# 第３章　ディジタル PI 制御と離散化

ディジタル制御器の設計を行う場合，まずよく知られた連続系で設計し，得られた制御則を差分方程式に直して（**離散化** discretization という），解析や実験を行ったうえで最終的な制御ゲインや補償器を決定する場合が多い。これを**ディジタル再設計**(digital redesign)という。離散化は，制御用のコンピュータに組み込むプログラムを作る場合になくてはならないものである。制御法は多く存在し，ディジタル制御特有のものもある。本章では，良く用いられている**PI制御（比例積分制御）**やフィルタなどを例にとり，離散化の方法を述べる。また，実際には入力値に上限や下限があるので**リミッタ**が設けられる。

## 3.1　ディジタルＰＩ制御

ＰＩ（比例＋積分）制御は簡単に実現でき，ステップ応答の定常偏差を 0 にすることから，ディジタル制御においても広く用いられている。実用上大変重要である。

まず，次式で表せる積分制御について考えよう。

$$u(t)=\int_0^t e(t)dt \qquad (3\text{-}1)$$

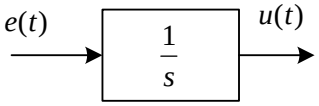



図 3-1　積分器

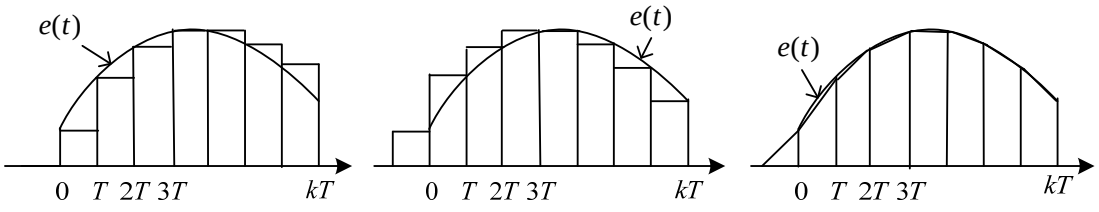
この積分を数値的に計算する場合，まず基本的な 3 つの方法を考える。

①　$u(k)=\sum_{m=0}^{k} e(m-1)T$　（**前進矩形近似**）　(3-2)

②　$u(k)=\sum_{m=0}^{k} e(m)T$　（**後退矩形近似**）　(3-3)

③　$u(k)=\frac{1}{2}\sum_{m=0}^{k}(e(m)+e(m-1))T$　（**台形近似**）　(3-4)

ただし，$u(k)\equiv u(kT), e(k)\equiv e(kT)$ と書いている。また，$e(-1)=0$ とする。これらを図形的に書くと図 3-2 となる。$t=kT$ までの長方形または台形の和が $u(k)$ である。




①　前進矩形近似　　②後退矩形近似　　③台形近似

図 3-2　積分の近似法

(3-2)から(3-4)を，差分方程式の形に書き換えてみると，次式が得られる。$u(k-1)$ は，(3-2)の $u(k)$ で $k$ を $k-1$ とおいた値である。

① $u(k)=u(k-1)+e(k-1)T$ (3-5)

② $u(k)=u(k-1)+e(k)T$ (3-6)

③ $u(k)=u(k-1)+\frac{1}{2}(e(k)+e(k-1))T$ (3-7)

これらは，それぞれ微分方程式を①前進オイラー法(単にオイラー法とも呼ばれる)，②後退オイラー法，③台形公式で差分近似したものに一致している。微分方程式の数値解析ではこれらの名称が使われる。いろいろの呼び方がある。

① **前進オイラー法（前進差分，前進矩形近似）**

(forward Euler's method, forward difference, forward rectangular rule)

$$\frac{du(t)}{dt}=e(t)\rightarrow\frac{u(k)-u(k-1)}{T}=e(k-1)$$

② **後退オイラー法（後退差分，後退矩形近似）**

(backward Euler's method, backward difference, backward rectangular rule)

$$\frac{du(t)}{dt}=e(t)\rightarrow\frac{u(k)-u(k-1)}{T}=e(k)$$

③ **台形公式（双一次変換，タスティン変換，台形近似）**

(trapezoidal rule, bilinear transform, Tustin's rule)

$$\frac{du(t)}{dt}=e(t)\rightarrow\frac{u(k)-u(k-1)}{T}=\frac{1}{2}\{e(k-1)+e(k)\}$$

台形近似は少し複雑となるが，精度はこの中では最も良い。サンプリング周期 $T$ を短く選ぶと精度は向上するが，短いほど良いというわけではない。**情報落ち**という問題がある。(3-5)〜(3-7)で $T$ が短かすぎると $e(k-1)T$ などの増分は小さな値となり，$u(k-1)$ と加えたときに，増分が全て切り捨てられてしまう場合が起こるのである。これは使用する制御用マイコンの有効桁数（固定小数点か浮動小数点か）とも関係する。積分器の近似を，上記の 3 つの方法で比較した結果，台形公式が位相角に関して最も正確であると述べられている(文献(3))。前進矩形近似では積分値に最新のデータが入っていないのは望ましくない。

(3-5)から(3-7)の差分方程式を z 変換すると，それぞれ以下のようになる。

① $U(z)=z^{-1}U(z)+z^{-1}E(z)T$

② $U(z)=z^{-1}U(z)+E(z)T$

③ $U(z)=z^{-1}U(z)+\frac{T}{2}(1+z^{-1})E(z)$

従って，パルス伝達関数は以下のようになる（積分要素$1/s$に対応）。

① $$\frac{U(z)}{E(z)}=\frac{Tz^{-1}}{1-z^{-1}}=\frac{T}{z-1}$$ （前進矩形近似） (3-8)

