ニューラルネットワークによるPIDパラメータの自動調整

藤田 透*・大松 繁*

1. はじめに

プロセス制御の現場では、プラントに設置された数多くの計測デーツがコンピュータへ送られ、オペレータは計測データをチェックしながら、PIDパラメータの調整を行っている。そこで選択されるデーツは膨大であり、オペレータがさまざまな状況に対応した制御入力（PIDゲインの値）と、そのときの制御結果が記録されている。したがって、これらのデータを有効活用してオペレータの知識が抽出できれば、オペレータの作業負担を軽減できると期待される。換言すれば、数値データから知識を抽出するデータマイニングへの期待は大きい。しかしながら、数値データから知識データへの変換は容易ではなく、エキスパートシステムの設計に見られたようにその信頼性にも疑問が残っている。

ところで、ニューラルネットワークは、人間の脳の機構を工学的に近似したもので、ニューロンとよばれる非線形要素を多数連結した回路網であり、学習可能な並列分散処理システムとして注目を集めている[1-5]。ニューラルネットワークはさまざまな分野に応用されているが、上記の蓄積データがインデータをPIDゲインを学習するようなニューラルネットワークを構成すれば、オペレータの知識を効率よく抽出できるものと期待される。

以下では、現場の運転データ（プラントの入出力データ）を利用してPID制御のパラメータチューニングを行う手法のひとつとして、ニューラルネットワークによるPIDゲインの自動調整法を述べる。

2. ニューロPID

この節では、まず、PID制御について概要を述べ、つぎに、ニューラルネットワークによるPIDゲインのチューニング法について述べる。

2.1 PID制御系

通常のPID制御系は、つきの第1図に示すような構成からなっている。

第1図において、uはプラントへの操作入力、dは外乱入力、yはプラントの出力、rはyの目標値、eは偏差（目標値とプラント出力との誤差）であり、sはラプラス変換の変数（微分作用素）を表す、破線で囲んだ部分がPID制御器に対応しており、k_pが比例動作、t_iが積分動作、t_dが微分動作に関するパラメータである。

PIDコントローラの入力は偏差e=r−y、出力は操作量uであり、yからuまでの伝達関数G_c(s)は次式で与えられる。

\[ G_c(s) = k_p \left( \frac{1}{t_i s} + \frac{1}{t_d s} \right) \]  \(1\)

このPIDコントローラの動作は時間領域でつきのように記述される。

\[ u(t) = k_p \left( e(t) + \frac{1}{t_i} \int_0^t e(\tau) d\tau + t_d \frac{d}{dt} e(t) \right) \]  \(2\)

ここで、どのようにして、k_p、t_i、t_dを決定するかというのが本研究の問題であるが、以前からよく用いられる方法のひとつとして、Ziegler-Nichols法とよばれるものがある。これは、プラントのステップ応答第2図に示すような線形をもつことを前提として、以下のようにPIDゲインを決定するものである。

![第2図 Ziegler-Nichols法](image)

まず、ステップ応答曲線で勾配が最大となる点に接線を引き、その傾きをRとする。そして、この接線が時間軸と交わる点を求め、その時刻をLとする。このLは等価とした時間に相当している。得られたRとLを使って、次式によりk_p、t_i、t_dの値を設定する。

Key Words: PID, neural network, parameter tuning.
\[ k_p = \frac{1.2}{RL}, \quad t_i = 2L, \quad t_d = \frac{L}{2} \]  
\[ u(n) = k_p \left[ e(n) + \sum_{j=0}^{n-1} \frac{T}{2} (e(j+1) - e(j)) \right] + \frac{t_d}{T} (e(n) - e(n-1)) \]  

ただし、上式では積分に対して台形近似を用いている。
これを変形すると、離散時間系のPIDに対する次式の表現を得る。
\[ u(n) = u(n-1) + K_p (e(n) - e(n-1)) + K_i e(n) \]
\[ + K_d (e(n) - 2e(n-1) + e(n-2)) \]  

ここで、\( K_p, K_i, K_d \)は離散時間系のPIDゲインであり、連続時間系のパラメタとの対応は次式のとおりである。
\[ K_p = k_p - \frac{1}{2} K_i, \quad K_i = k_p \frac{T}{t_i}, \quad K_d = k_p t_d \]  

