論　文

制約付き非線形最適化手法 α 制約法によるファジー制御ルールの最適化

高濱　徹行[†]　　阪井　節子[††]

Tunning Fuzzy Control Rules by α Constrained Method which Solves Constrained Nonlinear Optimization Problems

Tetsuyuki TAKAHAMA[†] and Setsuko SAKAI[††]

あらまし　ファジー制御ルールの学習は，目的関数が微分不可能である制約付き非線形最適化問題とみなすことができる．この場合には，目的関数値のみを用いる最適化手法である直接探索法とペナルティ法を組み合せて解くことが多い．ところが，この方法ではペナルティ関数の係数をどのような値にすれば確実に実行可能解が得られるのか，現在得られている解候補がどの程度制約を満足しているのかを把握することが非常に困難である．本研究では，制約の満足度をファジー制約満足度で表現し，通常の大小関係の代わりに制約満足度を考慮した大小関係である α レベル比較を定義し，この比較に基づき最適化を行うことにより，制約付き問題を制約のない問題に変換する α 制約法を提案する．α 制約法を直接探索法である Powell 法に適用することにより，本方法が制約満足度を把握でき，α レベルを 1 にすれば実行可能解が得られる方法であることをいくつかの例により示す．更に，倒立振子ファジー制御ルールの学習に応用することにより本手法の有効性を示す．

キーワード　ファジー制御，制御学習，制約付き最適化，非線形最適化，ファジー制約，Powell 法

1. まえがき

ファジー制御は，専門家の経験的知識をルール化するのに適しているため，地下鉄システムや車の自動変速器の制御など数多くの対象に利用され，成功を収めてきている [1], [2]．しかし，最初から十分な性能を達成するルールを記述することは困難であるため，ルールやメンバシップ関数をチューニングしなければならない．人間が試行錯誤的にチューニングを行うと，非常な労力が必要となるだけでなく，最適なルールが得られているかどうかを判断することが難しいため，チューニングを自動的に行うファジー制御ルールの学習に関する研究が注目されている．

従来の研究としては，強化学習 [3] を利用した研究 [4], [5]，ファジークラシファイアシステム [6] を利用した研究 [7]〜[9]，遺伝的アルゴリズム [10] を利用した研究 [11]〜[13]，非線形最適化手法を用いた研究 [14], [15] などがある．[5] は Bang-Bang 制御のための後件部ファジーラベルの組合せを，[7]〜[9], [11], [13] は後件部あるいは前件部と後件部両方のファジーラベルの組合せを最適化しているが，メンバシップ関数を最適化していないため，それほど精度の高い制御を実現できてはいない．[4] は倒立振子制御ルールのメンバシップ関数を最適化しており，角度制御については比較的精度の高い制御を実現しているが，位置制御については十分ではない．[12] は精度の高い制御を実現しているが，ファジー変数のレンジをダイナミックにスケーリングするファジースケーリング制御 [16] を利用しており，通常のファジー制御ルールのチューニングという観点からは参考にしにくい面がある．

これに対して，非線形最適化手法を用いた研究では，非常に多くの応用例をもつ推論方法である簡略ファジー推論 [17] を採用し，非常に高い精度の制御を実現している．簡略ファジー推論では，ルールの後件部である実数値を非線形最適化手法で最適化することにより，後件部メンバシップ関数を最適化するのと同等の

[†] 広島市立大学情報科学部知能情報システム工学科，広島市
Faculty of Information Sciences, Hiroshima City University, 3–4–1, Ozukahigashi, Asaminami-ku, Hiroshima-shi, 731–3194 Japan
[††] 広島修道大学商学部経営学科，広島市
Faculty of Commercial Sciences, Hiroshima Shudo University, 1–1–1, Ozukahigashi, Asaminami-ku, Hiroshima-shi, 731–3195 Japan

効果が得られるため，精度の高い制御を実現できているのである．この場合，制御学習は，後件部実数値という制御状態に影響を与える制御パラメータの空間において，制御目標値からの偏差を評価する偏差 2 乗面積や制御目標値に到達するまでに要する時間を目的関数とし，各パラメータの許容範囲でそれを最小化する制約付き非線形最適化問題とみなすことができる．

制約付き非線形最適化問題の解法 [18]〜[20] としては，逐次 2 次計画法，射影法，一般縮小勾配法などの効率の良い最適化手法が知られている．しかし，これらの方法は，目的関数の微分可能性を必要とするため，目的関数値が実際の制御実験やシミュレーションによって得られ，目的関数と制御パラメータとの間に明確な関数関係が定義できなかったり，定義できても微分可能性を仮定できないことが多い制御学習に用いることは難しい．そこで，制御学習では，制約付き最適化問題を制約のない最適化問題に変換して解く変換法 [21] がよく利用されている．変換法には，ペナルティ法，バリア法，乗数法などがあるが，制約条件の境界付近に解が存在する場合が多いこと，パラメータが少なく単純で使用しやすいことなどからペナルティ法を採用することが多い [14]．

制約のない非線形最適化問題において，目的関数が微分不可能な場合には，目的関数値のみを用いる直接探索法 [18] を利用する．直接探索法には，共役方向ベクトルを利用する Powell 法 [22]，直交する方向ベクトルを利用する直交方向探索法，図的方法である Simplex 法などがある．制約付きの場合には，直接探索法とペナルティ法を組み合せて解くことになるが，ペナルティ法や乗数法ではペナルティ関数の係数をどのような値にすれば制約を満足する実行可能解が得られるのか，現在得られている解候補がどの程度制約を満足しているのかを知ることが困難であるという問題がある．

そこで本研究では，ファジー数理計画法 [23]〜[26] で用いられているファジー制約を制約の満足度を表現する指標として導入し，制約満足度とよぶことにする．ファジー制約は，制約を完全に満足した場合に 1，満足する度合いが低くなると 0 に近づく関数である．これにより，制約付き最適化問題は，制約満足度を 1 にするという条件下で目的関数を最小化する，すなわち，制約満足度を最大化しつつ，目的関数を最小化するという 2 目的最適化問題とみなすことができる．この 2 目的最適化問題は，制約満足度を優先する辞書式配列法 [27] により解くことが可能である．

次に，本研究では，この辞書式配列法を直接探索法に組み込むために，通常の大小関係の代わりに制約満足度を優先した大小関係である α レベル比較を定義し，通常の比較の代わりに α レベル比較を用いて探索を行うことにより，制約付き問題を制約のない問題に変換して最適解を求める α 制約法を提案する．α 制約法は変換法の一種であり，それ自体に定まったアルゴリズムがあるのではなく，微係数を用いずに比較のみで探索を行う最適化手法，すなわち，直接探索法全般に応用可能な方法である．そこで，α 制約法を直接探索法である Powell 法に適用することにより，本方法が解候補の制約満足度を把握でき，α レベルを 1 にすれば実行可能解を容易に得ることのできる方法であることをいくつかの例により示す．更に，本手法により制御ルールをチューニングし，精度の高いルールを学習可能であることを，倒立振子制御を例題として，計算機シミュレーションにより示す．

