博士論文

二酸化炭素地中貯留を対象とした大域的最適化の実用性向上

田中　啓

目次

表目次

図目次

1. 序論

1.1. CO_2 地中貯留

気候変動を抑制する上で,温室効果ガスの削減は喫緊の課題である.二酸化炭素(以下,CO_2)は温室効果ガスの主要なソースであり，CO_2 の削減に向けた技術開発，法整備が OECD 諸国を中心に進んでいる.

IEA は， CO_2 削減対策を今後講じないシナリオ（6DS: 6 Degree Scenario）と 2050 年時点での温度上昇を現在の 2℃以内に抑えるシナリオ（2 Degree Scenario: 2DS）を設定し，2DS を達成するため必要となる幾つかの技術とその貢献度を提言している（IEA, 2015, Figure 1-1）．中でも，CO_2 の回収・貯留（CCS: CO_2 Capture and Storage）が CO_2 削減量全体に寄与する割合は 13%と高い．この値は年度によって異なるものの，ここ 10 年ほどは 10 から 20%の間で推移しており，今後も CCS の重要性は高いと予想される.

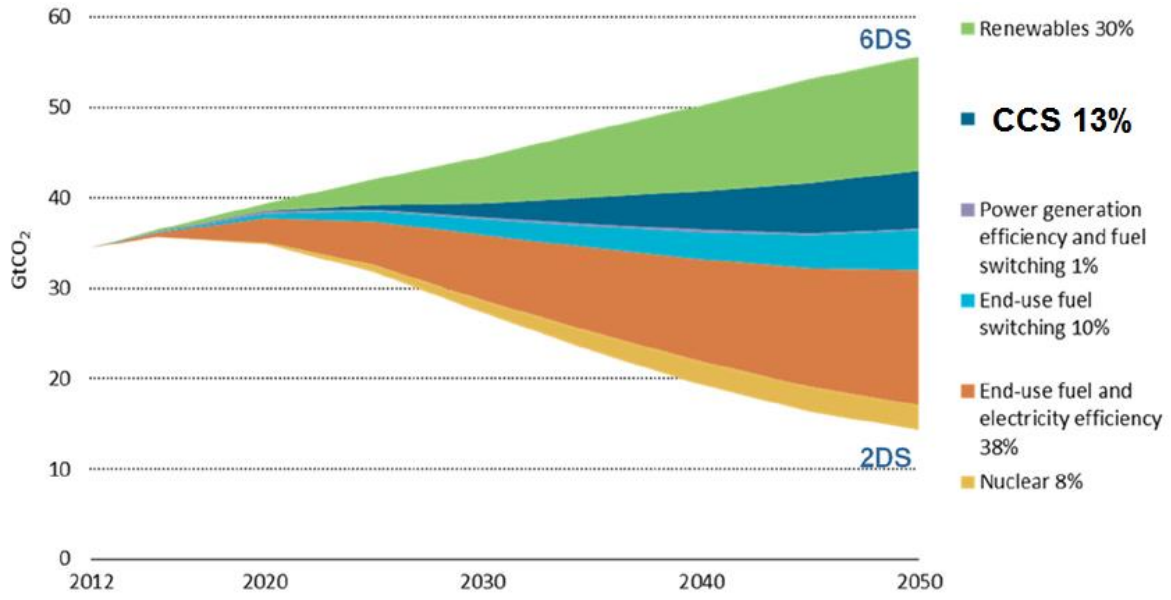

Figure 1-1 Contribution of technology area to global cumulative CO_2 reductions between 6DS and 2DS (IEA, 2015)

CCS の中でも CO_2 地中貯留（CO_2 geological storage）は，石油開発の技術を応用した実効性の高い手段の 1 つである．CO_2 地中貯留は，回収，輸送，貯留，モニタリングの段階に分けられる．火力発電所，製鉄所，セメント工場といった CO_2 の大規模排出源から，アミン回収法や膜分離といった方法で CO_2 を回収，パイプライン等を用いて圧入サイトまで輸送される．貯留層としては，廃油・ガス田，地下塩水層，石炭層が想定されており，浸透性の低い帽岩が存在すること，CO_2 が超臨界状態となる 800m 以深であること，断層が存在しないことが貯留に不可欠な条件である．圧入された CO_2 は圧入後数十年間モニタリングされ，安定的に貯留されて

いることが確認される（IEA, 2008; 住&島田, 2009）.

現在，世界には 55 の大規模プロジェクトが存在し（Figure 1-2），うち 22 プロジェクトが操業中か建設中である．しかしながら，前述の IEA の 2DS 達成には，今後 100 以上の大規模プロジェクトが必要とされており，CO_2 地中貯留の導入はまだ十分ではない．CCS を実施する上で障壁の 1 つとなるのがそのコストであり，一部で CO_2-EOR と組み合わせた商業プロジェクトが稼働しているものの，CO_2 地中貯留の 1 トン当たりの限界削減費用は，温暖化対策の選択肢の中でも非常に高い（McKinsey & Company, 2009; Global CCS Institute, 2011）.

一方で，CCS を除外して 2DS を達成しようとすると，CCS 以上のコストが必要になり（IEA, 2012），対策が遅れるほど，CO_2 削減に係る追加費用は増加する．そのため，技術開発，実証試験に向けた資金提供，法整備に向けた CO_2 地中貯留データベースの作成が各国で進んでいる．特に近年は，以下に挙げるような特徴的なプロジェクトが操業を開始している．

- カナダでの石炭発電所からの CO_2 回収と油田及び帯水層への貯留による世界初の大規模商業プロジェクト
- アブダビにおける世界初の製鉄プラントで発生した CO_2 の回収・貯留
- サウジアラビア初の CO_2 回収と EOR
- ブラジル初の CO_2 回収と EOR

また法整備の面でも，COP21 を意図した石油メジャー6 社による炭素価格の導入を求めた共同文書が UNFCCC において採択されるなど，進展が著しい．今後は，非 OECD 諸国における CO_2 地中貯留プロジェクトや，製鉄，セメントプラントからの CO_2 回収・貯留が進むものと期待されている．

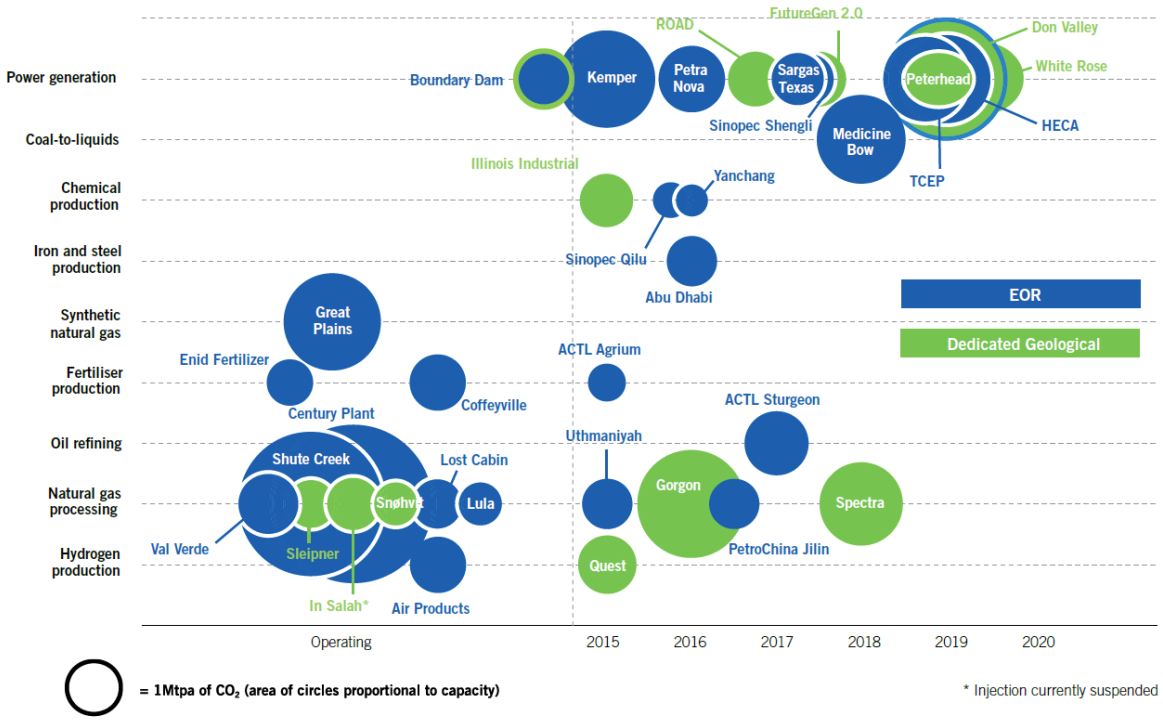

Figure 1-2 Actual and expected operation dates for large-scale CCS projects in the Operate, Execute and Defied stages bye industry and storage type (Global CCS Institute, 2014)

1.2. 貯留層シミュレーションを伴う最適化問題

地中貯留 CO_2 の挙動予測には，貯留層シミュレータを用いる．貯留層シミュレータは本来，石油・ガス開発のために利用されるもので，用途に合わせ商業用からオープンソースまで数多くのものが存在する．CO_2 地中貯留の数値シミュレーションでは，地層水や CO_2 の状態変化を表現することが必要とされるため，数万程度の比較的簡単なグリッドモデルでも 1 回の計算に数時間を要する．

また，前述のとおり CO_2 地中貯留は，石油・ガス開発と異なり膨大なコストをかけることが難しいため，利用可能な地質データは限定的である．従って，入力データの不確実性は高く，実際のプロジェクトにおいては，入力データを変化させ複数回の貯留層シミュレーションを行うことが想定される．

一般に CO_2 地中貯留や油ガス田開発では，計画の効率化や貯留層データの持つ不確実性低減を目的として，貯留層シミュレーションを伴う最適化計算が必要となる．CO_2 圧入・油ガス生産開始前は坑井配置の最適化，開始後であればヒストリーマッチング（以下，HM）が行われる（Figure 1-3）．前者では．物理探査等によって構築された貯留層モデルを最適化問題の入力データとし，CO_2 の圧入量や油ガス生産量を出力する．後者は，貯留層の圧力履歴や生産履歴から貯留層モデルの浸透率や孔隙率といったパラメータを逆推定するものである．最適化計算は過去には，浸透率や孔隙率等のパラメータを個別に変化させる”manual”な最適化が一般的であったものの，近年の計算機の性能向上や並列化計算の普及に伴い，複数のパラメータを含む目的関数を評価することで，最適化問題を”automatic”に計算する大域的最適化アルゴリズムの利用が進んでいる（Oliver & Chen, 2011）．HM では最適化するパラメータ，つまり問題の次元が坑井配置の最適化よりも大きくなるため，多くの計算資源と計算コストを必要とする．特に CO_2 地中貯留プロジェクトでは不確実性の高いデータが多いため，最適化問題の次元も増加する．

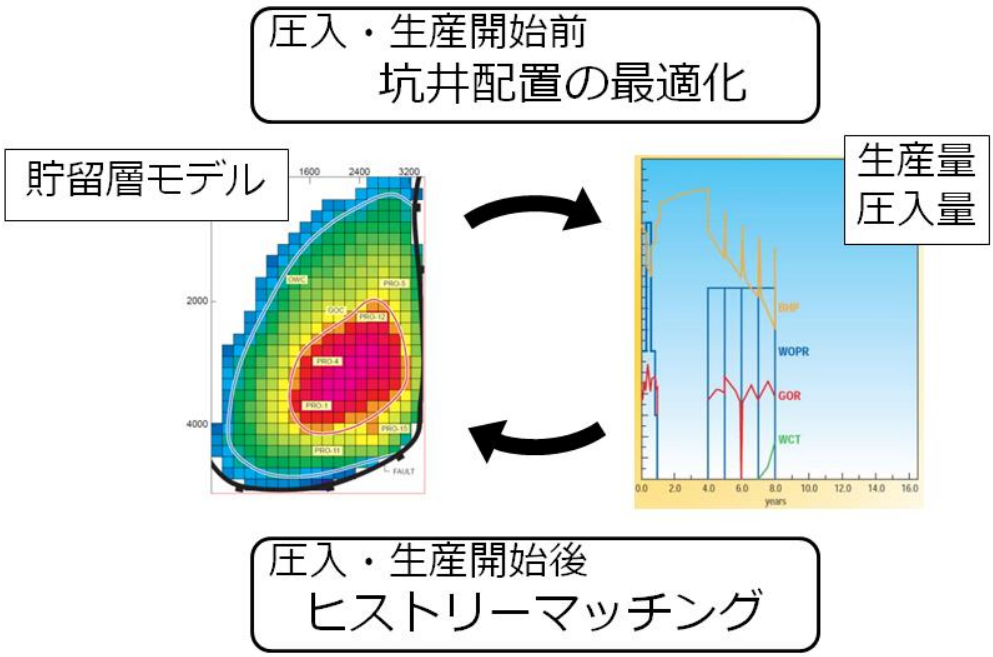

Figure 1-3 Concept of optimization problems with reservoir simulations

(Figure of geological data and production data are cited from Floris *et al.*, 2001)

1.3. 本論文の目的と位置付け

現在，最適化問題を計算するツールとして，大域的最適化アルゴリズムが提案されており，貯留層シミュレーションを伴う最適化計算に適用されている．しかしながら，貯留層シミュレーションはその計算に長時間を要し，利用される最適化アルゴリズムが貯留層シミュレーションに適しているのか，十分な解探索が行われたのかといった事項は十分に評価されていない．また，貯留層シミュレーションに要求される計算資源が節約できれば，利用できる最適化アルゴリズムの選択肢は拡大すると考えられる．加えて，最適化アルゴリズムと貯留層シミュレーションを繋ぐ目的関数の設定は非常に重要であるものの，貯留層シミュレーションを伴う最適化，特に CO_2 地中貯留においてはあまり議論されていない．

本研究では，貯留層シミュレーションに適した，実用性の高い大域的最適化手法の開発を目的とする．具体的には，

1) 貯留層シミュレーションに適した最適化アルゴリズムの開発
2) シミュレータの高速化
3) 目的関数の適切な設定

を課題とする．

1.4. 本論文の構成

本論文は全 7 章と付録によって構成される．以下にその概要を示す．

1. 序論

CO_2 地中貯留の現状と貯留層シミュレーションを伴う最適化問題について説明し，本研究の目的と構成を述べる．

2. 大域的最適化手法の開発

貯留層シミュレーションに適した大域的最適化手法について概説した後，課題 1)の解決のため，本研究で開発した大域的最適化手法について述べ，数値計算例を示す．

3. シミュレータの高速化

本研究で用いた数値シミュレータ TOUGH2/ECO2N 及び最適化アルゴリズムの高速化について述べ，課題 2)について検討する．

4. 坑井配置の最適化問題への適用

開発した手法及び数値シミュレータを CO_2 地中貯留に関わる坑井配置の最適化問題に適用し，手法の有効性を検証する．

5. ヒストリーマッチングへの適用

開発した手法及び数値シミュレータを CO_2 地中貯留に関わるヒストリーマッチングに適用し，手法の有効性を検証するとともに，3)について検討する．

6. 結論

本研究の成果をまとめる．

付録

本研究で用いた最適化アルゴリズムを記載する．

2. 大域的最適化手法の開発

本章では，本研究で用いた大域的最適化アルゴリズムについて述べる．初めに，貯留層シミュレーションを用いる大域的最適化には多点探索法が適していることを示す．次に，本研究で用いた大域的最適化アルゴリズムについて解説する．最後に，正規化エントロピーによる収束判定法，LHSD によるロバスト性の向上という 2 つの改良点について述べ，それぞれについて数値計算例をもとに開発した手法の有効性を示す．

2.1. 貯留層シミュレーションと大域的最適化

2.1.1. 貯留層シミュレーションを伴う大域的最適化

坑井配置の最適化問題を例にとれば，坑井配置を示すグリッドナンバーは離散変数，流体の生産レート及び圧入レートは連続変数である（Figure 2-1）．従って，坑井配置の最適化問題は混合変数の非線形計画問題（MINLP: Mixed Integer Nonlinear Problem）に帰着され，最適解の導出には多くの場合メタヒューリスティクスを用いた大域的最適化が行われる（久保&ペドロソ, 2009）．メタヒューリスティクスとは，最適化問題を解くための経験的手法を結合させたものである．全ての手法が必ずしも数学的な検証を経ているわけではないものの，少ない評価回数で大域的最適解，もしくは準最適解を求めることが可能であり，多くの工学分野において用いられている．

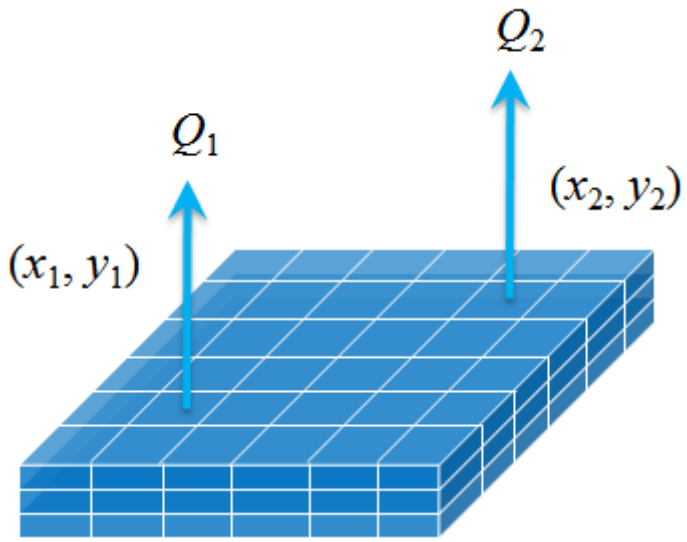

Figure 2-1 Concept of well optimization problem

大域的最適化とは，パラメータ空間 D 上の関数として目的関数 $f(x)$ $(x \in D)$が与えられたとき，f が最小もしくは最大となるときの x の値x^*を求めることである：

$$x^* = \arg\min_{x \in D} f(x). \tag{1}$$

解空間が d 次元であれば，x はベクトル$\boldsymbol{x} = (x_1,\ldots,x_d)$を意味する．坑井配置の最適化問題では，坑井位置や生産・圧入レートがベクトル x の各要素，貯留層シミュレーションによって得られる油・ガスの生産量や地中貯留 CO_2 量が目的関数 f にあたる．仮に最適化問題が線形計画問題といった既知の問題であれば，特定の解法を選択することができる．しかしながら，貯留層シミュレーションを伴う最適化は目的関数の事前情報が少ないブラックボックス最適化であり，メタヒューリスティクスを用いるのが効果的である．

メタヒューリスティクスは，自動車や橋梁の構造計算や製造工程のスケジューリング問題で多くの適用例が見られ，石油開発においてもその利用が進んでいる．しかしながら，計算機性能が向上しているとはいえ，貯留層シミュレーションは比較的長い時間を要することから，得られた最適解の評価が不十分なまま，現実的な評価回数で計算が打ち切られることが多い．また，仮に時間の制約がなく，十分な数値計算が可能であったとしても，大域的最適解の探索が十分に行われたかどうかを判断する指標は限られる．

2.1.2. 貯留層シミュレーションに適した最適化手法

貯留層工学に関わる最適化問題では，メタヒューリスティクスの中でも遺伝的アルゴリズム（GA: Genetic Algorithm）が利用される場合が多い（倉本, 2010）．しかしながら，GA はその適用範囲が広い反面，必ずしも解探索の性能が高いわけではなく，問題に応じたアルゴリズムの変更が必要とされる場合が多い．GA 以外では，焼きなまし法（SA: Simulated Annealing）や粒子群最適化（PSO: Particle Swarm Optimization）といった手法の適用例が報告されている（例えば Beckner *et al.*, 1995; Onwunalu *et al.*, 2011）．前者は Hill Climbing を基にした代表的な局所探索法の 1 つであり，大域的最適解が得られることが数学的に保証されているものの，問題の次元によっては実行時間が膨大になる．後者は一度に複数のサンプル点を生成し，各点の目的関数値を計算する多点探索法の 1 つであり，近年その適用例が増加している．他にも多点探索法では，差分進化法（DE: Deferential Evolution）や繰り返しラテン超方格法（ILHS: Iterative Latin Hypercube Sampling）の適用例が挙げられる（Nwankwor *et al.*, 2013; Goda & Sato, 2014）．

貯留層シミュレーションを伴う最適化問題の特徴として，第一に評価回数に制約があることが挙げられる．有限要素法による構造計算や工業製品製造のスケジューリング問題と比較したとき，数万以上のグリッドを対象とする貯留層シミュレーションは，目的関数の評価に比較的長い時間を要するため，少ない評価回数で大域的最適解を得られる多点探索法を用いるのが現実的である．

第二に，目的関数の非線形性に着目する．油・ガス田開発や CO_2 地中貯留における坑井配置の最適化問題では，油ガスの累積生産量や CO_2 の累積圧入量，トラップ量を最大化するような目的関数が設定される．これらの目的関数は坑井の位置（グリッドナンバー）や生産・圧入レートに対して非線形であり，複数の局所的最適解が存在する．一般に，目的関数の非線形性が

低ければ局所探索法が，高ければ多点探索法が適しているといわれており（Davidor, 1991），このことからも，貯留層工学における最適化問題では，局所探索法よりも多点探索法が適していると言える．

2.2. 多点探索法

代表的な多点探索法には，粒子群最適化（PSO），差分進化法（DE），繰り返しラテン超方格法（ILHS）の 3 つが挙げられる．メタヒューリスティクスの解探索性能を評価する場合，いくつかのベンチマーク関数に対する最適化が行われるが，上記手法は，それらに対し高い解探索能力を発揮することが分かっている．以下に 3 手法の概要を説明する．

2.2.1. 粒子群最適化

粒子群最適化（PSO）は Kennedy & Eberhart によって 1995 に提案された，魚や鳥のような群れを成す生物の習性をモデルとして開発された最適化アルゴリズムである．

PSO において，各粒子は位置と速度の情報を持つ．いま，d 次元の最適化問題を n 個の粒子によって探索することを考える． t 回目の探索において，j 番目の次元に対する i 番目の粒子の位置と速度をそれぞれx_{ij}^tとv_{ij}^tとすれば，t+1 回目の位置と速度は以下のようになる：

$$v_{ij}^{t+1} = w^t v_{ij}^t + c_1 R_1^t \left(p_{ij}^t - x_{ij}^t\right) + c_2 R_2^t \left(g_j^t - x_{ij}^t\right), \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3}$$

上式中のp_{ij}^tは t 回目までに発見した粒子 i の最良解，g_j^tは群全体の最良解を表す．R_1^tとR_2^tは区間[0,1]上の一様乱数を用いて決定される．c_1，c_2は cognitive, social と呼ばれるパラメータである．各粒子の最良探索点と群全体の最良探索点への探索に対する重み付けを表す．w^tは慣性重みと呼ばれ，その値が大きいほど粒子の位置は大きく変化する．解探索終了まで一定値をとる場合と，解探索の進行に伴い値を減少させる場合とがある．

PSO は多点探索法の中でも構造が単純であり，工学的問題一般に適用例が多い手法である．一方で, PSO には上で示したように, PSO にはc_1，c_2，w^tという主要な入力パラメータが存在し，これらパラメータの選び方によって解探索の効率は大きく変化する．そのため工学的問題の最適化には，多くの場合，事前の感度分析等によるパラメータの設定が必要である．

2.2.2. 差分進化法

差分進化法 (DE)は Storn & Price によって 1997 年に提案された，進化的アルゴリズムの 1 つである．DE にはいくつかの形式があり，DE/base/num/cross と表記される．base，num，cross はそれぞれ，基本ベクトルとなる親の選択方法，基本ベクトルを変化させるための差分ベクトルの個数，子個体を生成するための交叉方法をそれぞれ指定する（Feoktistov, 2006）．ここでは DE/1/rand/bin，つまり基底ベクトルの数が 1，親となる探索点をランダムに選択し，交叉方法は一様（Binomial）交叉の場合について説明する．

DE では探索点を突然変異と交叉によって新たな探索点を生成する．t 回目の探索において，i 番目の探索点の設計変数を$x_i^t = (x_{i1}^t, \ldots, x_{id}^t)$とし，設計変数の数は n とする．ランダムに 3 つの探索点x_{r1}^t，x_{r2}^t，x_{r3}^tを選び，以下の様に新たな探索点v_i^tを設定する（突然変異）．ただし

$d \neq r1 \neq r2 \neq r3$である．

$$v_i^t = x_{r1}^t + F(x_{r2}^t - x_{r3}^t). \tag{4}$$

F は突然変異確率という入力パラメータである．x_i^tとv_i^{t+1}から新たな探索点u_i^{t+1}を生成する（交叉）．つまり，

$$u_{ij}^t = \begin{cases} v_{ij}^t \ (R_{ij} \in [0,1) \leq Cr) \\ x_{ij}^t \ (R_{ij} \in [0,1) > Cr) \end{cases}, \tag{5}$$

とする．Crは入力パラメータの交叉確率，R_{ij}は設計変数の成分毎に設定する乱数である．以上の操作によって得られたu_i^tについて目的関数 f を評価し，x_{ij}^tとu_{ij}^tとで適合度の高いものを t 回目の最適解とする．この交叉と目的関数の再評価という構造が DE の大きな特徴である．1 回の探索で目的関数の評価回数が増加するため，少ない繰り返し回数では必ずしも最適解を得られないものの，十分な繰り返し回数で高い解探索性能を発揮する．

DE と PSO は以下のような類似構造を持つと考えられる（北山ら. 2010）.

- 基底ベクトルを有する点
- 探索方向ベクトルを有する点
- 確率的ステップ幅を有する点
- 探索集団として降下特性を有する点

これらの類似構造は DE において全て突然変異にあたる．逆に，交叉と目的関数の再評価という構造は PSO には見られない．また，DE は PSO と同様，F と Cr の 2 つの入力パラメータを持つ．ただ PSO と異なりそれらは[0.1]の値をとるため，パラメータチューニングが PSO と比較して容易である．

2.2.3. 繰り返しラテン超方格法

繰り返しラテン超方格法（ILHS: Iterative Latin Hypercube Sampling）は，計算機支援実験計画法の 1 つであるラテン超方格法（LHS: Latin Hypercube Sampling）（Mckay *et al.*, 1979）に基づくメタヒューリスティクスである．

ILHS のアルゴリズムの概要を以下に示す．目的関数の次元を d，サンプル点の個数を n とする．

1. d 個の主変数に対する初期（t=1）の累積分布関数の設定
2. LHS による各主変数に対する$n-1$個の分割点と n 個のサンプル点の設定
3. n 個のサンプルセットに対する目的関数の評価
4. 目的関数の値に応じたサンプル区間の重み付け
5. 次ステップ（t+1 回目）での各主変数の累積分布関数の導出
6. 2.から 5.を設定した基準が満たされるまで繰り返す．

まず，LHS について説明する．いま，d 個の主変数を持つ目的関数 $f(\boldsymbol{x})$の最適化を考える．

サンプル数は n 個とし，各パラメータ$x_j(j=1,\cdots,d)$は互いに独立した累積分布関数$F_j(x_j)$に従っているものとする．LHS では，各パラメータの累積確率 $[0, 1]$ を n 等分し，各区間から 1 回ずつランダムにサンプリングする．得られた n 個のサンプルはランダムな順序に並べ替えられ，d 個の入力パラメータを組み合わせることで主変数の組（入力パラメータセット）を構成する．変数x_jに対する i 番目のサンプルをx_{ij}とすれば，

$$x_{ij} = F_j^{-1}\left(\frac{\pi_{ij} - U_{ij}}{n}\right) = \frac{\pi_{ij} - 1}{n} + \frac{U_{ij}}{n}, \tag{6}$$

と表される．ここで，U_{ij}は$U(0, 1)$の一様乱数であり，$\{\pi_{1j}, \cdots, \pi_{nj}\}$は$\{1, \cdots, n\}$のランダムな並び替えである．$d=1$, $n=5$としたときのランダムサンプリングと LHS によるサンプリングの比較を Figure 2-2 に示す．

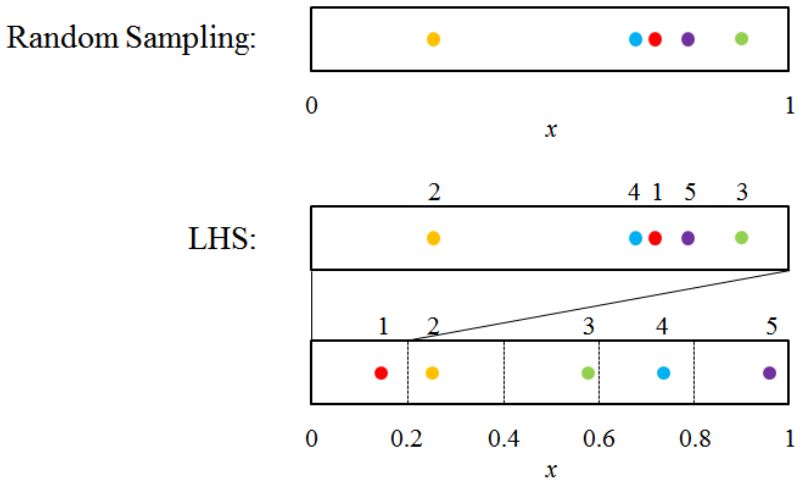

Figure 2-2 Concept of random sampling and LHS

LHS で発生させた主変数の組それぞれに対して目的関数を計算し，各主変数の累積分布関数を更新することを 1 つのステップとする．ILHS では，このステップを複数回繰り返すことで最適解の探索を行う．Figure 2-3 に $d = 2, n = 5$ とした場合の ILHS の概念図を示す．t 回目のステップにおいて，2 つの主変数x_1とx_2の累積分布関数から分割点$F_j^{-1}(i/n)$ $(i = 0, \cdots, n)$, 並びに主変数の組$\boldsymbol{x}_i = \{x_{i1},\cdots, x_{id}\}$ $(i = 0, \cdots, n)$ を得る（Figure 2-3 左上）．$\boldsymbol{x}_i$ は$\Pi_{j=1}^{d}\left[x_{ij}^{-}, x_{ij}^{+}\right]$ $(i = 1, \cdots, n)$ で囲まれる超方格の内部に存在する．ここで，区間の下限x_{ij}^{-}と上限x_{ij}^{+}は，

$$x_{ij}^{-} = F_j^{-1}\left(\frac{\pi_{ij} - 1}{n}\right), x_{ij}^{+} = F_j^{-1}\left(\frac{\pi_{ij}}{n}\right), \tag{7}$$

で表される．次に，各パラメータセットに対して目的関数を計算する．目的関数を f_j とすれば，

$$f_j = f(\boldsymbol{x}_i). \tag{8}$$

である．このとき最小化問題であれば，小さな目的関数の値を持つ区間ほど，最適解が存在する可能性が高いと考えられる．そこで，目的関数の値の大小に応じて各主変数の組が存在する領域$\Pi_{j=1}^{d}\left[x_{ij}^{-}, x_{ij}^{+}\right]$ $(i = 1, \cdots, n)$ の重み付けを行い，$(t + 1)$ 回目のステップでの累積分布関数を更新

する（Figure 2-3 右下）．いまx_jが$\left[x_{ij}^-, x_{ij}^+\right]$の区間に位置しているとき，累積分布関数$F_j$は次のように与えられる：

$$F_j(x_j) = \sum_{i=1}^{k-1} w_{(i)j} + w_{(k)j} \frac{x_j - x_{(k)j}^-}{x_{(k)j}^+ - x_{(k)j}^-}. \tag{9}$$

$F_j(x_j)$を$\alpha \in [0,1]$と置き，αに対応する k を m とすればx_jは以下の式のように表せる：

$$x_j = x_{(m)j}^- + \left(\alpha - \sum_{i=1}^{m-1} w_{(i)j}\right) \frac{x_{(m)j}^+ - x_{(\boldsymbol{m})j}^-}{w_{(m)j}}. \tag{10}$$

$\alpha = (\pi_{ij} - U_{ij})/n$を代入することで$x_j$に対する i 番目の成分x_{ij}を得る：

$$x_{ij} = x_{(m)j}^- + \left(\frac{\pi_{ij} - U_{ij}}{n} - \sum_{i=1}^{m-1} w_{(i)j}\right) \frac{x_{(m)j}^+ - x_{(\boldsymbol{m})j}^-}{w_{(m)j}}. \tag{11}$$

w_iの設定には Zipf の法則を用いる：

$$w_i = \frac{1}{Z} r_i^{-\gamma}. \tag{12}$$

r_iは n 個の目的関数の値の中における f_i の順位を表している．γは(0, 1)の任意の値をとる重み付け指数で，γの値が 1 に近いほど最適化計算の収束は速くなる．Z は正規化定数であり，

$$Z = \sum_{i=1}^{n} r_i^{-\gamma}, \tag{13}$$

によって計算される．式(9)により，目的関数の順位 r_i の小さい領域ほど大きい重みを与えられることになる．またw_iは目的関数の大小関係にのみ依存し，そのスケールには依存しない．そのため ILHS では，各サンプル点に対応する目的関数 f_i 間の差が小さい場合でも，最適解への収束性が担保される．

ILHS を DE，PSO と比較することでその特徴について考察する．PSO，DE，ILHS の多点探索法のサンプル点はそれぞれ，式(3)，式(4)，式(11)により計算される．ILHS が他の 2 手法と最も異なるのは，式(11)が繰り返し回数 t に対する漸化式でない点である．ILHS では，(t-1)回目の探索点から直接，t 回目の探索点が生成されるわけではなく，過去の探索点の情報は累積分布関数，つまり式(11)中の重みw_iに反映される．この目的関数の値の評価，累積分布関数の更新という一連の処理が ILHS と他の多点探索法とを区別する特徴である．前述の PSO と DE に共通する 4 つの類似構造は式(3)，(4)を基に導かれたものであるため，ILHS には厳密には当てはまらない．加えて，累積分布関数の更新にはサンプル領域の順位r_iと入力パラメータγのみで決定されることから，入力パラメータが少ないのも ILHS の特徴といえる．

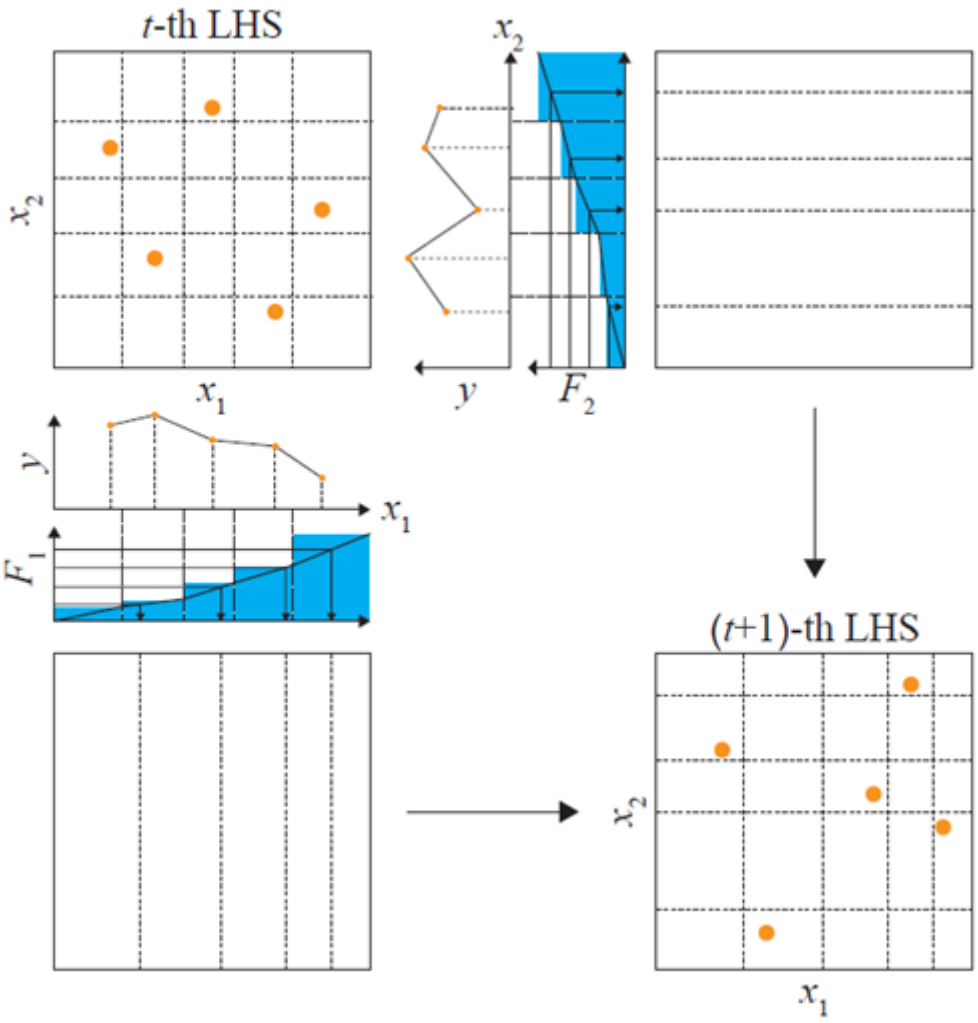

Figure 2-3 Concept of ILHS (Goda, 2014)

2.2.4. 多点探索法に求められる解探索性能

メタヒューリスティクスの定義の 1 つとして，「多くのパラメータを導入することによる自由度によって制御可能なアルゴリズム」というものが挙げられる（久保&ペドロソ, 2009）．入力パラメータが多いメタヒューリスティクスほど，自由度は高くなり，アルゴリズムの改良に伴って，入力パラメータは増加する傾向にある．その反面，実問題への適用には，事前に感度分析などを行い，パラメータチューニングを行う必要がある．このことは，ブラックボックス最適化への適用が可能であるというメタヒューリスティクスの利点とは必ずしも合致しない．その点，ILHS 独自の入力パラメータは式(9)中のγのみであり，必ずしも自由度は高くないものの，多くの最適化問題に対して容易に適用が可能である．

また，ILHS は，PSO や DE と比較して，少ない評価回数でより良い最適解を求められることが示されており（Goda & Sato, 2014），CO_2 地中貯留や石油開発における坑井配置の最適化においてもその有効性が確認されている．以上から，本研究では以降，メタヒューリスティクスとして ILHS を用いることとする．

メタヒューリスティクスの解探索の過程は，解探索の集中化（Intensification）と多様化（Diversification）という過程に分けて捉えることができる．前者は，それまでの解探索によって得られた局所的最適解の近傍を集中的に探索することを指し，後者は局所的最適解近傍よりもより広い領域を大域的に探索することを指す．これら 2 つのプロセスはトレード・オフの関係にあり，メタヒューリスティクスの解探索戦略において，両者のバランスをとることが重要であるとされる．過度の多様化は最適解への収束を阻害する一方で，過度の集中化は局所解への収束を引き起こしやすい. ILHS においては, LHS に基づくランダムサンプリングが多様化，累積分布関数の更新によるサンプル領域の設定が集中化に相当する．

多点探索法では，解探索の多様化プロセスを何らかの乱数を用いて表現することが多い．従って，1 回の独立試行内で繰り返し計算の回数を増加させたとしても，大域的最適解が得られるという保証はなく，解の探索には初期条件の異なる複数の独立試行が必要である．このことは評価回数の増大を意味するため，計算時間の制約がある場合には十分な解探索が行われないまま解探索が終了する可能性がある．また，複数の独立試行間で得られた最適解は多くの場合一致せずばらつきをもつ．

2.3. 新規収束判定基準の導入

前節の内容を踏まえ，貯留層シミュレーションを伴う最適化問題を念頭に，多点探索法で利用可能な新規収束判定基準を提案する．

2.3.1. 情報の定量的尺度としてのエントロピー

本節では，次章で述べる収束判定基準を設定するのに必要な，エントロピーと 2 値エントロピー関数（BEF: Binary Entropy Function）について概説する．

まず，ある事象 A の起こりうる確率を p とし，事象 A の自己情報量 $I(A)$について次式で定義する：

$$I(A) = -\log_2 p. \tag{14}$$

自己情報量の特徴として，独立な事象 A と B が同時に起きた場合の自己情報量 $I(AB)$は，それぞれの自己情報量の和となる（情報量の加法性）：

$$I(AB) = I(A) + I(B). \tag{15}$$

次に，有限集合 U 上の値を取る離散確率変数$X = (x_1, x_2, \cdots, x_n)$ が，確率分布$P = (p_1, p_2, \cdots, p_n)$ に従う場合，自己情報量の期待値は以下の様に与えられる：

$$H(X) = -\sum_{x \in U} P(X) \log P(X) = -\sum_{i=1}^{n} p_i \log_2 p_i . \tag{16}$$

$H(X)$を平均情報量，もしくはエントロピーと呼ぶ．ただし$0 \log_2 0 = 0$とする．$H(X)$は全ての i について p_iが等しいとき最大となる．また，$n = 2$のときのエントロピーを特に 2 値エントロピー関数と呼ぶ：

$$H_b(p) = -p \log_2 p - (1-p) \log_2 (1-p). \tag{17}$$

次章では，上記のエントロピーと BEF を用いて，多点探索法の収束判定法を提案する．

2.3.2. エントロピーを用いた収束判定

最適化手法による解探索の収束は，一般に以下の 3 つの基準により判定される．

I. 繰り返し計算の回数が指定した回数に達したとき

II. 繰り返し計算において，目的関数が改善しなくなったとき

III. 数学的な最適性の条件を満足したとき

基準 I は，解探索の収束だけでなく，計算環境にも依存して設定される基準であることから，収束判定の基準としては適当ではない．加えて，解の探索が十分に行われ，指定した回数以前

に大域的最適解が得られた場合には，それ以降の計算は計算効率の低下を招く．基準 II については，大域的最適解は未知であるため，目的関数の値がそれ以降更新されるかどうかを判断することは難しい．結果として，基準 III のように数学的な最適性の条件を設定することが望ましものの，統一的な基準が存在するわけではない．

多点探索法において，基準 III が設定されることは少ないものの，PSO に限ればサンプル点の変動係数を用いた基準が提案されている（槇野ら，2013）．多点探索法では初め，大域的探索が行われるが，解探索の進展に伴い局所的探索が支配的となる．特に PSO では解探索の収束に伴ってサンプル点の運動量が低下するため，徐々に最適解近傍でのみ探索が行われることになる．槇野らの基準は，目的関数の値からではなく，サンプル点の分布から収束を判定する点で多点探索法の解探索の特徴を反映しており，基準 II の様な，目的関数の値から収束を判断する基準よりも合理的であると言える．

ILHS では，最適化計算が進むに従って，Eq.(4)に基づき順位 r_{i} の低い区間の距離は減少し，逆に高い区間で増加する．ただし各区間から LHS に基づくサンプリングを行うため，PSO や DE といった他の多点探索法のように，全てのサンプル点が最適解近傍に集中するということはない．従って，前述の基準のように，サンプル点間の変動係数の変化から最適化計算の収束を判断することは難しい．

多点探索法の 1 つ，蟻コロニー最適化手法（ACO: Ant Colony Optimization）では，解探索にエントロピーを用いている．ACO は，巡回セールスマン問題等に有効な最適化手法であり，フェロモン濃度という探索結果の評価値を更新することで過去の探索情報を蓄積していく．その際，フェロモン濃度の分布状況を評価する指標として正規化エントロピーが用いられる．

この正規化エントロピーの概念を ILHS についても設定する．ILHS ではサンプル区間の距離が過去の目的関数の値を反映していることから，区間距離が ACO におけるフェロモンに相当する役割を担っていると考えられる．ここで，主変数の累積分布関数の正規化エントロピーを以下の式で定義する：

$$\eta(n,j) = -\sum_{i=1}^{n} \frac{l_{ij}\log_2 l_{ij}}{\log_2 n}. \tag{18}$$

l_{ij} は区間 $\left[x_{\mathrm{ij}}^{-}, x_{\mathrm{ij}}^{+}\right]$ の距離である．上式はエントロピーを $\log_2 n$ で除することで値の正規化を行っている．

解探索初期は l_{ij} の値が各区間で等しいため，$\eta(n,j)$ は最大値 1 をとる．解の探索が進むに従い，正規化エントロピーの値は減少し一定値に収束すると考えられる．ここで，

$$\lim_{x\to 0} x\log x = 0, \tag{19}$$

より，解探索が十分に進み，主変数 x_{j} が $x_{\mathrm{j}} = x_{\mathrm{j,opt}}$ に収束したと仮定すれば，エントロピーは前述の BEF で表すことが出来る：

$$H_b(x_{j,\mathrm{opt}}) = -x_{j,\mathrm{opt}}\log_2 x_{j,\mathrm{opt}} - (1-x_{j,\mathrm{opt}})\log_2(1-x_{j,\mathrm{opt}}). \tag{20}$$

これを正規化すれば,

$$\eta_b(n,j) = \frac{H_b(x_{j,\mathrm{opt}})}{\log_2 n}, \tag{21}$$

となる．Figure 2-4 に$x_{\mathrm{j,opt}}$を変化させたときのη_{b}の値を示す．図中の n_{pop} はサンプルサイズを意味し，サンプルサイズが大きくなるほど，η_{b}の値は小さくなることが分かる．$n=2$のときη_{b}は BEF に等しく，$x_{\mathrm{opt}} = (1-x_{\mathrm{opt}})$，すなわち$x_{\mathrm{opt}} = 0.5$のとき，BEF は最大値 1 をとる．

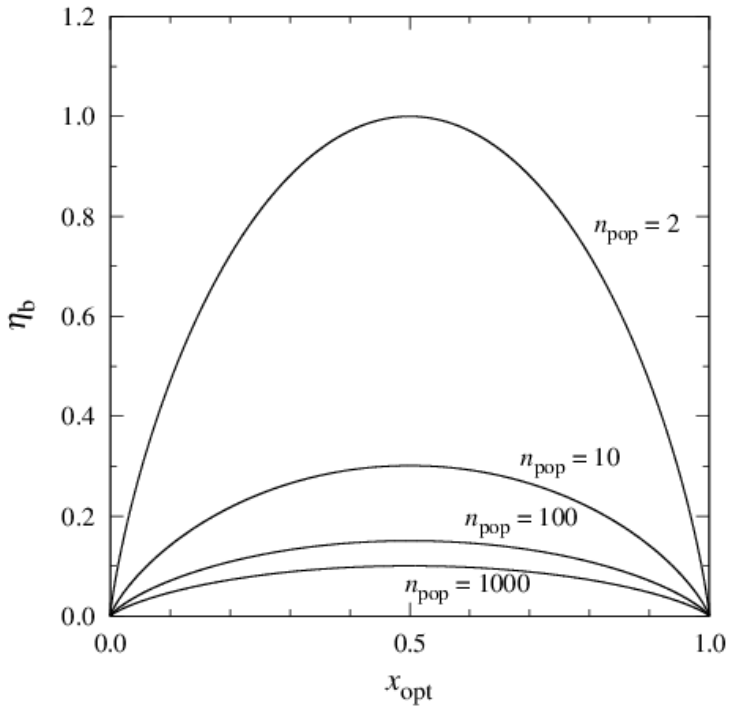

Figure 2-4 Normalized Binary Entropy Function

ここで，正規化エントロピーと正規化された BEF との差$d\eta(n,j)$を定義する：

$$d\eta(n,j) = \eta(n,j) - \eta_b(n,j). \tag{22}$$

解探索が進むに従って$\eta(n,j)$は$\eta_{\mathrm{b}}(n,j)$に近づく．つまり，$d\eta(n,j)$は 0 に収束する．本研究では，全ての主変数 x_{j} について$d\eta(n,j)$を計算し，$d\eta(n,j)$がある閾値$d\eta_{\min}(n,j)$に到達したとき，解探索が収束したとみなして，最適化計算を打ち切ることとする．本基準は ILHS での利用を念頭に置

いたものであるものの，サンプル区間の設定がないような他の多点探索法でも，サンプル点間の中点を分割点とすることで適用が可能である．

2.3.3. ベンチマーク関数への適用

複数のベンチマーク関数に対して最適化計算を行い，正規化エントロピーが最適化計算の収束基準として適切であることを示すとともに，閾値$d\eta_{\min}$について考察する．ベンチマーク関数には，多点探索法の性能評価に広く用いられている 19 の関数を利用した．PSO，DE，ACO といった主要な多点探索法への適用事例については，de Oca *et al.*, 2011; Wang *et al.*, 2011; Liao *et al.*, 2011 といったものが挙げられる．

ベンチマーク関数の概要を Table 2-1, Table 2-2,Table 2-3 に示す．関数 F1 から F6，F7 から F11 はそれぞれ Tang らと Herra & Lozano によって提案された関数であり，F12 から F19 は F1 から F11 内の単峰性関数と多峰性関数の複合関数である(Herrera *et al.*, 2010)．各関数は主変数ベクトル $\boldsymbol{x}$ について固有の定義域を持ち，ベクトル $\boldsymbol{z}$ は，$\boldsymbol{x}$ をベクトル $\boldsymbol{o}$ によっての平行移動（$\boldsymbol{z}=\boldsymbol{x}-\boldsymbol{o}$）したものを意味する．F3 についてのみ$\boldsymbol{z}=\boldsymbol{x}-\boldsymbol{o}+\boldsymbol{1}$であることに注意する．この平行移動により，最適解はベクトル $\boldsymbol{o}$ と等しくなるため，ベクトル $\boldsymbol{o}$ を設定することで最適解の値を自由に変更することができる．

Table 2-1 Description of basic benchmark functions: F1-F6 (Tang *et al.*, 2007)

Function	Name	Definition	Range
F1	Sphere	$\sum_{j=1}^{d} z_j^2$	$[-100, 100]^d$
F2	Schwefel 2.21	$\max\left\{\left\|z_j\right\|, 1 \le j \le d\right\}$	$[-100, 100]^d$
F3	Rosenbrock	$\sum_{j=1}^{d-1}\left(100\left(z_j^2 - z_{j+1}\right)^2 + \left(z_j - 1\right)^2\right)$	$[-100, 100]^d$
F4	Rastrigin	$\sum_{j=1}^{d}\left(z_j^2 - 10\cos\left(2\pi z_j\right) + 10\right)$	$[-5, 5]^d$
F5	Griewank	$\sum_{j=1}^{d}\frac{z_j^2}{4000} - \prod_{j=1}^{d}\cos\left(\frac{z_j}{\sqrt{j}}\right) + 1$	$[-600, 600]^d$
F6	Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} z_j^2}\right) - \exp\left(\frac{1}{d}\sum_{j=1}^{d}\cos\left(2\pi z_j\right)\right) + 20 + e$	$[-32, 32]^d$

Table 2-2 Description of benchmark functions: F7-F11 (Herrera & Lozano., 2009)

Function	Name	Definition	Range
F7	Schwefel 2.22	$\sum_{j=1}^{d}\lvert z_j\rvert+\prod_{j=1}^{d}\lvert z_j\rvert$	$[-10, 10]^d$
F8	Schwefel 1.2	$\sum_{i=1}^{d}\left(\sum_{j=1}^{i} z_j\right)^2$	$[-65.536, 65.536]^d$
F9	Extended f_{10}	$\left(\sum_{j=1}^{d-1} f_{10}(z_j, z_{j+1})\right)+f_{10}(z_d, z_1)$ where $f_{10}(x,y)=\left(x^2+y^2\right)^{0.25}\left(\sin^2\left(50\left(x^2+y^2\right)^{0.1}\right)+1\right)$	$[-100, 100]^d$
F10	Bohachevsky	$\sum_{j=1}^{d-1}\left(z_j^2+z_{j+1}^2-0.3\cos\left(3\pi z_j\right)-0.4\cos\left(4\pi z_{j+1}\right)+0.7\right)$	$[-15, 15]^d$
F11	Schaffer	$\sum_{j=1}^{d}\left(z_j^2+z_{j+1}^2\right)^{0.25}\left(\sin^2\left(50\left(z_j^2+z_{j+1}^2\right)^{0.1}\right)+1\right)$	$[-100, 100]^d$

Table 2-3 Description of benchmark functions: F12-F19 (Herrera *et al.*, 2010)

Function	Definition	Ratio	Range
F12	F9⊕F1	0.25: 0.75	$[-100, 100]^d$
F13	F9⊕F3	0.25: 0.75	$[-100, 100]^d$
F14	F9⊕F4	0.25: 0.75	$[-5, 5]^d$
F15	F10⊕F7	0.25: 0.75	$[-10, 10]^d$
F16	F9⊕F1	0.50: 0.50	$[-100, 100]^d$
F17	F9⊕F3	0.75: 0.25	$[-100, 100]^d$
F18	F9⊕F4	0.75: 0.25	$[-5, 5]^d$
F19	F10⊕F7	0.75: 0.25	$[-10, 10]^d$

上で述べたベンチマーク関数に対して最適化計算を行った．問題の次元 d，繰り返し回数

$\mathrm{Iter}_{\max}$はそれぞれ 10，300 とした．事前に行った感度分析により，関数評価回数に制限がある場合，サンプル数は最適化問題の次元の 1 から 2 倍に設定すると良いことが分かったため，サンプル数 n_{pop} を 15 とした. 従って, 関数評価回数 Fitness Evaluations (FES)は$n_{\mathrm{pop}} \times \mathrm{Iter}_{\max} = 4500$回である．また，同様の感度分析から ILHS における重み付けのパラメータ$\gamma = 0.95$とした．25 回の互いに独立な最適化計算を行い，各試行で得られた目的関数の最小値に対して，平均値μ, 分散σ，変動係数CV(= μ/σ) を算出した.

