RBF ネットワーク出力層の個別評価による PID パラメータチューニングの高速化

南山大学　石黒裕司, 高見勲

1 はじめに

ニューラルネットワークとは, 人間の脳の構造を工学的に真似たもので, ニューロンという非線形素子を多数連結した回路網であり, 学習可能なシステムとして注目を集めている．ニューラルネットワークは関数近似やパターン認識, 制御などに幅広い貢献をしている．

今日まで, 階層型ニューラルネットワークを用いた PID パラメータチューニングが行われてきた．非線形性が強く, 数式モデルの記述が困難である制御対象に対して学習機能を用いて柔軟に PID パラメータを求めていくことは有効な方法であると考えることが出来る [1]〜[3]. この手法は制御対象の入力と出力をニューラルネットワークの入力とし, 学習機能を用いて PID パラメータの二乗誤差が小さくなるように PID パラメータを逐次学習していくもの (以下, ニューロ PID と呼ぶ) である．しかし, ニューロ PID では収束の速さや一回の学習に時間が掛かることに問題を有していた．そのためニューラルネットワークの構造 (中間層数やニューラルネットワークの初期値) についても研究が行われてきた [4]〜[8].

ニューロ PID での問題点を解決するために, 本研究では RBF ネットワークを用いる (以下, RBFPID と呼ぶ). RBF ネットワークは, 階層型ニューラルネットワークの中間層に用いられるシグモイド関数の代わりに RBF (ラジアル基底関数) を用いたもので, シグモイド関数を用いたニューラルネットワークより収束が早いとされている [9][10].

ニューロ PID では PID パラメータを二乗誤差の和で評価してきた．それにより各 PID パラメータが一度に評価を満たすまで学習が行われる．PID パラメータは学習過程で, あるパラメータは一定の値に収束しているが, 別のパラメータは収束せずに学習を続けていくという事態に陥ることが多々ある．それによって, すでに収束したパラメータに対しほとんど変化のない学習を続け, 無駄な計算を行い続けることになる．その問題を解決するために PID パラメータの特徴に注目し, PID パラメータの評価を個々に行う．すでに一定の値に収束したパラメータは学習を終えるようにすることで計算量の削減を図る．

本研究では磁気浮上装置を対象として PID パラメータチューニングを行った．

2 RBF ネットワーク

RBF(ラジアル基底関数) ネットワークは, 非線形関数を円形の等高線を持つ基底関数で展開する方法であり, 関数近似に利用されるが, パターン識別法として利用することも可能である．階層型ニューラルネットワークと比較して, 優れた点が指摘できる．それは, 中間層素子のパラメータと重みのパラメータを別々に学習可能なことである．しかし, 一般的に RBF ネットワークはより多くの中間層素子が必要とされる．図 1 に RBF の応答図を示す．c は中心値, σ は幅である．

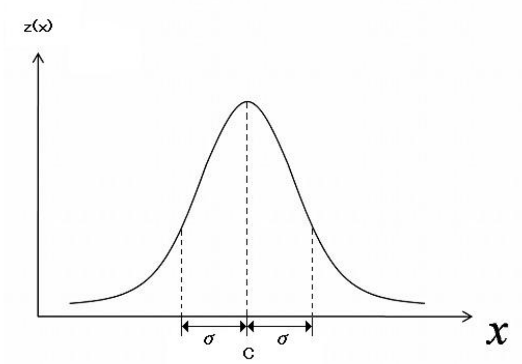

図 1 RBF の応答図

3 RBFPID

3.1 RBF ネットワークによる PID パラメータの出力

図 2 に RBFPID の概略図を示す．r は目標値, y は制御対象の出力, u は操作量, e は r と y の差であり偏差を表している．ここで RBF への入力は u と y であり, RBF の出力は PID ゲイン, K_p, K_i, K_d の差分の 3 つである．