離散時間系にZiegler-Nichols法を適用すると、PIDゲインはつきのように定められる。
\[ K_p = k_p - \frac{K_i}{2} = \frac{1.2}{G_0 L_o} - \frac{K_i}{2} \]  
\[ K_i = \frac{1.2}{RL} \frac{T}{2L} = \frac{0.6}{RT} = \frac{0.6}{G_0 L_o^2} \]  
\[ K_d = k_p t_d = \frac{0.6}{G_0} \]  

ただし、
\[ G_0 = \max \left( \frac{y(n)-y(n-1)}{\frac{L}{T}} \right), \quad L_o = \frac{T}{L} \]  

ここで、\( r \)は目標値を示しており、偏差\( e \)をPIDコントローラに入力して制御入力\( u \)を計算している。PIDコントローラの入力出力関係は(5)式で与えられている。

ニューラルPID制御系ではブラントの入力出力データを利用して離散時間系のPIDゲインを自動的にチューニングするが、その際の初期ゲインとして(9)式がよく用いられる。

\[ u(n) = u(n-1) + K_p (e(n) - e(n-1)) + K_i e(n) \]
\[ + K_d (e(n) - 2e(n-1) + e(n-2)) \]  

\[ G_0 = \max \left( \frac{y(n)-y(n-1)}{\frac{L}{T}} \right), \quad L_o = \frac{T}{L} \]  

ニューラルPID制御系ではブラントの入力出力データを利用して離散時間系のPIDゲインを自動的にチューニングするが、その際の初期ゲインとして(9)式がよく用いられる。

2.2 PIDゲインのチューニング法

以下の式に示すニューラルPID制御系を考察する。

ニューラルPID制御系の構造は、

ニューラルPID制御系の構造は、

これで、ニューラルネットワークの入力は、ブラントの現在および過去の入力と出力であり、また、その出力

ニューラルネットワークの入力は、ブラントの現在および過去の入力と出力であり、また、その出力

ニューラルネットワークの入力は、ブラントの現在および過去の入力と出力であり、また、その出力

ニューラルネットワークの入力は、ブラントの現在および過去の入力と出力であり、また、その出力
階層型ニューラルネットワークとなり，誤差逆伝播法が適用可能になる。

以下では，この考え方を基に，ニューラルPID制御系を設計する．PIDコントローラは，(5)式で与えられる．PIDゲイン $K_p, K_i, K_d$ の初期値は，たとえば高橋-Chenによる(9)式の値を設定すればよい．

さて，PIDゲインを更新するための評価指標を決めなければならない．ここでは以下の $E(n)$ を各時刻，$n=1,2,...$ で最小にするものとする．

$$E(n) = \frac{1}{2}e(n)^2, \quad n=1,2,...$$  

(10)

$$e(n) = r(n) - y(n)$$  

(11)

ニューラルネットワークの出力は，$K_p , K_i , K_d$ という3個のパラメータであるから，出力層のニューロン数は3である．また，出力層の出力は出力の入力の範囲に限定されないため，出力層のニューロンはシグモイド関数ではなく線形関数の特性をもつものとする．この場合，次式が成立する．

$$K_p = O_k(1), \quad K_i = O_k(2), \quad K_d = O_k(3)$$  

(12)

$$O_k(m) = \text{net}_k(m) = \sum_{j=0}^{N} W_{kj}O_j, \quad m=1,2,3$$  

(13)

ただし， $O_k(m), m=1,2,3$ は出力層のニューロン $O_k$ の第 $m$ 個の要素であることを示す．以後，記述を簡単にするために，

$$O_k(1) = O_1, \quad O_k(2) = O_2, \quad O_k(3) = O_3$$  

(14)

$$\text{net}_k(m) = \text{net}_m, \quad \delta_k(m) = \delta_m, \quad m=1,2,3$$  

(15)

と記述する．$W_{kj}, \quad \text{および} \quad W_{ji}$ の更新則は，誤差逆伝播法の場合と同様に，以下に示す勾配法を用いて導出する．

$$\Delta W_{kj}(n+1) = W_{kj}(n+1) - W_{kj}(n)$$  

$$= -\eta \frac{\partial E(n+1)}{\partial W_{kj}} \Bigg|_{W_{kj}=W_{kj}(n)} + \alpha \Delta W_{kj}(n)$$  