最適化計算の結果を Table 2-4 に示す.

Table 2-4 Average, standard deviation and CV of objective function values obtained on the 19 benchmark functions of 10 dimensions with 25 independent runs

Function	Avg.	Std.	CV
F1	7.00E-01	6.65E-01	9.50E-01
F2	4.11E+00	1.23E+00	3.01E-01
F3	4.36E+03	4.93E+03	1.13E+00
F4	5.93E+00	2.83E+00	4.78E-01
F5	7.21E-01	1.73E-01	2.40E-01
F6	1.55E+00	7.85E-01	5.08E-01
F7	1.16E-01	5.53E-02	4.75E-01
F8	4.42E+02	2.38E+02	5.39E-01
F9	9.56E+00	4.57E+00	4.78E-01
F10	1.08E+00	7.19E-01	6.68E-01
F11	9.43E+00	4.18E+00	4.43E-01
F12	2.57E+00	1.22E+00	4.74E-01
F13	9.44E+02	1.46E+03	1.55E+00
F14	3.56E+00	1.84E+00	5.15E-01
F15	1.45E-01	1.09E-01	7.51E-01
F16	4.33E+00	1.36E+00	3.13E-01
F17	5.32E+02	1.50E+03	2.81E+00
F18	1.85E+00	6.77E-01	3.66E-01
F19	5.15E-01	4.48E-01	8.71E-01

幾つかのベンチマーク関数について，目的関数の値と$d\eta$及び主変数の値の変化について示す. ここでは例として, 単峰性関数の F2, 多峰性関数の F6, また複合関数の中で解探索が難しく, 最も CV の値が大きい F17 を選んだ.

- 単峰性関数 F2

Figure 2-5 は F2 の目的関数の値の変化である．最適化計算に伴い，各試行間の目的関数の値の差は大きくなる．ただ，F2 は単峰性関数であることから，局所解に陥る可能性が低く，繰り返し計算回数を増やすほど目的関数の値は更新されるものと考えられる.

次に，最適解の値が最も小さかった試行について$d\eta$と主変数の変化を，x_1 から x_5 については Figure 2-6 に, x_6 から x_{10} については Figure 2-7 に示す．$d\eta$についてはそれまでの最小値を同時に示す（各グラフの(b)）．グラフから繰り返し計算回数 Iter が 100 回を過ぎると最適解の値は

あまり変化しないことが分かる．同様に$d\eta$の値も増減を繰り返し 0 に収束する．一方で前述のように，F2 の目的関数の値は常に改善する（Figure 2-5）．従って，最適解の値も同様に少しずつではあるものの改善している．ただ，$d\eta$の値が 0 に近いことから，大域的探索よりも局所的探索が支配的なプロセスとなっており，最適解の値が大きく変化する可能性は低い．

Table 2-5 はIter = 100, 200, 300のときの$d\eta$の値である．求められる問題の精度にもよるものの，仮にIter = 100付近の最適解が十分であるとすれば，おおよそ $d\eta = 0.01$が閾値にあたる．

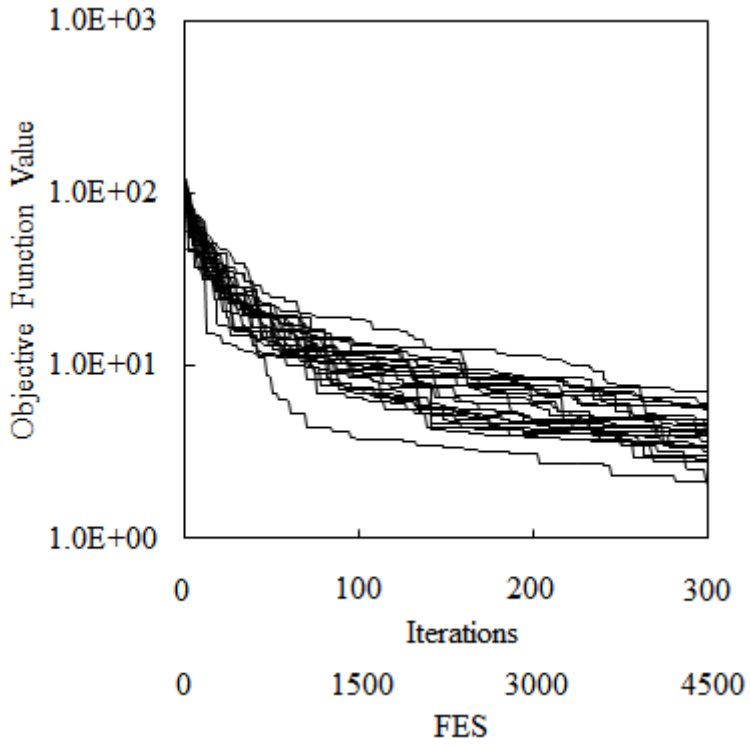

Figure 2-5 Convergence behaviors of objective function in F2

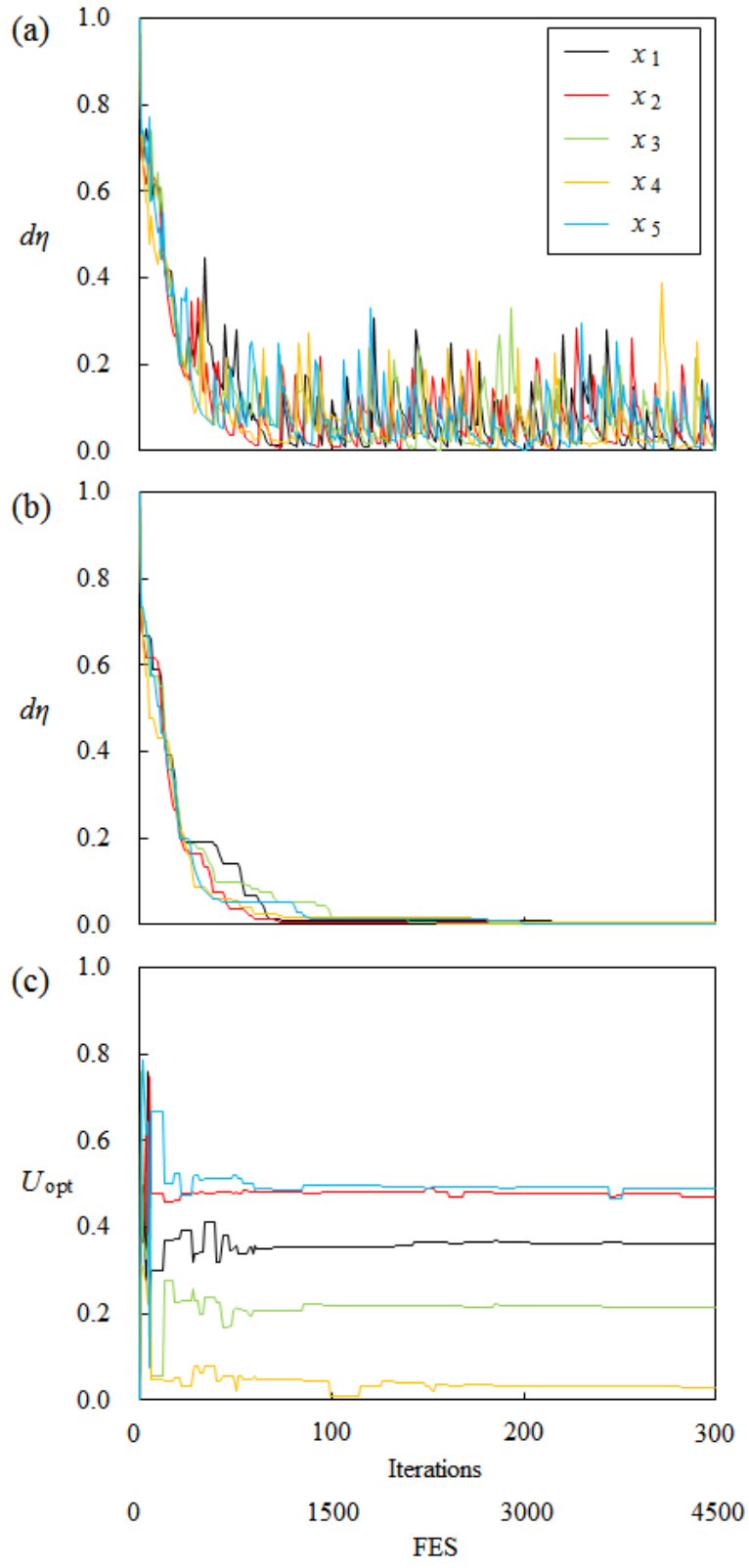

Figure 2-6 Convergence behavior of difference between normalized entropy and normalized BEF for 1-5 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in best case for benchmark function F2

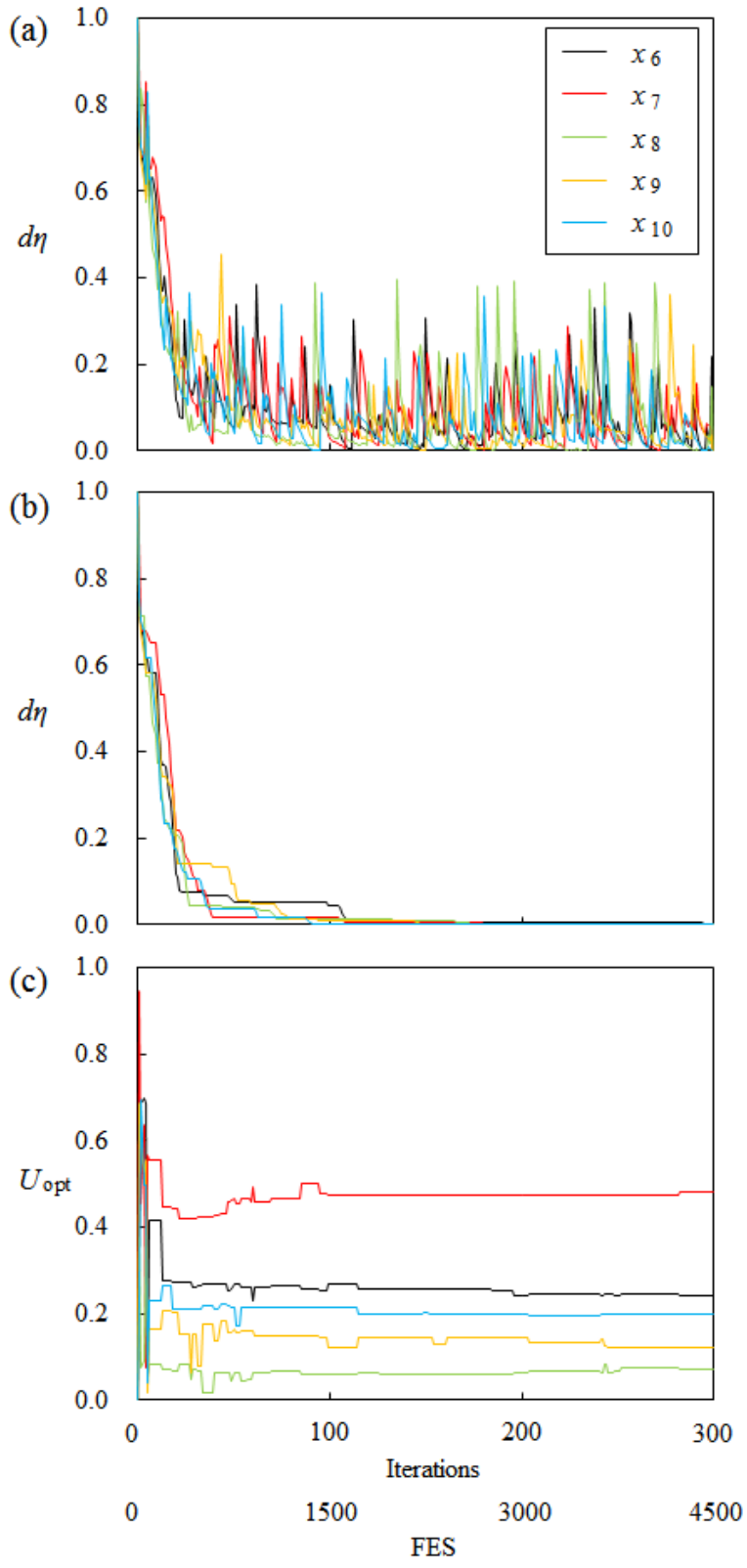

Figure 2-7 Convergence behavior of difference between normalized entropy and normalized BEF for 6-10 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in best case for benchmark function F2

Table 2-5 Minimum values of difference between normalized entropy and normalized BEF for

primary variables in best case for benchmark function F2

i	*Iter*		
	100	200	300
1	0.00834	0.00728	0.00085
2	0.00666	0.00457	0.00449
3	0.01769	0.00079	0.00079
4	0.01808	0.00434	0.00434
5	0.01243	0.00275	0.00247
6	0.04329	0.00397	0.00249
7	0.01698	0.00207	0.00207
8	0.01147	0.00170	0.00000
9	0.00938	0.00184	0.00184
10	0.00144	0.00144	0.00144

- 多峰性関数 F6

F2 と同様に，F6 の結果を以下に示す．F6 は多峰性関数であり，最適化計算では最適解が局所解に陥りやすい．このことは F6 の CV の値が，F2 のそれと比較して高いことからも説明できる．最適化計算終了時の最適解の値に関しても，F2 よりも F6 の方が広く分布していることが分かる（Figure 2-5，Figure 2-8）．

最適解と$d\eta$の値については，F6 も F2 と似た傾向を持っている．つまり，Iter = 100以降は最適解，$d\eta$ともに大きな変化は見られない（Figure 2-9， Figure 2-10）．F6 についても$d\eta$の値は 0.01 より小さくなれば十分であると考える（Table 2-6）．

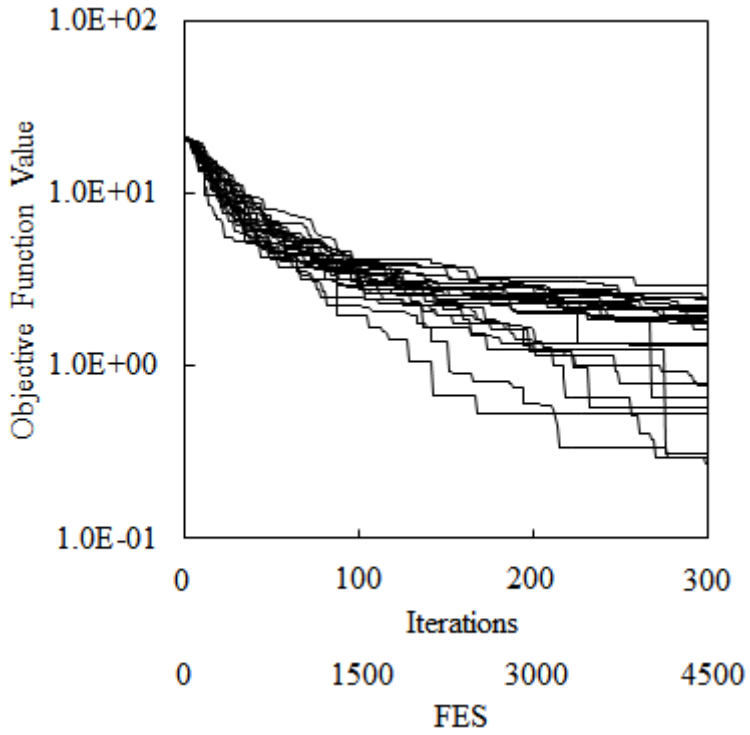

Figure 2-8 Convergence behaviors of objective function in F6

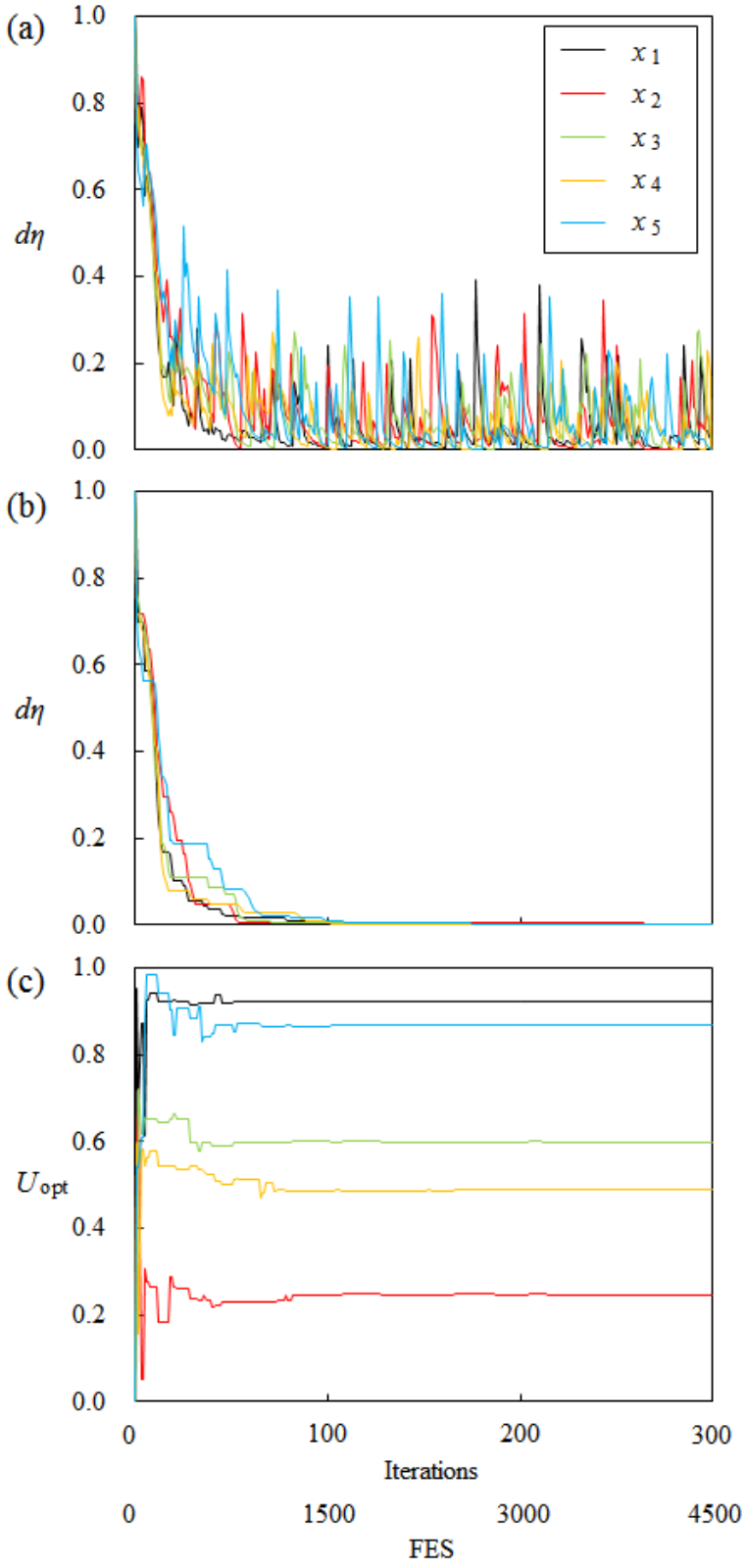

Figure 2-9 Convergence behavior of difference between normalized entropy and normalized BEF for 1-5 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in best case for benchmark function F6

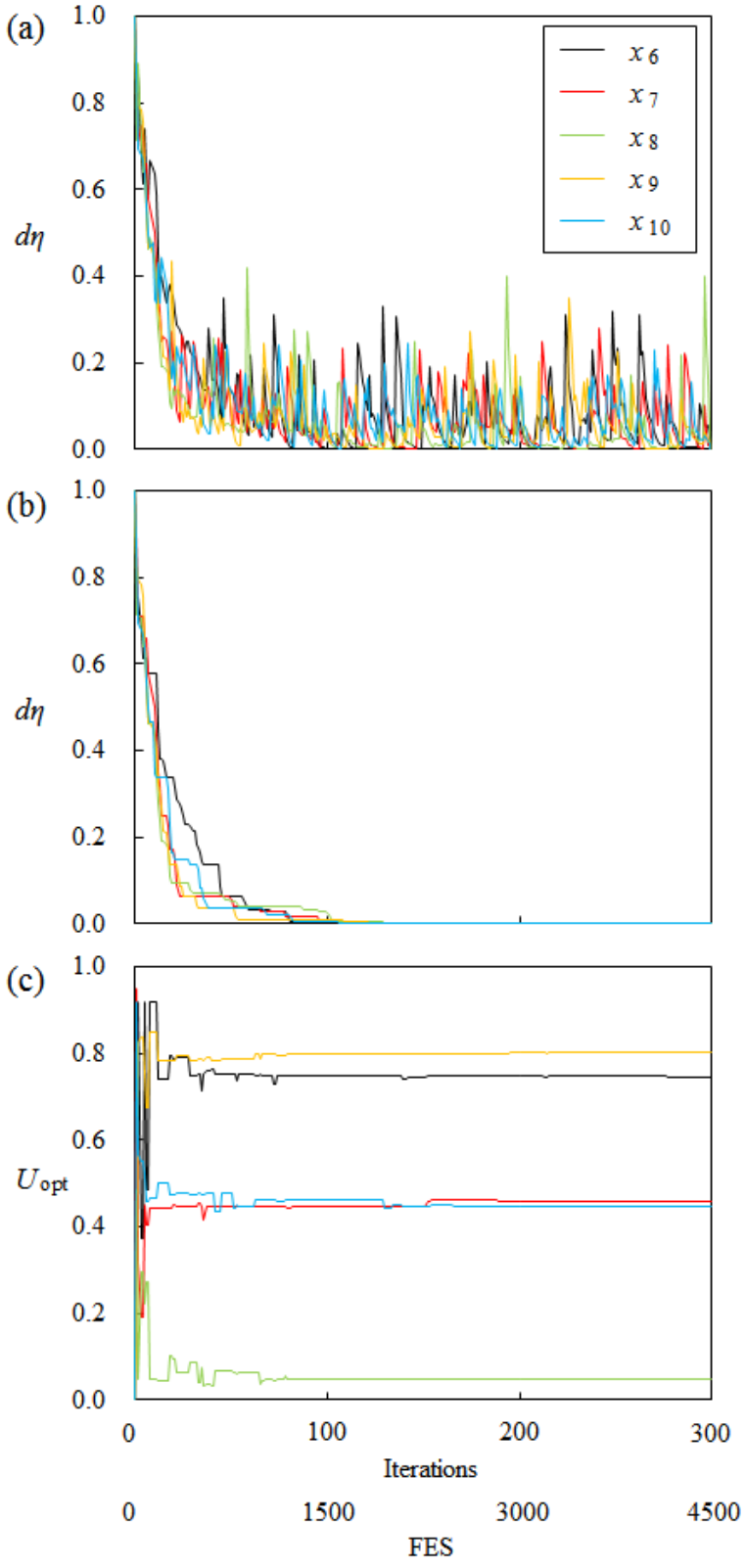

Figure 2-10 Convergence behavior of difference between normalized entropy and normalized BEF for 6-10 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in best case for benchmark function F6

Table 2-6 Minimum values of difference between normalized entropy and normalized BEF for

primary variables in best case for benchmark function F6

i	*Iter*		
	100	200	300
1	0.00659	0.00310	0.00137
2	0.00483	0.00483	0.00037
3	0.00569	0.00114	0.00056
4	0.00416	0.00101	0.00065
5	0.00792	0.00025	0.00025
6	0.00144	0.00015	0.00015
7	0.00683	0.00091	0.00045
8	0.02456	0.00056	0.00056
9	0.00981	0.00063	0.00020
10	0.00472	0.00137	0.00137

- 複合関数 F17

最後に，F17 について結果を示す．F17 は Table 2-4 から分かるように，19 のベンチマーク関数の中で CV の値が最も高い．F17 は F9 と F3 の複合関数であり，x_1 から x_7 は F9，x_8 から x_{10} は F3 に従う．同じく F9 と F3 で構成される F13 も CV が高くなっており，このような傾向は他の多点探索法，PSO においても共通である．この理由は，幾つかの要因が複合的に影響しているため，断定することは難しいものの，1 つには F3 と F9 は，x_i と x_{i+1} で構成される項を含むことが挙げられる．ILHS では主変数毎に目的関数を独立に評価し，次のサンプリングの重み付けを決定する．従って，主変数間の相互関係が考慮されるわけではなく，x_i と x_{i+1} で構成される項が存在する場合，効率的な解探索ができていない可能性がある．

F17 の目的関数の値の変化を Figure 2-11 に示す．縦軸のスケールが F2 と F6 とは異なることに注意する．目的関数の値は初め大きく改善するものの，ほぼ一定値を保つものが比較的多く，結果として計算終了時は広く分布していることが分かる．最適解の改善が進む試行とそうでない試行とでは，$d\eta$の変化が異なる可能性があるため，ここでは，目的関数の値が最も良かった試行と悪かった試行について，$d\eta$と最適解の変化を示し，提案基準の有効性を検証する．

初めに，目的関数の値が最も良かった試行について，Figure 2-12，Figure 2-13 に示す．F3 に関わる変数（x_8, x_9, x_{10}）は最適解への収束が速く，また，$d\eta$の収束と主変数の収束との間には相関があることが分かる．次に，目的関数の値が最も悪かった試行について，Figure 2-14，Figure 2-15 に示す．主変数の最適解への収束が，Figure 2-12，Figure 2-13 と比較して遅く，x_{10} は最適解と異なる局所解付近に収束してしまっており，計算終了まで局所解から抜け出すことができていない．従って，Figure 2-11 から分かるように，目的関数値の劇的な改善は起こらず，その結果，$d\eta$の値の減少も先に示した試行より遅い．このことから，$d\eta$は最適解への収束度合いを表現できているといえる．一方で，繰り返し計算に伴い$d\eta$の値は確実に減少し，解探索が大域的探索から局所的探索に推移している．従って，x_{10} が局所解から抜け出し，目的関数

の値が改善する確率も徐々に低下する．以上のことから，$d\eta$は最適解への収束度合いだけでなく，繰り返し計算の進み具合を考慮した統一的な指標であるといえる．

最後に，F17 の 2 つの独立試行について，Iter = 100, 200, 300のときの$d\eta$の値を見る（Table 2-7）．$d\eta$のグラフから明らかなように，目的関数の値が最も良かった試行（best）の方が悪かった試行（worst）よりも$d\eta$の値は低い．最適解への収束に伴い，$d\eta$の値は 0.01 から 0.02 の間で推移しており，計算終了時は全ての主変数に対して 0.01 を下回っている．

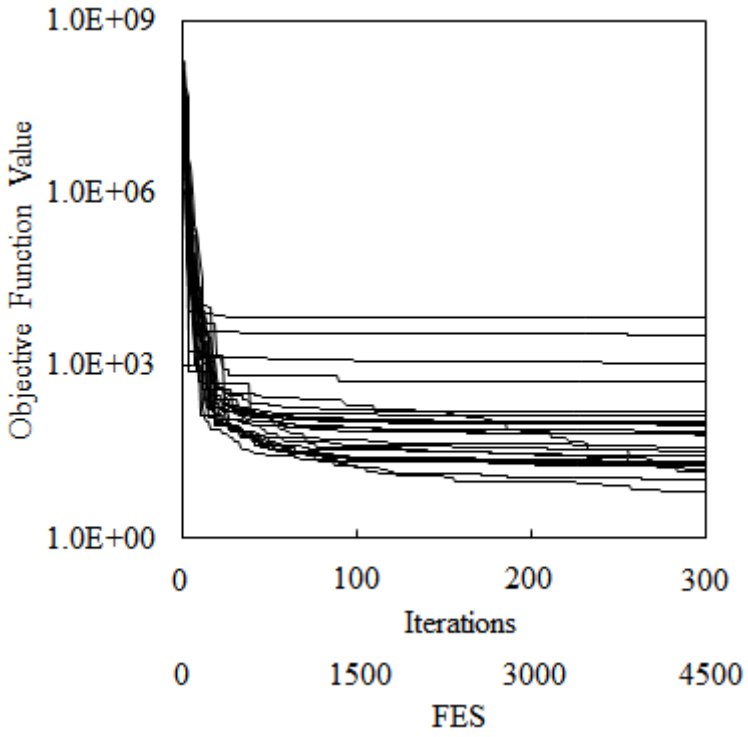

Figure 2-11 Convergence behaviors of objective function in F17

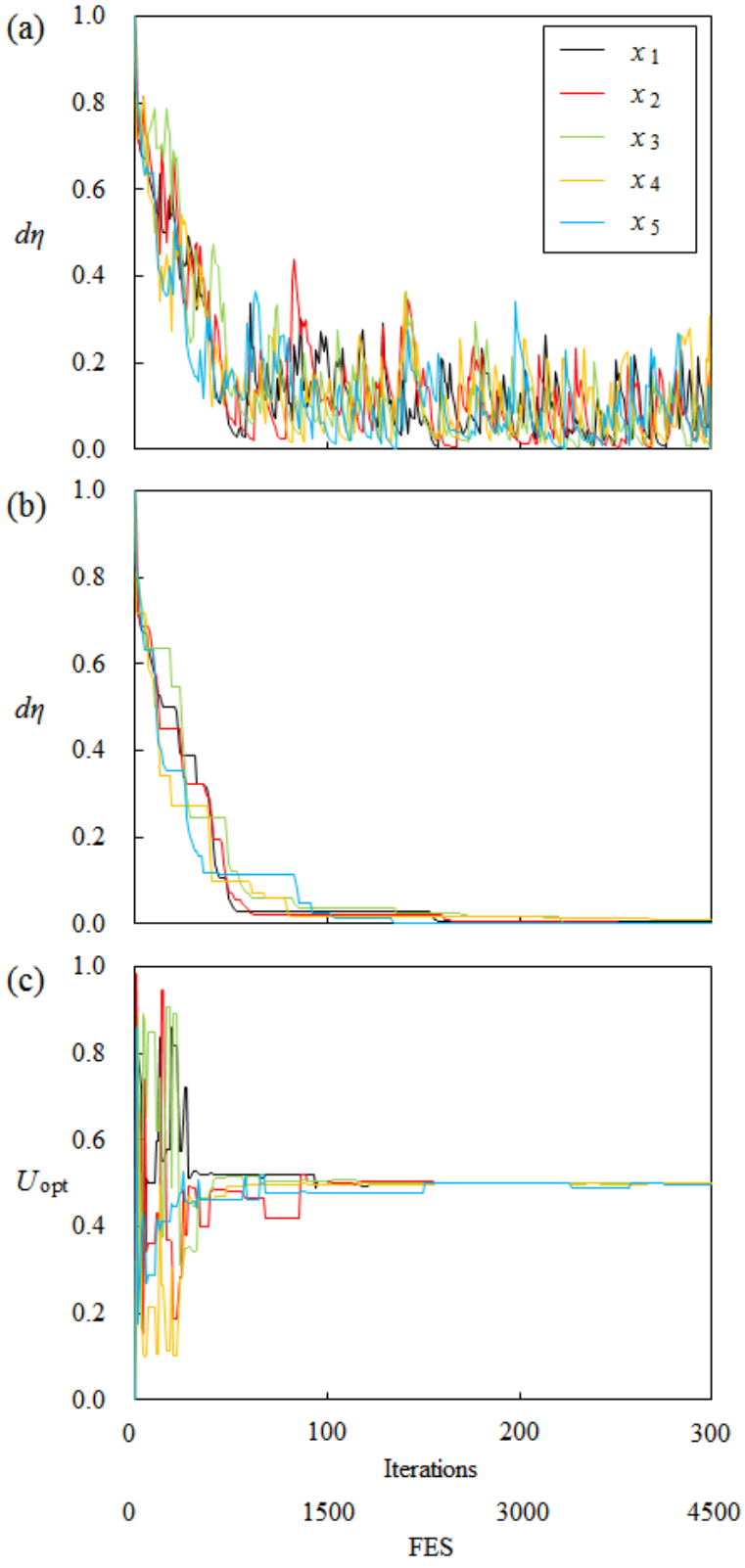

Figure 2-12 Convergence behavior of difference between normalized entropy and normalized BEF for 1-5 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in best case for benchmark function F17

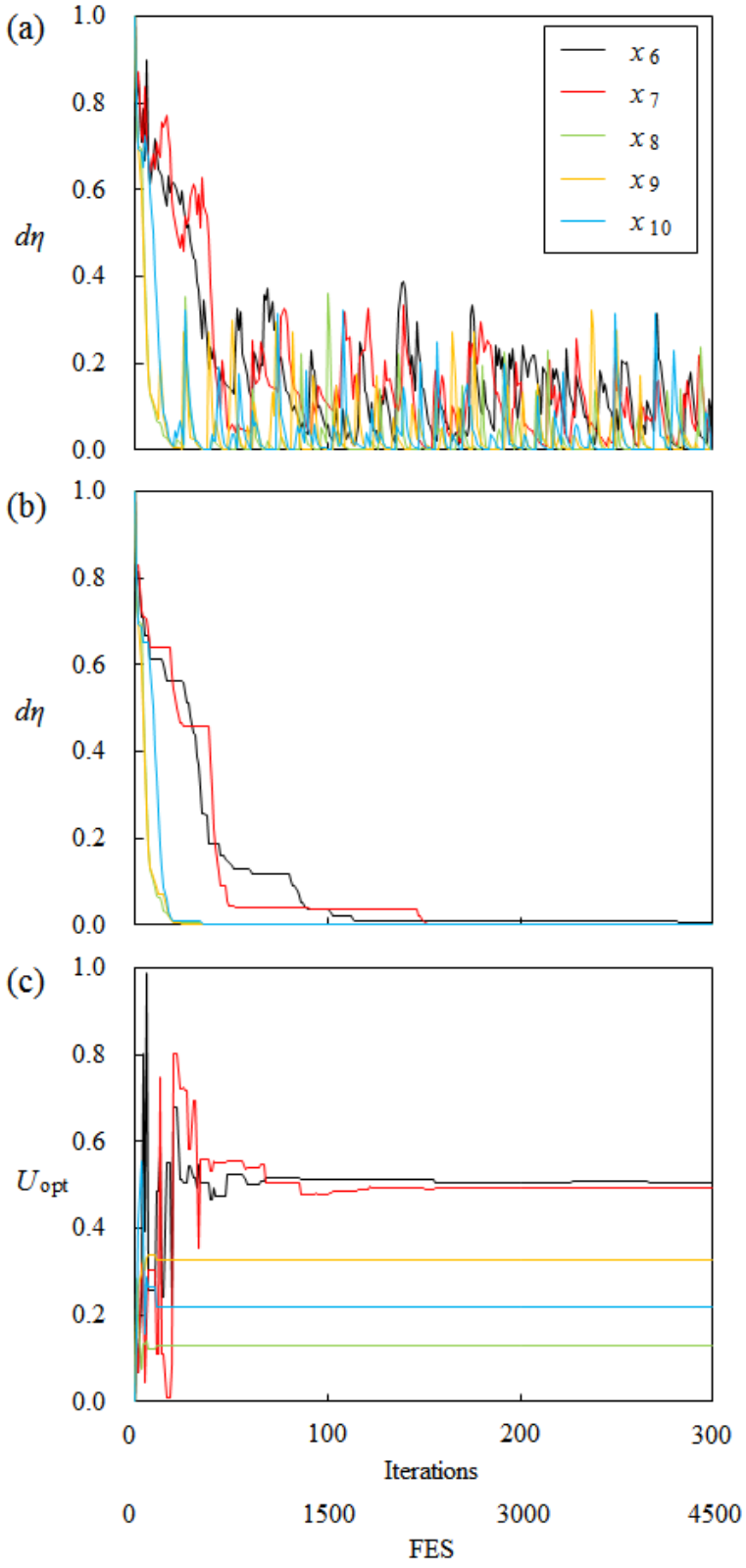

Figure 2-13 Convergence behavior of difference between normalized entropy and normalized BEF for 6-10 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in best case for benchmark function F17

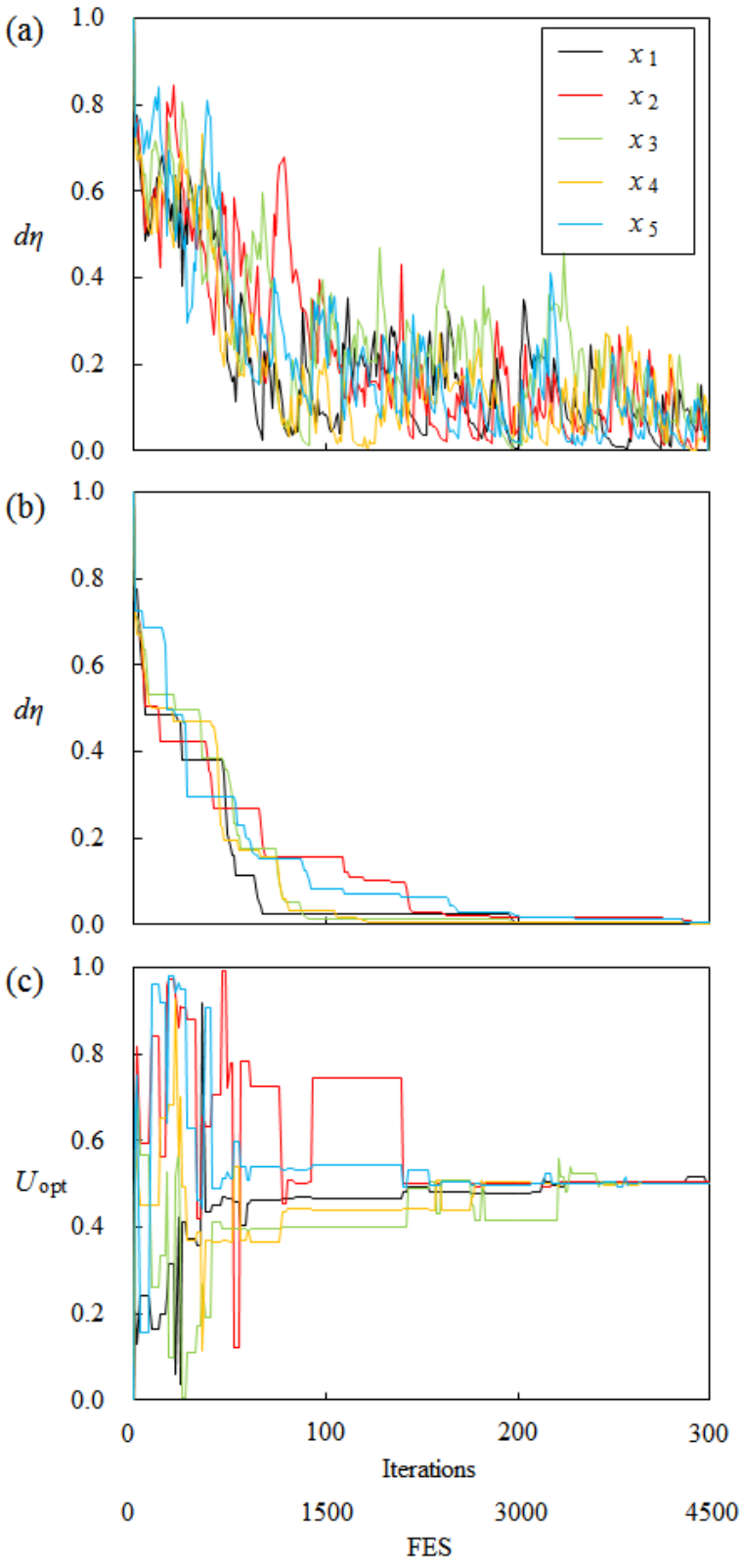

Figure 2-14 Convergence behavior of difference between normalized entropy and normalized BEF for 1-5 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in worst case for benchmark function F17

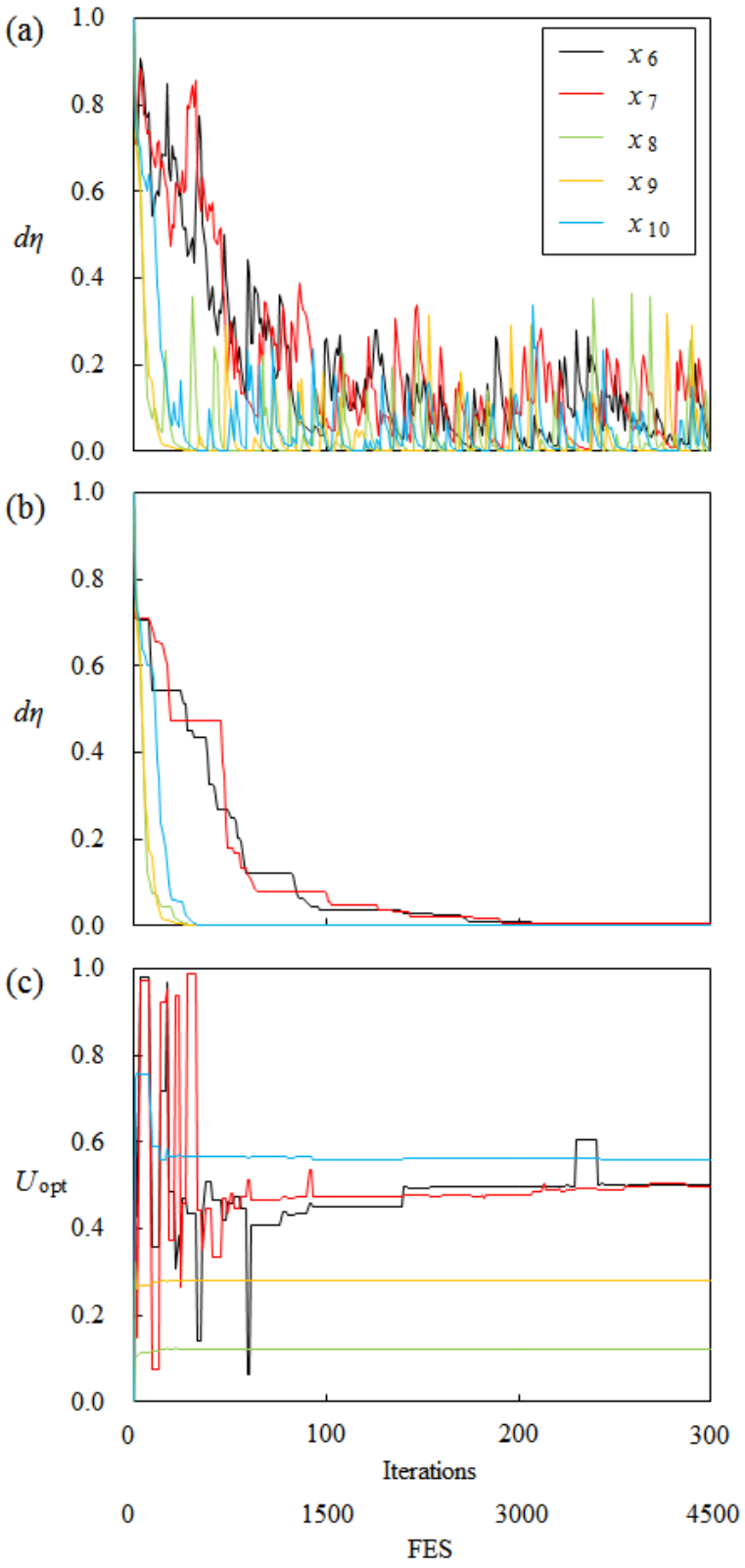

Figure 2-15 Convergence behavior of difference between normalized entropy and normalized BEF for 6-11 primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables in worst case for benchmark function F17

Table 2-7 Minimum values of difference between normalized entropy and normalized BEF for

primary variables in best and worst case for benchmark function F17

	i	*Iter*		
		100	200	300
Best	1	0.02744	0.00700	0.00467
	2	0.02006	0.00614	0.00330
	3	0.03543	0.01742	0.00123
	4	0.01648	0.01648	0.00880
	5	0.02341	0.00207	0.00207
	6	0.03650	0.00746	0.00442
	7	0.03801	0.00265	0.00265
	8	0.00029	0.00002	0.00001
	9	0.00024	0.00011	0.00001
	10	0.00022	0.00022	0.00015
Worst	1	0.02612	0.00457	0.00457
	2	0.15779	0.01880	0.00322
	3	0.01406	0.00597	0.00597
	4	0.03338	0.00528	0.00130
	5	0.08394	0.01961	0.00485
	6	0.03625	0.00834	0.00580
	7	0.07437	0.00591	0.00397
	8	0.00031	0.00002	0.00000
	9	0.00001	0.00000	0.00000
	10	0.00028	0.00002	0.00002

単峰性関数，多峰性関数，及び複合関数について，目的関数，最適解，正規化エントロピーの値の変化を調べた．結果から，最適解の変化と$d\eta$の変化との間には相関関係があることが分かった．$d\eta$の変化から最適解への収束を判断することが可能である．加えて，$d\eta$は繰り返し計算の進行も同時にを表現しており，多点探索法の収束を判断する統一的な指標であるといえる．

$d\eta$の閾値$d\eta_{\min}$については，主変数が最適解へ収束した場合，概ね$d\eta_{\min} \leq 0.01$となっており，以降の章では$d\eta_{\min} = 0.01$と設定する．ここで注意したいのは，要求される$d\eta_{\min}$の値は最適化計算の精度によって変更すべきであり，本研究で求めた値が必ずしも他の条件でも有効ではないということある．

正規化エントロピーによる収束基準は，ILHS による解探索を念頭に開発されたものであるものの，サンプル区間を設定していない他の多点探索法（PSO，DE）でも，サンプル点間の中点を分割点とすることで適用が可能である．

2.4. ロバスト性の向上

多点探索法は乱数を用いるため，1 回の試行で得られる最適解は初期値に大きく依存する．そのため，計算コストに制限が無ければ，1 回の試行でいたずらに評価回数を増加させるよりは，複数の独立試行を行った方が良い．しかしながら，前述のように最適解の値は初期値に大きく依存することから，各独立試行の計算結果は多くの場合，一致せず，また，多点探索法において，このようなばらつきを調整できるパラメータは存在しない．本研究では，このばらつきを低減し，最適化手法のロバスト性を向上させることを目指して，最適化計算で用いる乱数に変更を加えた．具体的には，各試行で発生させる乱数に，従属ラテン超方格法（LHSD）を参考に正規コピュラに基づく依存関係を持たせた．

2.4.1. 従属ラテン超方格法

Natalie & Wolfgang は拡張した LHS を提案した（Natalie & Wolfgang, 2010）：

$$V_{i,n}^{j} = \frac{r_{i,n}^{j} - 1}{n} + \frac{\eta_{i,n}^{j}}{n}, (i = 1,\ldots, n, j = 1,\ldots, d). \tag{23}$$

上式中の$r_{i,n}^{j}$は対応するある実数$X_i^j (i = 1,\ldots, n)$の順位を表す：

$$r_{i,n}^{j}(X_1^j,\ldots, X_n^j) = \sum_{k=1}^{n} 1_{\{X_k^j \le X_i^j\}}, (i = 1,\ldots, n). \tag{24}$$

一般に，X^j $(j = 1,\ldots, d)$には乱数$U^j \sim U(0,1), (j = 1,\ldots, d)$が用いられる．また，$\eta_{i,n}^{j}$は 0 から 1 の値をとる実数であり，選択肢として以下の 4 つが提案されている．

i. $\eta_{i,n}^{j} = B^{n}_{\eta_{i,n}^{j}}\left(U_i^j\right), (i = 1,\ldots, n, j = 1,\ldots, d)$

ii. $\eta_{i,n}^{j} = U_i^j, (i = 1,\ldots, n, j = 1,\ldots, d)$

iii. $\eta_{i,n}^{j} = 1/2$

iv. $\eta_{i,n}^{j} = 1$

B_k^nはベータ関数（Feller, 1971），U_i^jは標準乱数をそれぞれ表す．選択肢 ii を用いたとき，式(15)は式(6)と等しくなる．

ここで，$x^j (j = 1,\ldots, d)$を後述するコピュラ（接合分布関数）による相関を持つ乱数$U^j \sim U(0,1), (j = 1,\ldots, d)$とすれば，生成される d 個のサンプルベクトルも同様に各ベクトル間で相関を持つ．このような LHS を Latin Hypercube Sampling with Dependence (LHSD)と呼び，サンプル点間の分散の値を低減できることが知られている（Natalie & Wolfgang, 2010）．

LHSD は本来，パラメータ間の従属関係を記述するために用いられる．金融の分野でいえば資産価値がそれにあたる．本研究では，パラメータ間ではなく，ILHS の各試行間に従属性を設定することで，得られる最適解のばらつきを抑えることとした．

2.4.2. コピュラ（接合分布関数）

ここでは，LHSD 内の乱数生成に用いるコピュラ（接合分布関数）について説明する．コピュラとは Sklar によって提案された，多次元分布と 1 次元周辺分布の関係を表した分布関数である（Sklar, 1959）．コピュラは，スケールの異なる変数間の相関を記述できる，また相関のある多変量分布を生成できるという点から近年注目されており，主に金融の分野で応用例が見られる（Fisher, 1997; Li, 2000; 戸坂・吉羽, 2005; 篠塚, 2011）．

任意の連続な d 次元同時分布関数 F に対して，

$$F(x_1,\ldots,x_d) = C(F_1(x_1),\ldots,F_d(x_d)), \tag{25}$$

を満たす d 次元接合関数 C が一意に存在し，C をコピュラと呼ぶ（Sklar の定理）．ここで，F_iは F の i 番目の 1 次元周辺分布関数である．任意の$u = (u_1,\ldots,u_d)$ に対して，

$$C(u_1,\ldots,u_d) = F(F_1^{-1}(u_1),\ldots,F_d^{-1}(u_d)), \tag{26}$$

が成り立つ．C，X_iの密度関数をそれぞれ，$c(u_1,\ldots,u_d)$，$f_i(x_i)$，f とする．

ここで$c(u_1,\ldots,u_d)$は，

$$c(u_1,\ldots,u_d) = \frac{\partial^n C(u_1,\ldots,u_d)}{\partial u_1 \ldots \partial u_d}, \tag{27}$$

で表される．式(17)に代入することで，

$$f(x_1,\ldots,x_d) = c(F_1(x_1),\ldots,F_d(x_d)) \prod_{i=1}^{d} f_i(x_i). \tag{28}$$

を得る．

本研究では，最も基本的で適用事例の多い正規コピュラを用いる．正規コピュラとは多変量正規分布と同じ構造を有するコピュラであり，n 変量正規分布のコピュラは，

$$C(u_1,\ldots,u_d) = \Phi_n(\Phi_1^{-1}(u_1),\ldots,\Phi_d^{-1}(u_d);\Sigma), \tag{29}$$

と表される．ここで，Σは相関行列，Φ_nは n 変量正規分布の分布関数を示す．いま，式(20)より，

$$\frac{1}{(2\pi)^{\frac{N}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}x^T \Sigma^{-1} x\right) = c(\Phi_1(x_1),\ldots,\Phi_d(x_d)) \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x_i^2\right), \tag{30}$$

が成り立ち，このとき正規コピュラは，

$$C(u_1,\ldots,u_d) = \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\boldsymbol{\omega}^T (\boldsymbol{\Sigma}^{-1} - \boldsymbol{I})\boldsymbol{\omega}\right), \tag{31}$$

となる．ただし，$\boldsymbol{\omega} = \boldsymbol{\Phi}_1^{-1}(\boldsymbol{u})$，$\boldsymbol{I}$ は単位行列とする．正規コピュラに従う乱数発生法は以下の様

になる（Rank *et al.*, 2006）.