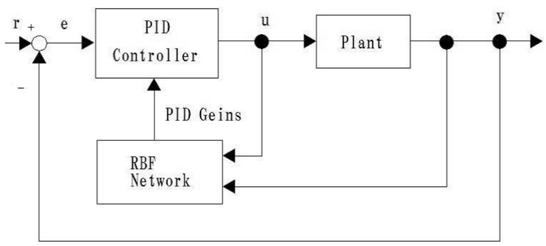

図 2 RBFPID の概略図

図 3 に RBFPID に用いられる RBF ネットワークの構造を示す．このとき, RBF ネットワークへの入力は操作量 $u(n)$ と制御対象の出力 $y(n)$ の 2 つである．そして本研究では RBF ネットワークの精度を増すため, それぞれの中間層に対して学習を行うように RBF の幅 σ_j を中間層素子のパラメータとして与える．また, それぞれの中間層素子からの出力を $z_j(n)$ とする．隔離時間ごとの u, y と RBF の中心値 c_{j1}, c_{j2} でそれぞれの差をとり, ノルム計算を行う．つまり $y(n) - c_{j1}$ と $u(n) - c_{j2}$ のノルムをとり, 各中間層への入力となる．そこで, 0〜1 の値で中間層の出力となる．各中間層の出力に対して, 結合強度 a_{kj} を掛け合わせたものの和 $O_k(n)$ が RBF ネットワークの出力となる．学習するパラメータは中間層素子数が J 個とすると, $c_{ji}, \sigma_j (i = 1, 2.j = 1, 2, ..., J)$ である．このとき RBF ネットワークの出力の数は PID パラメータの 3

つであるため, $a_{kj}(k=1,2,3)$ である. $z_j(n), O_k(n)$ は式 (1)(2) の数式である. また, $O_k(n)$ を式 (11) のように, 与えられた PID パラメータに足していくことで学習の精度を上げる. K_k は $K_1=K_p, K_2=K_i, K_3=K_d$ である. また, n は時刻 0〜T までを離散時間 Δt ごとに区切ったデータ番号であり, $n=1,2,\cdots,N(=\frac{T}{\Delta t})$ である.

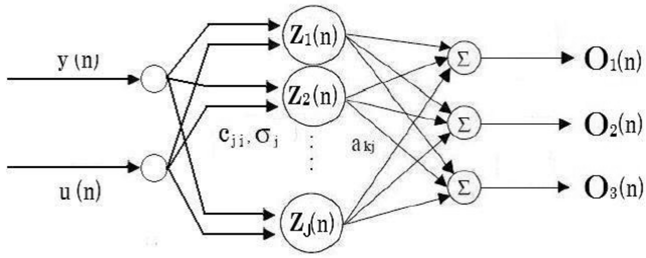

図 3 RBF ネットワークの構造

RBFPID では, 制御対象の応答を図 4 のように得た後, 制御対象の入出力を図 5 のように Δt 秒間隔で N 個のデータとして区切る.

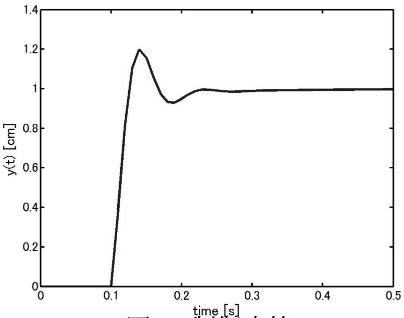

図 4 制御応答

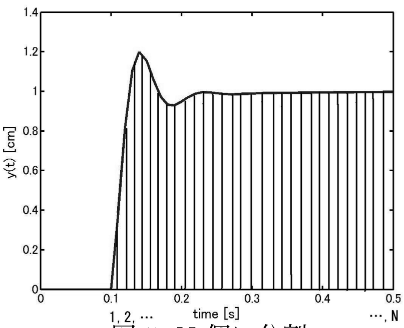

図 5 N 個に分割

その後, Δt 秒ごとに, 各データを RBF ネットワークに入力し, 式 (1), (2) の計算を行う. そして, 式 (3)〜式 (10) で RBF のパラメータを更新し, PID パラメータを式 (11) に従い更新する. それを $n=1,2,...,N$ まで行う. N までパラメータを更新した後の PID パラメータを用いて再び制御対象のシミュレーションによるステップ応答を見る. 応答の二乗誤差の和が一定の値以下の値ならば学習は終了し, そうでなければ上記の計算を繰り返す.