(16)

ここに，$k=1,2,3; \quad j=0,1,\ldots, N$ とする．ただし，$N$ は中間層のニューロン数を示している．

$$\Delta W_{ji}(n+1) = W_{ji}(n+1) - W_{ji}(n)$$  

$$= -\eta \frac{\partial E(n+1)}{\partial W_{ji}} \Bigg|_{W_{ji}=W_{ji}(n)} + \alpha \Delta W_{ji}(n)$$  

(17)

ここに，$j=1,2,\ldots, N; \quad i=0,1,\ldots, M$ とする．ただし，$M$ は入力層の総数を示す．$\eta > 0$ と $\alpha$ は，それぞれ，学習係数と慣性係数を示し，$n$ は更新時刻を示している．そこで，(16)式の勾配を以下で求める．

いま，$\delta_k$ を次式で定義する．

$$\delta_k = -\frac{\partial E(n+1)}{\partial \text{net}_k}$$  

(18)

微分の連鎖則を用い，(13)式と(18)式を用いると次式を得る．

$$\frac{\partial E(n+1)}{\partial W_{kj}} = \frac{\partial E(n+1)}{\partial \text{net}_k} \frac{\partial \text{net}_k}{W_{kj}} = \delta_k O_j$$  

(19)

さらに，微分の連鎖則を(18)式に適用すると次式を得る．

$$\delta_k = -\frac{\partial E(n+1)}{\partial y(n+1)} \frac{\partial y(n+1)}{\partial u(n)} \frac{\partial O_k}{\partial \text{net}_k}$$  

(20)

いま，(10)式の $E(n)$ と(11)式の $e(n)$ の定義を用いると次式を得る．

$$\frac{\partial E(n+1)}{\partial y(n+1)} = \frac{\partial E(n+1)}{\partial e(n+1)} \frac{\partial e(n+1)}{\partial y(n+1)} = e(n+1)$$  

(21)

出力層では，ニューロンの出力特性が線形だから，次式を得る．

$$\frac{\partial O_k}{\partial \text{net}_k} = 1$$  

(22)

また，$K_p = O_1, K_i = O_2, K_d = O_3$ に注意すると(5)式から次式を得る．

$$\frac{\partial u(n)}{\partial O_k} = \begin{cases} e(n) - e(n-1) & (k=1) \\ e(n) & (k=2) \\ e(n) - 2e(n-1) + e(n-2) & (k=3) \end{cases}$$  

(23)

さらに，(20)式の中に含まれている $\partial y(n+1)/\partial u(n)$ はブレント入力をわずかに変動したときに生じるブレント出力の変動と入力変動の比に対応するもので，システムの非線形があらわれている．以下ではこれを $J(n)$ とおく．

$$J(n) = \frac{\partial y(n+1)}{\partial u(n)}$$  

(24)

このとき，(20)式は次式で記述される．

$$\delta_k = e(n+1)J(n) \frac{\partial u(n)}{\partial O_k}, \quad k=1,2,3$$  

(25)

したがって，ニューラルネットワークの出力層の結合強度は，慣性項を含めて以下のように記述される．

$$\Delta W_{kj}(n+1) = W_{kj}(n+1) - W_{kj}(n)$$  

$$= \eta \delta_k O_j + \alpha \Delta W_{kj}(n)$$  

(26)

つぎに，中間層のニューロンの結合強度の更新について考える．(17)式から，以下の関係式を得る．

$$\Delta W_{ji}(n+1) = -\eta \frac{\partial E(n+1)}{\partial W_{ji}} \Bigg|_{W_{ji}=W_{ji}(n)} + \alpha \Delta W_{ji}(n)$$  

(27)

階層型ニューラルネットワークの誤差逆伝播法と同様に，微分の連鎖則より次式が成立する．
ここに，\( \delta_j \) は次式で定義する。

\[
\delta_j = \frac{\partial E(n+1)}{\partial \text{net}_j} = \frac{\partial E(n+1)}{\partial \text{net}_j} = \delta_j O_i
\]  (28)