② $$\frac{U(z)}{E(z)}=\frac{T}{1-z^{-1}}=\frac{Tz}{z-1}$$ （後退矩形近似） (3-9)

③ $$\frac{U(z)}{E(z)}=\frac{T}{2}(\frac{1+z^{-1}}{1-z^{-1}})=\frac{T}{2}(\frac{z+1}{z-1})$$ （台形近似） (3-10)

図 3-3 の伝達関数は(3-9)となり，これが(3-6)に対応している。図 3-3 を見ると前の値に増分を加えて積分値になる様子が良くわかる。

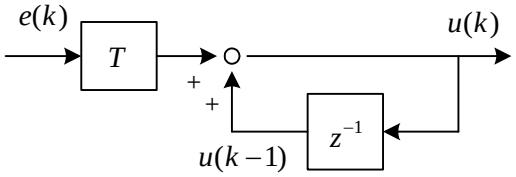



図 3-3 積分器のパルス伝達関数のブロック線図（後退矩形近似）

さて，いよいよ PI 制御の話をしよう。アナログ制御系(連続系）の場合，PI 制御は次式で表現される。

$$u(t)=K_P e(t)+K_I\int_0^t e(t)dt \tag{3-11}$$

ここで，偏差：$e(t)=r(t)-y(t)$

これを数値的に計算する場合，積分の近似の違いで基本的に 3 つの方法が考えられる。

① $$u(k)=K_P e(k)+K_I\sum_{m=0}^{k}e(m-1)T$$ （前進矩形近似） (3-12)

② $$u(k)=K_P e(k)+K_I\sum_{m=0}^{k}e(m)T$$ （後退矩形近似） (3-13)

③ $$u(k)=K_P e(k)+\frac{K_I}{2}\sum_{m=0}^{k}(e(m)+e(m-1))T$$ （台形近似） (3-14)

ここで，$e(k)=r(k)-y(k)$

これらは，**位置アルゴリズム**(position algorithm)と呼ばれている。

(3-12)から(3-14)を書き換えて，差分方程式の形に書き換えてみると，以下の式が得られる。$u(k-1)$ は，$u(k)$ で $k$ を $k-1$ とおいた値である。

① $u(k)=u(k-1)+K_P(e(k)-e(k-1))+K_ITe(k-1)$ (3-15)

② $u(k)=u(k-1)+K_P(e(k)-e(k-1))+K_ITe(k)$ (3-16)

③ $u(k)=u(k-1)+K_P(e(k)-e(k-1))+\frac{K_IT}{2}(e(k)+e(k-1))$ (3-17)

これらは，**速度アルゴリズム**(velocity algorithm)と呼ばれている。(3-15)から(3-17)をｚ変換して，ＰＩ制御器のパルス伝達関数は以下のようになる。

① $C(z)=\frac{U(z)}{E(z)}=K_P+\frac{K_IT}{z-1}$ (3-18)

② $C(z)=\frac{U(z)}{E(z)}=K_P+\frac{K_IT\,z}{z-1}$ (3-19)

③ $C(z)=\frac{U(z)}{E(z)}=K_P+\frac{K_IT}{2}(\frac{z+1}{z-1})$ (3-20)

実際のＰＩ制御としては，後退矩形近似か台形近似が用いられているようである。

次に，連続系の**ＰＩＤ制御**(proportional-integral-derivative control)は，次式で与えられる。

$$u(t)=K_Pe(t)+K_I\int_0^t e(t)dt+K_D\frac{d\,e(t)}{d\,t} \quad (3\text{-}21)$$

これを離散化する場合，積分制御と微分制御に後退矩形近似を用いると次式の**位置アルゴリズム**が得られる（文献(6)）。

$$u(k)=K_Pe(k)+K_I\sum_{m=0}^{k}e(m)T+\frac{K_D}{T}(e(k)-e(k-1)) \quad (3\text{-}22)$$

これより**速度アルゴリズム**は次式となる。

$$u(k)=u(k-1)+K_P(e(k)-e(k-1))+K_ITe(k)+\frac{K_D}{T}(e(k)-2e(k-1)+e(k-2)) \quad (3\text{-}23)$$

後述するが速度アルゴリズムにはワインドアップ防止の効果や目標値の大幅な変更に対して応答を良くする効果がある。なお$e(k)$がノイズを含む場合には，誤差が大きくなって問題である。その場合には，微分要素として後述の**擬似微分**(pseudo differential)を用いる。

## 3.2　フィルタおよび補償要素

ディジタル制御器には，ＰＩ制御の他に，一次遅れ要素や位相要素が用いられることがある。この場合の差分方程式を導いておく。連続系として表わした制御器の伝達関数を$C_c(s)=U(s)/E(s)$　とすると（$U(s),E(s)$はそれぞれ入力，偏差とは限らない），

ＰＩ制御　：$C_c(s)=K_P+\frac{K_I}{s}$，$U(s)=(K_P+\frac{K_I}{s})E(s)$ (3-24)

一次遅れ要素：　$C_c(s)=\frac{\omega_0K}{\omega_0+s}$　，　$U(s)=\frac{\omega_0K}{\omega_0+s}E(s)$ (3-25)

位相要素： $C_c(s)=K\dfrac{s+\omega_1}{s+\omega_2}$，$U(s)=K\dfrac{s+\omega_1}{s+\omega_2}E(s)$ (3-26)

と表わされる。伝達関数$C_c(s)$が与えられる場合，ディジタル補償要素$C(z)$の求め方として，次式が用いられることがある。

前進矩形近似： $C(z)=C_c(s)\big|_{s=\frac{z-1}{T}}$ (3-27)