2. 制約付き最適化問題

本研究では以下のような不等式制約と等式制約のある最適化問題 (P) を対象とする．

$$
\begin{array}{lll}
\text{(P)} \quad \text{minimize} & f(\boldsymbol{x}) & \\
\quad\ \ \text{subject to} & g_i(\boldsymbol{x}) \leqq 0 & i = 1, \ldots, m \\
& h_j(\boldsymbol{x}) = 0 & j = 1, \ldots, l \\
& \boldsymbol{x} \in \boldsymbol{R}^n &
\end{array}
$$

制約をどの程度満足しているかを表現するために，制約満足度 $\mu(\boldsymbol{x})$ を導入する．制約満足度は，ファジー数理計画法におけるファジー制約と同等のものである．しかし，ファジー数理計画法におけるファジー制約は，本来制約自体にあいまいさが存在する場合に，目的関数に対する満足度を表現するファジー目標と制約に対する満足度を表現するファジー制約を同時に満足する解を求めるためのものであり，あいまいさをもたない問題 (P) を解くために導入されたものではない．

本研究では，制約にあいまいさが存在していない場合を対象とし，制約が満足されている度合いを表現する指標として利用する．ここで，制約満足度 $\mu(\boldsymbol{x})$ は，以下を満足する関数である．

$$
\begin{cases}
\mu(\boldsymbol{x}) = 1, & \text{if } g_i(\boldsymbol{x}) \leqq 0, h_j(\boldsymbol{x}) = 0 \\
& \quad \text{for all } i, j \\
0 \leqq \mu(\boldsymbol{x}) < 1, & \text{otherwise}
\end{cases}
$$

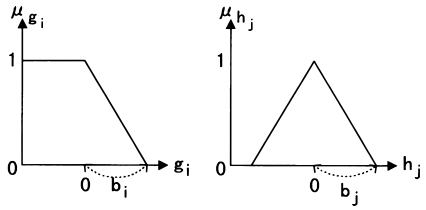

図 1　区分的線形の制約満足度関数
Fig. 1　Piecewise linear function for constraint satisfaction.

このような制約満足度を定義する一つの方法として，各制約の満足度を定義し，それらを合成する方法が考えられる．例えば，問題 (P) における各制約条件は，機械的に以下のような g_i, h_j に関する区分的線形の制約満足度関数に変換できる（図 1 参照）．ただし，b_i, $b_j(>0)$ は適当な定数であり，大きくとることにより広い範囲の初期探索点に対応可能となる．

$$\mu_{g_i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } g_i(\boldsymbol{x}) \leqq 0 \\ 1 - \frac{g_i(\boldsymbol{x})}{b_i}, & \text{if } 0 \leqq g_i(\boldsymbol{x}) \leqq b_i \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_{h_j}(\boldsymbol{x}) = \begin{cases} 1 - \frac{|h_j(\boldsymbol{x})|}{b_j}, & \text{if } |h_j(\boldsymbol{x})| \leqq b_j \\ 0, & \text{otherwise} \end{cases}$$

なお，問題によっては，滑らかな非線形の関数を用いた方が良い場合も考えられる．

この各制約満足度から制約全体の満足度 $\mu(\boldsymbol{x})$ を求めるためには，各制約満足度を結合する必要がある．結合演算としては，以下のような min 演算あるいは代数積演算が適当であると考えられる．

min 演算　　$\mu(\boldsymbol{x}) = \min_{i,j}\{\mu_{g_i}(\boldsymbol{x}), \mu_{h_j}(\boldsymbol{x})\}$
代数積演算　$\mu(\boldsymbol{x}) = \prod_{i,j} \mu_{g_i}(\boldsymbol{x}) \cdot \mu_{h_j}(\boldsymbol{x})$

制約満足度 $\mu(\boldsymbol{x}) = 1$ という条件は明らかに問題 (P) の制約条件と等価であるため，(P) は以下の問題 (P^1) に変換できる．

$$(\mathrm{P}^1) \quad \begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \mu(\boldsymbol{x}) = 1 \end{array}$$

問題 (P^1) について考察する．関数 f の定義域内の任意の点 $\boldsymbol{x}$ は，関数値 $f(\boldsymbol{x})$ と制約満足度 $\mu(\boldsymbol{x})$ という 2 種類の値をもつことになる．この二つ組 $(f(\boldsymbol{x}), \mu(\boldsymbol{x}))$ に着目すると，問題 (P^1) は，μ を 1 に最大化し，f を最小化する方向に解を探索する，すなわち，以下のような 2 目的最適化問題（MP）を解くことになる．

$$(\mathrm{MP}) \quad \begin{array}{ll} \text{maximize} & \mu(\boldsymbol{x}) \\ \text{minimize} & f(\boldsymbol{x}) \end{array}$$

しかし，2 目的最適化問題を直接解くことは困難であること，μ の最大化を優先したいことから，2 目的最適化問題を辞書式配列法により 1 目的最適化問題に変換して解くことが適当であると考えられる．

3. α レベル比較と α 制約法

本研究では，辞書式配列法における比較方法を取り入れ，関数値と制約満足度の組 (f, μ) の集合上での順序関係である α レベル比較を定義する．そして，通常の大小関係の代わりに α レベル比較を用いて最適化を行う α 制約法を提案する．

3.1 α レベル比較

関数値 f と制約満足度 μ の組 $(f(\boldsymbol{x}), \mu(\boldsymbol{x}))$ の集合上の大小関係である α レベル比較を定義する．制約満足度が 1 未満の解は，実行可能解とはならないので，解としての価値が低いと考えられる．そこで，制約満足度を 1 にすることを重視するために，制約満足度の大小関係を優先し，制約満足度が同じ場合は目的関数値の大小関係を考慮するという，大小関係を定義する．点 $\boldsymbol{x}_1, \boldsymbol{x}_2$ における関数値を f_1, f_2，制約満足度を μ_1, μ_2 とすると，通常の大小関係である $<, \leqq$ に対応する関数値と制約満足度の組 (f_i, μ_i) 間の大小関係 $<_1, \leqq_1$ は以下のようになる．

$$(f_1, \mu_1) <_1 (f_2, \mu_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{if } \mu_1 = \mu_2 \\ \mu_1 > \mu_2, & \text{otherwise} \end{cases}$$

$$(f_1, \mu_1) \leqq_1 (f_2, \mu_2) \Leftrightarrow \begin{cases} f_1 \leqq f_2, & \text{if } \mu_1 = \mu_2 \\ \mu_1 > \mu_2, & \text{otherwise} \end{cases}$$