階層型ニューラルネットワークの誤差逆伝播法と同様に，
微分の連続性則より次式が成立する。

\[
\delta_j = -\frac{\partial E(n+1)}{\partial \text{net}_j}
\]

\[
= \sum_{i=1}^{K} \left( \frac{-\partial E(n+1)}{\partial \text{net}_k} \right) \frac{\partial \text{net}_k}{\partial O_j} \frac{\partial O_j}{\partial \text{net}_j}
\]

\[
= \sum_{i=1}^{K} \delta_k W_{kj} O_j (1-O_j)
\]  (30)

したがって，以下の更新式を得る。

\[
\Delta W_{ji}(n+1) = \eta \delta_j O_i + \alpha \Delta W_{ji}(n)
\]  (31)

\[
\delta_j = \sum_{i=1}^{K} \delta_k W_{kj} O_j (1-O_j)
\]  (32)

2.3 学習アルゴリズム

以上をまとめると，学習用データセットに対して以下のようニューールPID制御の学習手順を得る。

Step 1 初期設定
ニューラルネットワークのパラメータ \( W_{kj} \)，\( W_{ji} \) は，学習前各戦 \( \epsilon \) でニューラルネットワークへの入力 \( O_i \) と出力の目標値 \( r(n) \) を設定する。

Step 2 プラント入出力の読み込み
プラント入力 \( u(n) \) を読み込む。

Step 3 ニューラルネットワーク出力の計算

\[
O_j = f(\text{net}_j), \text{net}_j = \sum_{i=0}^{M} W_{ji} O_i - \theta_j
\]

\[
f(x) = \frac{1}{1 + \exp(-x)}
\]  (33)

\[
O_k = \text{net}_k
\]

\[
\text{net}_k = \sum_{j=0}^{N} W_{kj} O_j, \quad k = 1, 2, 3
\]  (34)

Step 4 PID制御によるプラント入力と出力の計算

\[
u(n) = u(n-1) + K_p (e(n) - e(n-1)) + K_i e(n)
\]

\[
+ K_d (e(n) - 2e(n-1) + e(n-2))
\]  (35)

上記の入力 \( u(n) \) に対する出力 \( y(n+1) \) を計算（測定）する。

Step 5 誤差の計算と評価

\[
e(n+1) = r(n+1) - y(n+1)
\]  (36)

\[
E(n+1) = \frac{1}{2} e(n+1)^2
\]  (37)

Step 6 一般化誤差の計算

\[
\delta_k = e(n+1) J(n) \frac{\partial u(n)}{\partial \theta_k}, \quad k = 1, 2, 3
\]  (38)

\[
\delta_j = \sum_{k=1}^{3} \delta_k W_{kj} O_j (1-O_j),
\]

\[
j = 1, 2, ..., N, \quad i = 0, 1, ..., M
\]  (39)

Step 7 結合強度の更新

\[
W_{ji}(n+1) = W_{ji}(n) + \eta \delta_j O_i + \alpha \Delta W_{ji}(n)
\]

ここに，\( j = 1, 2, ..., N, \quad i = 0, 1, ..., M \)

\[
W_{kj}(n+1) = W_{kj}(n) + \eta \delta_k O_j + \alpha \Delta W_{kj}(n)
\]

ここに，\( k = 1, 2, 3, \quad j = 0, 1, ..., N \)である。

Step 8 終了判定

誤差 \( E(n) \) をおいて

\[
\sum_{m=0}^{n} E(m+1) \leq \epsilon
\]  (42)

ならば学習を終了し，それ以外ならStep 2へ戻る。

上記の学習アルゴリズムで得られたPIDゲインを元にして，未学習データに対してはStep 8の終了判定を除いた形で同様にオンライン学習を行う。

2.4 システムオロビアンの近似法

以上で述べたニューラルPIDアルゴリズムで，システムオロビアン \( J(n) \) の決定法について述べる。これは，Ziegler-Nichols法において，ステップ応答を求めて，未知システムの動特性を推定したことと同じ意味合いを有している。簡単な近似法は，次式

\[
J(n) = \left| \frac{\partial \hat{y}(n+1)}{\partial u(n)} \right| \text{sgn} \left( \frac{\partial \hat{y}(n+1)}{\partial u(n)} \right)
\]

\[
\text{sgn}(x) = \begin{cases} +1 \quad (x > 0) \\ 0 \quad (x = 0) \\ -1 \quad (x < 0) \end{cases}
\]  (43)