1. 相関係数ρより相関行列Σを求める.
2. Cholesky 分解を用いてΣ=AA'を満たす三角行列 A を求める.
3. 標準正規分布より独立な乱数$\hat{\mathbf{X}} = (\hat{x}_1,..., \hat{x}_d)'$を発生させる（Figure 2-16(a)）.
4. n 変量標準正規分布に従う乱数$\mathbf{X} = (x_1 ,..., x_d)'$を$\mathbf{X} = \mathbf{A}\hat{\mathbf{X}}$より計算する（Figure 2-16(b)）.
5. 1 変量標準正規分布の分布関数 Φ_1を用いて, $u_i = \Phi_1(x_i)$ $(i = 1,...,d)$とする（Figure 2-16(c)）.

正規コピュラにおいて，相関行列Σの要素σ_{ij}は相関係数ρを用いて以下の様に表される,

$$\sigma_{ij} = \begin{cases} 1 & (i = j) \\ \rho & (i \neq j) \end{cases}. \tag{32}$$

また，分布関数Φ_1は累積分布関数であり，エラー関数 erf を用いて,

$$\Phi_1(x_i) = \frac{1}{2}(1 + \mathrm{erf}) = \frac{1}{2}\left(1 + \frac{2}{\sqrt{\pi}}\int_0^{\frac{x}{\sqrt{\pi}}} e^{-t^2} dt\right) \tag{33}$$

と表せる．Figure 2-16 に$d = 2$のときの正規コピュラに従う乱数を例として示す．相関係数 $\rho = 0.90$, サンプル数$n_{\mathrm{pop}} = 1000$とした.

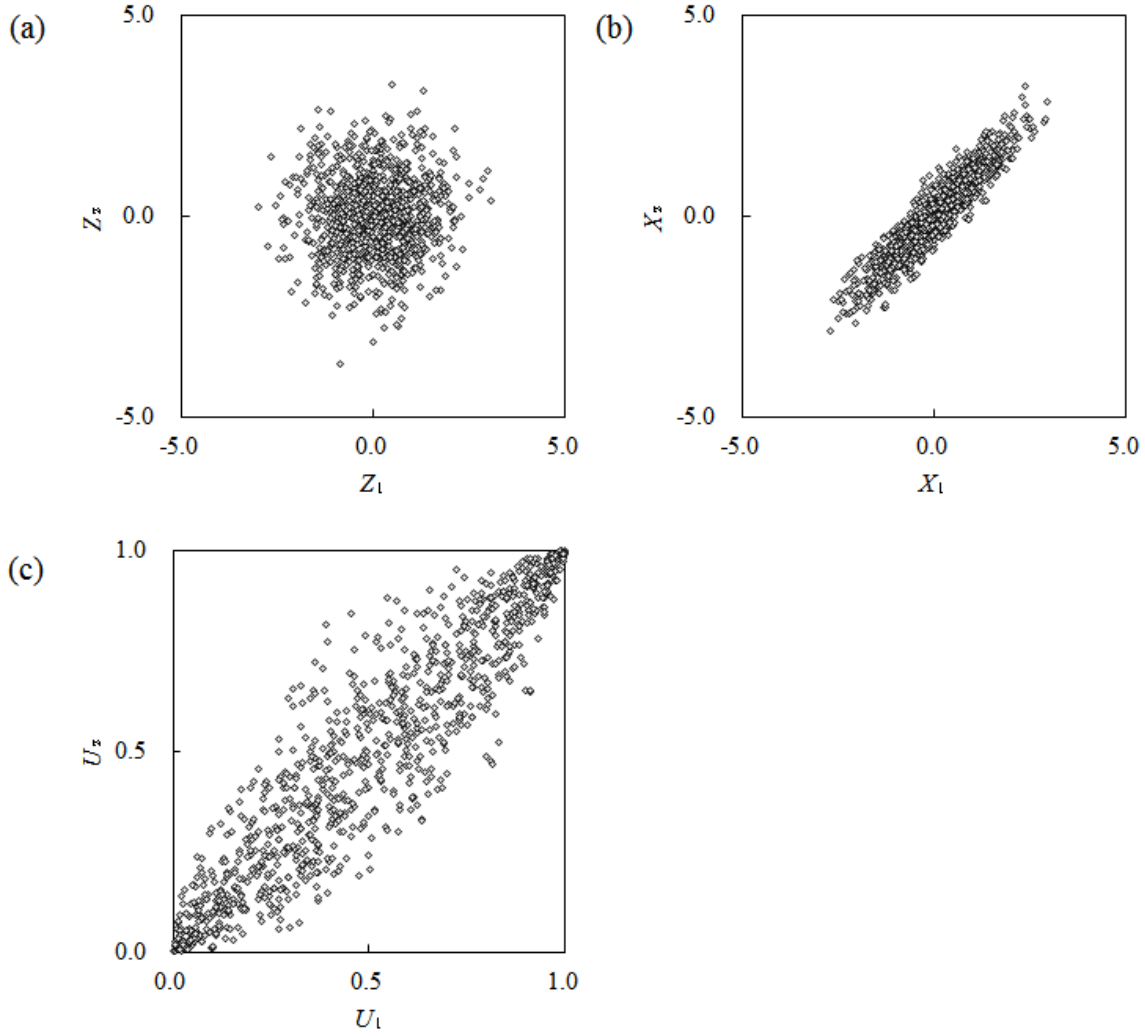

Figure 2-16 Two dimensional sets of (a) multivariate normal random numbers, (b) multivariate normal random numbers with dependence and (c) random numbers using Gaussian copula (n_{pop} = 1000, $\rho = 0.90$)

Figure 2-17 はρを変化させたときの正規コピュラに従う乱数である．Figure 2-16 と同様にサンプル数n_{pop} = 1000である．

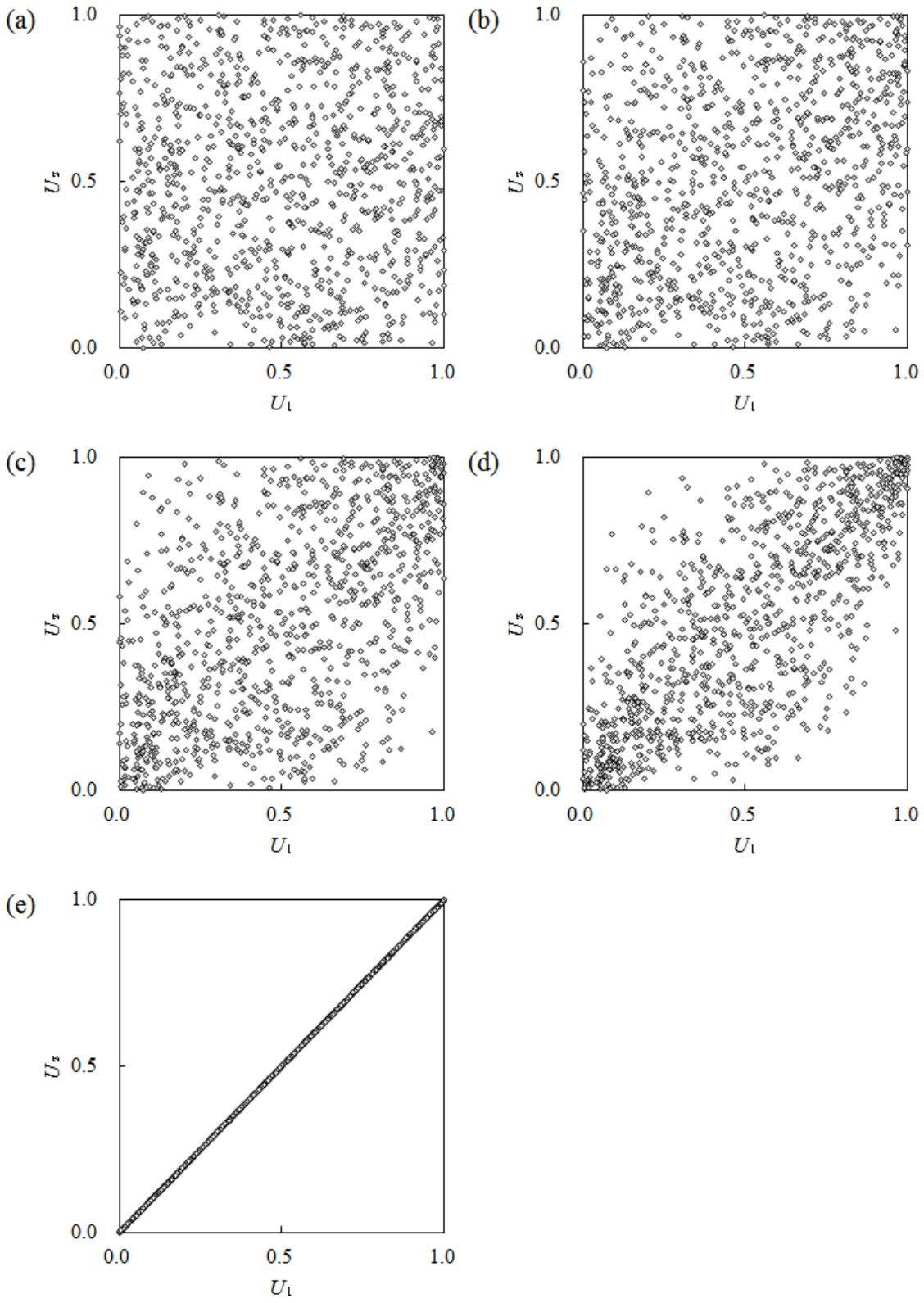

Figure 2-17 Two dimensional sets of random numbers using Gaussian copula when Sperman' s ρ is (a) 0.00, (b) 0.25, (c) 0.50, (d) 0.75 and (e) 1.00 (n_{pop} = 1000)

2.4.3. ベンチマーク関数への適用

前述の LHSD を用いて ILHS のロバスト性が向上するかを確認する．改良したアルゴリズムを前節と同じ 19 のベンチマーク関数に対して各 25 回の試行を行い，目的関数値の平均値，分散及び変動係数を比較した．$d = 10$，$n_{\mathrm{pop}} = 15$，$\mathrm{Iter}_{\max} = 300$ とし，相関係数は $\rho = 0.0, 0.1, 0.3, 0.5, 0.7, 0.9$の 6 ケースとした．$\rho = 0.0$のケースが従来の ILHS にあたる．選択肢 ii，iii の 2 つの設定方法を選択した．

目的関数値の平均値，分散及び変動係数の値を Table 2-8 から Table 2-13 に示す．Figure 2-18 から Figure 2-25 はρを変化させたときの平均値，変動係数についてベンチマーク関数ごとに示したものである．関数によって縦軸の値が異なることに注意が必要である．ρの値の上昇に伴い，目的関数の平均値は上昇，変動係数は低下する．この傾向は目的関数の値の大きい，つまり ILHS が苦手とする関数において顕著である．また，ILHSD の選択肢 ii と iii では，iii の場合の変化量が少ない．これは，選択肢 iii が乱数を含まない形式であり，試行間でより相関の高いサンプル点を生成するためと考えられる．

結果から，LHSD を用いることで，最適解の値を維持しつつ，ロバスト性を向上することが可能であることが示された．$\rho = 0.1, 0.3$といった比較的小さな値でも目的関数値の変動係数を低下させることが可能であり，最適化計算に効果があるといえる．

Table 2-8 Experimental results of average objective function values obtained by ILHSD with option ii

Function	ρ					
	0.0	0.1	0.3	0.5	0.7	0.9
F1	7.00E-01	1.96E+01	3.12E+01	3.71E+01	1.99E+02	3.11E+02
F2	4.11E+00	1.63E+01	2.00E+01	2.39E+01	3.96E+01	7.32E+01
F3	4.36E+03	7.38E+04	2.16E+05	4.00E+05	2.40E+06	3.50E+07
F4	5.93E+00	7.95E+00	9.58E+00	1.03E+01	1.53E+01	2.26E+01
F5	7.21E-01	1.36E+00	1.22E+00	1.31E+00	1.73E+00	3.81E+00
F6	1.55E+00	4.06E+00	3.63E+00	4.24E+00	4.91E+00	7.98E+00
F7	1.16E-01	9.31E-01	6.59E-01	7.80E-01	1.58E+00	2.28E+00
F8	4.42E+02	6.94E+02	6.72E+02	7.16E+02	6.38E+02	2.56E+03
F9	9.56E+00	2.12E+01	1.85E+01	2.16E+01	2.98E+01	4.37E+01
F10	1.08E+00	1.33E+01	1.02E+01	9.31E+00	1.97E+01	7.69E+01
F11	9.43E+00	1.60E+01	1.58E+01	1.88E+01	2.43E+01	3.49E+01
F12	2.57E+00	2.21E+01	1.78E+01	2.35E+01	5.35E+01	9.47E+01
F13	9.44E+02	1.11E+04	3.24E+03	1.88E+04	8.33E+05	5.24E+06
F14	3.56E+00	7.30E+00	8.39E+00	6.64E+00	7.61E+00	1.10E+01
F15	1.45E-01	1.08E+00	1.09E+00	1.25E+00	1.60E+00	4.78E+00
F16	4.33E+00	1.71E+01	1.96E+01	1.58E+01	2.17E+01	5.20E+01
F17	5.32E+02	1.60E+02	7.17E+01	7.27E+02	1.53E+02	1.86E+02
F18	1.85E+00	3.99E+00	4.18E+00	4.45E+00	5.20E+00	7.27E+00
F19	5.15E-01	2.78E+00	2.94E+00	3.47E+00	5.40E+00	7.35E+00
F20	2.22E+00	4.54E+00	4.26E+00	4.23E+00	5.40E+00	8.56E+00
F21	1.08E+01	1.64E+01	1.79E+01	1.66E+01	2.74E+01	3.99E+01
F22	6.87E-01	1.22E+00	1.30E+00	1.44E+00	1.95E+00	3.60E+00
F23	1.74E-01	5.26E+00	4.17E+00	9.79E+00	3.03E+01	7.08E+01
F24	3.45E+05	2.63E+08	1.45E+08	8.98E+07	2.34E+09	3.94E+10

Table 2-9 Experimental results of standard deviation of objective function values obtained by ILHSD with option ii

Function	ρ					
	0.0	0.1	0.3	0.5	0.7	0.9
F1	0.950	0.606	0.578	0.532	0.456	0.429
F2	0.301	0.231	0.186	0.211	0.205	0.138
F3	1.131	0.670	0.874	1.039	0.764	0.462
F4	0.478	0.351	0.347	0.236	0.284	0.268
F5	0.240	0.143	0.107	0.134	0.302	0.226
F6	0.508	0.188	0.197	0.174	0.158	0.101
F7	0.475	0.279	0.314	0.197	0.285	0.307
F8	0.539	0.396	0.287	0.292	0.432	0.347
F9	0.478	0.228	0.238	0.214	0.167	0.136
F10	0.668	0.341	0.329	0.276	0.236	0.234
F11	0.443	0.162	0.218	0.178	0.189	0.128
F12	0.474	0.385	0.338	0.456	0.390	0.664
F13	1.551	0.788	1.232	1.192	0.613	0.593
F14	0.515	0.338	0.293	0.321	0.326	0.237
F15	0.751	0.355	0.283	0.414	0.351	0.297
F16	0.313	0.291	0.207	0.297	0.275	0.241
F17	2.812	1.902	2.123	2.760	0.148	0.072
F18	0.366	0.269	0.311	0.254	0.275	0.257
F19	0.871	0.371	0.391	0.373	0.277	0.286
F20	0.443	0.196	0.171	0.184	0.176	0.109
F21	0.396	0.387	0.390	0.345	0.314	0.238
F22	0.234	0.165	0.134	0.227	0.256	0.236
F23	0.731	0.582	0.599	0.797	0.449	0.372
F24	0.714	0.714	1.342	1.197	0.896	0.535

Table 2-10 Experimental results of CV of objective function values obtained by ILHSD with option ii

Function	Avg					
	0.0	0.1	0.3	0.5	0.7	0.9
F1	7.00E-01	2.54E+01	3.12E+01	3.43E+01	1.51E+02	1.51E+02
F2	4.11E+00	1.56E+01	2.00E+01	2.35E+01	3.83E+01	3.83E+01
F3	4.36E+03	1.17E+05	2.16E+05	2.41E+05	1.46E+06	1.46E+06
F4	5.93E+00	8.80E+00	9.58E+00	9.00E+00	1.55E+01	1.55E+01
F5	7.21E-01	1.31E+00	1.22E+00	1.44E+00	1.76E+00	1.76E+00
F6	1.55E+00	4.38E+00	3.63E+00	4.25E+00	5.15E+00	5.15E+00
F7	1.16E-01	9.71E-01	6.59E-01	6.06E-01	1.66E+00	1.66E+00
F8	4.42E+02	6.62E+02	6.72E+02	6.38E+02	7.74E+02	7.74E+02
F9	9.56E+00	2.11E+01	1.85E+01	2.25E+01	2.77E+01	2.77E+01
F10	1.08E+00	1.32E+01	1.02E+01	8.87E+00	1.79E+01	1.79E+01
F11	9.43E+00	1.77E+01	1.58E+01	1.63E+01	2.33E+01	2.33E+01
F12	2.57E+00	2.40E+01	1.78E+01	2.14E+01	4.55E+01	4.55E+01
F13	9.44E+02	9.60E+03	3.24E+03	1.92E+04	6.18E+05	6.18E+05
F14	3.56E+00	7.36E+00	8.39E+00	7.45E+00	8.98E+00	8.98E+00
F15	1.45E-01	9.88E-01	1.09E+00	1.27E+00	1.90E+00	1.90E+00
F16	4.33E+00	1.73E+01	1.96E+01	1.82E+01	2.52E+01	2.52E+01
F17	5.32E+02	2.73E+02	7.17E+01	6.74E+02	1.51E+02	1.51E+02
F18	1.85E+00	4.41E+00	4.18E+00	4.46E+00	5.59E+00	5.59E+00
F19	5.15E-01	2.67E+00	2.94E+00	3.49E+00	5.48E+00	5.48E+00
F20	2.22E+00	4.56E+00	4.26E+00	4.55E+00	4.84E+00	4.84E+00
F21	1.08E+01	1.79E+01	1.79E+01	1.89E+01	2.67E+01	2.67E+01
F22	6.87E-01	1.28E+00	1.30E+00	1.32E+00	1.81E+00	1.81E+00
F23	1.74E-01	5.08E+00	4.17E+00	7.45E+00	2.16E+01	2.16E+01
F24	3.45E+05	3.20E+08	1.45E+08	9.12E+07	2.35E+09	2.35E+09

Table 2-11 Experimental results of average objective function values obtained by ILHSD with option iii

Function	Std					
	0.0	0.1	0.3	0.5	0.7	0.9
F1	6.65E-01	2.04E+01	1.80E+01	2.15E+01	5.84E+01	5.84E+01
F2	1.23E+00	3.53E+00	3.72E+00	5.08E+00	7.78E+00	7.78E+00
F3	4.93E+03	1.01E+05	1.88E+05	1.74E+05	1.09E+06	1.09E+06
F4	2.83E+00	3.39E+00	3.32E+00	3.13E+00	5.40E+00	5.40E+00
F5	1.73E-01	1.87E-01	1.32E-01	3.64E-01	4.15E-01	4.15E-01
F6	7.85E-01	5.45E-01	7.14E-01	7.54E-01	7.58E-01	7.58E-01
F7	5.53E-02	2.93E-01	2.07E-01	2.45E-01	4.34E-01	4.34E-01
F8	2.38E+02	2.52E+02	1.93E+02	2.06E+02	2.76E+02	2.76E+02
F9	4.57E+00	4.37E+00	4.39E+00	5.18E+00	4.68E+00	4.68E+00
F10	7.19E-01	4.13E+00	3.34E+00	3.01E+00	5.21E+00	5.21E+00
F11	4.18E+00	3.74E+00	3.45E+00	3.62E+00	3.86E+00	3.86E+00
F12	1.22E+00	1.06E+01	6.01E+00	7.97E+00	1.57E+01	1.57E+01
F13	1.46E+03	1.18E+04	3.99E+03	1.85E+04	3.74E+05	3.74E+05
F14	1.84E+00	2.01E+00	2.46E+00	2.41E+00	2.43E+00	2.43E+00
F15	1.09E-01	3.60E-01	3.09E-01	3.73E-01	7.83E-01	7.83E-01
F16	1.36E+00	5.16E+00	4.06E+00	3.90E+00	5.00E+00	5.00E+00
F17	1.50E+03	5.01E+02	1.52E+02	2.47E+03	2.47E+01	2.47E+01
F18	6.77E-01	1.14E+00	1.30E+00	1.12E+00	1.26E+00	1.26E+00
F19	4.48E-01	1.10E+00	1.15E+00	1.03E+00	1.64E+00	1.64E+00
F20	9.86E-01	6.37E-01	7.28E-01	7.87E-01	1.01E+00	1.01E+00
F21	4.28E+00	7.51E+00	6.97E+00	6.78E+00	7.53E+00	7.53E+00
F22	1.61E-01	1.57E-01	1.74E-01	3.09E-01	4.35E-01	4.35E-01
F23	1.27E-01	4.47E+00	2.50E+00	4.05E+00	9.88E+00	9.88E+00
F24	2.46E+05	6.48E+08	1.95E+08	1.21E+08	1.79E+09	1.79E+09

Table 2-12 Experimental results of standard deviation of objective function values obtained by ILHSD with option iii

Function	CV					
	0.0	0.1	0.3	0.5	0.7	0.9
F1	0.950	0.801	0.578	0.626	0.386	0.386
F2	0.301	0.227	0.186	0.216	0.203	0.203
F3	1.131	0.865	0.874	0.722	0.750	0.750
F4	0.478	0.385	0.347	0.348	0.349	0.349
F5	0.240	0.143	0.107	0.254	0.236	0.236
F6	0.508	0.124	0.197	0.177	0.147	0.147
F7	0.475	0.302	0.314	0.404	0.261	0.261
F8	0.539	0.380	0.287	0.323	0.356	0.356
F9	0.478	0.207	0.238	0.230	0.169	0.169
F10	0.668	0.312	0.329	0.340	0.291	0.291
F11	0.443	0.211	0.218	0.223	0.166	0.166
F12	0.474	0.440	0.338	0.372	0.345	0.345
F13	1.551	1.227	1.232	0.965	0.605	0.605
F14	0.515	0.274	0.293	0.324	0.271	0.271
F15	0.751	0.365	0.283	0.294	0.413	0.413
F16	0.313	0.299	0.207	0.214	0.198	0.198
F17	2.812	1.836	2.123	3.660	0.164	0.164
F18	0.366	0.259	0.311	0.250	0.225	0.225
F19	0.871	0.412	0.391	0.296	0.299	0.299
F20	0.443	0.140	0.171	0.173	0.208	0.208
F21	0.396	0.419	0.390	0.358	0.282	0.282
F22	0.234	0.122	0.134	0.234	0.240	0.240
F23	0.731	0.879	0.599	0.544	0.457	0.457
F24	0.714	2.024	1.342	1.328	0.762	0.762

Table 2-13 Experimental results of CV of objective function values obtained by ILHSD with option iii

Function	ρ					
	0.0	0.1	0.3	0.5	0.7	0.9
F1	6.65E-01	1.19E+01	1.80E+01	1.97E+01	9.05E+01	1.33E+02
F2	1.23E+00	3.76E+00	3.72E+00	5.03E+00	8.12E+00	1.01E+01
F3	4.93E+03	4.94E+04	1.88E+05	4.16E+05	1.83E+06	1.62E+07
F4	2.83E+00	2.79E+00	3.32E+00	2.42E+00	4.34E+00	6.05E+00
F5	1.73E-01	1.94E-01	1.32E-01	1.75E-01	5.24E-01	8.59E-01
F6	7.85E-01	7.64E-01	7.14E-01	7.37E-01	7.73E-01	8.09E-01
F7	5.53E-02	2.59E-01	2.07E-01	1.53E-01	4.50E-01	7.00E-01
F8	2.38E+02	2.75E+02	1.93E+02	2.09E+02	2.76E+02	8.87E+02
F9	4.57E+00	4.83E+00	4.39E+00	4.63E+00	4.99E+00	5.95E+00
F10	7.19E-01	4.53E+00	3.34E+00	2.57E+00	4.65E+00	1.80E+01
F11	4.18E+00	2.59E+00	3.45E+00	3.35E+00	4.60E+00	4.46E+00
F12	1.22E+00	8.53E+00	6.01E+00	1.07E+01	2.09E+01	6.28E+01
F13	1.46E+03	8.74E+03	3.99E+03	2.24E+04	5.11E+05	3.11E+06
F14	1.84E+00	2.47E+00	2.46E+00	2.14E+00	2.49E+00	2.60E+00
F15	1.09E-01	3.83E-01	3.09E-01	5.17E-01	5.61E-01	1.42E+00
F16	1.36E+00	4.99E+00	4.06E+00	4.70E+00	5.96E+00	1.25E+01
F17	1.50E+03	3.05E+02	1.52E+02	2.01E+03	2.26E+01	1.34E+01
F18	6.77E-01	1.07E+00	1.30E+00	1.13E+00	1.43E+00	1.87E+00
F19	4.48E-01	1.03E+00	1.15E+00	1.29E+00	1.50E+00	2.11E+00
F20	9.86E-01	8.90E-01	7.28E-01	7.79E-01	9.52E-01	9.32E-01
F21	4.28E+00	6.35E+00	6.97E+00	5.74E+00	8.60E+00	9.51E+00
F22	1.61E-01	2.01E-01	1.74E-01	3.26E-01	4.99E-01	8.48E-01
F23	1.27E-01	3.06E+00	2.50E+00	7.80E+00	1.36E+01	2.63E+01
F24	2.46E+05	1.88E+08	1.95E+08	1.07E+08	2.09E+09	2.11E+10

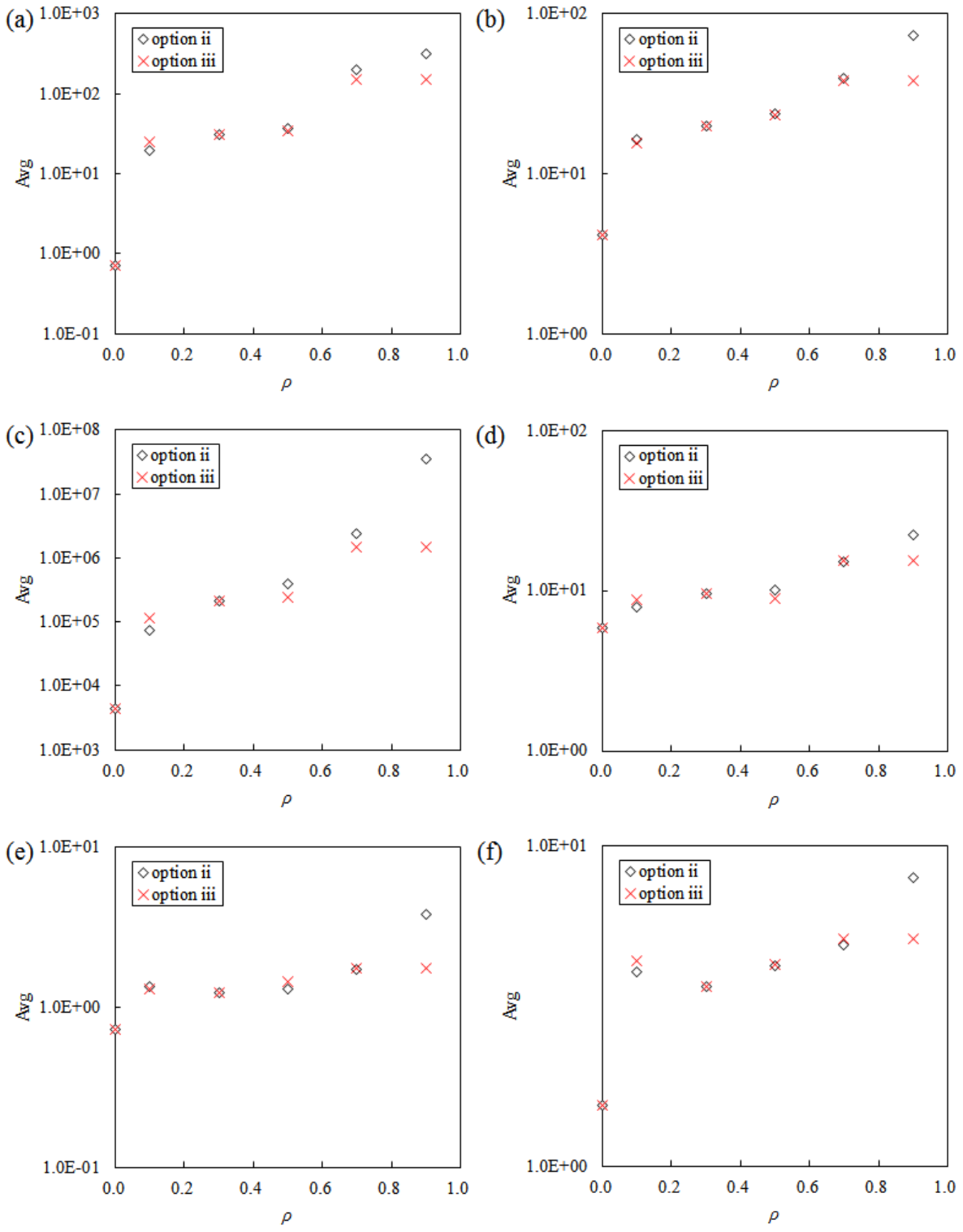

Figure 2-18 Distribution of average objective function values obtained by ILHSD with option ii and iii on the benchmark function (a) F1, (b) F2, (c) F3, (d) F4, (e) F5 and (f) F6

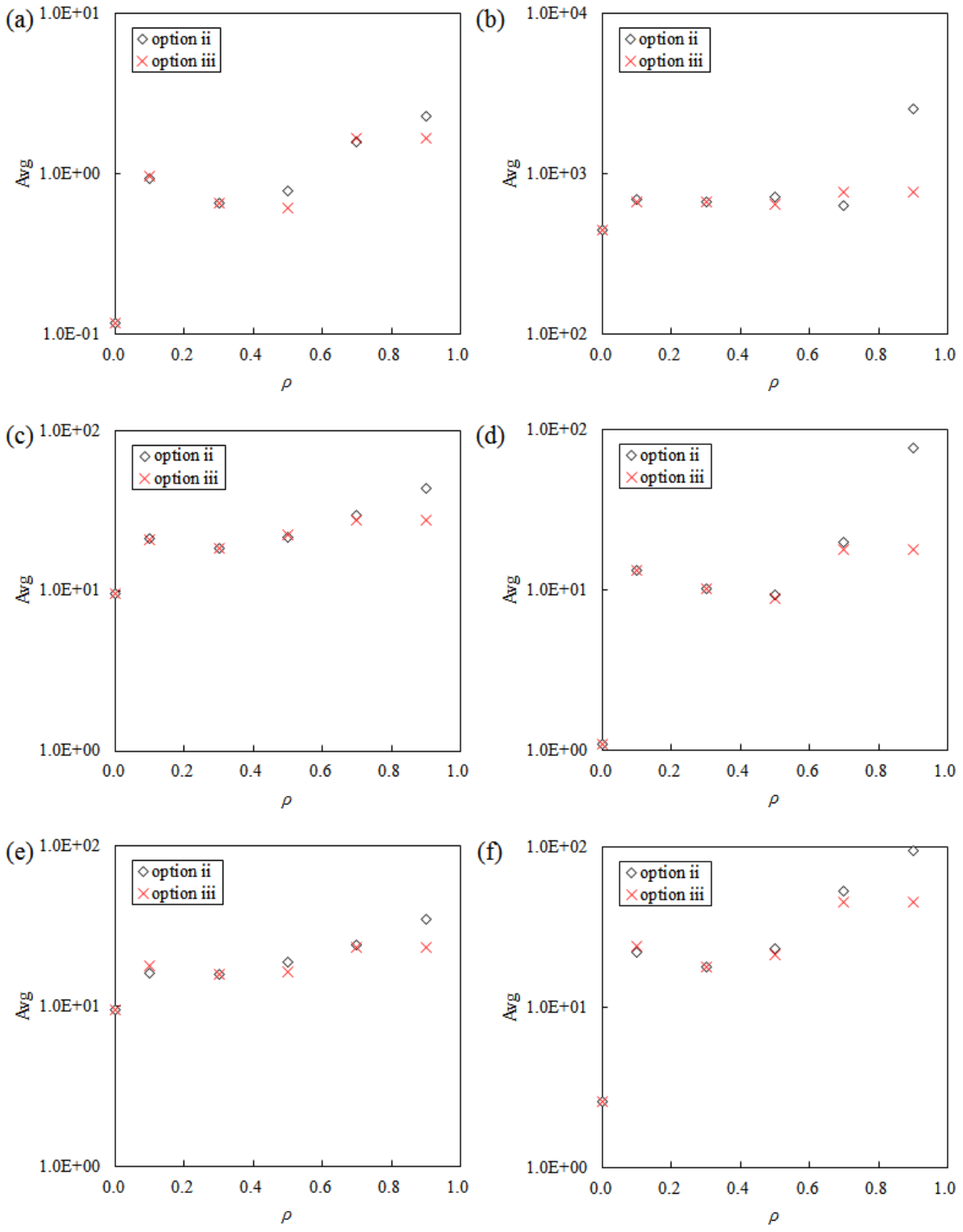

Figure 2-19 Distribution of average objective function values obtained by ILHSD with option ii and iii on the benchmark function (a) F7, (b) F8, (c) F9, (d) F10, (e) F11 and (f) F12

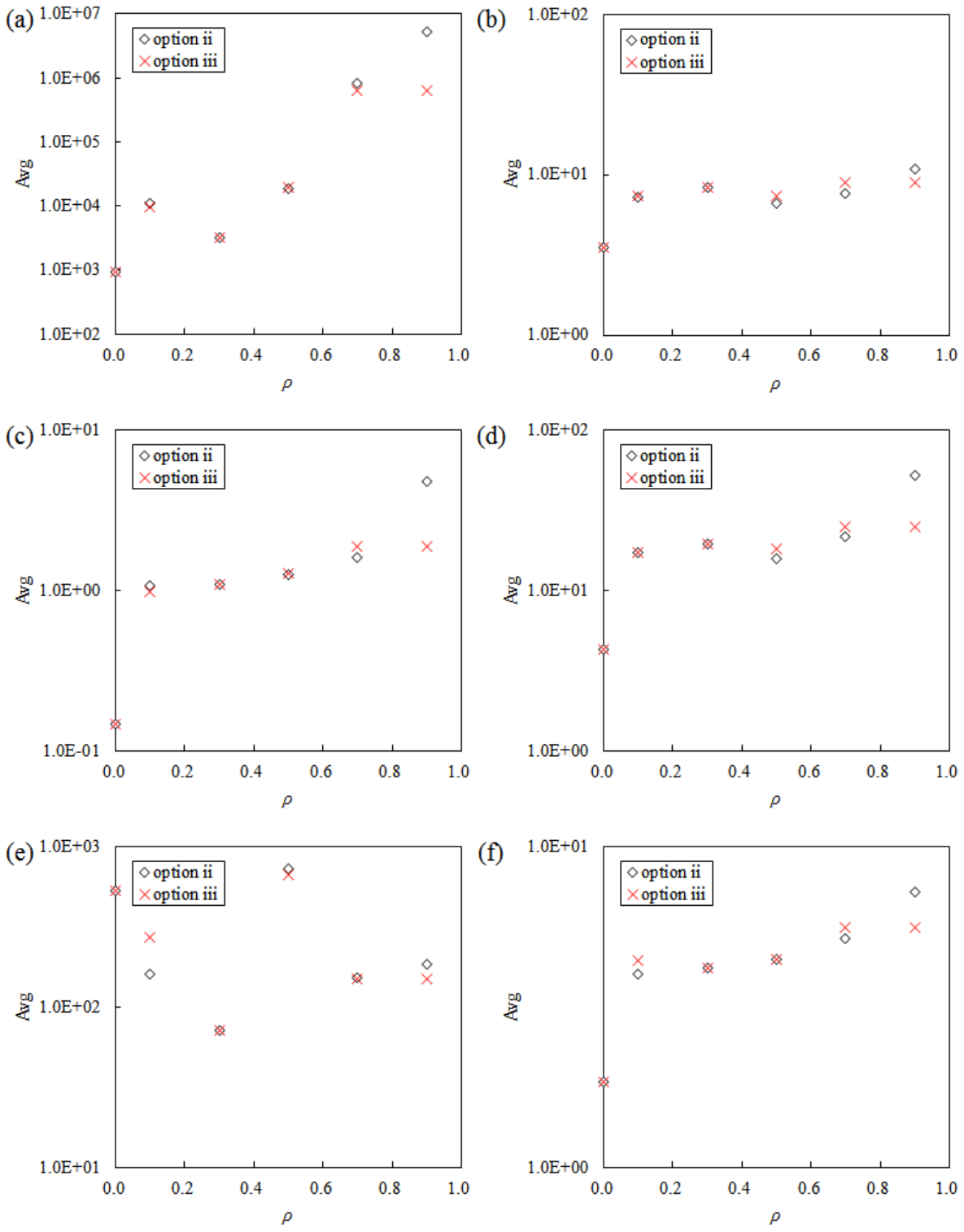

Figure 2-20 Distribution of CV average function values obtained by ILHSD with option ii and iii on the benchmark function (a) F13, (b) F14, (c) F15, (d) F16, (e) F17 and (f) F18

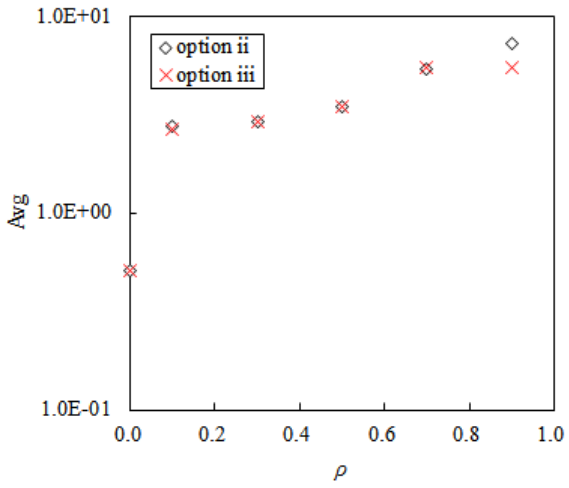

Figure 2-21 Distribution of average objective function values obtained by ILHSD with option ii and iii on the benchmark function F19

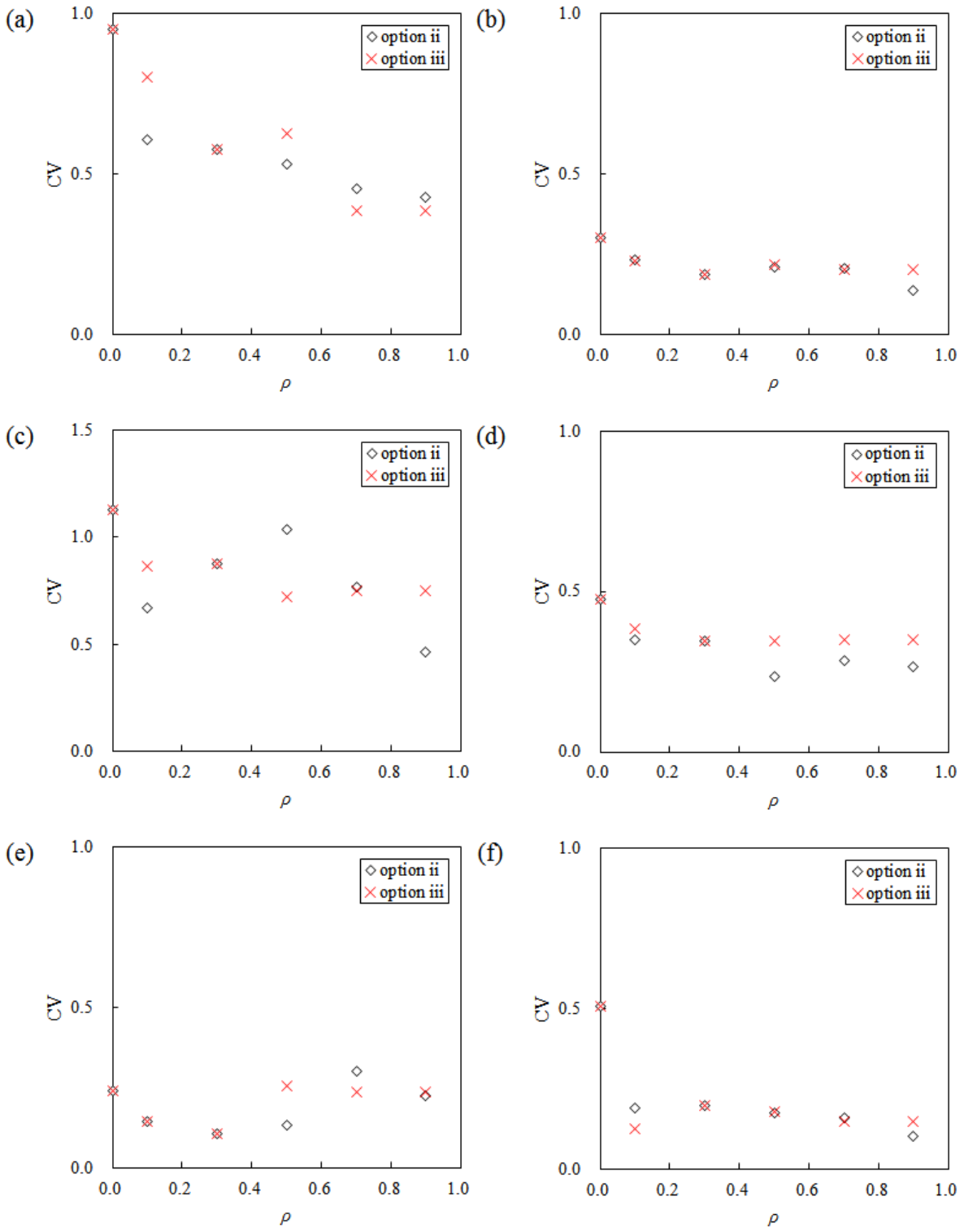

Figure 2-22 Distribution of CV of objective function values obtained by ILHSD with option ii and iii on the benchmark function (a) F1, (b) F2, (c) F3, (d) F4, (e) F5 and (f) F6

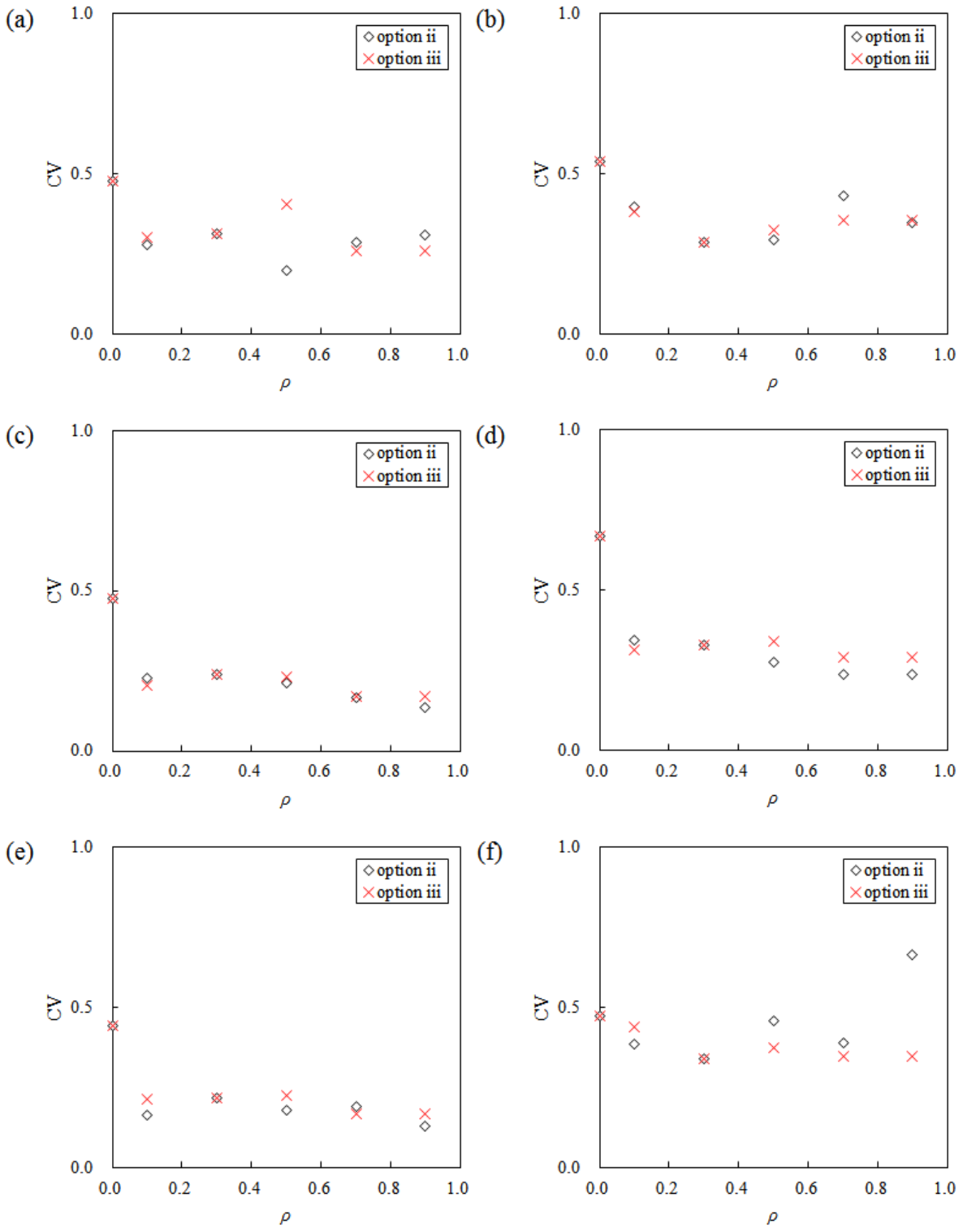

Figure 2-23 Distribution of CV of objective function values obtained by ILHSD with option ii and iii on the benchmark function (a) F7, (b) F8, (c) F9, (d) F10, (e) F11 and (f) F12

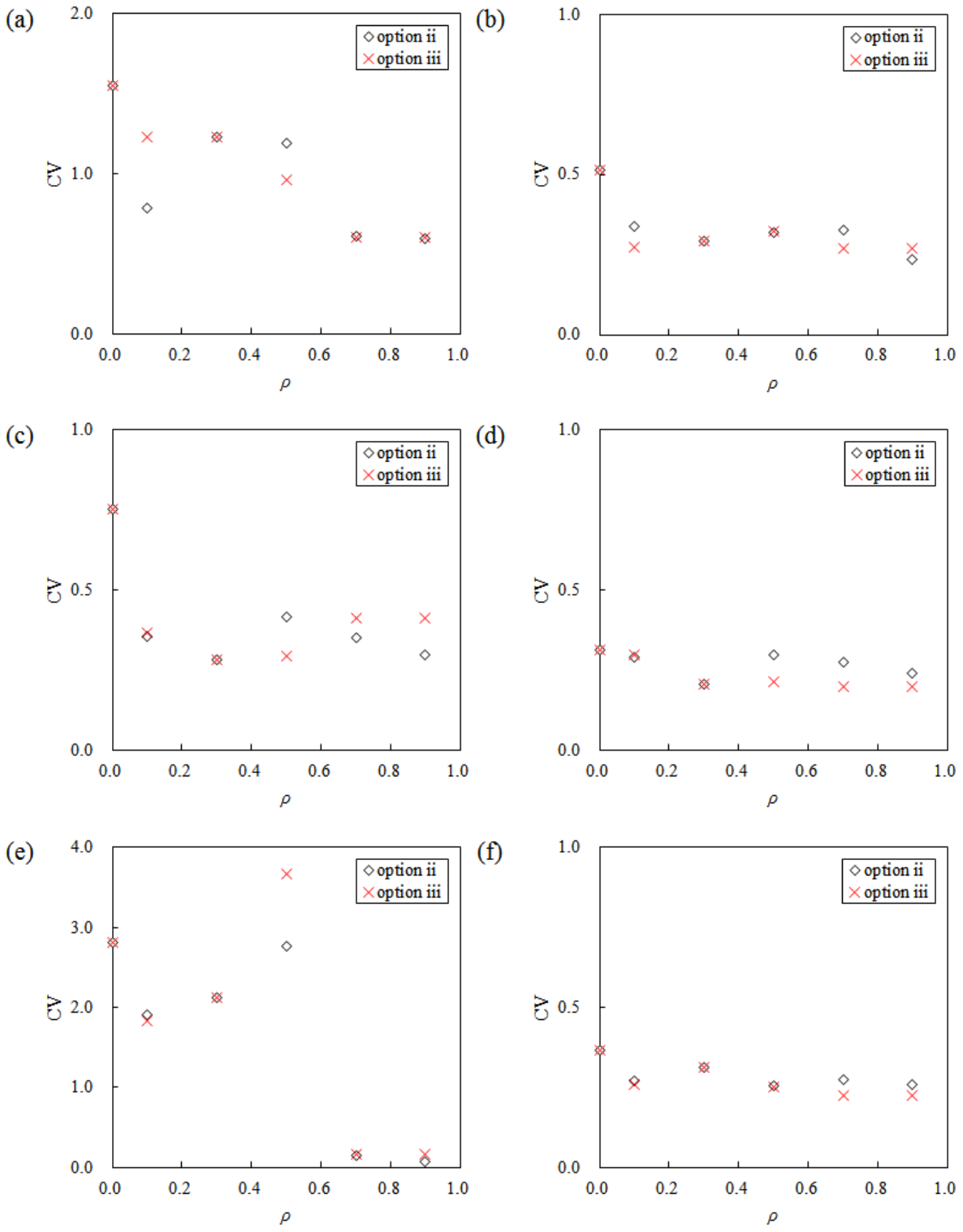

Figure 2-24 Distribution of CV of objective function values obtained by ILHSD with option ii and iii on the benchmark function (a) F13, (b) F14, (c) F15, (d) F16, (e) F17 and (f) F18

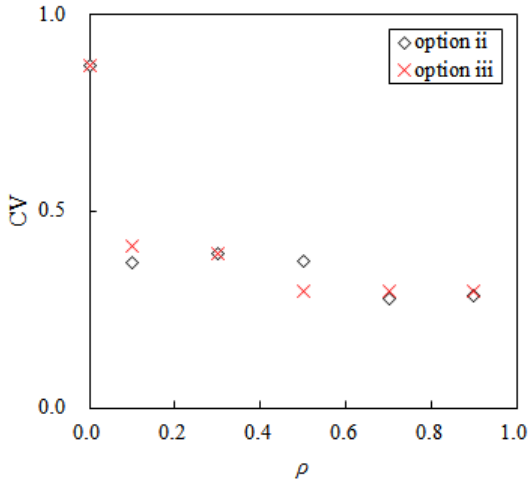

Figure 2-25 Distribution of CV of objective function values obtained by ILHSD with option ii and iii on the benchmark function F19

3. シミュレータの高速化

マルチコア CPU が一般的となった現在では，数値計算分野においても，OpenMP や MPI 等を利用したプログラムの並列化が一般的となっている．加えて，プログラムのアルゴリズムには依存するものの，GPU（Graphic Processing Unit）を用いた汎目的計算（GPGPU）は大幅な計算時間の短縮に寄与することが知られている．本章では貯留層シミュレータと大域的最適化アルゴリズムの改良による高速化を提案する．

3.1. 貯留層シミュレータの改良

本研究では貯留層シミュレータとして，米国 Lawrence Berkeley 国立研究所で開発された多次元多成分系シミュレータ TOUGH2（Pruess *et al*., 1999），及び CO_2 地中貯留用のモジュール ECO2N（Pruess., 2005）を利用した．TOUGH2 は多孔質媒体内の流体や熱の移動について計算するために開発されたオープンソースのシミュレータである．用途に応じた複数の EOS(Equation of State)モジュールが開発されており，CO_2 地中貯留をはじめ，地熱発電，放射性廃棄物の地層処分の数値シミュレーション等に広く用いられている．ECO2N モジュールはその中の 1 つであり，水-NaCl-CO_2 系の流動シミュレーションが可能である．プログラム言語はともに Fortran77 を使用している．過去の研究において，TOUGH2/ECO2N のアルゴリズムの一部が GPU 化されており，最新の GPU を用いることで 25000 グリッドの貯留層モデルで，最大で 18.5 倍の高速化を達成している(Table 3-1).