次に，制約満足度の優先の度合いを緩和するために，両方の制約満足度が α 以上の場合は目的関数値の大小関係を優先し，それ以外の場合は制約満足度の大小関係を優先するという α レベル比較 $<_\alpha, \leqq_\alpha$ $(0 \leqq \alpha < 1)$ を定義する．

$$(f_1, \mu_1) <_\alpha (f_2, \mu_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{if } \mu_1, \mu_2 \geqq \alpha \\ f_1 < f_2, & \text{if } \mu_1 = \mu_2 \\ \mu_1 > \mu_2, & \text{otherwise} \end{cases}$$

$$(f_1,\mu_1) \leqq_\alpha (f_2,\mu_2) \Leftrightarrow \begin{cases} f_1 \leqq f_2, & \text{if } \mu_1,\mu_2 \geqq \alpha \\ f_1 \leqq f_2, & \text{if } \mu_1 = \mu_2 \\ \mu_1 > \mu_2, & \text{otherwise} \end{cases}$$

$<_1, \leqq_1$ は $\alpha = 1$ のときの α レベル比較と一致するので，以下では α レベル比較を $0 \leqq \alpha \leqq 1$ の範囲で使用する．なお，$\alpha = 0$ の場合は，制約満足度 $\mu \geqq 0$ という性質のため，以下のように目的関数のみの比較と等価になる．

$$(f_1,\mu_1) <_0 (f_2,\mu_2) \Leftrightarrow f_1 < f_2$$
$$(f_1,\mu_1) \leqq_0 (f_2,\mu_2) \Leftrightarrow f_1 \leqq f_2$$

3.2 α 制約法

α 制約法は，制約付き最適化問題を直接探索法で解く際に，通常の比較の代わりに α レベル比較を用いるものである．Powell 法などの直接探索法では $<, \leqq$ という 2 種類の大小比較を用いるが，この部分を $<_1, \leqq_1$ に置き換えれば，辞書式配列法による最適化が行われることになる．通常の大小比較を α レベル比較に置き換えた最適化問題 $(\mathrm{P}_{\leqq_\alpha})$，すなわち，$\alpha$ 制約法による最適化問題は以下のように定義できる．ただし，$\mathrm{minimize}_{\leqq_\alpha}$ は $\leqq_\alpha$ の意味での最小化である．

$$(\mathrm{P}_{\leqq_\alpha}) \quad \mathrm{minimize}_{\leqq_\alpha} \quad f(\boldsymbol{x})$$

ここで，問題 (P^1) において制約条件を $\mu(\boldsymbol{x}) \geqq \alpha$ に緩和した問題 (P^α) を以下のように定義しておく．

$$(\mathrm{P}^\alpha) \quad \begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \mu(\boldsymbol{x}) \geqq \alpha \end{array}$$

問題 (P^α) と問題 $(\mathrm{P}_{\leqq_\alpha})$，及び問題 (P) に関して以下の定理が成り立つ．

［定理 1］ 問題 (P^1) に最適解が存在するならば，問題 $(\mathrm{P}_{\leqq_\alpha})$ の最適解は問題 (P^α) の最適解である．

証明　(P^1) の最適解を $\boldsymbol{x}^*$ とすると，制約条件を満足するため，$\mu(\boldsymbol{x}^*) = 1.$（$P_{\leqq_\alpha}$）の最適解を $\hat{\boldsymbol{x}}$ とすると，α レベル比較について，より小さい点は存在しないので，$(f(\hat{\boldsymbol{x}}),\mu(\hat{\boldsymbol{x}})) \leqq_\alpha (f(\boldsymbol{x}^*),\mu(\boldsymbol{x}^*))$．$\leqq_\alpha$ の定義と，$\mu(\boldsymbol{x}^*) \geqq \alpha$ より，$\mu(\hat{\boldsymbol{x}}) \geqq \alpha$．したがって，$\hat{\boldsymbol{x}}$ は問題 (P^α) の実行可能解である．

また，$\forall \boldsymbol{x} \in X^\alpha = \{\boldsymbol{x} | \mu(\boldsymbol{x}) \geqq \alpha\}$ に対して $(f(\hat{\boldsymbol{x}}),\mu(\hat{\boldsymbol{x}})) \leqq_\alpha (f(\boldsymbol{x}),\mu(\boldsymbol{x}))$ が成立するため，$\leqq_\alpha$ の定義と $\mu(\hat{\boldsymbol{x}}),\mu(\boldsymbol{x}) \geqq \alpha$ より，$f(\hat{\boldsymbol{x}}) \leqq f(\boldsymbol{x})$．したがって，$\hat{\boldsymbol{x}}$ は，X^α における $f(\boldsymbol{x})$ の最適解，すなわち，問題 (P^α) の最適解である．　□

［定理 2］ 問題 (P) に最適解が存在するならば，問題 $(\mathrm{P}_{\leqq_1})$ の最適解は，問題 (P) の最適解である．

証明　問題 (P) と問題 (P^1) は等価であるため，定理 1 で，$\alpha = 1$ とおけばよい．　□

定理 1，2 により，α レベル比較を行うことにより，制約付き問題がそれと等価な制約のない問題に変換されることが示された．したがって，既存の制約のない最適化手法に α レベル比較を導入することにより，制約付きの問題を解くことが可能となる．

3.3 α の制御

定理 2 より，$\alpha = 1$ として問題 $(\mathrm{P}_{\leqq_\alpha})$ を解くことにより，問題 (P) の最適解が得られるが，最小値の探索を数値的に行う際に，制約を満足する領域が狭いことなどが原因で局所解から移動できなくなる場合がある．特に，等式制約ではこのような状況が起こりやすいと考えられる．このような場合に，ペナルティ法においてペナルティ係数を ∞ まで増加させて行くのと同様に α を 1 まで増加させながら最適化を行うことが有効であると考えられる．このような α の制御を行っても，最適解が得られることを以下に示す．

［定理 3］ $\{\alpha_n\}$ を強い意味で単調増加し 1 に収束する点列とする．$f(\boldsymbol{x}), \mu(\boldsymbol{x})$ を連続関数とし，問題 (P^1) の最適解 $\boldsymbol{x}^*$ の存在と任意の α_n に対する問題 $(\mathrm{P}_{\leqq_{\alpha_n}})$ の最適解 $\hat{\boldsymbol{x}}_n$ の存在を仮定する．このとき，点列 $\{\hat{\boldsymbol{x}}_n\}$ の任意の集積点は問題 (P^1) の最適解である．

証明　定理 1 より，$\hat{\boldsymbol{x}}_n$ は問題 (P^{α_n}) の最適解となるので，$f(\hat{\boldsymbol{x}}_n) \leqq f(\boldsymbol{x}^*)$．

