

時変システムのオンライン同定のための適応 PSO

正員 西田 健* 上級会員 坂本 哲三*

Adaptive PSO for Online Identification of Time-Varying Systems

Takeshi Nishida*, Member, Tetsuzo Sakamoto*, Senior Member

(2010年12月27日受付, 2011年4月12日再受付)

Novel adaptive PSO (particle swarm optimization) algorithms called OPSO (online PSO) and ϵ PSO are proposed. These algorithms are designed to prevent increasing of calculation cost in order to embed into online systems, and they have high adaptive performances to the environmental changes. Then, the efficiency of the proposed PSO algorithms are demonstrated by numerical simulation and the online identification of an actual dynamic system.

キーワード: 適応 PSO, 時変システム, オンライン同定, OPSO, ϵ PSO

Keywords: adaptive PSO, time-varying systems, online identification, OPSO, ϵ PSO

1. はじめに

統計的な最適解探索法の一つである PSO (particle swarm optimization) は, 群知能を模倣した非線形計画問題の効率的な解法である⁽¹⁾。ロバストかつ高速に非線形問題, 微分不能問題, 多峰問題などの最適解探索を行う性能が数多くの研究によって検証され, 現在では数多くの応用事例に適用されている⁽²⁾。さらに, 当初, 静的な最適化問題を対象として提案された PSO は, 近年の数多くの研究により動的な実システムへの適用を考慮した改良が加えられ, 観測ノイズの混入や対象とするシステムの時間変化に起因する環境変化, すなわち探索空間の変化に適応可能な PSO が開発されている^{(3)~(9)}。これらの PSO アルゴリズムは, 時間経過に伴って変化する探索空間に適応することを目的としており, 環境変化を検知する方策と局所解への収束を回避する方策の構築に重点が置かれて設計されている。

一方で, PSO は強力な探索手法である反面, 多くの粒子を利用する全領域的な探索を確率的に実行するため, 最急降下法などの従来手法と比較して計算効率が悪い。実際のオンラインシステムに対して PSO を適用するためには, 環境変化に対して迅速に適応すると同時に, システム構成に依存するサンプリング周期内で各ステップ毎の計算を終了

する必要がある。したがって, PSO をオンラインシステムに組み込む場合には, 環境変化への適応性能と共に, 計算量の節約が重要な要素となる。しかし, オンラインシステムに組み込むことを想定して計算量を考慮した適応 PSO アルゴリズムは現在までに提案されていない。そこで本研究では, 環境変化に対する適応能力を備え, さらに計算量の増加を抑えた 2 種類の PSO アルゴリズムを提案する。また, 数値シミュレーションと実験により, 従来手法に対する提案手法の有用性を示す。

以下, 本論文の構成は次の通りである。まず, 2 章において PSO のアルゴリズムの説明を行い, 3 章において関連する従来の適応 PSO アルゴリズムを概観する。その後 4 章と 5 章においてオンラインシステムに適応可能な二種類の PSO アルゴリズムを提案し, 6 章で提案手法の戦略的特徴について考察する。さらに, 7 章において数値シミュレーションによりそれらの性能を評価し, 8 章で実システムへの実装実験により提案手法の有用性を示す。そして最後に 9 章において本論文の結論を述べる。

2. PSO アルゴリズム

PSO は, 粒子と呼ばれる多数の探索点を, 過去の行動履歴と動的に調整される速度に基づいて探索空間中を移動させることにより解を発見するアルゴリズムである。目的関数 $f: \mathbb{R}^l \rightarrow \mathbb{R}$ の最適化問題

$$\min_{\mathbf{x}} f(\mathbf{x}) \geq 0 \dots \dots \dots (1)$$

を解くための PSO の一般的な手順⁽²⁾は一般に次のように

*九州工業大学
〒804-8550 福岡県北九州市戸畑区仙水町 1-1
Kyushu Institute of Technology
1-1, Sensui, Tobata, Kitakyushu, Fukuoka 804-8550, Japan

表される: まず, 探索空間内の粒子の位置を $\mathbf{x}_k^{(m)} \in \mathbb{R}^l$, 速度を $\mathbf{v}_k^{(m)} \in \mathbb{R}^l$ と表す。ここで, $m = [1, M] \in \mathbb{N}_+$ は粒子の番号, $k = 1, 2, \dots$, は離散時刻を表す。これらを以下の手順で更新する。

given パラメータ ω, c_1, c_2 を設定する。 $\mathbf{x}_0^{(m)}$ と $\mathbf{v}_0^{(m)}$ を一様乱数で与える。一様乱数の値域は探索空間に合わせて調整する。

step 1 最小化を目的とするスカラー関数 $f(\cdot)$ によって各粒子の位置を評価し, 一般に $pbest$ と呼ばれる現時刻において最小の評価値を与える各粒子の位置

$$\hat{\mathbf{x}}^{(m)} := \begin{cases} \mathbf{x}_k^{(m)} & \text{if } f(\mathbf{x}_k^{(m)}) < f(\hat{\mathbf{x}}^{(m)}) \\ \hat{\mathbf{x}}^{(m)} & \text{otherwise} \end{cases} \dots\dots\dots (2)$$

とその評価値 $f(\hat{\mathbf{x}}^{(m)})$ を記憶する。ここで $:=$ は代入を表す。

step 2 群全体で最小の評価値を与える

$$\hat{\mathbf{x}}^g := \begin{cases} \hat{\mathbf{x}}^{(m)} & \text{if } f(\hat{\mathbf{x}}^{(m)}) < f(\hat{\mathbf{x}}^g) \\ \hat{\mathbf{x}}^g & \text{otherwise} \end{cases} \dots\dots\dots (3)$$

