平成 28 年度(2016 年度)　修士論文

進化的計算手法を用いた都市貯留関数モデルのパラメータ同定

平成 29 年 3 月

首都大学東京 大学院 都市環境科学研究科

15885410　金塚　匠

(指導教授　河村　明)

進化的計算手法を用いた都市貯留関数モデルのパラメータ同定

学修番号　15885410　　金塚　匠
都市基盤環境学域　環境システム分野
指導教授　河村　明

１．目的

最適化手法は経済学や工学の分野で必須のツールとなっており，その手法の開発も盛んに行われている．昨今，メタヒューリスティクスと呼ばれる特定の問題に依存しない手法が提案されており，その代表的な手法である粒子群最適化(PSO：Particle Swarm Optimization)のアルゴリズムを用いてタンクモデルのパラメータ同定が行われている．ところで，托卵というカッコウの特殊な生態に着目してアルゴリズム化された Cuckoo Search(CS)は，PSO よりも強力な探索性能を持つとされている．しかし，CS を用いた流出解析モデルのパラメータ同定の事例は見当たらない．

そこで，本研究では都市部のハイドログラフを良好に再現でき，7 つの未知パラメータをもつ都市貯留関数モデルに対し，パラメータの真値が明らかな仮想流域と，真値が不明な実流域のデータを入力し，メタヒューリスティックな進化的計算手法である，PSO，CS，そして流出解析モデルへの適用事例が多く，その有効性が示されている SCE-UA 法(Shuffled Complex Evolution method - University of Arizona)の 3 手法を用いてパラメータ同定を行い，それぞれの探索性能について比較検討を行った．

２．研究手法

2.1 都市貯留関数モデル

都市貯留関数モデルでは図-1 に示す都市流域の流出機構を考慮した，流域の総貯留高 s(mm)に関係する流入出成分を組み込んでおり，それらは二価関数の貯留関数モデルとして式(1)-(4)のように定式化される．

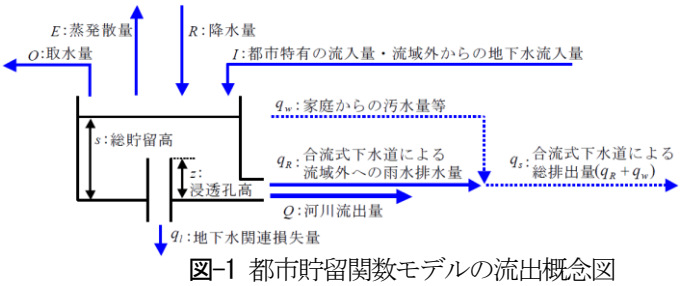

図-1 都市貯留関数モデルの流出概念図

$$s = k_1(Q+q_R)^{p_1} + k_2\frac{d}{dt}\{(Q+q_R)^{p_2}\} \quad (1) \qquad \frac{ds}{dt} = R + I - E - O - Q - q_R - q_l \quad (2)$$

$$q_l = \begin{cases} k_3(s-z) & (s \geq z) \\ 0 & (s < z) \end{cases} \quad (3) \qquad q_R = \begin{cases} \alpha(Q+q_R-Q_0) & (Q+q_R-Q_0) < q_{R_{max}} \\ q_{R_{max}} & (Q+q_R-Q_0) \geq q_{R_{max}} \end{cases} \quad (4)$$

ここに，t：時間(min)，Q：河川流出量(mm/min)，Q_0：初期河川流出量(mm/min)，R：降水量(mm/min)，I：降水量以外の流入成分(mm/min)，E：蒸発散量(mm/min)，O：取水量(mm/min)，q_R：合流式下水道による流域外への雨水排水量(mm/min)，$q_{R_{max}}$：最大雨水排水量(mm/min)，q_l：地下水関連損失量(mm/min)，k_1，k_2，k_3，p_1，p_2，z，α：モデルパラメータ．

式(1)，(2)を 1 階の非線形 2 元連立常微分方程式に変換すると，この微分方程式は 7 つのパラメータが既知であれば，種々の数値解法を利用して解くことができる．本研究ではそのうち比較的計算が速く，精度も高い Runge-Kutta-Gill 法を用いた．計算の結果$(Q+q_R)$の値が算定されると，式(4)によりq_Rが求まるので，結果として河川流出量Qを得ることができる．

2.2 進化的計算手法

進化的計算手法とは，生物学的機構をヒントとして解を進化させる最適化手法である．生物学的機構の例として，群知能や交配，突然変異，淘汰などの，生物が行ってきた戦略が採用される．

(1) PSO

PSO は Kennedy と Eberhart によって提案された最適化手法であり，鳥や魚の群れの振る舞いに着想を得たものである．生物に見立てた粒子群が，目的関数と位置の情報を共有しながら解空間を探索する．m個の粒子

$(i = 1, \ldots, m)$は，k回目の位置x_i^kと速度v_i^kを用いて$k+1$回目の更新を行って解を改良する．

(2) Cuckoo Search

CS はカッコウの托卵という特殊な生態を模した手法であり，Yang と Deb によって提案された．托卵とは，カッコウの親鳥が他の種の鳥の巣に卵を産み付け，先に孵ったひなが巣を乗っ取るものである．宿主の鳥はカッコウの卵を発見して排除することもあり，CS ではこれらの行動がアルゴリズム化されている．より良い巣を発見することは，より良い解を探索することに相当する．

(3) SCE-UA 法

Duan らによって提案された SCE-UA 法は流出解析モデルのパラメータ最適化を目的に開発され，シンプレックス法, ランダム探索, 競争進化, 集団混合の概念を組み合わせたアルゴリズムを持つ大域的探索法である．モデルパラメータ同定手法として強力かつ効率的な手法であることが示されており，都市貯留関数モデルに対しても適用実績がある．

2.3 入力データ

(1) 仮想流域

クリーブランド型の降雨強度式を用い，東京管区気象台の 20 年確率の定数を入力して中央集中型で 180 分の仮想の降雨を作成した．これを都市貯留関数モデルに入力して，真値パラメータを$k_1 = 50$, $k_2 = 500$, $k_3 = 0.005$, $p_1 = 0.4$, $p_2 = 0.3$, $z = 5$, $\alpha = 0.5$とすることで計算流量を得た．この計算流量と仮想の降雨を入力とし，3 つの進化的計算手法によって真値パラメータの同定を行った．

(2) 実流域

対象流域は，典型的な都市中小河川である神田川に合流する善福寺川の，西田端橋上流域とした．水位観測点である西田端橋では，平成 26 年度に東京都が委託して流量観測が実施されており，高精度な水位流量曲線が作成されている．対象期間は 2013 年から 2015 年とし，60 分最大雨量の上位から 8 イベントを抽出した．

３．研究結果

目的関数は，入力した流量と都市貯留関数モデルによる計算流量の平方根平均二乗誤差(RMSE)とし，この値の最小化を行った．パラメータ同定を行う際，プログラム上で発生させる乱数によって結果が異なる場合があるため，各手法，各イベントに対して，乱数を変化させて 100 回ずつ計算を行った．なお，PSO・CS・SCE-UA 法にはそれぞれ設定すべきパラメータが存在する．PSO・CS については，7 変数のベンチマーク関数(Ackley 関数・Rastrigin 関数)を上手く最適化した設定とし，SCE-UA 法については先行研究で都市貯留関数モデルのパラメータ同定に用いられてきた設定とした．その結果，PSO・CS の目的関数評価回数は 150,000 回，SCE-UA 法は 25,000 回前後となった．パラメータ同定にかかる計算時間は，目的関数の評価回数に比例するため，SCE-UA 法のそれは PSO・CS と比較して 15%程度であった．

3.1 仮想流域

真値パラメータを同定した場合，$\mathrm{RMSE} = 0$となるが，本研究で使用した手法は近似解を求めるものであるため，$\mathrm{RMSE} \leq 10^{-4}$となったものが真値を同定したとする．その結果，PSO は 0%，CS は 62%，SCE-UA 法は 99%の確率で真値を同定した．

3.2 実流域

実流域データは真値パラメータが不明なため，目的関数値によって各手法の探索性能を判定する．その結果，CS が頑健な結果を示した．PSO はイベント 1,3,7 で，SCE-UA 法はイベント 1,6 で準最適解となる解を多く同定した．

４．結論

本研究では，進化的計算手法である PSO・CS・SCE-UA 法を用いて，都市貯留関数モデルのパラメータ真値が明らかな仮想流域と真値が不明な実流域データに対し，パラメータ同定を行った．その結果，仮想流域では SCE-UA 法，実流域では CS が高い探索性能を示した．SCE-UA 法の計算時間は PSO・CS の 15%程度であったことから，乱数を変化させて複数回 SCE-UA 法を実行するという運用法が実用的と考えられる．

目　　次

第１章

序　論

第１章　序論

１－１　研究の背景と目的

近年，計算機器の進化により，様々な計算を高速で実行することが可能となっている．そこで，ある制約条件をもとにして，目的関数の最大値または最小値を与える解を探索する，最適化手法の開発が数多く行われている．最適化手法は，経済学をはじめとした様々な分野で重要なものとなっている．工学分野でも必須のツールであり，設計や計画などで使用されている．水文学の分野では，これまでに治水・利水やその他の現象解明のために数多くの降雨流出解析モデルが開発されており，これらのモデルを規定する，パラメータの最適値を探索する必要がある．降雨流出解析モデルのパラメータは，対象とする流域や降雨の特性などに応じて最適値が異なるものであり，これらをしらみつぶしに計算することは非効率である．そこで，非線形関数の最適化問題を扱うことの出来る計算知能を用いた，パラメータの探索が広く行われている．最適なパラメータを探索する際，準最適解と呼ばれる局所解に収束することがあり，局所解を回避して，最適値を求められる最適化手法を選択することは，パラメータ同定において最重要課題である．

本研究で対象とするモデルは都市貯留関数モデル [1]である．都市貯留関数モデルは，有効雨量の算定やハイドログラフの流出成分の分離作業を必要とせず，観測雨量と観測流量を直接用い，都市特有の流出機構を考慮して全流出成分を概念的に組み込んだ二価の貯留関数モデルである．このモデルは都市中小河川のハイドログラフを良好に再現できることが確認されている．都市貯留関数モデルは７つのパラメータを持っており，流域や降雨イベント等によって最適なパラメータは異なる．これまでに，大域的探索法である SCE-UA（Shuffled Complex Evolution method – University of Arizona）法 [2]を用いた都市貯留関数モデルのパラメータ同定が行われてきた [1]が，その他の最適化手法を適用した事例は見られない．そこで，本研究では他の最適化手法を用いた都市貯留関数モデルのパラメータ同定について議論する．なお，対象とするデータについては，まず特定の流域を念頭に置いた仮想流域を設定し，その後，代表的な都市中小河川である善福寺川の，2013 年から 2015 年の洪水イベントを抽出した．これらの仮想流域と実流域に対してパラメータ同定を実施した．

これまでに数多くの最適化手法が提案されているが，そのなかでも進化的計算手法（Evolutionary Computation）とは，進化の生物学的な機構に着想を得た手法である．交配や突然変異，自然淘汰といった概念をアルゴリズム化したものもあり，その過程で解となる個体が進化することを目的とするものである．これらは進化的アルゴリズムと呼ばれる．一方で，群知能と呼ばれる進化的計算手法も存在する．群知能は群れをなす生物の振る舞いを模したアルゴリズムをもっており，蟻のコロニー形成にヒントを得たものもある．本研究では，群知能の代表的な手法である粒子群最適化（PSO : Particle Swarm Optimization）[3]と，比較的最近開発されたカッコウ探索（CS : Cuckoo Search）[4]を用いて都市貯留関数モデルの

パラメータ同定を行う．PSO は鳥や魚の振る舞いをアルゴリズム化したもので，群れの一部がエサや逃げ道を発見した場合，その情報が全体に共有されて，最適な場所を発見するというものである．一方，CS は，カッコウの独特な生態を模したものである．カッコウは托卵を行うことで知られているが，CS ではこの卵が問題の解にあたる．より良い解を探索するために，既存の巣に置かれた卵が，目的関数の悪い他の卵を棄てるといったアルゴリズムが組み込まれている．これら PSO と CS 以外に，都市貯留関数モデルのパラメータ同定の実績がある SCE-UA 法についても比較を行う．なお SCE-UA 法は進化的アルゴリズムに分類される最適化手法であり，シンプレックス法・ランダム探索・競争進化・集団混合のアルゴリズムを持っている．加えて，SCE-UA 法は水文モデルのパラメータ同定を念頭に開発されたものである．

本研究では，上述した進化的計算手法である SCE-UA 法・PSO・CS を用いて，仮想流域と善福寺川の実流域のデータを対象に，都市貯留関数モデルのパラメータ同定を実施し，各手法の探索性能について比較検討を行った．

１－２　本論文の構成

本論文は全５章から構成されており，以下に各章の概要を述べる．

第１章は序論であり，本研究の背景と目的について述べ，本論文の構成について示した．

第２章では，本研究で使用した最適化手法の SCE-UA 法・PSO・CS に概説した．またそれぞれ最適化手法が適切に最適化を行っているかを確認するために，ベンチマーク関数を用いてその性能を検証した．ベンチマーク関数には，Ackley 関数と Rastrigin 関数を用いた．いずれの関数も多峰性の形状を有しており，最適化手法のベンチマークとしてよく引用されているものである．

第３章では，都市貯留関数モデルについて記した．まず都市貯留関数モデルの概要と数式について示し，解法について述べた．解法にはルンゲクッタギル(Runge-Kutta-Gill)法を用いた．その後，対象流域とした善福寺川の概要と，流量観測が行われている地点について述べた．そして対象とした期間の 2013 年から 2015 年に観測された洪水イベントについて記述した．

第４章では，３つの最適化手法を用いて実施したパラメータ同定について述べた．まず，パラメータの真値が明らかな仮想流域に適用し，その後実流域の計算を行った．

第５章は結論であり，本研究で得られた知見をまとめ，総括を述べた．

第２章

進化的計算手法

第２章　進化的計算手法

進化的計算（Evolutionary Computation）手法とは，組合せ最適化問題等を解くことのでる，人工知能の一分野である．厳密に表現すると，計算知能であるとされ，計算機科学の学問に分類される．進化的計算では，生物学的機構をヒントとして，解を進化させるということが肝である．ここで，生物学的機構の例として，群れの動きに着目した「群知能」，カッコウの「托卵」という生態，その他，「交配」，「突然変異」，「淘汰」などといった概念がある．

本研究では，SCE-UA 法，粒子群最適化（PSO：Particle Swarm Optimization），カッコウ探索（CS：Cuckoo Search）の３手法を採用し，都市貯留関数モデルのパラメータ同定を行う．いずれの手法もメタヒューリスティックなものとされており，様々な問題に適用することが可能である．なお，これらの手法は厳密解を求めるものではなく，近似解を求めるものである．

２－１　SCE-UA 法(Shuffled Complex Evolution method – University of Arizona)

誤差評価関数の最小化においてはこれまで局所的探索法が多く用いられてきた．局所的探索法は探索点近傍の関数応答面の勾配等をもとに関数の評価値が小さくなる方向に探索点を移動させていく方法である．この方法では複数の極小点が存在する場合には最小点が求められないため，事前に解の近似値を把握しておく必要がある．一方，遺伝的アルゴリズムや SCE-UA（Shuffled Complex Evolution method - University of Arizona）法等の大域的探索法は探索空間全域の中から誤差評価関数の値を最小にする点を探索するものであり，事前に解の近似値を把握していなくても解を求めることができる．

本研究では，モデルパラメータの同定に大域的探索法のひとつである SCE-UA 法を用いる．SCE-UA 法は Duan et al.[2]によって提案された手法であり，シンプレックス法，ランダム探索，競争進化，集団混合の概念を組み合わせたアルゴリズムを持つ．SCE-UA 法は多くの研究においてタンクモデル等の流出モデルにおけるパラメータ同定手法として強力かつ効率的な自動最適化手法であることが示されている[5),6)]．さらに森永ら[7)]は二価の貯留関数モデルを用いて既知パラメータによって発生させた模擬データに対して SCE-UA 法によるパラメータ同定を行い，精度良く同定が行われることを報告している．

図 2-1-1 に SCE-UA 法のアルゴリズムを示す．SCE-UA 法では**図 2-1-2** に示す CCE（Competitive Complex Evolution）アルゴリズムによって，解を進化させる．

1. 決定変数$nopt$，集団数nc，集団内の個体数ncpを設定
2. 初期世代をランダムに生成
3. **while** ($generation$<最大世代数)
4. 　全ての個体の目的関数を計算し，評価順に p 個の集団に振り分ける
5. 　各集団を CCE(Competitive Complex Evolution)アルゴリズムにより進化させる
6. **end while**

図 2-1-1　SCE-UA 法の擬似コード

i. 各集団から，適合度に応じた選択確率によってq個の個体を進化用に選択する．
ii. q点から，最悪点Uを除いた個体の重心Gを求める．
iii. Gに関するUの対象点Rを求める．Rが探索範囲外なら突然変異．適合度が$f_R > f_U$であれば v.へ．$f_R \leq f_U$なら iv.へ．
iv. GとUの中点Cを求めて，$f_C > f_U$ならばUとCを置き換えて v.へ．$f_C \leq f_U$ならば突然変異．
v. 進化用の個体を集団に戻し，適合度準に並べる．
vi. ii.から v.をα回繰り返す．
vii. q点を集団に戻し，集団内の個体を評価．
viii. i.から vii.をβ回繰り返す．

図 2-1-2　CCE アルゴリズム

２－２　粒子群最適化（PSO：Particle Swarm Optimization）

粒子群最適化（PSO：Particle Swarm Optimization）は，Kennedy と Eberhart[3]によって提案された最適化手法である．群知能（swarm intelligence）の一種であり，電気システムの最適化[8]やタンクモデルのパラメータ同定[9]などに使用されている．魚や鳥の群れが，エサや逃げ道を探して動き回る振る舞いに着目した手法である．生物に見立てた粒子は，適合度（目的関数）や位置の情報を共有して並列的に動作し，複雑な挙動を表現する．時点$k+1$の粒子の速度と位置は，kの情報をもとに順次更新される．式(2-1),(2-2)に，それぞれ速度と位置の更新式を示す．また**図 2-2-1** に PSO のアルゴリズムを示す．

$$v_i^{k+1} = wv_i^k + c_1 r_1\left(pbest_i^k - x_i^k\right) + c_2 r_2\left(gbest_i^k - x_i^k\right) \qquad (2\text{-}1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad (2\text{-}2)$$

ここに，x_{Pbesti}^k：粒子iのk回目の探索までの最良位置，x_{Gbesti}^k：粒子群全体でのk回目の探索までの最良位置，w：粒子の慣性パラメータ，c_1：粒子iの既往最良位置へ戻ろうとする強度パラメータ，c_2：群全体の最良位置へ近づこうとする強度パラメータ，r_1, r_2：[0,1]の一様乱数．

1. 決定変数の個数$nopt$，粒子数m，パラメータw, c_1, c_2を決定する．
2. 繰り返しの初期として，各粒子の位置x_i及び速度v_iをランダムに設定する．
3. **while**（k<最大繰り返し回数k_{max}） or （その他終了基準）
4. 　全ての粒子の目的関数$f(x_i^k)$を計算する．
5. 　**if** $f(x_i^k) > f(x_{Gbesti}^k)$
6. 　　$f\left(x_{Gbesti}^k\right) = f(x_i^k)$
7. 　**end if**
8. 　**if** $f(x_i^k) > f(x_{Pbesti}^k)$
9. 　　$f\left(x_{Pbesti}^k\right) = f(x_i^k)$
10. 　**end if**
11. 　各粒子の$k+1$時点での速度と位置を式(2-1),(2-2)により更新する．
12. **end while**

図 2-2-1　PSO の擬似コード

PSO の MATLAB コードは，ツールボックスを公開している Sandeep Solanki 氏の「psoToolbox」[10]を元に作成した．

２－３　カッコウ探索（CS：Cuckoo Search）

カッコウ探索（CS：Cuckoo Search）は Yang，Deb[4]によって開発された最適化手法である．カッコウの托卵という習性に着想を得た手法であり，ランダムウォークの一種である Lévy Flightsを組み合わせることで，PSO よりも頑強な探索性能を示すとされている．**図 2-3-1** に Cuckoo Search の擬似コードを示す.

托卵とは, カッコウの親が他の種の鳥の巣に卵を産み付け, 代わりに育ててもらうことで, 種の繁栄を目指すものである．多くの場合，カッコウの卵は他の種の卵よりも先に孵化し，自分以外の卵を蹴落とすという．Cuckoo Search のアルゴリズムでは，適合度の高い卵が残るように，成績の悪い卵が破棄される．

Lévy Flightsとは鳥や昆虫などがエサなどを探して動き回る際，ランダムに動き回るという動作を数式化したものである．Lévy Flightsによって周囲の探索を行うことで，より高い確率でエサなどに辿り着けるとされている．

---

1. 目的関数$f(\boldsymbol{x})$，$\boldsymbol{x} = (x_1, \dots, x_d)^T$
2. 初期の巣の個体群nを作成 $x_i (i = 1,2, \dots, n)$
3. **while** (t <最大繰返し回数t_{max}) or (その他終了基準)
4. 　Lévy Flightsを利用して，ランダムに選んだ巣に新たなカッコウの卵$\boldsymbol{x}_i^t$を作る
5. 　x_i^tを評価する f_i
6. 　ランダムに卵$\boldsymbol{x}_j^{t-1}$を選択する
7. 　**if** ($f_i > f_j$)
8. 　　$\boldsymbol{x}_j^{t-1}$を新しい解$\boldsymbol{x}_i^t$と取り替える
9. 　**end**
10. 　一定確率(p_a)で成績の悪い巣を破棄し，Lévy Flightsによって新しい巣を作る
11. 　巣を並び替え，最良の巣を保持する
12. **end while**

図 2-3-1　Cuckoo Search の擬似コード

CS の MATLAB コードは, 開発者である Yang の公開する「Cuckoo Search (CS) Algorithm」[11]を使用し，目的関数等を本研究用に変更した.

２－４　ベンチマーク関数

最適化手法の妥当性を検討する為に，ベンチマーク関数というものが存在する．これらは式が与えられ，最適解と，そのときの目的関数値が明らかになっているものである．本研究では, ベンチマーク関数としてよく引用される, Ackley 関数及び Rastrigin 関数を採用した．なお，変数の数は都市貯留関数モデルと同じ 7 つとした．

2-4-1　Ackley 関数

多峰性を有するベンチマーク関数であり, Rastrigin 関数と並んで非常によく使用されるベンチマーク関数である. 式(2-3)に Ackley 関数を示す. 大域的探索解は$x^O = (0, \dots, 0)$であり，その際の目的関数値は$f(x^O) = 0$である．**図 2-4-1** に 2 変数の Ackley 関数を示す.

$$\min_{x} \ -20\exp\left\{-0.2\sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}\right\} - \exp\left\{\frac{1}{N}\sum_{n=1}^{N}\cos(2\pi x_n)\right\} + e \qquad (2\text{-}3)$$

$$\mathrm{S} = \{x| -30 < x_n < 30\}, n = 1, \dots, N$$

2-4-2　Rastrigin 関数

式(2-4)に Rastrigin 関数を示す．大域的探索解は$x^O = (0, \dots, 0)$であり，その際の目的関数値は$f(x^O) = 0$である．**図 2-4-2** に 2 変数の Rastrigin 関数を示す.

$$\min_{x} \sum_{n=1}^{N}\{x_n^2 - 10\cos(2\pi x_n) + 10\} \qquad (2\text{-}4)$$

$$\mathrm{S} = \{x| -5 < x_n < 5\}, n = 1, \dots, N$$

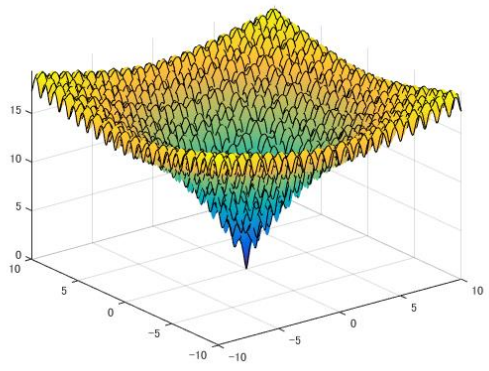

図 2-4-1　Ackley 関数（2 変数）

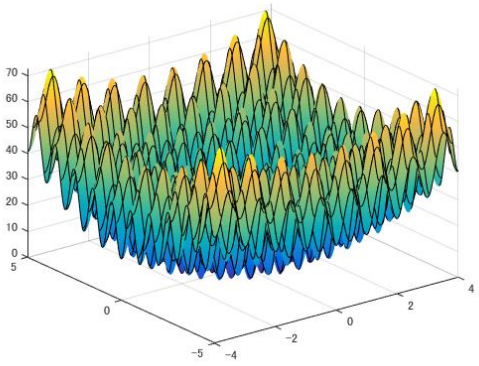

図 2-4-2　Rastrigin 関数（2 変数）

2-4-3 実行結果

SCE-UA 法，PSO，CS ともに，ベンチマーク関数である 7 変数の Ackley 関数・Rastrigin 関数を最適化することができた．その際の計算設定を以下の**表 2-4** に示す.

最適化手法が適切な探索を行うためには本来，適用する最適化問題に対して計算設定を変えて，最適なものを探すことが求められる．しかし PSO の場合は粒子を操作するパラメータの数が多く，都市貯留関数モデルのパラメータ同定の為に最適な設定を探すということが煩雑であったため，ツールボックスの初期設定値であり，かつベンチマーク関数を上手く最適化することの出来た**下表**の設定を採用した.

CS については巣の数nの初期値が 25 であったが，開発者 Yang らによるとnは局所解の数以上に大きければ最適値を同定することができるとあったため，便宜的にn =30 とした.

SCE-UA 法では，開発者 Duan らの推奨とする計算設定を採用し，繰り返し計算回数にあたる最大世代数は，これまでに都市貯留関数モデルのパラメータ同定に用いられてきた 50 世代を採用した.

計算終了条件については，SCE-UA 法による都市貯留関数モデルのパラメータ同定の実績を鑑み，50 世代とした．ただし，7 変数の Rastrigin 関数の収束には 200 世代程度を要したことを付記する．PSO の計算終了条件は$k_{max} = 3000$，Cuckoo Search は$t_{max} = 2500$として，ともに目的関数の評価回数は 150,000 回に揃えた．なお SCE-UA 法の目的関数の評価回数は計算の試行によって上下するが，概ね 25,000 回前後であった．パラメータ同定の計算時間は，目的関数の評価回数に比例しているため，SCE-UA 法の計算時間は PSO・CS と比較して 15%程度であった.

表 2-4 各手法の計算設定

SCE-UA 法	PSO	Cuckoo Search
$nc = 20$	$n = 50$	$n = 30$
$ncp = 2 * nopt + 1$	$w = 0.0004$	$p_a = 0.25$
$nscp = nopt + 1$	$c_1 = 1.2$	
$ntp = nc * ncp$	$c_2 = 0.012$	

第３章

都市貯留関数モデル

第３章　都市貯留関数モデル

３－１　都市貯留関数モデルの概要

都市貯留関数モデルとは，合流式下水道による流域外への排水など都市特有の流出機構を考慮し，全流出成分を概念的に組み込むことで事前の有効雨量や流出成分の分離作業が不要となるモデルである．

本モデルは，まず都市流域の流出機構を考慮し，流域の総貯留高に関係する流出成分を考えている．流域内へ入ってくる成分としては降水に加え，都市特有の流入成分として下水処理場からの放流水，水道管からの漏水，環境用水等の導水，灌水等の他，流域周辺からの地下水流入などもある．都市特有の流入成分として，下水処理場からの放流水が流入する河川は，その河川流量に占める下水処理水の割合は大きなものと推察される．水道管からの漏水は，近年になりその量が少なくなっているものの，都市域の地下水涵養に大きく影響を与えている．また，多くの都市河川では河川水量の確保や水質改善のため，他の河川水，下水処理水，余剰地下水，深層地下水等が環境用水として導入されている．

一方，流域外へ出て行く成分としては河川表流水，下水道による流域外への雨水排出，地下水に関連した損失とみなされる流出（伏流水，流域外への地下水流出，深層への浸透等），河川や地下水からの取水，蒸発散等が考えられる．分流式下水道が普及している地域では下水道に流入した雨水は全て河川に放流されるが，合流式下水道が普及している地域では雨水の一部が合流式下水道により流域外へ運ばれることとなる．このため合流式下水道が普及している地域では河川への流出のみならず，下水道による流域外への雨水排出も流域からの流出として考える必要がある．合流式下水道が整備された都市の数は，1999 年度において全国で 192 都市存在し，この区域の人口は日本の総人口の約 30%を占める．なお，分流式下水道が普及している場合には通常，下水道による流域外への雨水排出はないことから，流域からの流出は河川への流出のみを取り扱うこととなる．

以上で述べた流域の総貯留高 s (mm)の流入出概念図を**図 3-1** に示す．すなわち流域内に入ってくる成分としては，降水量 R (mm/min)および降水量以外の流入成分(都市特有の流入量・流域外からの地下水流入量) I (mm/min)である．流域外へ出て行く成分は，河川流出量 Q (mm/min) ，合流式下水道による流域外への雨水排水量 q_R (mm/min)，蒸発散量 E (mm/min)，取水量 O (mm/min)，地下水関連損失量(伏流水，流域外への地下水流出，深層への浸透等) q_l (mm/min)である．なお，流域の総貯留高 s には直接関係しないが，合流式下水道による流域外への雨水排水量 q_R と家庭からの汚水量等 q_w を合わせた q_s が合流式下水道により流域外に排出される水量となる．さらに，ここでは降雨終了後の河川流出量の逓減部を良好に再現するため，地下水浸透などの流入出機構を付け加えたタンクモデルである SMPT モデル[12]を参考に地下水関連損失量の浸透孔高 z (mm)を導入した．**図 3-1** においては，対象流域内に下水処理場が含まれていない場合を想定しているが，下水処理場が含まれている場合には，流域

外からの流入量を考慮したり，汚水処理水を都市特有の流入量とみなしたりする等の工夫が必要となる．以上より本論分では，**図 3-1** に示した都市特有の総貯留高 s の関係を二価関数の貯留関数モデルを用いて解析を行う．

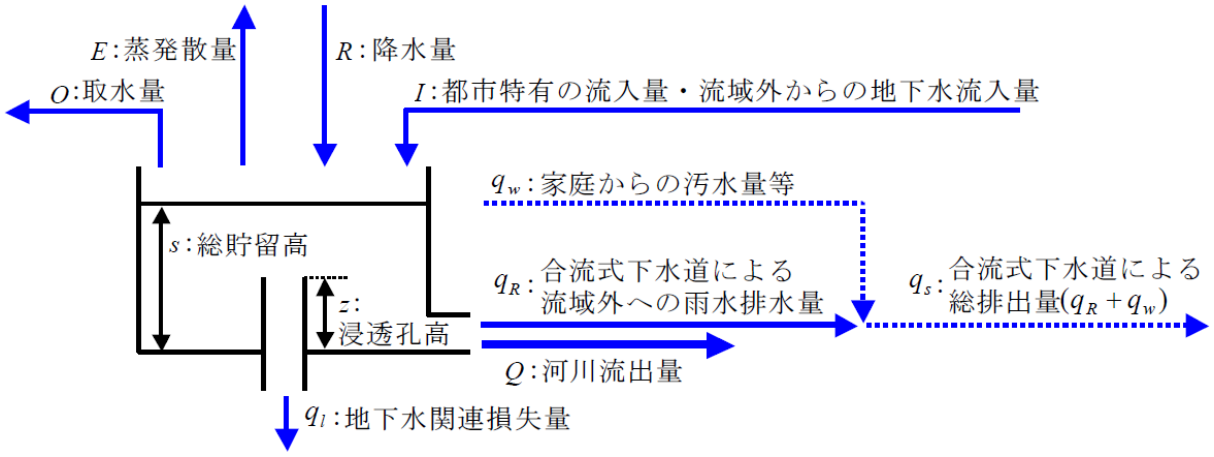

図 3-1　総貯留高sの流入出概念図

流域からの流出量(河川流出量 Q と合流式下水道による流域外への雨水排水量 q_R の合計)と総貯留高 s の関係を式(3-1)で，またその連続性を式(3-2)で表す．

$$s = k_1 (Q + q_R)^{p_1} + k_2 \frac{d}{dt}\left\{(Q + q_R)^{p_2}\right\} \tag{3-1}$$

$$\frac{ds}{dt} = R + I - E - O - Q - q_R - q_l \tag{3-2}$$

ここに，t：時間(min)，k_1 ，k_2 ，p_1 ，p_2 ：モデルパラメータ．

また，伏流水や地下水等として流域外へ出て行く水量は流域の総貯留高との関係が大きいと考えられることから，地下水関連損失量 q_l は総貯留高 s と浸透孔高 z の差に比例すると考え，次式で表す．ただし，s が z より小さい場合は，q_l は 0 となる．

$$q_l = \begin{cases} k_3 (s - z) & (s \geq z) \\ 0 & (s < z) \end{cases} \tag{3-3}$$