後退矩形近似 ： $C(z)=C_c(s)\big|_{s=\frac{z-1}{Tz}}$ (3-28)

台形近似 ： $C(z)=C_c(s)\big|_{s=\frac{2(z-1)}{T(z+1)}}$ (3-29)

(3-8)～(3-10)が積分要素$1/s$の z 変換であるから，これらの関係式が得られる。但し，純粋微分$s$には台形近似は適用せず，近似微分するかその部分だけに矩形近似を用いる。これはPID 制御の D 制御で現れる。PI 制御の$C(z)$は既に(3-18)～(3-20)で求め, 差分方程式も(3-15)～(3-17)にある。

**（１） 一次遅れ要素**

- **前進矩形近似**

$$C(z)=\frac{\omega_0 K}{\omega_0+\dfrac{z-1}{T}}=\frac{T\omega_0 K}{\omega_0 T+z-1}=\frac{z^{-1}T\omega_0 K}{(\omega_0 T-1)z^{-1}+1}$$

$U(z)=C(z)E(z)$ より， $((\omega_0 T-1)z^{-1}+1)U(z)=z^{-1}T\omega_0 K\,E(z)$

$\therefore\ U(z)=(1-\omega_0 T)z^{-1}U(z)+z^{-1}\omega_0 T\,KE(z)$

差分方程式で表わすと，

$u(k)=(1-\omega_0 T)u(k-1)+\omega_0 T\,Ke(k-1)$ (3-30)

- **後退矩形近似**

$$C(z)=\frac{\omega_0 K}{\omega_0+\dfrac{z-1}{Tz}}=\frac{zT\omega_0 K}{(\omega_0 T+1)z-1}=\frac{T\omega_0 K}{\omega_0 T+1-z^{-1}}$$

$U(z)=C(z)E(z)$ より， $(\omega_0 T+1-z^{-1})U(z)=T\omega_0 K\,E(z)$

$\therefore\ U(z)=\dfrac{1}{\omega_0 T+1}z^{-1}U(z)+\dfrac{\omega_0 T\,K}{\omega_0 T+1}E(z)$

差分方程式で表わすと，

$$u(k)=\frac{1}{\omega_0 T+1}u(k-1)+\frac{\omega_0 T\,K}{\omega_0 T+1}e(k) \tag{3-31}$$

・ 台形近似

$$C(z)=\frac{\omega_0 K}{\omega_0+\dfrac{2(z-1)}{T(z+1)}}=\frac{\omega_0(T\,z+T)K}{(\omega_0 T+2)z+\omega_0 T-2}=\frac{\omega_0(T+Tz^{-1})K}{\omega_0 T+2+(\omega_0 T-2)z^{-1}}$$

$$U(z)=C(z)E(z) \quad より \quad \left\{\omega_0 T+2+(\omega_0 T-2)z^{-1}\right\}U(z)=\omega_0(T+Tz^{-1})K\,E(z)$$

$$\therefore\ U(z)=\frac{2-\omega_0 T}{2+\omega_0 T}z^{-1}U(z)+\frac{\omega_0 T(1+z^{-1})}{2+\omega_0 T}K\,E(z)$$

差分方程式で表わすと，入力$e(k)$，出力$u(k)$に対し

$$u(k)=\frac{2-\omega_0 T}{2+\omega_0 T}u(k-1)+\frac{\omega_0 T\,K}{2+\omega_0 T}\{e(k)+e(k-1)\} \tag{3-32}$$

**（２） 位相要素**

・ 前進矩形近似

$$C(z)=K\frac{\dfrac{z-1}{T}+\omega_1}{\dfrac{z-1}{T}+\omega_2}=K\frac{\omega_1 T+z-1}{\omega_2 T+z-1}=K\frac{(\omega_1 T-1)z^{-1}+1}{(\omega_2 T-1)z^{-1}+1}$$

$$U(z)=C(z)E(z) \quad より \ ((\omega_2 T-1)z^{-1}+1)U(z)=K((\omega_1 T-1)z^{-1}+1)E(z)$$

$$\therefore\ U(z)=(1-\omega_2 T)z^{-1}U(z)+K\,E(z)+K(\omega_1 T-1)z^{-1}E(z)$$

差分方程式で表わすと，

$$u(k)=(1-\omega_2 T)u(k-1)+K\,e(k)+K(\omega_1 T-1)e(k-1) \tag{3-33}$$

・ 後退矩形近似

$$C(z)=K\frac{\dfrac{z-1}{T\,z}+\omega_1}{\dfrac{z-1}{T\,z}+\omega_2}=K\frac{\omega_1 T\,z+z-1}{\omega_2 T\,z+z-1}=K\frac{\omega_1 T+1-z^{-1}}{\omega_2 T+1-z^{-1}}$$

$$U(z)=C(z)E(z) \quad より \quad (\omega_2 T+1-z^{-1})U(z)=K(\omega_1 T+1-z^{-1})E(z)$$

$$\therefore\ U(z)=\frac{1}{\omega_2 T+1}z^{-1}U(z)+\frac{K(\omega_1 T+1)}{\omega_2 T+1}E(z)-\frac{K}{\omega_2 T+1}z^{-1}E(z)$$

差分方程式で表わすと，

$$u(k)=\frac{1}{\omega_2 T+1}u(k-1)+\frac{K(\omega_1 T+1)}{\omega_2 T+1}e(k)-\frac{K}{\omega_2 T+1}e(k-1) \qquad (3\text{-}34)$$