Table 3-1 Performance ratio of TOUGH2/ECO2N GPU version

TOUGH2 version [-]	GPU name [-]	Cal. Time [sec]	Performance ratio [-]
CPU ver.4.2	-	14400	1.00
GPU ver.1.2	Tesla C1060	1610	8.98
GPU ver.1.2	Tesla K40c	871	16.6
GPU ver.1.2	GeForce 780Ti	783	18.5

本研究で TOUGH2/ECO2N に施した改良点を以下に示す．プログラミング言語は Fortran90, CUDA Fortran を使用した(新井, 1998; 青木&額田, 2009; Sanders & Kandrot, 2010).

- タイムステップ幅の更新方法の変更（収束性の向上・高速化）
- 浸透率モデルの追加（汎用化）
- GPU 用アルゴリズムの改良（汎用化・高速化）
- ソルバーの改良（高速化・汎用化）

以降の節ではタイムステップ幅の更新方法の変更と，浸透率モデルの追加を中心に述べる．

3.1.1. タイムステップ幅の更新方法の変更

TOUGH2/ECO2N では，タイムステップ幅は初期値の 2 倍，もしくは 1/4 倍で計算される．CO_2 地中貯留において，CO_2 圧入初期はグリッドの圧力，CO_2 飽和度が大きく変化するため，タイムステップ幅は小さく設定する必要がある．また，計算終了時は変化量が小さいため，タイムステップ幅が大きくても計算される値への影響は小さい．TOUGH2/ECO2N の既存の機能では，これらが十分に考慮できなかったため，Aziz & Settari (1979)を参考にタイムステップ幅の計算方法の改良を行った．

n 回目のタイムステップ後，それぞれの主変数の変化量の最大値は以下の式で表される．

$$\Delta x^n_{m,\max} = \max_{ijk}\left\{\left|\Delta x^n_{m_{ijk}}\right|\right\} \tag{34}$$

ここでΔx^n_mは n 回目のタイムステップにおける m 番目の主変数の変化量であり，$\Delta x^n_{m,\max}$はΔx^n_mの最大値とする．一般に，地下空間における多相流の数値シミュレーションでは圧力とガス飽和度が主変数となることが多く，ECO2N では，圧力，温度，NaCl の質量分率と，CO_2 の質量分率もしくはガス飽和度が用いられている．

それぞれの主変数の変化量に対して，タイムステップ幅を以下のような式で計算する．

$$\Delta t_m = \Delta t^n \frac{\Delta x_{m,\lim}}{\Delta x^n_{m,\max}} \tag{35}$$

Δt^n は n 回目のタイムステップ幅で，$\Delta x_{m,\lim}$は m 番目の主変数の許容変化量である．

最小のΔt_mを n+1 回目のタイムステップ幅として採用する．

$$\Delta t^{n+1} = \min_m\{\Delta t_m\} \tag{36}$$

n+1 回目のタイムステップ後，n+1 回目における m 番目の主変数の最大値は，以下の条件を満たさなければならない．

$$\Delta x^{n+1}_{m,\max} \le C_m \cdot \Delta x_{m,\lim} \tag{37}$$

ここで，C_m は m 番目の主変数において一定である（>1）．

もし式(37)が満たされない場合，n+1 番目のタイムステップ幅は以下の式を使って再計算される，

$$\Delta t_m = \Delta t^{n+1} \frac{\Delta x_{m,\lim}}{\Delta x^{n+1}_{m,\max}}. \tag{38}$$

タイムステップ幅は式(36)によって再定義される．

以上のような改良を加えることにより，約 2 倍の高速化と収束性向上を実現した．

3.1.2. ソルバーの改良

本研究室で改良を行った TOUGH2 では，ソルバーに Cuthill-Mackee（CM）オーダリングア

ルゴリズムを用いることで，TOUGH2 内での並列化を行っている．CM オーダリングはグリッドを依存性の無いグループにレベル分けして，要素番号を付け直し並列処理を行なう手法である．CM オーダリングの手順を Figure 3-1 に示す．

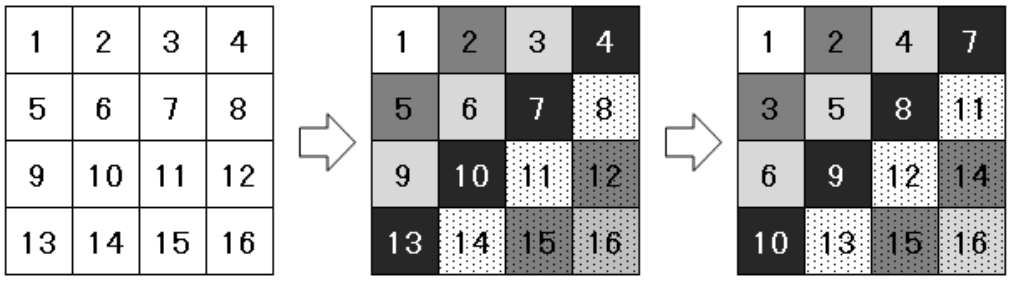

Figure 3-1 Process of Cuthill-Mackee Ordering

接するグリッド数が一番少ないグリッド，ここでは要素番号 1 をレベル 1（白）とし，次にレベル 1 に接するグリッドをレベル 2（濃灰），レベル 2 に接するグリッドをレベル 3（薄灰），同様にレベル 4（黒），レベル 5（白点），レベル 6（濃灰点），レベル 7（薄灰点）というようにレベル分けをする．レベルの低いグリッドから要素番号を付け直し、同じレベルのグリッドを並列に計算する．

CM オーダリングの他にもマルチカラーオーダリングというアルゴリズムも存在する．TOUGH2 では，CM オーダリングがマルチカラーオーダリングと比較して 1 割程度計算時間を短縮できることが確認されている．しかしながら，既存の CM オーダリングでは，貯留層モデル内でグリッドの大きさが変化する場合に，正しくオーダリングできないという問題があり，Reverse Cuthill-Mackee（RCM）オーダリングアルゴリズム（村田ら, 1991）を導入し解決をはかった．RCM オーダリングの手順を以下に示す．

1. CM オーダリングを行う．
2. 各要素に隣接する要素数を次数とし，最小次数の要素のレベルを 1 とする．
3. レベル k-1 の要素に隣接する要素をレベル k として，全ての要素に対してレベルが決定するまで行う．同レベルに属する要素はデータ依存性がないこととする．
4. レベルの順に従って，再番号付けを行う．

要素内でデータ依存性がある場合に，レベルを決定し，再番号付けを行うことでデータ依存性のない要素を先に計算することができる．これにより，TOUGH2 が扱う任意のリザーバーモデルについて計算を行うことが可能になった．

3.1.3. 相対浸透率モデルの追加

残留トラップ量は，一般的に相対浸透率のヒステリシスによって表される．相対浸透率にヒステリシスが存在する場合，CO_2 が塩水に飽和された孔隙に浸入する過程（Drainage 過程）と塩水が浸入する過程（Imbibition 過程）において，相対浸透率カーブは一致しない．以下に，Drainage 過程が累乗則，Imbibition 過程が Land のモデルに従うときの相対浸透率カーブを示す（Figure 3-2）．最大残留ガス飽和率 $S_{gr,\max}$ が分かっているとき，Imbibition 過程における相対浸透率曲線は，Drainage 過程の相対浸透率曲線から幾何学的に求めることが出来る．

あるガス飽和率 S_g において CO_2 が存在する孔隙内に塩水が浸入した場合，つまり，Drainage 過程から Imbibition 過程への変化が起こった場合，この時の残留ガス飽和度は以下の式で表される．

$$S_{gr} = \frac{S_g}{1+CS_g}. \tag{39}$$

ここで，C は Land の係数であり，

$$C = \frac{1}{S_{gr,\max}} - \frac{1}{1 - S_{gr}}, \tag{40}$$

で与えられる．

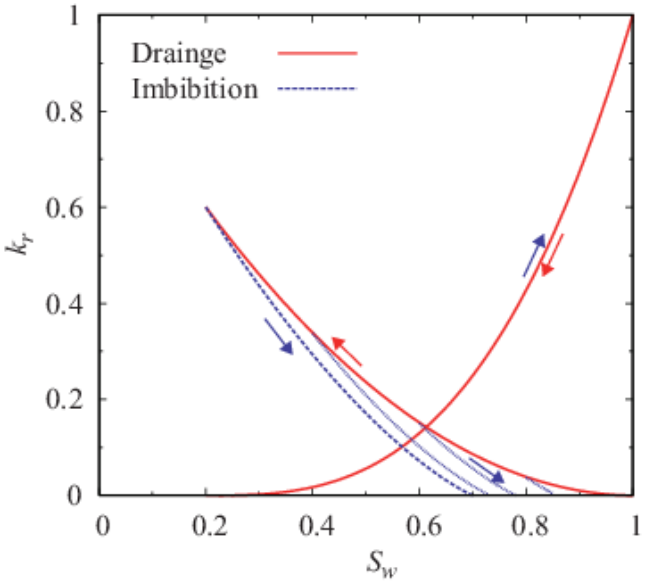

Figure 3-2 Relative permeability model

Drainage process: Power low model (S_{lr0} = 0.2, S_{gr0} = 0, k_{rw} = 1, k_{rg} = 0.6, N_w = 3, N_g = 2)

Imbibition process：Land's model ($S_{gr,\max}$ = 0.3)

変更前後での相対浸透率モデルを以下に示す．項目が 2 段になっているものは，上段が Drainage 過程，下段が Imbibition 過程におけるモデルを表す．また，GPU 版の TOUGH2/ECO2N では，相対浸透率モデルに関するサブルーチンを GPU 化しているため，これらモデルは CUDA

Fortran により記述している.

Table 3-2 Relative permeability models used in TOUGH2/ECO2N

irp	Name or Discription of models	
	Before modification	After modification
1	Linear function (Default)	Linear Function (Default)
2	Model of Pichens et al. (Default)	Model of Pichens et al. (Default)
3	Corey’s curves (Default)	Corey’s curves (Default)
4	Grant’s curves (Default)	Grant’s curves (Default)
5	Power law model Land's hysteresis model	All phases perfectly mobile (Default)
6	Power law model Killough's hysteresis model	functions of Fatt and Klikoff (Default)
7	van Genuchten-Mualem model Land's hysteresis model	van Genuchten-Mualem model (Default)
8	van Genuchten-Mualem model Killough's hysteresis model	function of Verma et al. (Default)
9	-	Power law model Land's hysteresis model
10	-	Power law model Killough's hysteresis model
11	-	van Genuchten-Mualem model Land's hysteresis model
12	-	van Genuchten-Mualem model Killough's hysteresis model
13	-	-
14	-	-
15	Table data (RELP.CSV)	Table data (RELP.CSV)

3.2. 最適化プログラム ILHS の並列化

OpenMP はメモリを共有する同一マシン内の並列化，MPI はメモリを共有する必要のない複数マシンを利用する並列化に適している．多点探索法では，各サンプルセットに対する目的関数の計算は独立であり，それぞれのプロセスはメモリを共有する必要がない（Figure 3-3）．そこで本研究では，MPI を利用した並列化を行い，その効果を確認した．

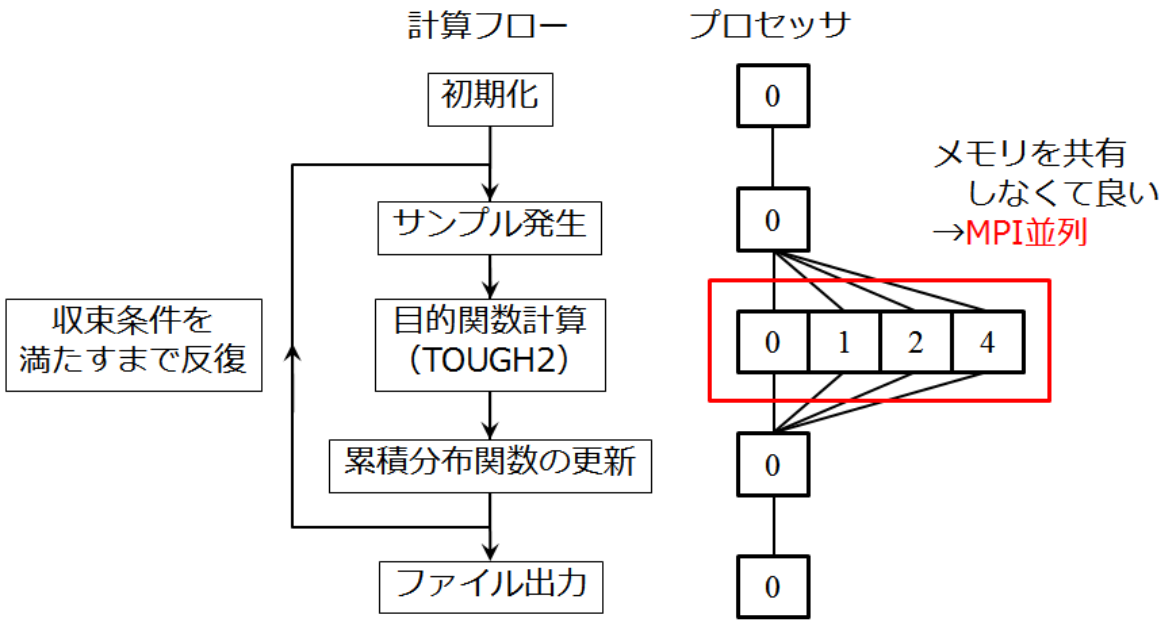

Figure 3-3 Concept of MPI parallelization of global optimization program

MPI 並列による高速化率を Table 3-3 と Figure 3-4 に示す．並列数が 10 まではほぼ線形に高速化率が上昇するものの，18 を超えると徐々に計算速度が飽和することが分かる．これは，メモリ転送性能の最も低い L3 キャッシュがボトルネックとなって，データのロード，ストア待ちが生じているためと考えられる．このことが正しければ，L3 キャッシュは最大 45MB であるため，1 回の貯留層シミュレーションに 4 から 5MB の L3 キャッシュが必要である．

Table 3-3 Performance ratio of TOUGH2/ECO2N using GPU and MPI parallelization

Case	1	2	3	4	5	6	7	8
CPU Process	1	4	9	18	35	23	17	11
Hyper-Q	0	0	0	0	0	2	3	4
Total CPU Process	1	4	9	18	35	27	23	19
Total Process	1	4	9	18	35	35	35	35
Total Time [sec]	94469	27415	12717	10559	10166	8172	7894	9350
Time for 1 Run [sec]	2699	783	363	302	290	233	226	267
Acceleration Ratio	1.00	3.45	7.43	8.95	9.29	11.56	11.97	10.10

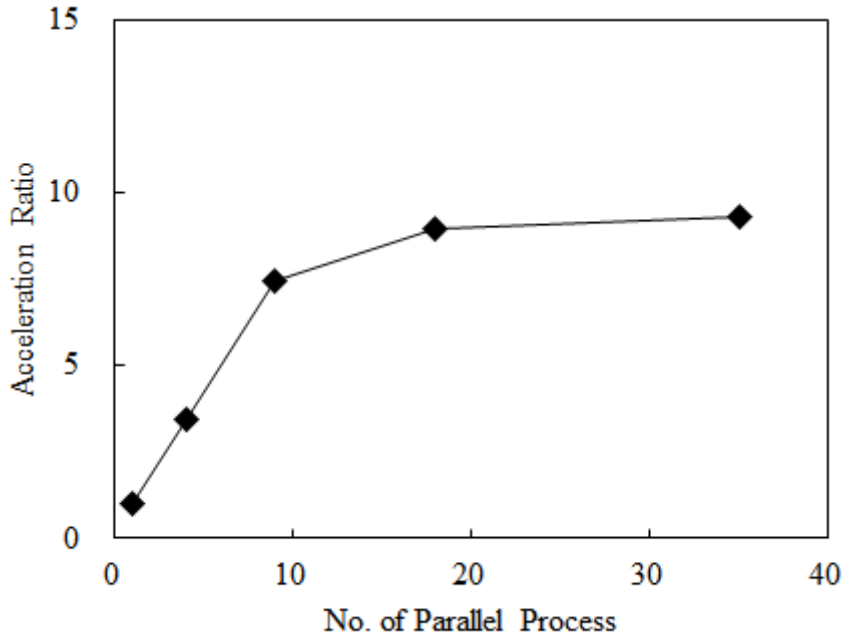

Figure 3-4 Acceleration ratio of parallelization using only CPU

次に，GPU による高速化を行った．Kepler 以降の GPU では，複数の CPU コアから送られた処理を 1 つの GPU が同時に使用できる機能，Hyper-Q を備えている．Hyper-Q を数値シミュレーションで利用した例は少なく，貯留層シミュレーションでは未報告である．石油開発においては弾性波探査のシミュレーションが 1 例報告されているのみで（Xue *et al.*, 2015），最大並列数が 8，高速化率が 18 に達している．Figure 3-5 に Hyper-Q による高速化率を示す．グラフから並列数が 4 以降で高速化率は収束していることが分かる．TOUGH2 の貯留層シミュレーションでは，GPU のメモリバンド幅が律速になることが分かっており，Hyper-Q を使用した場合でもメモリバンド幅，つまりデータの転送速度が限界に達していると考えられる．結論として，貯留層シミュレーションにおける Hyper-Q 機能の効果は限定的であるといえる．

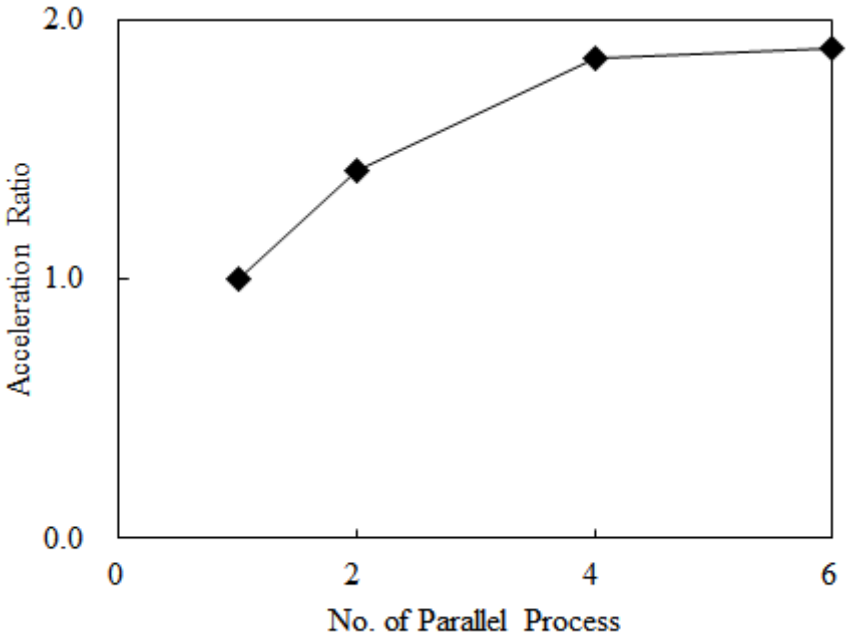

Figure 3-5 Acceleration ratio of Hyper-Q parallelization using GPU

最後に，MPI 並列と GPU の Hyper-Q を併用した場合の高速化率を示す（Figure 3-6）．並列数が 14，Hyper-Q が 2 のとき，約 12 倍の高速化を達成した．前述のように，更なる高速化には，CPU や GPU の処理速度よりも，メモリ転送性能に着目した改良を行うべきであり，例えば，貯留層シミュレータの更なる GPU 化，キャッシュチューニングが効果的であると考える．

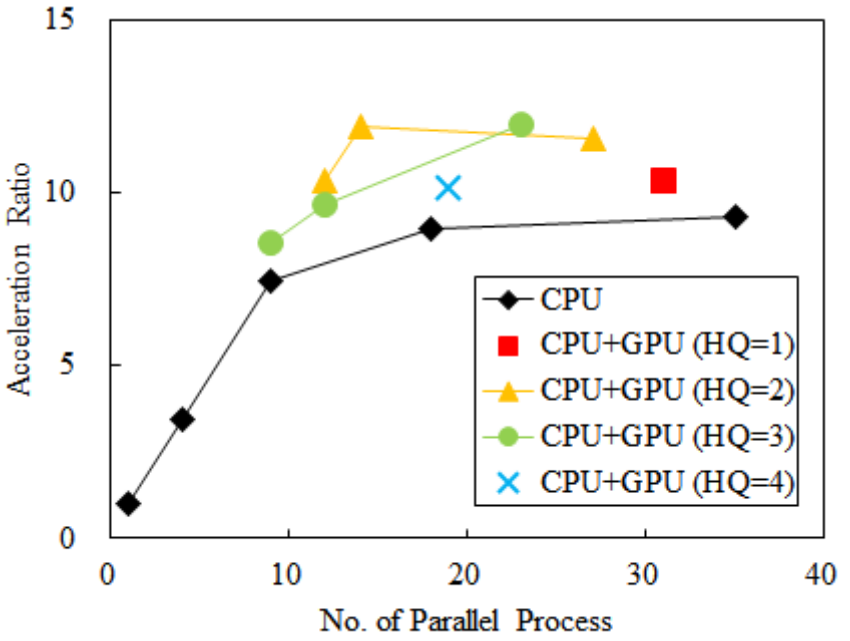

Figure 3-6 Acceleration ratio of parallelization using CPU with GPU

4. 坑井配置の最適化問題への適用

地下深部塩水層への 2 本の垂直坑井からの CO_2 圧入(ケース 1)と 2 本の水平坑井からの CO_2 圧入（ケース 2）について最適化計算を行うことで，提案手法の有効性を検証した.

2 つの最適化問題に共通する事項を以下に述べる．地層モデルは上下を不透水層に囲まれた不均質な地層を設定した（Figure 4-1）．水平方向に 2km×2km，深度 800m から 1300m の領域に年間 78.8 万 t の CO_2 を 30 年間圧入する．水平境界は圧力が一定とし，領域外への CO_2 の流出は許す. 坑底圧が設定した最大値に達した場合，CO_2 の圧入を停止する. 孔隙率は 0.3 で一定，浸透率は不均質とし，平均浸透率が 100md となるように設定した．相対浸透率は，Drainage 過程では van Genuchten モデル，Imbibition 過程では Land モデルに従うものとし，不動水飽和率を 0.20，臨界ガス飽和率を 0.05，モデル乗数 0.88 ，最大残留ガス飽和率を 0.30 とした.

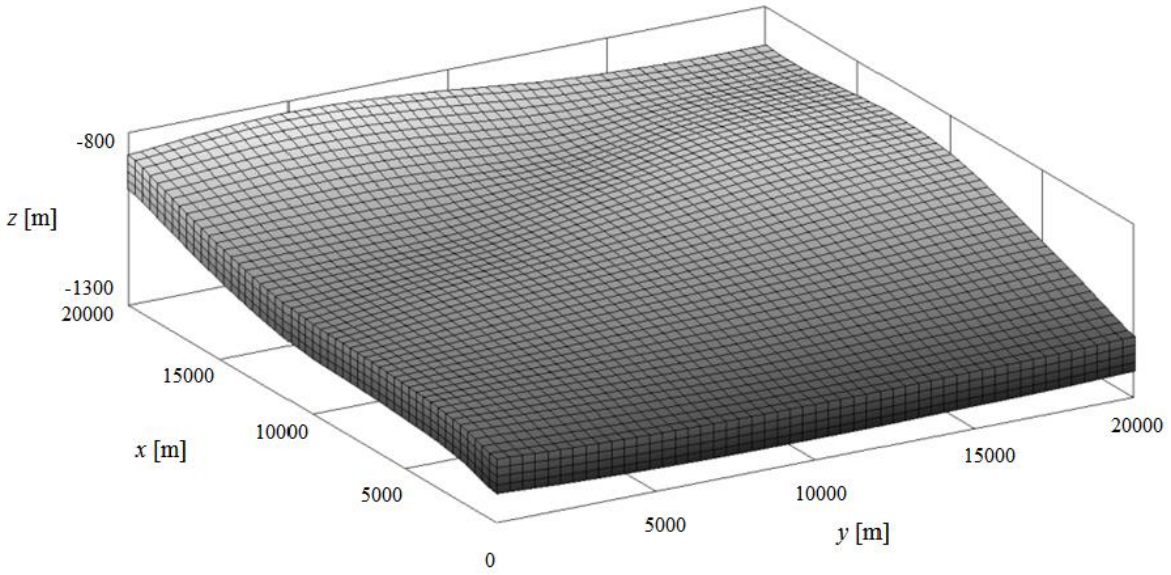

Figure 4-1 Over Head View of Reservoir Model

目的関数は圧入開始 100 年後における可動 CO_2 の質量割合とする：

$$g = \frac{m_{\text{movable}}}{m_{\text{total}}} = 1 - \frac{m_{\text{residual}} + m_{\text{dissolved}}}{m_{\text{total}}}. \tag{41}$$

m_{total}は設定された圧入総量であり，m_{movable}，m_{residual}，$m_{\text{dissolved}}$はそれぞれ圧入から 100 年後の可動 CO_2，残留トラップ CO_2，溶解トラップ CO_2 の質量を表す.

貯留層シミュレーションには，TOUGH2/ECO2N（Pruess *et al.*, 1999 : Pruess, 2005）を用いた. 計算コードは一部 GPU を使用するように改良されており，CPU のみを用いる場合よりも短時間での計算が可能である（Ishizawa et al., 2013）.

TOUGH2/ECO2N では，各タイムステップで収束までに要した計算回数を基に，次のタイムス

テップ幅を変更するアルゴリズムが用いられている．このアルゴリズムを，圧力やガス飽和度といった主変数の変化量に応じて，タイムステップ幅を変更できるように修正した．この修正により，主変数の変化量が大きい圧入開始直後では収束性の向上が，変化量の小さい圧入終了後は計算時間の短縮が可能となっている（Tanaka et al., 2013）．

以下の基準のどちらかを満たしたとき，解探索を終了する．

i. 繰り返し計算の回数 n_{iter} が $n_{\mathrm{iter,\,max}}$ に達したとき

ii. 全ての主変数 x_{j} に対して，$d\eta(n,j) \le d\eta_{\min}(n,j)$を満たすとき

基準 i は一般的に設定される基準で，基準 ii は本研究で新規に導入した基準であり，それぞれ 2.3.2 での基準 I，III に相当する．本計算例では$n_{\mathrm{iter,max}} = 100$，$d\eta_{\min}(n,j) = 0.01$とした．水平坑井の方向は，1, 2 の 2 値で表されるダミー変数として扱う．ダミー変数は連続変数や離散変数とは異なる収束挙動を示すため，$d\eta_{\min}(n,j) = 0.05$と設定している．次節以降では，基準 i のみを設定した場合と，基準 i に加えて基準 ii を設定した場合について目的関数の値を比較し，基準 ii を用いることの有効性を検証する．

4.1. 2 本の垂直坑井からの CO_2 圧入（ケース 1）

ケース 1 では，2 本の垂直坑井から CO_2 を貯留層に圧入することを考える．主変数ベクトル $\boldsymbol{X}$ は，

$$\boldsymbol{X} = [x_{\mathrm{w,1}}, y_{\mathrm{w,1}}, x_{\mathrm{w,2}}, y_{\mathrm{w,2}}, q_{\mathrm{inj,1}}]^T, \tag{42}$$

と表される．ここで，$x_{\mathrm{w,i}}$, $y_{\mathrm{w,i}}$及び$q_{\mathrm{inj,i}}$はそれぞれ坑井 i の x, y 座標と CO_2 の圧入レートである．本計算例では坑底圧が設定した最大圧に達しない限り CO_2 圧入量$q_{\mathrm{inj,total}}$は一定の値をとる．従って，坑井 2 の圧入レート$q_{\mathrm{inj,2}}$は，

$$q_{\mathrm{inj,2}} = q_{\mathrm{inj,total}} - q_{\mathrm{inj,1}}, \tag{43}$$

で表される従属変数となる．

ILHS で発生させるサンプル数n_{pop}は事前に行った感度分析から決定した．主変数 $\boldsymbol{X}$ の次元を d としたとき，サンプル数が d から $2d$ の間で効率的な解探索が行われたことから，本ケースでは$n_{\mathrm{pop}} = 2d = 10$としている．同様に感度分析から，Eq.(4)中の重み付けの指数は$\gamma = 0.839$を用いた．関数評価回数$FES\,(= n_{\mathrm{pop}} \times n_{\mathrm{iter,max}})$ は最大で 1000 回である．

独立した 5 回の試行について，得られた目的関数の値の変化を Figure 4-2 に示す．5 回の試行中 3 回で，条件 i よりも先に条件 ii を満たした（Figure 4-2 の Run 2, 4, 5 上の×印）．そのときの計算ステップ数は，試行 2, 4, 5 においてそれぞれ$n_{\mathrm{iter}} = 70, 67, 61$であった．試行 4, 5 では，基準 ii を満たして以降も計算終了まで目的関数の値は更新されていない．それに対して試行 2 では，エントロピーの収束以降に目的関数の値が更新されている．しかしながら，試行 2 では

n_{iter}=100 においても目的関数の値は 5 つの試行中で最も高いため, 最適解を導く可能性は低く, 逆に早い段階で最適解の候補から排除できているといえる. 前述の通り, 繰り返し計算の進行に伴い, 解探索の集中化, つまり最適解近傍の局所探索が多く行われるようになる. ただし ILHS はランダムサンプリングに基づく手法であるため, 必ずしも効率的な局所探索が行われるわけではない. そのため, 解探索が十分に進み, サンプル区間がほぼ固定化されていくに従い, 目的関数の大幅な更新は徐々に難しくなる. 結果として, 最適解となる可能性が低い試行は収束判定基準に則って打ち切り, 異なる初期条件で新たな試行を行う方が全体として効率的である.

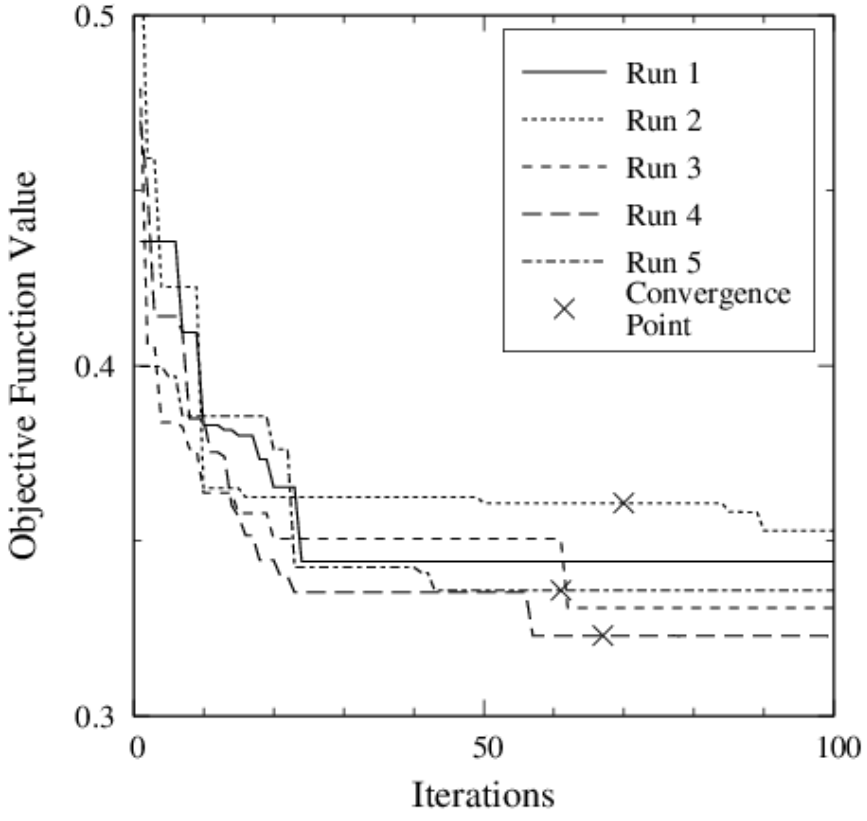

Figure 4-2 Convergence behavior of objective function value with criteria depend on normalized entropy in case 1

目的関数の値が最も小さい試行 4 について, 正規化した主変数, 及び$d\eta(n,j)$の値の変化を示す（Figure 4-3）. $d\eta(n,j)$の値はサンプル区間の幅が変化することで大きく上下するため（Figure 4-3a）, 最小値のみを別に抽出した（Figure 4-3b）. 繰り返し計算初期で$d\eta(n,j)$は急激に減少し, n_{iter} = 30以降は徐々に 0 に近い値に収束することが分かる. この傾向は他の試行でも同様であった. 主変数（Figure 4-3c）, 及び目的関数の値（Figure 4-2）に着目すると, $n_{iter} \geqq 67$で一定値を保っていることが分かる. 従って, 2.3.2 で述べた収束判定条件 II の様に, 目的関数の値から解探索の収束を判定することが考えられる. しかしながら, 目的関数の値は$n_{iter} < 67$でも 10 ス

テップ以上更新されておらず，必ずしも解探索の収束の度合いを表しているとはいえないため，目的関数の値は収束判定の基準としてはやはり適切ではない．加えて，試行 4 に限って$n_{\mathrm{iter}}=300$まで計算を行ったところ，これ以上の目的関数値の改善は見られなかった．このことからも，正規化エントロピーが解探索の収束を判定する指標となり得ることが示唆される．

条件 ii 導入前後で，FES と目的関数の値の変化を Table 4-1 に示す．全体として計算回数は 80%程度まで減少し，得られた目的関数値の差は 1%未満であった．本ケースの計算結果から，提案した基準は最適化計算の収束を判定する際に有効であると言える．

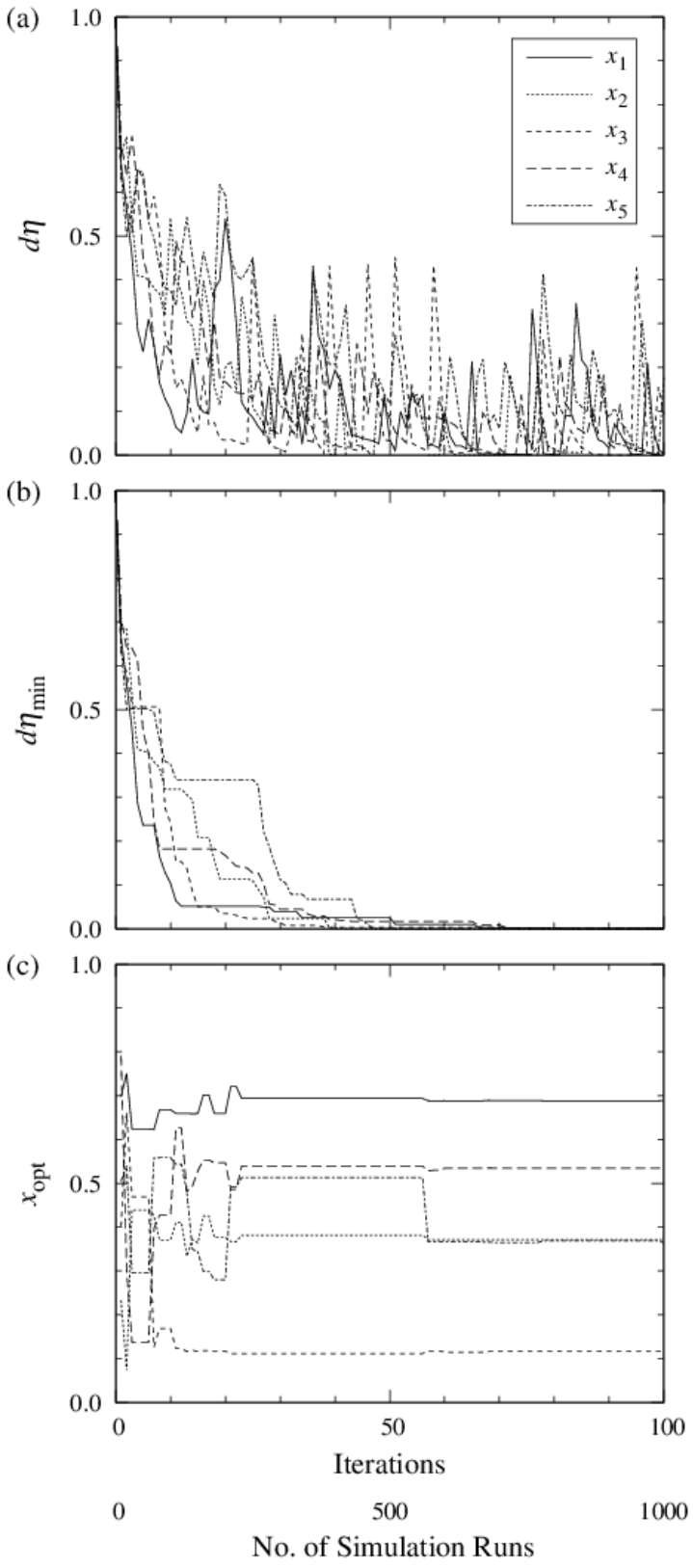

Figure 4-3 Convergence behavior of difference between normalized entropy and normalized BEF for five primary variables: (a) Values in each iteration step, (b) Minimum values in each iteration step and (c) Normalized primary variables

Table 4-1 The Number of function evaluations and optimum with the presence or absence of criterion

II in case 1

Run	Criterion I				Criterion I and II			
	FES		Optimum		*FES*		Optimum	
1		1000		0.34409		1000		0.34409
2		1000		0.35293		700		0.36073
3		1000		0.33083		1000		0.33083
4		1000		0.32270		670		0.32290
5		1000		0.33582		610		0.33582
	Total	5000	Best	0.32270	Total	3980	Best	0.32290

4.2. 2本の水平坑井からの CO_2 圧入（ケース 2）

次に，2 本の水平坑井から貯留層に CO_2 を圧入し，1 本の垂直坑井から塩水を生産した場合の坑井配置の最適化問題について考える（ケース 2）．水平坑井と帯水層中の塩水を生産することで，貯留層内の圧力低減やトラップ量の増加が期待できる．

主変数ベクトル $\boldsymbol{X}$ は，

$$\boldsymbol{X} = [x_{\mathrm{wi},1}, y_{\mathrm{wi},1}, x_{\mathrm{wi},2}, y_{\mathrm{wi},2}, x_{\mathrm{wp}}, y_{\mathrm{wp}}, q_{\mathrm{inj},1}, i_{\mathrm{dir},1}, i_{\mathrm{dir},2}]^T, \tag{44}$$

と表される．ここで，$x_{\mathrm{wi},i}, y_{\mathrm{wi},i}$ は圧入井 i の始点の x, y 座標，$x_{\mathrm{wp}}, y_{\mathrm{wp}}$ は生産井の x, y 座標，$i_{\mathrm{dir},i}$ は圧入井 i の方向である．$i_{\mathrm{dir},i}$ は 1 または 2 の値を取るダミー変数とし，それぞれ x，y 方向を意味する．サンプル数 $n_{\mathrm{pop}} = 2d = 18$，$FES = 1800$ とした．

前節と同様に，ILHS を用いて 5 回の独立試行について最適化計算を行った．目的関数の値の変化を Figure 4-4, 条件 ii 導入前後での FES と目的関数値の変化を Table 4-2 に示す．Table 4-2 から，70%程度の計算回数で最適解を得ることに成功している．加えて，Figure 4-4 より，エントロピー収束後はそれ以上，目的関数の値は改善されておらず，エントロピーが解探索の収束をうまく表していると言える．

本ケースではケース 1 と比較して主変数の数が増加したものの，連続変数について正規化エントロピーの挙動には大きな違いはなく，提案手法は最適化問題の次元に関わらず有効である．また，本ケースは 2 つのダミー変数を含んでいる．ダミー変数のエントロピーは，連続変数のそれと比較して収束までに時間を要するものの，$d\eta_{\min}(n, j)$ を適切に設定すれば連続変数と同様に収束を判定することが可能である．

最後に，得られた最適解に関して，CO_2 の状態別経時変化を Figure 4-5 に示す．ケース 2 の目的関数の値はケース 1 の 20%程度であり，トラップ CO_2 量，特に溶解トラップ量が増加した．これは水平坑井の導入により，CO_2 と地層水との接触面積が増加したことによるものと考えられる．従って，安定的な CO_2 の地下貯留を目指す上で，水平坑井の利用は効果的だといえる．

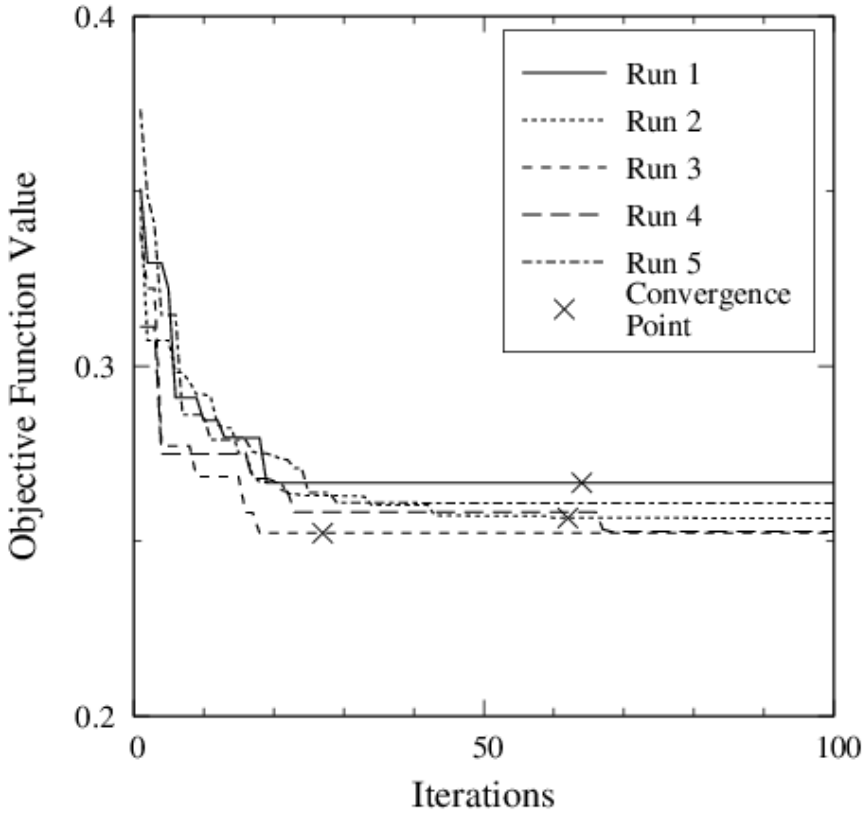

Figure 4-4 Convergence behavior of objective function value with criteria depend on normalized entropy in case 2

Table 4-2 The Number of function evaluations and optimum with the presence or absence of criterion II in case 2

Run	Criterion I				Criterion I and II			
	FES		Optimum		*FES*		Optimum	
1		1800		0.26664		1152		0.26664
2		1800		0.25647		1116		0.25657
3		1800		0.25227		486		0.25227
4		1800		0.25276		1800		0.25276
5		1800		0.26081		1800		0.26081
	Total	9000	Best	0.25227	Total	6354	Best	0.25227

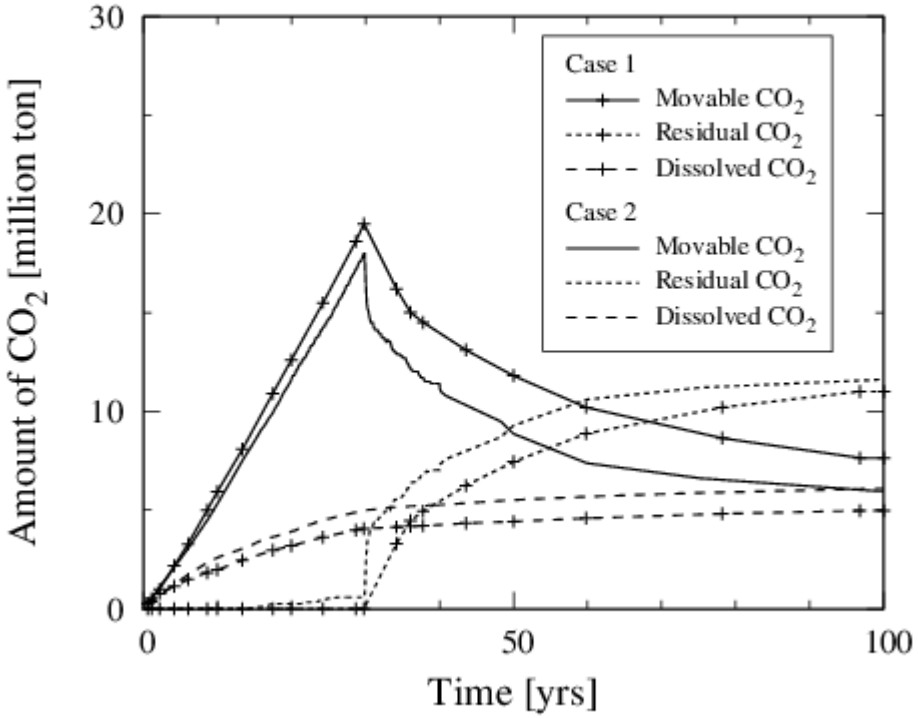

Figure 4-5 Time series data of CO_2 amount in case 1 and 2

5. ヒストリーマッチングへの適用

次元の大きい最適化問題として，岩野原実証試験を対象とした HM を行った．初めに，実証試験で観測されたデータを示す．観測データの中から最適化計算の目的関数として適切なものについて検討した．貯留層の浸透率を中心とした最適化（ケース 1），浸透率と孔隙率を中心とした最適化（ケース 2, 3）を行い，開発した大域的最適化アルゴリズムの性能と課題について検討した．

5.1. 岩野原実証試験

岩野原実証試験は国内初の CO_2 地中貯留の実証試験である．圧入対象層は，新潟県長岡地域の灰爪層（前期更新世）に卓越する，深さ約 1100m，層厚 12m の砂岩層である．貯留層温度，圧力はそれぞれ約 48℃，11MPa であり，圧入された CO_2 は超臨界状態となる．2003 年 7 月から 2005 年 1 月の期間に 1 本の垂直坑井（IW-1）から 10400t の CO_2 が圧入され，3 本の観測井（OB-2~4）によるモニタリングが継続的に行われている（Figure 5-1）．実証試験の詳細やモニタリングデータについては，地球環境産業技術研究機構（RITE）の各種報告書や外部発表に詳しい（Mito et al., 2008; Sato et al., 2011; Mito et al., 2013; Nakajima and Xue, 2013; Otake, 2013; 三戸ほか, 2008; 斎藤ほか, 2008; 薛&松岡, 2008; 薛&渡辺, 2008; 棚瀬ほか, 2008）.