$$z_j(n) = exp\left(-\frac{\sqrt{(y(n)-c_{j1})^2}^2+\sqrt{(u(n)-c_{j2})^2}^2}{\sigma_j^2}\right) \quad (1)$$

$$O_k(n) = \sum_{j=1}^{J} a_{kj} z_j(n)(k=1,2,3) \quad (2)$$

学習には勾配法を用いる. 次式が更新則である. 勾配法を用いるので N 回更新する. λ_c は更新の度合いを決めるパラメータである.

$$c_{ji}(n+1) = c_{ji}(n) - \lambda_c \Delta c_{ji}(n) \quad (3)$$

ここで, Δc_{ji} は

$$\Delta c_{ji} = \frac{\partial E(n)}{\partial c_{ji}} \quad (4)$$

$$E(n) = \frac{1}{2}e(n)^2 \quad (5)$$

である. このとき Δc_{ji} は微分の連鎖式を用いて次式のように求める.

$$\frac{\partial E(n)}{\partial c_{ji}} = \frac{\partial E(n)}{\partial y(n)}\frac{\partial y(n)}{\partial u(n)}\sum_{l=1}^{3}\frac{\partial u(n)}{\partial O_l(n)}\frac{\partial O_l(n)}{\partial z_j(n)}\frac{\partial z_j(n)}{\partial c_{ji}} \quad (6)$$

これを計算すると次式のようになる.

$$\begin{aligned}\frac{\partial E(n)}{\partial c_{ji}} &= -2e(n) \times \frac{\partial y(n)}{\partial u(n)} \times \sum_{l=1}^{3}\frac{\partial u(n)}{\partial O_l(n)} \\ &\times a_{lj} \times z_j(n) \times \frac{(y(n)-c_{j1})+(u(n)-c_{j2})}{\sigma_j^2} \quad (7)\end{aligned}$$

このとき $\frac{\partial u(n)}{\partial O_l(n)}$ は次式

$$\begin{aligned}u(n) &= u(n-1) + K_p(e(n)-e(n-1)) \\ &+ K_i e(n) + K_d(e(n) - 2e(n-1) + e(n-2)) \quad (8)\end{aligned}$$

を PID ゲインで微分したものであるので, 次式になる.

$$\frac{\partial u(n)}{\partial O_k(n)} = \begin{cases} e(n)-e(n-1) & (k=1) \\ e(n) & (k=2) \\ e(n)-2e(n-1)+e(n-2) & (k=3) \end{cases} \quad (9)$$

$\frac{\partial y(n)}{\partial u(n)}$ は制御対象の入力に関する制御対象の出力の勾配であるから, 次式になる.

$$\frac{\partial y(n)}{\partial u(n)} \simeq \frac{y(n)-y(n-1)}{u(n)-u(n-1)} \quad (10)$$

σ_j, a_{kj} も同様の計算を行い, 更新させる.
また, PID パラメータは次式で更新される.

$$K_k = K_k + O_k(n) \quad (11)$$

3.2 RBFPID の手順

勾配を用いた学習により PID パラメータを求める手順を示す.

Step1 初期値設定

RBF のパラメータ c_{ji}, σ_j, a_{kj}, PID パラメータ K_p, K_i, K_d の初期値を設定する.

Step2 制御対象の入出力の読み込み

制御対象への入力 $u(n)$, 制御対象の出力 $y(n)$ の読み込み.

Step3 RBF の出力計算

$z_j(n), O_k(n)$ を求め, c_{ji}, σ_j, a_{kj} を更新させる. $O_k(n)$ を足して PID パラメータを更新する. これを $n=N$ まで続ける.

Step4 PID 制御による制御対象入力と出力の計算

Step3 で求めた PID パラメータで制御対象の入出力を求める.

Step5　誤差の計算と評価

　Step4 で求めた出力から誤差を計算し, RBFPID の評価を行う. 一般的に評価式は次式となる.

$$E_e = \frac{1}{2}\sum_{h=1}^{N} e(h)^2 \tag{12}$$

一定の値以下ならば, 学習終了. そうでなければ, Step2 へ戻る.