において，絶対値部分を勾配法の学習パラメータ \( \eta \) の一部と見なすことができる。したがって，何らかの方法で，入力に対するプラント出力が増加するか減少するかを判断し，システムオロビアンをその符号のみとすることで
ある。
そこで、実際に稼働しているプラントに対して、このような推定実験を行う場合には、この推定実験によっ
tて、システムヤコピアン \( J(n) \) を近似することが可能である。
しかし保守的な立場からするとこのような実験は好ましくない。そこで、以下のニューラルネットワ
ークを用いてニューロエミュレータを構成する。ニューラルネットワークを用いてプラントのエミュレーションを
行う原理図を第5図に示す。ニューラルネットワークは階層型ニューラルネットワークとする。

第5図 ニューロエミュレータの構成図

このとき、以下の式を用いることによって、ニューラルネットワークによるシステム同定結果からシステムヤ
コピアンを決定することが可能である。換言すれば、事前に出力されるプラントの入力出力と、プラントを
模擬するニューロエミュレータを構築し、その入出力を使って、上式でシステムヤコピアンが求められる。
これはオフライン操作であるが、ニューロエミュレータを推定することによって、オンラインでシステムヤコピ
アンを変更することができる。

ニューラルネットワークの入力は \( u(n), u(n-1), \ldots, u(n-m) \), 出力は \( y(n+1) \) とする。ニューラルネットワークの学習が終了した時点で、システムヤコピアン
\( J(n) \) を次式で近似する。

\[
J(n) \approx \frac{\partial \hat{y}(n+1)}{\partial u(n)} = \sum_{j=1}^{N} W_{kj} O_j (1 - O_j) W_{j1}
\] (45)

ただし、上式で \( W_{j1} \) は \( u(n) \) が入力される入力層のニューロンが1番目に対応していることを示している。
もしプラント特性の概略を表す数式モデルがわかっていていれば、第6図に示すように数式モデルとニューラルネットワークを
併用し、概略モデルの出力 \( \hat{y}_m \) を差し引いた出力誤差を小さくするようにニューラルネットワークの
学習を行うのが得策である。この場合、システムヤコピアンは次式で近似する。

\[
J(n) \approx \frac{\partial \hat{y}(n+1)}{\partial u(n)} = \frac{\partial \hat{y}_m}{\partial u(n)} + \sum_{j=1}^{N} W_{kj} O_j (1 - O_j) W_{j1}
\] (46)

第6図 数式モデルを併用したニューロエミュレータの構成

3. ニューロ PID の応用例

以下では、これまでに著者らが適用してきたニューロ PID 制御例の概略について述べる。

3.1 プロセス制御系

化学プラントにおける温度制御や圧力制御のようなプロセス制御では、定常制御が多く、しかもプラントパラメータが未知のシステムが多い。したがって、数式モデルが陽の形で表現できないことがある。そこで、ステップ応答に基づいたPID制御が多用されている。そこで、ステップ応答のみならず様々な入力に対応したプラント出力を求めるためにニューロエミュレータを構築し、上記のニューロ PID 制御でいくつかの実験を行った。

3.1.1 プロセス温度制御

実験室規模の小さな恒温槽の温度制御実験について述べる。恒温槽の大きさは幅25cm×奥行29cm×高さ20cmで、ヒーターは1.3kW、温度測定器はK型熱電対である。この制御によると制御入力の変動も小さくしかも制御結果も従来のZiegler-Nichols法よりも速適応性がよく、ノイズにも強いことが示されている [4, 5]。

3.1.2 加熱炉温度制御

前節の制御実験をより複雑な温度制御問題に適用するために、プラストチャ爆射出形器に用いられている加熱
炉の温度制御実験を行った。この装置では材料投入口から
射出形成器の入口までが三つのセクタに分けられている。
各セクタには加熱装置（ヒータ）が設置され、各
セクタには望ましい目標温度が設定され、出口に近づく
につれて高温設定されている。また、セクタ間では密封が行われている。ニューロ PID 制御は、セ
ルフチューニング制御、従来のZiegler-Nichols法と比較
して、優れた制御結果を示し、しかも制御入力の変化が
緩やか中毒となっている [4, 5]。

3.1.3 バッチプロセスの反応釜温度制御

ポリエステル、プラスチック樹脂等の製品を生産するバッ
チプロセスの反応釜の温度制御に、ニューロ PID 制御を行った。これは、温度制御用パルプ特性が反応釜内の
複雑な化学反応などによる非線形特性の強い制御系であ
り、PIDゲインのチューニングが困難な問題である。し
たがって、数式モデルによる解析が困難であるため、熟練したオペレータの経験によってPIDゲインの調節が行われている。このようなオペレータによる制御結果と比較して、ニューロPID制御は、設定値との偏差が少なく、応答の時間遅れが小さい制御結果となっている[7]。

3.2 サーボ制御系
時間的に変化する目標値に追随するサーボ制御系に対してPID制御は幅広く使用されている。以下は、著者らが適用してきた以下の三つのテーマについて述べる。

3.2.1 電気自動車速度制御
電気自動車のトルク制御へ適用した結果について述べる。この電気自動車はPIVOTとよばれ、国産総合研究所が1990年に開発したものである。これは、車輪がモータの回転子となっているため、並進と回転がスムーズに行えるものである。路上走行が許可されないため、この自動車のシミュレータを作成し、様々な負荷に対するトルク制御や速度制御実験を行うことができるようになっている。このシミュレータには、PID制御が装置され、トルク制御や速度制御を行っている。このシミュレータを製作した電気メーカーから、制御系に対するPIDゲインの推奨値が付されている。ニューロPIDはかなり広範囲の初期値から出荷しても、ほぼ同じようなパラメータ値の近くに収束し、パラメータ値よりも優れた制御結果を与えることが示された[5]。

3.2.2 倒立振子の安定化制御
倒立振子の安定化に関しては、様々な制御方式による実験が行われている。ニューロPID制御結果は、制御対象である倒立振子の安定化の精度およびそれを用いたシステムやコンピュータの精度に関して、理論値に近い値になっていた。また、ニューロPID制御は目標位置に倒立振子を安定できるとともに、様々な外乱にロバストな制御結果となっている。とくに、他の手法と比べて、システムパラメータが未知でも制御可能であること、および振子の長さや重さが変わっても同じ手順で自動的にPIDパラメータをチューニングできる特徴がある[8]。

3.2.3 ハードディスクドライブ位置制御
ハードディスクドライブにおける磁気ヘッドの位置制御は、ハードディスクの小型化および高密度化に伴ってその重要度が高まっており、これまでに、様々なフィードバック制御手法が提案されている。著者らはハードディスクドライブのヘッドを目標とするトラック位置上に正確に追随させ、しかも機械的共振を抑えるために開ループゲインを数kHz近くでかなり小さくするという制御規範を満たすニューロPID制御を行い、良好な制御結果を得ている[9]。

4. おわりに
以上のように、ニューロPID制御は様々な制御対象に対して適用でき、しかもPIDゲインの自動チューニングを行う方法で、制御結果も良好なものになっている。また、2自由度PID制御系に対しても、同様にニューロPID制御を適用することが可能である。もちろん、これらの制御結果を得るには、制御対象に関する様々なモデル化や制御手法に関する知識に基づいて、ニューラルネットワークへの入力を決定することが必要である。したがって、すべてが自動化できるという目的にはまだほどが遠く、熟練した現場の技術者の経験や知見を活用してニューロPIDコントローラを設計すれば、比較的簡単な制御に対しても出力の望ましい値に近づけることができる。

(2006年8月1日受理)

参考文献
[1] 中野（編著）：ニューロコンピュータの基礎、コロナ社（1990）
[3] 馬場、小島、小澤：ニューラルネットワークの基礎、共立出版（1994）
[4] 大松、山本（編）：セルフチューニングコントロール、計測自動制御学会（1996）

著者略歴
藤中 達（正会員）
1956年10月生、1984年京都大学工学研究科博士後期課程研究指導認定研究員、1986年京都大学工学部助手、1990年大阪府立大学工学部講師、2001年助教授、インテリジェント制御などの研究に従事、工学博士、計測自動制御学会などの会員。

大松 愛（正会員）
1946年12月生、1974年大阪府立大学大学院工学研究科博士課程修了、同年同大工学部助手、1988年同教授、1995年大阪大学分子工学部教授となり現在に至る。ニューラルネットワークなどの研究に従事、工学博士、計測自動制御学会、電気学会、SIAM、IEEEなどの会員。