を $gbest$ と呼ばれる群全体での最良解とし, その評価値 $f(\hat{\mathbf{x}}^g)$ を記憶する。

step 3 以下の式に従って粒子の速度と位置の更新を行う。

$$\begin{aligned} \mathbf{v}_k^{(m)} &= \omega \mathbf{v}_{k-1}^{(m)} + c_1 r_1 (\hat{\mathbf{x}}^{(m)} - \mathbf{x}_{k-1}^{(m)}) \\ &\quad + c_2 r_2 (\hat{\mathbf{x}}^g - \mathbf{x}_{k-1}^{(m)}) \\ \mathbf{x}_k^{(m)} &= \mathbf{x}_{k-1}^{(m)} + \mathbf{v}_k^{(m)} \end{aligned}$$

ここで, $r_1, r_2 = [0, 1]$ は一様乱数を表す。

step 4 収束条件を満たすまで $k := k + 1$ として step 1 へ戻る。

以上の一般的な従来型 PSO アルゴリズムでは, 式 (3) より明らかなように $f(\hat{\mathbf{x}}^g)$ が単調減少する。ここで, 例えば評価関数が $f(\cdot)$ から $f^*(\cdot)$ へと変化し, それに伴って最適解が $\hat{\mathbf{x}}^g$ から $\hat{\mathbf{x}}^{g^*}$ に移動した場合を考えてみよう。従来型 PSO では評価関数の変化を考慮しないため, $f(\hat{\mathbf{x}}^{g^*}) < f(\hat{\mathbf{x}}^g)$ が満たされない限り最適解に関する評価値 $f(\hat{\mathbf{x}}^g)$ は更新されず, その場合には $\hat{\mathbf{x}}^g$ は不適切な解に拘束されたままになることがわかる。特に, 時間が十分に経過して評価値 $f(\hat{\mathbf{x}}^g)$ が収束した後は, このような環境変化は検出されない可能性が一層高くなる。すなわち, 従来型 PSO アルゴリズムでは評価関数 $f(\cdot)$ の時間変化を想定していないため, 最適解の移動や評価関数の変化などの環境変化に適応した最適解の更新が行われない。

3. 関連する研究

〈3・1〉 適応 PSO ここでは, 現在までに提案された環境変化に適応するための PSO アルゴリズムの性能と特徴を概観する。

まず, 初期の研究⁽³⁾⁽⁴⁾では, 定期的にすべての粒子の記憶を初期化し, 粒子の $pbest$ の値と位置ベクトルを現在の位置ベクトルに再配置する手法が提案された。しかし, この手法では再初期化の頻度の決定が困難であることが指摘されている⁽⁸⁾。すなわち, 環境変化の周期についての予備知識が無い場合には, 高い頻度で粒子の記憶を再初期化することによって環境変化に備えなければならないが, 高頻度の再初期化は収束速度を低下させ探索を不安定化させる。

次に, 「見張り (sentry)」という概念を導入することにより環境変化の監視と適応を行う APSO (Adaptive PSO)⁽⁵⁾が提案された。見張り粒子は, 環境変化を監視するために特別に設計された粒子であり, 探索空間に複数配置される。見張り粒子が環境変化を検知すると, その情報が他のすべての見張り粒子に伝達されると同時に, 通常の粒子の記憶を再初期化する。しかし, 見張り粒子は自らが存在する場所の局所的な環境変化しか検知できないことや, 環境変化と観測ノイズの判別ができないことから, 観測ノイズが存在する多くの実用では全粒子の記憶の再初期化が頻発するという問題が生ずる。

その後, 環境変化に頑健な PSO アルゴリズムとして, 粒子間に斥力や吸引力が発生するモデルを導入し, 動的環境への適応を可能にした CPSO (Charged PSO)⁽⁶⁾や, 複数の群を組み合わせて用いる MSO (Multi-swarm optimization)⁽⁷⁾が提案された。MSO は, すべての粒子が同じ最適解に収束するのを防ぐために, 斥力を生ずる複数の粒子群を用いる。これらのアルゴリズムは探索空間を効率良く探索する工夫によって, 粒子が局所解に拘束されることを防ぐ。しかし, 環境変化が連続的もしくは頻繁に生ずることを想定しておらず, 環境変化を検知した場合にはすべての粒子を初期化するため, やはり前述の PSO アルゴリズムと同様に再初期化の頻発が問題となる。さらに, 斥力を定めるための複数の設定パラメータが必要であるため調整手順が煩雑であり, 各粒子間の斥力を求めるための計算量が多い。

さらに近年, これらのアルゴリズムの問題点を解決することを目的としたいくつかの適応 PSO アルゴリズムが提案されている。その一つに, 粒子の最良解に対する評価値に定数を乗ずることによって連続的な環境変化に適応する DAPSO (distributed adaptive PSO)⁽⁸⁾がある。このアルゴリズムは, 1 よりも少し大きな定数を $pbest$ に乗じた値を現時刻の最小値探索に利用するという簡潔な手順の導入により, ほとんど計算量を増加させることなく環境変化への適応を可能としている。また一方, 粒子の更新のための高精度な評価を行うために, 環境変化と最適解の移動を考慮した粒子の評価を毎時刻行う MPSO (modified PSO)⁽⁹⁾が提案されている。このアルゴリズムは, 計算量は増加するものの, 連続的に時間変化する環境に対する迅速な適応性能を有する。

ただし, これらのいずれの先行研究でも, オンラインシステムに実装する際に重要になる計算量の節約は考慮されていない。

4. OPSO

各時刻において生ずる環境変化を陽に表現するために、評価関数を時変関数 $f_k(\cdot)$ とし、さらに $pbest$ と $gbest$ も時変として $\hat{\mathbf{x}}_k^{(m)}$ および $\hat{\mathbf{x}}_k^g$ とし、その最適値を求めることを目的とした適応 PSO アルゴリズム OPSO (Online PSO) を以下のように構成した。