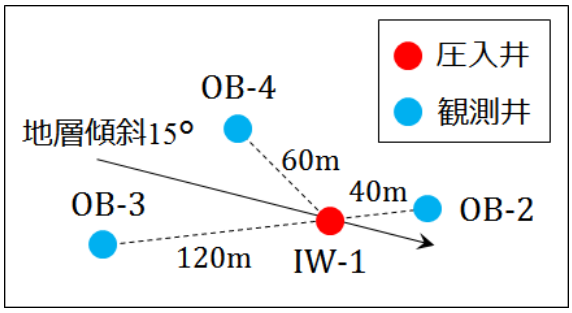

Figure 5-1 Over head view of wells in Iwanohara project (薛&松岡, 2008 より作成)

灰爪層は岩相の違いから 5 つの Zone に分けられる．各坑井での各 Zone の深度を以下に示す．

Table 5-1 Depth of zone bottom for each wells in Iwanohara project (RITE, 2005)

Zone	IW-1 [m]	OB-2 [m]	OB-3 [m]	OB-4 [m]
Zone-1	1093.0	1107.0	1073.0	1084.0
Zone-2	1105.0	1120.2	1085.0	1093.0
Zone-3	1125.0	1141.5	1105.0	1113.5
Zone-4	1131.0	1149.0	1111.0	1119.0
Zone-5	1150.0	1169.5	1129.0	1135.8

岩野原実証試験の特徴として，弾性波トモグラフィを代表とする観測項目の多さが挙げられる．圧入終了後に観測されている項目を以下に挙げる．

- 坑口・坑底での圧力・温度
- 坑内物理検層（中性子検層，音波検層，比抵抗検層）
- 坑井間弾性波トモグラフィ
- 地層流体サンプリング・分析
- 地盤微動観測

圧力・温度は圧入井 IW-1，観測井 OB-4 において観測されている（Figure 5-2）．IW-1，OB-4 の圧力計はそれぞれ，Zone-2 Top より 20m 上部の位置，Zone-2 Middle にあたる設置されている．

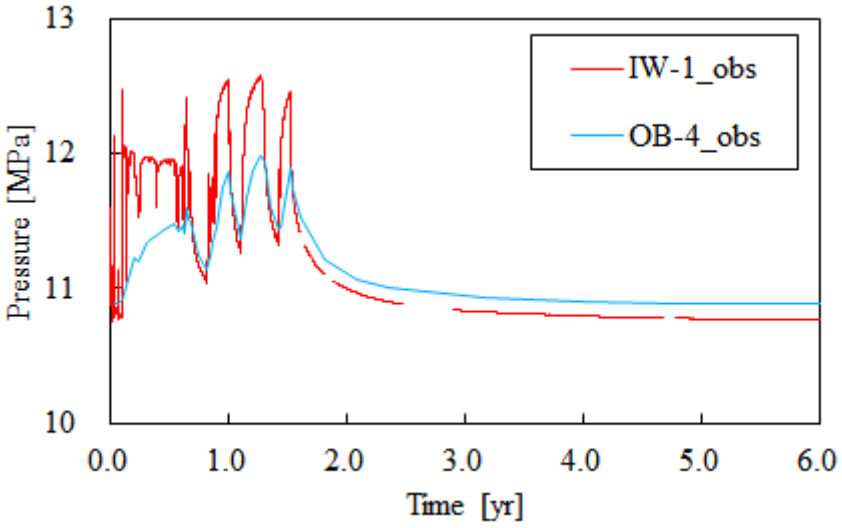

Figure 5-2 Observed bottom hole pressure at injection and observation wells in Iwanohara project

また，坑内中性子検層の観測値から下記の換算式を用いて，CO_2 飽和度を計算することが可能である（RITE, 2014）．

$$S_{CO2} = (NLP_{BL} - NLP_n)/NLP_{BL}. \quad (45)$$

NLP_n, NLP_{BL}はそれぞれ各観測での中性子孔隙率とベースラインの中性子孔隙率を示す. 各観測井への CO_2 到達時間は，中性子検層とは別に地層流体のサンプリングによって確かめられている．NLP_{BL}の値として，CO_2 到達以前のNLP_nの平均値を用いた．観測井 OB-2，OB-4 について，CO2 飽和度の経時変化をそれぞれ Figure 5-3，Figure 5-4 に示す．エラーバーは，CO_2 到達以前のNLP_nの分散である．中性子検層の測定誤差が大きく，定量的な評価は難しいものの，全体的な傾向として，どちらの観測井でも Zone-2 下部の CO_2 流動が支配的であり，CO_2 飽和度は一度上昇してから減少しているように見える．また OB-2 よりも OB-4 で CO_2 飽和度が高いようである.

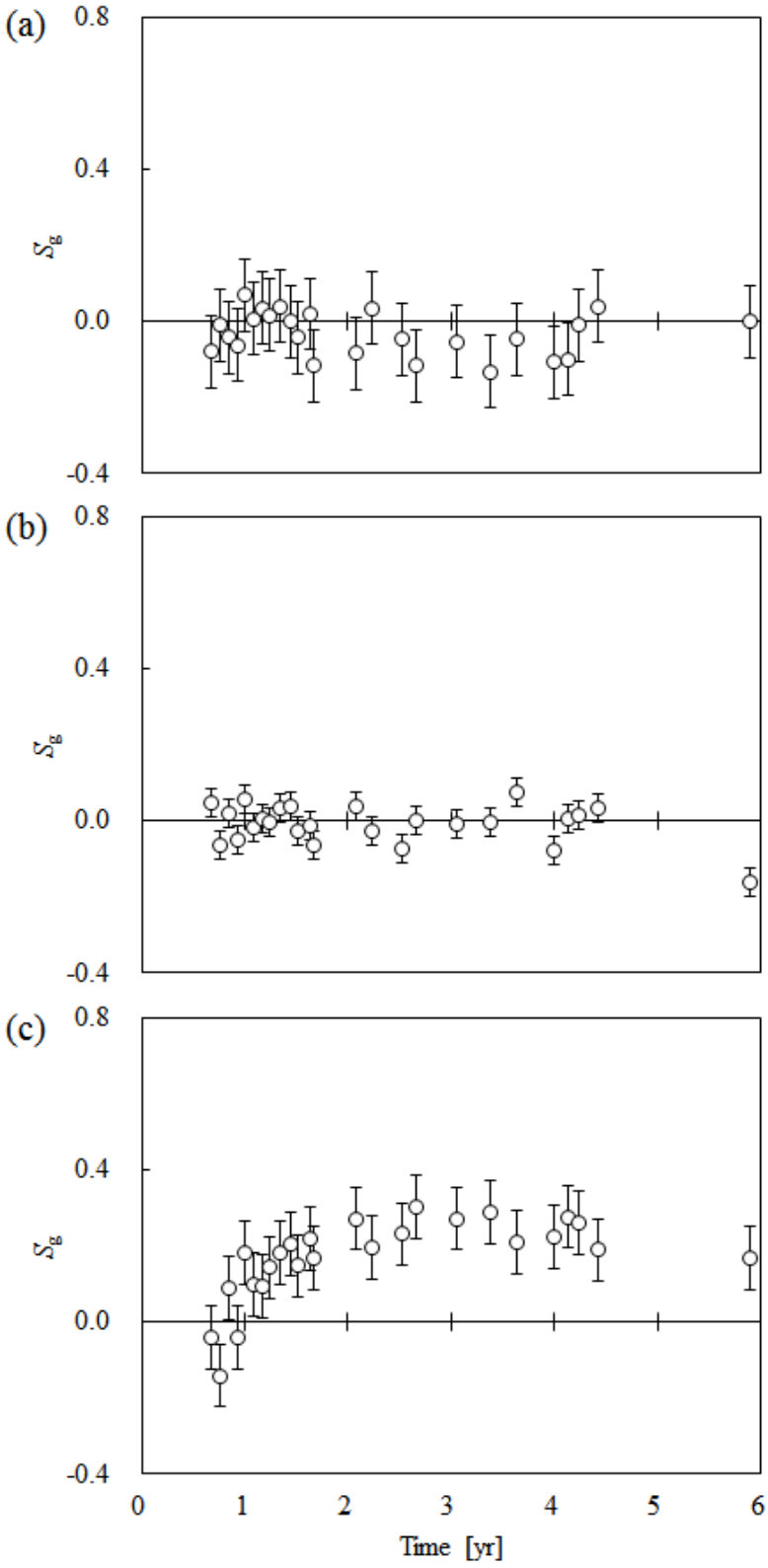

Figure 5-3 Time series of CO_2 saturation calculated using neutron logging data at OB-2 in Zone-2 (a) top, (b) middle and (c) bottom layer

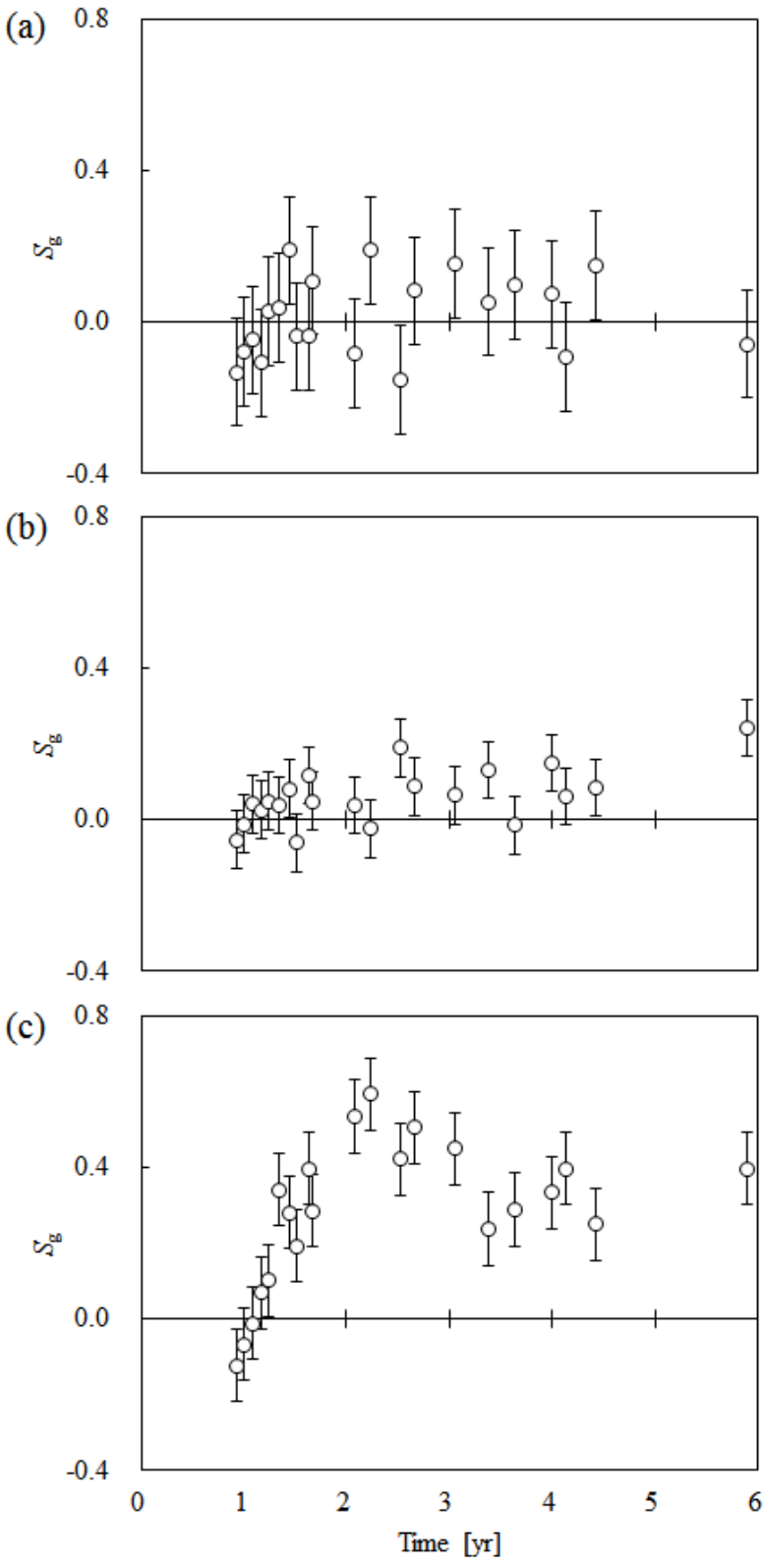

Figure 5-4 Time series of CO_2 saturation calculated using neutron logging data at OB-4 in Zone-2 (a) top, (b) middle and (c) bottom layer

以下は，RITE によって行われた手動 HM で検討された項目である（RITE, 2013）.

- Zone-2 middle 層について，絶対浸透率を 1.5 倍とした．
- 残留ガス飽和度を 0.2 から 0.25 に変更した．
- 相対浸透率曲線の形状をより実測値に近いものに変更した．
- 毛管圧力曲線を van Genuchten モデルから室内試験から得られた実測値とした．

上記の HM を行った結果を Figure 5-5 と Figure 5-6 に示す．どちらの井戸の圧力も実際より大きくなっている．加えて，個々での計算には坑底圧ではなくグリッドの圧力を用いており，IW-1 の計算値はより高くなると考えられる．一方で，観測井への CO_2 の到達時間は観測値とほぼ一致しており，CO_2 飽和度は，最大値については概ね等しい値となっている．

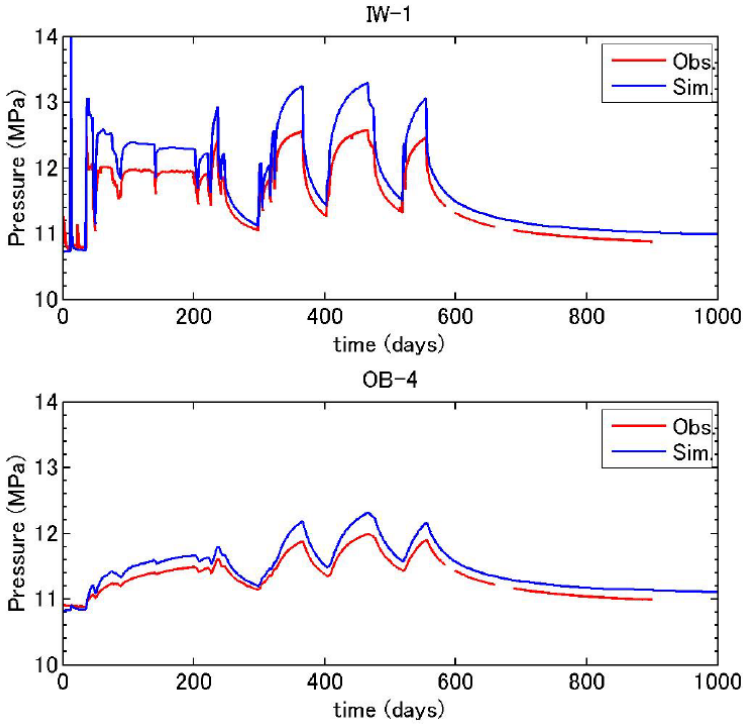

Figure 5-5 Time series of observed and calculated pressure data at injection and observation wells in Iwanohara project (RITE, 2013)

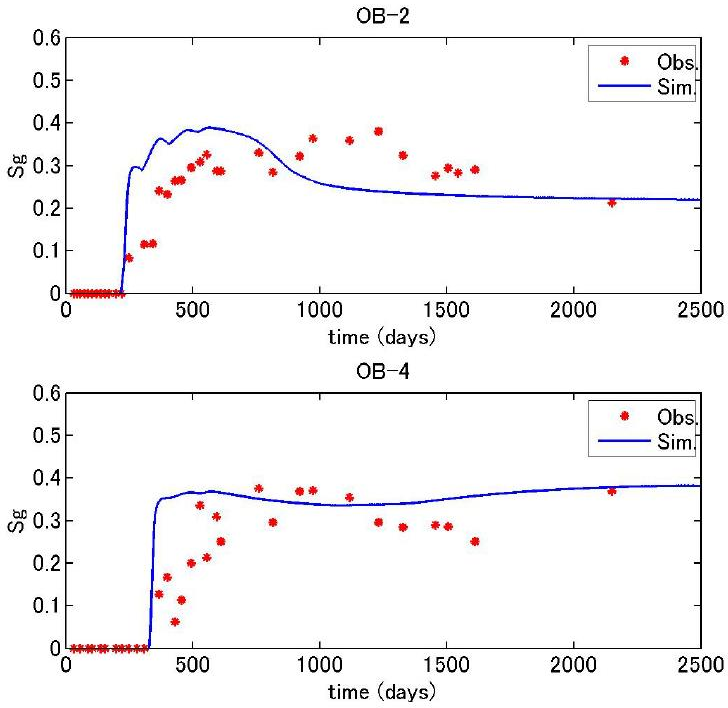

Figure 5-6 Time series of observed and calculated CO_2 saturation data at observation wells in Iwanohara project (RITE, 2013)

自動 HM については，本研究とは異なるグリッドモデルではあるものの，圧力履歴と CO_2 到達時間を目的関数として，浸透率と相対浸透率モデル，毛管圧力モデルのパラメータの最適化が行われており，最適解が観測値と良く一致することが示されている（合田，2012）．しかしながら，CO_2 飽和度に関する検討は行われておらず，また孔隙率と浸透率との間に相関はないとしている．

5.2. 問題設定

5.2.1. 貯留層シミュレーションに関する問題設定

HM に用いた問題設定を述べる．過去の研究では幾つかの種類の貯留層モデルが用いられているが，本研究では，Figure 5-2 に示した貯留層モデルを使用した．貯留層モデルは地質データを元に Petrel によって作成されている．グリッドサイズは，5m×5m，25m×25m，50m×50m の 3 種類である．Figure 5-7，Figure 5-8 にそれぞれ貯留層の各レイヤーの厚さ，及び孔隙率と浸透率の平均値，実際の圧入レートと計算に使用された簡略化された圧入レートを示す．

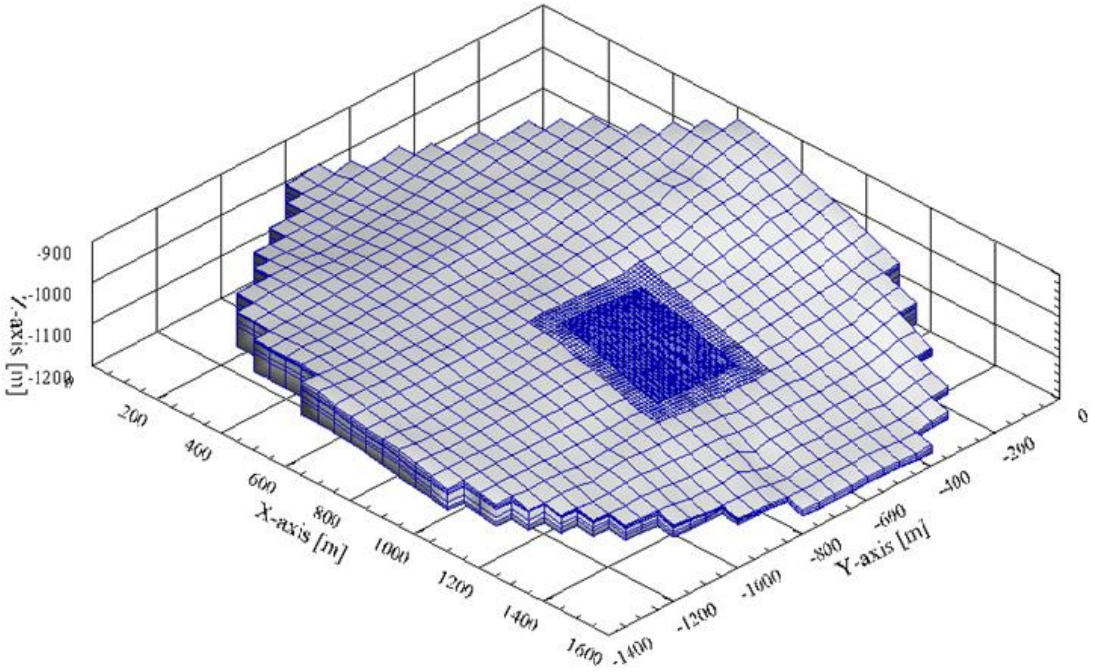

Figure 5-7 Over Head View of Reservoir Model in Iwanohara project

Table 5-2 Average value of geological data for each zone (大熊，2008)

Zone		Thickness [m]	Porosity [-]	Permeability [mD]
Zone2	Upper	5.5	0.26	3.2
	Middle	5.5	0.26	11
	Lower	1.0	0.26	1.6
Zone-3	Upper	10	0.20	0.33
	Lower	10	0.20	0.66
Zone4 & 5		25	0.23	0.46

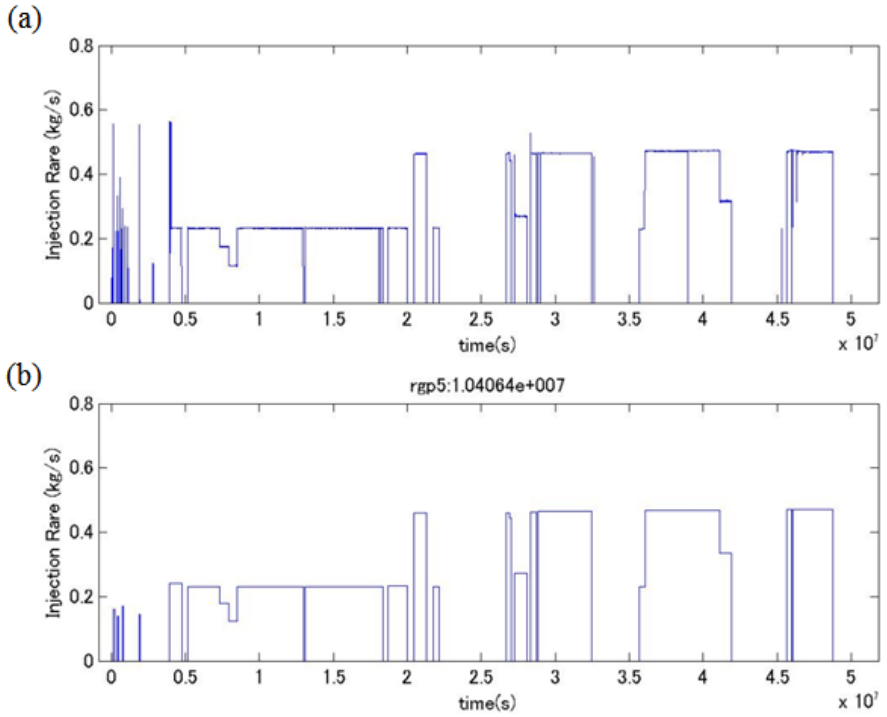

Figure 5-8 Time series of CO_2 injection rate: (a) original data and (b) simplified data

5.2.2. 最適化計算に関する問題設定

目的関数は各観測値と計算値との差の重みづけ 2 乗和で表される：

$$g = \sum_{i=1}^{5} \sum_{j=1}^{n_{\mathrm{obs,i}}} \frac{w_{\mathrm{i}}}{n_{\mathrm{obs,i}}} \left(\frac{y_{\mathrm{obs}}(i,j) - y_{\mathrm{sim}}(i,j)}{y_{\mathrm{obs}}(i,j)} \right)^2 \tag{46}$$

ここで，i = 1, 2 はそれぞれ IW-1，OB-4 の坑底圧，i = 3, 4 はそれぞれ OB-2，OB-4 への CO_2 到達時間である．添字の obs は観測値，sim は計算値であることを表す．各種変数と設定値を Table 5-3 に示す．石油ガス開発におけるヒストリーマッチングでは，坑底圧や油ガス生産量といった経時データを用いることが多い．CO_2 実証試験では圧入井と観測井の坑底圧のデータは利用可能であるものの，CO_2 圧入量は入力データであるため，他の観測データを用いる必要がある．岩野原実証試験では中性子検層による CO_2 飽和度が観測されているものの，前述のとおり観測のエラーは大きく，目的関数に利用することは難しい．本研究では IW-1，OB-4 の坑底圧，及び各観測井への CO_2 到達時間を目的関数とした．ただ OB-3 に限っては，いまだに CO_2 が到達していないため，$S_{CO2} = 0$という経時データがあるものとして計算している．収束判定基準は 4 章と同様である．

Table 5-3 Set of primary variables and parameters

i	1	2	3	4	5
Well No.	IW-1	OB-4	OB-2	OB-3	OB-4
Variable	Bottom hole pressure	Bottom hole pressure	CO_2 arrival	CO_2 saturation	CO_2 arrival
n_{obs}	45	45	1	45	1
w_{i}	0.206	0.411	0.0311	0.309	0.0431

主変数は以下の 3 ケースを考えた.

1. 孔隙率を考慮しない場合

$$\mathrm{X}=\left[\beta_1,\cdots,\beta_{28},\left(\frac{K_{\mathrm{v}}}{K_{\mathrm{h}}}\right)_1,\cdots,\left(\frac{K_{\mathrm{v}}}{K_{\mathrm{h}}}\right)_6,c\right]^T \tag{47}$$

2. 孔隙率を考慮する場合 1

$$\mathrm{X}=\left[\beta_1,\cdots,\beta_{30},\left(\frac{K_{\mathrm{v}}}{K_{\mathrm{h}}}\right)_1,\cdots,\left(\frac{K_{\mathrm{v}}}{K_{\mathrm{h}}}\right)_6,c,\rho\right]^T \tag{48}$$

3. 孔隙率を考慮する場合 2

$$\mathrm{X}=\left[\beta_1,\cdots,\beta_{56},\left(\frac{K_{\mathrm{v}}}{K_{\mathrm{h}}}\right)_1,\cdots,\left(\frac{K_{\mathrm{v}}}{K_{\mathrm{h}}}\right)_6,c,\rho\right]^T \tag{49}$$

βは浸透率分布，孔隙率分布に関わる Walsh 関数のパラメータ，$K_{\mathrm{v}}/K_{\mathrm{h}}$は垂直・水平方向の浸透率浸透率比，$c$ は岩石圧縮率，ρは相関係数を表す．孔隙率は複数の検層結果が比較的一致していることから，1,では観測データを使用した．

浸透率と孔隙率の分布は，仮にそれぞれグリッドに対して最適化を行う場合，グリッドの数だけ最適化問題の次元が増加するため，計算資源の制約から最適化することは難しい．本研究では分布を Walsh 関数の重ね合わせで表現することで対応した．ただ，Walsh 関数によって次元を削減したとしても次元は大きいため，グリッド全体をいくつかのグリッドの集合に再分割し，それぞれの集合において同一の浸透率を用いることとして計算した．x, y 方向の分割数をそれぞれ n_{x}, n_{y} とすれば，分布$\eta_k(n_x, n_y)$は Walsh 関数Ψを用いて，

$$\eta_k(n_x, n_y)=\sum\beta_{lm,k}\Psi_l(n_x)\Psi_m(,n_y), \tag{50}$$

と表される．従って，浸透率は以下のようになる，

$$k(n_x, n_y)=k_{av}\cdot 10^{\eta_k}. \tag{51}$$

孔隙率を考慮する場合は，2 章の LHSD を用いて$\beta_{lm,k}$と$\beta_{lm,\phi}$を発生させ，

$$\eta_\phi(n_x, n_y)=\sum\beta_{lm,\phi}\Psi_l(n_x)\Psi_m(n_y), \tag{52}$$

から孔隙率を得る．1.と 3.はグリッドを x, y 方向にそれぞれ 16 分割，2.は 4 分割することで浸透率と孔隙率を計算している．また，相対浸透率，及び毛管圧力はコア実験により得られたテーブルデータとした．

5.3. 浸透率の最適化

5.3.1. ケース 1

最適化の進行に伴う目的関数の値の変化を Figure 5-9 に示す．最適化計算初期で目的関数の値は急激に改善した後，関数評価回数が 3000 回以降で収束し，値はほとんど変化しない．

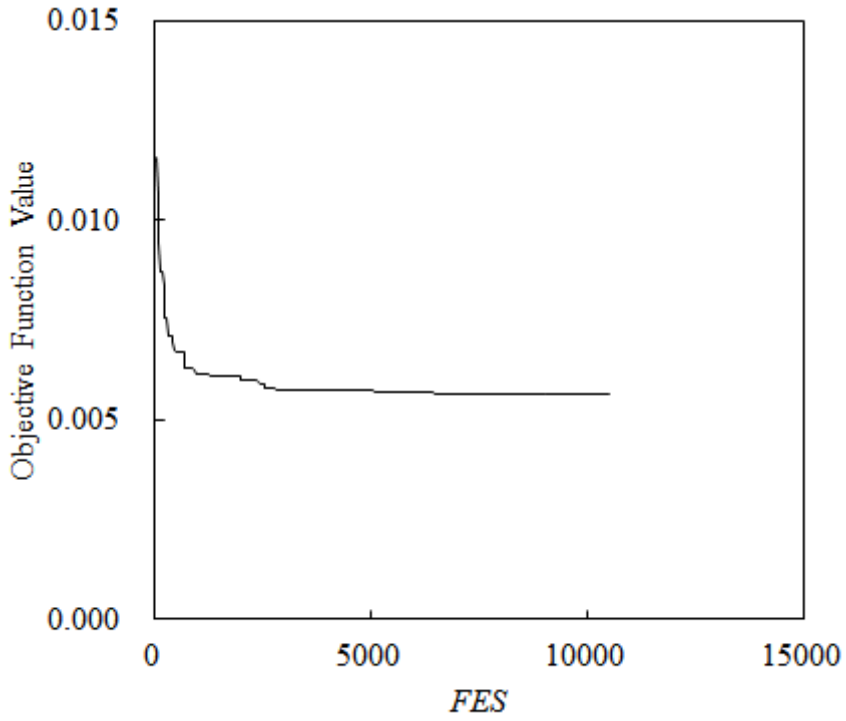

Figure 5-9 Convergence Behavior of Objective Function Value

次に，各種変数について得られた最適解について Figure 5-10 から Figure 5-12 に示す．全ての最適解の値は正規化している．岩石圧縮率 c ($U_{\mathrm{opt},35}$)，浸透率分布β ($U_{\mathrm{opt},i}$ ($i=1,\ldots,28$))，浸透率の垂直/水平方向の比$K_{\mathrm{v}}/K_{\mathrm{h}}$ ($U_{\mathrm{opt},i}$ ($i=29,\ldots,34$))に関わる変数の順で最適化が行われていることが見てとれる．再適化するパラメータの種類は多くないものの，貯留層全体に共通するパラメータから，貯留層の各レイヤーに関するパラメータの順で最適化されている．従来からの手動で行うヒストリーマッチングでは，初めに貯留層の平均浸透率を最適化した後，各レイヤー，グリッドの分布を変化させるといった方法をとる．このような全体から個別要素への最適化は，過去の知見を踏まえ，最適化の効率を考えて選択されるものと考えられる．ILHS による自動的な最適化においても，そのプロセスを踏襲していることが示唆される．

ここで，目的関数と主変数の値の変化を比較する．目的関数の値は評価回数 3000 回以降であまり変化しないものの，最適解については，岩石圧縮率を除く多くで変化している．特に最後に収束する$U_{\mathrm{opt},i}$ ($i=29,\ldots,34$) において，最適解の変化が顕著である．他にも，$U_{\mathrm{opt},3}$, $U_{\mathrm{opt},21}$は比較的大きく変化している．従って，2.3 でも述べたように，目的関数の値だけから最適化計

算の収束を判定することは困難である.

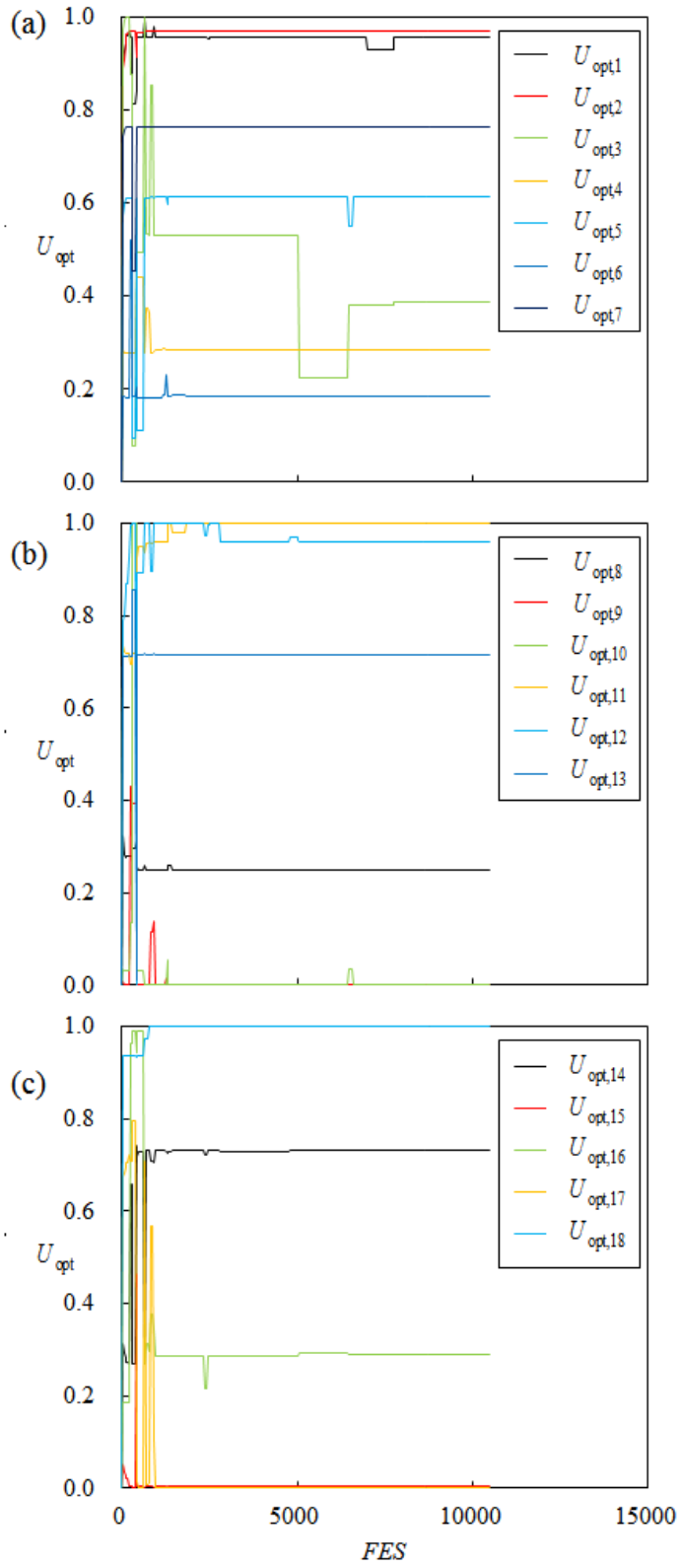

Figure 5-10 Convergence behavior of Normalized primary variables for permeability distribution

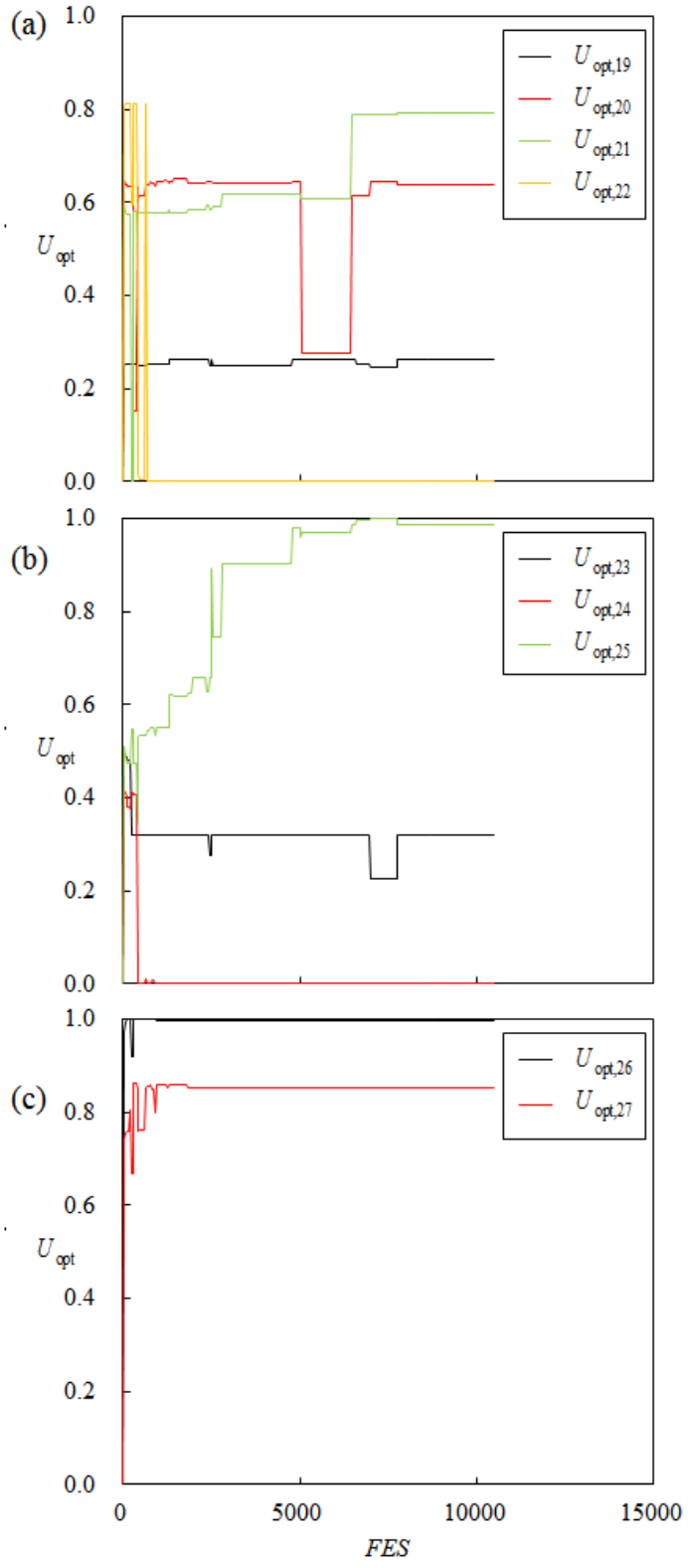

Figure 5-11 Convergence behavior of Normalized primary variables for permeability distribution

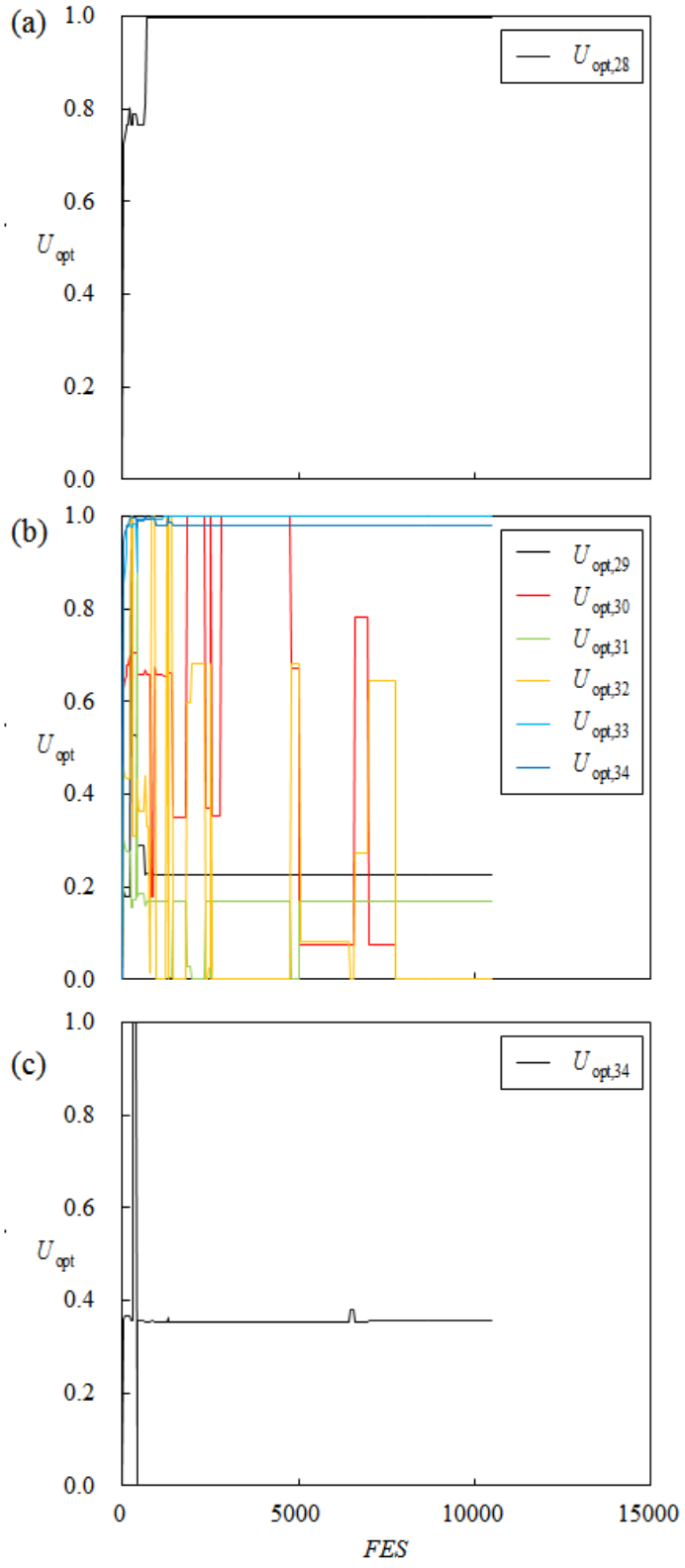

Figure 5-12 Convergence behavior of Normalized primary variables for (a) permeability distribution, (b) ratio of horizontal and vertical permeability and (c) rock compressibility

そこで，正規化エントロピーの変化を見ると，目的関数の収束が確認できて以降も一部の主

変数については正規化エントロピーがある程度高い値を保っている．$d\eta_3$，$d\eta_5$，$d\eta_{30}$，$d\eta_{32}$で iter = iter,maxでも閾値$d\eta_{\min}$を下回っていない．$d\eta_3$は前述の最適解が目的関数の収束以降も変化した変数である$U_{\mathrm{opt},3}$に対応し，$d\eta_{30}$，$d\eta_{32}$は収束が遅い$K_\mathrm{v}/K_\mathrm{h}$に関わる変数である．$U_{\mathrm{opt},5}$については早い段階で最適解を選択しているものの，計算初期で大きく解が変化しており，最適解を選択後も 2 度，異なる最適解を選んでいるため，エントロピーの収束が遅れたと考えられる．以上から，正規化エントロピーは，最適解の収束をよく表しており，最適化計算の収束基準として有効であるといえる．

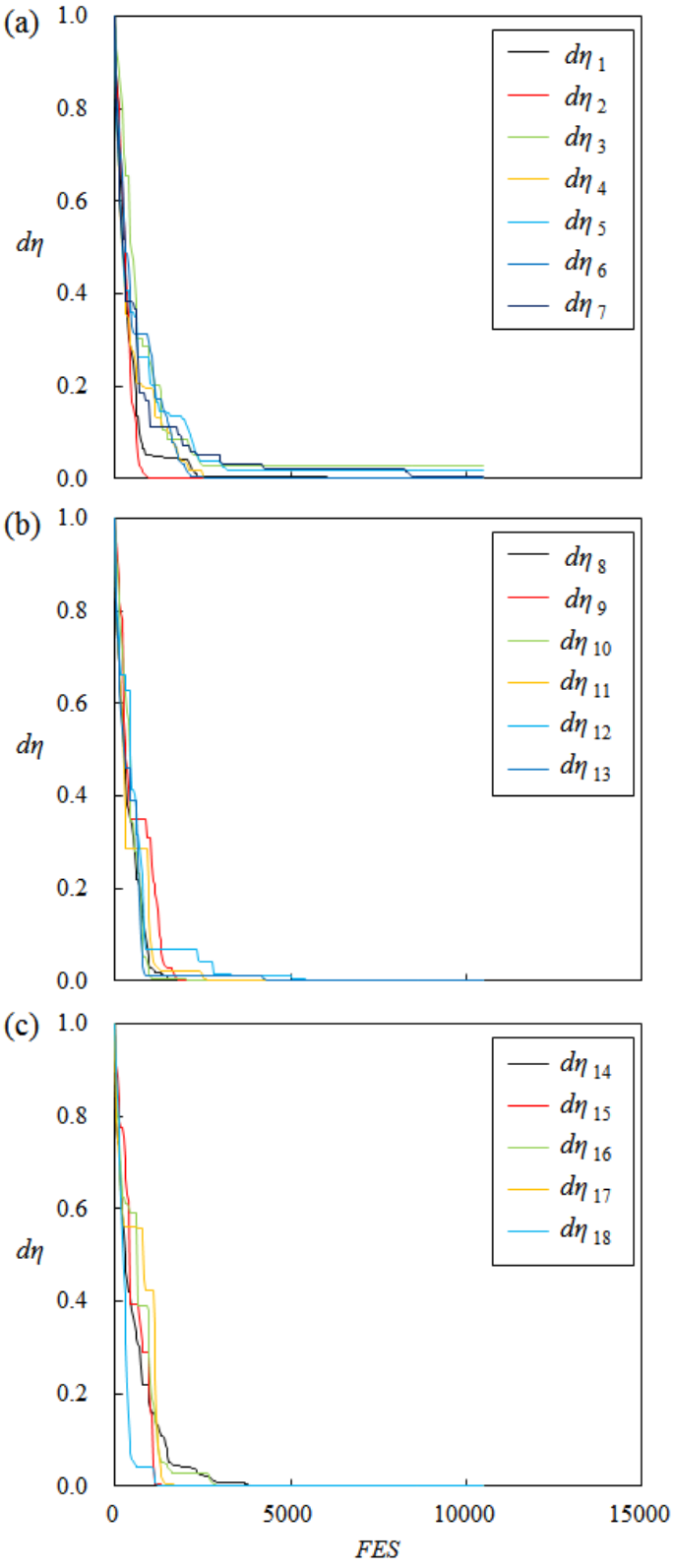

Figure 5-13 Convergence behavior of difference between normalized entropy and normalized BEF for permeability distribution

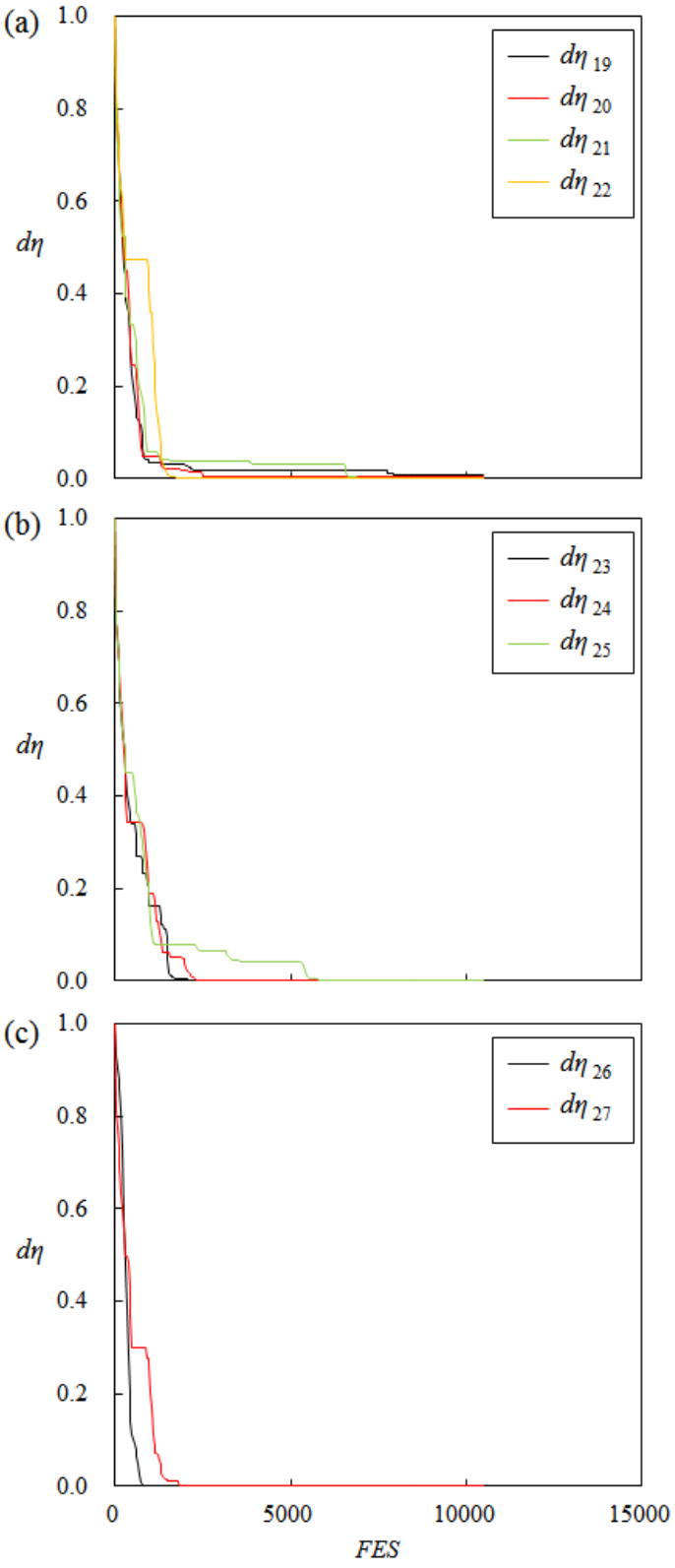

Figure 5-14 Convergence behavior of difference between normalized entropy and normalized BEF for permeability distribution

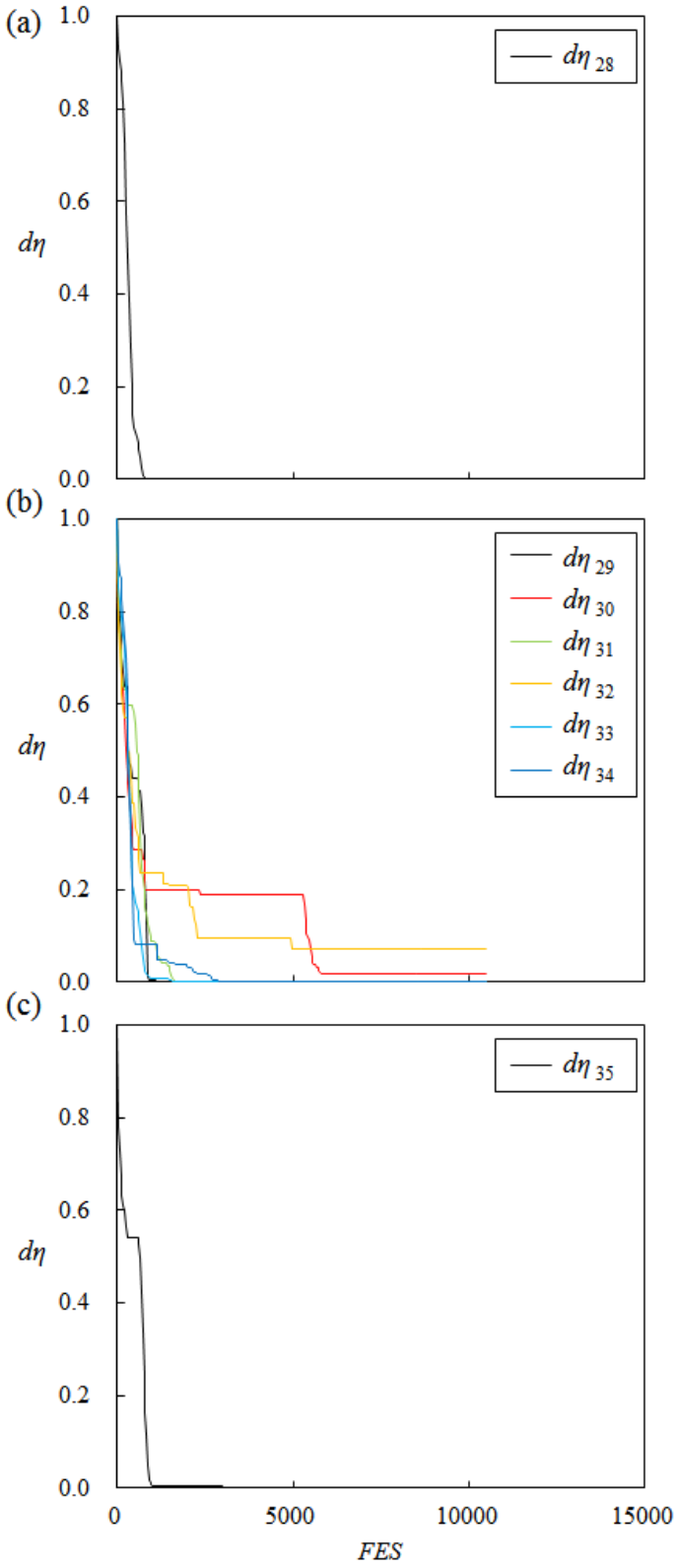

Figure 5-15 Convergence behavior of difference between normalized entropy and normalized BEF for (a) permeability distribution, (b) ratio of horizontal and vertical permeability and (c) rock compressibility