ここに，k_3：モデルパラメータ．

ここで，次式の変数変換を行い，

$$x_1 = (Q + q_R)^{p_2} \tag{3-4}$$

$$x_2 = \frac{d}{dt}\left\{(Q + q_R)^{p_2}\right\} \tag{3-5}$$

これらを式(3-1)に代入し，t で微分すると次式が得られる．

$$\frac{ds}{dt} = k_1 \frac{p_1}{p_2} x_1^{(p_1/p_2^{-1})} x_2 + k_2 \frac{d}{dt} x_2 \tag{3-6}$$

$s \geqq z$ のとき，式(3-3)の s に式(3-1)を代入し，式(3-4)，(3-5)の関係を用いると q_l は次式となる．

$$q_l = k_1 k_3 x_1^{(p_1/p_2^{-1})} + k_2 k_3 x_2 - k_3 z \tag{3-7}$$

式(3-2)に式(3-7)を代入し，式(3-4)，(3-5)の関係を用いると次式となる．

$$\frac{ds}{dt} = R + I - O - x_1^{(1/p_2)} - k_1 k_3 x_1^{(p_1/p2)} - k_2 k_3 x_2 + k_3 z \tag{3-8}$$

そして，式(3-6)，式(3-8)より，x_2 に関する一階の常微分方程式は式(3-9a)得られる．なお，$s < z$ のときは同様の過程により x_2 に関する一階の常微分方程式は(3-9b)となる．

$$\begin{cases} \dfrac{dx_2}{dt} = -(k_1/k_2)(p_1/p_2) x_1^{(p_1/p_2^{-1})} x_2 - (1/k_2) x_1^{(1/p_2)} \\ \qquad\qquad - (k_1 k_3/k_2) x_1^{(p_1/p_2)} - k_3 x_2 + (1/k_2)(R + I - E - O + k_3 z) \end{cases} \tag{3-9a}$$

$$\frac{dx_2}{dt} = -(k_1/k_2)(p_1/p_2) x_1^{(p_1/p_2^{-1})} x_2 - (1/k_2) x_1^{(1/p_2)} + (1/k_2)(R + I - E - O) \tag{3-9b}$$

また，x_1 に関する一階常微分方程式は式(3-4)，(3-5)より次式の関係となるので，

$$\frac{dx_1}{dt} = x_2 \tag{3-10}$$

式(3-9)と式(3-10)の連立常微分方程式を数値的に解くことで，河川流出量 Q と合流式下水道による流域外への雨水排水量 q_R の合計値を逐次求めることができる．なお，これらの連立常微分方程式の解放については Runge-Kutta-Gill 法を用いて計算を行った．

ここで，流域からの流出量$(Q+q_R)$と合流式下水道による流域外への雨水排水量 q_R の関係について考える．降雨直前においては下水管を流れる雨水はないため q_R=0 であり，流域からの流出量は河川流出量 Q_o のみである．降雨時においては下水管に流入した雨水は下水管内の水量が下水処理場へ送られる水量より小さい場合，雨水吐の越流堰によって河川への放流が阻止され下水道により流域外に運ばれる．そして，下水管内の水量が遮集量に達した場合，下水管に流入した雨水は合流式下水道により流域外に運ばれると同時に越流堰を溢流して流域内の河川へも放流される．合流式下水道による流域外への雨水排水量は下水道の流下能力によって制約を受けることからその最大雨水排水量 q_{Rmax} を超えることはできない．豪雨時において合流式下水道による流域外への雨水排水量が q_{Rmax} に達した場合，流域からの流出量から q_{Rmax} を差し引いた量が河川流出量 Q となる．なお，q_{Rmax} に家庭からの汚水量等 q_w をあわせた水量 q_s が遮集量に相当し，東京都の下水道整備では遮集量を時間最大汚水量の 3 倍としている．ここで，都市貯留関数モデルでは流域からの流出量$(Q+q_R)$と q_R の関係を簡単のため**図 3-2** のように線形と仮定している．すなわち，降雨直前における河川流出量を初期河川流出量 Q_o(mm/min)とし，線形関係の傾き α を下水道排出係数と呼び，この**図 3-2** の関係は次式で表

される．

$$q_R = \begin{cases} \alpha(Q + q_R + Q_o) & (\alpha(Q + q_R + Q_o) < q_{R\max}) \\ q_{R\max} & (\alpha(Q + q_R + Q_o) \geq q_{R\max}) \end{cases} \tag{3-11}$$

都市貯留関数モデルにおける同定すべき未知パラメータは k_1，k_2，k_3，p_1，p_2，z，α の 7 個である．パラメータ値が決定すれば，式(3-9)と式(3-10)を解くことより流域からの流出量 $(Q+q_R)$の値が算定され，また式(3-11)より q_R が求まるので，結果として河川流出量 Q を得ることができる．

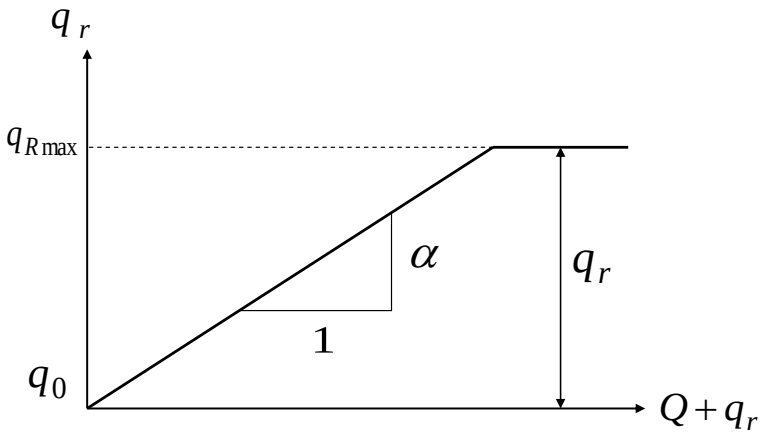

図 3-2　流域からの流出量と q_R の関係

３－２　仮想流域データの作成

進化的計算手法である SCE-UA 法，PSO，Cuckoo Search によって都市貯留関数モデルのパラメータ同定を行うにあたって，はじめに既知パラメータに対して計算を行うことで，それらの探索性能を検討する．

3-2-1 降雨データ

降雨データは，式(3-12)に示すクリーブランド型の降雨強度式により作成した．

$$I = \frac{a}{t^n + b} \tag{3-12}$$

ここで，I：降雨強度(mm/hr)，t：降雨継続時間(分)，a, b, n：定数．

なお定数については東京管区気象台（統計期間：昭和 2 年～平成 22 年，確率雨量算定方法：Gumbel 分布）で，20 年確率のものを使用した．また t は 180 分とし，中央集中型の降雨分布に配置することで，以下**図 3-3** のようなハイエトグラフを得た．

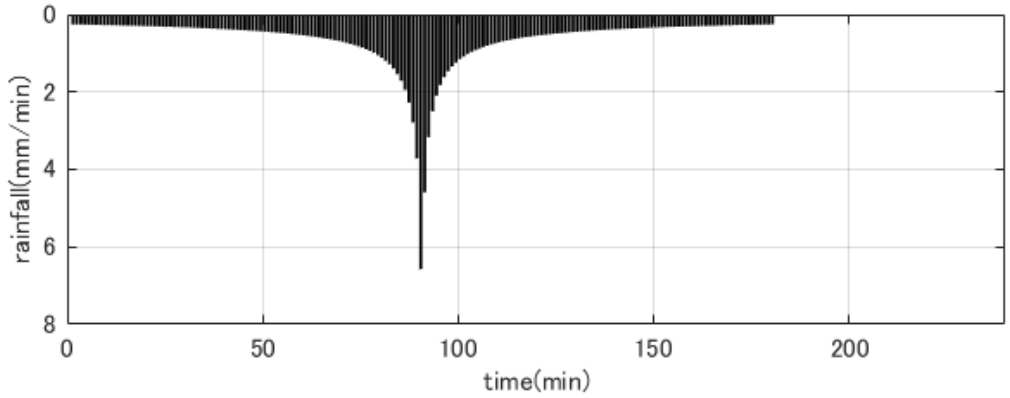

図 3-3　仮想降雨

3-2-2 真値パラメータ

真値パラメータは**表 3-1** のように設定した．この値は先行研究である神田川上流域のパラメータ [1)]を参考に設定したものである．

表 3-1　真値パラメータ

k_1	k_2	k_3	p_1	p_2	z	α
50	500	0.005	0.4	0.3	5	0.5

以上の真値パラメータと仮想の降雨データを都市貯留関数モデルに入力することで，**図 3-4** のようなハイドログラフを得た．パラメータ同定では，SCE-UA 法，PSO，Cuckoo Search に仮想の降雨データと流量データを入力することで，真値パラメータの同定を試みた．

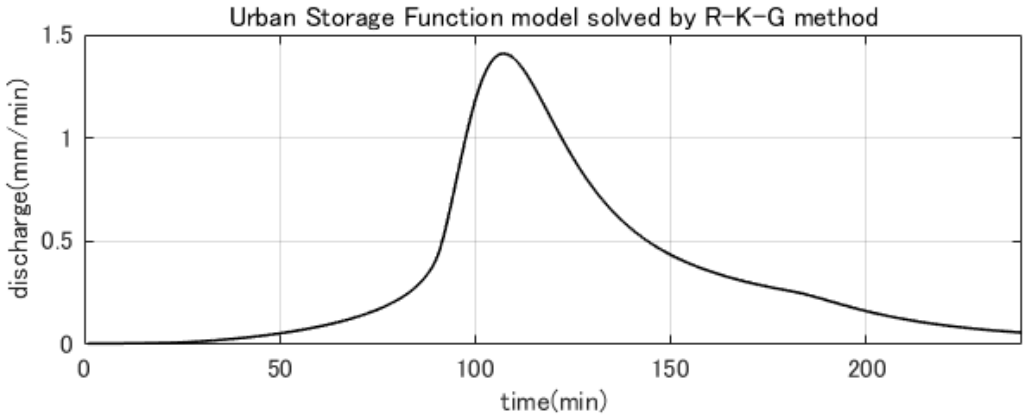

図 3-4　仮想降雨と真値パラメータによるハイドログラフ

都市貯留関数モデルでは，都市貯留関数モデルでは，対象流域における都市特有の流入出成分を算定し，入力する必要がある．仮想流域であるため，次のようにそれらを設定した．対象流域における降水量以外の流入成分 I，取水量 O および蒸発散量 E はいずれも 0mm/min とした．また，都市貯留関数モデルのパラメータ $q_{R\max}$ は 0.005 mm/min とした．

３－３　実流域データの作成

実流域データに対してパラメータ同定を行うにあたって，善福寺川西田端橋上流域を対象流域とすることとした．善福寺川は東京において典型的な都市中小河川でもあり，東京都が委託した「善福寺川流量観測委託」[13]（平成 26 年）によって，高精度な水位流量曲線が作成されていることから，本研究の対象流域とした．

3-3-1 善福寺川の概要と対象流域の設定

善福寺川は神田川の支流の一つであり，杉並区の善福寺池を水源としている．上流端は美濃山橋で，下流端は神田川への合流点で，延長は 10.5km，流域面積は 18.3km² である．**図 3-5** に善福寺川が属する荒川水系の河川を示す．

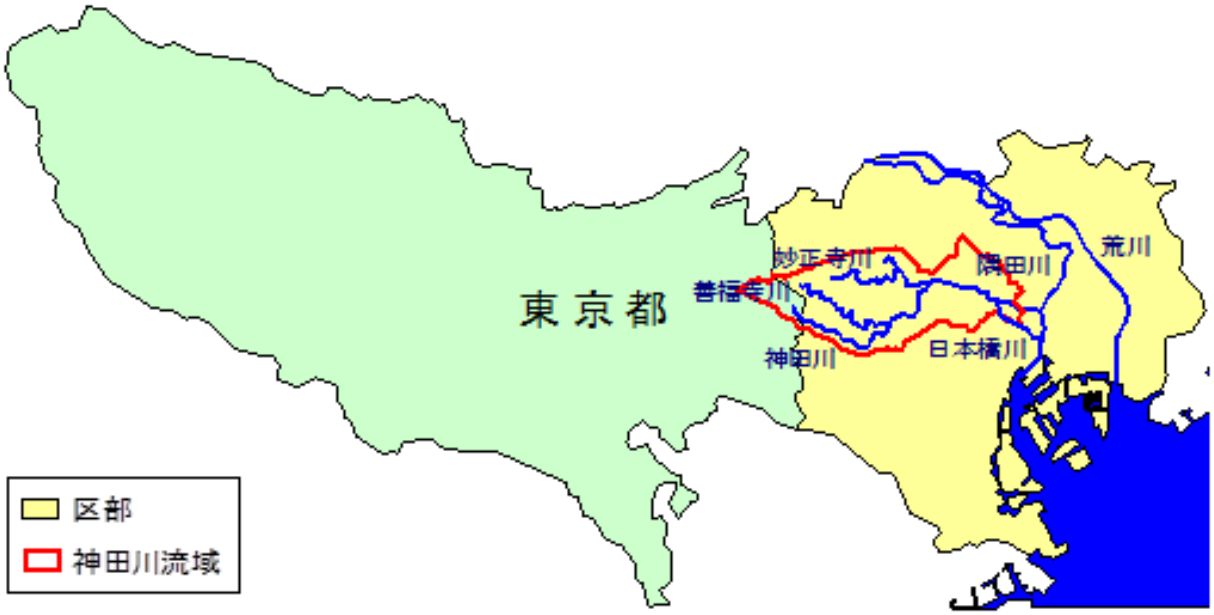

図 3-5　善福寺川の位置

対象とした水位観測地点である西田端橋周辺には，ティーセン法により流域平均雨量を算出するにあたって影響のある 11 箇所の雨量観測所が存在する．**図 3-6** に西田端橋水位観測所および周辺の雨量観測所を示す．

善福寺川始点から西田端橋までの河川延長は 4.3km，対象流域とした西田端橋上流域の流域面積は 14.4km² である．ここで，西田端橋上流域と周辺の雨量観測所の位置から算定したティーセン係数を**表 3-2** に示す．

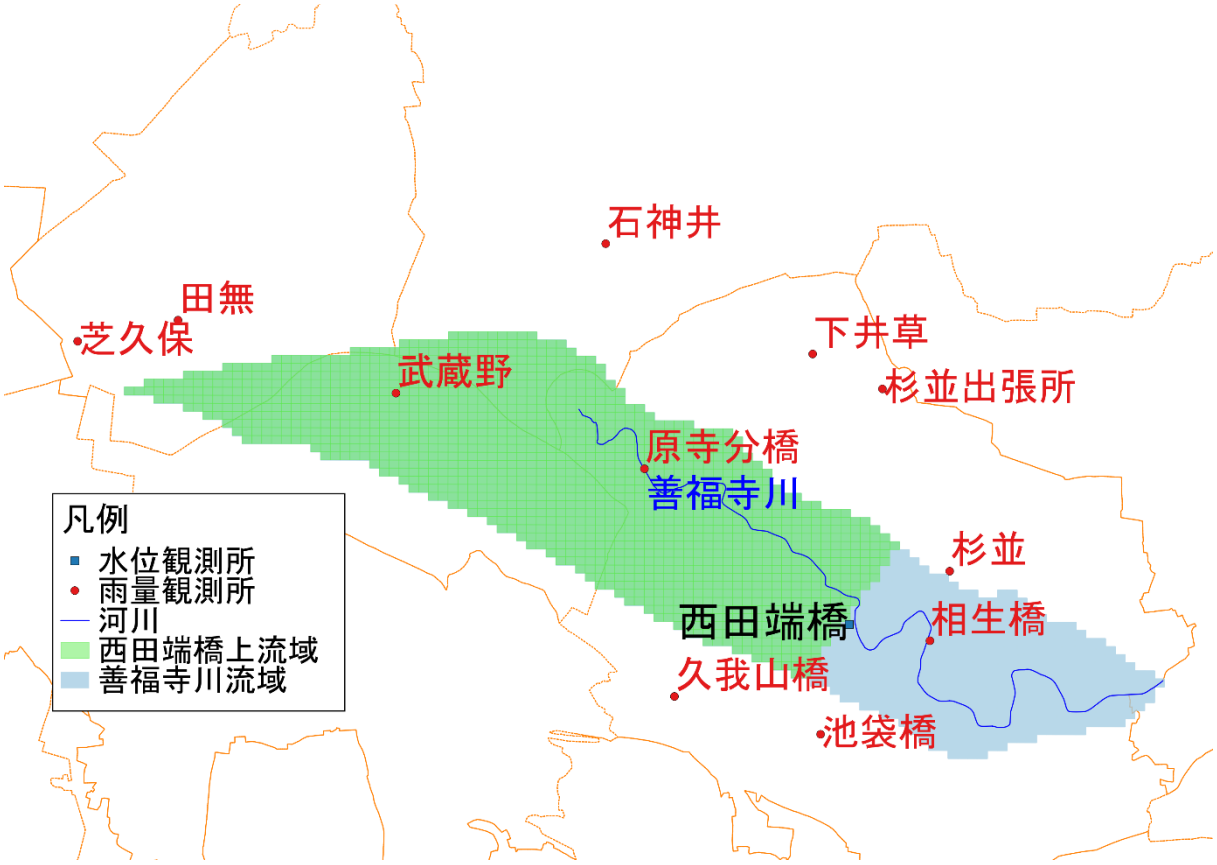

図 3-6　対象流域の水位観測所および雨量観測所

表 3-2　西田端橋上流域における雨量観測所のティーセン係数

雨量観測所名	観測所コード	ティーセン係数
下井草	1C09	0.014076
相生橋	1C13	0.023516
石神井	1D03	0.021478
杉並出張所	1S41	0.012692
原寺分橋	1C15	0.390942
芝久保	1K02	0.000870
杉並	1C14	0.030423
田無	1D05	0.063204
池袋橋	1C11	0.060925
久我山橋	1C12	0.077374
武蔵野	1C16	0.304499

都市貯留関数モデルでは，対象流域における都市特有の流入出成分を算定し，入力する必要がある．

対象流域における降水量以外の流入成分 I としては上水道の漏水や環境用水の導入が挙げられる．上水道の漏水量は東京都水道局の事業年報 [14]に記されている漏水率と使用水量から約 0.00005mm/min であると推定された．環境用水としての導水は千川上水に送水された下水処理水が $0.1m^3/s$ 善福寺川始点付近で流入している [15]．また，善福寺川の水源である善福寺池では水位維持のために $0.018m^3/s$ の地下水を汲み上げて補給している [15]. 以上が降水量以外の主な流入成分と考えられることから，I はこれらの合計量として 0.0001 mm/min と設定した．なお，流域外からの地下水流入量は 0mm/min とした．次に，対象流域では河川からの取水は行われていないこと，豪雨時においては蒸発散量が極めて小さいことから，取水量 O および蒸発散量 E はともに 0mm/min とした．

都市貯留関数モデルのパラメータ q_{Rmax} は合流式下水道により流域外に排出される最大雨水量であり，遮集量から汚水量，特殊水量，地下水量等を差し引いた水量である．ここで，下水管渠の流下能力をマニング式により算定し，下水処理場の計画汚水量を差し引いて，当該流域では q_{Rmax}=0.00213mm/min とした．

3-3-2 豪雨イベント

本研究で対象としたデータは，東京都水防災総合情報システム[16]により収集されているデータを基にしている．東京都水防災総合情報システムとは，東京都が独自に都内の雨量，河川水位，潮位などの観測情報をリアルタイムで自動集計し，水害防災活動を行う関係機関に提供するものである．

洪水予測計算に使用した河川流量については 1 分間隔で計測されている水位データを水位流量(H-Q)曲線により変換し，1 分値データとして得た．

本研究では複数の降雨イベントを対象に都市貯留関数モデルのパラメータ同定を行う．都市貯留関数モデルに入力する雨量は流域平均雨量なので，その値をイベント毎に算定した．また対象とする降雨イベントは，2013 年から 2015 年までの，河川溢水がみられなかったものとした．この 3 年間について降雨イベント毎に流域平均雨量の 60 分最大雨量を計算し，上位 8 イベントとして抽出した．降雨イベントの区切り方としては，降雨開始から，降雨終了後 180 分間の無降雨が継続したときとした．なお，流域平均雨量はティーセン法により算出することとし，対象流域内及び流域周辺に位置する雨量観測所のデータを用いて 1 分値として求めた．抽出された降雨イベントの詳細は**表 3-3** に示すとおりである．また各イベントのハイドログラフ・ハイエトグラフを**図 3-7** から**図 3-14** に示す．

表 3-3 対象降雨イベント

イベント番号	年/月/日	60 分最大雨量(mm)	総雨量(mm)
1	2014/06/24	30.0	34.2
2	2014/09/10	29.7	57.2
3	2013/08/21	28.6	28.6
4	2013/10/16	28.3	191.9
5	2013/09/05	27.8	73.6
6	2014/07/20	27.4	28.0
7	2013/06/25	25.0	29.0
8	2015/09/09	24.5	206.9