(4.1) アルゴリズム

given ω, c_1, c_2 の値を設定する。 $\mathbf{x}_0^{(m)}$ と $\mathbf{v}_0^{(m)}$ を一様乱数で与える。 $k = 1$ として step 1 へ移る。

step 1 前時刻の各粒子の最適解 $\hat{\mathbf{x}}_{k-1}^{(m)}$ を現時刻の評価関数で再評価し、それらの中で最小値を与える

$$\hat{\mathbf{x}}_k^g = \arg \min \left\{ f_k(\hat{\mathbf{x}}_{k-1}^{(m)}) \right\} \dots \dots \dots (4)$$

を求める。

step 2 各粒子の速度と位置の更新を行う。

$$\begin{aligned} \mathbf{v}_k^{(m)} &= \omega \mathbf{v}_{k-1}^{(m)} + c_1 r_1 (\hat{\mathbf{x}}_{k-1}^{(m)} - \mathbf{x}_{k-1}^{(m)}) \\ &\quad + c_2 r_2 (\hat{\mathbf{x}}_k^g - \mathbf{x}_{k-1}^{(m)}) \\ \mathbf{x}_k^{(m)} &= \mathbf{x}_{k-1}^{(m)} + \mathbf{v}_k^{(m)} \end{aligned}$$

step 3 最小の評価値を与える各粒子の位置

$$\hat{\mathbf{x}}_k^{(m)} = \begin{cases} \mathbf{x}_k^{(m)} & \text{if } f_k(\mathbf{x}_k^{(m)}) < f_k(\hat{\mathbf{x}}_{k-1}^{(m)}) \\ \hat{\mathbf{x}}_{k-1}^{(m)} & \text{otherwise} \end{cases}$$

とその評価値 $f_k(\hat{\mathbf{x}}_k^{(m)})$ を記憶する。

step 4 群全体での最良解

$$\hat{\mathbf{x}}_k^g = \arg \min \left\{ f_k(\hat{\mathbf{x}}_k^{(m)}) \right\}$$

を求める。これを時刻 k における OPSO の推定結果とする。

step 5 $k = k + 1$ として step 1 へ戻る。

(4.2) OPSO の特徴と性質 時変関数 $f_k(\cdot)$ の最良解を探索するために、まず step 1 で現時刻の評価関数によって前時刻の $pbest$ を再評価し、それらの最小値を与える粒子の位置 $\hat{\mathbf{x}}_k^g$ を求める。この粒子の位置は現時刻の評価関数における最良解に最も近い $pbest$ である。この時、評価関数は時間変化するため $f_k(\hat{\mathbf{x}}_{k-1}^{(m)}) \geq f_{k-1}(\hat{\mathbf{x}}_{k-1}^{(m)})$ という場合も生じ得ることから、 $f_k(\hat{\mathbf{x}}_{k-1}^{(m)})$ は単調減少しない。その結果、step 3 において算出される $\hat{\mathbf{x}}_k^{(m)}$ の評価値 $f_k(\hat{\mathbf{x}}_k^{(m)})$ も単調減少せず、したがって、step 4 で算出される $\hat{\mathbf{x}}_k^g$ の評価値 $f_k(\hat{\mathbf{x}}_k^g)$ も単調減少しない。このように、step 1 の手順の導入によって従来型 PSO が環境変化に適応できない原因であった「 $f(\cdot)$ が単調減少する」という問題が解消され、本手法では環境変化が $pbest$ や $gbest$ の更新に反映される。

次に step 2 と step 3 において、 $\hat{\mathbf{x}}_k^g$ を利用する粒子の速度と位置の更新を行う。最後に step 4 で更新後の粒子を評価し、その時刻の各粒子の最適解 $\hat{\mathbf{x}}_k^{(m)}$ と群の最良解 $\hat{\mathbf{x}}_k^g$ を得て、次の時刻の処理に進む。粒子の速度および位置の更

新と評価の順序が、従来型 PSO のそれらとは逆になっているが、これは step 1 の導入に伴う変更である。すなわち、step 1 における環境変化への適応結果を反映した粒子の更新を行い、その後各粒子を評価して現時刻の最良解を決定しようとするための手順の変更である。

OPSO には次に挙げるような特徴がある: 1) 時変の評価関数を用いることで $pbest$ と $gbest$ の評価値が単調減少しない、2) 新たな設計パラメータの増加が無い、3) 計算手順が簡潔で計算量の増加が抑えられている、4) 従来型 PSO アルゴリズムと比較して、OPSO における計算量の増加は step 1 に関する部分のみである。

5. ϵ PSO

以下に示す ϵ PSO と呼ぶアルゴリズムは、最適値に関する記憶、すなわち $pbest$ を時間経過に伴って忘却させる機構の導入により、 $f(\cdot)$ が単調減少するという従来型 PSO の問題を解決する。本手法は、新たに導入する手順が OPSO よりもさらに簡潔であり、また計算量の増加もさらに抑えられている。

以下では OPSO と同様に、各時刻において生ずる環境変化を陽に表現するために評価関数を $f_k(\cdot)$ と表す。さらに $pbest$ と $gbest$ も時変として $\hat{\mathbf{x}}_k^{(m)}$ および $\hat{\mathbf{x}}_k^g$ と表す。

(5.1) アルゴリズム

given ω, c_1, c_2 の値を設定する。 $\mathbf{x}_0^{(m)}$ と $\mathbf{v}_0^{(m)}$ を一様乱数で与える。 $k = 1$ として step 1 へ移る。

step 1 最良解の評価値に小さな正の定数を加え、過去の最適値の評価を忘却させる。

$$f_k(\hat{\mathbf{x}}_{k-1}^{(m)}) := f_{k-1}(\hat{\mathbf{x}}_{k-1}^{(m)}) + \varepsilon \quad (\forall m) \dots \dots \dots (5)$$

ここで $:=$ は代入を意味する。

step 2 最小の評価値を与える各粒子の位置

$$\hat{\mathbf{x}}_k^{(m)} = \begin{cases} \mathbf{x}_k^{(m)} & \text{if } f_k(\mathbf{x}_k^{(m)}) < f_k(\hat{\mathbf{x}}_{k-1}^{(m)}) \\ \hat{\mathbf{x}}_{k-1}^{(m)} & \text{otherwise} \end{cases}$$