目的関数を変更し，OB-3 の CO_2 飽和度の項を削除して再計算を行ったところ，最適解の値

は変化しないものの，計算終了時に収束している正規化エントロピーのパラメータ数は減少した．これは，解探索の初期段階で OB-3 の CO_2 飽和度に関して，現実的でない解を多く含まれたためと考えられる．このことから，目的関数の適切な設定は効率的な解探索をする上で重要であることが示唆される．

次に，観測データと最適解との比較を示す．IW-1 の坑底圧分布は既往の研究と比較して良い一致を示したものの（Figure 5-16），OB-4 については観測値よりも低い値となった（Figure 5-17）．CO_2 の到達時間についても，OB-4 への CO_2 到達時間は観測値と近い反面，OB-2 については実際と大きく離れた値となった（Figure 5-18）．

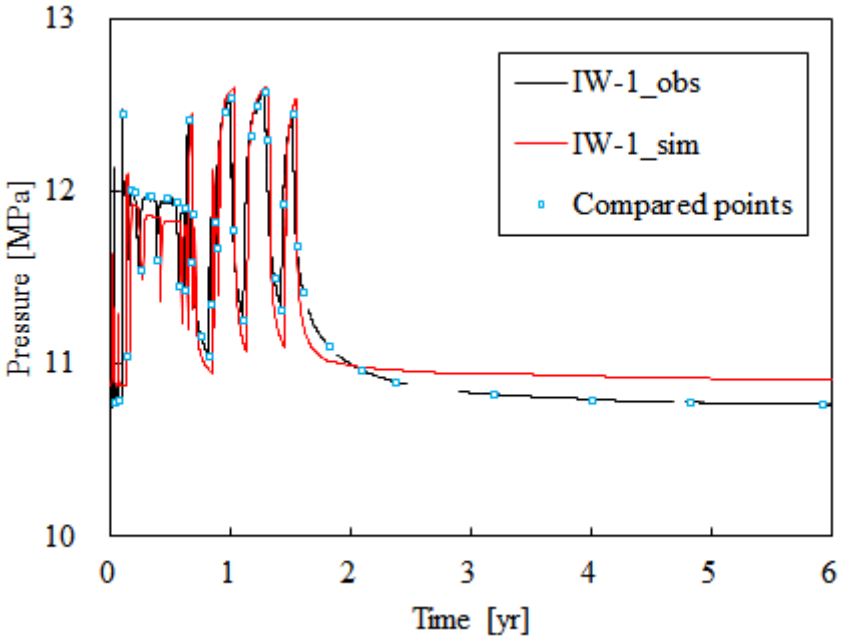

Figure 5-16 Time series of observed and calculated bottom hole pressure at the injection well attained by permeability optimization

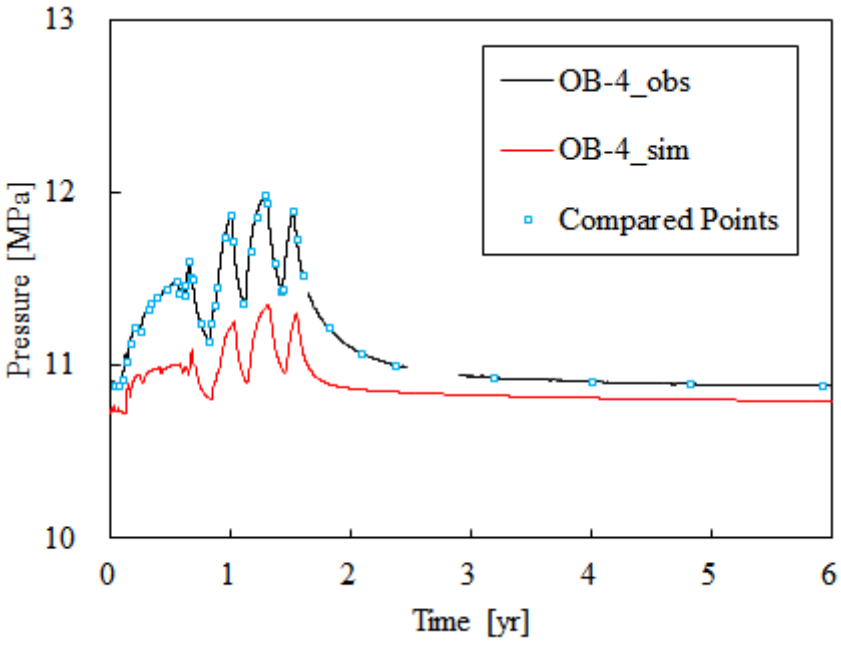

Figure 5-17 Time series of observed and calculated bottom hole pressure at the observation well attained by permeability optimization

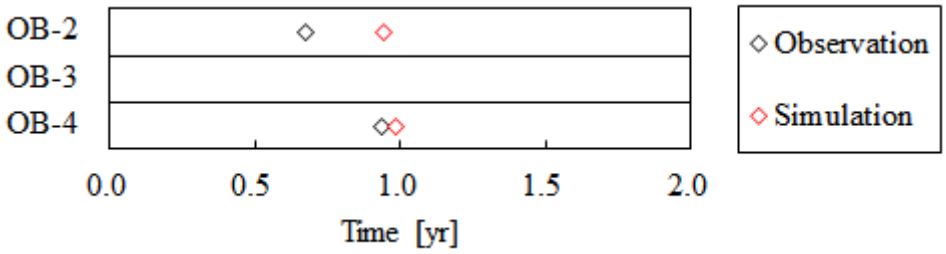

Figure 5-18 CO_2 arrival time difference between observation and simulation at observation wells

最適化計算の初期では，必ずしも各目的関数それぞれを同時に改善するような最適解は得られない．しかしながら，極端な各目的関数の重みづけを行わない限り，最適化計算の進行に伴って，各目的関数全てを改善するような，もしくは改悪しない最適解が得られるはずである．しかしながら，本計算例のように，一部のデータに関してはある程度高い精度で再現ができているにも関わらず，一部のデータでは再現の精度が低いだけでなく，最適化の初期よりも目的関数の値が悪化している．従って本計算例では，最適化計算の手法ではなく最適化問題の設定に実際とかけ離れた点があるものと考えられる．

そこで，より詳細に検討するため，貯留層の CO_2 飽和度の変化の比較を行った．Figure 5-19 と Figure 5-20 はそれぞれ，観測井 OB-2 と OB-4 における観測データから得られた CO_2 飽和度と計算値を示している．計測誤差が大きく定量的な評価は難しいものの，全体として，観測データでは主に下層 (Figure 5-19 (c)，Figure 5-20 (c)) での CO2 飽和度が大きく増加している一

方で，最適化による CO2 飽和度は主に上層(Figure 5-19 (a)，Figure 5-20 (a))において増加している．

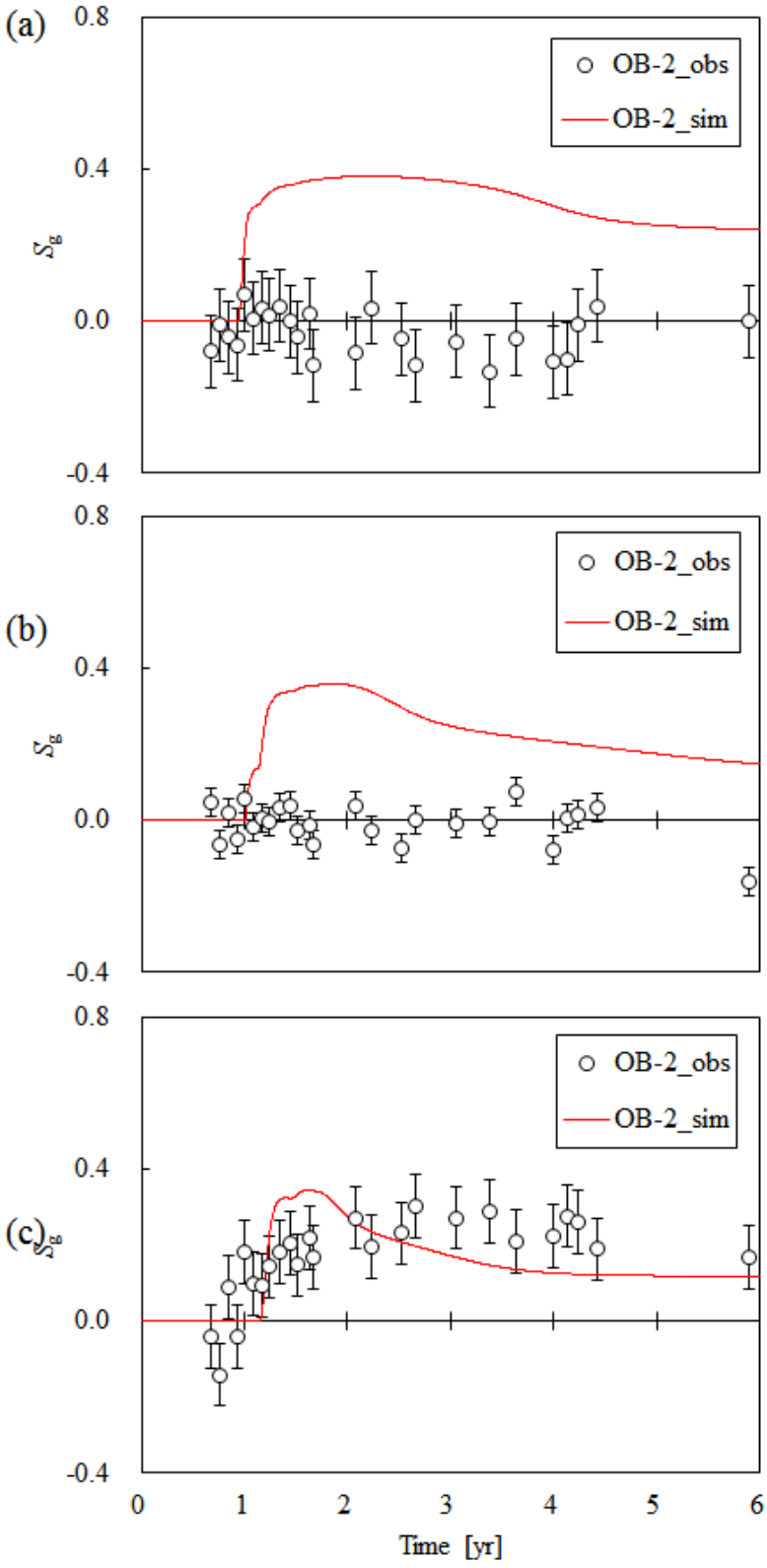

Figure 5-19 Time series of observed and calculated CO_2 saturation at OB-2 in (a) top, (b) middle and

(c) bottom layer

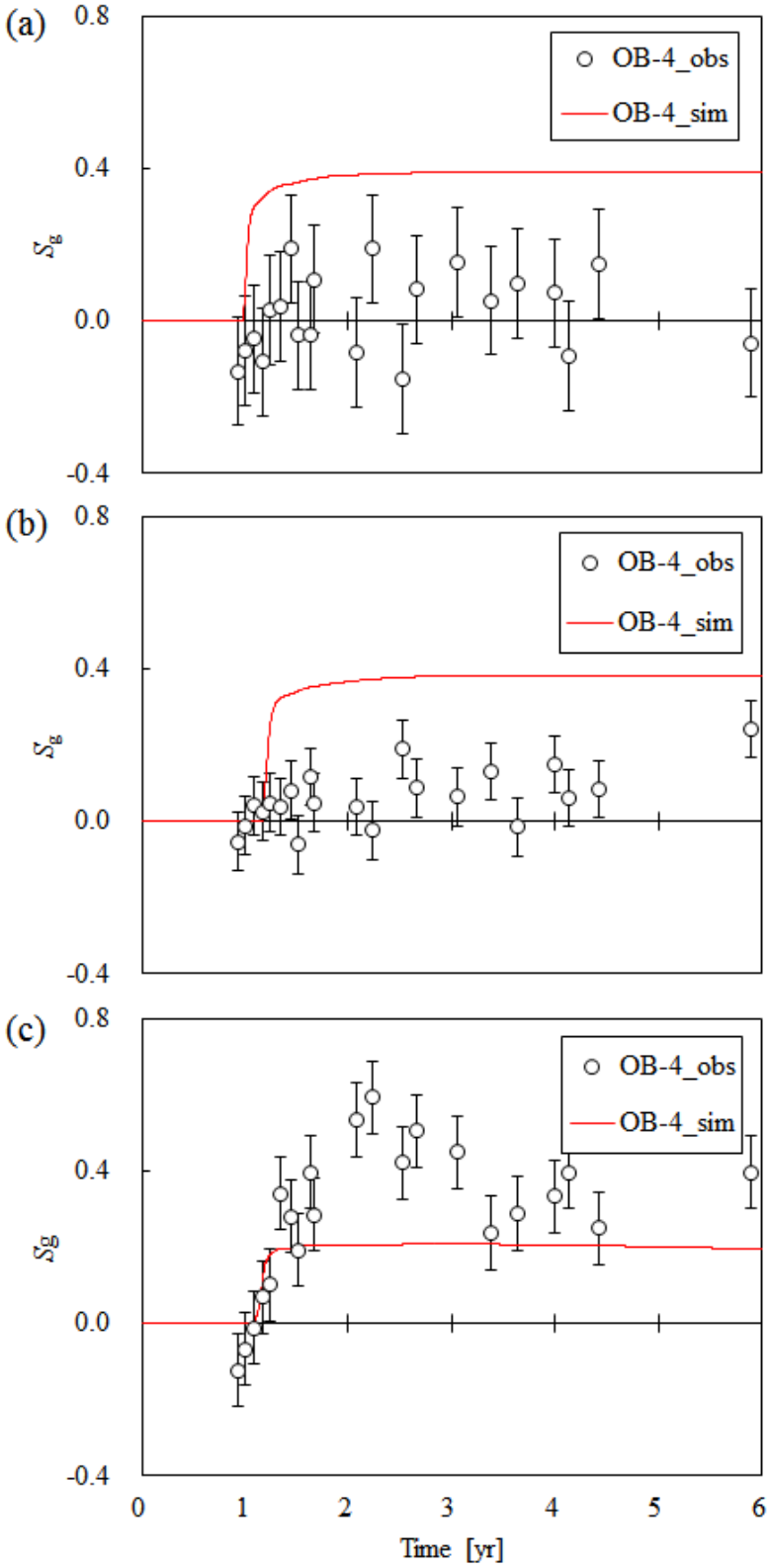

Figure 5-20 Time series of observed and calculated CO_2 saturation at OB-4 in (a) top, (b) middle and (c) bottom layer

CO_2の圧入中,圧入井の穿孔区間を対象とした PLT 測定が行われている(君島&佐藤, 2008). PLT 試験では坑井内の温度, 圧力, 流体の密度, 及びスピナーの回転数と速度を計測しており, 圧入井内の各深度での流速, 及び管外への流出量, つまり圧入量を把握することが出来る (Figure 5-21). 貯留層の下層において流速が低くなっており, 上層よりも下層の方が CO_2圧入の割合が高いと予測される. この傾向は, 中性子検層から計算される CO_2飽和度に関する傾向と同様である. 従って, 上層と下層との間に圧入 CO_2の流動に関与する地質構造の違いがあると考えられる. 本節の最適化計算では, 観測された孔隙率は比較的信頼性が高いものと考え, 浸透率を中心とした最適化を行ったものの, 浸透率の最適化だけでは不十分であり, 孔隙率を含めた最適化が必要であることが示唆される.

加えて, Figure 5-21 を考慮し, 深度によって異なる圧入レートを設定することが考えられ, 既往の研究でも上層から 25%, 下層から 75%の CO_2が圧入するように設定されている (RITE, 2014). しかしながら, 圧入割合は本来, 圧力差から決定されるものであり, 圧入割合を初めから決定してしまっては, 貯留層の地質データを不当に評価することにつながる. 従って, 本研究では坑井全体の圧入レートのみを設定し, 圧力水頭差が等しくなるよう圧入割合を決定している.

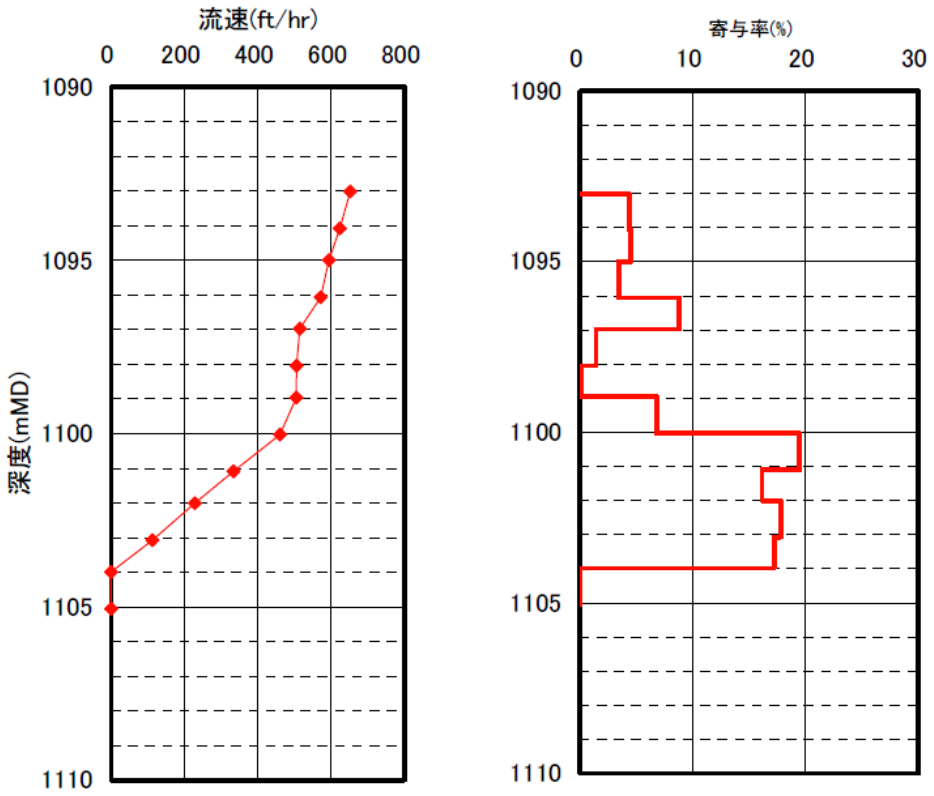

Figure 5-21 Flow rate inside the injection well, IW-1 and percentage of injected CO_2 corresponding to the depth (RITE, 2005)

5.4. 浸透率と孔隙率の最適化

5.4.1. ケース 2

次に，浸透率と孔隙率を主変数に含む最適化を行った．目的関数の値を Figure 5-22 に示す．前節の結果と比較して，目的関数の値が 1/2 以下になっており，全体としてヒストリーマッチングの精度は向上しているといえる．

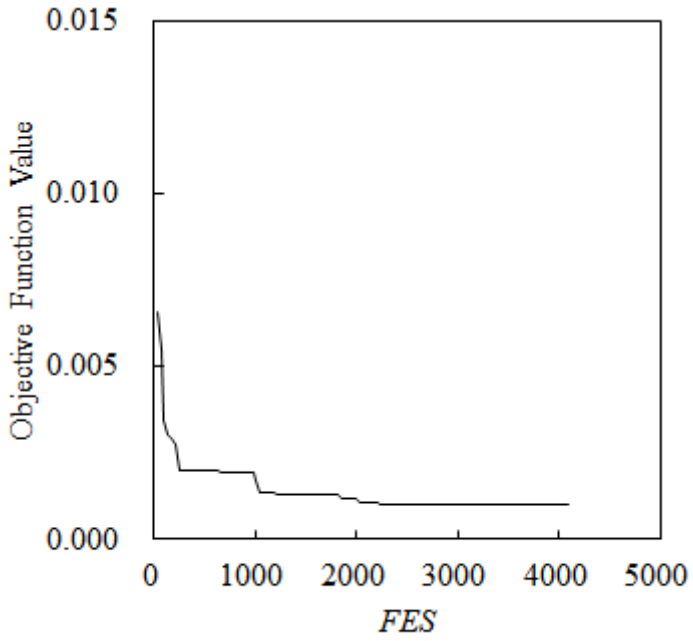

Figure 5-22 Convergence Behavior of Objective Function Value

Table 2-1 は Iter=100 での正規化エントロピーの最小値と正規化した最適解を示す．全ての正規化エントロピーが収束基準を満たすわけではないものの，孔隙率を考慮した場合の方が，最適化計算の収束は速い．

Table 5-4 Value of difference between normalized entropy and normalized BEF

i	1	2	3	4	5	6	7	8	9	10
$d\eta_{i,\min}$	0.000	0.021	0.005	0.000	0.000	0.000	0.000	0.038	0.000	0.005
$U_{\mathrm{opt},i}$	0.927	1.000	1.000	0.619	0.228	0.322	0.313	1.000	1.000	0.074
i	11	12	13	14	15	16	17	18	19	20
$d\eta_{i,\min}$	0.000	0.000	0.036	0.000	0.102	0.000	0.000	0.123	0.000	0.000
$U_{\mathrm{opt},i}$	0.962	0.657	0.000	0.199	0.117	0.757	0.862	0.099	0.911	0.927
i	21	22	23	24	25	26	27	28	29	30
$d\eta_{i,\min}$	0.096	0.000	0.008	0.099	0.000	0.006	0.010	0.047	0.000	0.000
$U_{\mathrm{opt},i}$	0.000	0.036	0.000	0.006	0.660	0.000	0.925	0.000	0.974	0.980
i	31	32	33	34	35	36	37	38		
$d\eta_{i,\min}$	0.000	0.000	0.149	0.123	0.010	0.000	0.000	0.000		
$U_{\mathrm{opt},i}$	0.235	0.858	0.958	0.000	0.981	0.974	0.686	1.000		

次に，坑底圧，CO_2の到達時間について示す（Figure 5-23, Figure 5-24, Figure 5-25）．OB-4の坑底圧，CO_2到達時間，特にOB-2への到達時間は大きく向上したものの，IW-1の坑底圧は実測値を超過している．

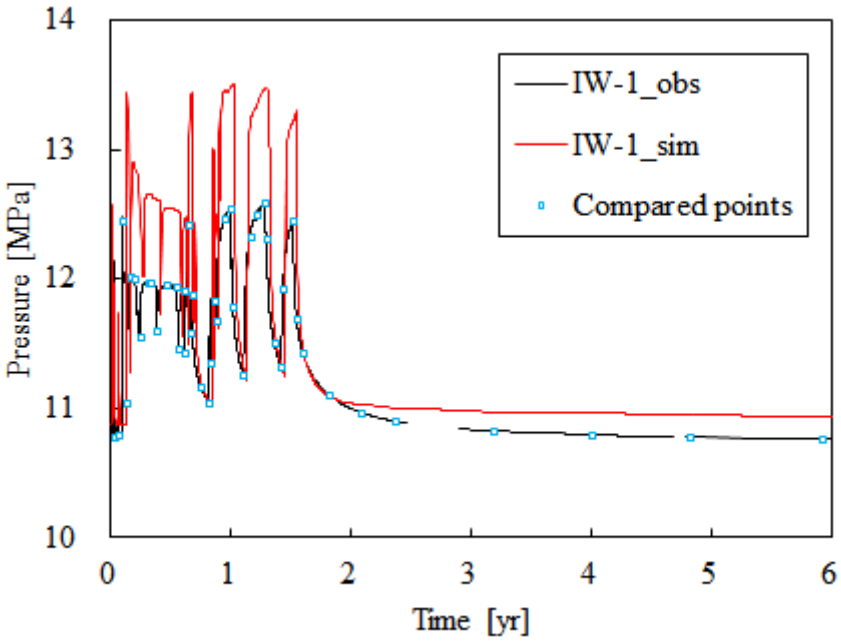

Figure 5-23 Time series of observed and calculated bottom hole pressure at the injection well attained by permeability optimization

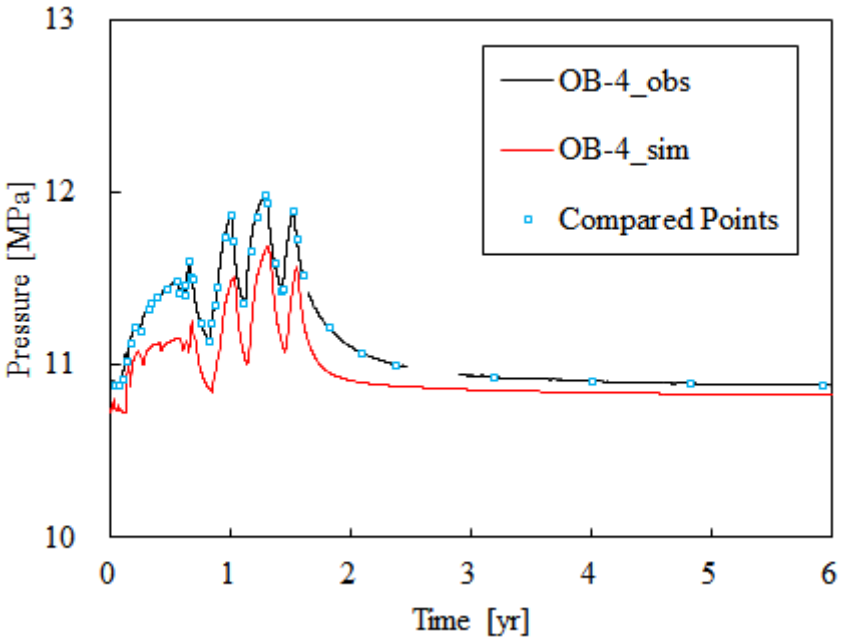

Figure 5-24 Time series of observed and calculated bottom hole pressure at the injection well attained

by permeability optimization

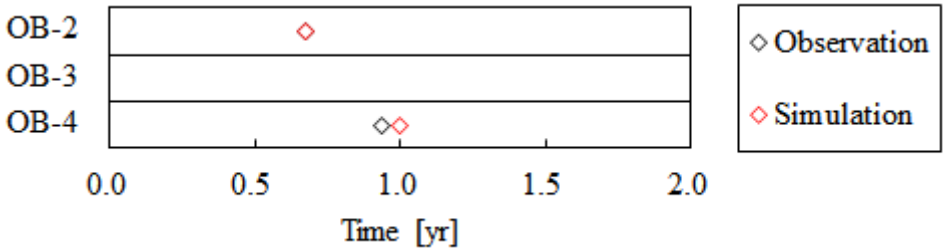

Figure 5-25 CO_2 arrival time difference between observation and simulation at observation wells

5.4.2. ケース 3

そこで，浸透率と孔隙率の分割数を *x*, *y* 方向にそれぞれ 4 倍に増やし，再度，最適化計算を行った．結果を Figure 5-26, Figure 5-27, Figure 5-28 に示す．IW-1，OB-4 ともに実測値と計算値との誤差は縮小しており，浸透率，孔隙率に用いるパラメータを増加させることで，より詳細な HM を行えると考える．

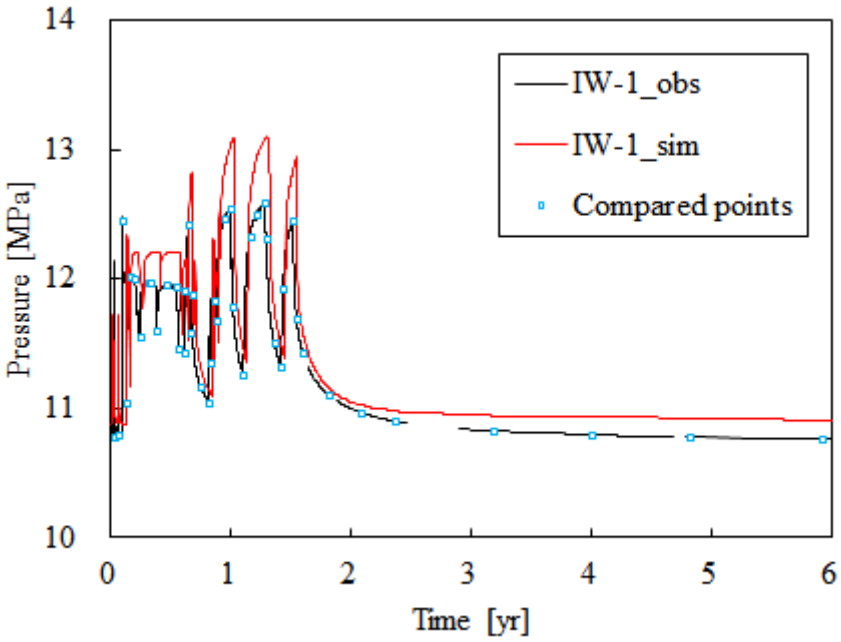

Figure 5-26 Time series of observed and calculated bottom hole pressure at the injection well attained by permeability optimization

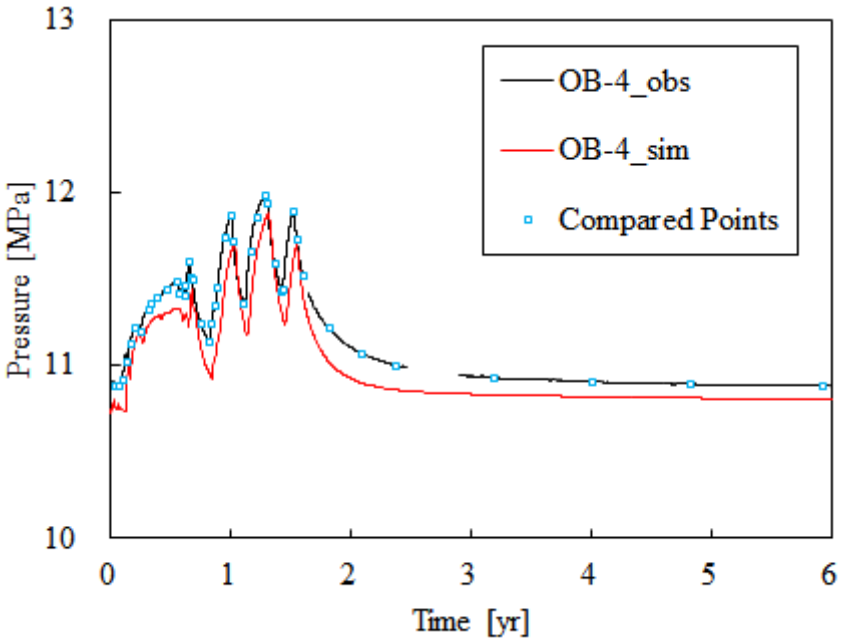

Figure 5-27 Time series of observed and calculated bottom hole pressure at the injection well attained by permeability optimization

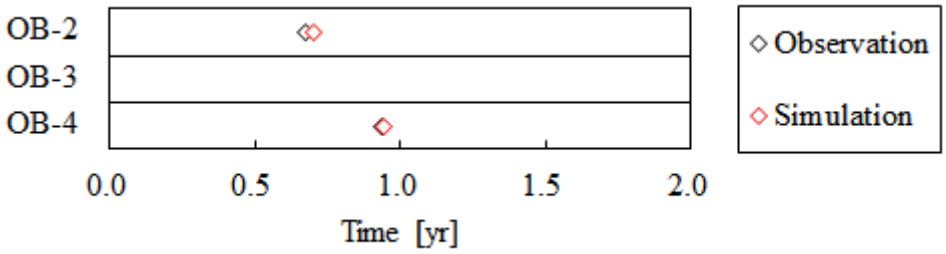

Figure 5-28 CO_2 arrival time difference between observation and simulation at observation wells

6. 結論

本研究では，貯留層シミュレーション，特に CO_2 地中貯留に関わる最適化問題を対象とした実用的な大域的最適化アルゴリズムを開発し，坑井配置の最適化問題，及び CO_2 地中貯留のヒストリーマッチングに適用することでその有効性を検証した．以下に，本研究により得られた結論と今後の検討課題について，第 1 章から順に述べる．

第 1 章では，CO_2 地中貯留の必要性の大きな高まりと，CO_2 地中貯留プロジェクトで行われる貯留層シミュレーションの特徴と課題について述べた．本研究は，「貯留層シミュレーションに適した最適化アルゴリズムの開発」，「シミュレータの高速化」，「目的関数の適切な設定」の 3 点に着目し，実用的な大域的最適化アルゴリズムの開発を行うことを述べた．

第 2 章では，いくつかの大域的最適化アルゴリズムについて述べ，貯留層シミュレーションを伴う最適化問題には多点探索法が有効であることを示した．次に，「貯留層シミュレーションに適した最適化アルゴリズムの開発」を目的として，多点探索法について，「新規収束判定手法の導入」と「ロバスト性の向上」という 2 点から最適化手法の改良を行い，それぞれベンチマーク関数に対して適用した．

貯留層シミュレーションはその計算に長時間を要すること，また最適化問題としての非線形性が高いことから，大域的最適化アルゴリズムの中でも多点探索法が有効である．多点探索法の中でも繰り返しラテン超方格法（ILHS）は，目的関数の値の順位から累積分布関数を更新するという点が特徴的であり，また入力パラメータが 1 つですむという点で，目的関数の値が未知のブラックボックス最適化に対して有効である．

多点探索法では解探索の進展に伴い，「大域的探索」から「局所的探索」に探索方法が変化するという点に着目し，正規化エントロピーを用いた収束判定基準を提案した．ILHS における探索領域のエントロピーは，探索領域間の差が大きくなるほど低下し，2 値エントロピー（BEF）と等しくなる．ベンチマーク関数の最適化から，実際に，探索領域のエントロピーは BEF に近づくことが示された．したがって，正規化エントロピーと正規化された BEF との差を指標とすることで，最適化計算の収束を判断できることを示した．また同時に，上記指標の閾値について評価し，0.01 を下回ったとき最適化計算は収束したとみなせる．

多点探索法のロバスト性の向上を目指して，従属ラテン超方格法（LHSD）を用いた ILHS の改良を行った．ベンチマーク関数への適用から，最適解の値を維持しつつ，変動係数を低減できることを示した．

3 章では，「シミュレータの高速化」を目的として，貯留層シミュレータ TOUGH2/ECO2N,

及び大域的最適化アルゴリズム ILHS の改良を行った. TOUGH2/ECO2N の改良については,「タイムステップ幅の更新方法の変更」,「ソルバーの改良」,「相対浸透率モデルの追加」という点について述べた．改良により約 2 倍の高速化を達成するとともに，TOUGH2/ECO2N での計算の収束性が向上した.

ILHS では，1 回のタイムステップ内で目的関数の計算が独立に行われる点に注目し，MPI による並列化を行った. 最後に, MPI 並列と Hyper-Q 機能を持つ GPU を併用することにより，約 12 倍の高速化を達成した．加えて，TOUGH2/ECO2N の計算では，CPU と GPU のどちらを用いる場合もデータのメモリへのアクセスがボトルネックとなることが示唆され，最適化計算に必要とされる性能について示した.

第 4 章では，開発された大域的最適化アルゴリズムを坑井配置の最適化問題に適用することで，実際の工学的問題に対する提案手法の有効性を検証した．地中貯留 CO_2 のトラップ量，ここでは残留ガストラップ量と溶解トラップ量の和が最大となるよう目的関数を設定し，垂直坑井からの CO_2 圧入, 水平坑井からの CO_2 圧入と地層水生産という 2 つの問題について計算した. 問題の次元 d に対して，サンプル数 $n_{\mathrm{pop}} = 2d$, 繰り返し計算の最大値を 100 とし，全ての主変数に対して正規化エントロピーと正規化された BEF との差が閾値 0.01 を下回れば，最適化計算を終了するものとした.

5 回の独立試行を行い，どちらの問題でも，5 回中 3 回の試行で，最適化計算は収束判定基準を満足した．収束以降，最適解は改善しないか，改善したとしても大きく向上することはないことから，正規化エントロピーによる収束判定基準は有効であるといえる．同時に，正規化エントロピーの導入により，どちらの問題に対しても最適解の値を維持しつつ，関数評価回数を約 30%削減することができた.

また 2 つの問題の比較から，水平坑井からの CO_2 圧入と地層水の生産は，トラップ CO_2 量を増加させ，CO_2 の安定的な貯留に寄与するという知見を得た.

第 5 章では，新潟県南長岡，岩野原で行われた CO_2 地中貯留の実証試験を対象としたヒストリーマッチングを行った. 目的関数は圧入井と観測井の坑底圧, 観測井への CO_2 到達時間, CO_2 が到達していない観測井 OB-3 に限っては CO2 飽和度とした．本研究では，孔隙率を観測データとして浸透率を最適化させる問題，浸透率と孔隙率を同時に最適化する問題を取り扱った. 浸透率と孔隙率には第 2 章で述べた LHSD によって，その相関関係を表現した.

前章と同様に，正規化エントロピーは最適解への収束を判断する指標となりえるものの，収束が起こるまでの関数評価回数が非常に多いため，計算終了までに全ての変数について正規化エントロピーの値が閾値を下回ることはなかった.

浸透率の最適化問題では，圧入井の坑底圧は精度よく再現できるものの，観測井に関しては実測値を大きく下回る結果となり，浸透率のみの最適化が不十分であるといえる．一方，浸透

率と孔隙率の最適化問題では，目的関数の値は浸透率だけの最適化と比較して 1/2 以下に向上し，観測データの再現性も向上した．加えて，浸透率，孔隙率分布をより細かく最適化することで，データの再現性はより向上する．

「目的関数の適切な設定」については，観測井 OB-3 の CO_2 飽和度を目的関数に加える事で，収束挙動が速くなることが確かめられ，適切な目的関数を設定することは最適化計算の収束の上で重要であるという知見を得た．

ヒストリーマッチングの結果は，まだ実測値との誤差が大きいため，今後も検討を重ねる余地がある．第一に検討すべき項目として挙げられるのは，浸透率・孔隙率分布のより詳細な最適化である．また，CO_2 の貯留形態による影響も考えられ，その点で，本研究では考慮しなかった，相対浸透率モデルによる残留トラップ CO_2 の表現が考えられる．上記の項目を考慮した場合，最適化計算の次元の増加が考えられる．その場合は，電子データの容量圧縮に用いられるニューラルネットワークを用いて，次元を削減することが可能であると考える．

大域的最適化，特に多点探索法については，入力パラメータを繰り返し計算回数の関数とする改良が多く見られ，ILHS でもそのような改良を行う余地がある．本研究で提案した収束判定基準とロバスト性向上のための従属性の設定は，探索領域を設定すれば ILHS だけでなく PSO や DE にも適用が可能である．

第 5 章の結果からも分かるように，目的関数の適切な設定は，最適解の導出だけでなく，最適化計算の収束性，つまり効率にも大きく影響すると考えられる．そのため，複数のパラメータを同時に含む目的関数の表現方法には更なる検討が必要である．複数の目的関数を同時に最適化する多目的最適化や，観測データが測定誤差を含むような場合にも適用可能なロバスト最適化の検討も必要である．他にも，複数の大域的最適化アルゴリズムを組み合わせたハイブリッドな手法が開発されており，ILHS と他の多点探索法，もしくは局所探索法を組み合わせることも考えられる．

参考文献

- Aziz, K., & Settari, A. (1979). Petroleum reservoir simulation. Chapman & Hall.
- Beckner, B. L., Song, X., (1995). Field Development Planning Using Simulated Annealing - Optimal Economic Well Scheduling and Placement. SPE Annual Conference and Exhibition, Dallas, Texas.
- Dake, L. P. (1983). Fundamentals of reservoir engineering. Elsevier.
- Davidor, Y. (1991). Epistasis Variance: A Viewpoint of GA-Hardness. *Foundations of Genetic Algorithms*, **1**, 23-35.
- de Oca, M. A. M., Aydın, D., & Stützle, T. (2011). An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re) design of optimization algorithms. *Soft Computing*, **15**(11), 2233-2255.
- Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE*, **1**(4), 28-39.
- Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science (Vol. 1, 39-43).
- Feoktistov, V. (2006). Differential evolution (1-24). Springer.
- Feller, W. (2008). An introduction to probability theory and its applications (Vol. 2). John Wiley & Sons.
- Fisher, N. I. (2006). Copulas, Encyclopedia of Statistical Sciences. John Wiley & Sons.
- Floris, F. J. T., Bush, M. D., Cuypers, M., Roggero, F., & Syversveen, A. R. (2001). Methods for quantifying the uncertainty of production forecasts: a comparative study. *Petroleum Geoscience*, **7**(S), S87-S96.
- Global CCS Institute. (2011). Strategic analysis of the global status of carbon capture and storage: Economic assessment of carbon capture and storage technologies.
- Global CCS Institute. (2013). The global status of CCS 2013.
- Global CCS Institute. (2014). The global status of CCS 2014.
- Goda, T., & Sato, K. (2014). History matching with iterative Latin hypercube samplings and parameterization of reservoir heterogeneity. *Journal of Petroleum Science and Engineering*, **114**, 61-73.
- Helmig, R. (1997). Multiphase flow and transport processes in the subsurface: a contribution to the modeling of hydrosystems. Springer-Verlag.
- Herrera, F., & Lozano, M. (2009). ISDA'09 workshop on evolutionary algorithms and other metaheuristics for continuous optimization problems—a scalability test.

Technical report, University of Granada, Pisa, Italy.

- Herrera, F., Lozano, M., & Molina, D. (2010). Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. Technical report, University of Granada, Spain.
- IEA. (2009). CO_2 capture and storage: A key carbon abatement option.
- IEA. (2009). Technology roadmap, Carbon capture and storage.
- IEA, (2012). Energy technology perspectives: Pathways to a clean energy system.
- IEA. (2015). Energy Technology Perspectives.
- Ishizawa, Y., Matsumoto, K., Sato, K., Okatsu, K., & Miyake, Y. (2013). Accelerating of the Reservoir Simulator TOUGH2 by GPU. *Energy Procedia*, **37**, 3764-3770.
- Land, C. S. (1968). Calculation of imbibition relative permeability for two-and three-phase flow from rock properties. *Society of Petroleum Engineering Journal*, **8**(02), 149-156.
- Li, D. X. (1999). On default correlation: A copula function approach. *The Journal of Fixed Income*, **9**(4), 43-54.
- Liao, T., Montes de Oca, M. A., Aydin, D., Stützle, T., & Dorigo, M. (2011). An incremental ant colony algorithm with local search for continuous optimization. In Proceedings of the 13th annual conference on Genetic and evolutionary computation. ACM.
- McKinsey&Company. (2009). Pathways to a Low-Carbon Economy version 2 of the global greenhouse gas abatement cost curve.
- Mito, S., Xue, Z., & Ohsumi, T. (2008). Case study of geochemical reactions at the Nagaoka CO 2 injection site, Japan. *International Journal of Greenhouse Gas Control*, **2**(3), 309-318.
- Mito, S., Xue, Z., & Sato, T. (2013). Effect of formation water composition on predicting CO_2 behavior: A case study at the Nagaoka post-injection monitoring site. *Applied Geochemistry*, **30**, 33-40.
- McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**(2), 239-245.
- Nakajima, T., & Xue, Z. (2013). Evaluation of a resistivity model derived from time-lapse well logging of a pilot-scale CO_2 injection site, Nagaoka, Japan. *International Journal of Greenhouse Gas Control*, **12**, 288-299.
- Nwankwor, E., Nagar, A. K., & Reid, D. C. (2013). Hybrid differential evolution and

particle swarm optimization for optimal well placement. Computational Geosciences, 17(2), 249-268.

- Oliver, D. S., & Chen, Y. (2011). Recent progress on reservoir history matching: a review. *Computational Geosciences*, **15**(1), 185-221.
- Onwunalu, J. E. Durlofsky, L. J. (2010). Application of a particle swarm optimization algorithm for determining optimum well location and type. *Computational Geosciences*, **14**(1), 183-198.
- Otake, M. (2013). Evaluation of CO_2 Underground Behavior from Injector's Time-lapse Pressure Fall off Analysis: A Case Study of CO_2 Aquifer Storage Project. *Energy Procedia*, **37**, 3307-3318.
- Packham, N. and Wolfgang M. S. (2010). Latin hypercube sampling with dependence and applications in finance. *The Journal of Computational Finance*, **13**(3), 81–111.
- Pruess, K. Oldenburg, C. Moridis, G. (1999). TOUGH2 user's guide, version 2.0. Report LBNL-43134. Lawrence Berkley National Laboratory.
- Pruess, K. (2005). ECO2N: A TOUGH2 .uid property module for mixtures of water, NaCl and CO_2. Report LBNL-57952. Lawrence Berkley National Laboratory.
- Rank, J. Schmidt, T. and et al. (2006). Copulas-From theory to application in finance. Bloomberg Press.
- Salmon, F. (2012). The formula that killed Wall Street. *Significance*, **9**(1), 16-20.
- Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D.Saisana, M. & Tarantola, S. (2008). Global sensitivity analysis: the primer. John Wiley & Sons.
- Sanders, J., & Kandrot, E. (2010). CUDA by example: an introduction to general-purpose GPU programming. Addison-Wesley Professional.
- Sato, K., Mito, S., Horie, T., Ohkuma, H., Saito, H., Watanabe, J., & Yoshimura, T. (2011). Monitoring and simulation studies for assessing macro-and meso-scale migration of CO_2 sequestered in an onshore aquifer: experiences from the Nagaoka pilot site, Japan. *International Journal of Greenhouse Gas Control*, **5**(1), 125-137.
- Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, **11**(4), 341-359.
- Tanaka, K., Vilcáez, J., & Sato, K. (2013). Improvement of CO 2 Geological Storage Efficiency by Injection and Production Well Design. *Energy Procedia*, **37**, 4591-4597.
- Tang, K., Yáo, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M., & Yang, Z. (2007). Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Nature Inspired Computation and Applications

Laboratory, USTC, China, 153-177.

- Van Genuchten, M. T. (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil science society of America journal*, **44**(5), 892-898.
- Wang, H., Wu, Z., & Rahnamayan, S. (2011). Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Computing*, **15**(11), 2127-2140.

- 青木尊之, & 額田彰. (2009). はじめての CUDA プログラミング. 工学社.
- 新井親夫. (1998). Fortran90 入門 －基礎から再帰手続まで－. 森北出版.
- 大熊宏. (2008). 地下深部塩水層における二酸化炭素地中貯留のシミュレーション技術および長岡圧入実証試験への適用. *Journal of MMIJ*, **124**(1), 87-94.
- 大熊宏, 栗原正典, & 島本辰夫. (2013). 石油鉱業便覧, 第 5 章 7 節. 石油技術協会.
- 北山哲士, 酒井忍, 荒川雅生, & 山﨑光悦. (2010). 大域的最適化法としての Differential Evolution と数値計算. 日本機械学会論文集, C, **76**(771), 2819-2828.
- 君島晋, & 佐藤章吾. (2008). 二酸化炭素地中貯留実証試験での油層工学的諸物性の測定と評価. *Journal of MMIJ*, **124**(1), 61-67.
- 久保幹雄, & J. P. ペドロソ. (2009). メタヒューリスティクスの数理. 共立出版.
- 倉本大輔. (2010). 油･ガス層シミュレーションにおける不確実性評価とヒストリーマッチング. *JOGMEC 石油天然ガスレビュー*, **44**(2), 61.
- 住明正, & 島田荘平. (2009). 温室効果ガス貯留・固定と社会システム. コロナ社.
- 薛自求, & 松岡俊文. (2008). 長岡プロジェクトからみた二酸化炭素地中貯留技術の現状と課題. *地学雑誌*, **117**(4), 734-752.
- 薛自求, & 渡辺二郎. (2008). 長岡実証試験サイトにおける二酸化炭素挙動モニタリングへの物理検層の適用. *Journal of MMIJ*, **124**(1), 68-77.
- 田中啓, & 佐藤光三. (2015). 繰り返しラテン超方格法における正規化エントロピーを利用した収束判定法の導入と CO_2 地中貯留への適用. *Journal of Japan Petroleum Institute*, **58**(6), 384-391.
- 棚瀬大爾, 薛自求, & 嘉納康二. (2008). 長岡における二酸化炭素圧入実証試験. *Journal of MMIJ*, **124**(1), 50-60.
- 合田隆. (2012). 二酸化炭素地中貯留の安全な実施のための大域的最適化アプローチの構築. 博士論文. 東京大学.
- 合田隆, & 佐藤光三. (2011). 繰り返しラテン超方格法を用いた CO_2 地中貯留における坑井配置の大局的最適化. *石油技術協会誌*, **76**(3), 233-243.
- 篠塚英敦. (2012). 21 世紀の統計科学 III, 増補 HP 版, 第 5 章, 日本統計学会.

    - http://park.itc.u-tokyo.ac.jp/atstat/jss75shunen/Vol3.pdf
- 地球環境産業技術研究機構 (RITE). (2005). 平成 16 年度 二酸化炭素固定化・有効利用技術対策事業 二酸化炭素地中貯留技術研究開発 成果報告書.
- 地球環境産業技術研究機構 (RITE). (2008). 平成 19 年度 二酸化炭素固定化・有効利用技術対策事業 二酸化炭素地中貯留技術研究開発 成果報告書.
- 地球環境産業技術研究機構 (RITE). (2013). 平成 24 年度 二酸化炭素・貯蔵安全性評価技術開発事業 成果報告書.
- 地球環境産業技術研究機構 (RITE). (2014). 平成 25 年度 二酸化炭素・貯蔵安全性評価技術開発事業 成果報告書.
- 戸坂凡展, & 吉羽要直. (2005).コピュラの金融実務での具体的な活用方法の解説. *金融研究*, **24**, 115.
- プログレス・テクノロジーズ株式会社. (2015). 平成 26 年度 二酸化炭素・貯蔵安全性評価技術開発事業 CO_2 挙動シミュレーションの高速化および計算パラメータ最適化法の検討作業 研究報告書.
- 槇野洋平, 高橋明子, 今井純, & 舩曳繁之. (2013). 変動係数を終了条件とする PSO と電力平準化制御への適用. *電気学会論文誌*. B, **133**(5), 488-494.
- 三戸彩絵子, 薛自求, & 大隅多加志. (2008). 二酸化炭素地中貯留における地球化学反応特性について. *地学雑誌*, **117**(4), 753-767.
- 棟朝雅晴. (2008). 遺伝的アルゴリズム-その理論と先端的手法. 森北出版.