4　RBF ネットワークの計算削減

4.1　中間層素子数の削減と結合

中間層素子数が多ければ多いほど学習の精度が増していく. しかし, 中間層素子数が多すぎるとネットワークの計算に多くの時間が掛かったり, 学習に悪影響を及ぼす可能性がある. そこで最適な中間層素子数にするために中間層素子数の削減と結合を行う. 2 つの RBF の中心値 c が 近い値ならばその中間層素子同士は結合させる. また, 中間層素子の出力が低いものは出力層に与える影響は少ないと判断し, 削減する. 結合, 削減の度合い, 結合後のパラメータは次式のようにした. α は削減を行う基準値, β は結合を行う基準値である.

$$削減：\frac{1}{N}\sum_{d=1}^{N} z_j(d) < \alpha \tag{13}$$

$$結合：\sqrt{(c_{p1}-c_{q1})^2 + (c_{p2}-c_{q2})^2} < \beta \tag{14}$$

$$c(new) = \frac{c(old1)+c(old2)}{2} \tag{15}$$

$$\sigma(new) = \sigma(old1) + \sigma(old2) \tag{16}$$

$$a(new) = a(old1) + a(old2) \tag{17}$$

5　出力層の個別評価

RBFPID では, ある程度学習を行うと PID パラメータの中の一つが, ある一定の値に収束し, 他の PID パラメータは変化を続けるということが起こる. これは, 二乗誤差の和を一定数以下にするために学習を続ける RBFPID において「一つのパラメータのみが収束しても, 二乗誤差の和が一定の値以下になっていなければ全てのパラメータの学習を続ける」という習性のために見られる. そこで PID パラメータが制御対象に効果を及ぼす範囲がそれぞれ異なることに注目して, 学習される PID パラメータを個別に評価させていく.

PID パラメータチューニングにおいて比例ゲイン，積分ゲイン，微分ゲインは制御対象の応答で着目する時間帯が各々異なる. 比例ゲインでは全区間に着目，積分ゲインでは定常偏差に関する区間に着目，微分ゲインではオーバーシュートに関わる区間に着目する. つまり, 積分ゲインは定常偏差に影響を与え, 微分ゲインはオーバーシュートに影響を与える. そこで, 制御対象の誤差全体ではなく各ゲインの影響が強い時間帯の誤差で評価を行い無駄な学習を抑える.

図 6 にオーバーシュートが見られる場合の評価区分を示す. K_p の評価には制御対象の全体の絶対値誤差の和を評価とする. K_i の評価には制御対象の出力がピークに達した時刻から, 応答時間の最後までの絶対値誤差の和を評価とする. K_d の評価には制御対象にステップ入力を入れた時刻から, 制御体調の出力がピークに達した時刻までの絶対値誤差の和を評価とする. 各ゲインの評価式は次式に示す. t_s は制御対象にステップ入力を入れた時刻, t_p は制御対象の出力がピークに達した時刻, ε_p は K_p の評価基準, ε_i は K_i の評価基準, ε_d は K_d の評価基準である.

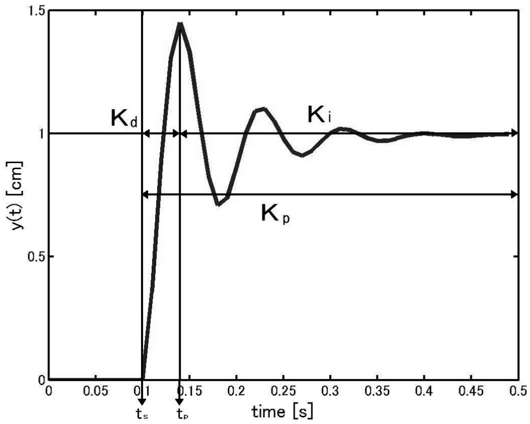

図 6　オーバーシュートがある場合の評価区分

$$K_p の評価式：\sum_{h=t_s}^{N} |e(h)| < \varepsilon_p \tag{18}$$

$$K_i の評価式：\sum_{h=t_p}^{N} |e(h)| < \varepsilon_i \tag{19}$$

$$K_d の評価式：\sum_{h=t_s}^{t_p} |e(h)| < \varepsilon_d \tag{20}$$

オーバーシュートが見られない場合の評価区分は, K_p は式 (18) とする. K_d は出力のピークが定常偏差になるので $t_p = N$ とする. よって, K_d の式は式 (18) と同じとする. K_i の評価は, 定常偏差の絶対値, 式 (21) とする.