とその評価値 $f_k(\hat{\mathbf{x}}_k^{(m)})$ を記憶する。

step 3 群全体での最良解

$$\hat{\mathbf{x}}_k^g = \arg \min \left\{ f_k(\hat{\mathbf{x}}_k^{(m)}) \right\}$$

を記憶する。

step 4 各粒子の速度と位置の更新を行う。

$$\begin{aligned} \mathbf{v}_k^{(m)} &= \omega \mathbf{v}_{k-1}^{(m)} + c_1 r_1 (\hat{\mathbf{x}}_{k-1}^{(m)} - \mathbf{x}_{k-1}^{(m)}) \\ &\quad + c_2 r_2 (\hat{\mathbf{x}}_k^g - \mathbf{x}_{k-1}^{(m)}) \\ \mathbf{x}_k^{(m)} &= \mathbf{x}_{k-1}^{(m)} + \mathbf{v}_k^{(m)} \end{aligned}$$

step 5 $k = k + 1$ として step 1 へ戻る。

(5.2) ϵ PSO の特徴と性質 まず step 1 の手順において、 $f_{k-1}(\mathbf{x}_{k-1}^{(m)})$ に微小な値 ε を毎時刻加え、時間経過とともにその値を徐々に増加させることにより各粒子の $pbest$

の評価値の忘却を行う。この手順によって、環境変化によって移動した最適解が新たに発見される可能性が生ずる。例えば、ある時刻 k における最適解 \hat{x}_k^g が環境変化によって l 時刻後に \hat{x}_{k+l}^{g*} に移動する場合、粒子は以下の条件を満たす解を探索して発見すればよい。

$$f_{k+l}(\hat{x}_{k+l}^g) < f_k(\hat{x}_k^g) + \varepsilon l$$

時間経過に伴って両辺の差は拡大するため、確率的に移動する粒子が \hat{x}_{k+l}^{g*} の近傍の解を発見し、 $pbest$ の情報が更新される確率は時間経過に伴い上昇する。またこの処理によって、粒子が過去の最適解と同じ評価値を持つ新たな最適解を発見した場合には、新たに発見された結果が最適値として採用されるようになる。

ただし、このアルゴリズムを利用する場合には、適切な ε の値の設定が必要である。小さすぎる ε は適応性能を低下させ、大きすぎる ε は解の忘却と再発見を高頻度に生じさせる。ただし、 ε の値が一定の範囲内で適切に設定できれば、 ε PSO による探索はこの値に敏感でないため、数度の試行で比較的容易に値を調整することができる。

以上で示したように、 ε PSO には次に挙げるような特徴がある: 1) $pbest$ を忘却するので $gbest$ の評価値が単調減少しない、2) 新たに必要となる設定パラメータは ε のみである、3) 処理手順が簡潔であり、従来の PSO からの計算量の増加は OPSO よりも少ない、

6. $pbest$ と $gbest$ の保存戦略

適応 PSO アルゴリズムにおける $pbest$ と $gbest$ の保存戦略は、環境変化に対する適応能力に影響を与える最も重要な要素の一つである。ここでは、3 章に示した従来の適応 PSO との関係から、OPSO と ε PSO の戦略の位置づけを考察する。

まず、文献(3)~(7)の手法は、環境変化を検知する度に $pbest$ と $gbest$ を初期化し、それ以前の粒子の記憶を破棄するという戦略に基づく。これらの手法における $pbest$ と $gbest$ の保存戦略は非効率的であることが指摘されている⁽⁸⁾。

一方、DAPSO⁽⁸⁾では、保存された $pbest$ の定数倍以内の評価値を持つ新たな解が抽出されることを許容するという戦略を用いる。すなわち、 $pbest$ の評価値にある一定の値を乗じて保存情報を劣化させるという戦略によって、 $pbest$ や $gbest$ が局所解に固定化されるのを避け、環境変化への適応を可能としている。 ε PSO の $pbest$ と $gbest$ の保存戦略は、この戦略を拡張したものとして位置づけることができる。すなわち、 ε PSO では $pbest$ の評価値の劣化の程度を時間経過に伴って増加させることにより、環境変化の適応能力を徐々に向上させようとする戦略を用いる。この戦略によって、 ε PSO は DAPSO とほぼ同等の計算量で、環境変化に対するより高い適応性能を達成することができる。

MPSO⁽⁹⁾は、毎時刻、時変の評価関数に基づく $gbest$ と $pbest$ の再評価と合理的な粒子速度の修正を行うという戦略

を採用している。ただし、前時刻と現時刻の $gbest$ と $pbest$ の位置関係を考慮して速度を詳細に再計算するため環境変化に対する適応の精度は高いが、計算量が大きく増加するという問題がある。OPSO の $gbest$ と $pbest$ の保存戦略は、これを簡略化したものとして位置づけられる。すなわち、環境変化に適応するための修正は、計算量の増加を抑えるために $pbest$ と $gbest$ の再評価のみに限定し、これに基づいて粒子を更新するという戦略を採用する。現時刻のステップ内では合理的な速度の算出がなされない場合も存在することが考えられるが、この確率的に定まる速度成分の差は、十分な時間経過によって零に収束することが期待できる。

7. 数値シミュレーション

ここでは時変評価関数を用いて、従来研究の適応 PSO アルゴリズムと 2 種類の提案手法との性能を比較する。

(7・1) 問題設定 ここでは、以下に示す時変関数の最小値探索を考える。

$$f_k(x_1, x_2) = 1 - \exp \left[- \frac{\{x_1 - 250 - 125 \sin(0.01k)\}^2}{2 \cdot 40^2} - \frac{\{x_2 - 250 + 125 \cos(0.01k)\}^2}{2 \cdot 40^2} \right]$$