付録

6.1. A.1.計算コード（ベンチマーク関数の最適化）

- はじめに

繰り返しラテン超方格法（以下，ILHS）は，実験計画法の 1 つであるラテン超方格法（LHS）を基に開発された大域的最適化手法である．本節ではベンチマーク関数に対して，目的関数の値を計算するプログラム ILHS_BMF について解説する．

- プログラムと実行環境

ILHS_BMF は Fortran90 に基づいた複数の計算コードとコンパイル用ファイル Makefile, 2 つの入力ファイルによって構成される．

コンパイルには make もしくは mingw32-make が必要であり，未実装の場合はコンパイラと同じディレクトリにインストールする．

開発・実行環境は Linux 64-bit 及び Windows 32-bit/64-bit であり，以下の環境で動作することを確認している．

	Linux	Unix	Windows
PGI	○	-	-
ifort	-	-	×
gfortran	○	○	○
g95	-	-	○

- コンパイル及び実行方法

ディレクトリ code と Makefile を同一のディレクトリに配置し，同ディレクトリに空のディレクトリ module と object を作成する．

Makefile を編集し，コンパイラを指定する変数 FC を実行環境に合わせて変更する．

・OS: Linux，Unix の場合

```
$ make
$ ./ilhsbmf_v*.*
```

・OS: Windows，コンパイラ: gfortran，g95 の場合

```
> mingw32-make
> ilhsbmf_v*.*
```

- 繰り返しラテン超方格法

ILHS のアルゴリズムの概要を以下に示す．目的関数の次元を d，サンプル点の個数を n_{pop} とする．

d 個の主変数に対する初期（t=1）の累積分布関数の設定

LHS による各主変数に対する n_{pop} -1 個の分割点と n_{pop} 個のサンプル点の設定

n 個のサンプルセットに対する目的関数の評価

目的関数の値に応じたサンプル区間の重み付け

次ステップ（t+1 回目）での各主変数の累積分布関数の導出

2.から 5.を設定した基準が満たされるまで繰り返す．

- ベンチマーク関数

使用しているベンチマーク関数は以下の通りである．

Basic Functions

Function1: Sphere Function

Function2: Schwefel's Problem 2.21

Function3: Rosenbrock's Function

Function4, Rastringin's Funstion

Function5, Griewank's Function

Function6, Ackley's Function

Function7: Schwefel's Problem 2.21

Function8: Schwefel's Problem 1.2

Function9: Extended f10

Function10: Bohachevsky

Function11: Schaffer

Hybrid Functions

Function12: Non-shifted Function9 + Function1

Function13: Non-shifted Function9 + Function3

Function14: Non-shifted Function9 + Function4

Function15: Non-shifted Function10 + Non-shifted Function7

Function16: Non-shifted Function9 + Function1

Function17: Non-shifted Function9 + Function3

Function18: Non-shifted Function9 + Function4

Function19: Non-shifted Function10 + Non-shifted Function7

Rotated Functions

Function20: Rotated Function4

Function21: Rotated Function5

Function22: Rotated Function6

Ill-scaled Functions

Function23: Ill-scaled Sphere Function

Function24: Ill-scaled Rosenbrock Function

- 計算コードの構成

以下に，計算コードの構成を示す．main は main program，それ以外は subroutine を示す．

```
[Program or Subroutine Name]   [File Name]
main                           (mainOPT.f90)
+--- input                     (subINPUT.f90)
|     +--- readc               (subREADC.f90)
+--- mkout                     (subMKOUT.f90)
+--- ilhs                      (subILHS.f90)
|     +--- input_obj           (subOBJ.f90)
|     |     +--- rotmat        (subOBJ.f90)
|     |     |     +--- matmat  (subOBJ.f90)
|     |     +--- readc         (subREADC.f90)
|     +--- gamma               (subGAMMA.f90)
|     +--- mvnrnd              (subMVNRND.f90)
|     |     +--- cholesky      (subCHLSKY.f90)
|     +--- gcopula             (subGCOPULA.f90)
|     +--- obj                 (subOBJ.f90)
|     |     +--- matvec        (subOBJ.f90)
|     +--- sort                (subSORT.f90)
|     +--- weight              (subWEIGHT.f90)
|     +--- cdf                 (subCDF.f90)
|     +--- ent                 (subENT.f90)
```

全てのサブルーチンはモジュール化されている．各サブルーチンで共有される変数，定数は上記とは別のモジュールファイル，

mdlILHS.f90

mdlOBJ.f90

で定義される．

- ILHS のプログラム

・main (mainOPT.f90)

メインファイル．各サブルーチンを呼び出す．出力ファイルの作成，時間の計測を行う．

出力ファイル：out_YMIN.dat, out_YMINavg.dat, out_YMINall.dat,
out_ENTROPY.dat, out_ROT1.dat,
out_NRNDN.dat, out_GCOPULA.dat, out_CHLSKY.dat

・input (subINPUT.f90)

ILHS の入力ファイルを読み込む．

入力ファイル：inp_ILHS.dat

・mkout (subMKOUT.f90)

ILHS の出力ファイル中のデータ項目（ヘッダー）を書き出す．

・ilhs (subILHS.f90)

ILHS のメインファイル．ILHS によるサンプル点を設定する．

・gamma (subGAMMA.f90)

重み付けパラメータのγを設定する．

・subsort.f90

目的関数の値に応じて，各サンプル区間のランク付けを行う．

・subweight.f90

各サンプル区間のランクに応じて，各サンプル区間の重み付けを行う．

・subcdf.f90

計算結果と重み付けの結果から，次のステップでの累積密度関数を計算する．

・subent.f90

正規化エントロピーを計算する．

・mvnrnd (subMVNRND.f90)

多変量正規分布によるランダムナンバーを発生させる．

・cholesky (subCHLSKY.f90)

コレスキー分解を行う．

・gcopula (subGCOPULA.f90)

正規コピュラ分布によるランダムナンバーを発生させる．

目的関数の計算に関わるプログラム

・input_obj (subOBJ.f90)

ベンチマーク関数用の入力ファイルを読み込む．

入力ファイル：inp_OBJ.dat

・obj (subOBJ.f90)

ベンチマーク関数の計算を行う.

・rotmat (subOBJ.f90), matmat (subOBJ.f90)

ベンチマーク関数用の回転行列を設定する.

・matvec (subOBJ.f90)

ベンチマーク関数が回転を伴う場合，その値を計算する.

その他のプログラム

・subreadc.f90

input ファイルを読む際にコメント行を読み飛ばす.

- 入力ファイル

inp_ILHS.dat

ILHS による最適化計算に使用する入力ファイル

・ndim：最適化問題の次元

・npop：サンプル点の個数

・itermax：最適化計算の最大回数

・ntrial：独立試行の回数

・nobjf：目的関数の個数（=24）

・nwm：重み付けモデルの番号

1：Zipf の法則，2：Canonical 分布

・entmin：重み付けパラメータを設定する際のエントロピーの最小値（=0.00）

・entmax：重み付けパラメータを設定する際のエントロピーの最大値（=0.95）

・nconvg：正規化エントロピーによる収束判定方法に関する指標

0：収束判定なし

1：収束判定あり，全ての主変数で同一の値に収束

2：収束判定あり，主変数によって異なる値に収束

・nsm：サンプル点の発生方法

1：コンパイラ依存の乱数

2：正規コピュラ分布による乱数

・seed_self：乱数のシード

・ngpu：使用する GPU の個数（未使用）

・nhq：1 つの GPU で実行する（未使用）

・ndvc：使用する GPU のワークステーション側でのデバイスナンバー（未使用）

inp_OBJ.dat

ベンチマーク関数の計算に使用する入力ファイル

・f1~f11：ベンチマーク関数 1~11 のパラメータ

・f15：ベンチマーク関数 15 のパラメータ

・f19：ベンチマーク関数 19 のパラメータ

・fbias：ベンチマーク関数のバイアスの値

- 出力ファイル

out_YMIN.dat

各独立試行の終了時における目的関数の値（ymin），及び主変数 *i* の最適解の値（yopt(i)）を，それぞれの目的関数に対して出力する．

out_YMINavg.dat

各独立試行の終了時における
目的関数の値（ymin）の平均値（Avg.），分散（Std.），変動係数（CV）を出力する．尚，この出力ファルのみ，プログラムの実行に要した時間を出力している．

out_YMINall.dat

各独立試行の目的関数の値（ymin），最適解の値（yopt），各サンプル区間の重み（wgt）の変化を，それぞれの目的関数に対して出力する．問題設定によってはファイルサイズが大きくなるため，場合によっては出力行をコメントアウトした方が良い．

out_ENTROPY.dat

各独立試行における，主変数 *i* の正規化エントロピーの値（NormEnt(i)）と正規化エントロピーと 2 値エントロピー関数との差（DifEntMin(i)）を出力する

out_ROT1.dat,

ベンチマーク関数の回転行列の値を出力する．

以下の出力ファイルは，nconvg≠0の場合のみ書き出す．

out_NRNDN.dat

多変量正規分布による乱数を出力する．

out_GCOPULA.dat

正規コピュラ分布による乱数を出力する．

out_CHLSKY.dat

コレスキー分解の結果を出力する．

- 計算コード
- Makefile

```
### TARGET
TARGET = ilhsbmf_v1.11

### OS
ifeq ($(OS), Windows_NT)
  EXEEXT = .exe
endif

### COMPILER
#FC=pgf90
#FC=ifort
FC=gfortran
#FC=g95

### OPTION
# for pgf90
ifeq (${FC}, pgf90)
  OPT=-fastsse -module ./module
#  OPT=-module module -g
#  OPT=-fastsse -module ./module -mcmodel=medium
endif

# for ifort in Linux and Unix environment
ifeq (${FC}, ifort)
  OPT=-fast -module./module
endif

# for gfortran
ifeq (${FC}, gfortran)
  OPT=-O3 -I./module -J./module
#        OPT=-O3    -I./module    -J./module    -Wall    -g    -fbacktrace
-ffpe-trap=zero,overflow,underflow
#  OPT=-O3 -I./module -J./module -Wall -g
endif

# for g95
ifeq (${FC}, g95)
  OPT=-fmod=./module
#  OPT=-O3 -fmod=./module
endif


### LIBRARY
LIB=

code= ¥
  object/mainOPT.o ¥
  object/subCDF.o ¥
  object/subENT.o ¥
  object/subGAMMA.o ¥
  object/subILHS.o ¥
  object/subINPUT.o ¥
  object/subMKOUT.o ¥
  object/subOBJ.o ¥
```

```
  object/subREADC.o ¥
  object/subSORT.o ¥
  object/subWEIGHT.o ¥
  object/subCHLSKY.o ¥
  object/subGCOPULA.o ¥
  object/subMVNRND.o

OBJ= ¥
  object/mdlILHS.o ¥
  object/mdlOBJ.o ¥
  $(code)

ilhs_bmf : $(OBJ)
    $(FC) $(OPT) -o $(TARGET)$(EXEEXT) $(OBJ) $(LIB)
#   $(FC) $(OPT) -o ilhsbmf_v1.10 $(OBJ) $(LIB)

### code ###
object/mainOPT.o : code/mainOPT.f90 ¥
                object/subILHS.o ¥
                object/subINPUT.o ¥
                object/subMKOUT.o
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move mainOPT.o object
  else
    mv mainOPT.o object/
  endif

object/subCDF.o : code/subCDF.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subCDF.o object
  else
    mv subCDF.o object/
  endif

object/subCHLSKY.o : code/subCHLSKY.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subCHLSKY.o object
  else
    mv subCHLSKY.o object/
  endif

object/subENT.o : code/subENT.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subENT.o object
  else
    mv subENT.o object/
  endif

object/subGAMMA.o : code/subGAMMA.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subGAMMA.o object
  else
    mv subGAMMA.o object/
```

```
  endif

object/subGCOPULA.o : code/subGCOPULA.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subGCOPULA.o object
  else
    mv subGCOPULA.o object/
  endif

object/subILHS.o : code/subILHS.f90 ¥
                 object/subCDF.o ¥
                 object/subENT.o ¥
                 object/subGAMMA.o ¥
                 object/subGCOPULA.o ¥
                 object/subMVNRND.o ¥
                 object/subOBJ.o ¥
                 object/subSORT.o ¥
                 object/subWEIGHT.o
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subILHS.o object
  else
    mv subILHS.o object/
  endif

object/subINPUT.o : code/subINPUT.f90 ¥
                  object/subREADC.o
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subINPUT.o object
  else
    mv subINPUT.o object/
  endif

object/subMKOUT.o : code/subMKOUT.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subMKOUT.o object
  else
    mv subMKOUT.o object/
  endif

object/subMVNRND.o : code/subMVNRND.f90 ¥
                   object/subCHLSKY.o
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subMVNRND.o object
  else
    mv subMVNRND.o object/
  endif

object/subOBJ.o : code/subOBJ.f90 ¥
                object/subREADC.o
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subOBJ.o object
  else
```

```
    mv subOBJ.o object/
  endif

object/subREADC.o : code/subREADC.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subREADC.o object
  else
    mv subREADC.o object/
  endif

#object/subSECOND.o : code/subSECOND.f90
#   $(FC) $(OPT) -c $<
#  ifeq ($(OS), Windows_NT)
#   cmd.exe /C move subSECOND.o object
#  else
#   mv subSECOND.o object/
#  endif

object/subSORT.o : code/subSORT.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subSORT.o object
  else
    mv subSORT.o object/
  endif

object/subWEIGHT.o : code/subWEIGHT.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move subWEIGHT.o object
  else
    mv subWEIGHT.o object/
  endif

#object/fctnDERF.o : code/fctnDERF.f90
#   $(FC) $(OPT) -c $<
#  ifeq ($(OS), Windows_NT)
#   cmd.exe /C move fctnDERF.o object
#  else
#   mv fctnDERF.o object/
#  endif

### module ###
object/mdlILHS.o : code/mdlILHS.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move mdlILHS.o object
  else
    mv mdlILHS.o object/
  endif

object/mdlOBJ.o : code/mdlOBJ.f90
    $(FC) $(OPT) -c $<
  ifeq ($(OS), Windows_NT)
    cmd.exe /C move mdlOBJ.o object
  else
    mv mdlOBJ.o object/
```

```
  endif


clean :
  ifeq ($(OS), Windows_NT)
    del module¥*.mod object¥*.o ilhsbmf_v*
  else
    rm -f module/*.mod object/*.o ilhsbmf_v*
  endif
```

- mainOPT.f90

```
Program main
!=============================================================
! Iterative Latin Hypercube Samplings ver.1.8
!=============================================================
  !### write by Goda 2011.
  !### modify by Tanaka 2015.07.21

  !! input parameters
! ndim     : dimension                                (former np)
! npop     : sample size                              (former nn)
! itermax  : maximum iteration steps                    (former nt)
! ntrial   : No. of repeat count for each optimization run (former nr)
! nobjf    : No. of objective function
! seed_self: random seed indicator
! ngpu     : No. of GPU (<= 2)
! nhq      : No. of thread for each GPU (No. of Hyper-Q)

! nwm      : weighting model
! entmin   : Min. value of entropy
! entmax   : Max. value of entropy
! nsm      : sampling model
! ncut     : No. of sample cut
! reent    : convergnece criterion for entropy
! nconvg   : permissible no. of entropy's convergence

  !! output parameters
! ymin_avg : average value of objective functions
! ymin_end : Min. value of objective functions at the end of optimization
! ymin_sd  : standard deviation of objective functions

  !! other parameters
! xpop     : value of sorted sample point [0, 1] (former xlhs)
! yopt     : value of objective function        (former yout)

! nunit    : device number

  !! modules for parameter arrays
  use param_label
  !### modify by Tanaka 2015.03.30 --->
  use input1_module
  use input2_module
  use input3_module
  use gpu_module
!  use param_obj
  !### modify by Tanaka 2015.03.30 <---
  use param_seed
  use param_time

  !! modules for subroutines
  !### add by Tanaka 2015.04.01 --->
  use ilhs_module
  use input_module
  use mkout_module
!  use second_module
  !### add by Tanaka 2015.04.01 <---
  implicit none
```

```
  !! other parameters
  integer(kind = i4) :: i, nunit
!  external log2


  !### add by Tanaka 2015.03.17 --->
  call random_seed(size = seedsize)
  allocate (seed(seedsize))
  call random_seed(get = seed)
  !### add by Tanaka 2015.03.17 <---

  !! open output files
  open (17, file = 'out_YMIN.dat',    status = 'replace')
  open (18, file = 'out_YMINavg.dat', status = 'replace')
  open (19, file = 'out_YMINall.dat', status = 'replace')
  open (20, file = 'out_ENTROPY.dat', status = 'replace')

!  open (21, file = 'out_RANGE.dat',   status = 'replace')
  !### add by Tanaka 2015.04.12 --->
  open (22, file = 'out_ROT1.dat',    status = 'replace')
  open (23, file = 'out_NRNDM.dat',   status = 'replace')
  open (24, file = 'out_MNRNDM.dat',  status = 'replace')
  open (25, file = 'out_GCOPULA.dat', status = 'replace')
  open (26, file = 'out_CHLSKY.dat',  status = 'replace')
  !### add by Tanaka 2015.04.12 <---

  !! set start time
  !### modify by Tanaka 2015.07.21 --->
  call cpu_time(tzero)
!  call second (tzero)
  !### modify by Tanaka 2015.07.21 <---

  call date_and_time (exedate, exetime, exezone, exeiv)
  nunit = 18
  write (nunit, '(a15, i5, 2(a1, i2.2), 3(a1, i2.2))') &
   "Start Time :", exeiv(1), "/", exeiv(2), "/", exeiv(3), " ", &
                exeiv(5), ":", exeiv(6), ":", exeiv(7)

  !! read input files
  call input

  !### add by Tanaka 2015.03.17 --->
  do i = 1, seedsize
    seed(i) = seed(i) + seed_self
  end do
  call random_seed(put = seed)
  !### add by Tanaka 2015.03.17 <---

  !! set time to read input
  !### modify by Tanaka 2015.07.21 --->
  call cpu_time(elt1)
!  call second (elt1)
  !### modify by Tanaka 2015.07.21 <---
  elt1 = elt1 - tzero

  !! write header lists of output files
  call mkout
```

```
  !! main part of program
  call ilhs

  !! calculate elapsed time
  !### modify by tanaka 2015.02.11 --->
  !### modify by Tanaka 2015.07.21 --->
  call cpu_time(elt)
!  call second (elt)
  !### modify by Tanaka 2015.07.21 <---
  elt  = elt - tzero
  eltc = elt - elt1

  !! write elapsed time
  nunit = 18
  write (nunit, *)
  write (nunit, '(a30)') " End of ILHS Simulation Run "
  write (nunit, '(a25, f9.2, a5)') " ------ Elapsed Time = ", elt,  " sec "
  write (nunit, '(a25, f9.2, a5)') " -- Calculation Time = ", eltc, " sec "
  write (nunit, '(a25, f9.2, a5)') " --- Data Input Time = ", elt1, " sec "

  call date_and_time (exedate, exetime, exezone, exeiv)
  write (nunit, '(a13, i5, 2(a1, i2.2), 3(a1, i2.2))') &
   "End time :", exeiv(1), "/", exeiv(2), "/", exeiv(3)," ", &
               exeiv(5), ":", exeiv(6), ":", exeiv(7)

  !! close output files
  do nunit = 17, 20
!  do nunit = 17, 21
    close (nunit)
  end do
  !### add by Tanaka 2015.04.12 --->
  close (22)
  if ( nsm == 2 ) then
    do nunit = 23, 26
      close (nunit)
    end do
  end if
  !### add by Tanaka 2015.04.12 <---
  !### modify by Tanaka 2015.02.11 <---

  stop

end program main


!============================================================
!  function log2(x)
!    implicit none
!
!    log2 = log(x) / log(2.d0)
!    return
!  end function log2
!============================================================
```

- mdlILHS.f90

```
!### write by Tanaka 2015.01.28
!### modify by Tanaka 2015.07.08

module param_label
  implicit none

  integer, parameter :: i4 = 4, &
                        r8 = 8
end module param_label


module input1_module
  use param_label
  implicit none

  integer(kind = i4) :: ndim, npop, &
                        ntrial, nobjf, itermax, &
                        itrial, iobjf, iter
end module input1_module

module input2_module
  use param_label
  implicit none

  integer(kind = i4) :: nwm, nsm
  real   (kind = r8) :: entmin, entmax
end module input2_module

module input3_module
  use param_label
  implicit none

  integer(kind = i4) :: ncut, nconvg
  real   (kind = r8) :: reent
end module input3_module


module gpu_module
  use param_label
  implicit none

  integer(kind = i4) :: ngpu, nhq, ndvc(2)
end module gpu_module


module param_obj
  use param_label
  implicit none

  real   (kind = r8), allocatable :: xpop(:,:), yopt(:)

contains

  subroutine alloc_obj
    use input1_module, only: ndim, npop
    implicit none
```

```
    allocate( xpop(npop, ndim), yopt(npop) )
  end subroutine alloc_obj
end module param_obj


module param_ymin
  use param_label
  implicit none

  real   (kind = r8) :: ymin
end module param_ymin


module param_rank
  use param_label
  use input1_module, only: npop
  implicit none

  integer(kind = i4), allocatable :: irank(:)

contains

  subroutine alloc_rank
    implicit none

    allocate( irank(npop) )
  end subroutine alloc_rank

end module param_rank


module param_lhs
  use param_label
  use input1_module, only: npop
  implicit none

  real   (kind = r8), allocatable :: xbdy(:), xpop_new(:), xbdy_new(:)

contains

  subroutine alloc_lhs
    implicit none

    allocate( xbdy(npop), xpop_new(npop), xbdy_new(npop) )
  end subroutine alloc_lhs

end module param_lhs


module param_wgt
  use param_label
  use input1_module, only: npop
  implicit none

  real   (kind = r8)            :: sum_wgt1
  real   (kind = r8), allocatable :: wgt(:), wgt_mat(:)
```

```
contains

  subroutine alloc_wgt
    implicit none

    allocate( wgt(npop), wgt_mat(npop) )
  end subroutine alloc_wgt

end module param_wgt


module param_ent
  use param_label
  use input1_module, only: ndim, npop
  implicit none

  real   (kind = r8)            :: gam1, beta, ent_yopt, ent_tmp
  real   (kind = r8), allocatable :: xside(:,:)

contains

  subroutine alloc_ent
    implicit none

    allocate( xside(npop, ndim) )
  end subroutine alloc_ent

end module param_ent


!### add by Tanaka 2015.07.08 --->
module param_dent
  use param_label
  use input1_module, only: ndim
  implicit none

  integer(kind = i4)            :: ndent
  real   (kind = r8), allocatable :: ent_xpop(:), bef_xpop(:), &
                                dent_xpop(:), dent_min(:), dent_lim(:)

contains

  subroutine alloc_dent1
    implicit none

    allocate( ent_xpop(ndim), bef_xpop(ndim), &
              dent_xpop(ndim), dent_min(ndim) )
  end subroutine alloc_dent1

  subroutine alloc_dent2
    implicit none

    allocate( dent_lim(ndim) )
  end subroutine alloc_dent2

end module param_dent
!### add by Tanaka 2015.07.08 <---
```

```
module param_others
  use param_label

  use input1_module
  implicit none

  integer(kind = i4), allocatable :: nent(:), &
                                norder1(:,:), norder2(:)
  real   (kind = r8), allocatable :: xorder(:,:), &
                                xbdy_mat(:,:), xpop_mat(:,:), &
                                ymin_end(:), xopt(:), x_dum(:)

contains

  subroutine alloc_others
    implicit none

    allocate( nent(ndim), &
             norder1(npop, ndim), norder2(npop) )
    allocate( xorder(npop, ndim), &
             xpop_mat(npop, ndim), xbdy_mat(npop, ndim), &
             xopt(ndim), ymin_end(ntrial), x_dum(0:npop) )
  end subroutine alloc_others

end module param_others


module param_copula
  use param_label
  implicit none

  integer(kind = i4)            :: nrows, ncols
  real   (kind = r8)            :: rho, tau
  real   (kind = r8), allocatable :: u(:,:)
  real   (kind = r8), allocatable :: a(:,:)
  real   (kind = r8), allocatable :: rdum_gc(:,:,:)

contains

  subroutine alloc_copula
    use input1_module, only: itermax, ndim, npop
    implicit none

    allocate( u(nrows, ncols) )
    allocate( a(ncols, ncols) )
    allocate( rdum_gc(npop, ndim, itermax) )
  end subroutine alloc_copula

end module param_copula


module param_seed
  use param_label
  implicit none

  integer(kind = i4), allocatable :: seed(:)
  integer(kind = i4)            :: seedsize
```

```
  integer(kind = i4)            :: seed_self
end module param_seed


module param_time
  use param_label
  implicit none

  !! variables for elapsed time measurement
  real(kind = r8)    :: tzero, elt, elt1, eltc
  character(len=10)  :: exedate, exetime, exezone
  integer(kind = i4) :: exeiv(1:8)
   end module param_time
```

- md|OBJ.f90

```
!### write by Tanaka 2015.01.28
!### modify by Tanaka 2015.03.17

module param_bmf

  use param_label
  implicit none

  real(kind = r8) :: f1(1000), f2(1000), f3(1000), f4(1000), f5(1000), &
                     f6(1000), f7(1000), f8(1000), f9(1000), f10(1000), &
                     f11(1000), f15(1000), f19(1000), fbias(6)

  real(kind = r8), allocatable :: rot1(:,:), rot2(:,:) !### add by Tanaka
2015.03.17
   end module param_bmf
```

- subCDF.f90

```
module cdf_module
  implicit none
contains

  subroutine cdf

    !! cumulative distribution function
    !### write by Goda 2011.
    !### modify by tanaka 2015.07.24

    !### this subroutine calculate lower and upper limits of primary variables
    !### in next iteration step depending on values of objective function

    !! input parameters
!   xbdy    : upper limit of xij
!   wgt_mat : weighting parameter
!   sum_wgt1: normalizition constant (sum of wgt_mat)
!   npop    : number of sampling points

    !! output parameters
!   xpop_new: value of sampling point (xij) in next iteration step
!   xbdy_new: upper limit of xij in next iteration step

    !! other parameters
!   rdum    : uniform random number
!   xdum_tmp: upper limit of xij
!   c2      : alpha
!   ctab    : sum of weighting parameters


    !! module for parameter arrays
    use param_label
    !### add by Tanaka 2015.03.26 --->
    use input1_module, only: npop
    use param_lhs
    use param_wgt,    only: wgt_mat, sum_wgt1
    !### add by Tanaka 2015.03.26 <---
    implicit none

    !! other parameters
    integer(kind = i4) :: i, i1, i2
    !### modify by Tanaka 2015.02.18 --->
    real  (kind = r8) :: rdum, &
!    real  (kind = r8) :: rdum_mat(npop), &
                         c2, ctab(0:npop), xdum_tmp(0:npop)
    !### modify by Tanaka 2015.02.18 <---


    !! initialize length of sample space
    ctab(0) = 0.d0
    do i = 1, npop
      ctab(i) = ctab(i-1) + wgt_mat(i) / sum_wgt1
    end do
    xdum_tmp(0)       = 0.d0
    xdum_tmp(npop)    = 1.d0
    xdum_tmp(1:npop-1) = xbdy(1:npop-1)
```

```
!###   do i = 1, npop-1
!###     xdum_tmp(i) = xbdy(i)
!###   end do

    !! calculate values of sampling points in next iteration step
    do i1 = 1, npop
      call random_number(rdum)
      !### modify by Tanaka 2015.02.18 --->
!      c2 = (dble(i1) - rdum_mat(i1)) / dble(npop)
      c2 = (dble(i1) - rdum) / dble(npop)
      !### modify by Tanaka 2015.02.18 <---
      do i2 = 1, npop
        if ( c2 >= ctab(i2-1) .and. c2 < ctab(i2) ) then
          xpop_new(i1) = xdum_tmp(i2-1) + (c2 - ctab(i2-1)) &
         &           / (ctab(i2) - ctab(i2-1))*(xdum_tmp(i2) - xdum_tmp(i2-1))
          goto 100
        end if
      end do
  100 continue
    end do

    !! calculate upper limit of sampling point in next iteration step
    do i1 = 1, npop - 1
      c2 = dble(i1) / dble(npop)
      do i2 = 1, npop
        if ( c2 >= ctab(i2-1) .and. c2 < ctab(i2) ) then
          xbdy_new(i1) = xdum_tmp(i2-1) + (c2 - ctab(i2-1)) &
         &           / (ctab(i2) - ctab(i2-1))*(xdum_tmp(i2) - xdum_tmp(i2-1))
          goto 200
        end if
      end do
  200 continue
    end do

    xbdy_new(npop) = 1.d0 !### add by Tanaka 2015.02.13

    return

  end subroutine cdf

end module cdf_module
```

- subCHLSKY.f90

```
module cholesky_module
  implicit none
contains

  subroutine cholesky

    !! This subroutine provide Cholesky decomposition of matrix a

    !### write by Tanaka 2015.02.23
    !### modify by Tanaka 2015.04.15

    !! input parameters
!   a       : input matrix

    !! output parameters
!   a       : output matrix

    use param_label
    use param_copula, only: ncols, a !### modify by Tanaka 2015.04.15
    implicit none

    !! formal vars

    !! local vars
    integer(kind = i4) :: j   ! iteration counter


    !! begin loop
    chol: do j = 1, ncols

      !! perform diagonal component
      a(j, j) = sqrt(a(j, j) - dot_product(a(j, 1:j-1), a(j, 1:j-1)))

      !! perform off-diagonal component
      !### modify by Tanaka 2015.04.14 --->
      a(1:j-1, j)    = 0.d0
      a(j+1:ncols, j) = (a(j+1:ncols, j) &
     &              - matmul(a(j+1:ncols, 1:j-1), a(j, 1:j-1))) / a(j, j)
!     a(j+1:ndim, j) = (a(j+1:ndim, j) &
!    &             - matmul(a(j+1:ndim, 1:j-1), a(j, 1:j-1))) / a(j, j)
!!      if ( j < ndim ) a(j+1:ndim, j) = (a(j+1:ndim, j) &
!!     &                          - matmul(a(j+1:ndim, 1:j-1), a(j, 1:j-1))) /
a(j, j)
      !### modify by Tanaka 2015.04.14 <---

    end do chol

  end subroutine cholesky

end module cholesky_module
```

- subENT.f90

```
module ent_module
  implicit none
contains

  subroutine ent (pdf)

    !! This subroutine calculates Normalized Entropy (bit: binary logarithm)

    !### write by Tanaka 2014.
    !### modify by Tanaka 2015.07.24


    !! input parameters
!   npop    : number of sampling points
!   pdf     : probability density function (0, 1]

    !! output parameters
!   ent1    : normalized entropy
!
    !! other parameters
!   a1      : summation of probability density function
!   a2      : summation of noermalized entropy

    !! modules for parameter arrays
    use param_label
    !### add by Tanaka 2015.03.26 --->
    use input1_module, only: npop
    use param_ent,     only: ent1 => ent_tmp
    !### add by Tanaka 2015.03.26 <---
    implicit none

    !! input & output parameters
    real  (kind = r8) :: pdf(npop)

    !! other parameters
    integer(kind = i4) :: i
    real  (kind = r8) :: a1, a2, xnpop

    a1   = 0.d0
    a2   = 0.d0
    xnpop = dble(npop)

    !### modify by Tanaka 2014.04.08 --->
!    do i = 1, npop
!      a1 = a1 + 1.d0 / pdf(i)**gam
!    end do
    a1 = 1.d0
    !### modify by Tanaka 2014.04.08 <---

    do i = 1, npop
      a2 = a2 + pdf(i) / a1 * log(pdf(i) / a1) / log(2.d0)
    end do

    ent1 = - a2 / log(xnpop) * log(2.d0)

    return
```

```
  end subroutine ent

end module ent_module
```

- subGAMMA.f90

```
module gamma_module
  implicit none
contains

  subroutine gamma

    !### write by Tanaka 2015.01.25
    !### modify by Tanaka 2015.07.24

    !### This subroutine sets the exponent parameter of Zipf's law, gamma

    !! input parameters
!   npop    : number of sampling points

    !! output parameters
!   ent_yopt: normalized entropy
!   gam1    : gamma, exponential constant in Zipf's law
!
    !! other parameters
!   z          :
!   gam     : gamma, exponential constant in Zipf's law
!   a1      :
!   a2      :

    !! modules for parameter arrays
    use param_label
    !### add by Tanaka 2015.07.17 --->
    use input1_module, only: npop
    use input2_module, only: entmax
    use param_ent,     only: gam1, ent_yopt
    !### add by Tanaka 2015.07.17 <---
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j
    real   (kind = r8) :: z, gam, a1, a2, xj, xn

    z       = 0.d0
    ent_yopt = 0.d0
    gam     = 0.d0

    !### modify by Tanaka 2015.01.21 --->
    do i = 1000, 1, -1
      if ( ent_yopt >= entmax ) then
!###    do i = 1, 1000
!###      if (ent_yopt >= entmin .and. ent_yopt <= entmax) then
    !### modify by Tanaka 2015.01.21 <---
       gam1 = gam
      else
        gam = dble(i) / 100.d0 !gam: index of zipf's law (gamma)
        a1 = 0.d0
        a2 = 0.d0
        do j = 1, npop
          xj = dble(j)
          a1 = a1 + log(xj) / xj**gam
!          a1 = a1 + log(xi) / log(2.d0) / xj**gam
```

```
         a2 = a2 + 1.d0 / xj**gam
       end do
       xn = dble(npop)
       z  = a2
       ent_yopt = (gam*a1/a2 + log(a2)) / log(xn) !ent: normalized entropy (h)
!       ent_yopt = (gam*a1/a2 + log(a2)/log(2.d0)) / (log(xn)/log(2.d0))
     end if
   end do
   write (*, *) "gamma            = ", gam1
   write (*, *) "Normalized Entropy = ", ent_yopt
!###   z = 0.0
!###   gam = 0.0
!### 10 continue
!###   gam = gam + 1.0e-5
!###   a1 = 0.0
!###   a2 = 0.0
!###   do i = 1, npop
!###     xi = real(i)
!###     a1 = a1 + alog(xi) / alog(2.0) / real(i)**gam
!###     a2 = a2 + 1.0 / real(i)**gam
!###   end do
!###   xn = real(npop)
!###   ent_yopt = (gam*a1/a2 + alog(a2)/alog(2.0)) / (alog(xn)/alog(2.0))
!###   if ( entmax .gt. ent_yopt ) goto 20
!###   goto 10
!### 20 continue
!###   write (*, *) gam

   return

  end subroutine gamma

end module gamma_module
```

- subGCOPULA.f90

```
module gcopula_module
  implicit none
contains

  subroutine gcopula

    !! This subroutine generates random number depending on gaussian copula

    !### write by Tanaka 2015.02.23
    !### modify by Tanaka 2015.07.24

    !! input parameters
!   rho     : Sperman's rho
!   tau     : Kendall's tau

    !! output parameters
!   z       : random number depending on standard normal distribution
!   x       :
!   u       : random number depending on gausian copula

    !! other parameters
!   derf    : error function in double precision

    !! modules for parameter arrays
    use param_label
    use param_copula !### modify by Tanaka 2015.04.14
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, k
    real  (kind = r8) :: pi, fac, rndm1, rndm2, z_tmp1
!    real  (kind = r8) :: pi, fac, rndm1, rndm2, z_tmp1, z_tmp2
    real  (kind = r8) :: z(ncols), x(ncols)
    real  (kind = r8) :: erf
    integer(kind = i4), parameter :: nunit1 = 23, &
                                     nunit2 = 24, &
                                     nunit3 = 25   !### add by Tanaka 2015.04.1

    pi = atan(1.d0) * 4.d0

    !! Set the number of variables and variates
!    nrows =   4
!    ncols = 1000
!    rho   =  0.5d0
!    tau   = 0.75d0

    !! open input & output files
!    open (40, file = 'out_NRNDM.dat',   status = 'replace')
!    open (41, file = 'out_MNRNDM.dat',  status = 'replace')
!    open (42, file = 'out_GCOPULA.dat', status = 'replace')

    !### add by Tanaka 2015.04.15 --->
    write (nunit1, '(a6, i3, a1)') "z(", ncols, ")"
    write (nunit2, '(a6, i3, a1)') "x(", ncols, ")"
    write (nunit3, '(a12, 2(i3, a1))') "u(", nrows, ",", ncols, ")"
    !### add by Tanaka 2015.04.15 <---
```

```
   !! Input the upper triangle portion of the covariance matrix
!    call mvnrnd !### modify by Tanaka 2015.04.11

   !! main part of program
   do k = 1, nrows
     x = 0.d0

     !! Standard Normal Distribution Random Number, Z=(Z1,...,Zd)
     !! Z~Nd(0,1)
     do j = 1, ncols
       call random_number(rndm1)
       fac    = sqrt(-2.d0 * log(rndm1))
!       fac    = sqrt(-2.d0 * log(rndm1))
       call random_number(rndm2)
       z_tmp1 = fac * cos(2.d0 * pi * rndm2)
!       z_tmp2 = fac * sin(2.d0 * pi * rndm2)
       z(j)   = z_tmp1
     end do

     !! X = AZ
     do j = 1, ncols
       do i = 1, ncols
         x(i) = x(i) + a(i, j) * z(j)
       end do
     end do

     !! Gaussian Copula Random Number, U=(u1,...,ud)
     !! U = (Phi(X1),...,Phi(Xd))
     do i = 1, ncols
       u(k, i) = (1.d0 + erf(x(i) / sqrt(2.d0))) / 2.d0
!       u(i) = (1.d0 + erf(x(i) / sqrt(2.d0))) / 2.d0
     end do

!     do i = nunit1, nunit2
!       write (i, '(3i5)', advance = 'no') iobjf, iter, k
!     end do
     do i = 1, ncols
       write (nunit1, '(f10.5)', advance = 'no') z(i)
       write (nunit2, '(f10.5)', advance = 'no') x(i)
     end do
     do i = nunit1, nunit2
       write (i, *)
     end do

   end do

   !### add by Tanaka 2015.04.12 --->
   do i = 1, nrows
     do j = 1, ncols
       write (nunit3, '(f10.5)', advance = 'no') u(i, j)
     end do
     write (nunit3, *)
   end do

   do i = nunit1, nunit3
     write (i, *)
   end do
```

```
    !### add by Tanaka 2015.04.12 <---

    !! close input & output files
!    do i = 40, 42
!      close (i)
!    end do

    return

  end subroutine gcopula

end module gcopula_module
```

- subILHS.f90

```
module ilhs_module
  implicit none
contains

  subroutine ilhs

    !### write by Tanaka 2015.03.30
    !### modify by Tanaka 2015.07.24

    !! input parameters
!   xpop    : value of sorted sample point [0, 1] (former xlhs)

    !! output parameters
!   xopt    : value of sorted sample point [0, 1]
!   yopt    : value of objective function (former yout)

!   ymin_avg : average value of objective functions
!   ymin_end : Min. value of objective functions at the end of optimization
!   ymin_sd  : standard deviation of objective functions

    !! other parameters

!   norder1  : order of sampling point                  [1, npop] (former lhs)
!   norder2  : order of sampling point                 [1, npop] (former lhs2)
!   xpop_mat : value of sampling point                  [0, 1] (former xlhs_mat,
xc)
!   xbdy_mat : boundary value of sample space             (0, 1] (former xc2)
!   xbdy     : boundary value of sample space            (0, 1] (former xdum2)
!   xpop_new : value of sample point in next step       [0, 1] (former xlhs_new,
xdum3)
!   xbdy_new : boundary value of sample space in next step    (0, 1] (former xdum4)
!   xorder   : random number to decide sample order        [0, 1] (former xdum)
!   wgt     : weighting depending on objective function value      (former ydum)
!   wgt_mat  : weighting depending on objective function value       (former
ydum2)
!   sum_wgt1 : sum of weighting depending on objective function value (former c1)
!   ymin    : Min. value of objective function
!   ymin_tmp : Min. value of objective function for each step
!   irank   : rank of objective function value          [1, npop]
!   xside    : length of sample space side                  [0, 1] (former
range_pv)
!   ent_yopt : normalized entropy of yopt                          (former
ent_yout, ent1)
```

```
!   ent_xpop : normalized entropy of xpop                                (former
ent_xlhs, ent2)
!   bef_xpop : binary entropy function of xpop
!   dent_xpop: deference between normalized entropy and binary entropy function
of xpop
!   dent_min : Min. value of deference between normalized entropy and binary entropy
function of xpop
!   dent_lim : lower limit value of deference between normalized entropy and binary
entropy function of xpop
!   gam1    : gamma, exponential constant in Zipf's law
!   beta    : exponential constant in canonical distribution
!   nent    : No. of entropy convergence
!   rdum    : random number

!   ndim    : dimension                                          (former np)
!   npop    : sample size                                        (former nn)
!   itermax  : maximum iteration steps                             (former nt)
!   ntrial   : No. of repeat count for each optimization run         (former nr)
!   nobjf   : No. of objective function
!   seed_self: random seed indicator
!   ngpu    : No. of GPU (<= 2)
!   nhq     : No. of thread for each GPU (No. of Hyper-Q)

!   nwm     : weighting model
!   rho     : Sperman's rho
!   tau     : Kendall's tau
!   nrows   : No. of rows of random numbers depending on copula
!   ncols   : No. of columns of random numbers depending on copula
!   entmin   : Min. value of entropy
!   entmax   : Max. value of entropy
!   nsm     : sampling model
!   ncut    : No. of sample cut
!   reent   : convergnece criterion for entropy
!   nconvg   : permissible no. of entropy's convergence

!   nunit   : device number

   !! modules for parameter arrays
   use param_label
   use input1_module
   use input2_module
   use input3_module
   use gpu_module

   use param_obj     !### add by Tanaka 2015.04.08
   use param_ymin
   use param_rank
   use param_lhs
   use param_wgt
   use param_ent
   use param_dent    !### add by Tanaka 2015.07.08
   use param_others   !### add by Tanaka 2015.03.30
   use param_copula   !### add by Tanaka 2015.04.11

   !! modules for subroutines
   !### add by Tanaka 2015.04.01 --->
   use cdf_module
   use ent_module
```

```
    use gamma_module
    use obj_module
    use sort_module
    use weight_module
    !### add by Tanaka 2015.04.01 <---
    use gcopula_module !### add by Tanaka 2015.04.11
    use mvnrnd_module  !### add by Tanaka 2015.04.13
    implicit none

    !! other parameters
    !### modify by Tanaka 2015.03.30 --->
    integer(kind = i4) :: i, j, im, nunit
    real  (kind = r8) :: rdum
    real  (kind = r8) :: ymin_tmp, ymin_avg, ymin_std, ymin_cv
    !### modify by Tanaka 2015.03.30 <---


    !! read input files for objective functions
    call input_obj    !### modify by Tanaka 2015.03.26

    !! allocate variables
    !### moidfy by Tanaka 2015.03.30 --->
    call alloc_rank
    call alloc_lhs
    call alloc_wgt
    call alloc_ent
    call alloc_dent1   !### add by Tanaka 2015.07.08
    call alloc_others
    call alloc_obj    !### add by Tanaka 2015.04.08
    !### modify by Tanaka 2015.03.30 <---


    !! main part of program
    !! set zipf's law parameter, gamma
    if ( nwm == 1 ) call gamma !### modify by Tanaka 2015.03.26

    !! initialize variables
    ymin_end(1:ntrial) = 0 !### add by Tanaka 2015.03.17
    !! set random numbers using copula function
    !### modify by Tanaka 2015.04.15 --->
    if ( nsm == 2 ) then
!     rho = 0.50d0
!     tau = 0.75d0
      nrows = itermax
      ncols = npop
      call alloc_copula
      call mvnrnd
      do i = 1, ndim
        call gcopula
        do j = 1, itermax
          rdum_gc(:,i,j) = u(j,:)
        end do
      end do
    end if
    !### modify by Tanaka 2015.04.15 <---

    !! optimization start
    !! set values of sampling parameters
```

```
   do iobjf = 1, nobjf
     do itrial = 1, ntrial
       write (*, *) iobjf, itrial

       !### modify by Tanaka 2015.01.15 --->
       ymin  = 1.d+100
!       ymin2 = 1.d+100
       if ( nwm == 2 ) then
         ent_yopt = entmax
       end if
!       nxmin = nxmin_ini
!       nxmax = nxmax_ini
       !### modify by Tanaka 2015.01.15 <---
       nent(1:ndim)  = 0 !### add by Tanaka 2014.04.08
       !### add by Tanaka 2015.06.18 --->
       ent_xpop = 1.d0
       bef_xpop = 0.d0
       dent_min = 1.d0
       !### add by Tanaka 2015.06.18 <---

       !! construction of initial lhs
       do j = 1, ndim
         do i = 1, npop
           call random_number(rdum)
           xpop_mat(i, j) = (dble(i) - rdum) / dble(npop)
           xbdy_mat(i, j) = dble(i) / dble(npop)
         end do
!         do i = 1, npop
!           xpop_mat(i, j) = xdum(i)
!           xbdy_mat(i, j) = xbdy(i)
!         end do
       end do

       !! calculate benchmark functions
       do iter = 1, itermax

         !### modify by Tanaka 2015.04.14 --->
         !! set value of LHS
         if ( nsm == 1 ) then
           do j = 1, ndim
             do i = 1, npop
               call random_number(rdum)
               xorder(i, j) = rdum
             end do
           end do
         else if ( nsm == 2 ) then
           xorder(:,:) = rdum_gc(iter,:,:)
         end if
         !### modify by Tanaka 2015.04.14 <---

         !! set order of LHS
         do j = 1, ndim !### add by Tanaka 2015.04.11
           do i = 1, npop
             norder2(i) = 1
             do im = 1, npop
               if ( xorder(i, j) > xorder(im, j) ) then
                 norder2(i) = norder2(i) + 1
               end if
```

```
            end do
            norder1(i, j) = norder2(i)
            xpop(i, j)   = xpop_mat(norder2(i), j)
          end do
        end do

        !! Calculation of Objective Functions
        call obj !### modify by Tanaka 2015.04.08

        !! npop: number of sample
        do i = 1, npop
!          do j = 1, 2*nw
!            idumx(j) = nlhs(i,j)
!          end do
!          do k = 1, nrot
!            xsum(k) = 0.
!            do j = 1, nw
!              xw = real(nw)
!              if( icnt1 < cntbfr ) then
!                xrate(j, k) = (xmax - xmin) / xw
!              else if( j == 1 ) then
!                xrate(j, k) = xmin
!                 + (xmax - xmin) * xpop(i,2*nw+j*k)
!                xsum(k) = xsum(k) + xrate(j,k)
!              else if( j > 1 .and. j < nw ) then
!                xrate(j, k) = xmin
!                 + (xmax - xmin - xsum(k)) * xpop(i,2*nw+j*k)
!                xsum(k) = xsum(k) + xrate(j, k)
!              else
!                xrate(j, k) = xmin + (xmax - xmin - xsum(k))
!              end if
!            end do
!          end do

          !! TOUGH2 Simulation
!          call init(idumx, xrate)

          !! Read TOUGH2 Output
!          open (5, file = 'mass', status = 'old')
!          j = 0
!          read (5, '()')  !### add by tanaka 2012.10.25
!       10 j = j + 1
!          read (5, *, end = 100) time(i), co2m(i), co2r(i), co2d(i)
!          goto 10
!      100 continue
!          close (5)
!          co2i(i) = co2m(i) + co2r(i) + co2d(i)
!          yopt(i) = 1. - (co2r(i) + co2d(i)) / co2a

          !! optimum or not
          if ( yopt(i) < ymin ) then
!            ymin2 = yopt(i)
            ymin = yopt(i)
            do j = 1, ndim
              xopt(j) = xpop(i, j)
            end do
          end if
          irank(i) = i
```

```
        end do

        !### modify by Tanaka 2015.05.16 --->
        ymin_tmp = minval(yopt, mask = yopt > 0)

        nunit = 19
        !### modify by Tanaka 2015.02.11 --->
        write (nunit, '(3i5, 2e15.7)', advance = 'no') &
       & iobjf, itrial, iter, ymin, ymin_tmp
        !### modify by Tanaka 2015.02.11 <---

        do j = 1, ndim
          write (nunit, '(e15.7)', advance = 'no') xopt(j)
        end do
        do j = 1, npop
          write (nunit, '(e15.7)', advance = 'no') wgt(j)
        end do
        if ( nwm == 1 ) then
          write (nunit, '(2e15.7)', advance = 'no') ent_yopt, gam1
        else if ( nwm == 2 ) then
          write (nunit, '(2e15.7)', advance = 'no') ent_yopt, beta
        end if
        write (nunit, *)

        !### modify by Tanaka 2015.02.11 --->
        nunit = 20
        write (nunit, '(3i5)', advance = 'no') iobjf, itrial, iter
        !### modify by Tanaka 2015.02.11 <---
        !### modify by Tanaka 2015.05.16 <---

        !! Update Objective Function Value Using TOUGH2
!        do i = 1, npop
!          if ( yopt(i) < ymin ) then
!            ymin = yopt(i)
!            tmin = time(i)
!            cmmin = co2m(i)
!            crmin = co2r(i)
!            cdmin = co2d(i)
!            cimin = co2i(i)

!            do j = 1, 2*nw
!              nopt(j) = nlhs(i,j)
!            end do

!            do k = 1, nrot
!              do j = 1, nw
!                xopt(j,k) = xrate(j,k)
!              end do
!            end do
!            icnt1 = 0
!          else if ( yopt(i) == ymin ) then
!            icnt1 = icnt1 + 1
!          end if
!          irank(i) = i
!        end do

        !! rank transformation
```

```
          call sort !### modify by Tanaka 2015.04.08

          !### modify by Tanaka 2015.06.18 --->
          do j = 1, ndim
            !! Binary Entropy Function
            bef_xpop(j)  = - (xopt(j) * log(xopt(j)) + (1.d0 - xopt(j)) &
!           bef_xpop(j)  = - (xopt(j) * log(xopt(j)) + (1.d0 - xopt(j)) &
           &           * log(1.d0 - xopt(j))) / log(dble(npop))
            !! Difference between Normalized Entropy and Normalized BEF
            dent_xpop(j) = ent_xpop(j) - bef_xpop(j)

            if ( dent_xpop(j) < dent_min(j) .and. dent_xpop(j) >= 0.d0 ) then
              dent_min(j) = dent_xpop(j)
            end if
          end do

          nunit = 20
          do j = 1, ndim
            write (nunit, '(e15.7)', advance = 'no') ent_xpop(j)
          end do
!          do j = 1, ndim
!            write (nunit, '(e15.7)', advance = 'no') bef_xpop(j)
!          end do
!          do j = 1, ndim
!            write (nunit, '(e15.7)', advance = 'no') dent_xpop(j)
!          end do
          do j = 1, ndim
            write (nunit, '(e15.7)', advance = 'no') dent_min(j)
          end do
          write (nunit, *)
          !### modify by Tanaka 2015.06.18 <---

          !! Set Weight Parameter
          call weight !### modify by Tanaka 2015.04.08

          !! set value and order of new LHS
          do j = 1, ndim
            do i = 1, npop
!              xdum(norder1(i, j))   = xpop(i, j)
              xbdy(i)             = xbdy_mat(i, j)
              wgt_mat(norder1(i, j)) = wgt(i)
            end do

            !! Set Cumulative Density Function
            call cdf !### modify by Tanaka 2015.03.26

            !! calculate probability density function (pdf, [0,1]) to calculate
Normalized Entropy
            !### modify by Tanaka 2015.02.13 --->
            x_dum(0)     = 0.d0
            x_dum(1:npop) = xbdy_new(1:npop)
            do i = 1, npop
              xpop_mat(i, j) = xpop_new(i)
              xbdy_mat(i, j) = xbdy_new(i)
              xside(i, j)   = x_dum(i) - x_dum(i-1)
            end do
            !### modify by Tanaka 2015.02.13 <---
```

```
          !! calculate Normalized Entropy
          call ent (xside(1:npop, j)) !### modify by Tanaka 2015.03.26
          ent_xpop(j) = ent_tmp
!          ent_xpop(iter, j) = ent_tmp
        end do

        do j = 1, ndim
          if ( dent_min(j) >= dent_lim(j) ) exit
          goto 1111
        end do

      end do


     !### modify by tanaka 2015.02.13 --->
       !! write output files
  1111 ymin_end(itrial) = ymin
!       ymin_end(itrial) = ymin

       nunit = 17
       write (nunit, '(2i5, e15.7)', advance = 'no') iobjf, itrial, ymin !###
modify by Tanaka 2015.02.11

       do i = 1, ndim
         write (nunit, '(e15.7)', advance = 'no') xopt(i)
       end do
       write (nunit, *)

       write (19, *)
       write (20, *)

     end do !### add by Tanaka 2015.02.11

     write (17, *)
     ymin_avg = sum(ymin_end) / ntrial
     ymin_std = sqrt(sum(ymin_end**2) / ntrial - ymin_avg**2) !### add by Tanaka
2015.02.11
     ymin_cv  = ymin_std / ymin_avg                          !### add by Tanaka
2015.03.25

     write (18, '(i6, 3e15.7)', advance ='no') iobjf, ymin_avg, ymin_std, ymin_cv

     do i = 18, 20
       nunit = i
       write (nunit, *)
     end do
     if ( ncut /= 0 ) write (21, *) !### add by Tanaka 2015.04.13
     !### modify by Tanaka 2015.02.13 <---

   end do

   return

  end subroutine ilhs

end module ilhs_module
```