・ **台形近似**

$$C(z)=K\frac{\dfrac{2(z-1)}{T(z+1)}+\omega_1}{\dfrac{2(z-1)}{T(z+1)}+\omega_2}=K\frac{2(z-1)+\omega_1 T(z+1)}{2(z-1)+\omega_2 T(z+1)}=K\frac{2(1-z^{-1})+\omega_1 T(1+z^{-1})}{2(1-z^{-1})+\omega_2 T(1+z^{-1})}$$

$U(z)=C(z)E(z)$　より

$$\left\{\omega_2 T(1+z^{-1})+2(1-z^{-1})\right\}U(z)=K\left\{\omega_1 T(1+z^{-1})+2(1-z^{-1})\right\}E(z)$$

$$\therefore\ U(z)=\frac{2-\omega_2 T}{2+\omega_2 T}z^{-1}U(z)+\frac{K(2+\omega_1 T_1)}{2+\omega_2 T_2}E(z)-\frac{K(2-\omega_1 T)}{2+\omega_2 T_2}z^{-1}E(z)$$

差分方程式で表わすと，

$$u(k)=\frac{2-\omega_2 T}{2+\omega_2 T}u(k-1)+\frac{K(2+\omega_1 T)}{2+\omega_2 T}e(k)-\frac{K(2-\omega_1 T)}{2+\omega_2 T}e(k-1) \qquad (3\text{-}35)$$

[問題 3-1]　制御器の伝達関数を$C_c(s)=\dfrac{U(s)}{E(s)}=\dfrac{s\omega_0 K}{\omega_0+s}$　（**擬似微分**）について，台形近似を用いて，差分方程式を導け。

（答）$u(k)=\dfrac{2-\omega_0 T}{2+\omega_0 T}u(k-1)+\dfrac{2\omega_0 K}{2+\omega_0 T}\{e(k)-e(k-1)\}$

## 3.3 FIR フィルタおよび IIR フィルタ

ディジタル信号処理の分野では，出力信号をフィードバックしない **FIR**(Finite Impulse Response:有限長インパルス応答)**フィルタ**と出力信号をフィードバックする **IIR**(Infinite Impulse Response:無限長インパルス応答)**フィルタ**に分類される。これらのフィルタも当然ノイズ除去などに利用できる。

2 次の **FIR フィルタ**は次式で与えられ，そのブロック線図を図 3-4 に示す。

$$u(k)=a_0 e(k)+a_1 e(k-1)+a_2 e(k-2) \qquad (3\text{-}36)$$

伝達関数は次式で与えられる。

$$C(z)=\frac{U(z)}{E(z)}=a_0+a_1 z^{-1}+a_2 z^{-2} \qquad (3\text{-}37)$$

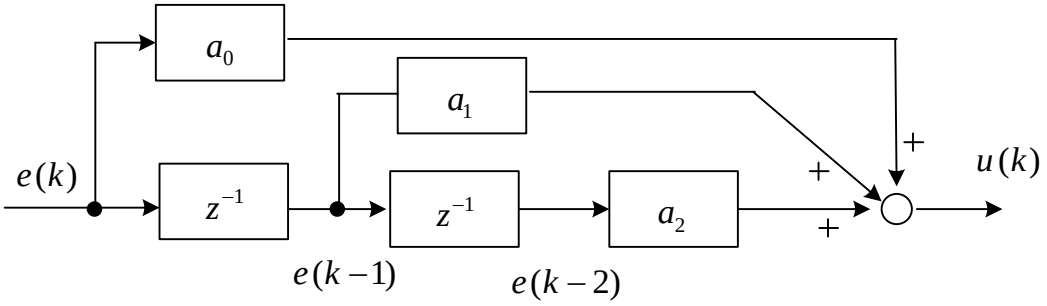



図 3-4　2 次の FIR フィルタ

$a_0 = a_1 = a_2 = 1/3$　の場合，連続する 3 サンプル値の平均をとることを意味し，**移動平均フィルタ**(moving average filter)の一例である。平均をとることでノイズを低減できる。FIR フィルタの次数を決定する方法として**窓関数法**がある(文献(13))。

1 次の IIR **フィルタ**は次式で与えられる。

$$u(k) = b_1 u(k-1) + a_0 e(k) + a_1 e(k-1) \tag{3-38}$$

(3-32)は，この一例である。(3-38)をブロック線図に示すと図のように表わせる。

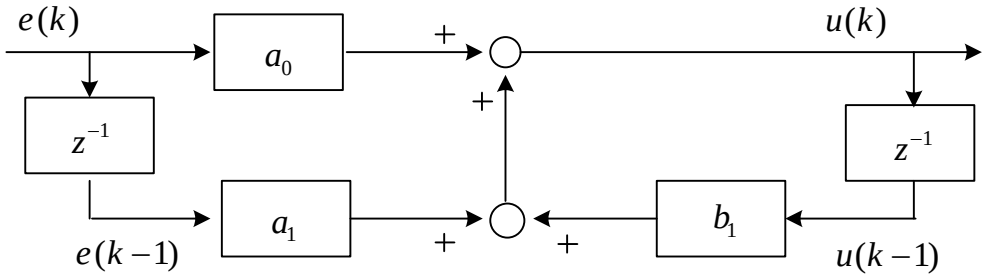



図 3-5　1 次の IIR フィルタ

伝達関数は，次式で与えられる。

$$C(z) = \frac{U(z)}{E(z)} = \frac{a_0 + a_1 z^{-1}}{1 - b_1 z^{-1}} \tag{3-39}$$