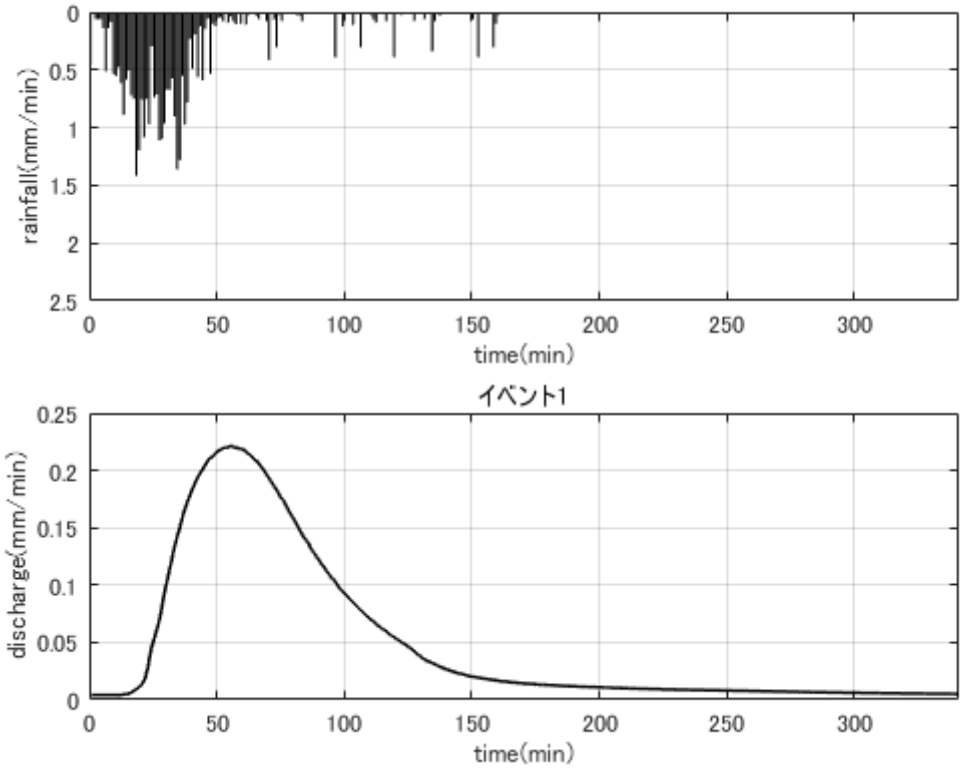

図 3-7　イベント 1

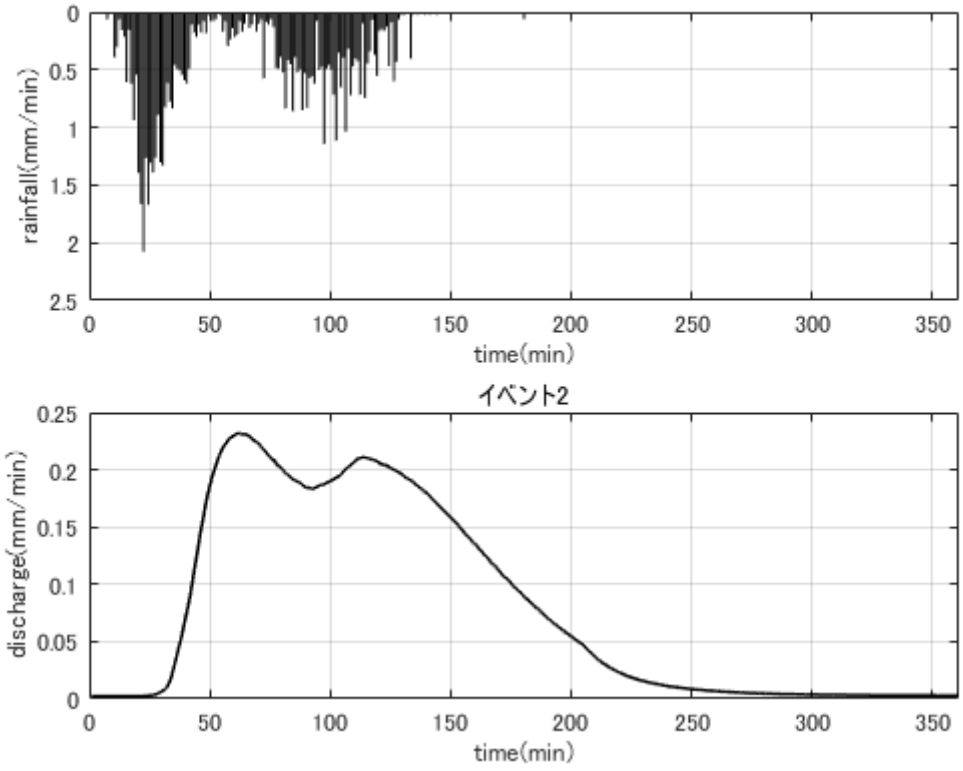

図 3-8　イベント 2

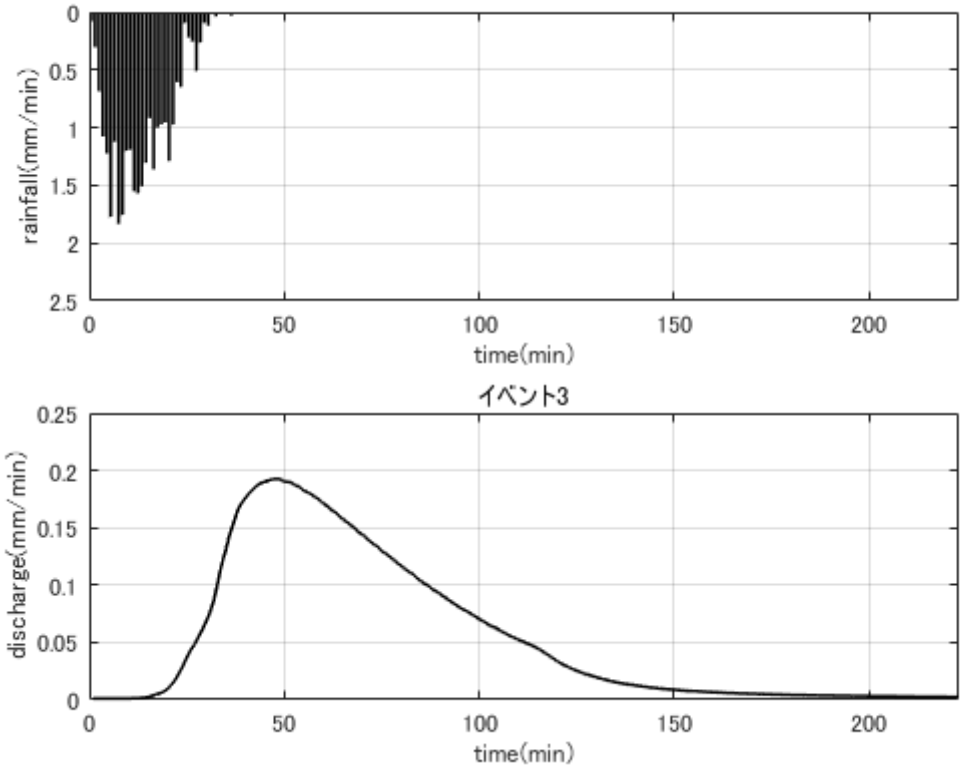

図 3-9　イベント 3

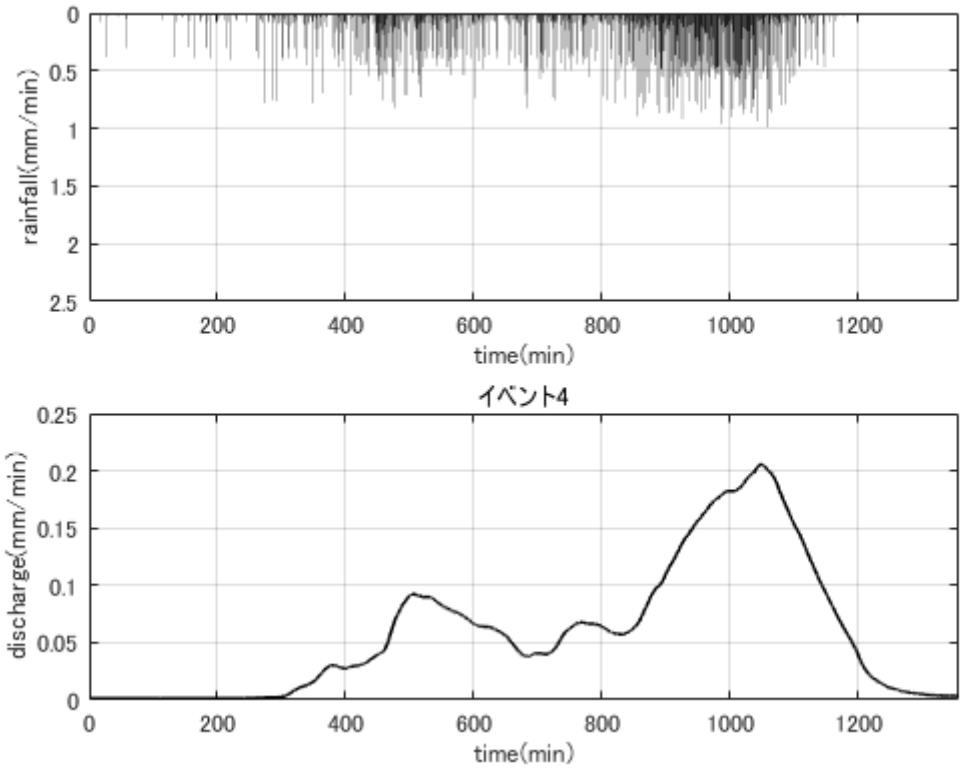

図 3-10　イベント 4

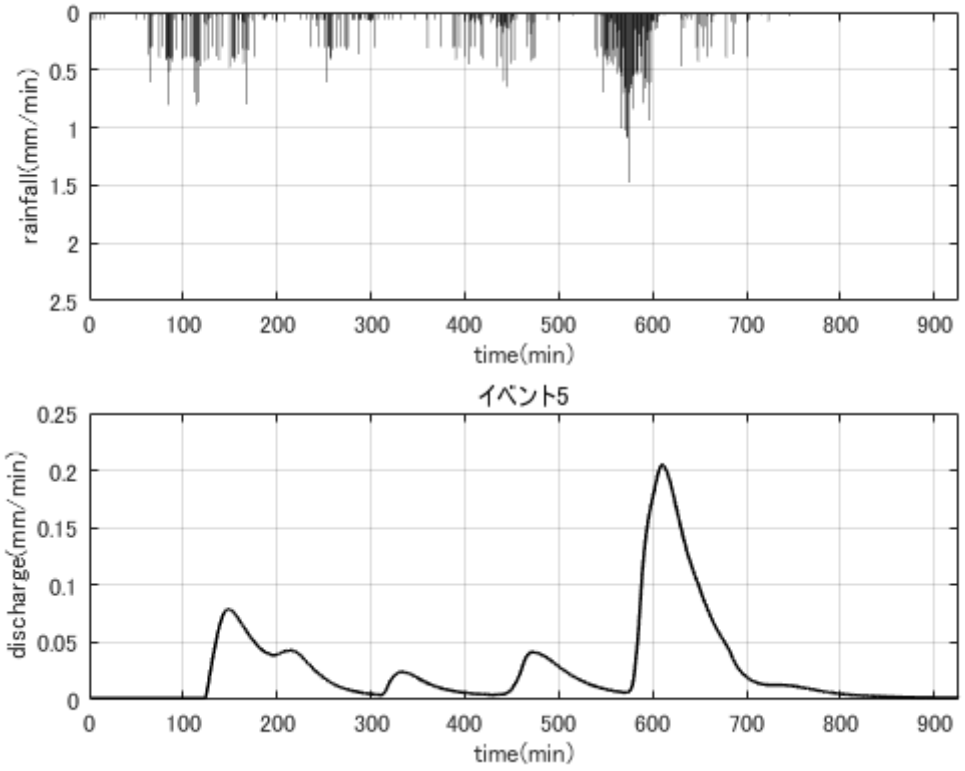

図 3-11　イベント 5

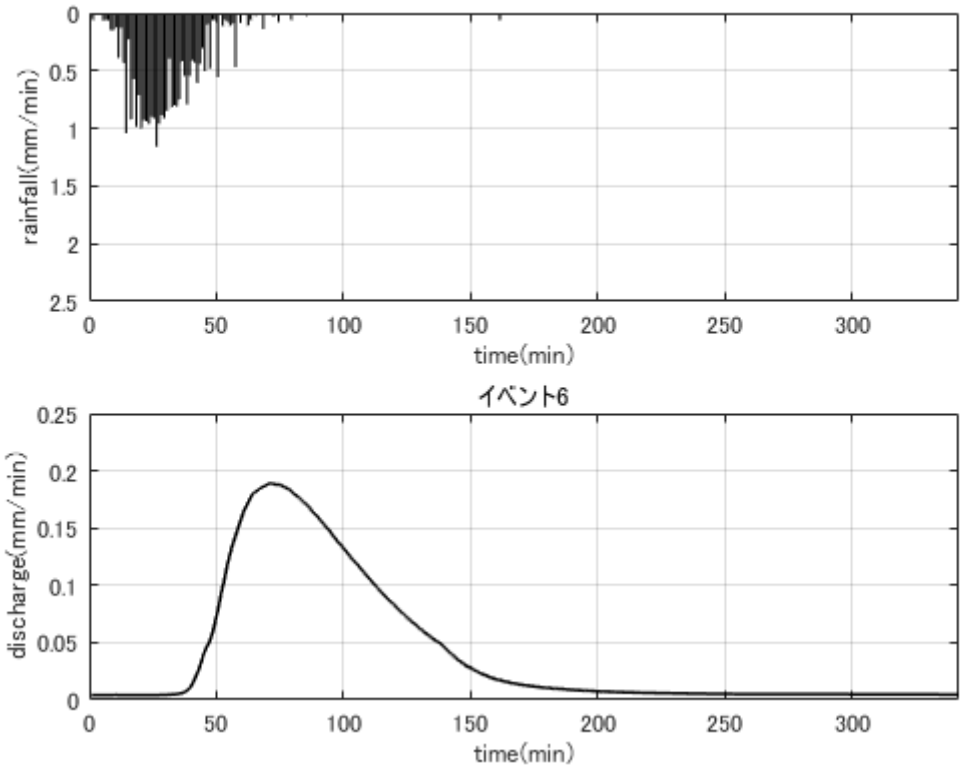

図 3-12　イベント 6

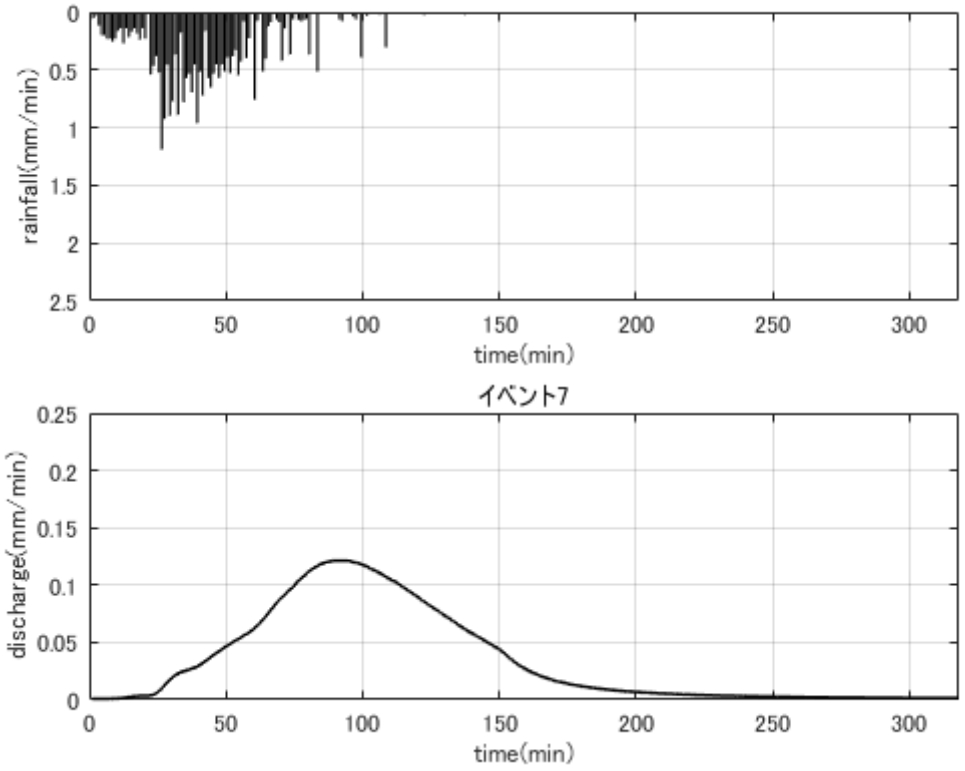

図 3-13　イベント 7

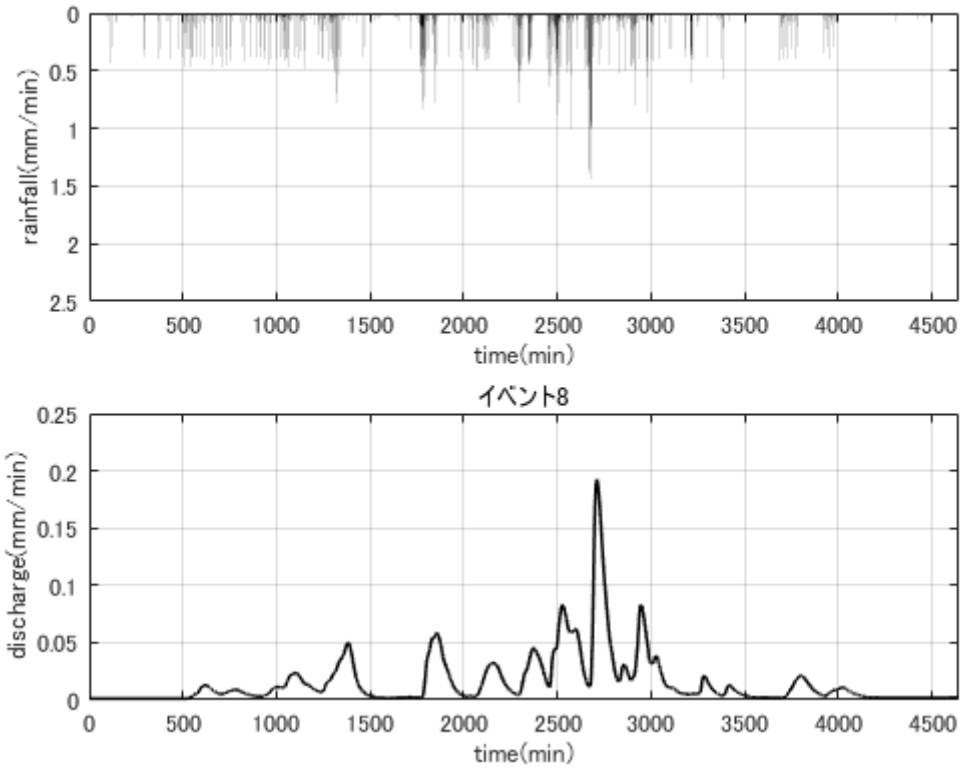

図 3-14　イベント 8

第４章

パラメータ同定

第４章　パラメータ同定

４－１　仮想流域への適用

第３章　３－２仮想流域データの作成に示した，都市貯留関数モデルの真値パラメータが明らかな仮想流域データに対し，SCE-UA 法・PSO・Cuckoo Search によってパラメータ同定を行った．本研究で使用した 3 つの進化的計算手法によるパラメータ同定においては，プログラム上で発生させる乱数によって計算結果が異なるため，それぞれ乱数を変化させて 100 回ずつパラメータ同定を行った．厳密に真値パラメータを同定した場合，目的関数である RMSEは 0 となるが，本研究で使用した 3 手法は厳密解を求める手法ではなく，近似解を求める手法であるため，RMSE = 0となるパラメータを求めることは，現実的ではない．そこで，ここでは便宜的に，RMSE ≤ 10^{-4}となるパラメータを同定したものを，真値パラメータを同定したものとしてみなした．

図 4-1 に，各手法で乱数を変化させて 100 回ずつパラメータ同定を行った結果のRMSEを，箱ひげ図として示す．箱ひげ図は統計などの分野でよく使用される図であり，標本のばらつきを表現することができる．長い赤の横棒は中央値を示しており，青い箱は第一・第三四分位を示している．黒線のひげの外にあるものは外れ値である．**図 4-1** を見ると，SCE-UA 法によるパラメータ同定結果によるRMSEは殆どが 0 に近づいている．一方 PSO は 0 に近いものがなく，真値を同定したとは考えられない．また Cuckoo Search は中央値が 0 に近いことから，比較的高い探索性能を示したことが読み取れる．ここで，上述のRMSE ≤ 10^{-4}以下に収束したパラメータを真値とすると，100 回の試行で**表 4-1** のような確率で真値を同定したこととなる．SCE-UA 法は SCE-UA 法は 99%と非常に高い確率で真値を同定した．一方，PSO は 0%であり，十分な探索性能を発揮していない．Cuckoo Search は 62%であった．

ここで，PSO の探索結果が良好ではなかった理由について考察を行う．PSO には，粒子の動きを制御するパラメータが存在し，これらを適用する最適化問題に応じて設定する必要がある．本来ならば，いくつかの仮想流域データを作成して，都市貯留関数モデルによって PSO のパラメータを設定するべきである．しかし，本研究では Ackley 関数，Rastrigin 関数というベンチマーク関数によってパラメータを設定したため，都市貯留関数モデルに適合しなかったものと考えられる．一方で Cuckoo Search も同様の条件であることから，Cuckoo Search の真値パラメータ探索結果を考えると，PSO よりも Cuckoo Search に分があると言える．Cuckoo Search 開発者の Yang らは，Cuckoo Search には PSO と比較して設定すべきパラメータ数が少ないことを利点として挙げており [17]，本研究でもそれによる頑健さが確認された．

第２章で示したとおり，SCE-UA 法によるパラメータ同定における目的関数の評価回数は 25,000 回前後，PSO と Cuckoo Search は 150,000 回である．計算の効率を考慮すると，SCE-UA 法は少ない目的関数の評価回数にかかわらず，非常に高い精度で真値パラメータを探索したことが確認された．

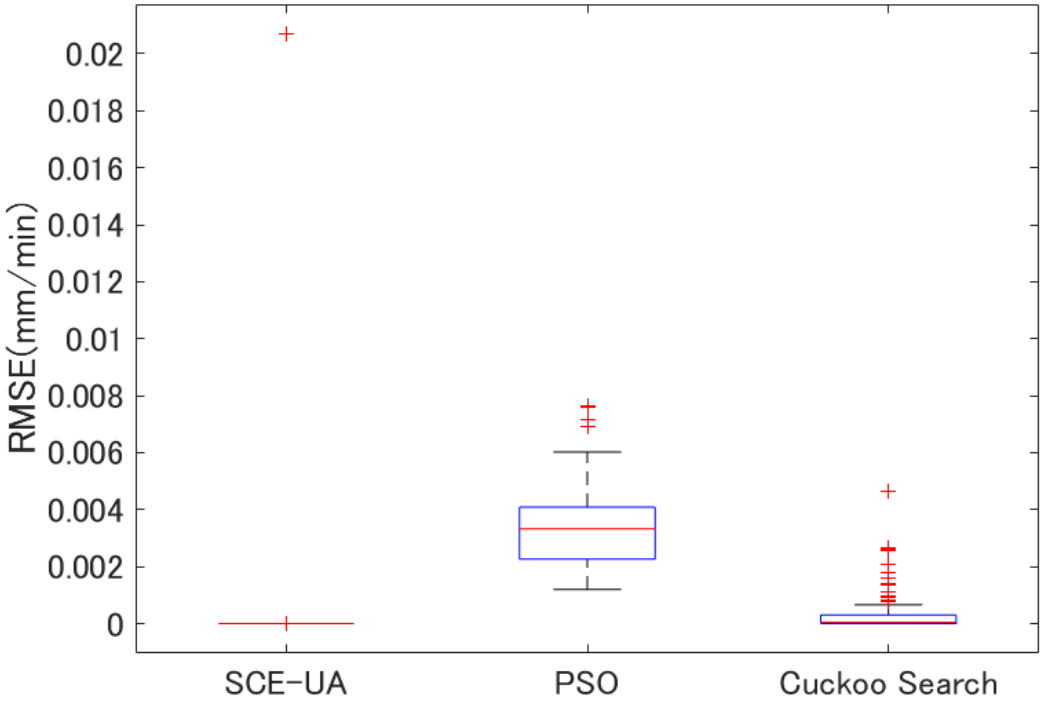

図 4-1　仮想流域でのパラメータ同定結果によるRMSE

表 4-1　RMSE $\leq 10^{-4}$に収束した確率（n = 100）

SCE-UA 法	PSO	Cuckoo Search
99%	0%	62%

図 4-2 から**図 4-4** は，実際に収束したパラメータ値を示したものである．左に示す折れ線グラフは 100 本の線が描かれており，横軸は都市貯留関数モデルパラメータのインデックス，縦軸はパラメータの探索範囲を[0 1]に標準化したものである．なお都市貯留関数モデルパラメータのインデックスは，1,2,3,4,5,6,7 がそれぞれk_1，k_2，k_3，p_1，p_2，z，αにあたる．また右の図は箱ひげ図となっており，実際にどのようなパラメータ値に収束したか，そのばらつき表現している．左右の図には黒の正方形で真値をプロットしている．

図 4-2 によると，SCE-UA 法が唯一準最適解に陥った計算結果では，真値から大きく外れている．パラメータk_1は真値に近いものを示していることから，これは突然変異によるものではないと考えられる．SCE-UA 法は計算の世代数が進むと探索の動きが鈍くなるため，これが原因で局所解から抜け出せなかったものとみられる．また**図 4-1** によるとRMSEの値が，3 手法の全試行中で最悪となっている．しかし，局所解に至った確率が 1%であったことは，SCE-UA 法の探索性能の高さを示している．

図 4-3 と**図 4-4** をみると，PSO と Cuckoo Search の収束パラメータのばらつきは似たような分布を示している．ただし，箱ひげ図を見ると Cuckoo Search の方が優秀と言える．

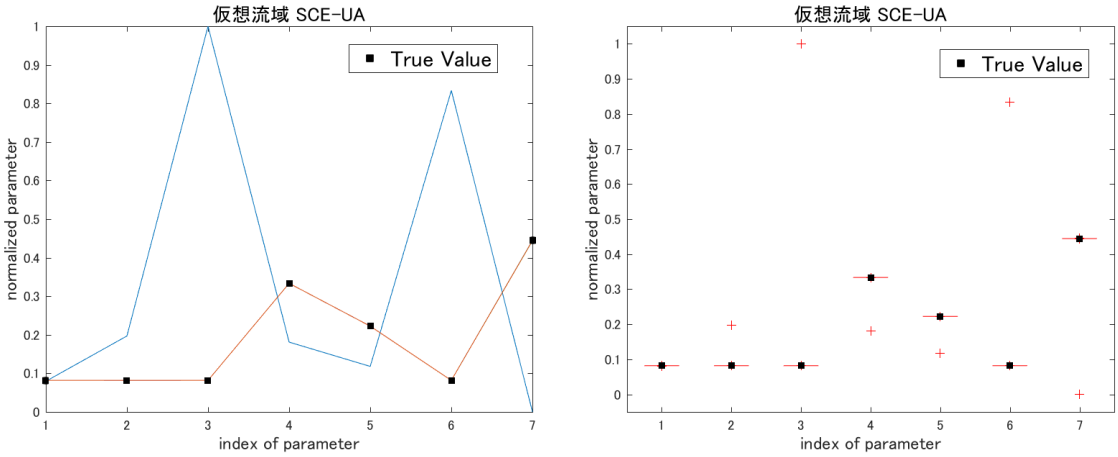

図 4-2　SCE-UA 法による同定パラメータ

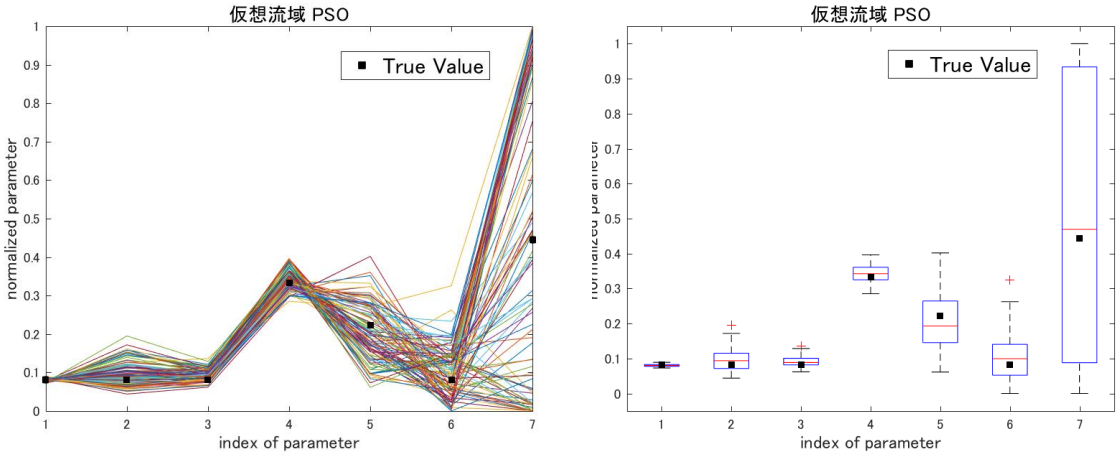

図 4-3　PSO による同定パラメータ

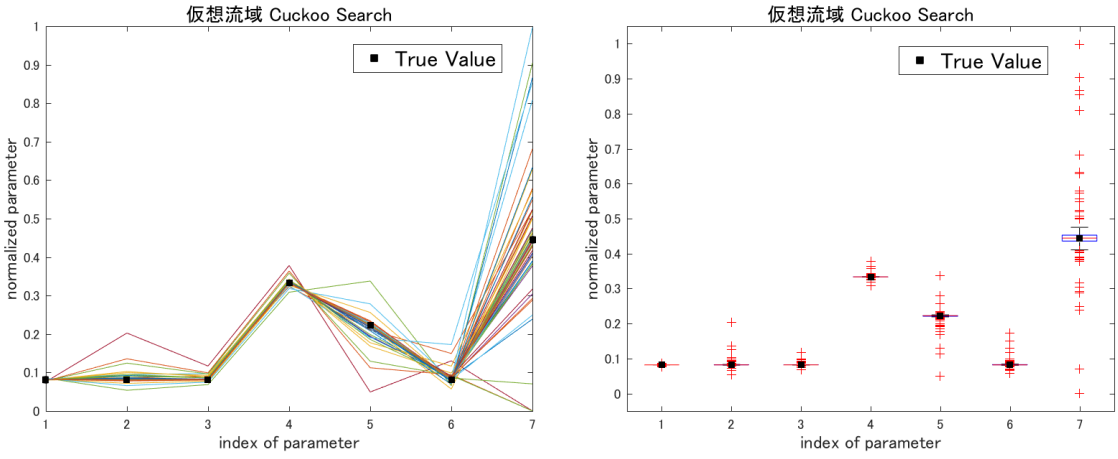

図 4-4　Cuckoo Search による同定パラメータ

図 4-5 に，目的関数の評価回数によって，3 手法がどのようなRMSEに収束したかを示す．計算は SCE-UA 法・PSO・CS で乱数を変化させて各 100 回ずつ行ったものである．PSO と CS は，評価回数が 150,000 回と揃えてあるものの，SCE-UA 法は 25,000 回前後であるため，プロットされている範囲が異なる．

同図をみると，目的関数の評価回数が小さいときは，SCE-UA 法と CS が拮抗しており，互角の収束性を示している．特に最初期の辺りでは CS が勝っているものも見られ，Lévy Flightsによる探索が上手くされていることがわかる．

評価回数が 10,000 回となった辺りから，SCE-UA 法が強力に真値に向かって収束していることが分かる．一方，CS は緩やかな収束を示している．また PSO は全体に渡って，収束の具合が良くないと言える．

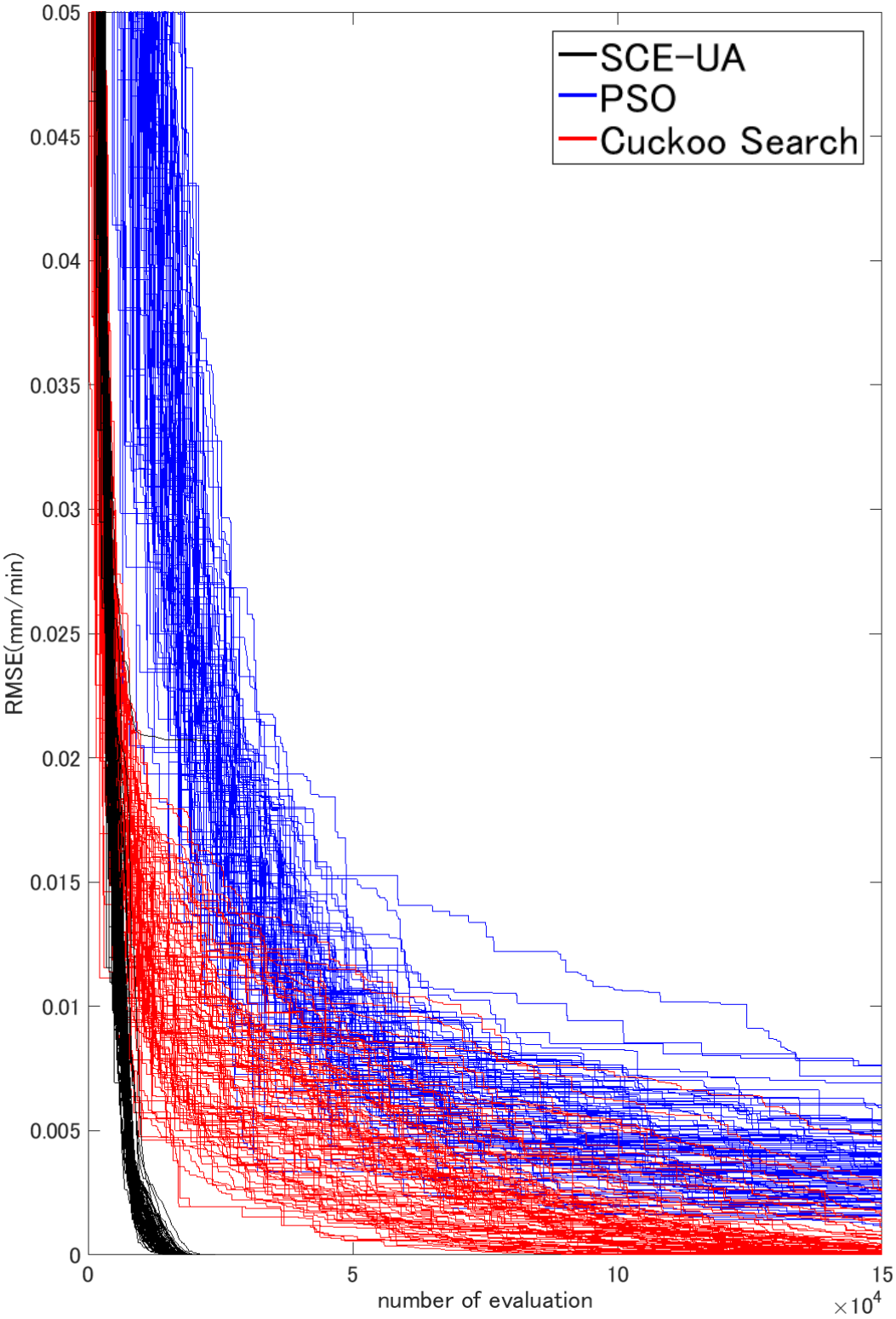

図 4-5　RMSEの変動（各 100 回ずつ実行）

４－２　実流域への適用

第３章　３－３実流域データの作成に示した善福寺川西田端橋流域の 8 洪水イベントに対し，SCE-UA 法・PSO・Cuckoo Search の 3 手法で，発生させる乱数を変化させて 100 回ずつパラメータ同定を行った．計算の試行回数が$8 \times 3 \times 100 = 2{,}400$回となり，計算時間が非常に長くなると考えられたため，都市貯留関数モデルの数値解法 Runge-Kutta-Gill 法は$\Delta t = 1$とした．なお Runge-Kutta-Gill 法は精度の高い手法であるため，$\Delta t = 1$としても計算結果に大きな変化がないことを確認している．全試行が完了するまでに，MATLAB2016b，windows7 Enterprise，Intel(R) Core(TM) i7-4770S CPU @ 3.10GHz，RAM6.00GB の環境で，約 420 時間を要した．

善福寺川西田端橋流域の 8 洪水イベントに対してパラメータ同定を行った結果，Cuckoo Search が最も頑健な結果を示した．次点で SCE-UA 法が優れていた．**図 4-5** に，SCE-UA 法・PSO・Cuckoo Search で 8 つの洪水イベントに対し，乱数を変化させて 100 回ずつパラメータ同定を行った結果の，RMSEの平均値を示す．Cuckoo Search は全 8 イベントで最も小さいRMSEを示している．一方，SCE-UA 法はイベント 1,6 で，PSO はイベント 1,3,7 で悪い成績を示している．

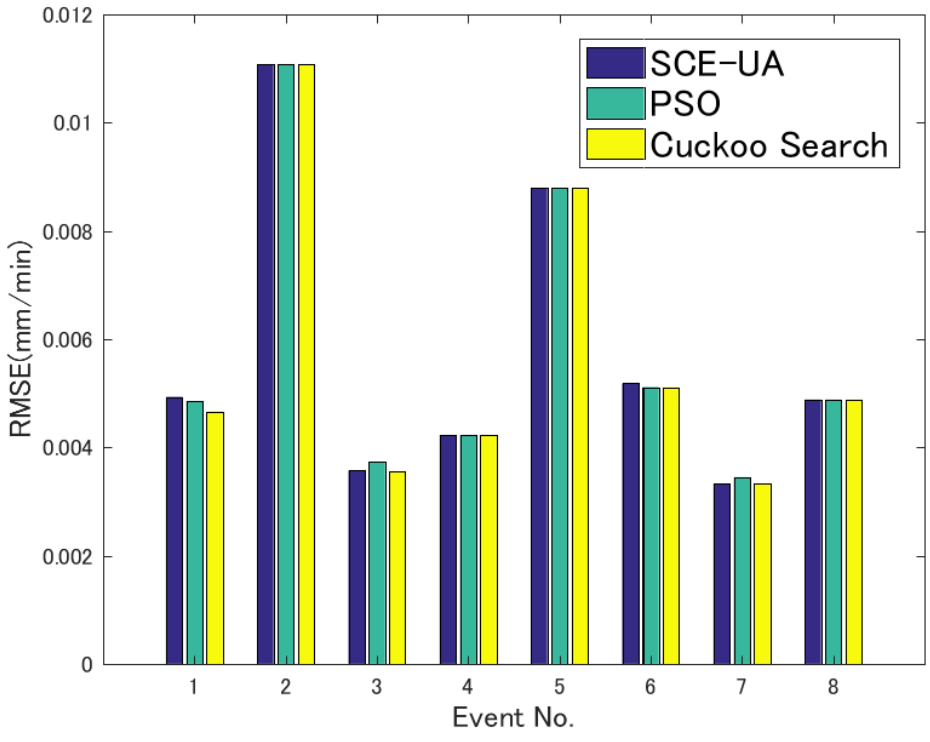

図 4-5　同定されたパラメータによるRMSEの平均値（各 100 回試行）

図 4-6 は，各手法，各イベントで，100 回の試行でどのようなRMSEに収束したかを示す，箱ひげ図である．箱ひげ図では 100 回の結果のうち，中央値や最小値・最大値，第一四分位値や第三四分位値が表示される．

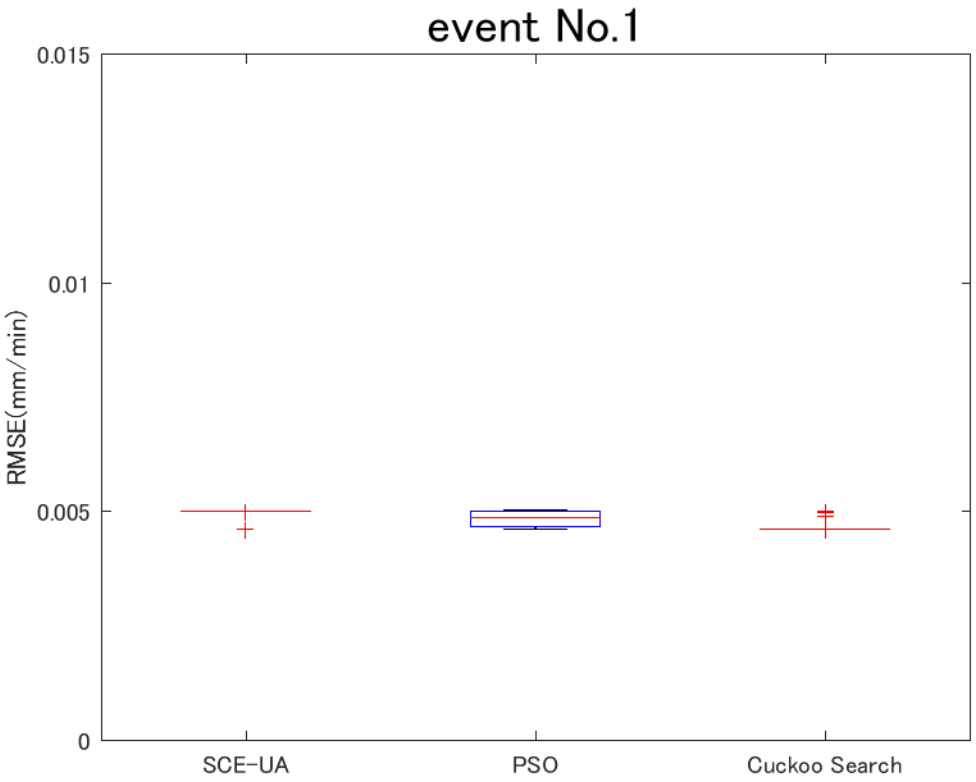

図 4-6　イベント 1 のパラメータ同定結果

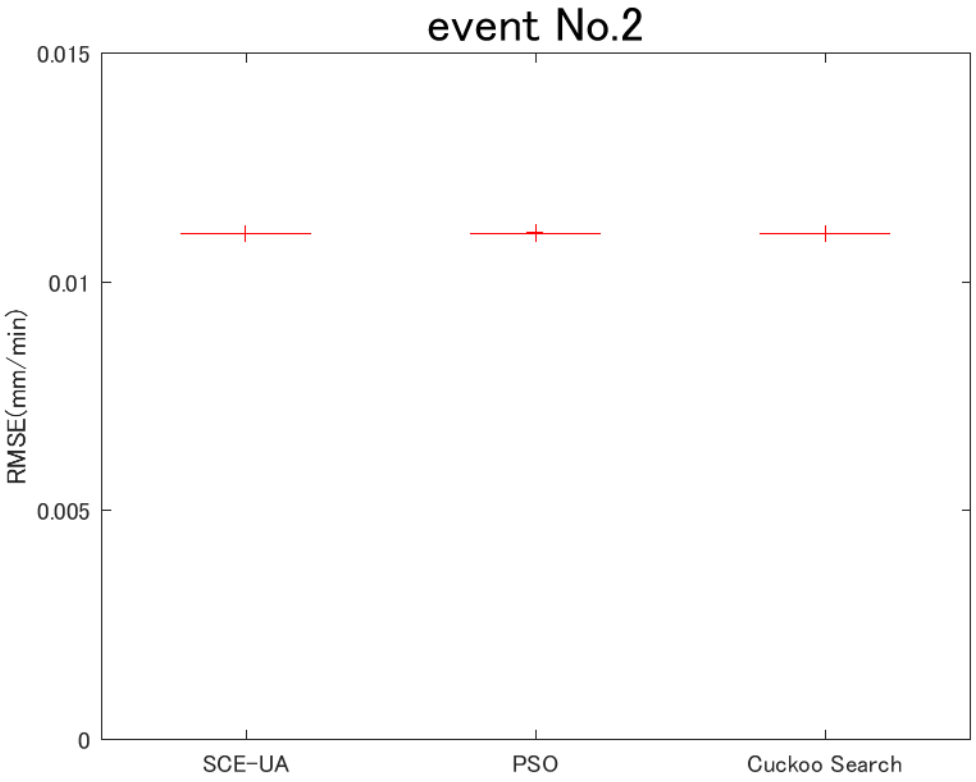

図 4-7　イベント 2 のパラメータ同定結果

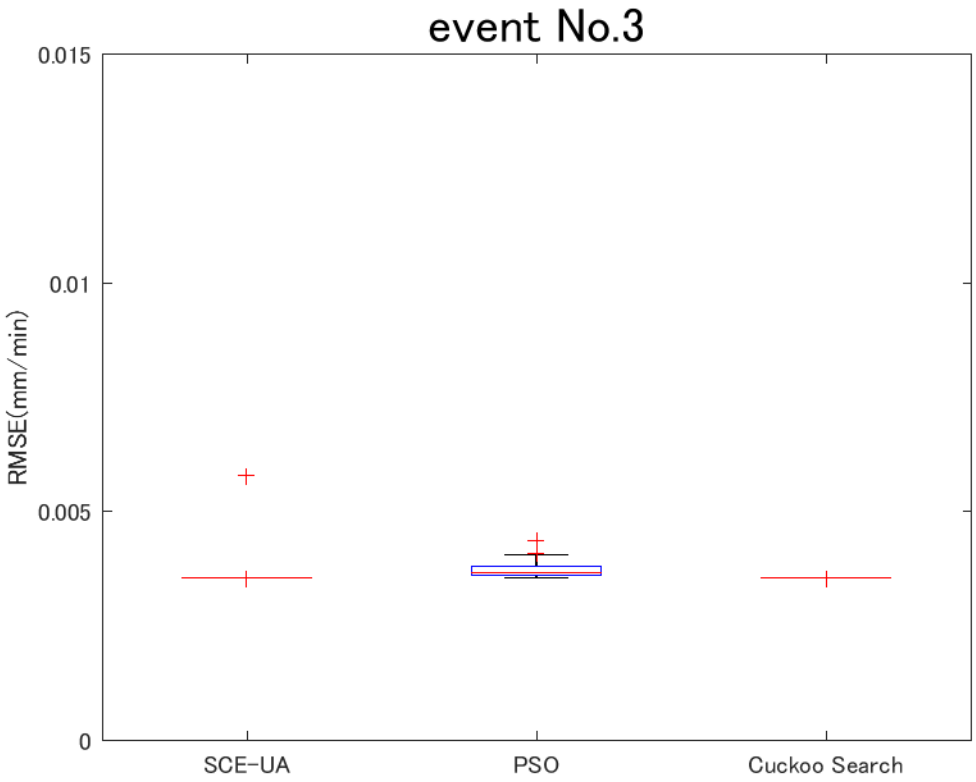

図 4-8　イベント 3 のパラメータ同定結果

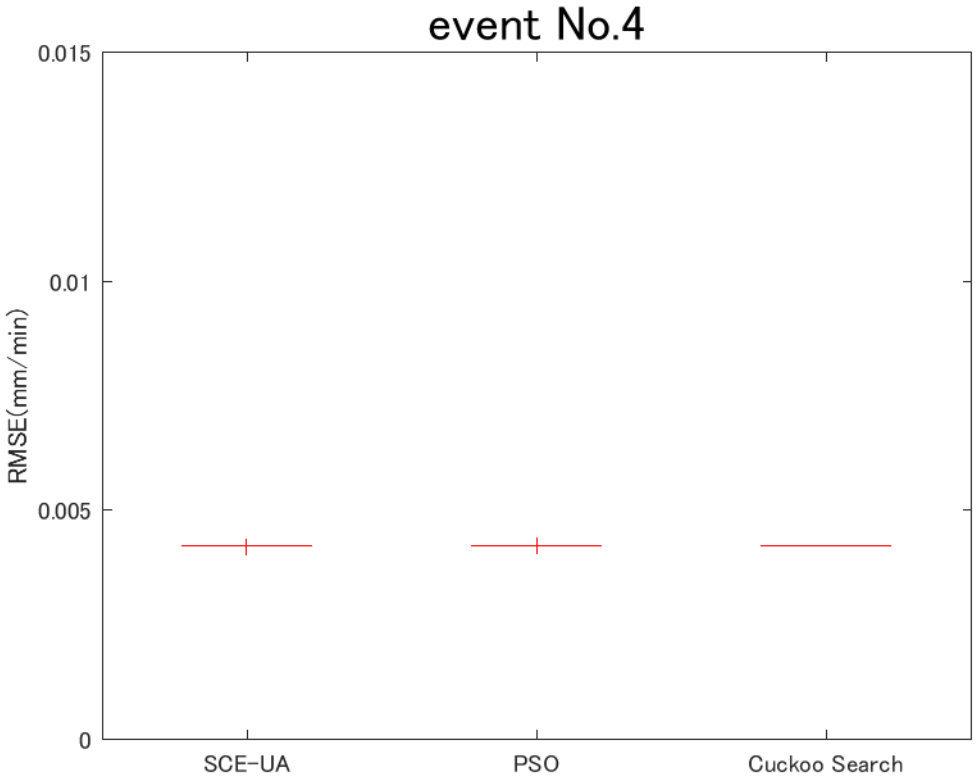

図 4-9　イベント 4 のパラメータ同定結果

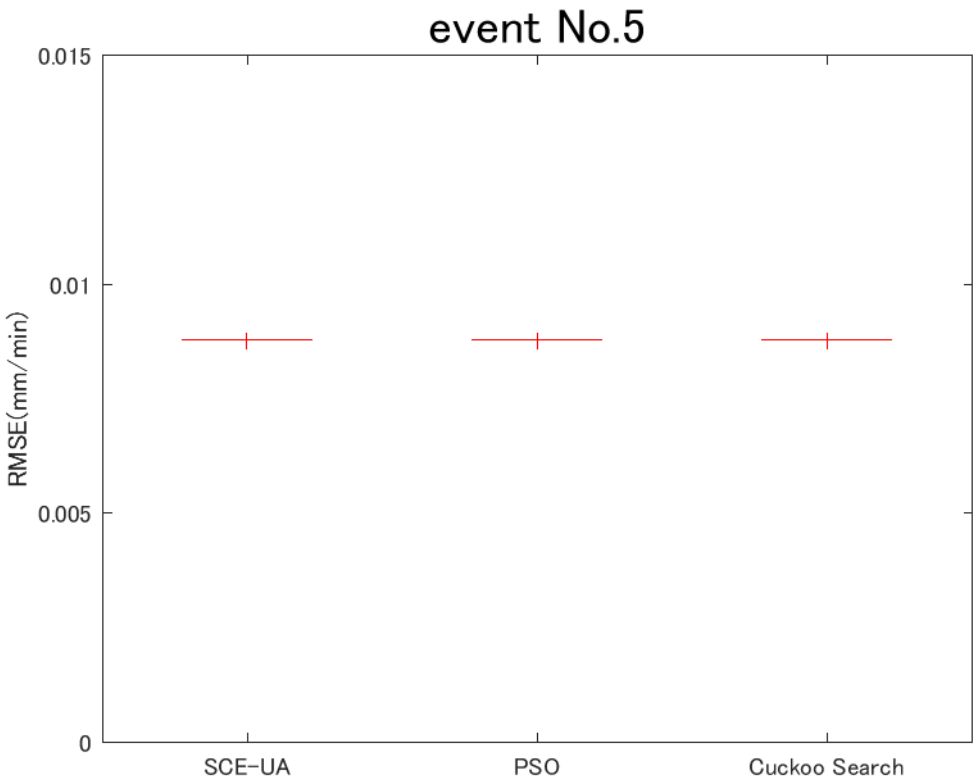

図 4-10　イベント 5 のパラメータ同定結果

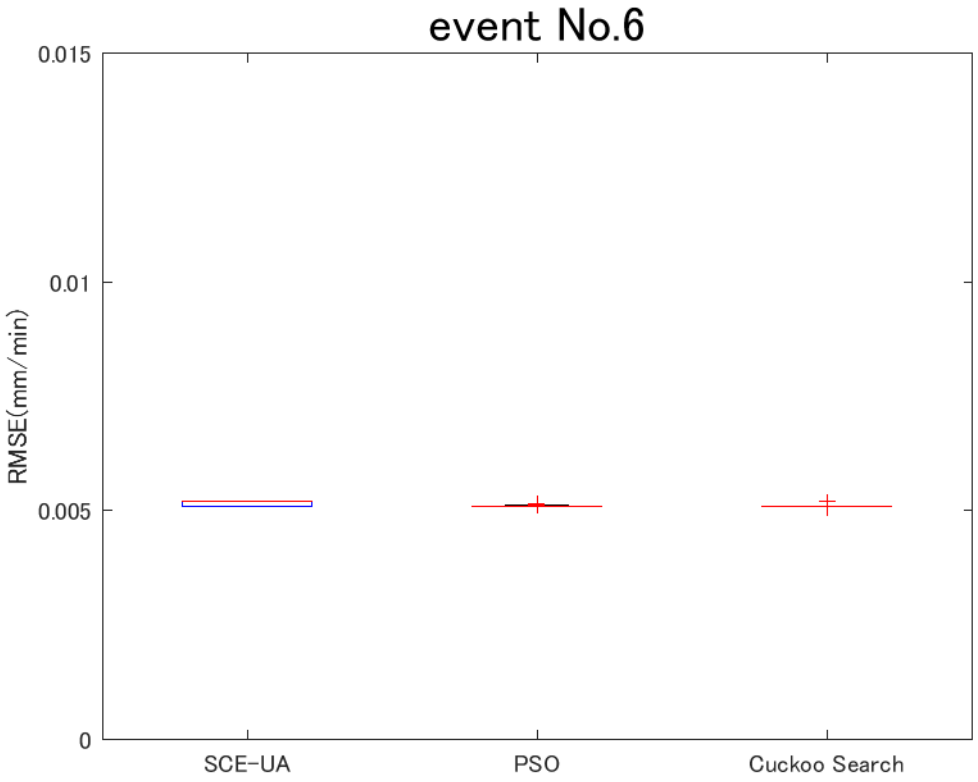

図 4-11　イベント 6 のパラメータ同定結果

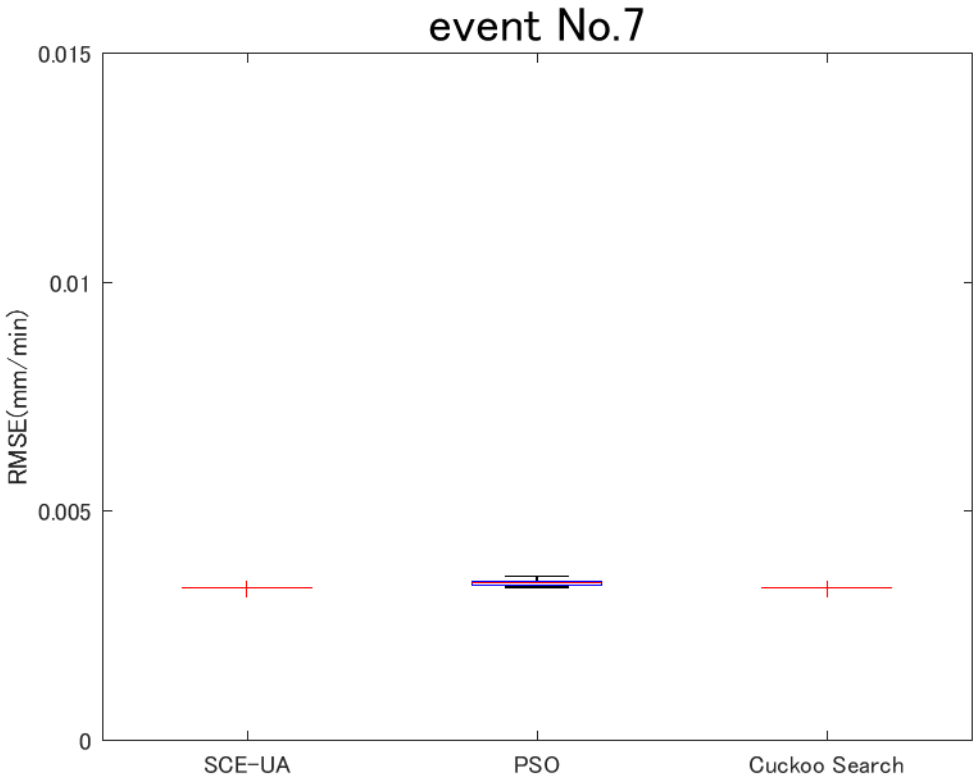

図 4-12　イベント 7 のパラメータ同定結果

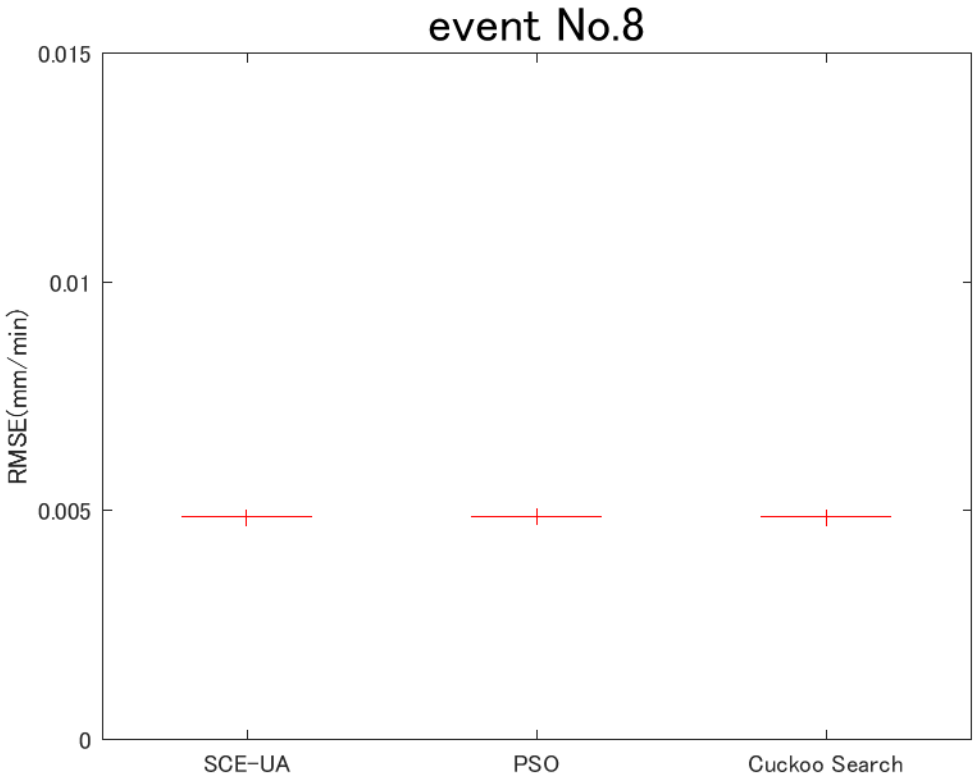

図 4-13　イベント 8 のパラメータ同定結果

図 4-6 を見ると，イベント 1 ではRMSEの中央値が，SCE-UA 法のそれよりも Cuckoo Search のものの方が優れている．これは SCE-UA 法による同定パラメータの多くが準最適解に陥っていることを示している．一方で最小値を見ると，Cuckoo Search も SCE-UA 法も同様の値を示している．PSO の最小値は他よりもやや劣っているが，平均的には SCE-UA 法よりも優れている．また，イベント 6 でも同様の傾向が見て取れる．Cuckoo Search でもイベント 1,6 で，複数回にわたって準最適解に収束していることから，これらの降雨イベントのパラメータ解空間は複雑であり，準最適解に陥りやすいものであるということが示唆される．

イベント 1,3,7 では，PSO による同定パラメータのRMSEに，ばらつきが見られる．第 3 章に示したハイドログラフによると，これらの降雨イベントは単一ピークを示すものである．しかし，イベント 6 も単一ピークのハイドログラフであり，PSO によるRMSEにばらつきが見られないことから，PSO は単一ピークのパラメータ同定が不得手とは断言できない．

ハイドログラフの形状に着目すると，複数ピークをもつイベント 2,4,5,8 では，RMSEのばらつきが小さいということが分かる．複数ピークの洪水イベントの解空間は，比較的単純であると考えられる．

続いて，**図 4-6** から**図 4-13** においてRMSEが中央値に収束したパラメータ同定結果を都市貯留関数モデルに入力して，得られたハイドログラフを**図 4-14** から**図 4-21** に示す．それらによると，局所解に陥ったイベント 1 での SCE-UA 法の結果であっても，ハイドログラフには殆ど差が見られないことがわかる．

図 4-22 から**図 4-45** では，各イベントに対し，100 回の試行のうちで各手法が同定したパラメータ値を示す．左側には 100 本の線が描かれたものを，右側には 100 回の結果のうちの中央値や最小値・最大値などが分かる箱ひげ図を示す．なお左右の図とも，横軸は都市貯留関数モデルパラメータのインデックスを示しており，1,2,3,4,5,6,7 はそれぞれk_1，k_2，k_3，p_1，p_2，z，αにあたる．縦軸は各パラメータの探索範囲を，[0 1]に標準化したものである．

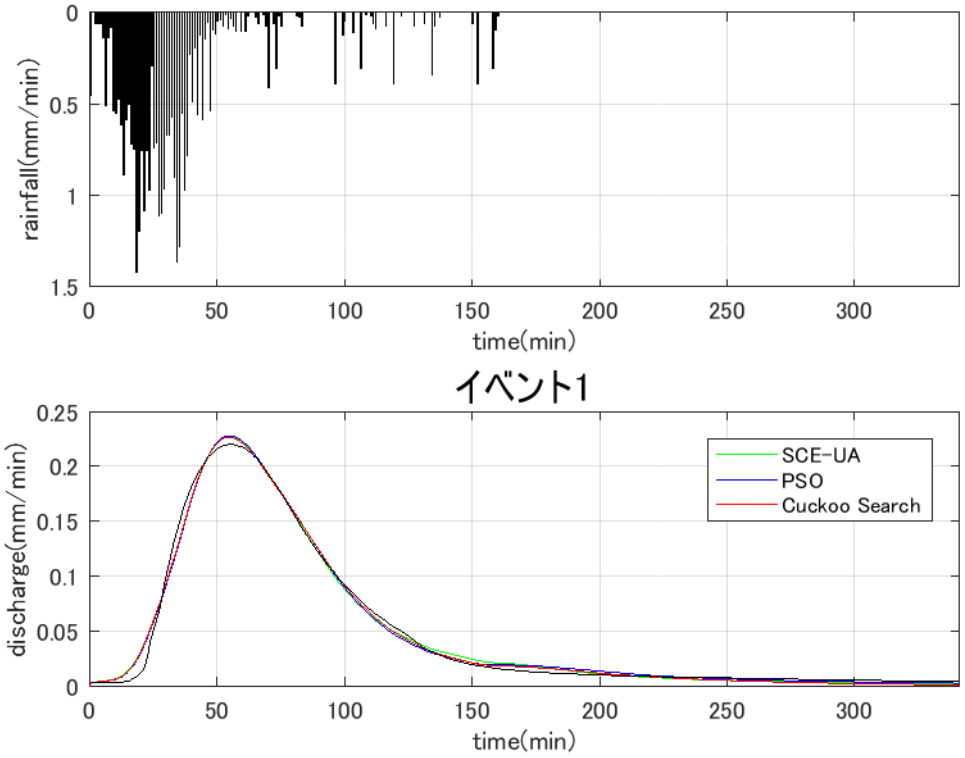

図 4-14　RMSE中央値のパラメータによるハイドログラフ(イベント 1)

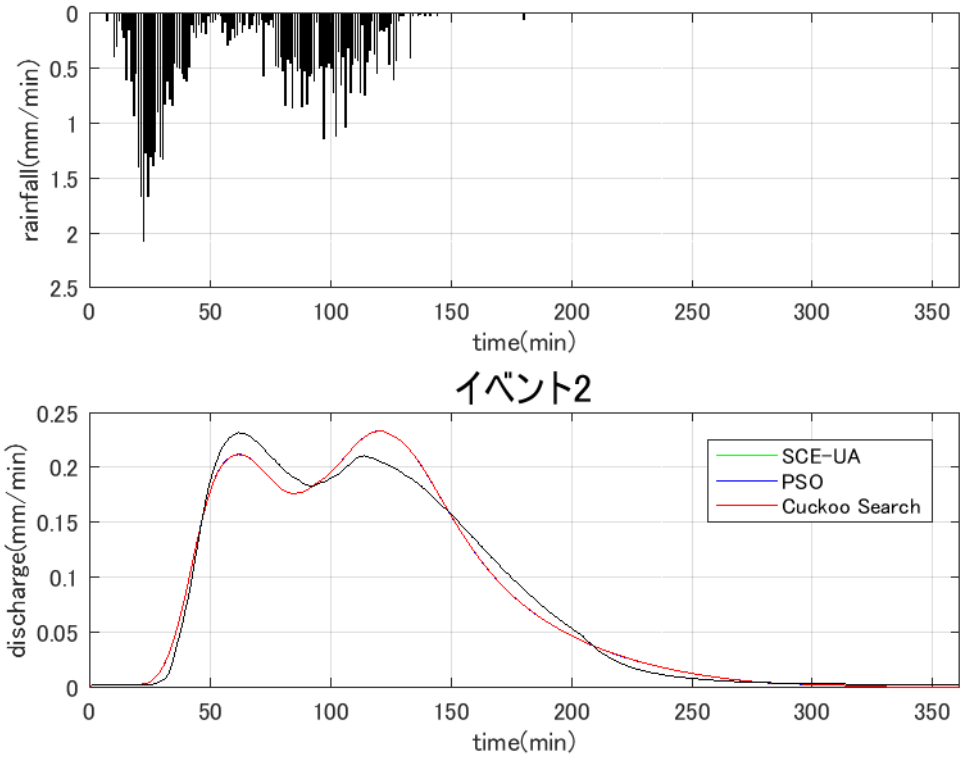

図 4-15　RMSE中央値のパラメータによるハイドログラフ(イベント 2)

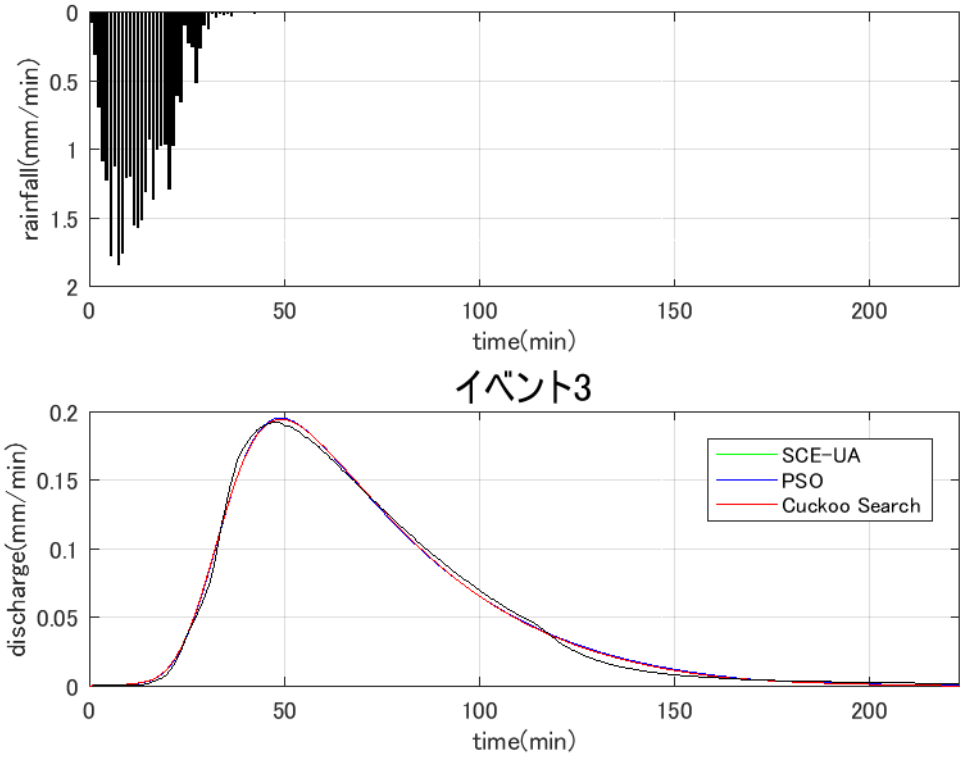

図 4-16　RMSE中央値のパラメータによるハイドログラフ(イベント 3)

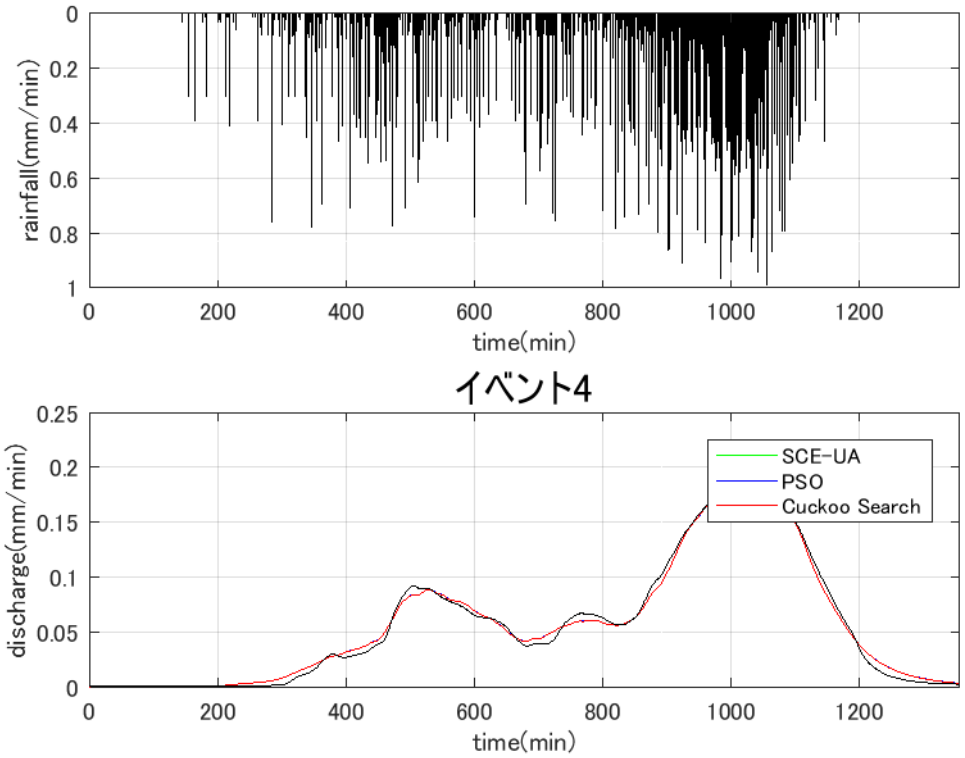

図 4-17　RMSE中央値のパラメータによるハイドログラフ(イベント 4)

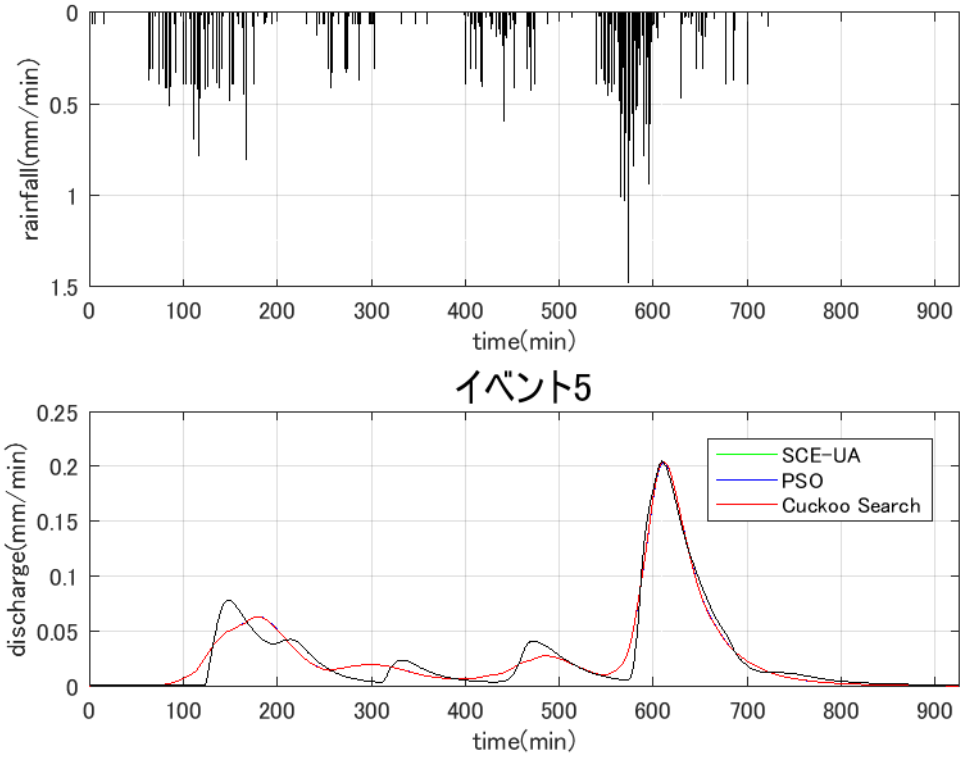

図 4-18　RMSE中央値のパラメータによるハイドログラフ(イベント 5)

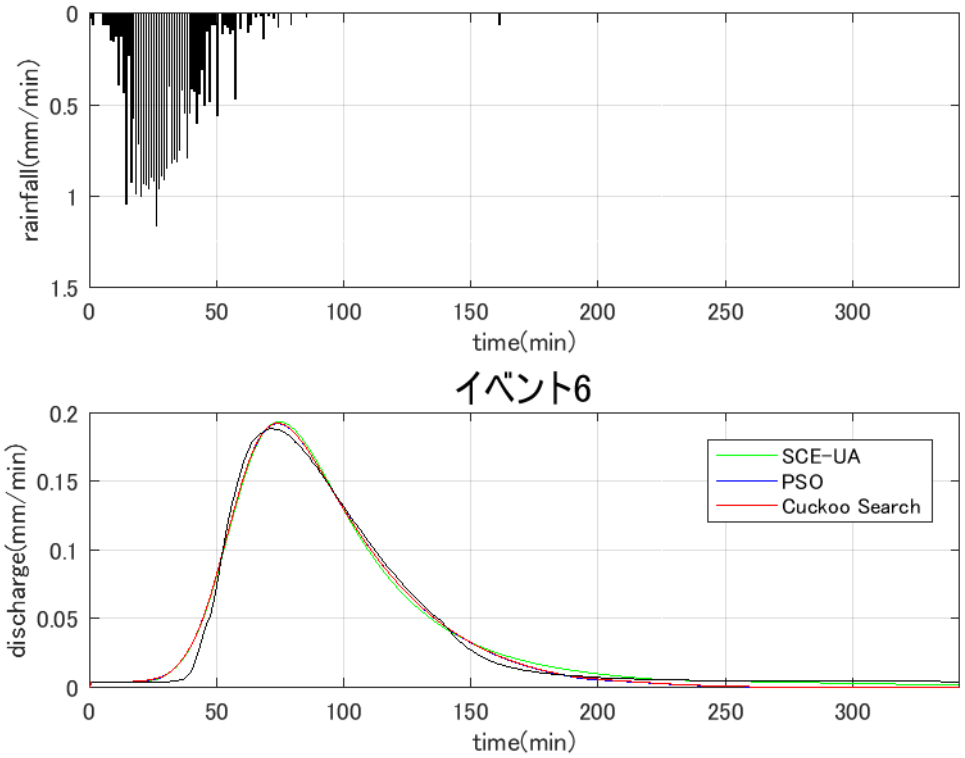

図 4-19　RMSE中央値のパラメータによるハイドログラフ(イベント 6)

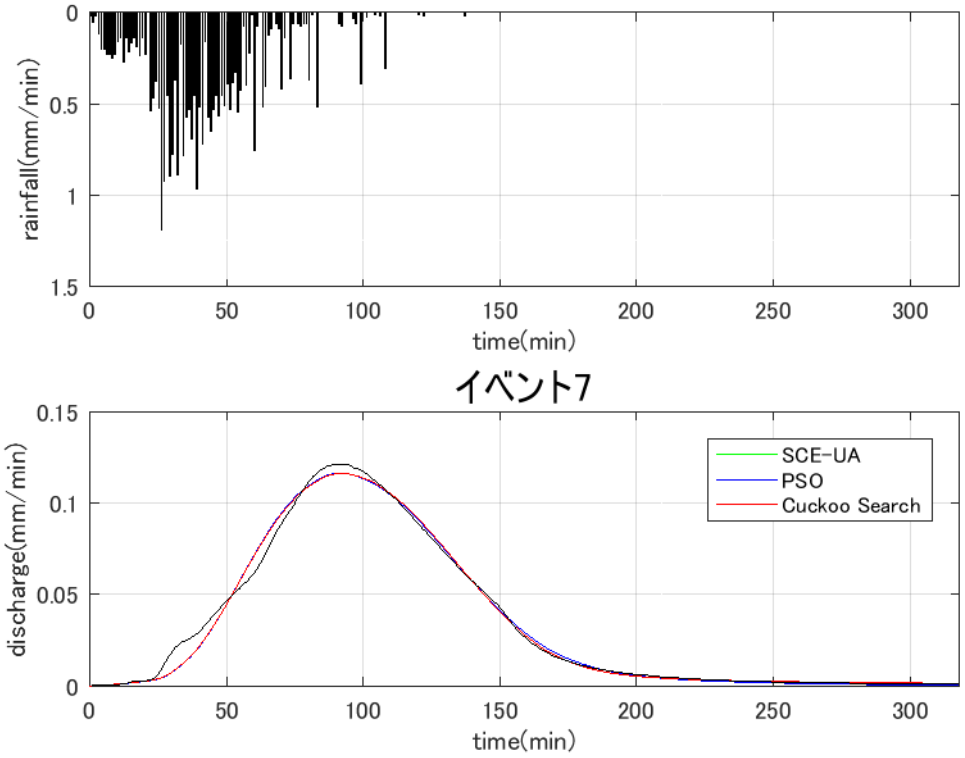

図 4-20　RMSE中央値のパラメータによるハイドログラフ(イベント 7)

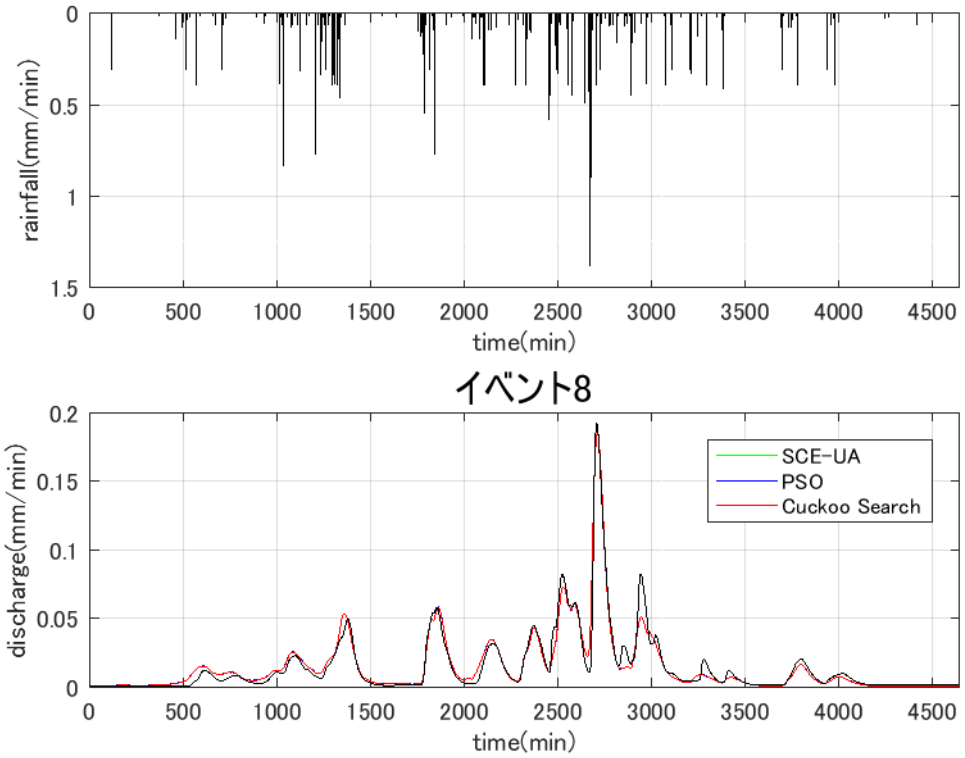

図 4-21　RMSE中央値のパラメータによるハイドログラフ(イベント 8)

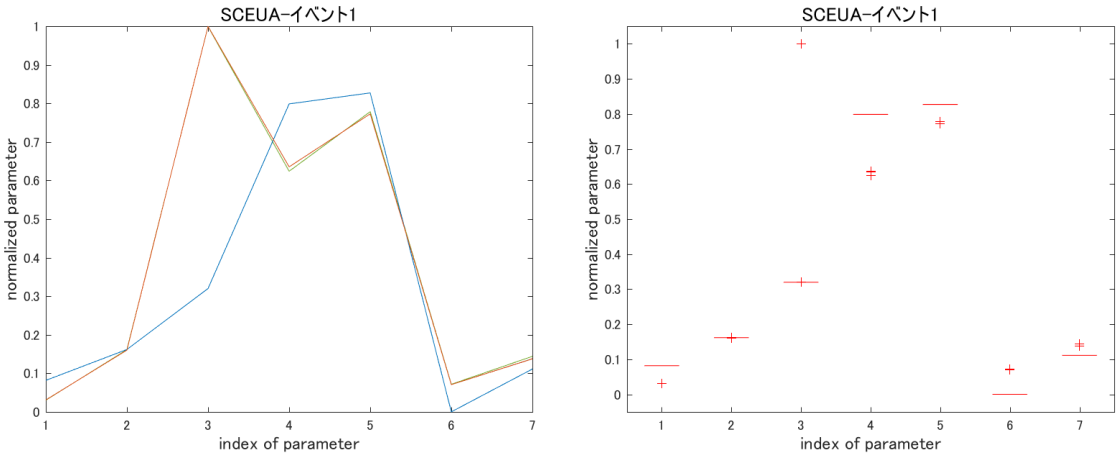

図 4-22　イベント 1 の同定パラメータ（SCE-UA 法）

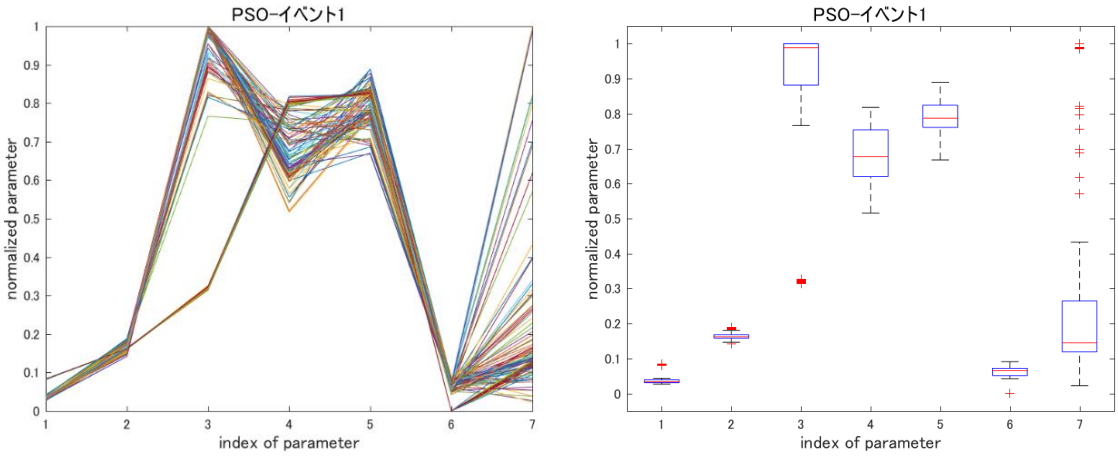

図 4-23　イベント 1 の同定パラメータ（PSO）

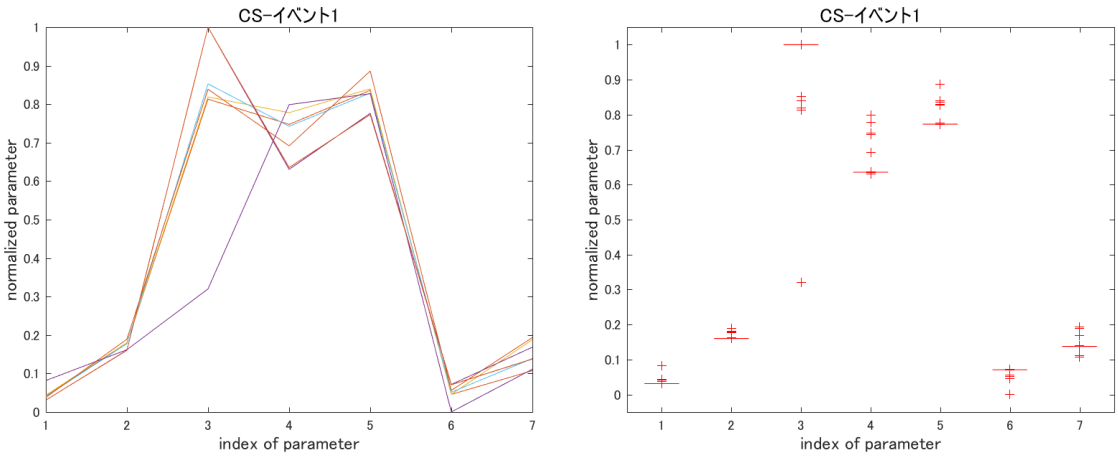

図 4-24　イベント 1 の同定パラメータ（Cuckoo Search）

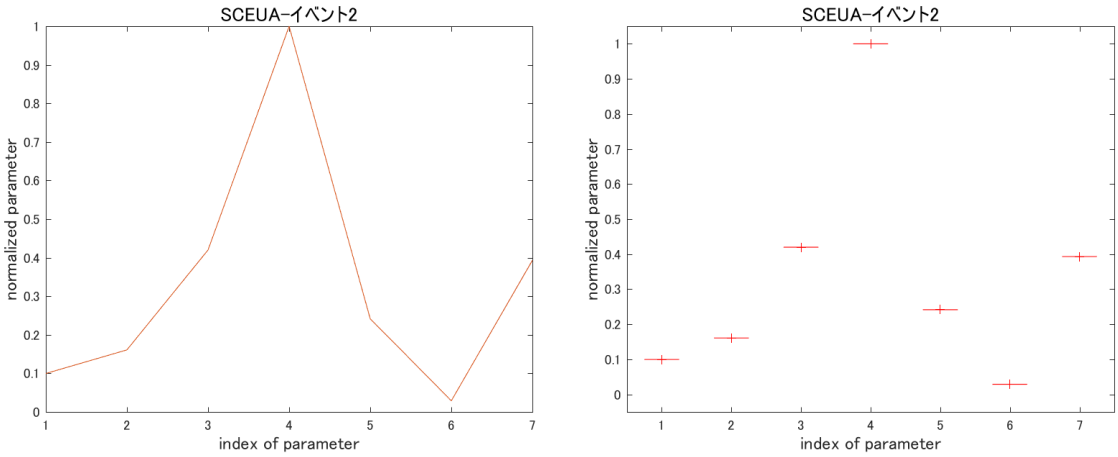

図 4-25　イベント 2 の同定パラメータ（SCE-UA 法）

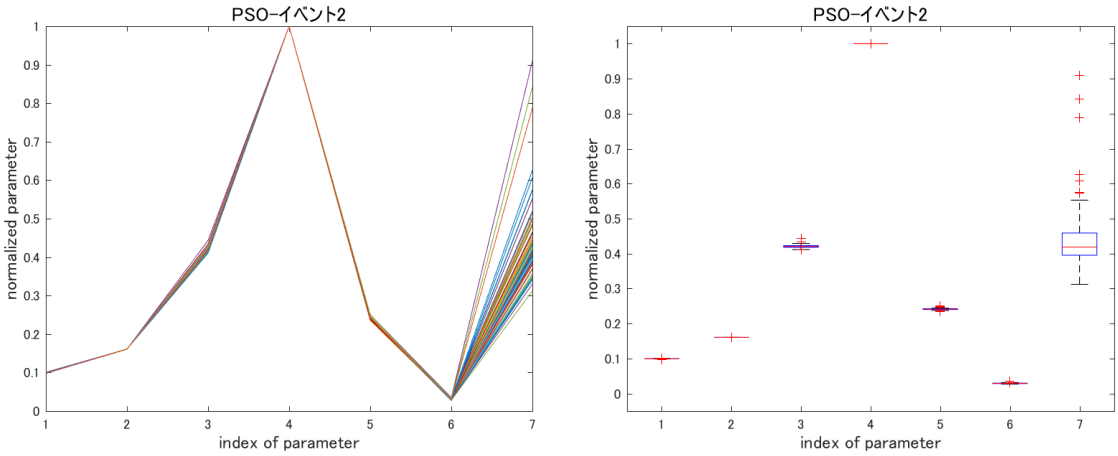

図 4-26　イベント 2 の同定パラメータ（PSO）

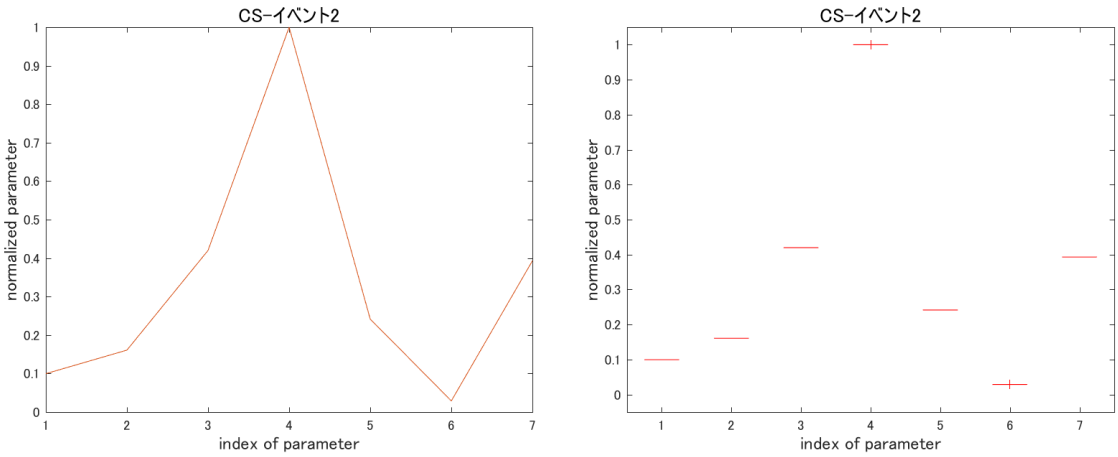

図 4-27　イベント 2 の同定パラメータ（Cuckoo Search）

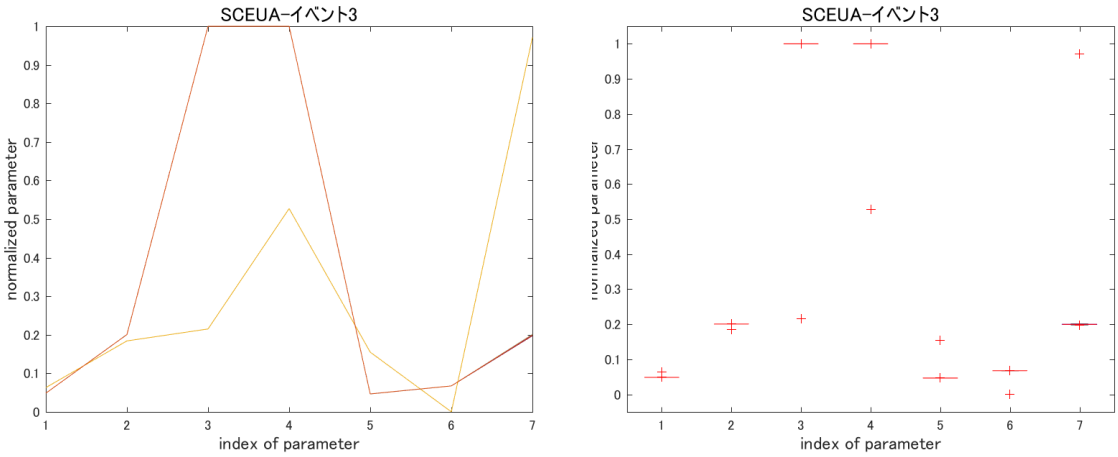

図 4-28　イベント 3 の同定パラメータ（SCE-UA 法）

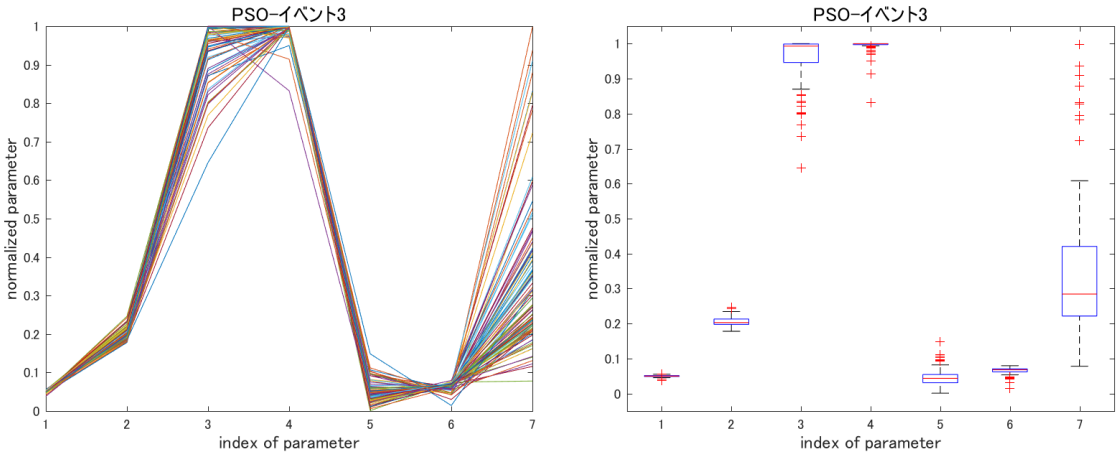

図 4-29　イベント 3 の同定パラメータ（PSO）

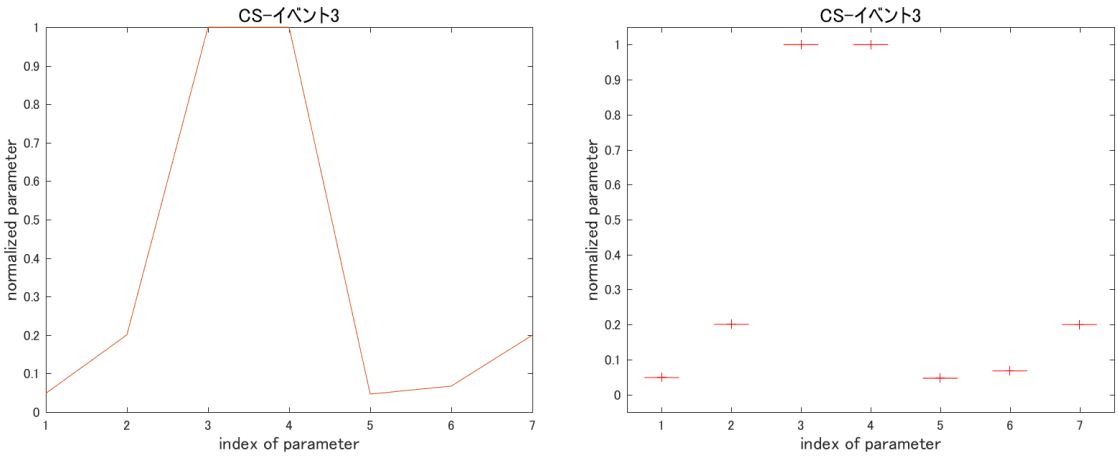

図 4-30　イベント 3 の同定パラメータ（Cuckoo Search）

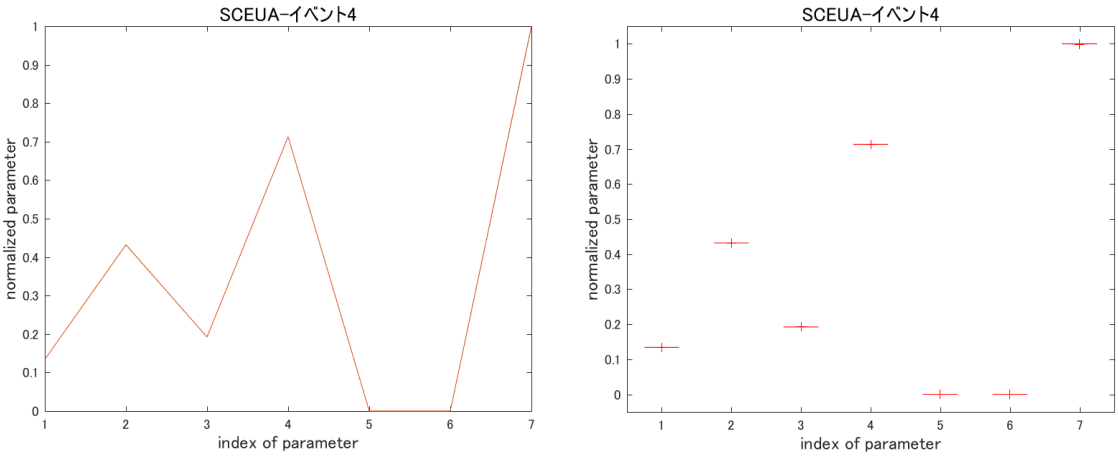

図 4-31　イベント 4 の同定パラメータ（SCE-UA 法）

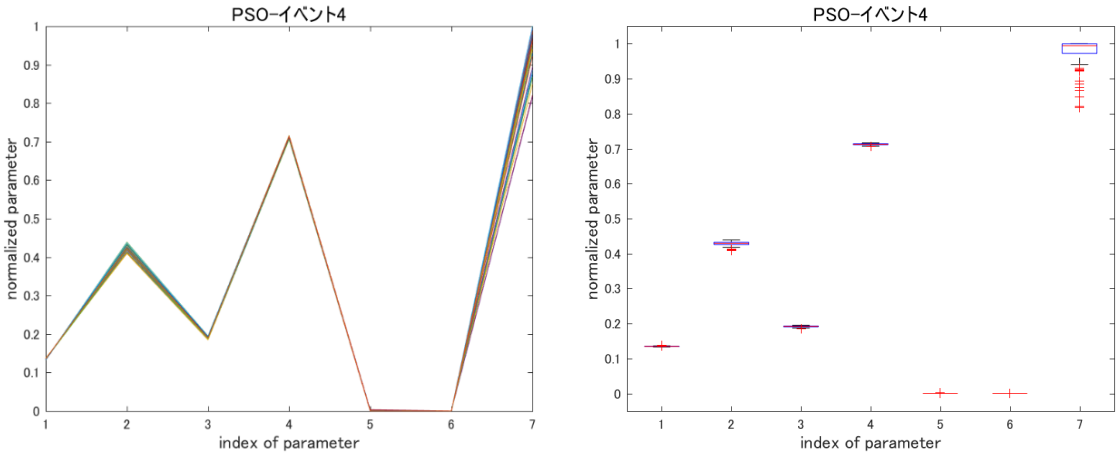

図 4-32　イベント 4 の同定パラメータ（PSO）

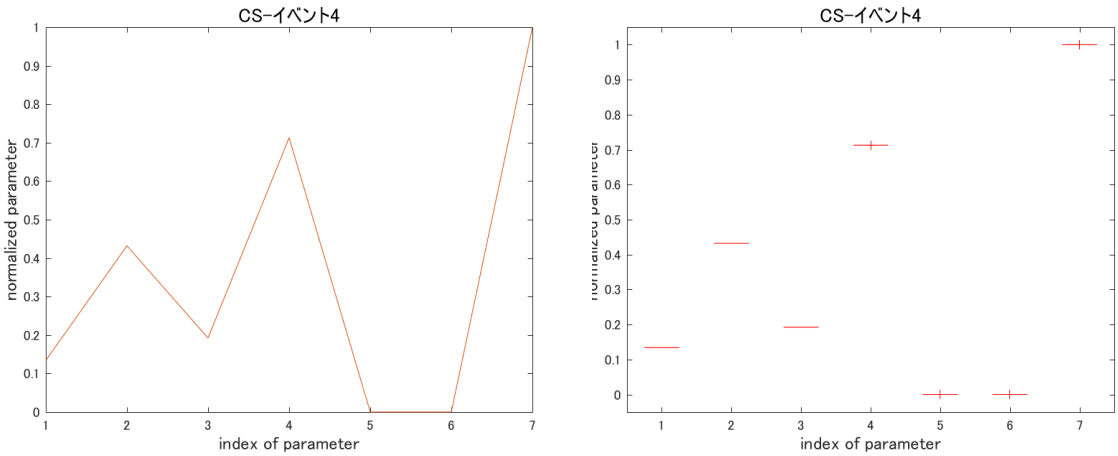

図 4-33　イベント 4 の同定パラメータ（Cuckoo Search）

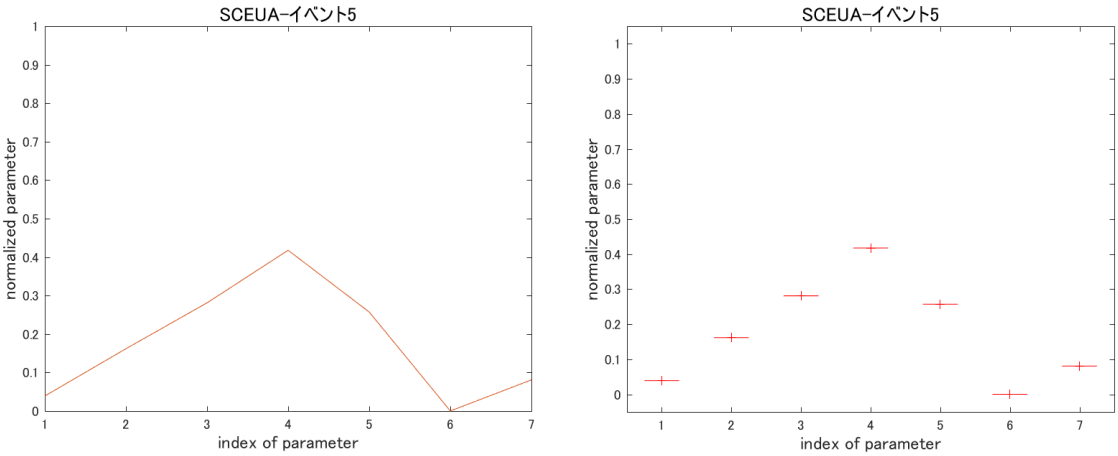

図 4-34　イベント 5 の同定パラメータ（SCE-UA 法）

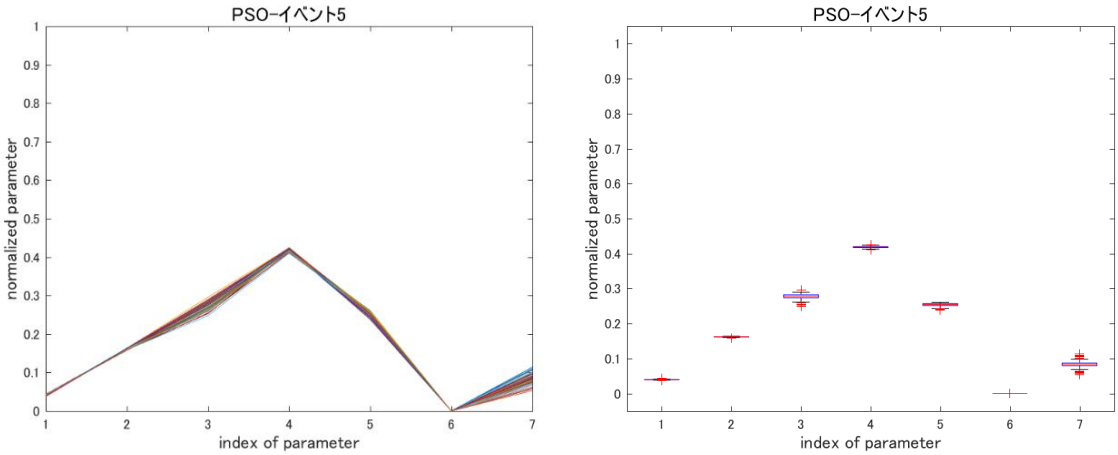

図 4-35　イベント 5 の同定パラメータ（PSO）

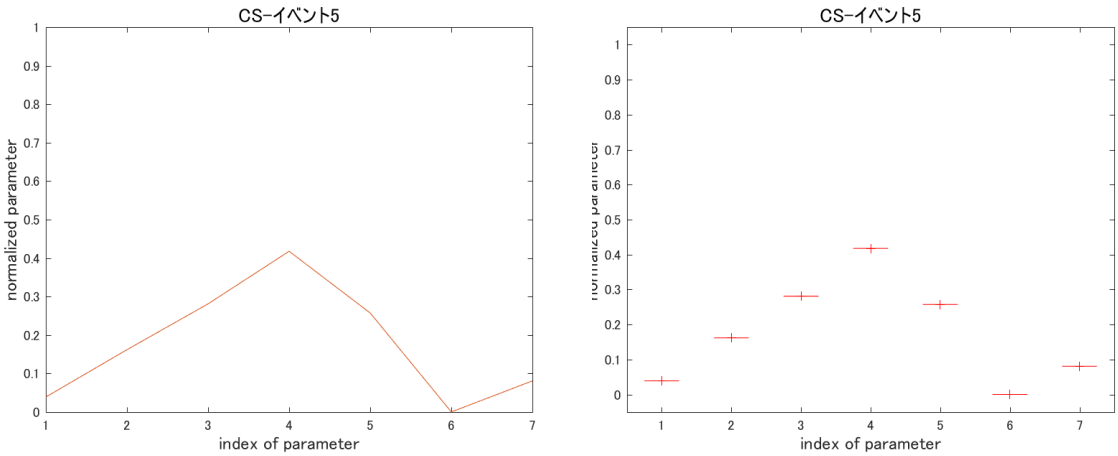

図 4-36　イベント 5 の同定パラメータ（Cuckoo Search）

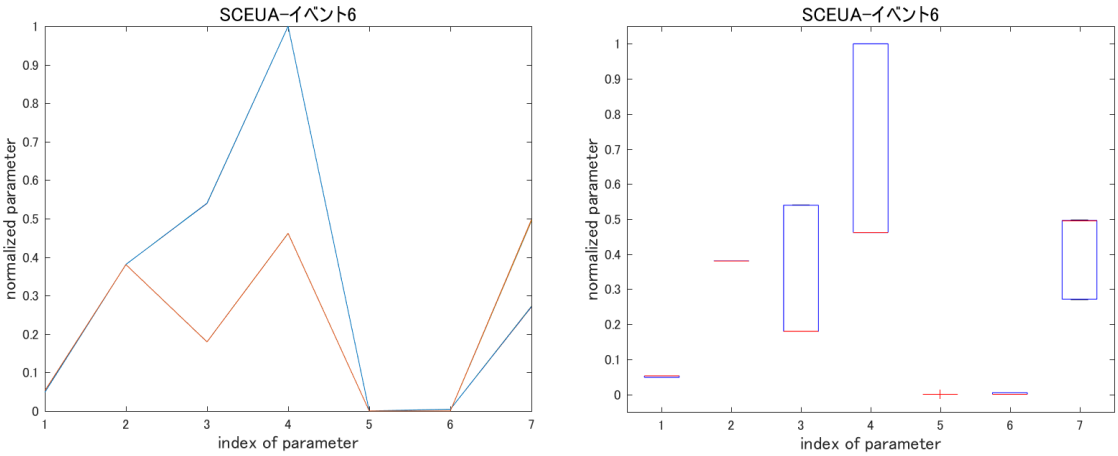

図 4-37　イベント 6 の同定パラメータ（SCE-UA 法）

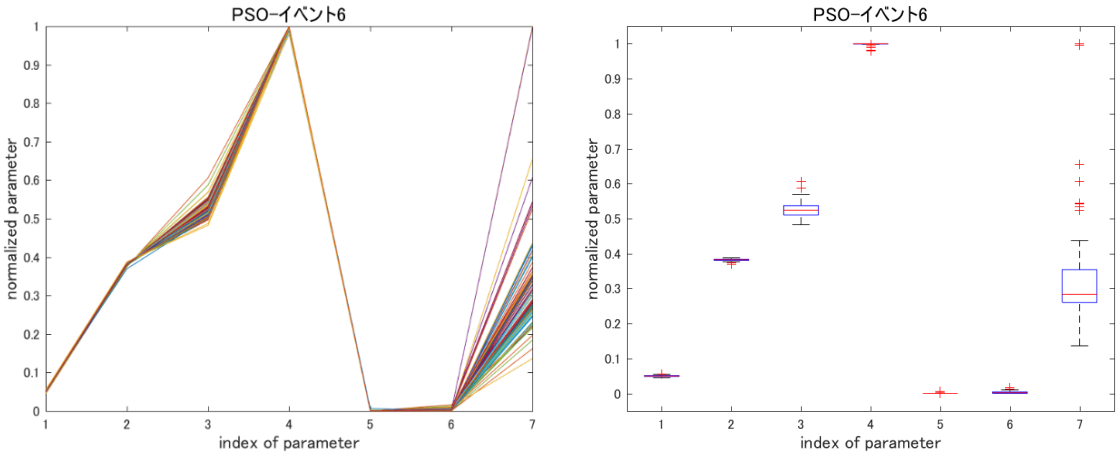

図 4-38　イベント 6 の同定パラメータ（PSO）

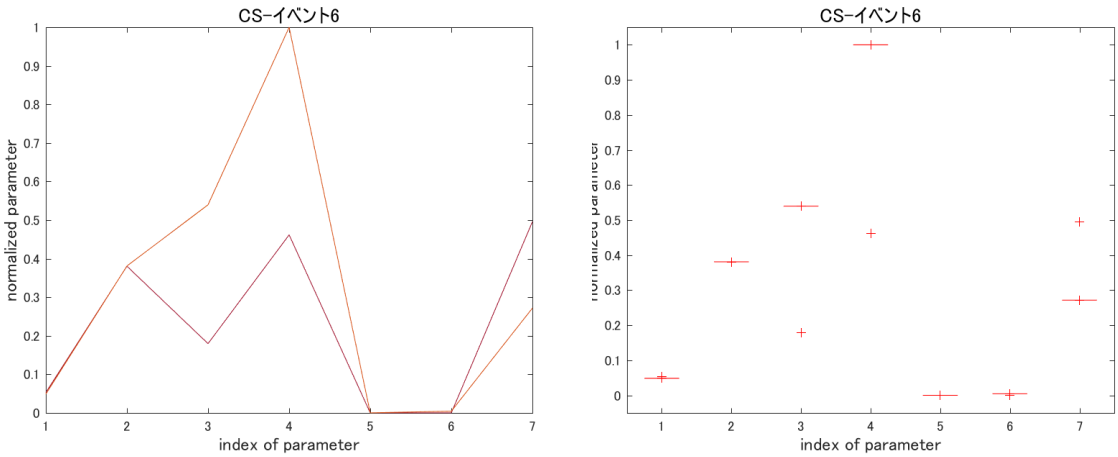

図 4-39　イベント 6 の同定パラメータ（Cuckoo Search）

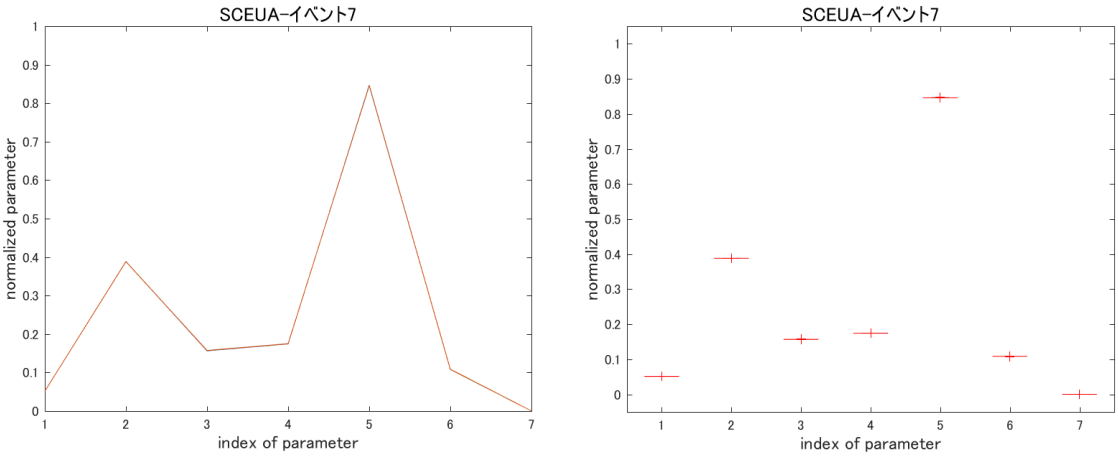

図 4-40　イベント 7 の同定パラメータ（SCE-UA 法）

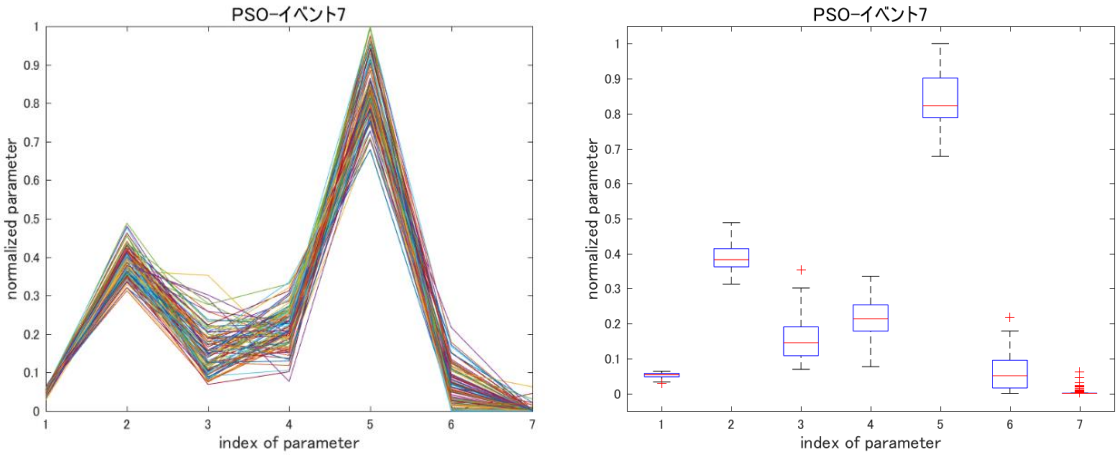

図 4-41　イベント 7 の同定パラメータ（PSO）

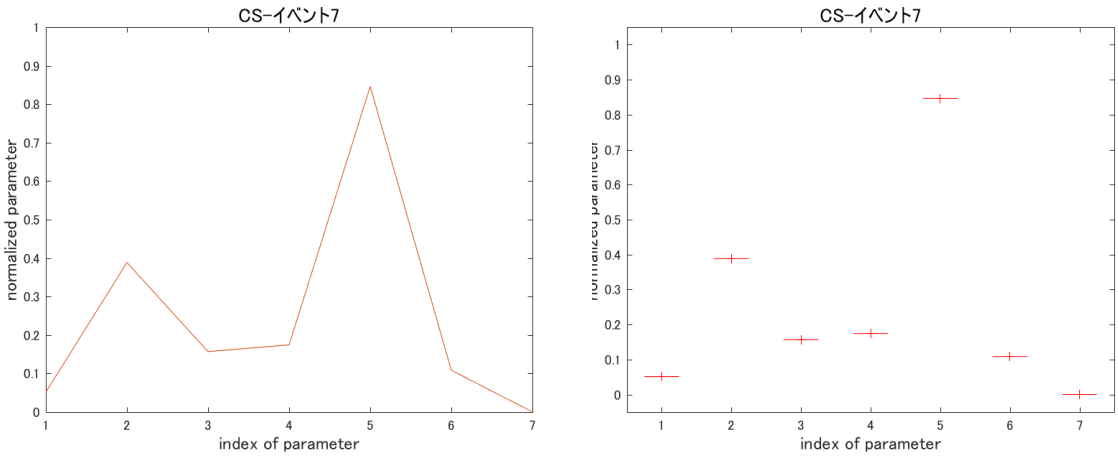

図 4-42　イベント 7 の同定パラメータ（Cuckoo Search）

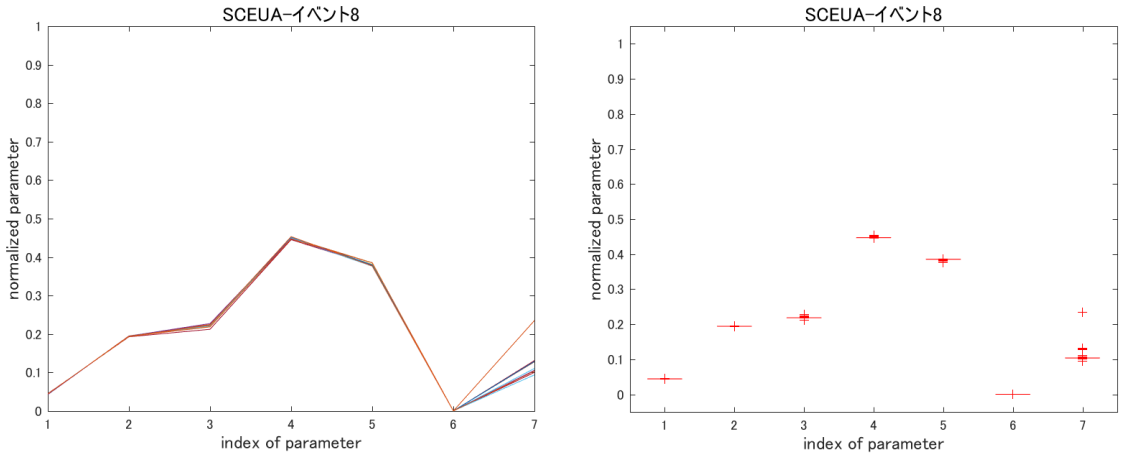

図 4-43　イベント 8 の同定パラメータ（SCE-UA 法）

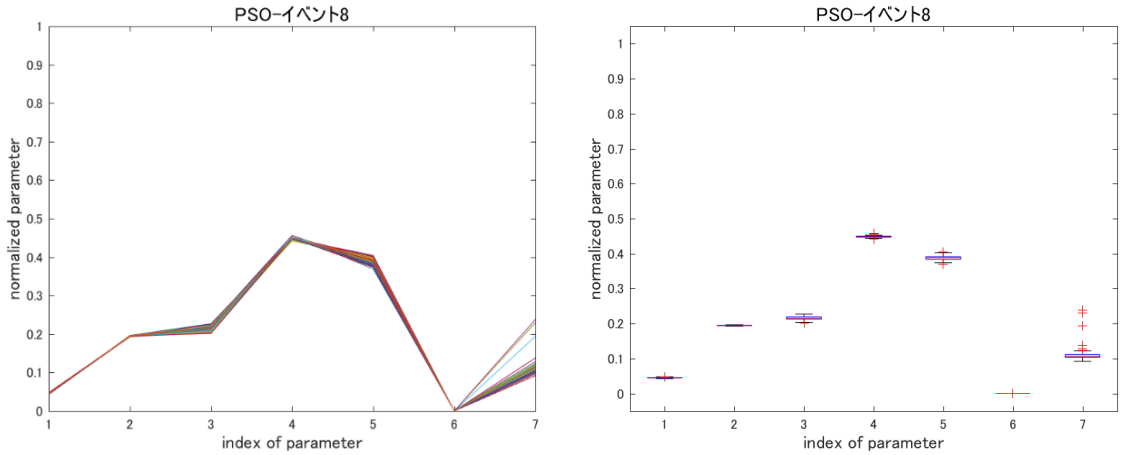

図 4-44　イベント 8 の同定パラメータ（PSO）

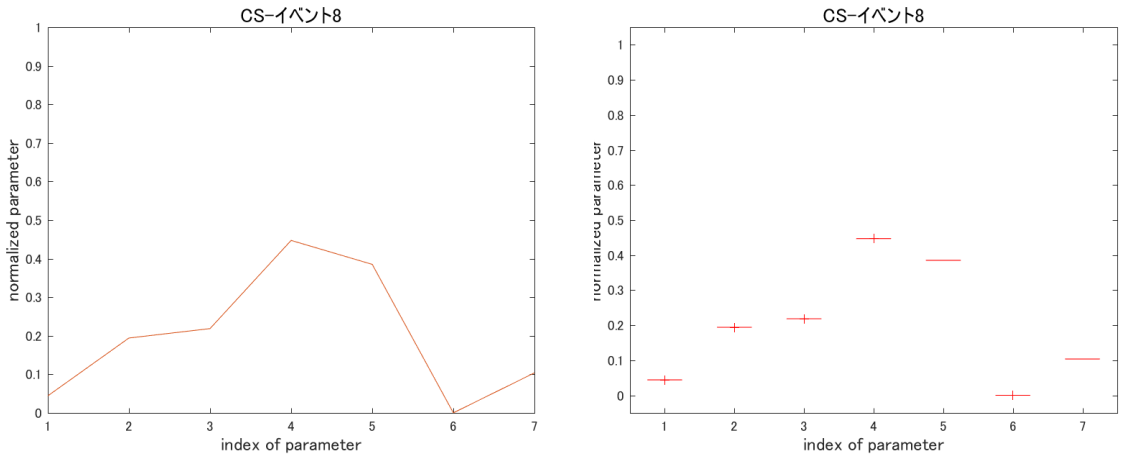

図 4-45　イベント 8 の同定パラメータ（Cuckoo Search）

実際に収束したパラメータ値を示す，**図 4-22** から**図 4-45** を見ると，PSO のばらつきの大きさが突出している．特にイベント 1,3,7 で顕著である．第２章で述べた通り，PSO のパラメータは，ベンチマーク関数である Ackley 関数，Rastrigin 関数を上手く最適化できるものとしたため，都市貯留関数モデルに対しては，解の収束性能が適切ではなかったものと考えられる．また使用した PSO は最も単純なものである．**図 2-4-1** と**図 2-4-2** を見ると，Ackley 関数，Rastrigin 関数は中央に向かって凹んだ形状をしており，PSO の特徴とする，粒子群がこれまでに見つけた最良位置に収束するというアルゴリズムと相性が良いように考えられる．特に粒子数が十分に多かった場合，はじめは平均的に散った粒子が，互いに情報を共有し，ベクトルの集中する中央に向かって収束してゆく様子が容易に想像できる．一方，イベント 1,3,7 の解空間は局所解の峰が複雑に存在しているために，PSO では上手く最適化が出来なかったように考えられる．大域的探索法には，「intensification」と「diversification」の 2 要素が重要であるとされており [17]，それぞれ「集約化」，「多様化」と訳すことができる．本研究で設定した PSO のパラメータでは，「多様性」は確保できていたものの，「集約化」に関しては不十分であったと考えられる．そのため，特定の降雨イベントに対して，固有のパラメータ値に収束するという性能が欠けていたものと言える．ところで，特定のイベントに対して最適なパラメータを同定した場合，オーバーフィッティングといって，他のイベントに対して汎用でないパラメータに収束するという問題が存在する．この問題に対し，目的関数を複数設ける多目的最適化などが提案されているが，PSO の緩やかな収束性能は，オーバーフィッティングに対して有効なのではないかと考えられる．

ハイドログラフの形状に関連して考察を行うと，複数ピークをもつイベント 2,4,5,8 では，SCE-UA 法・PSO・Cuckoo Search に共通して，収束したパラメータ値にばらつきが小さいことがわかる．これは，複数ピークのハイドログラフをもつ降雨イベントの解空間は，比較的単純であり，パラメータ同定が容易なものと考えられる．実際に，これらのイベントでは Cuckoo Search は 100 回の試行で全てが単一のパラメータ値に収束しており，SCE-UA 法もイベント 8 を除くと単一パラメータに収束している．一方で，単一ピークのイベント 1,6 では，最適パラメータに対して収束性の良い Cuckoo Search であっても，一部が準最適解に収束している．これは，単一ピークのハイドログラフをもつイベントが，複数ピークと比較して解空間が複雑であることに起因しているものと考えられる．

都市貯留関数モデルの各パラメータについて考察すると，index=1 であるパラメータk_1に関しては，比較的ばらつきが小さいことがわかる．これはパラメータk_1が都市貯留関数モデルに与える影響が大きく，最も重要な要素であるためだと考えられる．

計算時間を考慮すると，SCE-UA 法は PSO・Cuckoo Search と比較して 15%程度であったことから，SCE-UA 法が効率的な手法であると言える．イベント 1,6 を除くと，Cuckoo Search と同様の最適値に収束していることからも，強力な探索法であると考えられる．しかし，イベント 1,6 のような準最適解に陥りやすい降雨イベントの存在も明らかとなった．**図 4-6** の箱ひげ図によると SCE-UA 法は，イベント 1 では 75%以上 100%未満，イベント 6 では 50%以上 75%未満の確率で準最適解に陥っていることを考慮すると，SCE-UA 法による都

市貯留関数モデルのパラメータ同定には，乱数を変化させて，複数回実行すべきであるということが示唆される．あるいは，本研究や先行研究で採用してきたSCE-UA法の設定では不十分な洪水イベントが存在することから，集団数を増やしたり，計算終了条件である世代数を増やしたりするべきであると考えられる．一方，Cuckoo Searchは全イベントに対して，25%未満の確率で準最適解に収束していることを考えると，乱数を変化させて2回以上パラメータ同定を実行すると，どのような洪水イベント対しても妥当な結果が得られるものと考えられる．

最後に，目的関数の評価回数によって，どのようにRMSEが収束したかを確認する．**図 4-46**から**図 4-53**には，イベント1から8において，各手法で乱数を変化させて100回ずつパラメータ同定を行って，目的関数評価回数を基準にプロットしたものである．縦軸はRMSEを示しており，横軸は目的関数評価回数である．PSOとCSはその数が150,000回であるが，SCE-UA法は計算の試行によって数が異なり，概ね25,000回前後であった．この理由より，SCE-UA法はプロットの範囲が異なる．

図 4-46から**図 4-53**を見ると，全イベントを通して，SCE-UA法が素早い収束を示している．ただし，イベント1では準最適解に陥っているものが多く見られる．イベント1のSCE-UA法の一部は，途中まで準最適解に嵌ってしまっているものの，抜け出して最適解付近に収束していることも確認できる．

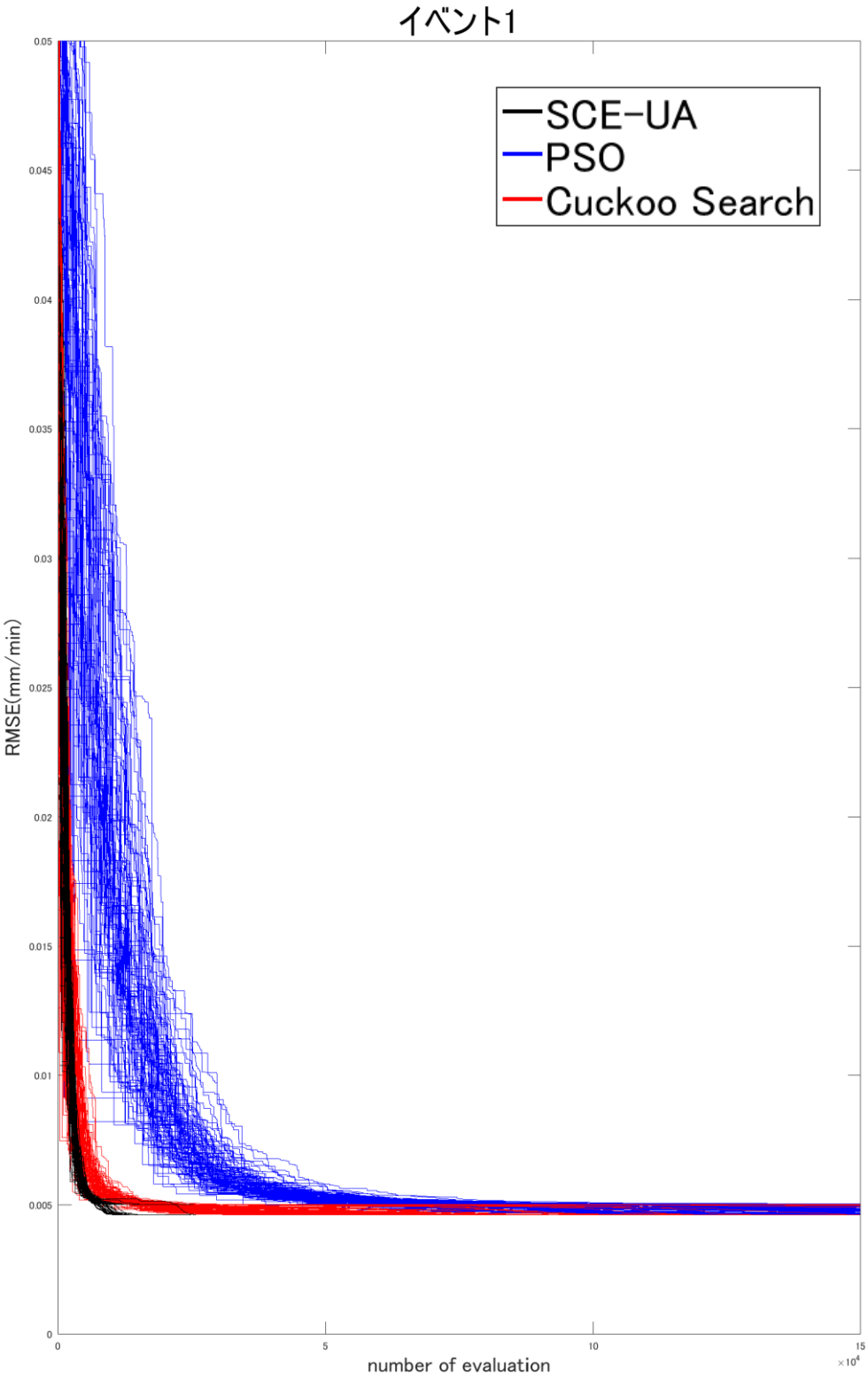

図 4-46　イベント 1 のRMSEの変動

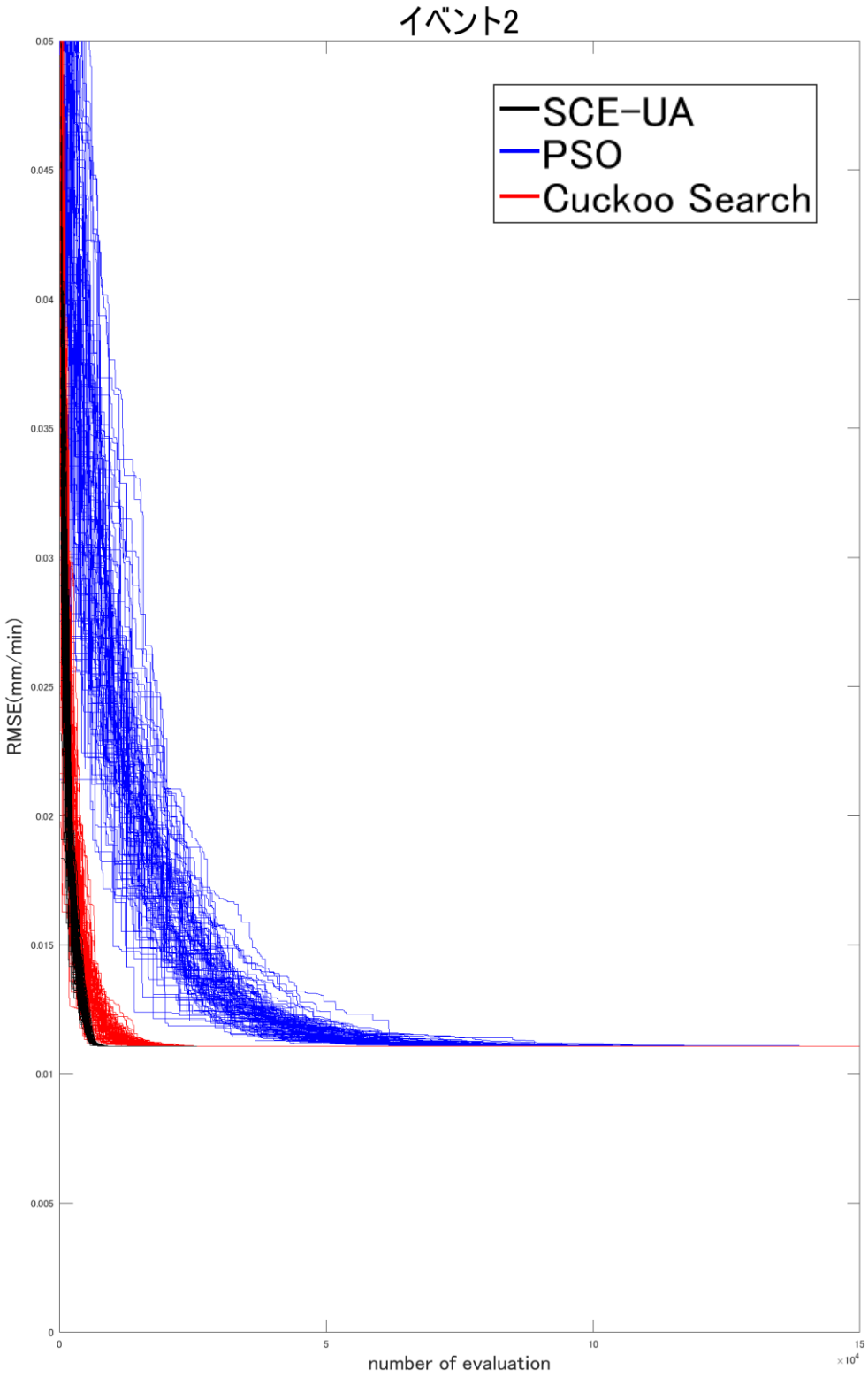

図 4-47　イベント 2 のRMSEの変動

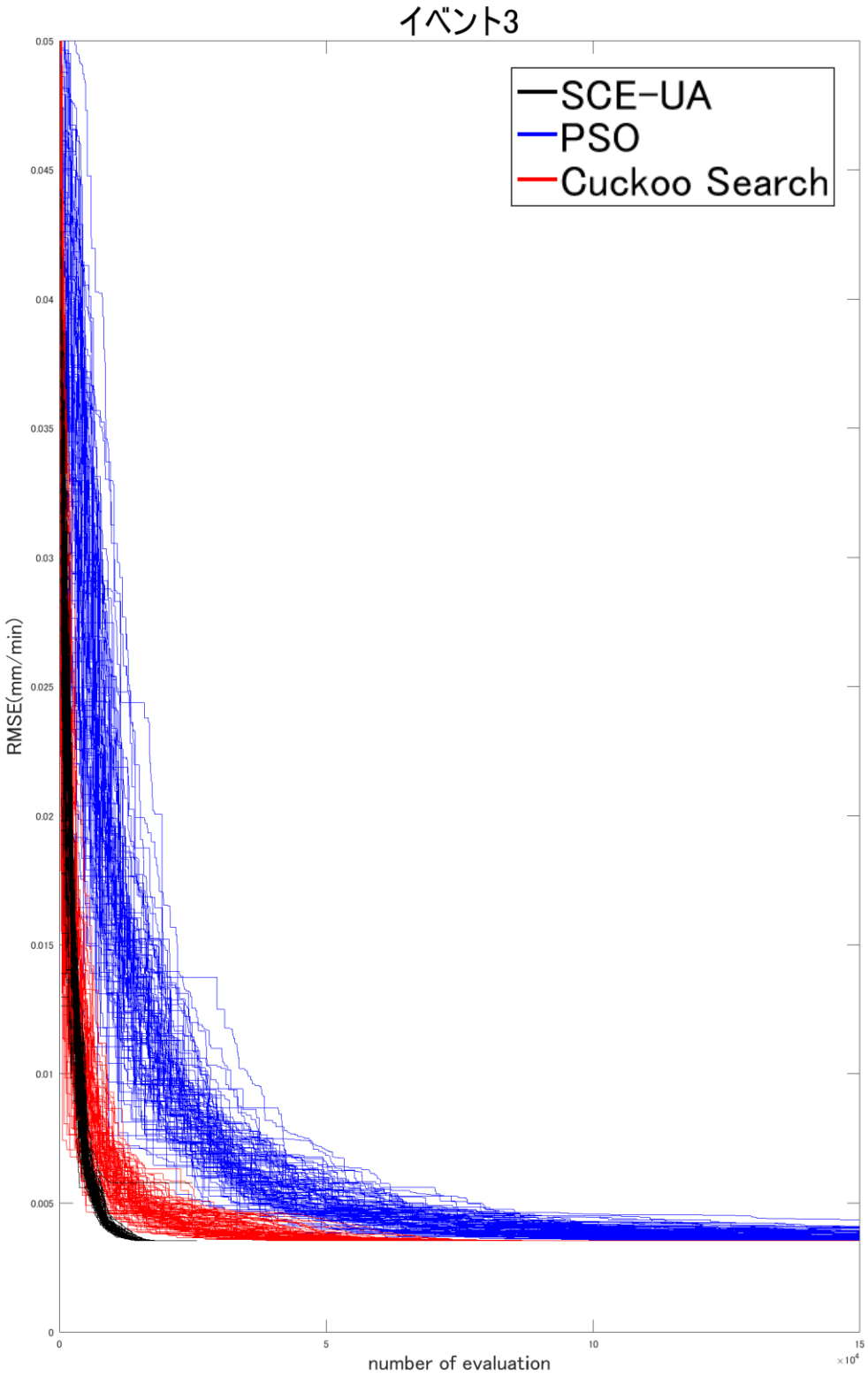

図 4-48　イベント 3 のRMSEの変動

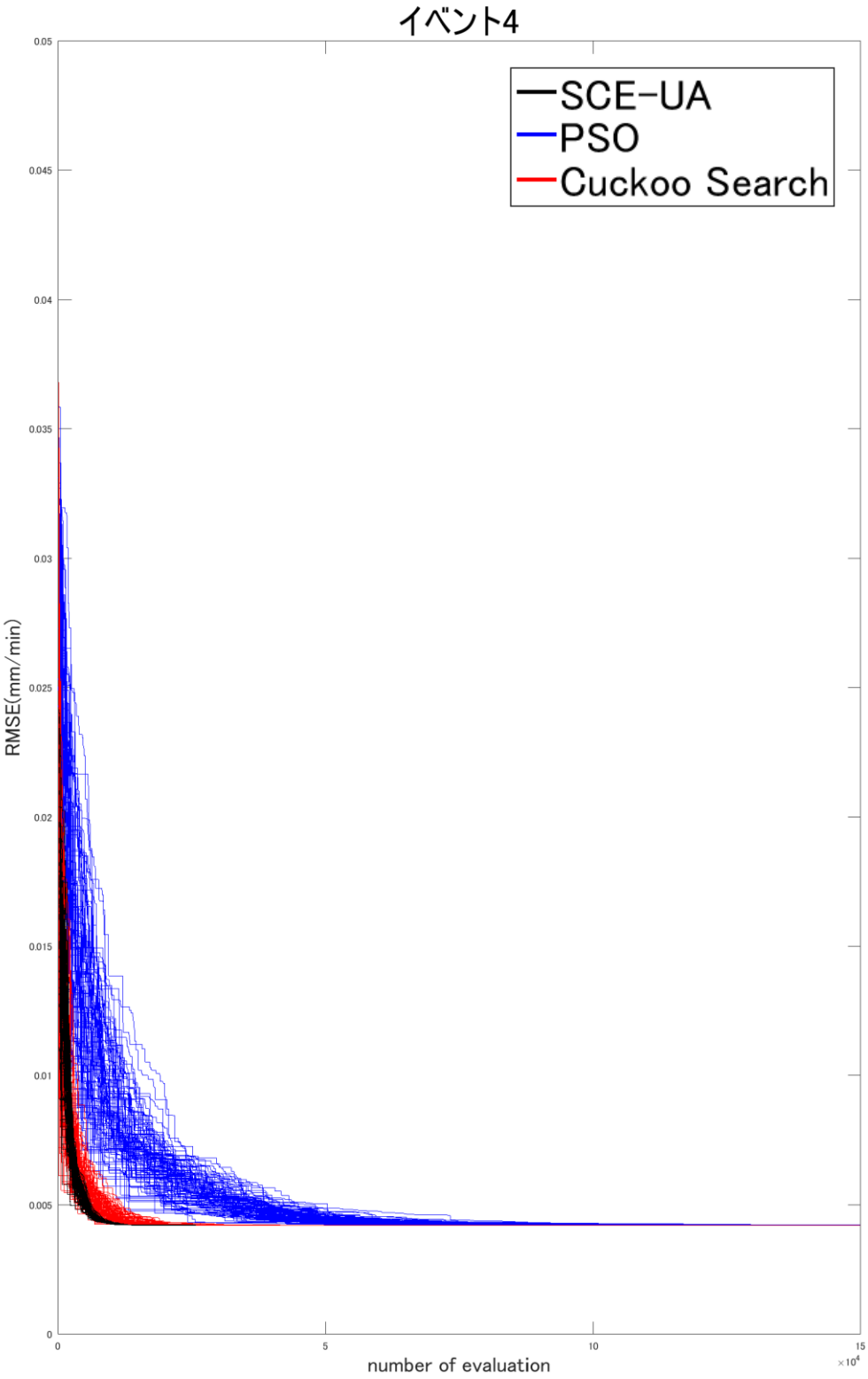

図 4-49　イベント 4 のRMSEの変動

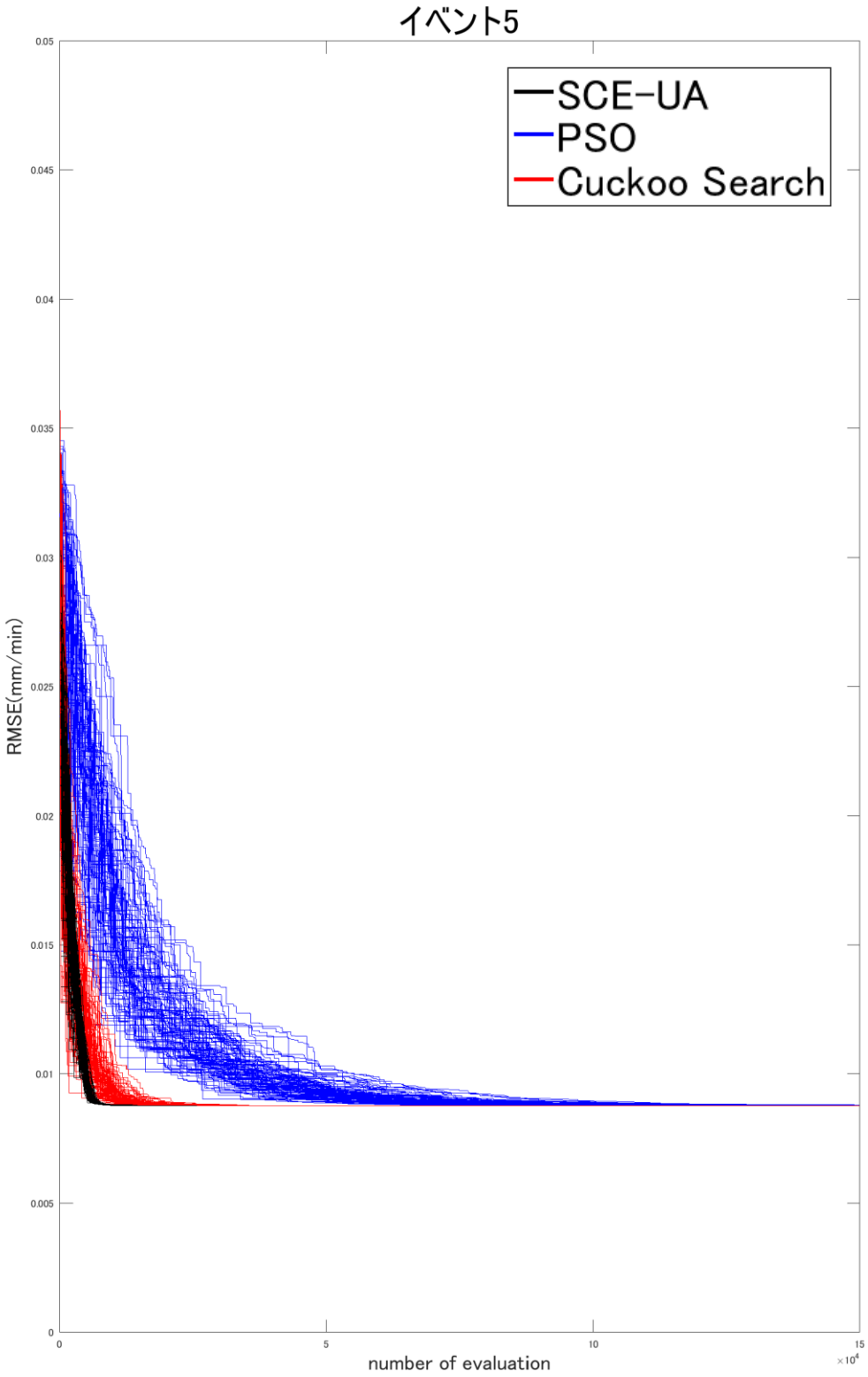

図 4-50　イベント 5 のRMSEの変動

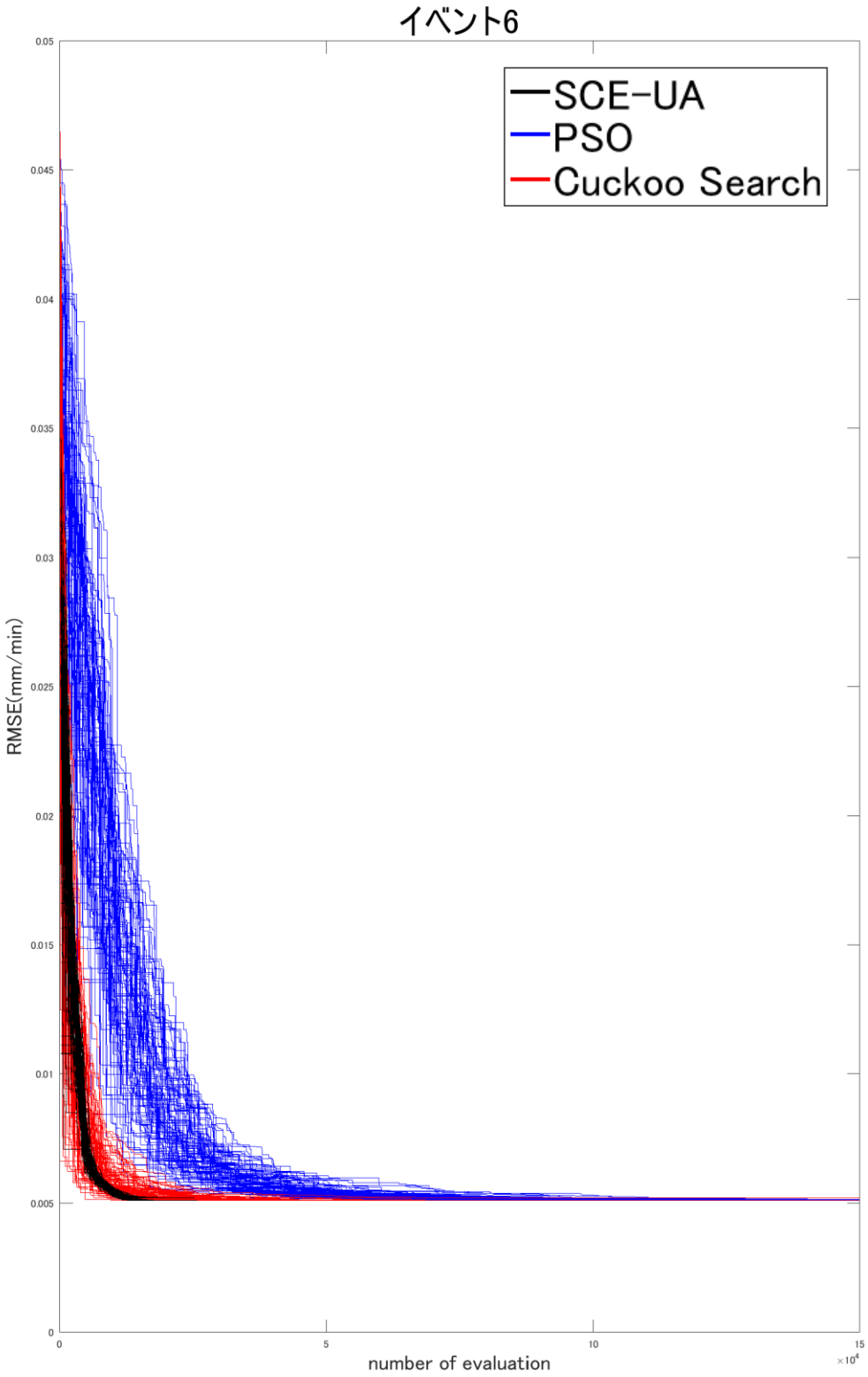

図 4-51　イベント 6 のRMSEの変動

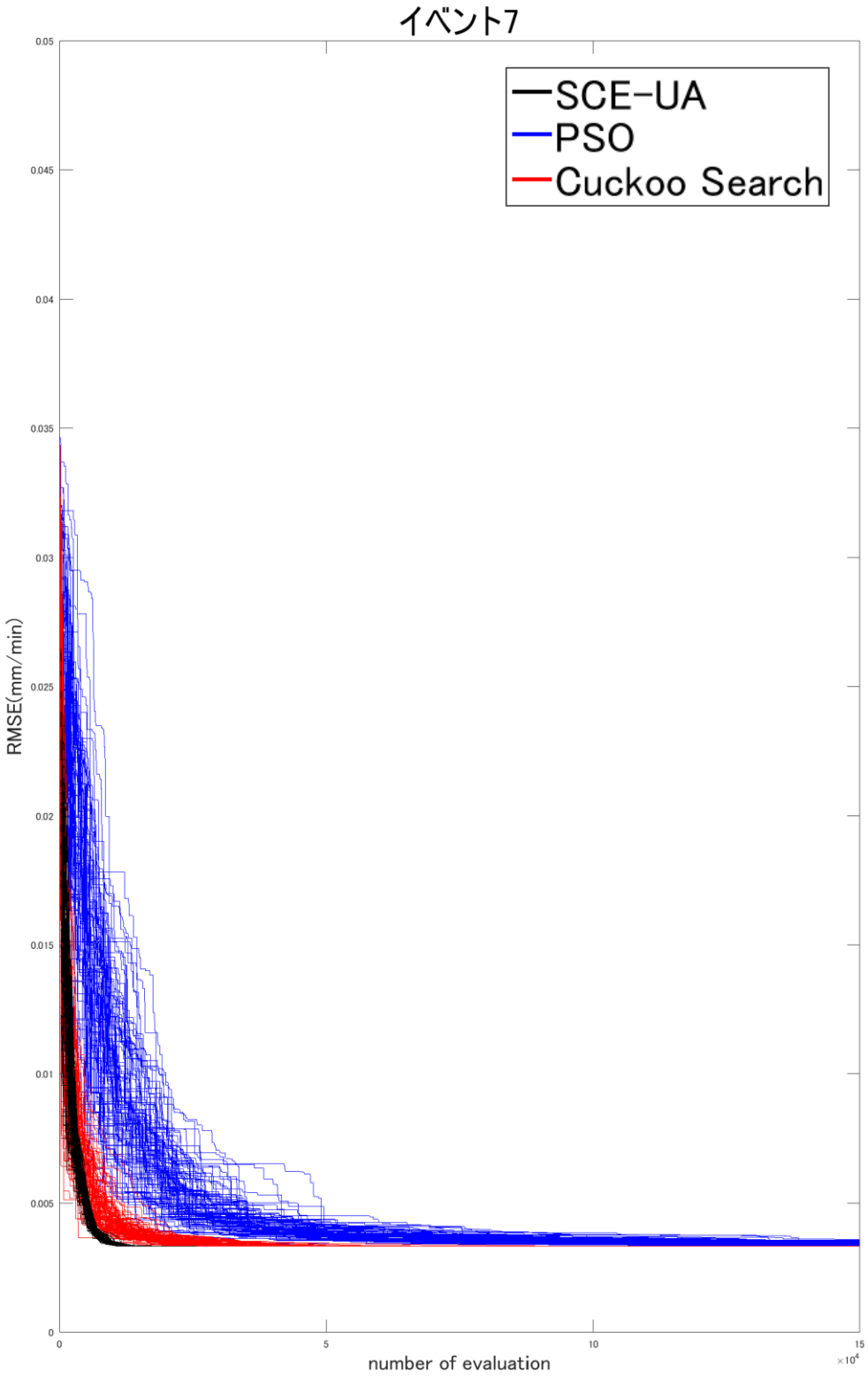

図 4-52　イベント 7 のRMSEの変動

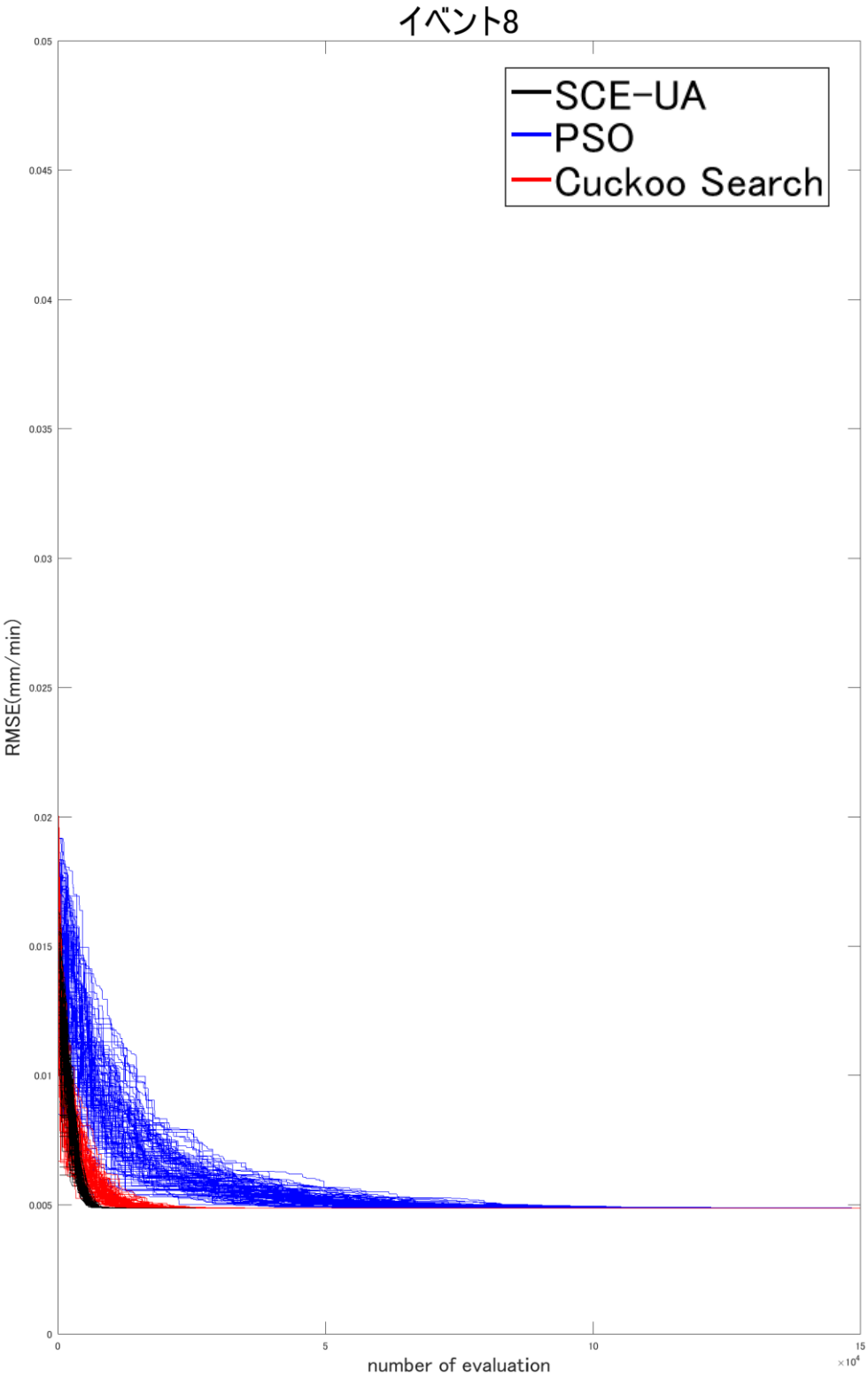

図 4-53　イベント 8 のRMSEの変動

第５章

結　論

第５章　結論

本研究では，メタヒューリスティクスな進化的計算手法である，SCE-UA 法，粒子群最適化（PSO：Particle Swarm Optimization），カッコウ探索（CS：Cuckoo Search）の 3 手法を適用し，都市中小河川のハイドログラフを良好に再現でき，7 つの未知パラメータを持つ都市貯留関数モデルのパラメータ同定を行い，その探索性能について比較検討を行った．

使用した 3 つの進化的計算手法は，それぞれ最適化の計算を行う前に，設定すべき項目がいくつか存在する．それらは各手法の探索性能や計算時間をコントロールするものであり，適用する問題に応じて，適切に設定すべきものである．SCE-UA 法については，開発者 Duan らの推奨設定が存在するため，それを採用した．計算終了条件については，都市貯留関数モデルへの適用実績から，50 世代とした．PSO・CS については都市貯留関数モデルに使用されたことがないため，ベンチマーク関数を良好に最適化できる設定とした．ベンチマーク関数には，引用されることの多い Ackley 関数と Rastrigin 関数を採用し，変数の数は都市貯留関数モデルと同じ 7 つとした．ベンチマークにより，PSO は粒子数$n = 50$，パラメータ$w = 0.0004$，$c_1 = 1.2$，$c_2 = 0.012$とし，CS は$n = 30$，$p_a = 0.25$とした．PSO の計算終了条件は$k_{max} = 3000$，CS は$t_{max} = 2500$とすることで，ともに目的関数の評価回数は 150,000 回となった．SCE-UA 法の評価回数は計算の試行によって変化するものの，概ね 25,000 回前後であった．パラメータ同定にかかる計算時間は目的関数の評価回数に比例し，SCE-UA 法のそれは PSO・CS の 15%程度であった．

仮想流域データに対しパラメータ同定を行った結果，SCE-UA 法が最も高い探索性能を示し，CS，PSO の順に続いた．真値パラメータは先行研究の神田川上流域のパラメータを参考に設定したものであり，入力した降雨はクリーブランド型の設計降雨である．東京管区気象台 20 年確率で 180 分の降雨を作成し，中央集中型に配置した．この降雨と真値パラメータを都市貯留関数モデルに入力すると計算流量が得られる．仮想流域データに対するパラメータ同定は，この計算流量と設計降雨を入力することで，3 つの最適化手法により真値パラメータの探索を行った．パラメータが真値の場合，目的関数であるRMSEは 0 となる．RMSE $\leq 10^{-4}$となったものが真値を同定したとすると，SCE-UA 法は 99%，CS は 62%，PSO は 0%の確率で真値を同定した（3 手法とも乱数を変化させて各 100 回ずつ試行した）．以上より，仮想流域に対しては SCE-UA 法が，最も強力かつ効率的な手法であることが確認された．

実流域である善福寺川西田端橋上流域の 2013 年～2015 年の 8 洪水に対してパラメータ同定を行った結果，乱数を変化させて 100 回ずつ試行した平均で，3 手法とも同程度の目的関数値を得ることができた．しかし全イベントを通して最も良い結果を示したのは CS であった．CS は，3 手法の各 100 回の試行で求まった最適値と同様のパラメータを高確率で同定しており，その頑健さを示した．SCE-UA 法はイベント 1,6 で多く局所解に至り，PSO はイベント 1,3,7 で局所解を多く同定した．SCE-UA 法は収束するパラメータがばらつくということはなく，最適値か，特定の準最適解に収束していた．一方，PSO は収束パラメータのばら

つきが大きく，特に成績の悪かったイベント 1,3,7 で顕著であった．以上より，都市貯留関数モデルを実流域に適用し，パラメータ同定を行う場合，降雨イベントによっては局所解に至りやすいものがあることが確認された．本研究では，SCE-UA 法のパラメータを先行研究に倣って設定したものの，今後はその対策が求められる．具体的には，集団数を増やす，計算終了条件の世代数を大きくする，乱数を変化させてパラメータ同定を複数回行うというものが挙げられる．また，計算時間に余裕がある場合は，本研究で使用した設定の CS を用いるということも考えられる．目的関数としたRMSEの収束の早さという観点では，SCE-UA 法が最も高性能であった．計算の効率を考えると SCE-UA 法が優れており，前述した通り，準最適解に陥りにくいような対策を講じた上での運用が望まれる．その他，世代数が進んで探索の動きが鈍くなったときに，幾つかの個体に Cuckoo Search で使用したLévy Flightsを作用させ，強制的に動きを取り戻させて効果的な最適値の探索を行うといったことも考えられる．

参考文献

1) 高崎忠勝，河村明，天口英雄，荒木千博：都市の流出機構を考慮した新たな貯留関数モデルの提案，土木学会論文集 B Vol.65 No.3，pp.217-230，2009.
2) Duan，Q.，Sorooshian，S. and Gupta，V.K.：Effective and efficient global optimization for conceptual rainfall-runoff models，*Water Resources Research*，Vol.28，No.4，pp.1015-1031，1992.
3) J.Kennedy，R.Eberhart：Particle Swarm Optimization，IEEE Int. Conf. on Neural Networks，Vol.4，pp1942-1948，1995.
4) Xin-She Yang, Suash Deb：Cuckoo Search via Levy Flights， Nature & Biologically Inspired Computing，No.37，pp.210-214，September 2009.
5) 田中丸治哉：タンクモデル定数の大域的探索，農業土木学会論文集，No.178，pp.103-112，1995.
6) 藤原洋一，田中丸治哉，畑武志，多田明夫：流出モデル定数の最適同定における誤差評価関数の選択に関する研究，農業土木学会論文集 No.225，pp.137-149，2003.
7) 森永陽子，河村明，神野健二：SCE-UA 法による貯留関数モデルの大域的パラメータ同定について，土木学会西部支部研究発表会講演集，pp.B198-B199，2002.
8) 斎藤利通：粒子群最適化と非線形システム，IEICE Fundamentals Review，Vol.5 No.2.
9) 多田毅：PSO アルゴリズムによる流出モデルパラメータの最適化，水文・水資源学会誌 J. Japan Soc. Hydrol. & Water Resour.，Vol.20 No.5，pp.450-461，Sep.2007.
10) Sandeep Solanki：psoToolbox，「MathWorks File Exchange」，< https://jp.mathworks.com/matlabcentral/fileexchange/44154-psotoolbox>
11) Xin-She Yang：Cuckoo Search (CS) Algorithm，「MathWorks File Exchange」，<https://jp.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search--cs--algorithm>
12) 天口英雄，安藤義久：SMPT モデルを用いた分布型水循環モデルの開発と東川試験流域への適用，水工学論文集，第 45 巻，pp.97-102，2001.
13) 東京都 建設局土木技術支援・人材育成センター技術支援課：神田川善福寺川流量観測調査委託，2013.
14) 東京都水道局総務部調査課：平成 26 年度事業年報，pp.119，2015.
15) 東京都：荒川水系神田川流域河川整備計画，2016.
16) 天口英雄，河村明，高崎忠勝，荒川大樹：東京都水防災システム降雨データの特性，水文・水資源学会 2007 年研究発表会要旨集，pp14-15，2007.
17) Yang, X.-S., and Deb, S. (2010), “Engineering Optimisation by Cuckoo Search”,Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4,330–343 (2010).