$$K_i の評価式：|e(N)| < \varepsilon_i \tag{21}$$

PID パラメータを個別に評価することで RBFPID の無駄な学習を削減させる.

6　制御対象

制御対象に磁気浮上装置を用いる. 図 7 に磁気浮上装置を示す. 伝達関数は次式となる.

図 7　磁気浮上装置

$$G(s) = \frac{3.454}{s^2 + 6.275s + 384.3} \tag{22}$$

7 シミュレーション・実験結果

磁気浮上装置を用いて理論の実証を行う. RBF ネットワークのパラメータ初期値を設定する. 磁気浮上装置の目標値は磁石を $1cm$ 浮かせることとしたので, 磁気浮上装置の出力は 1 に近い数値が出されると推測出来る. よって, c_{ji} は $(0,0)$ ∼ $(1,1)$ を 0.5 の間隔で値を取った. 中間層素子数の数は 9 個とした. σ_j は, 大きすぎると中間層の出力が小さくなるので 2 で統一した. a_{kj} は, 中間層は 0∼1 の間の出力となるので小さすぎると出力層の出力が小さくなり, その逆は大きくなるのであまり大きくない値 10 で統一した. パラメータの更新度合いを決める $\lambda_c, \lambda_\sigma, \lambda_a$ はシミュレーションを繰り返した結果から $\lambda_c = 30, \lambda_\sigma = 30, \lambda_a = 60$ とした. 出力層の評価を決める値 $\varepsilon_p, \varepsilon_i, \varepsilon_d$ はそれぞれ $1.5, 0.5, 1.3$ とした. PID パラメータの値はジーグラニコラス法より $K_p = 598.51, K_i = 4202.2, K_d = 7.1382$ とした.

図 8 にシミュレーション結果の推移を示す. 横軸は時間, 縦軸は出力を示している. 学習回数は 100 回とした. 図 8 から制御対象の応答は学習を繰り返すたびに改善されていることがわかる. 学習を 100 回行う前に, 40 回の学習で評価式を満たした.

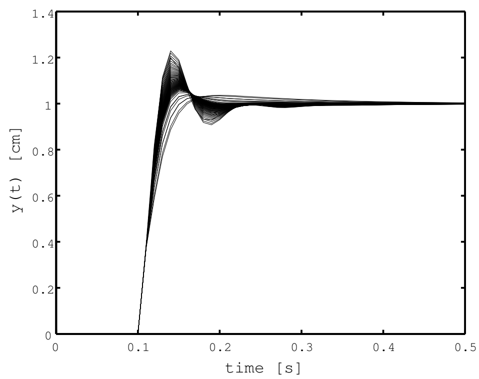

図 8 シミュレーション推移

図 9, 10, 11 にシミュレーションによる各 PID パラメータの推移を示す. 横軸は学習回数, 縦軸はそれぞれ K_p, K_i, K_d の値である. 実線は RBF による学習, 点線はシグモイド関数による学習の推移である. 図より, シグモイド関数より RBF のほうが学習精度が高いことが解る. 図 10 より K_i が 36 回の時に一定の値に収束した. 次に K_d が 39 回目の学習の時に一定の値に収束した. そして, 40 回目の学習で K_p の値が一定に収束した. 学習後の PID パラメータは $K_p = 574.88, K_i = 4173.5, K_d = 11.302$ となった.

図 12 に実験結果の推移を示す. 横軸は時間, 縦軸は制御対象の出力である. 図より, オーバーシュートが見られた初期の波形から, 学習するごとにそれが抑えられ, 制御対象の応答反応が改善されているのがわかる.

図 13, 14, 15 に実験による各 PID パラメータの推移をを示す. 横軸は学習回数, 縦軸は K_p, K_i, K_d である. 図より, K_i の値が学習回数が 35 回目のときに一定の値に収束している. K_d の値も学習回数が 36 回目のときに一

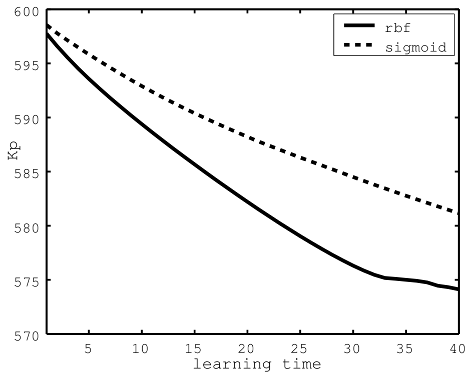

図 9 Kp の値の推移

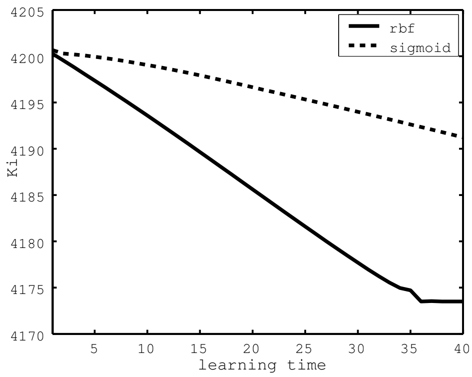

図 10 Ki の値の推移

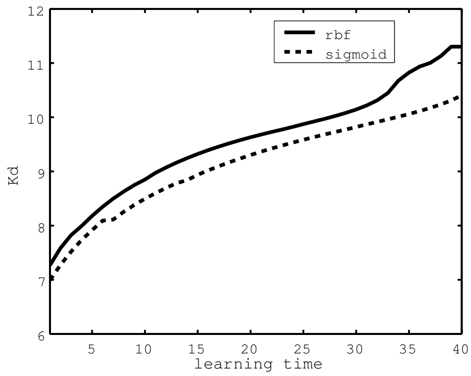

図 11 Kd の値の推移

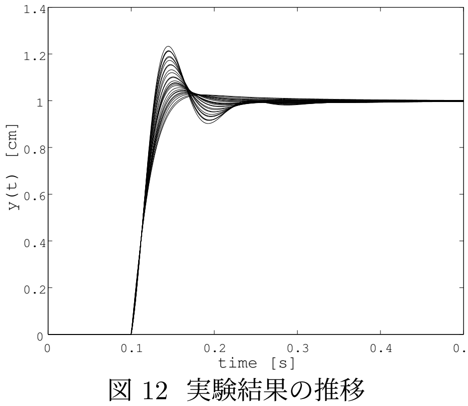

図 12 実験結果の推移

定の値に収束している．その後, 39 回目の学習で K_p も評価式を満たし, RBF は学習を終えた．これより出力層が個別に評価されており, 学習に掛かる無駄な学習が削減されていることがわかる．また, 学習後の PID パラメータは $K_p = 574.85, K_i = 4174.7, K_d = 11.203$ となった．

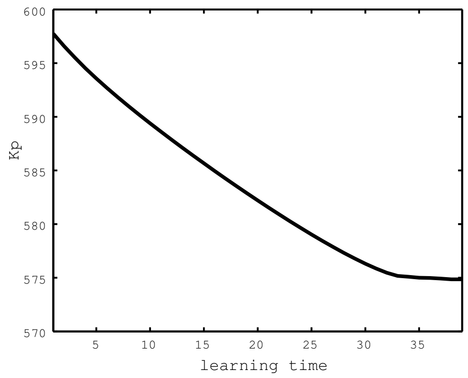

図 13　Kp の値の学習の推移

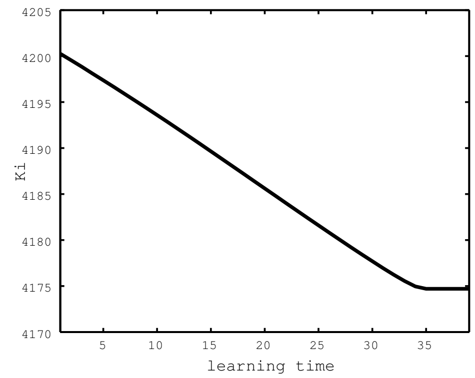

図 14　Ki の値の学習の推移

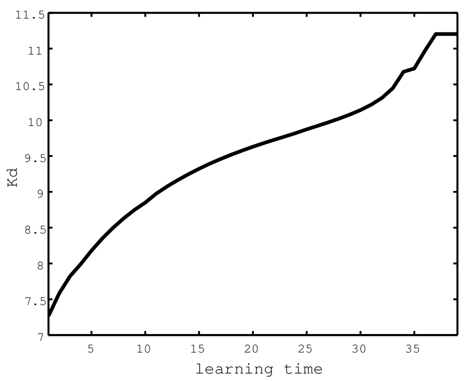

図 15　Kd の値の学習の推移

8　終わりに

本研究では RBF ネットワークを用いて PID 制御系の学習を行った．RBF ネットワークの勾配法を用いた学習法により制御対象の出力結果を改善した．また, 無駄な学習の削減のため出力層の評価を個別に行った．PID パラメータが制御対象の出力に与える影響に着目し, PID パラメータの評価を二乗誤差の和だけで行うのではなく, PID パラメータが影響を及ぼす範囲の絶対値誤差で評価を行った．それにより, すでに評価式を満たした出力層は学習をせず, 評価式を満たしていない出力層は学習を続けさせることで, RBFPID を 39 回の学習で終えることが出来た．また, 磁気浮上装置を用いてシミュレーション, 実験を行い, 出力結果を改善させることが出来た．

参考文献

[1] Robert S. Scalero, Nazif Tepedelenlioglu: A Fast New Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Signal Procrssing, **40**-1, 202/210, (1992)

[2] Youshen Xia,Gang Feng: On Convergence Rate of Projection Neural Networks. IEEE Transaction on Automatic Control, **49**-1, 91/96, (2004)

[3] 森田譲, 前田保憲, 日隈崇文: ニューラルネットワークによる倒立振子制御における PID ゲインのセルフチューニング．日本知能情報ファジィ学会誌, **16**-3, 262/270 (2004)

[4] 宮嶋照行, 長谷川孝明, 羽石操: ニューラルネットワークの高速学習アルゴリズムとその適応等化器への応用. 電子情報通信学会論文誌 A, **J76-A**-8, 1136/1143, (1993)

[5] 塩谷和敏, 吉川敏則: シグモイド関数の導関数変更による誤差逆伝搬学習の高速化. 電子情報通信学会総合大会講演論文集, **1**, 26, (1995)

[6] 石田和子, 亀山啓輔, 小杉幸音: RBF を含むニューラルネット中間層の適応的構成法. 電子情報通信学会技術研究報告, **95**-506, 83/90 (1996)

[7] 鈴木賢治, 原直子, 堀場勇夫, 杉江昇, 石川謙: 教師有りニューラルネットのユニット削減手法とニューラルフィルターへの適用. 電子情報通信学会技術研究報告. NC, ニューロコンピューティング ,**96**-430, 71/78 (1996)

[8] 稲葉寿久: RBF ネットワークの最適化による PID パラメータチューニング．2007 年度南山大学大学院修士論文, (2008)

[9] 八野智博, 松本雅裕, 高田等: RBF ネットワークと免疫的アルゴリズムによる非線形システムのオンライン同定法. 鹿児島大学工学部研究報告, **47**, 27/34, (2005)

[10] HACHINO Tomohiro, TAKATA Hitoshi: On-line Identification Method of Continuous-time Nonlinear Systems Using Radial Basis Function Network Model Adjusted by Genetic Algorithm. IEICE transactions on fundamentals of electronics, communications and computer sciences, **E87-A**-9, 2372/2378, (2004)