この関数の形状と時間変化の様子を Fig. 1 に示す。この関数の最適解における評価関数の値は 0 であり、その位置は $(x_1, x_2) = (250, 250)$ を中心とした半径 125 の円周上を移動し、およそ 628 ステップで元の位置に戻る。本シミュレーションにおける各種 PSO の目的は、一定速度で移動し続ける目的関数の最適解の位置を \hat{x}_k^g として捉え続けることである。ここでは比較のために、従来型 PSO, CPSO, DAPSO, MPSO, OPSO および ε PSO を用いてシミュレーションを行った。また、すべてのアルゴリズムにおいて 3 種類のパラメータは文献⁽¹⁰⁾の解析に基づいて $\omega = 0.729$, $c_1 = c_2 = 1.4955$ とし、粒子数を $M = 100$ とした。さらに、各 PSO アルゴリズムの設定パラメータは各文献に記載されている値を用いた[†]。 ε PSO のパラメータは試行錯誤により $\varepsilon = 0.1$ とした^{††}。

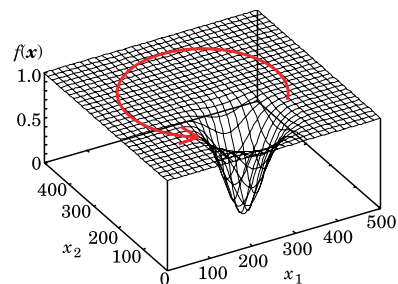


Fig. 1. Target function. This function moves with following the red line.

[†] DAPSO の設定パラメータを $P = 2.0$ とした。

^{††} この試行錯誤において、 ε の値が 10^{-5} を下回ると、最適解に対する適応ができなくなることが確認された。また一方、この例では評価関数が常に変化するので、 ε を大きな値で設定した場合の解の忘却と再発見の頻発による PSO の性能の劣化は確認されなかった。

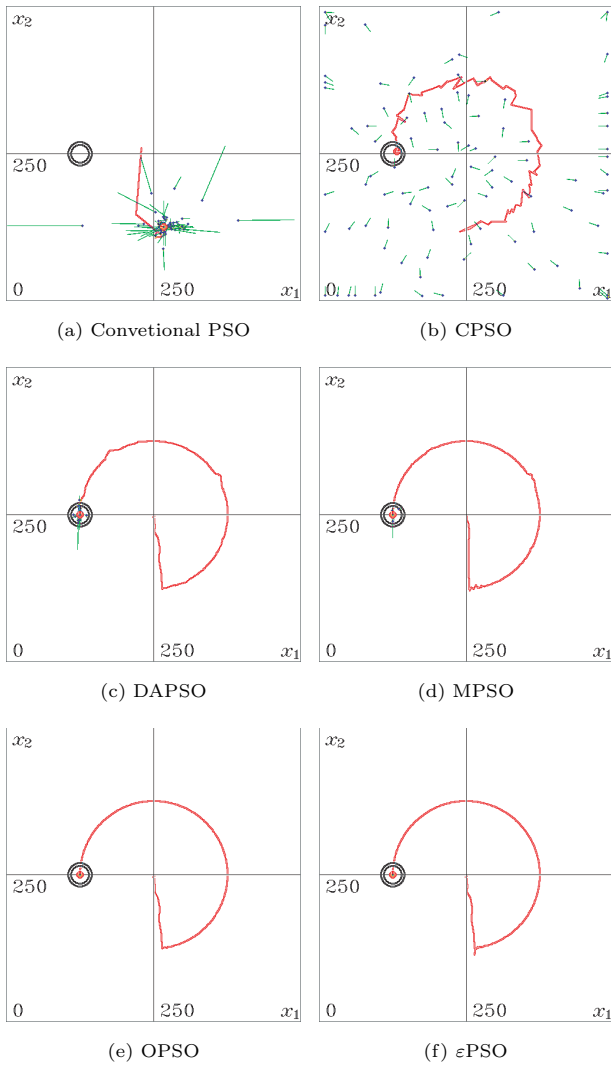


Fig. 2. Result of numerical simulation. The red solid line represents the trajectory of $\hat{\mathbf{x}}_k^g$, and the red circle is position of $\hat{\mathbf{x}}_k^g$ at $k = 471$. The blue dots and the green line segments are the position and velocity of particles. The gray double circle represents the position of the global minimum solution.

〈7・2〉 適応性能 これらのアルゴリズムを対象関数に適用した結果, 得られた $\hat{\mathbf{x}}_k^g$ の軌跡を Fig. 2 に示す。また, $f_k(\hat{\mathbf{x}}_k^g)$ の時間推移のグラフを Fig. 3 に示す。Fig. 2 (a) および Fig. 3 (a) より, 従来の PSO は一度でも $f(\hat{\mathbf{x}}^g)$ が最小値を捉えたと, この値の更新が止まってしまうため, 解の位置が変化しても追従できないことがわかる。Fig. 2 (b) および Fig. 3 (a) より, CPSO では粒子の再初期化が毎時刻発生し, $\hat{\mathbf{x}}_k^g$ の軌跡が安定しないことがわかる。Fig. 2 (c) より, DAPSO では一定時間間隔で $f_k(\hat{\mathbf{x}}_k^g)$ の更新が行われ, 変化する関数に追従していることがわかる。ただし, Fig. 3 (a) よりわかるように $f_k(\hat{\mathbf{x}}_k^g)$ の推移には脈動が生じ, その結果として $\hat{\mathbf{x}}_k^g$ の軌跡の一部が歪んでいることがわかる。これらとは対照的に, Fig. 2 (d) より MPSO の $\hat{\mathbf{x}}_k^g$ の軌跡の歪みが小さく, さらに Fig. 2 (e) と (f) より OPSO および

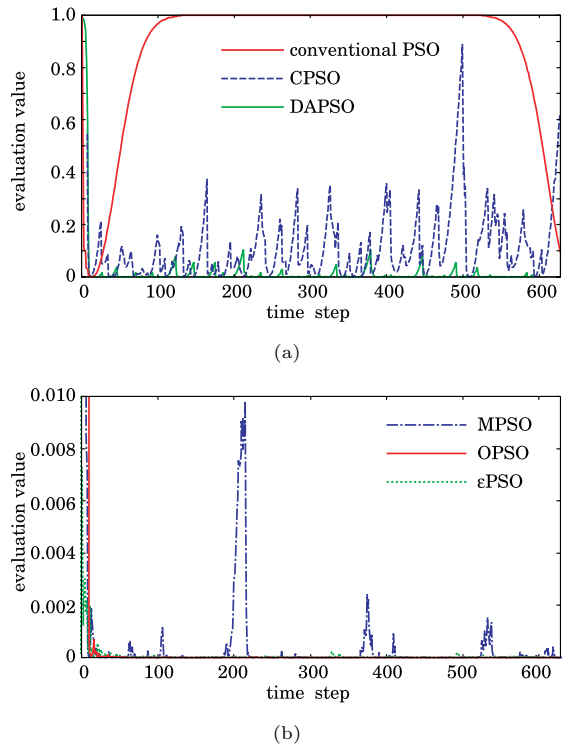


Fig. 3. Time evolution of $f_k(\hat{\mathbf{x}}_k^g)$. Results of the conventional PSO, CPSO, and DAPSO are shown in (a), and of the MPSO, OPSO, and ϵ PSO are shown in (b).

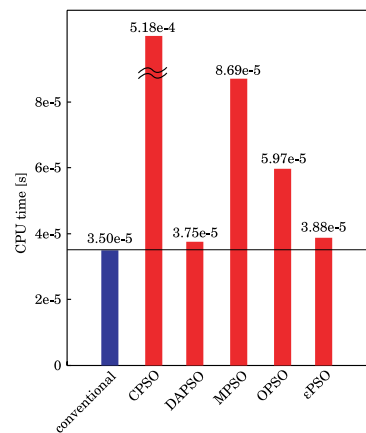


Fig. 4. CPU time of the execution for each step.

ϵ PSO の $\hat{\mathbf{x}}_k^g$ の軌跡には歪みがなく変化する対象関数に滑らかに追従できていることがわかる。また Fig. 3 (b) より, 上述の従来型 PSO, CPSO および DAPSO よりも, これらの 3 種のアルゴリズムによる結果は $f_k(\hat{\mathbf{x}}_k^g)$ が小さな値で推移していることがわかる。ただし, MPSO の $f_k(\hat{\mathbf{x}}_k^g)$ の値には脈動が見られ, OPSO や ϵ PSO よりも大きな値で推移したことがわかる。

〈7・3〉 計算量 次に, アルゴリズムを 1 ステップ実行するのに費やされた CPU 時間の平均を Fig. 4 に示す。計測に用いた計算機の CPU のクロック数は 2.40 GHz であり, OS は VineLinux である。この結果より, 前述の適応能力に関するシミュレーション結果において良い性能を

示した MPSO, OPPO, ϵ PSO の中で, ϵ PSO の計算量が最も少ないことがわかる。また OPPO は, 環境変化に対する追従の最適性能の向上を目的とした MPSO よりも計算量が少なく, 適応能力が高いことがわかる。

これらの数値シミュレーションの結果より, 本研究で提案する OPPO と ϵ PSO は適応能力の高さと計算量の低さにおいて, 従来の適応 PSO アルゴリズムよりも優れていることが示された。

8. 実験

ここでは, 提案アルゴリズムをオンラインシステムに実装し, それらの性能評価実験を行った結果を示す。また, 上述のシミュレーションで高い適応性能を示した MPSO と従来型の PSO についても同様の実験を行い提案アルゴリズムとの性能の比較を行う。

〈8・1〉 実験対象 本実験では, フィルム巻き取り装置に取り付けられたサーボモータの回転速度制御系を対象として, その離散伝達関数のオンラインパラメータ推定を行った。まず, 入力をサーボモータの発生トルク, 出力を巻き取られるフィルムの速度とし, 対象系の動特性を零次ホールドと一次遅れ系の合成によって近似できると仮定すると, その離散伝達関数は次のように表される。

$$G(z^{-1}) = \frac{Y(z^{-1})}{U(z^{-1})} = \frac{K \left\{ 1 - \exp\left(-\frac{1}{T_a} T_s\right) \right\} z^{-1}}{1 - \exp\left(-\frac{1}{T_a} T_s\right) z^{-1}} \dots \dots \dots (6)$$

ここで, T_a は時定数, $T_s (= 10[\text{ms}])$ はサンプリング時間である。これより, 時刻 k における系の出力は以下のように表すことができる。

$$y_k = \frac{b_k z^{-1}}{1 + a_k z^{-1}} u_k = \mathbf{z}_k^T \mathbf{x}_k \dots \dots \dots (7)$$

ここで,

$$a_k = -\exp(-T_s/T_a) \dots \dots \dots (8)$$

$$b_k = K \{1 - \exp(-T_s/T_a)\} \dots \dots \dots (9)$$

であり, また $\mathbf{x}_k \triangleq (-a_k, b_k)^T$ および $\mathbf{z}_k = (y_{k-1}, u_{k-1})^T$ である。事前の同定実験によるノミナル値は $T_a = 1.432$ および $K = 0.455$ であり, これらの値を基にすると $a = -0.993$ および $b = 3.16 \times 10^{-3}$ となった。

次に, 本実験において発生した対象の入力と出力の時間推移を Fig. 5 に示す。モータの発生トルクは二次遅れフィルタを用いて生成した。また, 目標値を 10 [s] の時点で 1.5 [Nm] から 2.0 [Nm] に変更した。この実験では時間経過に伴ってフィルムを巻き取っていくので, 巻き取りロールの半径はフィルムの厚みによって増加し, それに伴う系のパラメータの変化が生ずる。また, 駆動部には摩擦要素などの非線形なダイナミクスを生じさせる要因が存在するため, 目標値を変更すると系のパラメータは非線形に変動する。本実験では各種 PSO により, Fig. 5 の入出力の推移に基づいて

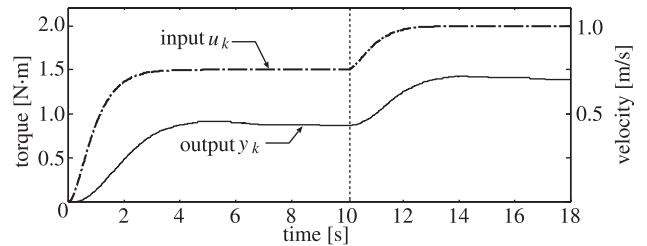


Fig. 5. Time evolution of the input and output of the system.

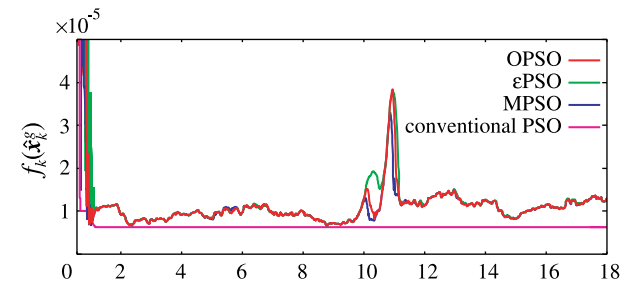


Fig. 6. Time evolution of $f_k(\hat{\mathbf{x}}_k^g)$.

この対象の式 (7) に示すシステムパラメータ \mathbf{x}_k をオンラインで推定した。

〈8・2〉 PSO の設定 各 PSO アルゴリズムにおける粒子の評価には, 以下の評価関数を用いる。

$$f_k(\mathbf{x}_k^{(m)}) = \sum_{j=0}^I |y_{k-j} - \hat{y}_{k-j}|$$

$$= \sum_{j=0}^I |y_{k-j} - \mathbf{z}_{k-j}^T \mathbf{x}_k^{(m)}|$$

ここで, I は時刻を遡って評価を行うステップ数であり $I = 100$ とした。また以下の実験では, すべてのアルゴリズムのパラメータを共通に $\omega = 0.729$, $c_1 = 1.4955$, $c_2 = 1.4955$, $M = 100$ と設定した。さらに, ϵ PSO に関しては, 前述の考察に基づいた試行錯誤により $\epsilon = 10^{-4}$ と設定した。MPSO の固有のパラメータは, 文献⁽⁹⁾中に記載されている値, すなわち $\lambda = 0.1$ を用いた。

〈8・3〉 実験結果 まず, g_{best} すなわち $f_k(\hat{\mathbf{x}}_k^g)$ の値の時間推移を Fig. 6 に示す。この図より, 従来型 PSO の g_{best} の値は単調減少し一定の値に収束しているが, その他の PSO による g_{best} の値は時間経過に伴って増減していることがわかる。特に目標値が変化した 10 [s] 付近では, 入出力関係が変化したために $f_k(\hat{\mathbf{x}}_k^g)$ の値が増加し, その後システムのパラメータ推定が適応的に進むことによって再び減少している様子が見られる。

次に, 推定パラメータ $\hat{\mathbf{x}}_k^g$ に基づいて再現した出力と実際の出力の推定誤差, すなわち $|y_k - \mathbf{z}_k^T \hat{\mathbf{x}}_k^g|$ の値の時間推移を Fig. 7 に示す。この図より, 従来型 PSO では対象の時間変化に適応できず推定誤差が時間経過とともに増加しているが, それ以外の適応 PSO アルゴリズムでは出力を高精度に推定できたことがわかる。

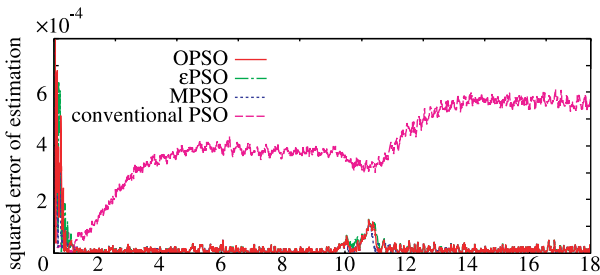
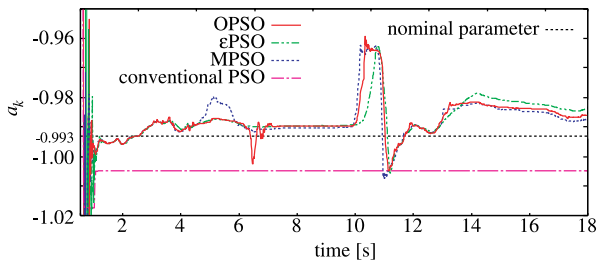
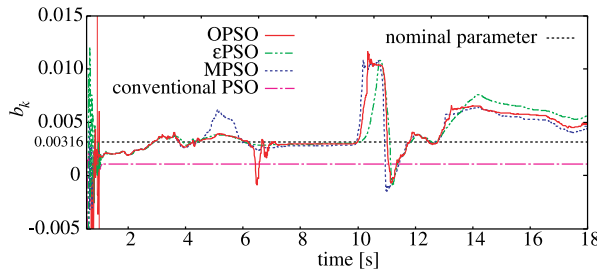


Fig. 7. Estimation error $|y_k - z_k^T \hat{x}_k^g|$.



(a) a_k



(b) b_k

Fig. 8. Time evolution of the estimated parameters.

Table 1. Average CPU time of calculation of each iteration of the PSO algorithms.

	MPSO	OPSO	ϵ PSO
time [ms]	0.337	0.170	0.099

さらに、推定されたパラメータ a_k と b_k の時間推移を Fig. 8 に示す。これらの図より、従来型 PSO では g_{best} の値の収束に伴ってパラメータの推定が更新されず、事前に求めたノミナル値からも離れた推定結果が得られたことがわかる。一方、その他の適応 PSO では各パラメータの推定結果は速やかに系の変化に対して適応しており、事前の同定で得たノミナル値付近の値が推定されていることがわかる。

最後に、各サンプリング周期で要した各 PSO の平均の計算時間を Table 1 に示す。計算機の性能は、前述のシミュレーションと同様である。この結果より、OPSO の計算量は MPSO の 50.4%、 ϵ PSO の計算量は MPSO の 29.4% であることがわかる。高次の応答特性持つ対象のパラメータ推定や、多数のサブシステムを同時に制御するような系への複数の PSO の実装では、PSO の計算に必要とされる時間と資源は比例もしくは指数的に増加するため、これらの

計算量の削減の有用性は大きいと考えられる。

以上の実験結果より、提案手法である OPSO と ϵ PSO は、従来手法と同程度の最適性能をより少ない計算量で実現できることが示された。

9. おわりに

本論文では、まず従来型 PSO が計測ノイズやシステムパラメータの変化に起因する環境変化に適応できない原因は、評価関数 $f(\cdot)$ が単調減少するという問題にあることを述べた。次に、計算量の増加を抑えた上でこの問題を解消し、高い適応性能を有する適応 PSO アルゴリズム OPSO と ϵ PSO を提案した。前者は、評価関数の時間変化を陽に表現してアルゴリズムを構築し、さらに p_{best} の評価値が単調減少とならないような手順を導入することでこの問題の解決を図った。後者は、 p_{best} の評価値を時間経過に伴って忘却する手順の導入により問題の解決を図った。また、これらのアルゴリズムは、オンラインシステムに組み込んでシステムパラメータの同定を行うことを想定し、簡潔な手順となるよう設計された。数値シミュレーションによる従来の適応 PSO アルゴリズムとの比較により、これらの有用性を示した。さらに、実際のオンラインシステムに実装することによって、これらのアルゴリズムの性能を検証した。その結果、提案手法である OPSO と ϵ PSO は、従来手法と同程度の最適性能をより高速に実現できることが示された。

本論文では、従来型 PSO に対して推奨されるパラメータを用いて種々のシミュレーションを行ったが、今後の課題として、これらの値の変更が本手法の性質に与える影響を詳細に考察することが挙げられる。また、対象とする時変の評価関数の性質と設定パラメータの関連性の検証も挙げられる。さらに、本研究では比較的低次元のシミュレーションや実験によって提案手法の性能を評価したが、より高次元の探索空間を対象とする場合の提案手法の性能評価についても今後の課題である。


今後の発展として、離散伝達関数の係数ベクトルのオンライン推定を高精度に行うことができれば、セルフチューニング適応制御が可能になる⁽¹²⁾ため、提案した適応 PSO を用いるセルフチューニング制御系の構成が考えられる。

文 献

- (1) J. Kennedy and R. C. Eberhart: "Particle Swarm Optimization", Proc. of IEEE Int. Conf. on Neural Network IV, pp.1942-1948 (1995)
- (2) 石亀篤司:「Particle Swarm Optimization -群れでの探索-」, 計測と制御, Vol.47, No.6, pp.459-465 (2008)
- (3) A. Carlisle and G. Dozier: "Adaptive particle swarm optimization to dynamic environments", Proc. of Int. Conf. on Artificial Intelligence, pp.429-433 (2000)
- (4) R. C. Eberhart and S. Yuhui: "Tracking and optimizing dynamic systems with particle swarms", Proc. of Congress on Evolutionary Computation, pp.94-100 (2001)
- (5) A. Carlisle and G. Dozier: "Tracking changing extrema with adaptive particle swarm optimizer", Proc. of Soft Computing,

- Multimedia Biomedicine, Image Processing and Financial Engineering, pp.265–270 (2002)
- (6) T. M. Blackwell: “Swarms in dynamic environments”, LNCS, Vol.2723, pp.1–12 (2003)
 - (7) T. Blackwell and J. Branke: “Multi-swarms optimization in dynamic environments”, LNCS, Vol.3005, pp.489–500 (2004)
 - (8) X. Cui and T. E. Potok: “Distributed Adaptive Particle Swarm Optimizer in Dynamic Environment”, Proc. of IEEE Int. Parallel and Distributed Processing Symposium, pp.244–250 (2007)
 - (9) X. Zhang, Y. Du, Zheng Qin, G. Qin, and J. Lu: “A modified particle swarm optimizer for tracking dynamic system”, LNCS, Vol.3612, pp.592–601 (2005)
 - (10) M. Clerc and J. Kennedy: “The particle swarm - explosion, stability, and convergence in a multidimensional complex space”, *IEEE Trans. EC*, Vol.6, No.1, pp.58–73 (2002)
 - (11) K. Mizoguchi and T. Sakamoto: “Self-tuning decentralized controller design of web tension control system”, Proc. of EUROSIM congress on Modeling and Simulation (2010)
 - (12) M. Doi, T. Kamiya, and Y. Mori: “A study on robust asymptotic tracking performance for generalized minimum variance control”, *Trans. IEE Japan*, Vol.119-C, No.11, pp.1420–1426 (1999)

西田 健 (正員) 1998年九工大・工・設計生産工卒業。2002年九工大大学院博士後期課程修了。同年より九工大・機械知能工学・助手。2007年より助教, 博士(工学)。屋外移動ロボットに関する研究に従事。日本ロボット学会, 計測自動制御学会, 日本神経回路学会, 電子情報通信学会などの会員。



坂本 哲三 (上級会員) 1984年九大院博士課程修了, 同年九大助手, 翌年九工大助手。以後, 同講師・助教を経て, 2002年同教授, 工博, 主にリニアドライブ・磁気浮上およびウェブ張力系などの解析・制御の研究に従事。計測自動制御学会などの会員。