謝辞

はじめに，3 年もの間お世話になった河村教授に感謝申し上げます．学部 3 年から首都大学東京に編入学し，研究室選びで悩んでいたところ，河村先生の授業が面白かったことがきっかけでこの水文研究室を選択しました．この研究室を選んでいなければ，趣味とする旅行，富士山登頂，フィリピン留学，JICA 就職など，到底叶えられなかったと思います．下呂温泉で行った 2015 年のゼミ合宿では，一切の前情報をなしにフィリピン留学を勧められ，それから私の人生の方向が変わりました．河村先生は研究者としてだけではなく，教育者としても尊敬する部分が大きいため，これからも一層のご活躍を祈っております．次に，天口助教には，研究内容をはじめとして，プログラミング技術や PC 回りの設定など，多方面にわたりお世話になりました．河村先生のような強烈なコンダクターが居るところで，水文研究室が上手く機能しているということは，何より天口先生の存在が大きいかと思います．またお忙しいところ，副査を担当して下さった稲員教授に御礼申し上げます．稲員先生には就職担当教員としても大変お世話になりました．そして東京都土木技術人材育成センターの高崎様には，研究面のみならず，副査にもなって頂きました．他の学生の研究もご覧になっていたことから，私はなるべく手間とならないように心がけようとしましたが，高崎様にはいつも一歩二歩進んだアドバイスや関連データの提供，ゼミの開催などでお世話となりました．東京都建設局河川部の石原様は休日に大学でお目にかかることが多く，大変熱心に研究や学生の指導に当たられていることが印象的でした．また昨年は副査をされていたとのことで，私と高崎様が円滑にやりとり出来るようにご配慮もいただきました．

研究室の学生にも非常に支えて頂きました．PhD 学生の Nuong さんには，私がフィリピンに留学へ行く前，英語プレゼンを見て頂いたり，励ましの言葉などをかけてもらったりしました．私の帰国後は，幾らか成長した英語力で，チューターとしてサポートが出来たかと思います．Saritha さんには，しばしば英語を聞き直してしまい，申し訳ありませんでした．同期の学生である大崎は，主要諸元が似ていることから，いつも切磋琢磨する間柄でした．また解さんには，そんな我々を温かく見守って頂きました．後輩の M1 諸君は，可愛げのないくらい隙がないという意味で，先輩としてはプレッシャー感じるところがありました．私がいきなりフィリピン留学に行ってしまうことで大塚・戸野塚の面倒を見ることを放棄しましたが，二人には謎の文通のようなもので元気づけて貰いました．太田・高山は，体育会系なところや強気な姿勢など，学ぶところがありました．ハイさんは，私の外国人留学生観を破壊されるほど面白く，少し不真面目で，そして気前のいい人でした．そして B4 諸君は，私が先輩として影響を与えられるような存在になれなかったという点で，残念に思うところがあります．皆さんが卒業するときに，水文研で何らかの転機や成果が見つかって，素晴らしい学生生活となることを期待しております．

水文研究室 金塚 匠（かなづか たくみ）

付　録

A-1　SCE-UA 法の MATLAB コード

SCE-UA 法の MATLAB コードを示す．適用するデータは仮想流域で，真値が明らかなものである．MATLAB のカレントディレクトリをメインプログラムの存在するフォルダに設定し，「SCEUA_USFM_VirtualRain.m」を読み込むことで動作する．途中，都市貯留関数モデルを Runge-Kutta-Gill 法で解くサブルーチンである「funUsfmRKG.m」を参照しており，その中で都市貯留関数モデルの計算式である「Usf.m」を自動的に読み込むという構造になっている．

「SCEUA_USFM_VirtualRain.m」

```
%%% SCEUA PROGRAM FOR URBAN STORGE FUNCTION MOÐEL %%%

clear;
tic;
nopt=7;        % Number of variables to be optimized
nc=20;         % Number of Complex 20 is recommended
ncp=2*nopt+1;  % Number of Complex Population
nscp=nopt+1;   % Number of Sub-Complex Population
ntp=nc*ncp;    % Number of Total Population
maxgene=50;   % Maximum Generation

OptResult=zeros(maxgene,nopt+1);

%% Ðata for Urban Storage Function Model %%
load designed_raunfall_20year_3hr_long.txt;
rain=designed_raunfall_20year_3hr_long(:,1);
obsq=designed_raunfall_20year_3hr_long(:,2);
Ev=designed_raunfall_20year_3hr_long(:,3);
Iu=designed_raunfall_20year_3hr_long(:,4);
Ou=designed_raunfall_20year_3hr_long(:,5);
qRmax=0.005; % maximum sewage discharge. assumed
Q0=obsq(1);
ndata=length(designed_raunfall_20year_3hr_long);

%%---------------------------------%%
x=zeros(ntp,nopt);
xf=zeros(ntp,1);
cx=zeros(ncp,nopt);
cf=zeros(ncp,1);
scx=zeros(nscp,nopt);
scf=zeros(nscp,1);
xbest=zeros(maxgene,nopt);
xworst=zeros(maxgene,nopt);
```

```
xfbest=zeros(maxgene,1);
xfworst=zeros(maxgene,1);
xrange=zeros(maxgene,nopt);

bl=[10  100  0.001 0.1 0.1 1  0.1]; % Lower Boundary
bu=[500 5000 0.05  1   1   50 1];  % Upper Boundary
range=bu-bl;
if range > 0
   else
   '*** ERROR *** At least one range is MINUS'
   end

%% rand('state',0)
% rng('default');
% rng(0);

%% -- START -- %%
nrun=100;
for irun=1:nrun
    Neval=0;
    fprintf ('\n SCE-UA Run Number --> %2g\n',irun);

%%% Initial Generation igene=1 %%%
igene=1;
fprintf ('\n GeneNo-->%3g',igene);

ibig=0;
ix1m=0;
R=rand(ntp,nopt);
for ii=1:nopt
    x(:,ii)=bl(ii)+range(ii).*R(:,ii);
end
% x(1,:)=1.5*[26.0  41.0  0.6  0.465  0.6];  % initially included x-point
for j=1:ntp
    [xf(j)]=funUsfmRKG(x(j,:),rain,obsq,Iu,Ev,Ou,qRmax,Q0);
end

% count number of evaluation
Neval=Neval+ntp;
xfx=[xf,x];
xfx=sortrows(xfx,1);
xf=xfx(:,1);
x=xfx(:,2:nopt+1);
xbest(igene,:)=x(1,:);
```

```
xfbest(igene)=xf(1);
xworst(igene,:)=x(ntp,:);
xfworst(igene)=xf(ntp);
xsort=sort(x);
xrange(igene,:)=xsort(ntp,:)-xsort(1,:);
fprintf('                                                                                xf=%6.7f
x=%6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f',xf(1),x(1,:));

%%
OptResult(1,1)=xf(1,1);
OptResult(1,2:8)=x(1,1:7);

%%%%%% Generation Loop %%%%%%
for igene=2:maxgene % Generation Loop
    fprintf ('\n GeneNo-->%3g',igene);

    ibig=0;
    ix1m=0;
    for inc=1:nc  % Complex Loop
        %% Assign Points into Complexes from Total Population %%
        k1=1:ncp;
        k2=inc+(k1-1)*nc;
        cx(k1,:)=x(k2,:);
        cf(k1)=xf(k2);
        nbeta=ncp; % recommended
        for ibeta=1:nbeta
%%%% Select Individuals into Sub-Complex (Extermination Room) %%%%
            for inscp=1:nscp
                ihit=1;
                while ihit > 0
                   ran1val=rand;
                   prob=0;
                   for incp=1:ncp
                        prob=prob+2*(ncp+1-incp)/(ncp*(ncp+1));
                        if ran1val <= prob
                            j=incp;
                            break;
                        end
                   end % incp
                   ihit=0;
                   if inscp == 1
                      k3(1)=j;
                   else
                      for jcheck=1:inscp-1
```

```
                        if j == k3(jcheck)
                           ihit=ihit+1;
                        end
                     end % jcheck
                     if ihit==0, k3(inscp)=j; end
                  end
               end % ihit

               scx(inscp,:)=cx(k3(inscp),:);
               scf(inscp)=cf(k3(inscp));
            end % inscp

%--- Sub-Complex Manipulation (exterminate the worst one) ---%
                        nalpha=1;  % recommended
            for ialpha=1:nalpha
            scfscx=[scf,scx];
            scfscx=sortrows(scfscx,1);
            scf=scfscx(:,1);
            scx=scfscx(:,2:nopt+1);
            scxbest=scx(1,:);
            scxworst=scx(nscp,:);
            scx1=scx(1:nscp-1,:);
            scx1mean=mean(scx1);
            scfworst=scf(nscp);
          % Reflection Step %
            scxnew=2*scx1mean-scxworst;
            if bl <= scxnew & scxnew <= bu
            else
               ran1=rand(1,nopt);
               scxnew=bl+range.*ran1;
            end
            [scfnew]=funUsfmRKG(scxnew,rain,obsq,Iu,Ev,Ou,qRmax,Q0);
            Neval=Neval+length(scfnew);
            if scfnew >= scfworst
               % Contraction Step %
               scxnew=(scx1mean + scxworst)/2;
               [scfnew]=funUsfmRKG(scxnew,rain,obsq,Iu,Ev,Ou,qRmax,Q0);
               Neval=Neval+length(scfnew);
               if scfnew >= scfworst
                  ran1=rand(1,nopt);
                  scxnew=bl+range.*ran1;
                  [scfnew]=funUsfmRKG(scxnew,rain,obsq,Iu,Ev,Ou,qRmax,Q0);
                  Neval=Neval+length(scfnew);
               end
```

```
166                 end
167                 scx(nscp,:)=scxnew(1,:);
168                 scf(nscp)=scfnew;
169               end % ialph
170     %--- Get out of Extermination Room ---%
171               cx(k3,:)=scx(1:nscp,:);
172               cf(k3)=scf(1:nscp);
173               cfcx=[cf,cx];
174               cfcx=sortrows(cfcx,1);
175               cf=cfcx(:,1);
176               cx=cfcx(:,2:nopt+1);
177           end % ibeta
178           x(k2,:)=cx(k1,:);
179           xf(k2)=cf(k1);
180         end % inc
181         xfx=[xf,x];
182         xfx=sortrows(xfx,1);
183         xf=xfx(:,1);
184         x=xfx(:,2:nopt+1);
185         xbest(igene,:)=x(1,:);
186         xfbest(igene)=xf(1);
187         xworst(igene,:)=x(ntp,:);
188         xfworst(igene)=xf(ntp);
189         xsort=sort(x);
190         xrange(igene,:)=xsort(ntp,:)-xsort(1,:);
191   fprintf('xf=%6.7f   x=%6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f',xf(1),x(1,:));
192
193   %%
194   OptResult(igene,1)=xf(1,1);
195   OptResult(igene,2:8)=x(1,1:7);
196
197   end % igene
198   toc
199   filename1=num2str(irun,'%03d');
200   filename2=num2str(Neval,'%06d');
201   filename=strcat('Result_SCEUA_No',filename1,'_Neval',filename2,'.xlsx');
202   xlswrite(filename,OptResult);
203   end % irun
```

「funUsfmRKG.m」

```
1   % Urban Storage Function Model Solved by Runge-Kutta-Gill Method
2   % Parameter Known -- 2*2 Matrix System   2014.12.20
3   function rmse=funUsfmRKG(parameters, r, obsq, Iu, Ev, Ou, qRmax, Q0)
4   %global k1 k2 k3 p1 p2 z alpha
```

```
n=length(r);
dt = 0.1;
nstep = n/dt;
int = 1/dt;

k1 = parameters(1);
k2 = parameters(2);
k3 = parameters(3);
p1 = parameters(4);
p2 = parameters(5);
z  = parameters(6);
alpha = parameters(7);

x1 = Q0^p2;
x2 = 0;
X = [x1 ; x2];
qsum = zeros(nstep+1,1);
qsum(1)=x1^(1/p2);
qR=zeros(1,nstep+1);
q=zeros(1,nstep+1);

for k=1:nstep
    jiten = floor(((k-1) + int)/int);
    rieo=r(jiten)+Iu(jiten)-Ev(jiten)-Ou(jiten);

    u1=dt*Usf(X,rieo,k1,k2,k3,p1,p2,z);
    X1=X+u1./2;
    u2=dt*Usf(X1,rieo,k1,k2,k3,p1,p2,z);
    X2=X+((sqrt(2)-1)/2).*u1+((2-sqrt(2))/2).*u2;
    u3=dt*Usf(X2,rieo,k1,k2,k3,p1,p2,z);
    X3=X-(sqrt(2)/2).*u2+((2+sqrt(2))/2).*u3;
    u4=dt*Usf(X3,rieo,k1,k2,k3,p1,p2,z);
    X=X+(u1+(2-sqrt(2)).*u2+(2+sqrt(2)).*u3+u4)./6;

    qsum(k+1)=X(1)^(1/p2);

end

for p=1:nstep
        if alpha*(qsum(p+1)-Q0)<qRmax
            qR(p+1)=alpha*(qsum(p+1)-Q0);
        else
            qR(p+1)=qRmax;
        end
```

```
        if qR(p+1)<0
            qR(p+1)=0;
        end

        if qR(p+1)>qsum(p+1)
            qR(p+1)=qsum(p+1);
        end
        q(p+1)=qsum(p+1)-qR(p+1);
end

jj=int+1:int:nstep+1;
rmse=sqrt(mean((obsq-q(jj)').^2));
```

「Usf.m」

```
%% Urban Storage Function

function F = Usf(X,rieo,k1,k2,k3,p1,p2,z)

%global k1 k2 k3 p1 p2 z alpha

if X(1) >= 0
    if k1*X(1)^(p1/p2)+X(2)*k2 >= z %s>=z
           F =[X(2) ; -k1/k2*p1/p2*X(2)*X(1)^(p1/p2-1) - 1/k2*X(1)^(1/p2)...
            - k1*k3/k2*X(1)^(p1/p2) - k3*X(2) + 1/k2*(rieo+ k3*z)];
    else
           F =[X(2) ; -k1/k2*p1/p2*X(2)*X(1)^(p1/p2-1) - 1/k2*X(1)^(1/p2)...
            + 1/k2*(rieo)];
    end
else
           F =[X(2) ; 99999];
end
```

A-8　粒子群最適化の MATLAB コード

PSO の MATLAB コードを示す. 適用するデータは仮想流域で, 真値が明らかなものである．MATLAB のカレントディレクトリをメインプログラムの存在するフォルダに設定し，「PSObySandeepSolanki_USF.m」を読み込むことで動作する．途中，都市貯留関数モデルを Runge-Kutta-Gill 法で解くサブルーチンである「funUsfmRKG.m」を参照しており，その中で都市貯留関数モデルの計算式である「Usf.m」を自動的に読み込むという構造になっている.

プログラムの作成にあたっては，Sandeep Solanki 氏の psoToolbox を参考にした.

「PSObySandeepSolanki_USF.m」

```
%%% Particle Swarm Optimization
% Refferring to "psoToolbox" made by Sandeep Solanki
clear
tic
%% Number of Particles and Iterations
% np=input('Number of Particles = ');
np=50;
% ni=input('Number of Iterations = ');
ni=3000;
nopt=7;

%% parameter for PSO
w=0.0004;
c1=1.2;
c2=0.012;
OptResult=zeros(ni,nopt+1);

%% Data for Urban Storage Function Model %%
load designed_raunfall_20year_3hr_long.txt;
rain=designed_raunfall_20year_3hr_long(:,1);
obsq=designed_raunfall_20year_3hr_long(:,2);
Ev=designed_raunfall_20year_3hr_long(:,3);
Iu=designed_raunfall_20year_3hr_long(:,4);
Ou=designed_raunfall_20year_3hr_long(:,5);
qRmax=0.005; % maximum sewage discharge. assumed
Q0=obsq(1);
ndata=length(designed_raunfall_20year_3hr_long);

%% fix random
% rng('default');
% rng(0);

%% Range of Variables for Rastrigin function
```

```
bl=[10  100  0.001 0.1 0.1 1  0.1]; % Lower Boundary
bu=[500 5000 0.05  1   1   50 1];  % Upper Boundary

%% ----------- MAIN ----------- %%
nrun=100;
for irun=1:nrun
    Neval=0;
    fprintf ('\n PSO Run Number --> %2g\n',irun);

    %% initial loop
    t=1;
    fprintf ('\n IterationNo-->%3g',t);
    CurrentPosition=zeros(np,nopt);
    for i=1:nopt
       CurrentPosition(:,i)=random('unif',bl(i),bu(i),np,1);
    end
    Velocity=w.*rand(np,nopt); % initioal Velocity
    CurrentFitness=zeros(np,1);
    for i=1:np
        CurrentFitness(i)=funUsfmRKG(CurrentPosition(i,:),rain,obsq,Iu,Ev,Ou,qRmax,Q0);
    end
    Neval=Neval+np;
    % update Local Best
    LocalBestPosition=CurrentPosition;
    LocalBestFitness=CurrentFitness;
    % update Global Best
    [GlobalBestFitness,index]=min(LocalBestFitness);
    GlobalBestPosition=repmat(LocalBestPosition(index,:),np,1);

    % update Velocity and Position
    r1=rand(np,nopt);
    r2=rand(np,nopt);

    Velocity=w.*Velocity + c1.*r1.*(LocalBestPosition-CurrentPosition)...
        + c2.*r2.*(GlobalBestPosition-CurrentPosition);
    CurrentPosition=CurrentPosition+Velocity;
    % Bound Đata bl bu
    for i=1:nopt
        indexes=CurrentPosition(:,i)<bl(i).*ones(np,1);
        CurrentPosition(indexes,i)=bl(i);
        indexes=CurrentPosition(:,i)>bu(i).*ones(np,1);
        CurrentPosition(indexes,i)=bu(i);
    end
    fprintf('         xf=%6.7f       x=%6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f
```

```
',GlobalBestFitness,GlobalBestPosition(1,:));
    OptResult(1,:)=[GlobalBestFitness GlobalBestPosition(1,:)];

    %% iterate

    for t=2:ni
        fprintf ('\n IterationNo-->%3g',t);
        % evaluate current position
        for i=1:np

CurrentFitness(i)=funUsfmRKG(CurrentPosition(i,:),rain,obsq,Iu,Ev,Ou,qRmax,Q0);
        end
        Neval=Neval+np;
        % update local best
        indexes=find(CurrentFitness<LocalBestFitness);
        LocalBestFitness(indexes)=CurrentFitness(indexes);
        LocalBestPosition(indexes,:)=CurrentPosition(indexes,:);
        % update Global Best
        [GlobalBestFitnessNew,index]=min(LocalBestFitness);
        if GlobalBestFitnessNew<GlobalBestFitness
            GlobalBestFitness=GlobalBestFitnessNew;
            GlobalBestPosition=repmat(LocalBestPosition(index,:),np,1);
        end
        % update Velocity and Position
        r1=randn(np,nopt);
        r2=randn(np,nopt);
        Velocity=w.*Velocity + c1.*r1.*(LocalBestPosition-CurrentPosition)...
            + c2.*r2.*(GlobalBestPosition-CurrentPosition);
        CurrentPosition=CurrentPosition+Velocity;
        % Bound Ðata bl bu
        for i=1:nopt
            indexes=CurrentPosition(:,i)<bl(i).*ones(np,1);
            CurrentPosition(indexes,i)=bl(i);
            indexes=CurrentPosition(:,i)>bu(i).*ones(np,1);
            CurrentPosition(indexes,i)=bu(i);
        end
        fprintf('       xf=%6.7f      x=%6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f  %6.7f
',GlobalBestFitness,GlobalBestPosition(1,:));
        OptResult(t,:)=[GlobalBestFitness GlobalBestPosition(1,:)];

    end % t
    toc
    filename=num2str(irun,'%03d');
    filename=strcat('Result_PSO_No',filename,'_Neval15000.xlsx');
```

```
122         xlswrite(filename,OptResult);
123     end % irun
```

「funUsfmRKG.m」

```
1   % Urban Storage Function Model Solved by Runge-Kutta-Gill Method
2   % Parameter Known -- 2*2 Matrix System   2014.12.20
3   function rmse=funUsfmRKG(parameters, r, obsq, Iu, Ev, Ou, qRmax, Q0)
4   %global k1 k2 k3 p1 p2 z alpha
5
6
7   n=length(r);
8   dt = 0.1;
9   nstep = n/dt;
10  int = 1/dt;
11
12  k1 = parameters(1);
13  k2 = parameters(2);
14  k3 = parameters(3);
15  p1 = parameters(4);
16  p2 = parameters(5);
17  z  = parameters(6);
18  alpha = parameters(7);
19
20  x1 = Q0^p2;
21  x2 = 0;
22  X = [x1 ; x2];
23  qsum = zeros(nstep+1,1);
24  qsum(1)=x1^(1/p2);
25  qR=zeros(1,nstep+1);
26  q=zeros(1,nstep+1);
27
28  for k=1:nstep
29      jiten = floor(((k-1) + int)/int);
30      rieo=r(jiten)+Iu(jiten)-Ev(jiten)-Ou(jiten);
31
32      u1=dt*Usf(X,rieo,k1,k2,k3,p1,p2,z);
33      X1=X+u1./2;
34      u2=dt*Usf(X1,rieo,k1,k2,k3,p1,p2,z);
35      X2=X+((sqrt(2)-1)/2).*u1+((2-sqrt(2))/2).*u2;
36      u3=dt*Usf(X2,rieo,k1,k2,k3,p1,p2,z);
37      X3=X-(sqrt(2)/2).*u2+((2+sqrt(2))/2).*u3;
38      u4=dt*Usf(X3,rieo,k1,k2,k3,p1,p2,z);
39      X=X+(u1+(2-sqrt(2)).*u2+(2+sqrt(2)).*u3+u4)./6;
40
```

```
41      qsum(k+1)=X(1)^(1/p2);
42
43  end
44
45  for p=1:nstep
46          if alpha*(qsum(p+1)-Q0)<qRmax
47              qR(p+1)=alpha*(qsum(p+1)-Q0);
48          else
49              qR(p+1)=qRmax;
50          end
51          if qR(p+1)<0
52              qR(p+1)=0;
53          end
54
55          if qR(p+1)>qsum(p+1)
56              qR(p+1)=qsum(p+1);
57          end
58          q(p+1)=qsum(p+1)-qR(p+1);
59  end
60
61  jj=int+1:int:nstep+1;
62  rmse=sqrt(mean((obsq-q(jj)').^2));
```

「Usf.m」

```
1   %% Urban Storage Function
2
3   function F = Usf(X,rieo,k1,k2,k3,p1,p2,z)
4
5   %global k1 k2 k3 p1 p2 z alpha
6
7   if X(1) >= 0
8       if k1*X(1)^(p1/p2)+X(2)*k2 >= z %s>=z
9              F =[X(2) ; -k1/k2*p1/p2*X(2)*X(1)^(p1/p2-1) - 1/k2*X(1)^(1/p2)...
10              - k1*k3/k2*X(1)^(p1/p2) - k3*X(2) + 1/k2*(rieo+ k3*z)];
11      else
12             F =[X(2) ; -k1/k2*p1/p2*X(2)*X(1)^(p1/p2-1) - 1/k2*X(1)^(1/p2)...
13              + 1/k2*(rieo)];
14      end
15  else
16             F =[X(2) ; 99999];
17  end
```

A-13　カッコウ探索の MATLAB コード

Cuckoo Search のプログラムは，開発者 Yang らの配布する「Cuckoo Search (CS) Algorithm」を使用した．メインプログラムは「cuckoo_search_new_USF.m」であるが，全てサブルーチンにより作成されているため，これを参照する「CSnewforUSF.m」を読み込むことで動作するようにした．

「CSnewforUSF.m」

```
for i=1:100
['CS',num2str(i),'running']
%% Đata for Urban Storage Function Model %%
clearvars -except i
tic
%global rain obsq Ev Iu Ou qRmax Q0 ndata
load designed_raunfall_20year_3hr_long.txt;
rain=designed_raunfall_20year_3hr_long(:,1);
obsq=designed_raunfall_20year_3hr_long(:,2);
Ev=designed_raunfall_20year_3hr_long(:,3);
Iu=designed_raunfall_20year_3hr_long(:,4);
Ou=designed_raunfall_20year_3hr_long(:,5);
qRmax=0.005; % maximum sewage discharge. assumed
Q0=obsq(1);
ndata=length(designed_raunfall_20year_3hr_long);

[bestnest,fmin,OptResult]=cuckoo_search_new_USF;
filename=num2str(i,'%03d');
filename=strcat('Result_CS_No',filename,'_Neval15000.xlsx');
xlswrite(filename,OptResult);

toc
end

「cuckoo_search_new_USF.m」
% ---------------------------------------------------------------
% Cuckoo Search (CS) algorithm by Xin-She Yang and Suash Đeb      %
% Programmed by Xin-She Yang at Cambridge University              %
% Programming dates: Nov 2008 to June 2009                        %
% Last revised: Đec  2009   (simplified version for demo only)    %
% ---------------------------------------------------------------
% Papers -- Citation Đetails:
% 1) X.-S. Yang, S. Đeb, Cuckoo search via Levy flights,
% in: Proc. of World Congress on Nature & Biologically Inspired
% Computing (NaBIC 2009), Đecember 2009, India,
% IEEE Publications, USA,  pp. 210-214 (2009).
```

```
% http://arxiv.org/PS_cache/arxiv/pdf/1003/1003.1594v1.pdf
% 2) X.-S. Yang, S. Đeb, Engineering optimization by cuckoo search,
% Int. J. Mathematical Modelling and Numerical Optimisation,
% Vol. 1, No. 4, 330-343 (2010).
% http://arxiv.org/PS_cache/arxiv/pdf/1005/1005.2908v2.pdf
% ----------------------------------------------------------------%
% This demo program only implements a standard version of         %
% Cuckoo Search (CS), as the Levy flights and generation of        %
% new solutions may use slightly different methods.               %
% The pseudo code was given sequentially (select a cuckoo etc),   %
% but the implementation here uses Matlab's vector capability,    %
% which results in neater/better codes and shorter running time.  %
% This implementation is different and more efficient than the    %
% the demo code provided in the book by                           %
%    "Yang X. S., Nature-Inspired Metaheuristic Algoirthms,       %
%     2nd Edition, Luniver Press, (2010).                "       %
% ---------------------------------------------------------------- %

% ================================================================ %
% Notes:                                                          %
% Đifferent implementations may lead to slightly different        %
% behavour and/or results, but there is nothing wrong with it,    %
% as this is the nature of random walks and all metaheuristics.   %
% ----------------------------------------------------------------

% Additional Note: This version uses a fixed number of generation %
% (not a given tolerance) because many readers asked me to add    %
%  or implement this option.                                Thanks.%
function [bestnest,fmin,OptResult]=cuckoo_search_new_USF(n)
if nargin<1,
% Number of nests (or different solutions)
n=30;
end

% Điscovery rate of alien eggs/solutions
pa=0.25;

%% Change this if you want to get better results
N_IterTotal=2500;
%% Simple bounds of the search domain
% Lower bounds
nd=7;
Lb=[10  100  0.001 0.1 0.1 1  0.1];
% Upper bounds
```

```
Ub=[500 5000 0.05  1   1   50 1];
% for recording Bestnet by each iteration
OptResult=zeros(N_IterTotal,nd+1);
% Random initial solutions
for i=1:n,
nest(i,:)=Lb+(Ub-Lb).*rand(size(Lb));
end

% Get the current best
fitness=10^10*ones(n,1);
[fmin,bestnest,nest,fitness]=get_best_nest(nest,nest,fitness);

N_iter=0;
%% Starting iterations
for iter=1:N_IterTotal,
    % Generate new solutions (but keep the current best)
     new_nest=get_cuckoos(nest,bestnest,Lb,Ub);
     [fnew,best,nest,fitness]=get_best_nest(nest,new_nest,fitness);
    % Update the counter
      N_iter=N_iter+n;
    % Ðiscovery and randomization
      new_nest=empty_nests(nest,Lb,Ub,pa) ;

    % Evaluate this set of solutions
      [fnew,best,nest,fitness]=get_best_nest(nest,new_nest,fitness);
    % Update the counter again
      N_iter=N_iter+n;
    % Find the best objective so far
    if fnew<fmin,
        fmin=fnew;
        bestnest=best;
    end
    OptResult(iter,:)=[fmin bestnest];
end %% End of iterations

%% Post-optimization processing
%% Ðisplay all the nests
disp(strcat('Total number of iterations=',num2str(N_iter)));
fmin
bestnest

%% --------------- All subfunctions are list below ------------------
%% Get cuckoos by ramdom walk
function nest=get_cuckoos(nest,best,Lb,Ub)
```

```
% Levy flights
n=size(nest,1);
% Levy exponent and coefficient
% For details, see equation (2.21), Page 16 (chapter 2) of the book
% X. S. Yang, Nature-Inspired Metaheuristic Algorithms, 2nd Edition, Luniver Press,
(2010).
beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)))^(1/beta);

for j=1:n,
    s=nest(j,:);
    % This is a simple way of implementing Levy flights
    % For standard random walks, use step=1;
    %% Levy flights by Mantegna's algorithm
    u=randn(size(s))*sigma;
    v=randn(size(s));
    step=u./abs(v).^(1/beta);

    % In the next equation, the difference factor (s-best) means that
    % when the solution is the best solution, it remains unchanged.
    stepsize=0.01*step.*(s-best);
    % Here the factor 0.01 comes from the fact that L/100 should the typical
    % step size of walks/flights where L is the typical lenghtscale;
    % otherwise, Levy flights may become too aggresive/efficient,
    % which makes new solutions (even) jump out side of the design domain
    % (and thus wasting evaluations).
    % Now the actual random walks or flights
    s=s+stepsize.*randn(size(s));
   % Apply simple bounds/limits
   nest(j,:)=simplebounds(s,Lb,Ub);
end

%% Find the current best nest
function [fmin,best,nest,fitness]=get_best_nest(nest,newnest,fitness)
% Evaluating all new solutions
for j=1:size(nest,1),
    fnew=fobj(newnest(j,:));
    if fnew<=fitness(j),
       fitness(j)=fnew;
       nest(j,:)=newnest(j,:);
    end
end
% Find the current best
[fmin,K]=min(fitness) ;
```

```
best=nest(K,:);

%% Replace some nests by constructing new solutions/nests
function new_nest=empty_nests(nest,Lb,Ub,pa)
% A fraction of worse nests are discovered with a probability pa
n=size(nest,1);
% Discovered or not -- a status vector
K=rand(size(nest))>pa;

% In the real world, if a cuckoo's egg is very similar to a host's eggs, then
% this cuckoo's egg is less likely to be discovered, thus the fitness should
% be related to the difference in solutions.  Therefore, it is a good idea
% to do a random walk in a biased way with some random step sizes.
%% New solution by biased/selective random walks
stepsize=rand*(nest(randperm(n),:)-nest(randperm(n),:));
new_nest=nest+stepsize.*K;
for j=1:size(new_nest,1)
    s=new_nest(j,:);
  new_nest(j,:)=simplebounds(s,Lb,Ub);
end

% Application of simple constraints
function s=simplebounds(s,Lb,Ub)
  % Apply the lower bound
  ns_tmp=s;
  I=ns_tmp<Lb;
  ns_tmp(I)=Lb(I);

  % Apply the upper bounds
  J=ns_tmp>Ub;
  ns_tmp(J)=Ub(J);
  % Update this new move
  s=ns_tmp;

%% You can replace the following by your own functions
% A d-dimensional objective function
function rmse=fobj(var)
global rain obsq Ev Iu Ou qRmax Q0 ndata
%% Urban Storage Function model 7 variables
%  with a minimum at [50 500 0.005 0.4 0.3 5 0.5]
dt = 0.1;
nstep = ndata/dt;
int = 1/dt;
k1 = var(1);
```

```
k2 = var(2);
k3 = var(3);
p1 = var(4);
p2 = var(5);
z  = var(6);
alpha = var(7);
x1 = Q0^p2;
x2 = 0;
X = [x1 ; x2];
qsum = zeros(nstep+1,1);
qsum(1)=x1^(1/p2);
qR=zeros(1,nstep+1);
q=zeros(1,nstep+1);
for k=1:nstep
    jiten = floor(((k-1) + int)/int);
    rieo=rain(jiten)+Iu(jiten)-Ev(jiten)-Ou(jiten);

    u1=dt*Usf(X,rieo,k1,k2,k3,p1,p2,z);
    X1=X+u1./2;
    u2=dt*Usf(X1,rieo,k1,k2,k3,p1,p2,z);
    X2=X+((sqrt(2)-1)/2).*u1+((2-sqrt(2))/2).*u2;
    u3=dt*Usf(X2,rieo,k1,k2,k3,p1,p2,z);
    X3=X-(sqrt(2)/2).*u2+((2+sqrt(2))/2).*u3;
    u4=dt*Usf(X3,rieo,k1,k2,k3,p1,p2,z);
    X=X+(u1+(2-sqrt(2)).*u2+(2+sqrt(2)).*u3+u4)./6;

    qsum(k+1)=X(1)^(1/p2);
end
for p=1:nstep
        if alpha*(qsum(p+1)-Q0)<qRmax
            qR(p+1)=alpha*(qsum(p+1)-Q0);
        else
            qR(p+1)=qRmax;
        end
        if qR(p+1)<0
            qR(p+1)=0;
        end
        if qR(p+1)>qsum(p+1)
            qR(p+1)=qsum(p+1);
        end
        q(p+1)=qsum(p+1)-qR(p+1);
end
jj=int+1:int:nstep+1;
rmse=sqrt(mean((obsq-q(jj)').^2));
```

```
257
258   %% RKG
259   function F = Usf(X,rieo,k1,k2,k3,p1,p2,z)
260
261   if X(1) >= 0
262       if k1*X(1)^(p1/p2)+X(2)*k2 >= z %s>=z
263              F =[X(2) ; -k1/k2*p1/p2*X(2)*X(1)^(p1/p2-1) - 1/k2*X(1)^(1/p2)...
264               - k1*k3/k2*X(1)^(p1/p2) - k3*X(2) + 1/k2*(rieo+ k3*z)];
265       else
266              F =[X(2) ; -k1/k2*p1/p2*X(2)*X(1)^(p1/p2-1) - 1/k2*X(1)^(1/p2)...
267               + 1/k2*(rieo)];
268       end
269   else
270              F =[X(2) ; 99999];
271   end
```