$\{\hat{\boldsymbol{x}}_n\}$ の集積点を $\bar{\boldsymbol{x}}$，$\{\hat{\boldsymbol{x}}_{n_k}\}$ を $\bar{\boldsymbol{x}}$ に収束する部分列とすると，$\lim_{k\to\infty} \hat{\boldsymbol{x}}_{n_k} = \bar{\boldsymbol{x}}$．$f$ の連続性より，$\lim_{k\to\infty} f(\hat{\boldsymbol{x}}_{n_k}) = f(\bar{\boldsymbol{x}})$．したがって，$f(\boldsymbol{x}^*) \geqq \lim_{k\to\infty} f(\hat{\boldsymbol{x}}_{n_k}) = f(\bar{\boldsymbol{x}})$．

これに対して，μ の連続性及び $\lim_{k\to\infty} \alpha_{n_k} = 1$ より，$\mu(\bar{\boldsymbol{x}}) = \lim_{k\to\infty} \mu(\hat{\boldsymbol{x}}_{n_k}) \geqq \lim_{k\to\infty} \alpha_{n_k} = 1.$ $\bar{\boldsymbol{x}}$ は (P^1) の実行可能解となり，$f(\boldsymbol{x}^*) \leqq f(\bar{\boldsymbol{x}})$．

したがって，$f(\bar{\boldsymbol{x}}) = f(\boldsymbol{x}^*)$，すなわち，$\bar{\boldsymbol{x}}$ は (P^1) の最適解である．　□

ペナルティ法においては，ペナルティ係数を初期値 r_0 から $r_n = r_0 c^n$（c は $c > 1$ の定数）のように増加させることが多い．これに対して，α 制約法では，初期値 $\alpha_0 (0 < \alpha_0 < 1)$ から $\alpha_{n+1} = \beta + (1-\beta)\alpha_n$ のように増加させ，1 に収束させる方法が考えられる．$\beta (0 < \beta < 1)$ は収束の速さを制御する定数である．なお，r_0, c と同様に，α_0, β を一般的に決めるのは困難であり，問題によって適切な値を選択する必要がある．

4. α 制約 Powell 法

α 制約法における直接探索法として Powell 法を採用した方法を α 制約 Powell 法とよぶことにする．なお，ここで最適化の対象とする点 $\boldsymbol{x}$ は，関数値と制約満足度という二つの値の組 $(f(\boldsymbol{x}), \mu(\boldsymbol{x}))$ をもつ．

4.1 α 制約 Powell 法のアルゴリズム

Powell 法では，共役方向に直線探索を行うことを繰り返し，関数 $f(\boldsymbol{x})$ の最小値を探索する．直線探索では，ある点 $\boldsymbol{x}$ から方向ベクトル $\boldsymbol{d}$ に沿って 1 次元的に探索する，すなわち，$f(\boldsymbol{x}+a\boldsymbol{d})$ を a に関する関数と解釈して探索を行う．Powell 法に対して α レベル比較を適用したアルゴリズムを Powell$_\alpha$ と名付け，以下に C 言語風に記述する．なお，直線探索$_\alpha$ は，直線探索に対して α レベル比較を適用したものである．

```
Powell_α()
{
    方向ベクトル d_0, ..., d_{n-1} を
      R^n 上の一次独立なベクトルとする
    出発点を x_B^0 とする
    for(k=0; ; k++) {
        x_0 = x_B^k;
        for(i=0; i < n; i++) {
            直線探索 α により，
                a に関する二つ組 (f(x_i + a d_i), μ(x_i + a d_i)) の
                α レベル比較に対する最小値を有する a を求める
            x_{i+1} = x_i + a d_i;
        }
        d_n = x_n - x_0;
        直線探索 α により，
            a に関する二つ組 (f(x_n + a d_n), μ(x_n + a d_n)) の
            α レベル比較に対する最小値を有する a を求める
        x_B^{k+1} = x_n + a d_n;
        if(||x_B^{k+1} - x_B^k|| < ε_powell) break;
        for(i=0; i < n; i++)
            d_i = d_{i+1};
    }
}
```

なお，ϵ_{powell} は探索の精度である．また，方向ベクトルの初期値 $\boldsymbol{d}_0, \cdots, \boldsymbol{d}_{n-1}$ は，座標軸方向の単位ベクトルをとればよい．

4.2 直線探索$_\alpha$

直線探索$_\alpha$ として，囲い込みと黄金分割法を組み合せた方法を採用する．囲い込み，黄金分割法に対して α レベル比較を応用したアルゴリズムをそれぞれ，bracketing$_\alpha$，golden$_\alpha$ と名付け，以下に C 言語風に記述する．

```
直線探索_α()
{
    bracketing_α(0) により，(f(a), μ(a)) の α レベル比較
        に関する極小値が存在する a の区間 [a_1, a_2] を求める
    golden_α(a_1, a_2) により，区間 [a_1, a_2] で (f(a), μ(a)) の
        α レベル比較に関する最小値を求める
}

bracketing_α(a_0)
{
    ステップ幅を h とする
    a_1 = a_0 + h;
    if((f(a_0), μ(a_0)) <_α (f(a_1), μ(a_1))) {
        a_0 と a_1 を交換する
        h = -h;
    }
    while(1) {
        h = h * 2;
        a_2 = a_1 + h;
        if((f(a_2), μ(a_2)) ≦_α (f(a_1), μ(a_1))) {
            a_0 = a_1;
            a_1 = a_2;
        }
        else break;
    }
    if(h >= 0) 区間を [a_0, a_2] とする
               else 区間を [a_2, a_0] とする
}

golden_α(a, b)
{
    x_1 = a + F_1 * (b - a);
    x_2 = a + F_2 * (b - a);
    while(|a - b| >= ε_golden) {
        if((f(x_1), μ(x_1)) <_α (f(x_2), μ(x_2))) {
            b = x_2;
            x_2 = x_1;
            x_1 = a + F_1 * (b - a);
        }
        else {
            a = x_1;
            x_1 = x_2;
            x_2 = a + F_2 * (b - a);
        }
    }
    if((f(x_1), μ(x_1)) ≦_α (f(x_2), μ(x_2)))
        x_1 を解とする
    else
        x_2 を解とする
}
```

なお，F_1, F_2 は，$F_1 = (3-\sqrt{5})/2, F_2 = (\sqrt{5}-1)/2$ となる定数，ϵ_{golden} は探索の精度である．

5. 数 値 実 験

実行可能解を素早く確実に探索できるという，α 制約法のもつ性質を示すため，最も基本的であり，多くの研究で取り上げられている，2 次関数を中心とする簡単な例題を対象に数値実験を行う．実験では，α 制約 Powell 法とペナルティ法，すなわち，目的関数にペナルティ関数を加えた拡張目的関数を Powell 法で最小化する場合とを比較することにより，本研究の有

効性を示す．問題 (P) に対するペナルティ法における拡張目的関数 $F(\boldsymbol{x})$ は以下のように定義する．

$F(\boldsymbol{x}) = f(\boldsymbol{x}) + rP(\boldsymbol{x})$

$P(\boldsymbol{x}) = \sum_{i=1}^{m} (\max\{0, g_i(\boldsymbol{x})\})^p + \sum_{i=1}^{l} |h_j(\boldsymbol{x})|^p$

ここで，$p = 2$ とし，ペナルティ係数を $r = c^n (n = 0, 1, \cdots)$ のように増加させることにより実行可能解を探索する．c の設定により結果が異なるため，$r = 2^n, 5^n, 10^n, 20^n, 50^n, 100^n$ の 6 通りの設定で実験し，最も良い結果が得られた設定と比較する．

α 制約法における制約満足度関数はすべて図 1 の区分的線形関数を採用し，制約満足度の結合には min 演算を用い，$b_i = b_j = 10$ とした．探索の初期点は，比較のためにすべて制約領域外である $(2, 2)$ にとり，探索の精度 $\epsilon_{powell} = \epsilon_{golden} = 10^{-5}$，囲い込みのステップ幅 $h = 1$ とした．なお，実験は最適解との誤差が 10^{-3} 以下となる点が得られるまで行った．

5.1 等式制約を含まない場合

例 1 minimize $(x_1 - 1)^2 + (x_2 - 2)^2$

subject to $x_1^2 + x_2^2 - 2 \leqq 0$

$-x_1 + x_2 \leqq 0, -x_2 \leqq 0$

例 1 は凸計画問題であり，最適解 $\boldsymbol{x}^* = (1, 1)$ は Kuhn-Tucker 条件を満足する比較的易しい問題である．α 制約法では $\alpha = 1$，ペナルティ法では $r = 1, 50, 50^2, 50^3$ とした．目的関数及び制約満足度関数の形状を図 2 に，探索点の変化の様子を図 3 に示す．なお，図 3 で，実線は目的関数の等高線及び制約領域の境界，太い点線は α 制約法の探索点，細い点線はペナルティ法の探索点を示す（以下同様）．

図 3 からわかるように，ペナルティ法では，制約領域外から制約境界上の最適解へと収束しているのに対して，α 制約法では，制約満足度関数が最大となる方向を探索し，$(0, 0)$ に到達した後，制約境界上を移動して最適解へと収束している．

例 2 minimize $\min\{(x_1 - 2)^2 + (x_2 + 1)^2,$

$\frac{1}{2}|x_1 + 2|(x_2 + 2)^2\}$

subject to $(x_1 - 1)^3 + x_2 \leqq 0$

$-x_1 \leqq 0, -x_2 \leqq 0$

例 2 は，最小値を返す関数 min を含む，二つの関数の最小値を目的関数とする微分不可能な例であり，最適解 $\boldsymbol{x}^* = (1, 0)$ は Kuhn-Tucker 条件を満足しない．目的関数は，最小となる関数が切り替わる境界線を実行可能領域中にもつ．α 制約法では $\alpha = 1$ とし，実行可

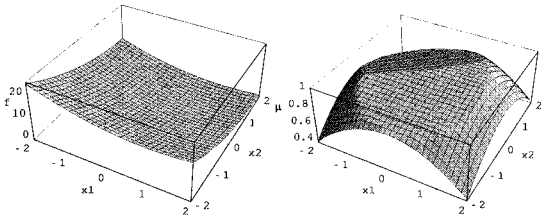

図 2 例 1 の目的関数と制約満足度
Fig. 2 Objective function and μ for example 1.

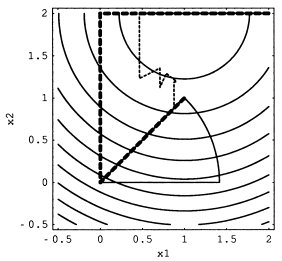

図 3 例 1 における探索点の変化の様子
Fig. 3 Search path for example 1.

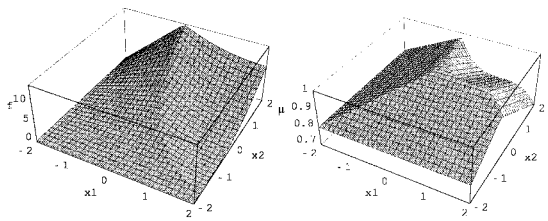

図 4 例 2 の目的関数と制約満足度
Fig. 4 Objective function and μ for example 2.

能解 $(0.999909, 0.000000)$ を得た．ペナルティ法では，ペナルティ係数を 6 通りの方法で変化させたが，いずれの場合も誤差が 10^{-3} 以内の解は探索できなかったため，最も誤差の少ない結果が得られた 50^n の場合を対象に比較する．この場合には，$r = 1, 50, 50^2, \cdots, 50^{11}$ と変化させることにより $(1.002379, 0.000000)$ の解を得たが，実行可能解とはならなかった．目的関数及び制約満足度関数の形状を図 4 に示し，探索点の変化の様子を図 5 に示す．

図 5 からわかるように，ペナルティ法では，目的関数の最小値の影響を大きく受け，制約領域外の最小値の方向へ探索を進めてしまうため，r を非常に大きく取らなければ最適解の方へ収束しない．これに対して α 制約法では例 1 と同様に制約を満足する点を発見

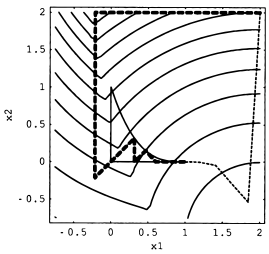

図 5　例 2 における探索点の変化の様子
Fig. 5　Search path for example 2.

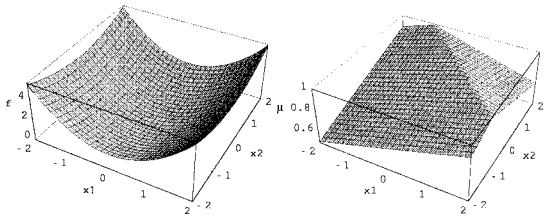

図 6　例 3 の目的関数と制約満足度
Fig. 6　Objective function and μ for example 3.

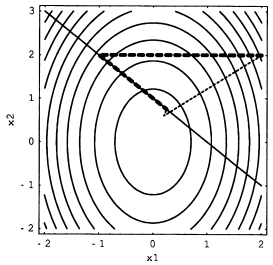

図 7　例 3 における探索点の変化の様子
Fig. 7　Search path for example 3.

し，制約領域内を移動して最適解に素早く収束している．ただし，1 回目の探索（Powell$_\alpha$ の呼び出し）では，方向ベクトル $\boldsymbol{d}$ が縮退したため，$\alpha = 1$ の解を見つけられず，2 回目の探索で最適解に到達した．

5.2　等式制約を含む場合

例 3　minimize　$x_1^2 + \frac{1}{3}x_2^2$
　　subject to　$-x_1 - x_2 + 1 = 0$

例 3 は，線形の等式制約をもつ例であり，最適解 $\boldsymbol{x}^* = (0.25, 0.75)$ である．等式制約の場合，α 制約法では，制約領域が非常に狭いため，一度 $\alpha = 1$ の解が見つかると，完全に制約領域と一致する方向に探索を行う場合にのみ最適解が得られる．残念ながら，Powell 法では方向ベクトルに沿って探索を行うため，このような方向を見出すことが困難であり，$\alpha = 1$ では解を見つけることができない．そこで，制約を少しだけゆるめて $\alpha = 0.9999$ から始めることにより，実行可能解 $(0.250903, 0.749097)$ を得た．ペナルティ法では，20^n の場合に最良の結果となり，$r = 1, 20, 20^2$ と変化させることにより $(0.249844, 0.749532)$ の解を得たが，実行可能解とはならなかった．目的関数及び制約満足度関数の形状を図 6 に，探索点の変化の様子を図 7 に示す．

図 7 からわかるように，ペナルティ法では，初めに目的関数の最小値の方向に進み，ペナルティ係数を大きくすることにより，制約領域の方向へ進んでいる．これに対して，α 制約法は，$\alpha \geqq 0.9999$ を満足する点に到達後，$\alpha \geqq 0.9999$ を満足する狭い帯状の領域を直線的に移動して最適解を見つけている．

5.3　評　　価

いずれの例においても，α 制約法はペナルティ法と同等以上の解を見つけることが可能であり，実行可能解を探索する能力はペナルティ法より優れていた．また，表 1 に直線探索を用いた回数，目的関数を評価した回数，及び参考として CPU 時間を示したが，いずれの値もペナルティ法より優れている．

表 1　ペナルティ法との比較
Table 1　Comparison with penalty method.

		α 制約法	ペナルティ法
例 1	直線探索	5	35
	目的関数	164	1153
	CPU 時間（秒）	3.05×10^{-4}	16.51×10^{-4}
例 2	直線探索	16	114
	目的関数	516	3673
	CPU 時間（秒）	7.35×10^{-4}	59.68×10^{-4}
例 3	直線探索	16	24
	目的関数	533	794
	CPU 時間（秒）	7.19×10^{-4}	7.86×10^{-4}

したがって，α 制約法は，ペナルティ法と同等の探査能力を有すると考えられる．しかも，制約満足度関数は，比較的最大値を探索しやすい関数となっているので，実行可能解を簡単に見つけることが可能であるという特徴も有している．

なお，α 制約法では，α の制御以外に満足度関数 μ に関するパラメータ b_i, b_j の設定が必要となるが，通常は初期探索点が $\mu > 0$ の領域内に来るように十分に大きく設定すればよい．これは，パラメータを変更することによって μ の傾きが変化するが，直接探索法では値の大小関係のみで探索を進めるため，μ の最大値の探索にはほとんど影響しないからである．ただし，

μ の値の変化に伴い，制約満足度の値に対する意味付けや，制約をゆるめて $\alpha < 1$ で探索するときの α の選択に影響するため，できる限り解に対して要求する満足度に応じて適当な大きさに設定すべきである．

6. ファジー制御ルールの学習

α 制約法の応用として，代表的な不安定系であり，制御学習の課題としてしばしば採り上げられている倒立振子のためのファジー制御ルールの学習を取り上げる．

6.1 倒立振子とファジー制御ルール

倒立振子の制御における状態変数は，鉛直方向に対する振子の角度 θ，振子の角速度 ω，台車の中央位置からの距離 x，台車の速度 v である．操作変数は，台車を押す力 F である．制御目標は中央位置で直立すること，すなわち，(θ,x)=(0,0) である．

ファジー制御ルールとしては，以下のように前件部変数を (θ, ω) 及び (x, v) の組合せとし，後件部変数を F としたものを採用する．

if θ is A_i^{θ} and ω is A_j^{ω} then F is $C_{ij}^{\theta\omega}$

if x is A_i^{x} and v is A_j^{v} then F is C_{ij}^{xv}

ファジー推論としては，and を代数積とする簡略ファジー推論を採用し，前件部ファジー変数を，NB, NS, ZO, PS, PB のファジーラベルで表現し，後件部ファジー変数を一つの実数値で表現する．倒立振子の対称性，及び制御目標を達成した場合に $F = 0$ とすべきであることを考慮し，これらのルールを制御ルール表で表現すれば，表 2 のような θ/ω 表と x/v 表となり，学習対象のパラメータは 24 個である [14], [15].

6.2 倒立振子の制御学習

倒立振子における制御目標は，中央位置で直立させることである．したがって，倒立振子の制御ルールの学習は，角度と位置に関する制御誤差を最小化するという 2 目的 24 次元の非線形最適化問題となるが，ここでは正規化し，その和を取ることにより 1 目的化して定義する．制御誤差はシミュレーションあるいは実験により得られるため，直接探索法で解く必要がある．この直接探索法として，α 制約 Powell 法を使用する．

$$
\begin{aligned}
&\text{minimize} && f(\boldsymbol{p}) = f_\theta + f_x \\
&\text{subject to} && |p_i| \leqq F_{\max}, i = 1, 2, \cdots, 24 \\
&\text{where} \quad \boldsymbol{p} &=& (p_1, p_2, \cdots, p_{24}) \\
&\qquad f_\theta &=& \int_0^T |\theta_t| dt / (\theta_{\max} T) \\
&\qquad f_x &=& \int_0^T |x_t| dt / (x_{\max} T)
\end{aligned}
$$

ただし，$F_{\max}$ は台車を押す力の最大値である．$\theta_{\max}$, $x_{\max}$ は，それぞれ角度と位置の最大値であり，f_θ, f_x を [0,1] の範囲に規格化するために使用した．

実験条件は，次のとおりである．倒立振子の台車の質量を 1.0 kg，振子の質量を 0.1 kg，振子の長さを 0.5 m, レールの長さを ± 3 m とする．$F_{\max}$ を 10 N とし，簡単な制御実験の結果から，(θ, ω, x, v) の範囲を（± 15°, ± 120°/s, ± 3.0 m, ± 3.0 m/s）とする．(15, 0, 0, 0), (−15, 0, 3, 0) という二つの初期状態から実験を行い，各目的関数の平均値を対象に最適化を行う．タイムメッシュ 0.02 s で離散化した Euler 法によりシミュレーションを行い，各初期状態から 500 ステップ（10 秒間）行ったところで終了する．

文献 [16] を参考にして，α 制約 Powell 法の初期探索点を表 3 のように設定し，$\epsilon_{powell} = 10^{-3}, \epsilon_{golden} = 10^{-2}$ とした．

6.3 評　　価

表 4 が学習結果として得られた制御ルール表である．目的関数の値は，学習前が $f = 0.35445$ ($f_\theta = 0.14747$, $f_x = 0.20698$)，学習後が $f = 0.19159$ ($f_\theta = 0.09881$, $f_x = 0.09278$) であり，学習前と比べて制

表 2　制御ルール表
Table 2　Table of fuzzy control rule.

θ/ω	NB	NS	ZO	PS	PB
NB	p_1	p_2	p_3	p_4	p_5
NS	p_6	p_7	p_8	p_9	p_{10}
ZO	p_{11}	p_{12}	0.0	$-p_{12}$	$-p_{11}$
PS	$-p_{10}$	$-p_9$	$-p_8$	$-p_7$	$-p_6$
PB	$-p_5$	$-p_4$	$-p_3$	$-p_2$	$-p_1$
x/v	NB	NS	ZO	PS	PB
NB	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}
NS	p_{18}	p_{19}	p_{20}	p_{21}	p_{22}
ZO	p_{23}	p_{24}	0.0	$-p_{24}$	$-p_{23}$
PS	$-p_{22}$	$-p_{21}$	$-p_{20}$	$-p_{19}$	$-p_{18}$
PB	$-p_{17}$	$-p_{16}$	$-p_{15}$	$-p_{14}$	$-p_{13}$

表 3　初期の制御ルール
Table 3　Initial rule table.

θ/ω	NB	NS	ZO	PS	PB
NB	-10	-10	-10	-5	0
NS	-10	-10	-5	0	5
ZO	-10	-5	0	5	10
PS	-5	0	5	10	10
PB	0	5	10	10	10
x/v	NB	NS	ZO	PS	PB
NB	-5	-5	-3	-1	0
NS	-5	-3	-1	0	1
ZO	-3	-1	0	1	3
PS	-1	0	1	3	5
PB	0	1	3	5	5

表 4　学習後の制御ルール
Table 4　Learned rule table.

θ/ω	NB	NS	ZO	PS	PB
NB	8.827	9.109	-10.000	-9.975	0.000
NS	-10.000	-8.397	-4.700	-5.256	5.000
ZO	-10.000	-10.000	0.000	10.000	10.000
PS	-5.000	5.256	4.700	8.397	10.000
PB	-0.000	9.975	10.000	-9.109	-8.827
x/v	NB	NS	ZO	PS	PB
NB	-5.000	-5.000	-1.993	-1.778	0.000
NS	-5.000	-4.333	-0.177	1.156	1.000
ZO	-3.000	-1.836	0.000	1.836	3.000
PS	-1.000	-1.156	0.177	4.333	5.000
PB	-0.000	1.778	1.993	5.000	5.000

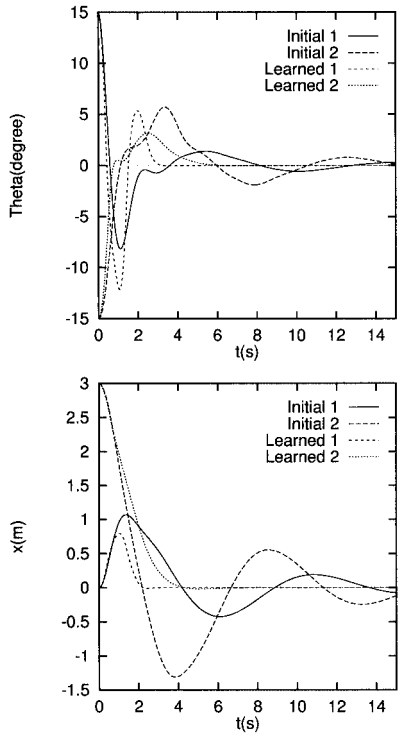

図 8　制 御 結 果
Fig. 8　Control result.

御誤差が半分程度にまで減少している．制御の状態を比較するために，学習前と学習後の制御ルールにより，学習に用いた初期値から制御を行った結果を図 8 に示す．上図が θ，下図が x，Initial が学習前のルール，Learned が学習後のルール，1 が初期値 (15,0,0,0)，2 が初期値 (−15,0,3,0) に対応している．初期探索点である表 3 では，角度及び位置の制御誤差がなかなか 0 に収束せず，ルールの最適化が不十分な状態である．これに対して，学習後の表 4 では，角度と位置の制御誤差が両方とも非常に高速に 0 に収束し，精度の高い制御に成功している．したがって，α 制約法は，制御ルールの学習という，より実際的な問題に対しても有効であることが示された．

表 5　ペナルティ法との比較
Table 5　Comparison with penalty method.

	関数値	評価回数	CPU（秒）	ペナルティ
α 制約法	0.19159	8,489	154.28	0
r 0.1	0.19057	25,728	472.25	5.06×10^{-3}
0.2	0.19031	22,694	415.99	7.95×10^{-4}
0.4	0.19207	10,693	194.96	8.57×10^{-5}
0.6	0.19247	14,960	273.37	3.75×10^{-6}
0.8	0.19181	10,652	194.04	7.07×10^{-6}

α 制約法を評価するために，ペナルティ法との比較を行ったが，ペナルティ法ではペナルティ係数の選択が容易ではなく，比較的良い結果が得られたペナルティ係数 $r = 0.1$ から 0.8 までの 5 通りを選択した．最終的に得られた関数 f の値，関数の評価回数，CPU 時間，及びペナルティ関数 $P(\boldsymbol{x})$ の値について，ペナルティ法と比較した結果を表 5 に示す．なお，数値の部分がペナルティ法であり，数値はペナルティ係数を意味する．また，ペナルティ関数の値が 0 でなければ，実行可能解ではない．

α 制約法では，$\alpha = 1$ の設定で，ペナルティ法と同等の解を，比較的高速に，しかも実行可能領域で探索できている．したがって，実際的な問題についても，α 制約法は，ペナルティ法と同等の能力を有している．

7. む す び

α 制約法は，制約付きの非線形最適化問題に対して，制約満足度を定義し，α レベル比較を導入することにより，制約付き問題を制約のない問題に変換し，制約を満足する解を探索する方法である．いくつかの例でペナルティ法と比較することにより，同等以上の能力をもつ方法であることを示した．更に，ファジー制御ルールの学習という，より実際的な問題に適用することにより，現実的な問題に利用可能な方法であることも示した．

α 制約法は，α 制約法と同様に直接探索法に応用可能な変換法と比較すると，以下のような特徴がある．

- 実行可能解を確実に発見できる

ペナルティ法の場合には，計算の途中で生成される点は一般には実行可能とならないが，α 制約法では，$\alpha = 1$ とすることで，通常目的関数に比べて単純である制約満足度関数の最大化を行うことにより，即座に

制約条件を満足する実行可能解が得られる．

- 広い初期探索点に対応できる

バリア法では，実行可能解から探索を行わなければならないが，α 制約法では，$\mu > 0$ を満足する領域を広く取ることにより，広い範囲の初期値から探索を始めることが可能である．

- 制約条件の境界上の解を見つけやすい

境界上の点を探索するためには，ペナルティ法ではペナルティ関数の係数を大きくして制約条件を満足する方向へ収束させる必要があり，バリア法でも，同様に係数を大きくして制限範囲を境界まで広げる必要があるが，係数が大きくなるにつれ，目的関数値の影響が小さくなり，拡張目的関数の最小化を数値的に行うことが困難になるという問題点がある．これに対して，α 制約法では，制約満足度関数の最大化方向に探索するためこのような点を見つけやすい．

今後は，α 制約法を Powell 法以外の他の直接探索法と組み合せること，より実験回数の少ない高速な制御学習の可能な方法を考案することなどを予定している．

文　献

[1] S. Yasunobu and S. Miyamoto, “Automatic train operation by predictive fuzzy control,” Industrial Applications of Fuzzy Control, North-Holland, pp.1–18, 1985.

[2] 長町三生編，“ファジィ化製品開発の基礎と実際，” 海文堂出版，Sept. 1991.

[3] R.S. Sutton, “Learning to Predict by the Methods of Temporal Differences,” Machine Learning 3, pp.9–44, 1988.

[4] 唐　政，小森雅和，石塚興彦，淡野公一，“増強学習に基づく適応性をもつ ULR ファジーコントローラ，” 信学論（A），vol.J78-A, no.8, pp.1051–1058, Aug. 1995.

[5] 高濱徹行，阪井節子，小倉久和，中村正郎，“強化学習法による離散値制御のためのファジィ制御規則の学習，” 日本ファジィ学会誌，vol.8, no.1, pp.115–122, Feb. 1996.

[6] M.V. Rendon, “The fuzzy classifier system: A classifier system for continuously varing variables,” Proc. of the 4th ICGA, San Diego, U.S.A., pp.346–353, June 1991.

[7] 古橋　武，中岡　兼，森川幸治，内川嘉樹，“多段ファジィクラシファイアシステムによる制御知識獲得，” 日本ファジィ学会誌，vol.6, no.3, pp.603–609, June 1994.

[8] 古橋　武，中岡　兼，森川幸治，内川嘉樹，前田　宏，“ファジィクラシファイアシステムによる知識発見に関する一考察，” 日本ファジィ学会誌，vol.7, no.4, pp.839–848, Aug. 1995.

[9] 中岡　兼，古橋　武，内川嘉樹，前田　宏，“ファジィクラシファイアシステムの報酬と信頼度割り当てに関する一提案，” 日本ファジィ学会誌，vol.8, no.1, pp.65–71, Feb. 1995.

[10] D.E. Goldberg, “Genetic Algorithms in Search, Optimization, and Machine Learning,” Addison-Wesley, Massachusetts, 1989.

[11] 高濱徹行，宮本誠司，小倉久和，中村正郎，“遺伝アルゴリズムによるファジィ制御規則の学習，” 第 8 回ファジィシンポジウム講演論文集，pp.241–244, May 1992.

[12] 高濱徹行，阪井節子，“遺伝的アルゴリズムによる多目的ファジースケーリング制御規則の学習，” 信学論（D-II），vol.J81-D-II, no.1, pp.119–126, Jan. 1998.

[13] T.C. Chin and X.M. Qi, “Genetic algorithms for learning the rule base of fuzzy logic controller,” Fuzzy Sets & Syst., vol.97, no.1, pp.1–7, July 1998.

[14] 阪井節子，高濱徹行，“非線形最適化手法を用いた倒立振子ファジィ制御規則の学習，” 第 13 回ファジィシステムシンポジウム講演論文集，pp.61–64, June 1997.

[15] 高濱徹行，阪井節子，“多目的最適化手法とファジィ制御規則の学習について，” 第 56 回情報処理学会全国大会講演論文集，no.2, pp.72–73, March 1998.

[16] ムハマドロムジ，高濱徹行，小高知宏，小倉久和，“倒立二重振子系に対するファジィ制御知識の表現とスケーリングによる適応制御，” 日本ファジィ学会誌，vol.8, no.3, pp.576–585, June 1996.

[17] 市橋秀友，渡辺俊彦，“簡略ファジィ推論を用いたファジィモデルによる学習型制御，” 日本ファジィ学会誌，vol.2, no.3, pp.429–437, Aug. 1990.

[18] 今野　浩，山下　浩，“非線形計画法，” 日科技連，March 1978.

[19] 坂和正敏，“非線形システムの最適化，” 森北出版，May 1986.

[20] D.G. Luenberger, “Linear and nonlinear programming,” Addison-Wesley, Massachusetts, 1984.

[21] A.V. Fiacco and G.P. McCormick, “Nonlinear programming: sequential unconstrained minimization techniques,” Society for Industrial and Applied Mathematics, Philadelphia, 1990.

[22] M.J.D. Powell, “An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives,” Comput. J., vol.7, pp.155–162, 1964.

[23] 寺野寿郎，浅居喜代治，菅野道夫，“ファジィシステム入門，” オーム社，April 1987.

[24] 坂和正敏，“ファジィ理論の基礎と応用，” 森北出版，May 1989.

[25] Y.J. Lai and C.L. Hwang, “Fuzzy mathematical programming: methods and applications,” Springer-Verlag, Berlin, 1992.

[26] 中原陽三，玄　光男，“ファジィ数理計画問題の統一的表現，” 信学論（A），vol.J78-A, no.8, pp.992–1001, Aug. 1995.

[27] 西川よし一，三宮信夫，茨木俊秀，“岩波講座情報科学 19 最適化，” 岩波書店，Sept. 1982.

（平成 10 年 7 月 13 日受付，11 月 18 日再受付）

高濱　徹行　（正員）

1982 京大・工・電気第二工卒．1987 同大大学院博士課程研究認定退学．同年福井大学工学部助手．1994 同大講師．1998 より広島市立大学情報科学部知能情報システム工学科助教授．非線形最適化，ファジィ制御を対象とした学習，機械学習，推論，CAI，自然言語処理などに関する研究に従事．情報処理学会，人工知能学会，教育システム情報学会，言語処理学会各会員．工博．

阪井　節子

1979 福井大・教育学部卒．1984 阪大大学院基礎工学研究科数理系後期課程単位取得退学．1986 甲子園大学経営情報学部講師．1990 福井大教育学部助教授．1998 より広島修道大学商学部経営学科教授．ゲーム理論，意志決定，ファジィ数理計画，GA によるファジィ制御，CAI などに関する研究に従事．日本 OR 学会，日本ファジィ学会各会員．工博．