- subINPUT.f90

```
module input_module
  implicit none
contains

  subroutine input

    !### write by Tanaka 2015.01.25
    !### modify by Tanaka 2015.07.08

    !! output parameters
!   ndim     : dimension (former np)
!   npop     : sample size (former nn)
!   itermax  : maximum iteration steps (former nt)
!   ntrial   : No. of repeat count for each optimization run (former nr)
!   nobjf    : No. of objective function
!   seed_self: random seed indicator
!   ngpu     : No. of GPU (<= 2)
!   nhq      : No. of thread for each GPU (No. of Hyper-Q)

!   nwm      : weighting model No.
!   nconvg   : flag of entropy convergnece
!   ndent    : No. of dent_lim
!   dent_lim : lower limit value of dent
!   rho      : Sperman's rho
!   tau      : Kendall's tau
!   entmin   : Min. value of entropy
!   entmax   : Max. value of entropy
!   nsm      : sampling model
!   ncut     : No. of sample cut
!   reent    : convergnece criterion for entropy

    !! other parameter
!   nunit    : device number

    !! module for parameter arrays
    use param_label
    use param_seed, only: seed_self
    !### add by Tanaka 2015.03.26 --->
    use input1_module
    use input2_module
    use input3_module
    use gpu_module
    !### add by Tanaka 2015.03.26 <---
    use param_dent                      !### add by Tanaka 2015.07.08
!    use param_dent,   only: ndent, dent_lim !### add by Tanaka 2015.07.08
    use param_copula, only: rho, tau       !### add by Tanaka 2015.04.15

    !! modules for subroutines
    use readc_module !### add by Tanaka 2015.04.01
    implicit none

    !! other parameters
    integer(kind = i4) :: i, nunit
    real   (kind = r8) :: dent_tmp !### add by Tanaka 2015.07.08
    character(len=128) :: line !text line buffer added by Tanaka 2015.04.16
```

```
   !! read input file
   nunit = 7
   open (nunit, file = 'inp_ILHS.dat', status = 'old') !### add by Tanaka
2015.01.25

   call readc (nunit)                !skip a line
   read (nunit, *) ndim, npop, itermax, ntrial, nobjf !dimension, sample size,
maximum iteration steps, no. of trials
   !### add by Tanaka 2013.07.18 --->
   !! Input for TOUGH2
!   call readc (nunit)
!   read (nunit, *) nw                !No. of wells
!   call readc (nunit)
!   read (nunit, *) nrot              !No. of inj./pro. rotation
!   call readc (nunit)
!   do j = 1, 2*nw
!     read (nunit, *) nxmin(j), nxmax(j)  !range of well grid No.
!   end do
!   call readc (nunit)
!   read (nunit, *) xmin, xmax          !range of inj./pro. rate
!   call readc (nunit)
!   read (nunit, *) co2a              !mass of injected CO2 without BHP limit
   !### add by Tanaka 2013.07.18 <---

   !### add by Tanaka 2014.04.08 --->
!   call readc (nunit)
!   read (nunit, *) ns                !No. of simulation run, range of normalized
entropy
   call readc (nunit)
   !### modify by Tanaka 2015.07.08 --->
   read (nunit, '(a)') line
   read (line, *) nwm, entmin, entmax, nconvg
   call alloc_dent2
   if ( nconvg == 1 ) then
     read (line, *) nwm, entmin, entmax, nconvg, dent_tmp
     dent_lim(1:ndim) = dent_tmp
   else if ( nconvg == 2 ) then
     read (nunit, *) (dent_lim(i), i = 1, ndim)
   end if
!   read (nunit, *) nwm, entmin, entmax   !weighting model
   !### modify by Tanaka 2015.07.08 <---

   call readc (nunit)
   !### modify by Tanaka 2015.04.15 --->
   read (nunit, '(a)') line
   read (line, *) nsm                !sampling model
   if ( nsm == 2 ) then
     read (line, *) nsm, rho, tau      !sampling model, Sperman's rho, Kendall's
tau
   end if
!   read (nunit, *) nsm                !sampling model
   !### modify by Tanaka 2015.04.15 <---

!   do j = 1, 16
!     read (nunit, '(i1)') ifnm(j)
!   end do
```

```
    !### add by Tanaka 2015.01.14 --->
    call readc (nunit)
    read (nunit, *) seed_self
    !### add by Tanaka 2015.01.14 <---
    !### add by Tanaka 2014.04.08 <---

    !### add by Tanaka 2015.01.25 --->
    call readc (nunit)
    read (nunit, *) ngpu, nhq, ndvc(1), ndvc(2)
    !### add by Tanaka 2015.01.25 <---

    print *, ngpu, nhq, ndvc(1), ndvc(2) !debug

    return

  end subroutine input

end module input_module
```

- subMKOUT.f90

```
module mkout_module
  implicit none
contains

  subroutine mkout

    !### write by Tanaka 2015.01.25
    !### modify by Tanaka 2015.07.17

    !! input parameters
!   ndim    : number of primary variables
!   npop    : number of sampling points
!   itermax : number of iteration steps
!   ntrial  : number of trials

    !! modules for parameter arrays
    use param_label
    use param_seed, only: seed_self
    !### add by Tanaka 2015.07.17 --->
    use input1_module
    use input2_module, only: nwm
    !### add by Tanaka 2015.07.17 <---
    use input3_module, only: ncut !### add by Tanaka 2015.04.13
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, nunit
    character(len=15)  :: item


    do i = 17, 20
      write (i, '(a36, 4(i4, a1))') &
       "(ndim, npop, itermax, ntrial) = (", ndim, ",", npop, ",", itermax, ",",
ntrial, ")"
      write (i, '(a14, i9)') "seed_self =", seed_self
      write (i, *)
    end do

    nunit = 17
    write (nunit, '(2a5, a15)', advance = 'no') "func", "ntri", "ymin"
    do i = 1, ndim
      write (item, '(a5, i0, a1)') "xopt(", i, ")"
      write (nunit, '(a15)', advance = 'no') trim(item)
    end do
    write (nunit, *)

!    nunit = 18
!    write (nunit, '(a7, 2(i3, a1))') "rot1(", ndim, ",", ndim, ")"

    nunit = 18
    write (nunit, '(a6, 3a15)') "func", "Avg.", "Std.", "CV"

    nunit = 19
    write (nunit, '(3a5, 2a15)', advance = 'no') &
   & "func", "ntri", "iter", "ymin", "ymin_in_nn"
    do i = 1, ndim
```

```
      write (item, '(a5, i0, a1)') "xopt(", i, ")"
      write (nunit, '(a15)', advance = 'no') trim(item)
    end do
    do i = 1, npop
      write (item, '(a4, i0, a1)') "wgt(", i, ")"
      write (nunit, '(a15)', advance = 'no') trim(item)
    end do
    if ( nwm == 1 ) then
      write (nunit, '(2a15)', advance = 'no') "NormEntropy", "Gamma"
    else if ( nwm == 2 ) then
      write (nunit, '(2a15)', advance = 'no') "NormEntropy", "Beta"
    end if
    write (nunit, *)

    nunit = 20
    write (nunit, '(3a5)', advance = 'no') "func", "ntri", "iter"
    !### add by Tanaka 2015.06.12 --->
    do j = 1, ndim
      write (item, '(a8, i0, a1)') "NormEnt(", j, ")"
      write (nunit, '(a15)', advance = 'no') trim(item)
    end do
!    do j = 1, ndim
!      write (item, '(a4, i0, a1)') "BEF(", j, ")"
!      write (nunit, '(a15)', advance = 'no') trim(item)
!    end do
!    do j = 1, ndim
!      write (item, '(a7, i0, a1)') "DifEnt(", j, ")"
!      write (nunit, '(a15)', advance = 'no') trim(item)
!    end do
    do j = 1, ndim
      write (item, '(a10, i0, a1)') "DifEntMin(", j, ")"
      write (nunit, '(a15)', advance = 'no') trim(item)
    end do
    !### add by Tanaka 2015.06.12 <---
    write (nunit, *)

    if ( ncut /= 0 ) then
      nunit = 21
      write (nunit, '(4a5)', advance = 'no') "func", "ntri", "iter", "ndim"
      do i = 1, npop
        write (item, '(a9, i0, a1)') "xbdy_old(", i, ")"
        write (nunit, '(a15)', advance = 'no') trim(item)
      end do
      do i = 1, npop
        write (item, '(a9, i0, a1)') "xbdy_new(", i, ")"
        write (nunit, '(a15)', advance = 'no') trim(item)
      end do
      write (nunit, *)
    end if

    return

  end subroutine mkout

end module mkout_module
```

- subMVNRND.f90

```
module mvnrnd_module
  implicit none
contains

  subroutine mvnrnd

    !! This subroutine generates random number vector from multivariate normal
distribution

    !### write by Tanaka 2015.02.23
    !### modify by Tanaka 2015.04.15

    !! input parameters
!   rho     : Sperman's rho
!   tau     : Kendall's tau
!   a       : input matrix

    !! output parameters
!   a       : output matrix

    !! modules for parameter arrays
    use param_label
    use param_copula, only: ncols, rho, tau, a !### add by Tanaka 2015.04.12

    !! modules for subroutines
    use cholesky_module              !### add by Tanaka 2015.04.11
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j              ! loop counter
    real   (kind = r8) :: pi
    integer(kind = i4), parameter :: nunit1 = 26, &
                                nunit2 =  9    !### add by Tanaka 2015.04.12

    pi  = atan(1.d0) * 4.d0
!    rho = 0.5d0
!    tau = 0.75d0

    write (nunit1, '(a12, 2(i3, a1))') "a(", ncols, ",", ncols, ")" !### add by
Tanaka 2015.04.15

    !! ask user for dimensions of matrix
    !### modify by Tanaka 2015.02.23 --->
!    print *,'Enter number of rows/cols in matrix (must be square) '
!    read *,ndim
    !### modify by Tanaka 2015.02.23 <---

    !! have user enter matrix
    !### modify by Tanaka 2015.02.23 --->
    if ( rho /= 0.d0 .or. tau /= 0.d0 ) then

      !! Set covariance matrix, Sigma
      do j = 1, ncols
        do i = 1, ncols
          if ( i == j ) then
            a(i, j) = 1.d0
```

```
        else
          a(i, j) = rho
!          a(i, j) = sin(pi / 2.d0 * tau)
!          a(i, j) = 2.d0 * sin(pi / 6.d0 * rho)
        end if
       end do
     end do

   else

     !! Set covariance matrix, Sigma from table data
     open (nunit2, file ='inp_SIGMA.dat', status = 'old')
     do i = 1, ncols
       read (nunit2, *) (a(i, j), j = 1, ncols)
!     do i = 1, ndim
!       read (nunit2, *) (a(i, j), j = 1, ndim)
     end do
     close (nunit2)

   end if

!    print *, 'Enter matrix in column-major order'
!    read *, a
   !### modify by Tanaka 2015.02.23 <---

   !! open output file
!    open (nunit1, file = 'out_CHLSKY.dat', status = 'replace') !### add by Tanaka
2015.02.23

   !! perform cholesky decomposition
   call cholesky !### modify by Tanaka 2015.04.15

   !! print matrix to user
!    print *,'Result::'
   do i = 1, ncols
!     print *, a(i, :)
     !### add by Tanaka 2015.02.23 --->
     do j = 1, ncols
       write (nunit1, '(f10.5)', advance = 'no') a(i, j)
     end do
     write (nunit1, *)
     !### add by Tanaka 2015.02.23 <---
   end do
   write (nunit1, *)
!    close (nunit1)

   !! terminate the program
   return
!    stop

  end subroutine mvnrnd

end module mvnrnd_module
```

- subOBJ.f90

```
module obj_module
  implicit none
contains

!*********************************************************
! read input file of objective function                  *
!*********************************************************
  subroutine input_obj

    !### write by Tanaka 2015.01.25
    !### modify by Tanaka 2015.04.01

    !! modules for parameter arrays
    use param_label
    use param_bmf
    use input1_module, only: ndim !### add by Tanaka 2015.03.25

    !! modules for subroutines
    use readc_module !### add by Tanaka 2015.04.01
    implicit none

    !! other parameters
    integer(kind = i4) :: i, nunit


    !! read input file
    nunit = 8
    open (nunit, file = 'inp_OBJ.dat', status = 'old')
    call readc (nunit) !skip a line
    read (nunit, *) (f1(i),  i = 1, 1000)
    read (nunit, *) (f2(i),  i = 1, 1000)
    read (nunit, *) (f3(i),  i = 1, 1000)
    read (nunit, *) (f4(i),  i = 1, 1000)
    read (nunit, *) (f5(i),  i = 1, 1000)
    read (nunit, *) (f6(i),  i = 1, 1000)
    read (nunit, *) (f7(i),  i = 1, 1000)
    read (nunit, *) (f8(i),  i = 1, 1000)
    read (nunit, *) (f9(i),  i = 1, 1000)
    read (nunit, *) (f10(i), i = 1, 1000)
    read (nunit, *) (f11(i), i = 1, 1000)
    read (nunit, *) (f15(i), i = 1, 1000)
    read (nunit, *) (f19(i), i = 1, 1000)
    read (nunit, *) (fbias(i), i = 1, 6)

    !### add by Tanaka 2015.03.17 --->
    allocate( rot1(ndim, ndim), rot2(ndim, ndim) )

    !! rotation matrix
    call rotmat (ndim) !modify by Tanaka 2015.04.01
!    call rotmat (ndim, rot1, rot2) !### modify by Tanaka 2015.01.25
    !### add by Tanaka 2015.03.17 <---

    return

  end subroutine input_obj
```

```
!*********************************************************
! main part of objective function                        *
!*********************************************************
  subroutine obj

    !### write by Goda 2011.
    !### modify by Tanaka 2015.07.24

    use param_label
    use param_bmf
    !### add by Tanaka 2015.03.29 --->
    use input1_module, only: ndim, npop, iobjf
    use param_obj
    !### add by Tanaka 2015.03.29 <---
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, np1
    !### modify by Tanaka 2015.07.07 --->
    real  (kind = r8) :: pi, xx, y1, y2, y3, z, z1, z2, xfunc(ndim+1)
!    real  (kind = r8) :: pi, xx, y1, y2, y3, z, z1, z2, xfunc(ndim)
    !### modify by Tanaka 2015.07.07 <---

    pi = 4.d0 * atan(1.d0)


    !! Function1: Sphere Function
    if ( iobjf == 1 ) then
      do i = 1, npop
        yopt(i) = 0.d0
        y1   = 0.d0
        do j = 1, ndim
          xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f1(j)
          y1      = y1 + xfunc(j)**2.d0
        end do
        yopt(i) = y1 + fbias(1)
      end do

    !! Function2: Schwefel's Problem 2.21
    else if ( iobjf == 2 ) then
      do i = 1, npop
        do j = 1, ndim
          xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f2(j)
        end do
        y1 = 0.d0
        do j = 1, ndim
          y2 = abs(xfunc(j))
          y1 = max(y1, y2)
        end do
        yopt(i) = y1 + fbias(2)
      end do

    !! Function3: Rosenbrock's Function
    else if ( iobjf == 3 ) then
      do i = 1, npop
        do j = 1, ndim
          xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f3(j)
```

```
    end do
    y1 = 0.d0
    do j = 1, ndim-1
      y1 = y1 + 100.d0*(xfunc(j)**2.d0 - xfunc(j+1))**2.d0 &
     &   + (xfunc(j) - 1.d0)**2.d0
    end do
    yopt(i) = y1 + fbias(3)
  end do

!! Function4, Rastringin's Funstion
else if ( iobjf == 4 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 10.d0 - f4(j)
    end do
    y1 = 0.d0
    do j = 1, ndim
      y1 = y1 + xfunc(j)**2.d0 - 10.d0*cos(2.d0*pi*xfunc(j)) + 10.d0
    end do
    yopt(i) = y1 + fbias(4)
  end do

!! Function5, Griewank's Function
else if ( iobjf == 5 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 1200.d0 - f5(j)
    end do
    y1 = 0.d0
    y2 = 1.d0
    do j = 1, ndim
      y1 = y1 + (xfunc(j)**2.d0) / 4000.d0
      y2 = y2 * cos(xfunc(j) / sqrt(dfloat(j)))
    end do
    yopt(i) = y1 - y2 + 1.d0 + fbias(5)
  end do

!! Function6, Ackley's Function
else if ( iobjf == 6 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 64.d0 - f6(j)
    end do
    y1 = 0.d0
    y2 = 0.d0
    do j = 1, ndim
      y1 = y1 + xfunc(j)**2.d0 / dble(ndim)
      y2 = y2 + cos(2.d0*pi*xfunc(j)) / dble(ndim)
    end do
    yopt(i) = 20.d0*(1.d0 - exp(-sqrt(y1)/5.d0)) &
   &       + exp(1.d0) - exp(y2) + fbias(6)
  end do

!! Function7: Schwefel's Problem 2.21
else if ( iobjf == 7 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 20.d0 - f7(j)
```

```
    end do
    y1 = 0.d0
    y2 = 1.d0
    do j = 1, ndim
      y1 = y1 + abs(xfunc(j))
      y2 = y2 * abs(xfunc(j))
    end do
    yopt(i) = y1 + y2
  end do

!! Function8: Schwefel's Problem 1.2
else if ( iobjf == 8 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 131.072d0 - f8(j)
    end do
    y1 = 0.d0
    y2 = 0.d0
    do j = 1, ndim
      y2 = y2 + xfunc(j)
      y1 = y1 + y2 * y2
    end do
    yopt(i) = y1
  end do

!! Function9: Extended f10
else if ( iobjf == 9 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f9(j)
    end do
    xfunc(ndim+1) = (xpop(i, 1) - 0.5d0) * 200.d0 - f9(1)
    y1 = 0.d0
    do j = 1, ndim
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    yopt(i) = y1
  end do

!! Function10: Bohachevsky
else if ( iobjf == 10 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 30.d0 - f10(j)
    end do
    y1 = 0.d0
    do j = 1, ndim-1
      z1 = xfunc(j)
      z2 = xfunc(j+1)
      y1 = y1 + z1**2.d0 + 2.d0*z2**2.d0
      y1 = y1 - 0.3d0 * cos(3.d0*pi*z1) - 0.4d0 * cos(4.d0*pi*z2) + 0.7d0
    end do
    yopt(i) = y1
  end do

!! Function11: Schaffer
else if ( iobjf == 11 ) then
```

```
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f11(j)
    end do
    y1 = 0.d0
    do j = 1, ndim - 1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    yopt(i) = y1
  end do

!! Hybrid
!! Function12: Non-shifted Function9 + Function1
else if ( iobjf == 12 ) then
  do i = 1, npop
    np1 = int(ndim/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0
    end do
    do j = np1+1, ndim
      xfunc(j) = xfunc(j) - f1(j-np1)
    end do
    y1 = 0.d0
    do j = 1, np1-1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    z  = xfunc(np1)**2.d0 + xfunc(1)**2.d0
    y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    do j = np1+1, ndim
      y1 = y1 + xfunc(j)**2.d0
    end do
    yopt(i) = y1
  end do

!! Function13: Non-shifted Function9 + Function3
else if ( iobjf == 13 ) then
  do i = 1, npop
    np1 = int(ndim/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0
    end do
    do j = np1 + 1, ndim
      xfunc(j) = xfunc(j) - f3(j-np1)
    end do
    y1 = 0.d0
    do j = 1, np1 - 1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    z  = xfunc(np1)**2.d0 + xfunc(1)**2.d0
    y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    do j = np1 + 1, ndim - 1
      y1 = y1 + 100.d0*(xfunc(j)**2.d0 - xfunc(j+1))**2.d0 &
         + (xfunc(j) - 1.d0)**2.d0
    end do
    yopt(i) = y1
```

```
  end do

!! Function14: Non-shifted Function9 + Function4
else if ( iobjf == 14 ) then
  do i = 1, npop
    np1 = int(ndim/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 10.d0
    end do
    do j = np1 + 1, ndim
      xfunc(j) = xfunc(j) - f4(j-np1)
    end do
    y1 = 0.d0
    do j = 1, np1 - 1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    z  = xfunc(np1)**2.d0 + xfunc(1)**2.d0
    y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    do j = np1 + 1, ndim - 1
      y1 = y1 + xfunc(j)**2.d0 - 10.d0 * cos(2.d0*pi*xfunc(j)) + 10.d0
    end do
    yopt(i) = y1
  end do

!! Function15: Non-shifted Function10 + Non-shifted Function7
else if ( iobjf == 15 ) then
  do i = 1, npop
    np1 = int(ndim/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 20.d0 - f15(j)
    end do
    y1 = 0.d0
    do j = 1, np1 - 1
      z1 = xfunc(j)
      z2 = xfunc(j+1)
      y1 = y1 + z1**2.d0 + 2.d0 * z2**2.d0
      y1 = y1 - 0.3d0 * cos(3.d0*pi*z1) - 0.4d0 * cos(4.d0*pi*z2) + 0.7d0
    end do
    y2 = 0.d0
    y3 = 1.d0
    do j = np1 + 1, ndim
      y2 = y2 + abs(xfunc(j))
      y3 = y3 * abs(xfunc(j))
    end do
    yopt(i) = y1 + y2 + y3
  end do

!! Function16: Non-shifted Function9 + Function1
else if ( iobjf == 16 ) then
  do i = 1, npop
    np1 = int(ndim/2)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0
    end do
    do j = np1 + 1, ndim
      xfunc(j) = xfunc(j) - f1(j-np1)
    end do
```

```
    y1 = 0.d0
    do j = 1, np1 - 1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    z  = xfunc(np1)**2.d0 + xfunc(1)**2.d0
    y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    do j = np1 + 1, ndim
      y1 = y1 + xfunc(j)**2.d0
    end do
    yopt(i) = y1
  end do

!! Function17: Non-shifted Function9 + Function3
else if ( iobjf == 17 ) then
  do i = 1, npop
    np1 = int(ndim*3/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0
    end do
    do j = np1 + 1, ndim
      xfunc(j) = xfunc(j) - f3(j-np1)
    end do
    y1 = 0.d0
    do j = 1, np1 - 1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    z  = xfunc(np1)**2.d0 + xfunc(1)**2.d0
    y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    do j = np1 + 1, ndim - 1
      y1 = y1 + 100.d0 * (xfunc(j)**2.d0 - xfunc(j+1))**2.d0 &
     &   + (xfunc(j) - 1.d0)**2.d0
    end do
    yopt(i) = y1
  end do

!! Function18: Non-shifted Function9 + Function4
else if ( iobjf == 18 ) then
  do i = 1, npop
    np1 = int(ndim*3/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 10.d0
    end do
    do j = np1 + 1, ndim
      xfunc(j) = xfunc(j) - f4(j-np1)
    end do
    y1 = 0.d0
    do j = 1, np1 - 1
      z  = xfunc(j)**2.d0 + xfunc(j+1)**2.d0
      y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    end do
    z  = xfunc(np1)**2.d0 + xfunc(1)**2.d0
    y1 = y1 + (z**0.25d0) * (sin(50.d0*(z**0.1d0))**2.d0 + 1.d0)
    do j = np1 + 1, ndim - 1
      y1 = y1 + xfunc(j)**2.d0 - 10.d0*cos(2.d0*pi*xfunc(j)) + 10.d0
    end do
    yopt(i) = y1
```

```
  end do

!! Function19: Non-shifted Function10 + Non-shifted Function7
else if ( iobjf == 19 ) then
  do i = 1, npop
    np1 = int(ndim*3/4)
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 20.d0 - f19(j)
    end do
    y1 = 0.d0
    do j = 1, np1 - 1
      z1 = xfunc(j)
      z2 = xfunc(j+1)
      y1 = y1 + z1**2.d0 + 2.d0 * z2**2.d0
      y1 = y1 - 0.3d0 * cos(3.d0*pi*z1) - 0.4d0 * cos(4.d0*pi*z2) + 0.7d0
    end do
    y2 = 0.d0
    y3 = 1.d0
    do j = np1 + 1, ndim
      y2 = y2 + abs(xfunc(j))
      y3 = y3 * abs(xfunc(j))
    end do
    yopt(i) = y1 + y2 + y3
  end do

!! Function20: Rotated Function4
else if ( iobjf == 20 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 64.d0 - f6(j)
    end do
    call matvec(xfunc, ndim, ndim) !### modify by Tanaka 2015.03.17

    y1 = 0.d0
    y2 = 0.d0
    do j = 1, ndim
      y1 = y1 + xfunc(j)**2.d0 / dble(ndim)
      y2 = y2 + cos(2.d0 * pi * xfunc(j)) / dble(ndim)
    end do
    yopt(i) = 20.d0*(1.d0 - exp(-sqrt(y1) / 5.d0)) + exp(1.d0) &
   &       - exp(y2) + fbias(6)
  end do

!! Function21: Rotated Function5
else if ( iobjf == 21 ) then
  do i = 1, npop
    do j = 1, ndim
      xfunc(j) = (xpop(i, j) - 0.5d0) * 10.d0 - f4(j)
    end do
    call matvec(xfunc, ndim, ndim) !### modify by Tanaka 2015.03.17
    y1 = 0.d0
    do j = 1, ndim
      y1 = y1 + xfunc(j)**2.d0 - 10.d0 * cos(2.d0*pi*xfunc(j)) + 10.d0
    end do
    yopt(i) = y1 + fbias(4)
  end do

!! Function22: Rotated Function6
```

```
    else if ( iobjf == 22 ) then
      do i = 1, npop
        do j = 1, ndim
          xfunc(j) = (xpop(i, j) - 0.5d0) * 1200.d0 - f5(j)
        end do
        call matvec(xfunc, ndim, ndim) !### modify by Tanaka 2015.03.17
        y1 = 0.d0
        y2 = 1.d0
        do j = 1, ndim
          y1 = y1 + (xfunc(j)**2.d0) / 4000.d0
          y2 = y2 * cos(xfunc(j) / sqrt(dble(j)))
        end do
        yopt(i) = y1 - y2 + 1.d0 + fbias(5)
      end do

    !! Function23: Ill-scaled Sphere Function
    else if ( iobjf == 23 ) then
      do i = 1, npop
        yopt(i) = 0.d0
        y1   = 0.d0
        do j = 1, ndim
          xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f1(j)
          y1      = y1 + xfunc(j)**2.d0 / dble(j)
        end do
        yopt(i) = y1 + fbias(1)
      end do

    !! Function24: Ill-scaled Rosenbrock Function
    else if ( iobjf == 24 ) then
      do i = 1, npop
        do j = 1, ndim
          xfunc(j) = (xpop(i, j) - 0.5d0) * 200.d0 - f3(j)
        end do
        y1 = 0.d0
        do j = 2, ndim
          xx = xfunc(j) * dble(j)
          y1 = y1 + 100.d0 * (xx**2.d0 - xfunc(1))**2.d0 + (xx - 1.d0)**2.d0
        end do
        yopt(i) = y1 + fbias(3)
      end do

    end if

    return

  end subroutine obj


!*********************************************************
! rotation matrix                                        *
!*********************************************************
  subroutine rotmat (ndim)

    !### write by Tanaka 2015.01.25
    !### modify by Tanaka 2015.07.24

    use param_label
    use param_bmf !### add by Tanaka 2015.03.17
```

```
    implicit none

    !! input & output parameters
    integer(kind = i4) :: ndim

    !! other parameters
    integer(kind = i4) :: j, j1, j2, nunit
    real(kind = r8) :: pi, rdum, alpha


    pi = 4.d0 * atan(1.d0)

    nunit = 22
!   open (nunit, file = 'out_ROT1.dat', status = 'replace')       !### add by
Tanaka 2015.03.17
    write (nunit, '(a7, 2(i3, a1))') "rot1(", ndim, ",", ndim, ")" !### add by
Tanaka 2015.03.30

    do j1 = 1, ndim
      do j2 = 1, ndim
        rot1(j1, j2) = 0.d0
      end do
      rot1(j1, j1) = 1.d0
    end do

    do j = 2, ndim
      do j1 = 1, ndim
        do j2 = 1, ndim
          rot2(j1, j2) = 0.d0
        end do
        rot2(j1, j1) = 1.d0
      end do
      call random_number(rdum)
      alpha     = (rdum - 0.5d0) * pi * 0.5d0
      rot2(1, 1) =  cos(alpha)
      rot2(j, j) =  cos(alpha)
      rot2(1, j) =  sin(alpha)
      rot2(j, 1) = - sin(alpha)
      call matmat (ndim, ndim, ndim)
    end do

    do j = 2, ndim - 1
      do j1 = 1, ndim
        do j2 = 1, ndim
          rot2(j1, j2) = 0.d0
        end do
        rot2(j1, j1) = 1.d0
      end do
      call random_number(rdum)
      alpha          = (rdum - 0.5d0) * pi * 0.5d0
      rot2(j   , j   ) =  cos(alpha)
      rot2(ndim, ndim) =  cos(alpha)
      rot2(j   , ndim) =  sin(alpha)
      rot2(ndim, j   ) = - sin(alpha)
      call matmat (ndim, ndim, ndim)
    end do

    do j2 = 1, ndim
```

```
      write (nunit, '(100e15.7)') (rot1(j1, j2), j1=1, ndim)
    end do

    return

  end subroutine rotmat


!************************************************************
! matrix c = matrix a * matrix b                            *
!************************************************************
  subroutine matmat (nxa, npop, nyb)

    !### write by Goda 2011.
    !### modify by Tanaka 2015.04.01

    use param_label
    use param_bmf   !### add by Tanaka 2015.03.17
    implicit none

    !! input & output parameters
    integer(kind = i4) :: npop, nxa, nyb

    !! other parameters
    integer(kind = i4) :: i, j, k
    real   (kind = r8) :: cc, c(nxa, nyb)


    do i = 1, nxa
      do j = 1, nyb
        cc = 0.d0
        do k = 1, npop
          cc = cc + rot1(i, k) * rot2(k, j)
        end do
        c(i, j) = cc
      end do
    end do

    do i = 1, nxa
      do j = 1, nyb
        rot1(i, j) = c(i, j)
      end do
    end do

    return

  end subroutine matmat


!************************************************************
! vector c = matrix a * vector b                            *
!************************************************************
  subroutine matvec (b, nx, ny)

    !### write by Goda 2011.
    !### modify by Tanaka 2015.04.01

    use param_label
```

```
    use param_bmf   !### add by Tanaka 2015.03.17
    implicit none

    !! input & output parameters
    integer(kind = i4) :: nx, ny
    real   (kind = r8) :: b(ny)

    !! other parameters
    integer(kind = i4) :: i, k
    real   (kind = r8) :: cc, c(nx)


    do i = 1, nx
      cc = 0.d0
      do k = 1, ny
        cc = cc + rot1(i, k) * b(k)
      end do
      c(i) = cc
    end do
    do i = 1, nx
      b(i) = c(i)
    end do

    return

  end subroutine matvec

end module obj_module
```

- subREADC.f90

```
! ******************************** T.O.C. - TECHNICAL RESEARCH CENTER **
! ***********************************************************************
! **                                                                   **
! **                      SKIP COMMENT CARDS                             **
! **                                                                   **
! ***********************************************************************
! ** FRCWEL-SUB *********************************************************
module readc_module
  implicit none
contains

  subroutine readc (nunit)

    !### modify by Tanaka 2015.04.01

    implicit none

    integer         :: i, nunit
    character(len=1) :: ic

    do i = 1, 50
      read (nunit, '(a)') ic
!     read (5, '(a)') ic
!     read (10, '(a)') ic
      if ( ic /= '/' ) then
        backspace nunit
!       backspace 5
!       backspace 10
        return
      end if
    end do

    write (6, *) 'comment cards should be less than 50 in a series'

    return

  end subroutine readc

end module readc_module
```

- subSECND.f90

```
module second_module
  implicit none
!  implicit double precision(A-H,O-Z)
contains

  subroutine second (time)

    ! timing function for IBM RS/6000

    !### write by Tanaka 2014.
    !### modify by Tanaka 2014.07.19

!    implicit real*8(a-h,o-z)
    implicit none

    !! output parameter
    double precision :: time
!    real           :: time

    !! other parameters
!    integer :: icall, mclock

!    save icall
!    data icall/0/
!    icall = icall + 1
!###    if (icall .eq. 1) write (11, 899)
!###899 FORMAT(6X,'SECOND  1.0      6 September 1994',6X,&
!#    & 'CPU timing function for IBM RS/6000')
    !### modify by Tanaka 2015.07.19 --->
    call cpu_time(time)
!    time = dble(mclock()) / 100.d0
!    time = real(mclock()) / 100.0
    !### modify by Tanaka 2015.07.19 <---

    return

  end subroutine second

end module second_module
```

- subSORT.f90

```
module sort_module
  implicit none
contains

  subroutine sort

    !! program by houlsby, g.t. and sloan, s.w. in adv. eng. software, 1984, vol.6,
no.4
    !! modify by Tanaka 2015.04.08

    !### add by Tanaka 2012.10.20 --->
    !! subroutine sort
    !! order integers stored in 'irank' in ascending sequence of their key
    !! values stored in 'yopt'

    !! input parameters:
!   npop  : positive integer giving length of list
!   irank : a list of length npop of integers
!   yopt  : a list of length of couble precision keys

    !! output parameters:
!   npop  : unchanged
!   irank : a list of length npop of integers sorted in ascending
!         sequence of their double precision keys
!   yopt  : a list of length of couble precision keys

    !! notes:
!   uses quicksort algorithm, effecient for npop values greater than
!   about 12 (although may be system dependent)

!   routine sorts lists up to length 2**maxstk

    !! other parameters:
!   maxstk: parameter giving maximum stack size
!   stktop: current size of stacks
!   lstack: stack of pointers to left ends of xub-lists
!   rstack: stack of pointers to right ends of xub-lists
!   ltemp:  temporary used in list swapping
    !### add by Tanaka 2012.10.20 <---

    !! modules for parameter arrays
    use param_label
    !### add by Tanaka 2015.03.26 --->
    use input1_module, only: npop
    use param_obj,    only: yopt
    use param_rank
    !### add by Tanaka 2015.03.26 <---
!    implicit real*8(a-h,o-z)
    implicit none

    !! other parameters
    integer(kind = i4), parameter :: maxstk = 32
!    parameter(maxstk=32)
    integer(kind = i4) :: ll, lr, lm, nl, nr, ltemp, stktop
    integer(kind = i4) :: lstack(maxstk), rstack(maxstk)
    real   (kind = r8) :: guess !### add by Tanaka 2012.10.20
```

```
   ll = 1
   lr = npop
   stktop = 0
10 continue
   if (ll .lt. lr) then
     nl    = ll
     nr    = lr
     lm    = (ll + lr) / 2
     guess = yopt(irank(lm))

     !! find keys for exchange
  20 continue
     if (yopt(irank(nl)) .lt. guess) then
       nl = nl + 1
       goto 20
     end if
  30 continue
     if (guess .lt. yopt(irank(nr))) then
       nr = nr - 1
       goto 30
     end if

     if (nl .lt. (nr-1)) then
       ltemp = irank(nl)
       irank(nl) = irank(nr)
       irank(nr) = ltemp
       nl = nl + 1
       nr = nr - 1
       goto 20
     end if

     !! deal with crossing of pointers
     if (nl .le. nr) then
       if (nl .lt. nr) then
         ltemp    = irank(nl)
         irank(nl) = irank(nr)
         irank(nr) = ltemp
       end if
       nl = nl + 1
       nr = nr - 1
     end if

     !! select sub-list to be proceed next
     stktop = stktop + 1
     if (nr .lt. lm) then
       lstack(stktop) = nl
       rstack(stktop) = lr
       lr = nr
     else
       lstack(stktop) = ll
       rstack(stktop) = nr
       ll = nl
     end if
     goto 10
   end if
```

```
    !! process any stacked sub-lists
    if (stktop .ne. 0) then
      ll    = lstack(stktop)
      lr    = rstack(stktop)
      stktop = stktop - 1
      goto 10
    end if

    return

  end subroutine sort

end module sort_module
```

- subWEIGHT.f90

```
module weight_module
  implicit none
contains

  subroutine weight

    !### write by Tanaka 2013.
    !### modify by Tanaka 2015.07.24

    !! module for parameter arrays
    use param_label
    !### add by Tanaka 2015.03.29 --->
    use input1_module, only: npop, iter
    use input2_module, only: nwm, entmin, entmax
    use param_obj,    only: yopt
    use param_ymin
    use param_rank
    use param_ent,    only: gam1, beta, ent_yopt
    use param_wgt,    only: sum_wgt1, wgt
    !### add by Tanaka 2015.03.29 <---
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, ii
    real   (kind = r8) :: sum_wgt2, ynorm, xn, enelv(npop), ymin3, ymax3


    if ( nwm == 1 ) then
      !! Zipf's model
      !! set parameter gam1 and weighting coefficient
      ii = npop
      sum_wgt1        = 0.d0
      wgt(irank(npop)) = 1.d0 / dble(npop)**gam1
      sum_wgt1        = sum_wgt1 + wgt(irank(npop))
      do i = npop-1, 1, -1
        if ( yopt(irank(i)) == yopt(irank(i+1)) ) then
          wgt(irank(i)) = 1.d0 / dble(ii)**gam1
          sum_wgt1     = sum_wgt1 + wgt(irank(i))
        else
          wgt(irank(i)) = 1.d0 / dble(i)**gam1
          sum_wgt1     = sum_wgt1 + wgt(irank(i))
          ii = i
        end if
      end do

    else if ( nwm == 2 ) then
      !! Canonical Distribution
      !! set parameter beta and weighting coefficient
      ynorm = 1.d0
      xn    = dble(npop)
      xn    = 1.d0 / log(xn)
      ymin3 = ymin
!     ymin3 = yopt(irank(1))
      ymax3 = yopt(irank(npop))
      if ( iter == 1 ) then
        !### modify by Tanaka 2015.01.21 --->
```

```
!       do j = 10000, 1, -1
!         if ( ent_yopt .lt. entmax ) then
        do j = 1, 10000
          if ( ent_yopt < entmin .or. ent_yopt > entmax ) then
        !### modify by Tanaka 2015.01.21 <---
           sum_wgt1 = 0.d0
           sum_wgt2 = 0.d0
           beta      = dble(j) / 100.d0
           enelv(irank(npop)) = ymax3 - ymin3
           wgt(irank(npop))   = exp(-beta*enelv(irank(npop)))
!           wgt(irank(npop))   = exp(beta*enelv(irank(npop)))
           sum_wgt1          = sum_wgt1 + wgt(irank(npop))
           do i = npop-1, 1, -1
             if ( yopt(irank(i)) == yopt(irank(i+1)) ) then
               enelv(irank(i)) = (yopt(irank(i+1)) - ymin3) * ynorm
             else
               enelv(irank(i)) = (yopt(irank(i)) - ymin3) * ynorm
             end if
             wgt(irank(i)) = exp(-beta*enelv(irank(i)))
!             wgt(irank(i)) = exp(beta*enelv(irank(i)))
             sum_wgt1     = sum_wgt1 + wgt(irank(i))
             sum_wgt2     = sum_wgt2 + beta * enelv(irank(i)) * wgt(irank(i))
!             sum_wgt2     = sum_wgt2 - beta * enelv(irank(i)) * wgt(irank(i))
           end do
           ent_yopt = (log(sum_wgt1) + sum_wgt2 / sum_wgt1) * xn
          end if
        end do
      else
!        beta = beta - 0.05d0 !linear cooling
        beta = beta / 0.95d0 !exponential cooling
!        beta = beta * log(1.d0 + dble(iter)) / log(dble(iter)) !logarithmic cooling
        sum_wgt1 = 0.d0
        sum_wgt2 = 0.d0
        enelv(irank(npop)) = ymax3 - ymin3
        wgt(irank(npop))   = exp(-beta*enelv(irank(npop)))
!        wgt(irank(npop))   = exp(beta*enelv(irank(npop)))
        sum_wgt1          = sum_wgt1 + wgt(irank(npop))
        do i = npop-1, 1, -1
          if ( yopt(irank(i)) == yopt(irank(i+1)) ) then
            enelv(irank(i)) = (yopt(irank(i+1)) - ymin3) * ynorm
          else
            enelv(irank(i)) = (yopt(irank(i)) - ymin3) * ynorm
          end if
          wgt(irank(i)) = exp(-beta * enelv(irank(i)))
!          wgt(irank(i)) = exp(beta * enelv(irank(i)))
          sum_wgt1     = sum_wgt1 + wgt(irank(i))
          sum_wgt2     = sum_wgt2 + beta * enelv(irank(i)) * wgt(irank(i))
!          sum_wgt2     = sum_wgt2 - beta * enelv(irank(i)) * wgt(irank(i))
        end do
        ent_yopt = (log(sum_wgt1) + sum_wgt2 / sum_wgt1) * xn
      end if
    end if

  end subroutine weight

    end module weight_module
```

6.2. A.2.計算コード（ヒストリーマッチング）

- はじめに

本節では貯留層シミュレーションを伴うヒストリーマッチングを行うプログラム ILHS_HM について解説する．貯留層シミュレーションは TOUGH2/ECO2N によって行っている．前節の ILHS_BMF でのベンチマーク関数の部分を，TOUGH2/ECO2N のシミュレーションに置き換えたものと考えて良い．

- プログラムと実行環境

ILHS_HM は Fortran90 に基づいた複数の計算コードとコンパイル用ファイル Makefile，複数の入力ファイルによって構成される．開発・実行環境は Linux 64-bit であり，PGI コンパイラー環境についてのみ動作することを確認している．プログラムは MPI を利用しており，mpich のインストールが必要である．以下の実行方法は mpich3 に対応している．

- コンパイル及び実行方法

ディレクトリ code と input，Makefile を同一のディレクトリに配置し，同ディレクトリに空のディレクトリ module と object を作成する．Makefile を編集し，コンパイラを指定する変数 FC を実行環境に合わせて変更する．次に，TOUGH2/ECO2N の実行ファイル t2c_v*.*と t2g_v*.*，及びコピーした TOUGH2/ECO2N によって事前に生成された MESHA と MESHB をディレクトリ input 内にコピーする．以下のコマンドを打ち込み，実行する

```
$ make
$ ./ilhsbmf_v*.*
$ cp ilhshm_v* input/
$ cd input
$ mpiexec -n 1 ./ilhshm_v*.* inp_ilhs_16.dat >log
```

- 計算コードの構成

以下に，計算コードの構成を示す．main は main program，それ以外は subroutine を示す．

```
[Program or Subroutine Name]   [File Name]
main                          (mainOPT.f90)
  +--- second                  (subSECOND.f90)
  +--- input                   (subINPUT.f90)
  |     +--- readc             (subREADC.f90)
  +--- mkout                   (subMKOUT.f90)
  +--- ilhs                    (subILHS.f90)
  |     +--- gamma             (subGAMMA.f90)
  |     +--- mvnrnd            (subMVNRND.f90)
  |     |     +--- cholesky    (subCHLSKY.f90)
  |     +--- gcopula           (subGCOPULA.f90)
  |     +--- lhs_ini           (subLHS.f90)
  |           +--- lhs_ord     (subLHS.f90)
  |                 +--- mvnrnd  (subMVNRND.f90)
  |                 +--- gcopula (subGCOPULA.f90)
  |           +--- lhs_set     (subLHS.f90)
  |     +--- loopdiv           (subLOOPDIV.f90)
  |     |
  |     +--- init              (subGAMMA.f90)
  |     |     +--- walsh       (fctnWALSH.f90)
  |     +--- obj               (subOBJ.f90)
  |     +--- mvout             (subMVOUT.f90)
  |     |
  |     +--- sort              (subSORT.f90)
  |     +--- weight            (subWEIGHT.f90)
  |     +--- cdf               (subCDF.f90)
  |     +--- ent               (subENT.f90)
  +--- second                  (subSECOND.f90)
```

全てのサブルーチンはモジュール化されている．各サブルーチンで共有される変数，定数は上記とは別のモジュールファイル，

mdlILHS.f90

mdlINIT.f90

mdlOBJ.f90

で定義される.

- ILHS のプログラム

・main (mainOPT.f90)

メインファイル．各サブルーチンを呼び出す．出力ファイルの作成，時間の計測を行う．

出力ファイル：out_YMIN.dat, out_YMINavg.dat, out_YMINall.dat,
out_ENTROPY.dat, out_NRNDN.dat, out_GCOPULA.dat, out_CHLSKY.dat

・input (subINPUT.f90)

ILHS の入力ファイルを読み込む．

入力ファイル：inp_ILHS.dat

・mkout (subMKOUT.f90)

ILHS の出力ファイル中のデータ項目（ヘッダー）を書き出す．

・ilhs (subILHS.f90)

ILHS のメインファイル．ILHS によるサンプル点を設定する．

・gamma (subGAMMA.f90)

重み付けパラメータのγを設定する．

・subsort.f90

目的関数の値に応じて，各サンプル区間のランク付けを行う．

・subweight.f90

各サンプル区間のランクに応じて，各サンプル区間の重み付けを行う．

・subcdf.f90

計算結果と重み付けの結果から，次のステップでの累積密度関数を計算する．

・subent.f90

正規化エントロピーを計算する．

・mvnrnd (subMVNRND.f90)

多変量正規分布によるランダムナンバーを発生させる．

・cholesky (subCHLSKY.f90)

コレスキー分解を行う．

・gcopula (subGCOPULA.f90)

正規コピュラ分布によるランダムナンバーを発生させる．

目的関数の計算に関わるプログラム

・obj (subOBJ.f90)

目的関数の計算を行う．

・init (subINIT.f90)

TOUGH2/ECO2N 用の input ファイルを作成する.

・walsh (fctnWALSH.f90)

Walsh 関数の計算を行う.

その他のプログラム

・loopdiv (subLOOPDIV.f90)

MPI 用のプロセス数を計算する.

・subreadc.f90

input ファイルを読む際にコメント行を読み飛ばす.

- 入力ファイル

inp_ILHS.dat

ILHS による最適化計算に使用する入力ファイル

・ndim：最適化問題の次元

・npop：サンプル点の個数

・itermax：最適化計算の最大回数

・ntrial：独立試行の回数

・nobjf：目的関数の個数（=24）

・idistr：主変数の設定用パラメータ

1：浸透率

2：浸透率と孔隙率

・nwm：重み付けモデルの番号

1：Zipf の法則，2：Canonical 分布

・entmin：重み付けパラメータを設定する際のエントロピーの最小値（=0.00）

・entmax：重み付けパラメータを設定する際のエントロピーの最大値（=0.95）

・nconvg：正規化エントロピーによる収束判定方法に関する指標

0：収束判定なし

1：収束判定あり，全ての主変数で同一の値に収束

2：収束判定あり，主変数によって異なる値に収束

・nsm：サンプル点の発生方法

1：コンパイラ依存の乱数

2：正規コピュラ分布による乱数

・seed_self：乱数のシード

・ngpu：使用する GPU の個数

・nhq：1 つの GPU で実行する

・ndvc：使用する GPU のワークステーション側でのデバイスナンバー（未使用）

- 出力ファイル

out_SORT.dat

各独立試行の終了時における目的関数の値（ymin），及び主変数 *i* の最適解の値（yopt(i)）を出力する．

out_XPOP.dat

各サンプル点に対する目的関数の値，最適解の値を出力する．

out_ENTROPY.dat

各独立試行における，主変数 *i* の正規化エントロピーの値（NormEnt(i)）と正規化エントロピーと 2 値エントロピー関数との差（DifEntMin(i)）を出力する

以下の出力ファイルは，idistr=2の場合のみ書き出す．

out_NRNDN.dat

多変量正規分布による乱数を出力する．

out_GCOPULA.dat

正規コピュラ分布による乱数を出力する．

out_CHLSKY.dat

コレスキー分解の結果を出力する．

- 計算コード
- Makefile

```
### VERSION
VER=3.5

### COMPILER
FC=mpif90

### OPTION
#OPT=-fastsse -module module -r8 -Mmpi=mpich -ta=nvidia -Mcuda=nvidia,cuda7.0
OPT=-fastsse -module module -r8 -Mmpi=mpich
#OPT=-fastsse -module module -r8
#OPT=-fastsse -module module -r8 -mcmodel=medium
## debug (pgdbg)
#OPT=-module module -r8 -Mmpi=mpich -g

### LIBRARY
LIB=

code= ¥
  object/mainOPT.o ¥
  object/subCDF.o ¥
  object/subENT.o ¥
  object/subGAMMA.o ¥
  object/subILHS.o ