -4) \mathbf{x} \quad (7-18)$$

極配置のためのフィードバックゲインを決める方法には，システムの状態方程式を可制御正準形(*controllability canonical form*)に変換して係数を比較する方法(上記の例)や，可制御正準形を用いない疋田，Ackermann らの方法がある．

7-2-2 最適レギュレータ

可制御なシステムは状態フィードバックで任意の極配置ができる．しかし，極の値はいくつにすればよいのだろうか？

例：入力と出力

つぎのシステムを考えよう．

$$\dot{x} = x + u \quad (7-19)$$

入力を,

$$u = -kx \quad (7-20)$$

とする. このとき, 閉ループ系は,

$$\dot{x} = (1-k)x \quad (7-21)$$

となる. (7-21)式を解くと, 状態 x の時間変化は,

$$x(t) = e^{(1-k)t} x(0) \quad (7-22)$$

である. これより, k が大きければ大きいほど, 状態は速く 0 に収束することがわかる. 一方, k が大きいということは, (7-20)式から, 入力 u の絶対値が大きくなることわかる.

この例に示すように, 一般に状態の収束を速くするためには, 大きな入力を必要とする. 実システムでは, 制御入力の大きさに制約がある場合が多いので, あまり大きな制御入力は望ましくない. そこで, 入力と状態の両方をできるだけ小さくするために, 評価関数を導入して, 評価関数を最小にする制御を考える.

可制御なシステムに対して, 評価関数

$$J = \int_0^{\infty} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt, \quad Q \geq 0, R > 0 \quad (7-23)$$

を最小にする入力は, 状態フィードバックで与えられ,

$$\mathbf{u} = -R^{-1} B^T P \mathbf{x} \quad (7-24)$$

である. ただし, P は, Riccati 方程式,

$$PA + A^T P - PBR^{-1} B^T P + Q = 0 \quad (7-25)$$

の唯一の正定対称解である. この制御則を最適レギュレータ (*Linear Quadratic Regulator, LQR*) とする. このとき, 評価関数の最小値は,

$$J = \mathbf{x}(0)^T P \mathbf{x}(0) \quad (7-26)$$

となる.

評価関数の中の Q , R は重み関数と呼ばれ, 入力と出力のどちらを重視するかを決める値である. 例えば, 入力をより小さくしたいならば, R を大きく設定する.

7-3 オブザーバ

可制御なシステムは、状態フィードバックで安定化できる。しかし、通常は、内部状態を直接計測することはできない。そこで、入力と出力から内部状態を推定するオブザーバ(観測器, *observer*)を用いる。

制御対象のモデルは次式でわかっているとす。

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \quad (7-27)$$

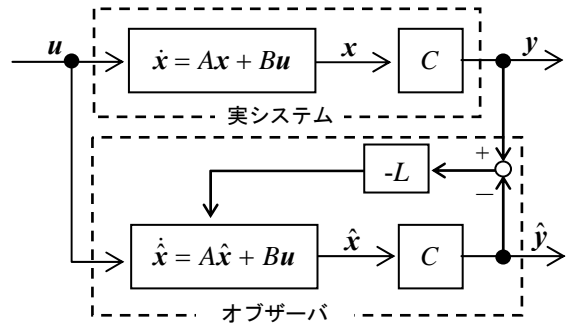


図 7-1 オブザーバ

このとき、最も直感的な方法としては、実システムへの入力をモデルにいれて、内部状態を推定する方法である。すなわち、 \mathbf{x} の推定値 $\hat{\mathbf{x}}$ を、

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} \quad (7-28)$$

という式を解いて得ることとする。この解は、

$$\hat{\mathbf{x}}(t) = e^{\mathbf{A}t} \hat{\mathbf{x}}(0) + \int_0^t e^{\mathbf{A}(t-\tau)} \mathbf{B}\mathbf{u}(\tau) d\tau \quad (7-29)$$

である。入力 $\mathbf{u}(t)$ は制御対象への入力と同じ値を用いればよい。しかし、オブザーバの状態はシステムの状態によって修正されないため、システムに外乱が加わって $\hat{\mathbf{x}}(0) \neq \mathbf{x}(0)$ となった場合、その差が残ってしまう。そこで、図 7-1 のように、制御対象とオブザーバの出力の差を用いて推定値を修正することを考える。

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} - \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \quad (7-30)$$

このオブザーバは、同一次元オブザーバ(*Luenberger observer*)と呼ばれる。このオブザーバに関して、次のことが知られている。

「可観測なシステムの状態は、オブザーバ(7-30)によって推定できる」

これは、簡単に証明することができる。まず、状態の推定誤差を $\mathbf{e} = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ とする。これを微分すると、

$$\begin{aligned} \dot{\mathbf{e}} &= \dot{\mathbf{x}}(t) - \dot{\hat{\mathbf{x}}}(t) \\ &= (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) - (\mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} - \mathbf{L}(\mathbf{y} - \hat{\mathbf{y}})) \\ &= \mathbf{A}\mathbf{x} - (\mathbf{A}\hat{\mathbf{x}} - \mathbf{L}(\mathbf{C}\mathbf{x} - \mathbf{C}\hat{\mathbf{x}})) \\ &= (\mathbf{A} + \mathbf{L}\mathbf{C})\mathbf{e} \end{aligned} \quad (7-31)$$

となるので、 $(\mathbf{A} + \mathbf{L}\mathbf{C})$ が安定行列ならば、 $t \rightarrow \infty$ で $\mathbf{e} \rightarrow 0$ に収束する。すなわち、 $\hat{\mathbf{x}} \rightarrow \mathbf{x}$ である。

ところで、 $(\mathbf{A} + \mathbf{L}\mathbf{C})$ の固有値は、 $(\mathbf{A} + \mathbf{L}\mathbf{C})^T = \mathbf{A}^T + \mathbf{C}^T \mathbf{L}^T$ の固有値と同じだから、 $(\mathbf{A}^T, \mathbf{C}^T)$ が可制御ならば、任意に配置できる。可制御性、可観測性の判定条件の双対性から、これは、 (\mathbf{C}, \mathbf{A}) が可観測であることと等価である。

演習7「極と応答／レギュレータ」（MATLAB/Simulink, control system toolbox）

	MATLAB のコマンド
極配置(Ackermann の方法)	$\text{acker}(A, b, p)$ p は, 配置したい極を並べたベクトル $p=[p_1, p_2, \dots, p_n]$ 答えは, $u=-kx$ の k . 符号に注意.
最適レギュレータ	$\text{lqr}(A, B, Q, R, N)$ $\dot{x} = Ax + Bu$ 評価関数は, $J = \int_0^{\infty} (x^T Qx + u^T Ru + 2x^T Nu) dt$ N は <i>cross term</i> という. 通常は 0 でよい. 答えは, $u=-kx$ の k . 符号に注意.

※ シミュレーションを行う場合, 「コンフィギュレーションパラメータ」の設定で, 「相対許容誤差」を小さくすると, 計算精度が上がる.

1. Simulink を用いて, 応答を見てみましょう.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -9 & -9 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$c_2 = (9 \ 0 \ 1)$$

- i) 入力を 0 として, 初期値を適当に変えて, 状態と出力を見てみましょう.
 ※ システムの極を求めてみましょう. 固有の振動モードを持っていることがわかります.
- ii) 入力をステップ関数 $u(t)=1$, 正弦波関数 $u(t)=5\sin t$ にした場合はどうなるでしょうか.
 ※ システムの固有のモードに加えて, 入力モードが加わっていることがわかれると思います.

2. 極配置を行う状態フィードバック則を求めなさい.

$$A = \begin{pmatrix} 0 & 1 & 3 & -1 \\ 0 & -1 & -2 & 2 \\ -1 & 0 & -2 & -1 \\ -1 & 0 & -3 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$c = (1 \ 0 \ 0 \ 0)$$

- ① 全ての極を -1 にする
- ② 全ての極を -3 にする

3. 2で求めたフィードバック則を用いた場合, それぞれ, 出力, 状態, 制御入力の時間変化を見てみましょう.

4. 2のシステムに対して、最適レギュレータを求めましょう。ただし、評価関数の重み行列を
つぎのように変えて求めなさい。
 - ① $Q=I_4$, $r=1$ (I_4 は、 4×4 の単位行列)
 - ② $Q=I_4$, $r=1000$
 - ③ $Q=1000 \cdot I_4$, $r=1$
5. 4で設計した最適レギュレータを用いた場合の閉ループ系の極を求めなさい。
6. 4で設計した最適レギュレータを用いた場合の初期値応答を比べてみましょう。
7. 2のシステムに対して、オブザーバを構成しなさい。オブザーバの極を
 - ① 全ての極を -1 にする
 - ② 全ての極を -3 にするとします。このときの推定誤差の収束の様子をグラフで確認しましょう。
8. 上記で設計したオブザーバを用い、2で求めたフィードバックゲインを用いて、フィードバック制御を行ったときの応答をみてみましょう。

8. サーボ系

通常、制御系設計では、まずシステムの安定化を考える。そのうえで各制御系に対する要求に応じて目的を達成させるコントローラを設計する。この制御目的の一つに目標値追従がある。目標値追従を目的としたシステムをサーボ系(*servo system*)と呼ぶ。目標値追従が可能かどうかは、制御対象の特性と目標値の種類に依存する。たとえば、自動車は緩やかなS字状の目標コースには追従できるが、直角に曲がる目標値には追従させられない。

制御対象 $P(s)$ の出力 $Y(s)$ を目標値 $R(s)$ に追従させるための制御系構成は、大きく分けて図 8-1 のようなものがある。図 8-1 は上から、フィードフォワード制御系、フィードバック制御系、2自由度制御系と呼ばれる。

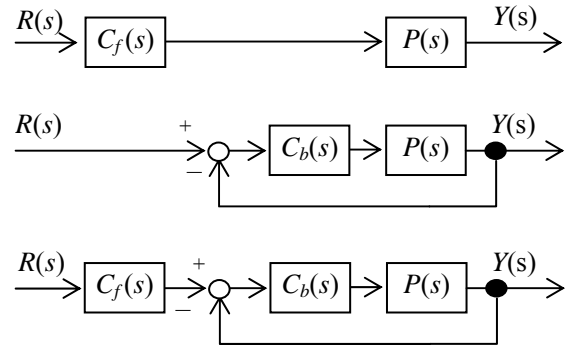


図 8-1 体系的な制御系の構成

フィードフォワード制御によって、目標値追従を達成するための直感的な方法は、逆システム(*inverse system*)を構成することである。すなわち、 $C_f(s) = 1/P(s)$ となるコントローラ $C_f(s)$ を用いれば、 $Y(s) = R(s)$ となり、出力は任意の入力に追従する。しかし、一般的には厳密な逆システムを構成することは困難である。通常、制御対象はプロパー(分母の次数 \geq 分子の次数)である。したがって、要求されるコントローラ $1/P(s)$ はプロパーではなく、微分器が必要となる。加えて、 $P(s)$ が不安定零点を持つ場合には、 $1/P(s)$ は不安定となるため、システム内部で信号が発散する。さらに、フィードフォワード制御系は一般に外乱に弱いなどの問題がある。

一方、フィードバック制御系はシステムに加わった外乱に応じて制御入力に補正をかけるので、ある程度のロバスト性が期待できる。そのため、安定化だけでなく、サーボ系の構成でもよく用いられる。本章では、フィードバック制御系について説明する。

しかし、フィードバック制御は、出力を用いて制御入力を変更するため、どうしてもシステムの動特性に応じた遅れが生じてしまう。これを改善する目的で、フィードフォワード制御を組み合わせた2自由度制御系が用いられることがある。

8-1 システムの形と定常偏差

ロボットアームを決められた位置に停止させる問題などを考えてみよう。このとき、目標位置は制御の開始から終了まで一定値である。

そこで、一定の目標値 r_d に対して、図 8-1 のフィードバック制御系の定常偏差、

$$\lim_{t \rightarrow \infty} e_{ss}(t) = \lim_{t \rightarrow \infty} (r_d - y(t)) \quad (8-1)$$

を調べてみよう。目標値をラプラス変換した信号 $R(s)$ は、

$$R(s) = r_d \frac{1}{s} \quad (8-2)$$

であるから、最終値の定理を用いれば、定常偏差 e_{ss} は、

$$\begin{aligned} \lim_{t \rightarrow \infty} e_{ss}(t) &= \lim_{s \rightarrow 0} s \frac{1}{1 + P(s)C_b(s)} \frac{r_d}{s} \\ &= \lim_{s \rightarrow 0} \frac{1}{1 + P(s)C_b(s)} r_d \end{aligned} \tag{8-3}$$

となる。これが 0 になるためには、制御対象 $P(s)$ がコントローラ $C_b(s)$ に積分器を一つ含めばよい。例えば、コントローラが $C_b(s) = C_b'(s)/s$ であれば、

$$\begin{aligned} \lim_{t \rightarrow \infty} e_{ss}(t) &= \lim_{s \rightarrow 0} \frac{1}{1 + P(s) \frac{C_b'(s)}{s}} r_d \\ &= \lim_{s \rightarrow 0} \frac{s}{s + P(s)C_b'(s)} r_d \\ &= 0 \end{aligned} \tag{8-4}$$

となり、定常偏差は 0 になることがわかる。

このことから、一定の位置に追従させる場合には積分器を用いることが多い。積分器によって定常偏差は 0 にできるが、過渡状態では偏差が残る。この過渡状態での収束を速くするためには比例要素、微分要素を併用すればよい(PID 制御)。

また、積分器を用いて定常偏差をなくす考え方は、「システムの形」として古典制御のテキストなどで述べられている(表 8-1)。

本節での定式化においては、システムの形とは、 $P(s)C_b(s)$ に含まれる積分器の数である。

表 8-1 システムの形

システムの形	定常位置偏差 (目標値がステップ)	定常速度偏差 (目標値がランプ)	定常加速度偏差 (目標値が二次関数)
0	定数 (0 以外)	∞	∞
1	0	定数 (0 以外)	∞
2	0	0	定数 (0 以外)
3	0	0	0

8-2 内部モデル原理 (internal model principle)

つぎに、さらに一般的な目標値に追従させる方法について述べる。目標値をラプラス変換した関数が一般的に、

$$R(s) = \frac{p(s)}{r(s)} \tag{8-5}$$

であるとする。図 8-1 のフィードバック制御系の制御対象 $P(s)$ とコントローラ $C_b(s)$ をそれぞれ分母と分子に分けて、

$$P(s) = \frac{b(s)}{a(s)}, \quad C_b(s) = \frac{d(s)}{c(s)} \tag{8-6}$$

と表わす。ここで、多項式 $r(s)$ と $p(s)$, $a(s)$ と $b(s)$, $c(s)$ と $d(s)$ は、それぞれ互いに素である(coprime)とする。互いに素(=既約)とはそれ以上約分できないことを意味する。

このシステムの偏差は、

$$\begin{aligned}
 e(s) &= \frac{1}{1 + \frac{b(s)d(s)}{a(s)c(s)}} \frac{p(s)}{r(s)} \\
 &= \frac{a(s)c(s)}{a(s)c(s) + b(s)d(s)} \frac{p(s)}{r(s)}
 \end{aligned}
 \tag{8-7}$$

と計算できる．この $e(s)$ の分母が安定多項式(多項式を 0 にする s の実部がすべて負)ならば，偏差 $e(t)$ は 0 に収束する．

そのためには，まず閉ループ系の特性多項式 $a(s)c(s) + b(s)d(s)$ を安定にする必要がある．つぎに， $r(s)$ に関しては，一般にサーボ系で対象にする目標値は 0 に収束するような関数ではない．例えば，前節で扱った一定目標値は $r(\infty) = r_d$ である．したがって，偏差が 0 に収束するためには，分子の $a(s)c(s)$ が $r(s)$ の不安定モード(0 に収束しないモード)を含むことが必要となる．制御対象の分母が $r(s)$ を含まない場合にはコントローラの方の分母 $c(s)$ に $r(s)$ を入れる．しかし，このとき制御対象の分子 $b(s)$ に $r(s)$ を持っているとき， $a(s)c(s) + b(s)d(s)$ が $r(s)$ を共通項として持つことになり，定常偏差は 0 にできないことに注意する．

以上をまとめると，つぎのように言える．

制御対象の分子と目標値関数の分母は互いに素であるとする．このとき，サーボ系を構成することができ，その条件は，

- ① 閉ループ系を安定にする
→ $a(s)c(s) + b(s)d(s)$ を安定多項式にする
- ② 制御対象かコントローラに目標値の不安定モードを持つ
→ $a(s)c(s)$ が $r(s)$ を含む (内部モデル原理)

である．

8-3 その他

ロボットアームなど，アクチュエータの数が多いシステムにおいては，内部モデル原理を考慮することなく，もっと直接的な制御を行うことができる．

一般に，ロボットアームの運動方程式は，一般化座標 q として各関節の角度を用いて，

$$M(q)\ddot{q} + h(q, \dot{q}) + V\dot{q} + g(q) = \tau \tag{8-8}$$

という形で表わされる．ここで， M は慣性行列， h は回転による遠心力やコリオリ力を表わす項， g は重力項， V は粘性抵抗， τ は関節モータへのトルク入力である．ここで，注意して欲しいのは，右辺の制御入力 τ の係数がない(=単位行列)である点である．このことから，モデルがわかっている場合，目標値 q_d に追従させるためには，

$$\tau = M(q, \dot{q})\ddot{q} + h(q, \dot{q}) + V\dot{q} + g(q) + (\ddot{q} - \ddot{q}_d) \tag{8-9}$$

という入力を用いれば，(8-8)に代入することで，

$$\ddot{q} = \ddot{q}_d \tag{8-10}$$

が理想的には達成できる．しかし， \ddot{q} が測定できない場合には測定値を微分することになり，制

御に用いることは一般には好ましくない。また、モデル化誤差や外乱も避けられない。そこで、例えば、

$$\begin{aligned}\boldsymbol{\tau} &= M(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{u} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + V\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \\ \boldsymbol{u} &= \ddot{\boldsymbol{q}}_d + K_v(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_d) + K_p(\boldsymbol{q} - \boldsymbol{q}_d)\end{aligned}\tag{8-11}$$

として、

$$(\ddot{\boldsymbol{q}}_d - \ddot{\boldsymbol{q}}) + K_v(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + K_p(\boldsymbol{q}_d - \boldsymbol{q}) = 0\tag{8-12}$$

を達成させる方法がある。適当な K_v , K_p として正定な行列を選ぶことで、漸近的に $\boldsymbol{q} \rightarrow \boldsymbol{q}_d$ が達成できる。

演習8 「サーボ系」 (MATLAB/Simulink, control system toolbox)

1. 次の伝達関数で表わされる制御対象に対して，サーボ系を構成しましょう．

$$y = \frac{1}{s^2 + s + 1} u$$

目標値

- ① 一定位置 $y_d = 3$
- ② 一定速度 $y_d = 2t$
- ③ 周期運動 $y_d = \sin t$

2. 次の伝達関数で表わされる不安定な制御対象に対して，サーボ系を構成しましょう．

$$y = \frac{1}{s^2 - s + 1} u$$

目標値

- ① 一定位置 $y_d = 3$
- ② 一定速度 $y_d = 2t$
- ③ 周期運動 $y_d = \sin t$

ロボットアームの制御

9. ロボットアームの制御

ロボットアームの制御目的としては、大きく分けて二つ考えられる。一つ目は、手先の位置を決められた軌道に追従させることであり、たとえば、自動の塗装ロボットなどはこの応用例である。一方、物体の把持などでは、対象物を壊さないために手先での発生力を制御する必要がある。人を対象とする福祉応用においては、この発生力の制御は不可欠となるだろう。

ここでは、一つ目の位置制御について考えることにする。ロボットアームへの制御入力は多くの場合、各関節を動かすアクチュエータの発生トルクである。よって、制御問題は、

「目標の手先座標 (x, y, z) を達成するための入力トルク τ は？」

となる。しかし、モータによって直接制御できるものは各関節の角度 θ である。この関節角度と手先位置には幾何学的な関係がある。よって、希望の手先座標を達成するための各関節角度 θ_d を算出し、モータを制御して θ を θ_d に追従させれば良い。そのため、制御問題は、次のように分けて考えられることもある。

① 目標の手先座標 (x, y, z) を達成するための各関節角度 $(\theta_1, \theta_2, \dots, \theta_m)$ は？

② その関節角度 $(\theta_1, \theta_2, \dots, \theta_m)$ を達成するための入力トルク τ は？」

さて、ここで図 9-1 の平面リンクを考えてみよう。図 9-1a と図 9-1b ではどちらが回転させやすいだろうか？ 図 9-1b の方が回転させやすいことは直感的にわかるだろう。このように、姿勢角度が異なると、同じトルクを加えてもアームの回転角速度が異なるので、②の問題は複雑なものとなる。

ここでは、2リンクの平面マニピュレータを例にして簡単な制御系設計の例を示そう。

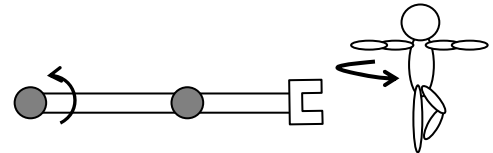


図 9-1a 伸ばした状態でのアームの回転

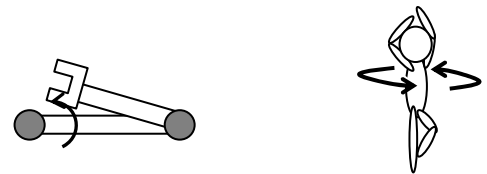


図 9-1b 曲げた状態でのアームの回転

9-1 運動学と運動学にもとづいた制御法

まず、つぎの例で、関節角度と手先の座標の関係をみてみよう。

例：2リンクマニピュレータの運動学

図 9-2 で表わされるマニピュレータを考える。手先の座標は (x, y) であり、1つ目のリンクの一端は原点にある。リンク 1 の x 軸からの角度は θ_1 、リンク 1 に対するリンク 2 の角度は θ_2 である。各リンクの長さは l_1, l_2 である。

このとき、手先の座標は幾何学的な関係から、

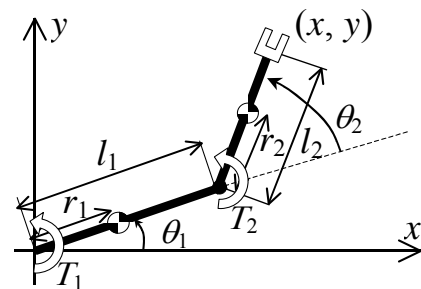


図 9-2 2リンクマニピュレータ

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{pmatrix} \quad (9-1)$$

である。なお、この式は、回転の座標変換行列を用いると、

$$\begin{pmatrix} x \\ y \end{pmatrix} = R(\theta_1) \begin{pmatrix} l_1 \\ 0 \end{pmatrix} + R(\theta_1)R(\theta_2) \begin{pmatrix} l_2 \\ 0 \end{pmatrix}, \quad R(\theta_i) = \begin{pmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{pmatrix} \quad (9-2)$$

とも表わせる。

θ_1, θ_2 が与えられた場合に、(9-1)式に代入して手先座標 (x, y) を計算することを順運動学、逆に手先座標から角度 θ_1, θ_2 を決定することを逆運動学という。順運動学は必ず解があるが、逆運動学は、関節の数が少ないと解があるとは限らない。また、 (x, y) がアームの可動範囲を超えている場合には解がない。それ以外にも次の問題がある。この例では、逆運動学の解は、

$$\begin{aligned} \theta_2 &= \cos^{-1} \frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1l_2} \\ \theta_1 &= -\sin^{-1} \frac{l_2 \sin \theta_2}{\sqrt{x^2 + y^2}} - \phi, \quad \tan \phi = \frac{-y}{x} \end{aligned} \quad (9-3)$$

であるが、逆三角関数を含んでいる。したがって、これを満たす角度の組み合わせは無数にある。どれが求める解であるかを注意する必要がある。

さて、手先の目標位置 (x_d, y_d) が与えられた場合、(9-3)の右辺の (x, y) に代入すれば、この目標を達成するための関節角度の目標値 $(\theta_{1d}, \theta_{2d})$ が計算できる。従来、産業用ロボットなどでは、各関節ごとに目標角度を達成するためのフィードバックをかける方法が用いられることが多かった。すなわち、関節 i への入力トルク τ_i を、

$$\tau_i = -k_{p_i}(\theta_i - \theta_{id}) - k_{v_i}\dot{\theta}_i \quad (9-4)$$

のように決定する方法である。この方法ではシステムの動特性を考慮しておらず、PID 制御のようにゲイン k_{p_i}, k_{v_i} は経験的に決定する。また、先に述べた②の問題の角度による動特性の変化を考慮していないので、動作速度や位置誤差の要求が厳しくなると、この制御法では追従特性が不十分になる。

9-2 動力学と運動方程式にもとづいた制御法

ロボットの動特性を考慮した制御法を考える。まず、ロボットの動特性は、1章で説明したラグランジュの運動方程式やニュートン・オイラーの運動方程式にもとづいて導出するのが一般的である。

一般にロボットアームの運動方程式は、8-3で述べたように、

$$M(q)\ddot{q} + h(q, \dot{q}) + V\dot{q} + g(q) = \tau \quad (8-8)$$

と表現される。

このモデルに対し、位置速度フィードバックを行って手先位置を一定値の目標に収束させるた

めに、ポテンシャル項の補償を加えて、つぎのような制御則を考える。

$$\boldsymbol{\tau} = -K_p(\mathbf{q} - \mathbf{q}_d) - K_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \quad (9-5)$$

(9-5)式を(8-8)式に代入すると、閉ループ系は、

$$M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + V\dot{\mathbf{q}} = -K_p(\mathbf{q} - \mathbf{q}_d) - K_v\dot{\mathbf{q}} \quad (9-6)$$

となる。このシステムの安定性を考えてみよう。

いま、スカラ関数として、

$$V = \frac{1}{2}\dot{\mathbf{q}}^T M(\mathbf{q})\dot{\mathbf{q}} + \frac{1}{2}(\mathbf{q} - \mathbf{q}_d)^T K_p(\mathbf{q} - \mathbf{q}_d) \quad (9-7)$$

を考える。このスカラ関数は常に非負である。これを時間で微分すると、

$$\begin{aligned} \frac{d}{dt}V &= \dot{\mathbf{q}}^T M(\mathbf{q})\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T \frac{dM(\mathbf{q})}{dt}\dot{\mathbf{q}} + \dot{\mathbf{q}}^T K_p(\mathbf{q} - \mathbf{q}_d) \\ &= \dot{\mathbf{q}}^T (-\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) - V\dot{\mathbf{q}} - K_p(\mathbf{q} - \mathbf{q}_d) - K_v\dot{\mathbf{q}}) + \frac{1}{2}\dot{\mathbf{q}}^T \frac{dM(\mathbf{q})}{dt}\dot{\mathbf{q}} + \dot{\mathbf{q}}^T K_p(\mathbf{q} - \mathbf{q}_d) \\ &= -\dot{\mathbf{q}}^T (V + K_v)\dot{\mathbf{q}} + \dot{\mathbf{q}}^T \left(\frac{1}{2} \frac{dM(\mathbf{q})}{dt}\dot{\mathbf{q}} - \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \right) \end{aligned} \quad (9-8)$$

となる。ここで、二項目は M と \mathbf{h} の関係から 0 になる(2リンクのモデルで確認して欲しい)。したがって、 $V + K_v > 0$ を満たすゲインを用いれば、 V は時間とともに減少を続け、最小値の 0 に収束する。 V が 0 になるとき、 $\dot{\mathbf{q}} \rightarrow 0$ 、 $\mathbf{q} \rightarrow \mathbf{q}_d$ が達成される。これはリアプノフの安定理論と呼ばれる考え方である。

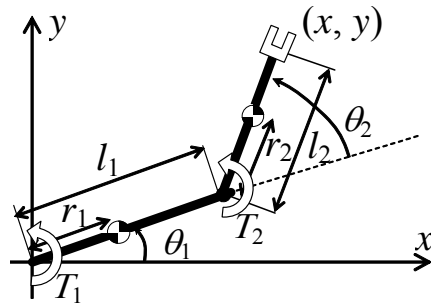
9-3 目標軌道

安定化制御の考えにもとづいた(9-5)のような制御法では、任意の初期値から目標値への収束が補償される。一方で、初期値が目標値から離れている場合には、制御初期において、大きな入力を必要とすることが(9-5)などから容易に想像がつくだろう。これに対し、適切な速度で変化する目標値（目標軌道）を与えてサーボ系を構成することで、この問題を回避することができる。

安定化制御を用いる場合、途中の目標軌道を与える必要はない。途中の軌道は制御則に応じて自動的に決まる。一方で、途中の軌道を設計者が制御することは出来ない。したがって、ロボット同士の干渉を考慮したり、障害物があるような場合にも目標軌道を与えて追従させることが必要となる。

目標軌道を与える方法として、何点かの到達させたい座標とその到達時間を与えて、それらの点を通るように関数を決定する方法がある。詳しくは、吉川著：ロボット制御基礎論，1988(コロナ社)などを参照して欲しい。

 大学院 マイクロメカトロニクス・制御特論レポート課題1



平面内で運動する2リンクのマニピュレータを考える。

1リンク目の支点を原点とし，2リンク目に取り付けられたエンドエフェクタの位置を (x, y) とする。

このマニピュレータは，1リンク目の支点，およびリンク間のジョイント部分にモータが取り付けられており，回転トルクを発生させることができる。

このモータのトルクを制御することで，エンドエフェクタの位置を目標値に追従させる問題を考える。

まず，制御系設計のためにマニピュレータの運動を表わすモデルを作成する。

そこで，つぎの2点を求めなさい。

1. 各関節角度と位置 (x, y) の幾何学的な関係を表わす式を求めなさい。
2. 各関節角度の運動を表わす運動方程式を立てなさい。

それぞれのパラメータは以下の記号を用いなさい。必要ならば，自分で適当な記号を定義して用いても良い。

	リンク1	リンク2
質量	m_1	m_2
長さ(ジョイント間)	l_1	l_2
重心まわりの慣性モーメント	I_1	I_2
支点から重心までの距離	r_1	r_2
関節の摩擦係数(回転の摩擦力は，摩擦係数×角速度)	d_1	d_2
回転角度	θ_1	θ_2
入力トルク	T_1	T_2

 大学院 マイクロメカトロニクス・制御特論レポート課題2

課題1において、制御対象の物理パラメータを下記のように指定する。

	リンク 1	リンク 2	値
質量	M_1	m_2	2.5 [kg]
長さ(ジョイント間)	l_1	l_2	0.5 [m]
重心まわりの慣性モーメント	I_1	I_2	0.05 [kgm ²]
支点から重心までの距離	r_1	r_2	0.25 [m]
関節の摩擦係数(回転の摩擦力は、 摩擦係数×角速度)	d_1	d_2	0.02 [kgm ² /s]
回転角度	θ_1	θ_2	
入力トルク	T_1	T_2	

このモデルに対して、アームの手先を指定された目標値に到達させるように制御系を設計しなさい。ただし、目標位置は、学籍番号の下二桁を用いて、

学籍番号 3*****ab

目標座標 $(x_d, y_d) = (0.05 \times (a+1), 0.05 \times b)$

とする。

制御法は問いません。授業で説明した方法でもいいし、独自に考えたり、論文等を参考にしてもかまいません。レポートには、

①制御系設計の方針（文献等を参考にした場合には、必ず出典を記載すること）

②設計したコントローラ（導出の計算を含む）

を記載しなさい。また、制御結果として

③横軸を時間、縦軸を x, y 軸としたグラフ

を載せなさい。今回の課題では、オーバーシュートが少なく、できるだけ速く目標値に到達するコントローラを良いコントローラとして評価します。

制御対象の Simlink 用のモデルは、

http://www.eng.toyo.ac.jp/~yamakawa/data/lec2009_rep2.mdl

を利用してください。モデルのファイルは、ファイル名を右クリックして、「名前をつけてリンク先を保存」してから、MATLAB を起動して実行してください。

付録 A : Mathematica と MATLAB/Simulink

◆Mathematica と MATLAB のコマンド対応

	Mathematica	MATLAB
変数定義	不要	syms t
行列の定義	$A = \{\{1,0\},\{0,1\}\}$	$A = [-1 \ 0; 0 \ -1]$
指数関数(スカラー) e^x	Exp(x)	exp(x)
行列の指数関数 e^A	MatrixExp[A]	expm(A)
行列の掛け算	$A.B$	$A*B$
単位行列 ($n \times n$)	IdentityMatrix[n]	eye(n)
階乗 $n!$	$n!$	factorial(n)
逆行列 A^{-1}	InverseMatrix[A]	inv(A)
微分 $\frac{d}{dt} x(t)$	D[x(t), t]	diff(x(t), t)
積分 $\int x(t)dt$	Integrate[x(t), t]	int(x(t), t)
ラプラス変換	LaplaceTransform[x(t), t, s]	laplace(x(t), t, s)
逆ラプラス変換	InverseLaplaceTransform[x(t), s, t]	ilaplace(x(t), s, t)

◆Mathematica でグラフを描く

Plot[f(t), {t, 0, 10}, PlotRange->{{t_{min}, t_{max}},{f_{min}, f_{max}}}]... 関数 f(t) を t=0 から t=10 までプロット。ただし、グラフは横軸が t_{min} から t_{max}、縦軸は f_{min}, f_{max} の範囲。

◆MATLAB(control system toolbox) でボード線図を描く

コマンドラインで行います。伝達関数を作る関数を用います。

伝達関数 $g = (a_1s+a_0) / (b_1s+b_0)$ を作る	$g = tf([a_1, a_0], [b_1, b_0]$)
ボード線図を描く	bodeplot(g)
ゲイン余裕, 位相余裕求め, ボード線図を描く	margin(g)
伝達関数 g の極を求める	pole(g)

◆MATLAB の「m ファイル」を作る


コマンドラインで複数の命令を繰り返し行う場合は、m ファイルを作成すると便利です。

「ファイル」→「新規作成」→「M-ファイル」とすると、新規画面が出ます。


コマンドを書き込んで、「デバッグ」→「実行」で実行します。ファイルに書かれたコマンドをまとめて実行します。結果は、コマンドウィンドウに表示されます。

付録 B : MATLAB/Simulink の使い方

◆Simulink の起動方法

MATLAB を起動した後、コマンドウィンドウで“simulink”と入力するか、ツールバーの  をクリックする。Simulink Library Browser が起動したら、メニューの「新規作成」→「モデル」で新規画面を起動する。

◆伝達関数のステップ応答をみる

1. Simulink Library Browser の「Continuous」→「Transfer Fcn」の伝達関数のブロックをドラッグアンドドロップで、モデルの新規画面に持っていく(図 B-1 上)。
2. 画面に貼り付けたモデルのブロックをダブルクリックすると、図 B-2 のような画面が表示されるので、伝達関数のパラメータを入力する。たとえば、 $G(s) = \frac{b_1s + b_0}{a_2s^2 + a_1s + a_0}$ という伝達関数ならば、分子係数を[b_1 b_0], 分母係数を[a_2 a_1 a_0]と入力する。
3. 一定値入力(ステップ入力)を入れるために、「Commonly Used Block」→「Constant」をドラッグアンドドロップ。その後、マウスをクリックしながら動かして、矢印を繋げる(図 B-1 中)。
4. 「Sinks」の Scope をモデル画面へ貼り付けて、伝達関数ブロックの出力側に繋ぐ(図 B-1 下)。
5. Scope をダブルクリックして、グラフウィンドウを表示させる。
6. メニューの「シミュレーション」→「開始」をクリックするか、ツールバーの  をクリックするとシミュレーションが開始する。

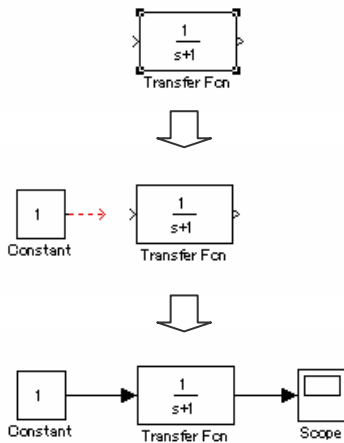


図 B-1 Simulink 上でのブロック線図



図 B-2 伝達関数のパラメータ設定画面

※ シミュレーションを行う場合、「コンフィギュレーションパラメータ」の設定で、「相対許容誤差」を小さくすると、計算精度が上がる。

付録 C：ラプラス変換

◆ ラプラス変換・逆ラプラス変換でよく使う関数

時間関数	ラプラス変換	複素関数
$\delta(t)$	\longleftrightarrow	1
1	\longleftrightarrow	$\frac{1}{s}$
t	\longleftrightarrow	$\frac{1}{s^2}$
e^{-at}	\longleftrightarrow	$\frac{1}{s+a}$
$\cos \omega t$	\longleftrightarrow	$\frac{s}{s^2 + \omega^2}$
$\sin \omega t$	\longleftrightarrow	$\frac{\omega}{s^2 + \omega^2}$

◆ ラプラス変換・逆ラプラス変換の諸定理

$$\mathcal{L}[f(t)] = F(s)$$

1. 線形性 $\mathcal{L}[a f(t) + b g(t)] = a F(s) + b G(s)$
2. 微分 $\mathcal{L}[\dot{f}(t)] = sF(s) - f(0)$
3. 積分 $\mathcal{L}\left[\int_0^t f(\tau) d\tau\right] = \frac{1}{s} F(s)$
4. むだ時間 $\mathcal{L}[f(t-T)] = e^{-sT} F(s)$
5. e^{-at} との積 $\mathcal{L}[e^{-at} f(t)] = F(s+a)$
6. たたみ込み積分 $\mathcal{L}\left[\int_0^t f(t-\tau) g(\tau) d\tau\right] = F(s)G(s)$
7. 初期値の定理 $\lim_{t \rightarrow 0} f(t) = \lim_{s \rightarrow \infty} sF(s)$
8. 最終値の定理 $\lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s)$