急峻なカットオフ特性を有するフィルタを作る場合，IIR フィルタは FIR フィルタより格段に少ない次数で実現できるという特徴がある。

**2次の IIR フィルタ**として，バタワース低域フィルタ(Butterworth low pass filter)を紹介す

る。ディジタルフィルタの設計の基礎となるアナログフィルタ（プロトタイプフィルタとよばれる）は，次式で与えられる。$\omega_0$ は遮断角周波数である。

$$C_c(s)=\frac{\omega_0^2}{s^2+\sqrt{2}\,\omega_0\,s+\omega_0^2} \tag{3-40}$$

減衰係数は　$\zeta=\dfrac{1}{\sqrt{2}}=0.707$　に選ばれる。この値は振幅特性$\left|G_c(j\omega)\right|$に極値を持たない減衰係数の最小値である。双一次変換を用いると

$$C(z)=\frac{\omega_0^2}{(\frac{2}{T})^2\frac{(z-1)^2}{(z+1)^2}+\sqrt{2}\,\omega_0\frac{2}{T}\frac{z-1}{z+1}+\omega_0^2}=\frac{a_0(1+2z^{-1}+z^{-2})}{1-b_1z^{-1}-b_2z^{-2}}$$

ここで，$a_0=\dfrac{(\omega_0T)^2}{4+2\sqrt{2}\,\omega_0T+(\omega_0T)^2}$

$$b_1=\frac{8-2(\omega_0T)^2}{4+2\sqrt{2}\,\omega_0T+(\omega_0T)^2},\ b_2=\frac{2\sqrt{2}\,\omega_0T-4-(\omega_0T)^2}{4+2\sqrt{2}\,\omega_0T+(\omega_0T)^2}$$

差分方程式で表わすと，

$$u(k)=b_1u(k-1)+b_2u(k-1)+a_0(e(k)+2e(k-1)+e(k-2)) \tag{3-41}$$

2 次のローパスフィルタは，1 次のローパスフィルタに比べて急峻なカットオフ特性が得られるが，位相の遅れには注意が必要だろう。

## 3.4　微分方程式で与えられた制御則の離散化

制御器の式が最初から微分方程式で与えられている場合には，直接，前進オイラー法，後退オイラー法または台形公式を使って差分近似する方が簡単である。台形公式が最も精度が良いので，一般的に利用されているであろう。非線形微分方程式（これは状態オブザーバを利用した制御などで現れる）を差分近似する場合，後退オイラー法または台形公式を使うと，非線形方程式を解く必要があるので離散化しにくいことがある。この場合には精度は悪くても前進オイラー法が使われているだろう。精度を良くしたい場合には，前進オイラー法と台形公式を組み合わせた 2 次の Runge-Kutta 法（修正オイラー法）がある。サンプリング周期が非常に短い場合には，これらの方法に大きな差はなくなるだろうが，先に述べた情報落ちの問題がある。

一般の非線形微分方程式を前進オイラー法, 後退オイラー法, 台形公式, 2 次の Runge-Kutta

法で差分近似する。

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{3-42}$$

例：
$$\frac{dx_1(t)}{dt} = f_1 = -2x_1^2(t) + u_1(t) + \sin 2t$$

$$\frac{dx_2(t)}{dt} = f_2 = -2x_1(t)x_2(t) + u_1(t)u_2(t)$$

① **前進オイラー法**

$$\frac{\boldsymbol{x}(k) - \boldsymbol{x}(k-1)}{T} = \boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), (k-1)T)$$

$$\therefore \quad \boldsymbol{x}(k) = \boldsymbol{x}(k-1) + T\,\boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), (k-1)T) \tag{3-43}$$

常に容易に計算できる。

② **後退オイラー法**

$$\frac{\boldsymbol{x}(k) - \boldsymbol{x}(k-1)}{T} = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), kT) \tag{3-44}$$

右辺にも $\boldsymbol{x}(k)$ が含まれるので一般に方程式を解かないといけない。

③ **台形公式**

$$\frac{\boldsymbol{x}(k) - \boldsymbol{x}(k-1)}{T} = \frac{1}{2}\{\boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), kT) + \boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), (k-1)T)\} \tag{3-45}$$

右辺にも $\boldsymbol{x}(k)$ が含まれるので一般に方程式を解かないといけない。

④ **2 次の Runge-Kutta(ルンゲクッタ)法（修正オイラー法，ホイン法）**

$$\hat{\boldsymbol{x}}(k) = \boldsymbol{x}(k-1) + T\,\boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), (k-1)T) \tag{3-46}$$

$$\boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \frac{T}{2}(\boldsymbol{f}(\hat{\boldsymbol{x}}(k), \boldsymbol{u}(k), kT) + \boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), (k-1)T)) \tag{3-47}$$

台形公式で微係数 $\boldsymbol{f}$ を求めるときに， $\boldsymbol{x}(k)$ の代わりにオイラー法で求めた $\hat{\boldsymbol{x}}(k)$ を使用する。方程式を解く必要はない。

[例題 3-1]　次の微分方程式を前進オイラー法, 後退オイラー法, 台形公式で差分近似せよ。

$$\frac{dx(t)}{dt} = -a\,x(t) + b\,u(t)$$

(解)

① **前進オイラー法**

$$\frac{x(k) - x(k-1)}{T} = -a\,x(k-1) + b\,u(k-1) \quad \therefore \quad x(k) = (1 - aT)\,x(k-1) + bT\,u(k-1)$$

② **後退オイラー法**

$$\frac{x(k)-x(k-1)}{T}=-a\,x(k)+b\,u(k)$$

$$\therefore \quad x(k)=\frac{1}{1+aT}x(k-1)+\frac{bT}{1+aT}u(k)$$

② **台形公式**

$$\frac{x(k)-x(k-1)}{T}=\frac{1}{2}\left\{-a\,x(k)+b\,u(k)-a\,x(k-1)+b\,u(k-1)\right\}$$

$$\therefore \quad x(k)=\frac{2-aT}{2+aT}x(k-1)+\frac{bT}{2+aT}u(k)+\frac{bT}{2+aT}u(k-1)$$

［例題 3-2］ 制御器の伝達関数を $\dfrac{U(s)}{E(s)}=K\dfrac{s+\omega_1}{s+\omega_2}$ とする。微分方程式に戻して，台形公式を適用し差分方程式を導け。

（解）この微分方程式については，次式が成立する。

$$\frac{d}{d\,t}\left(u(t)-K\,e(t)\right)=K\,\omega_1 e(t)-\omega_2 u(t)$$

これに，台形公式を適用して，次式が得られる。

$$\frac{u(k)-K\,e(k)-(u(k-1)-K\,e(k-1))}{T}$$

$$=\frac{1}{2}\left\{K\,\omega_1 e(k)-\omega_2 u(k)+K\,\omega_1 e(k-1)-\omega_2 u(k-1)\right\}$$

故に $u(k)=\dfrac{2-\omega_2 T}{2+\omega_2 T}u(k-1)+\dfrac{K(2+\omega_1 T)}{2+\omega_2 T}e(k)-\dfrac{K(2-\omega_1 T)}{2+\omega_2 T}e(k-1)$

［問題 3-2］ 次の非線形微分方程式を 2 次のルンゲクッタ法で離散化せよ。

$$\frac{dx}{dt}=x\sin(2t)+1$$

（解） $\hat{x}(k)=x(k-1)+T\{x(k-1)\sin(2(k-1)T)+1\}$

$$x(k)=x(k-1)+\frac{T}{2}\{\,\hat{x}(k)\sin(2kT)+1+x(k-1)\sin(2(k-1)T)+1\}$$

## 3.5 リミッタ

制御対象への入力 $u(k)$ は装置の容量によって，その大きさが制限される。そのため，次式の**リミッタ**（limiter）が設けられる。

$$|u(k)| \le u_{\max} \tag{3-48}$$

ところが，このリミッタをどのようにプログラムするかで，応答に大きな違いが生じる。PI 制御の場合を例にとり，以下に詳しく述べる。

$$u(k) = K_P e(k) + K_I \sum_{m=0}^{k} e(m)T \qquad \text{（後退矩形近似）} \tag{3-13}$$

を用いてプログラムする場合，比例制御と積分制御を分けて，以下の様にプログラムしたとしよう。なお，このプログラムはサンプリング周期$T$ ごとに繰り返し実行される。この様子を図 3-6 に示す。積分の計算はメモリに値を蓄えることで容易に行える。

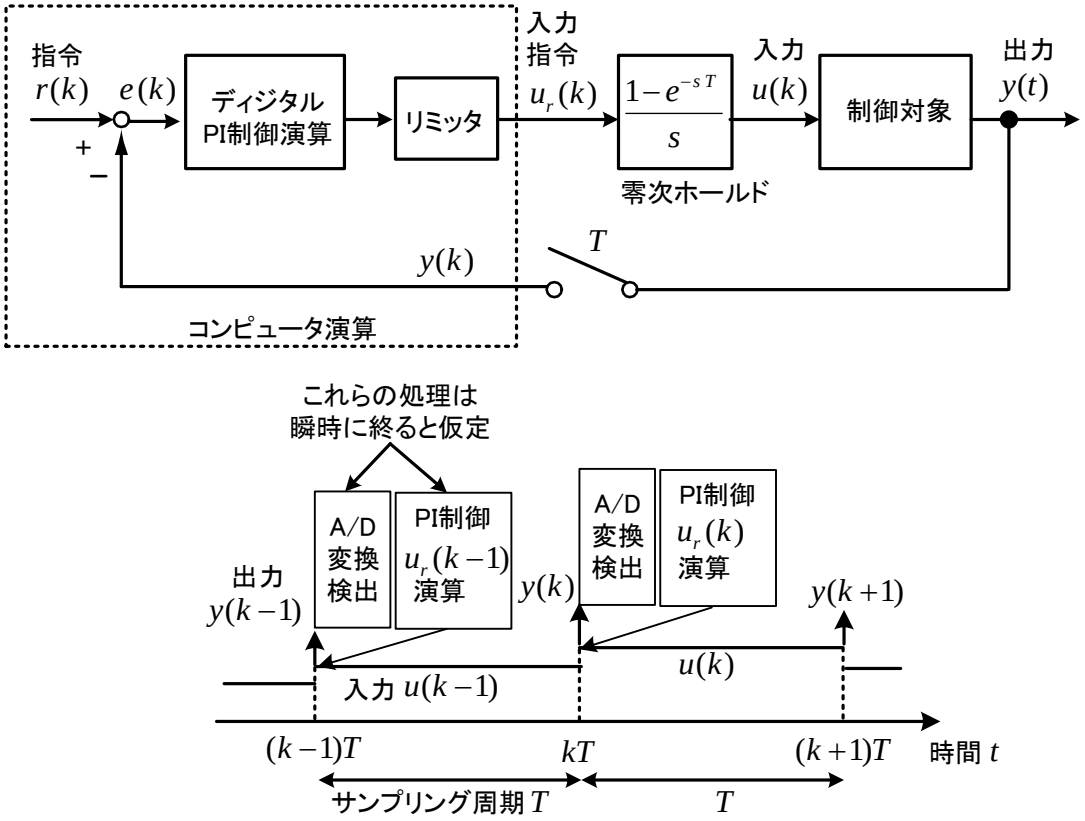



図 3-6　ディジタル PI 制御の実行過程

リミッタを含む PI 制御プログラム（その 1）

| | | |
|---|---|---|
| $e = r - y$ ； | 偏差$e$の計算 | ① |
| $u_p = K_p * e$ ； | 比例量計算 | ② |
| $u_{inew} = u_{iold} + K_I * T * e$ ； | 積分量計算 | ③ |
| $u = u_p + u_{inew}$ ； | 入力指令＝比例量＋積分量 | ④ |

$if\ (u > u_{\max})\quad u = u_{\max}\quad ;$　　リミッタ上限　　⑤

$if\ (u < (-u_{\max}))\quad u = -u_{\max}\quad ;$　　リミッタ下限　　⑥

$u_{iold} = u_{inew}\quad ;$　　積分量更新（次回使用）　　⑦

（注意）一番最初だけ$u_{iold} = 0$として変数を初期化する。

素直なプログラムであるが，これには大きな問題がある。指令値のステップ変化に対する応答についてこの問題を考える。単純に$u$のみにリミッタを設けているため，$u$がリミッタで制限されているときでも積分量は制限されることなくどんどん値が蓄積される。このため，$u$がリミッタを抜けるまでに時間がかかり，出力$y$に大きなオーバーシュートが生じる。これは**ワインドアップ現象**（windup）と呼ばれている。図 3-7 はモータの PI 速度制御の例である。出力$y$[min$^{-1}$] は回転速度, 入力$u$ [A] はトルク電流で, 速度指令$r$を 1000min$^{-1}$ から 1500min$^{-1}$ さらに 1000min$^{-1}$ に変化させている。入力$u$ [A] の最大値は 10A に設定している。1500min$^{-1}$ に上昇させる場合，偏差$e$が負になる点で$u_{inew}$は減少し始めるが，それまでの積分で$u_{inew}$が大きく$u$はその後もリミッタの上限値となってワインドアップ現象が生じている。その後，1000min$^{-1}$ に減少させる場合には，積分量$u_{inew}$は−10A に達しておらず，入力$u$がリミッタにかかる時間は短い。速度のアンダーシュートは, モータの負荷トルク（電流換算で$u$の 5A 相当）とモータが出す負のトルクとが同じ向きになるためと考えられる（加速時の加速トルク 10-5=5A 相当：減速時の減速トルク-10-5=-15A 相当）。

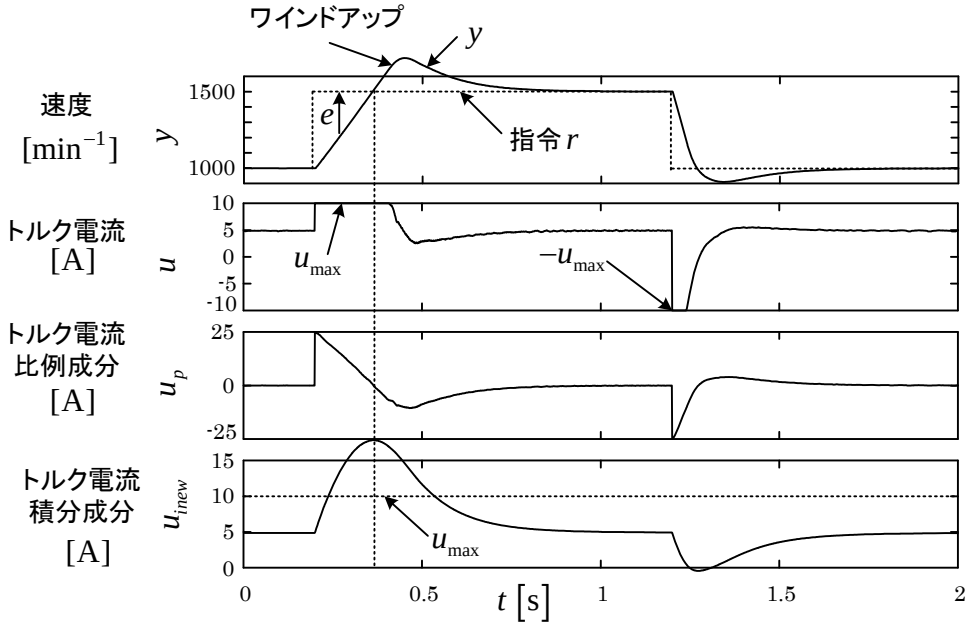



図 3-7 プログラムその 1 の場合

ワインドアップを防ぐためには, $u$ がリミッタにかかったときに積分器に値をためないようする必要がある。つまり $u_{inew}$ にも何らかのリミッタをかければよい。

(3-16)の差分方程式を用いる場合，以下のプログラムで $u(k)$ にリミッタをかけた場合には，ワインドアップ現象は生じない。

リミッタを含むPI制御プログラム（その2）

| | | |
|---|---|---|
| $e = r - y$ ； | 偏差 $e$ の計算 | ① |
| $u = u_{old} + K_P * (e - e_{old}) + K_I * T * e$ ； | 入力指令計算 | ② |
| $if\ (u > u_{\max})\quad u = u_{\max}$ ； | リミッタ上限 | ③ |
| $if\ (u < (-u_{\max}))\quad u = -u_{\max}$ ； | リミッタ下限 | ④ |
| $u_{old} = u$ ； | 入力指令更新（次回使用） | ⑤ |
| $e_{old} = e$ ； | 偏差更新（次回使用） | ⑥ |

（注意）一番最初だけ $u_{old} = 0,\ e_{old} = 0$ として変数を初期化すること。

ワインドアップが生じないのは，$u(k-1)$ に偏差の項を加えて，新しく $u(k)$ が求まるため，$u(k)$ がリミッタにかかり変化しなければ，その間は値をためる量が存在しないことによる。ただし，⑤を②の後に書くと，$u_{old}$ に値がたまりワインドアップ現象は起きる。(3-16)の注意点としては，指令値が大きくステップ変化するとき，出力 $y$ を速く立ち上げるためには入力 $u$ は $u_{\max}$ となることが望ましいが，$e(k)$ と $e(k-1)$ の差を $K_p$ 倍することになるため，$u$ が $u_{\max}$ とならないことが考えられる。ただし，穏やかな変化を望むならばそれはメリットにもなり得る。図 3-8 はモータの PI 速度制御の例である。図のように比例成分 $u_p = K_P e$ が $u_{\max}$ を超えても，$u$ が $u_{\max}$ となっていない。

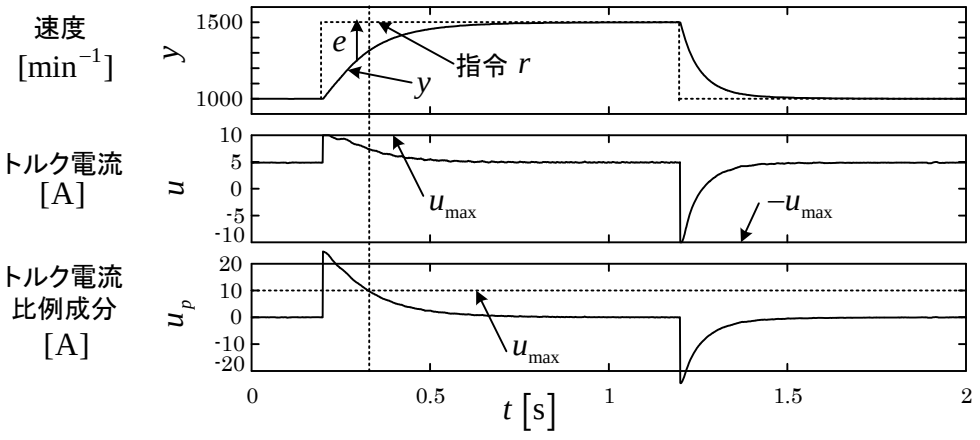



図 3-8 プログラムその２の場合

応答を速くするために，$u$ だけでなく比例成分 $u_p = K_P e$ が制限値を超えたら，最大値または最小値を出力することが考えられる。以下に，そのプログラムを示す。

リミッタを含む PI 制御プログラム（その 3）

$e = r - y$ ； 偏差 $e$ の計算 ①

$u = u_{old} + K_P * (e - e_{old}) + K_I * T * e$ ； 入力指令計算 ②

$if\ (u > u_{\max} \,||\, K_P * e > u_{\max})\quad u = u_{\max}$ ； リミッタ上限 ③

$if\ (u < (-u_{\max}) \,||\, K_P * e < (-u_{\max}))\quad u = -u_{\max}$ ； リミッタ下限 ④

$u_{old} = u$ ； 入力指令更新（次回使用） ⑤

$e_{old} = e$ ； 偏差更新（次回使用） ⑥

（注意）一番最初だけ $u_{old} = 0,\ e_{old} = 0$ として変数を初期化すること。

|| は OR の演算子である。

図 3-9 はモータの PI 速度制御の例である。図のように比例成分 $u_p = K_P e$ が $u_{\max}$ を超えると，$u$ が $u_{\max}$ となって図 3-8 に比べて速度の応答が速くなっている。ただ，減速の場合には速度のアンダーシュートが生じている。プログラムその１は明らかに良くないが，その２とその３は目的によって選択すべきであろう。

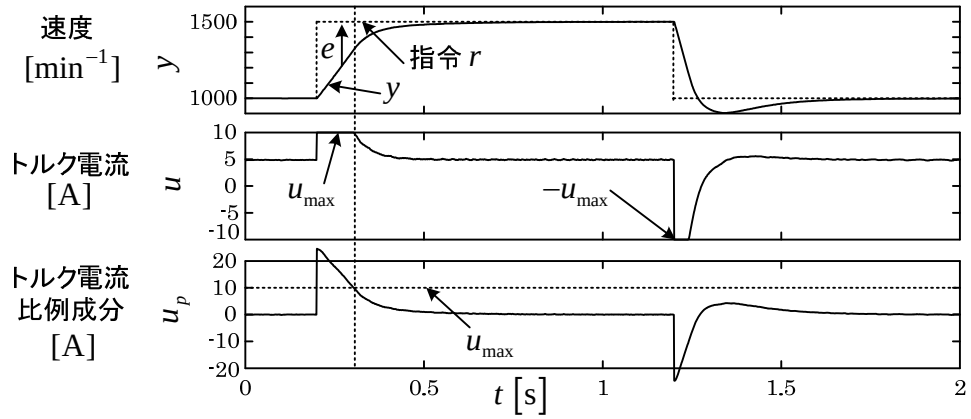



図 3−9 プログラムその３の場合

(3-13)と(3-16)は，リミッタにかからなければ等価であるが，リミッタにかかると，その処理の仕方によって応答が大きく異なるので注意して欲しい。

［問題 3−3］ (3-13)のプログラムその１ではワインドアップが生じるのに，数学的に等価な (3-16) のプログラムその２ではワインドアップが生じない。何故か。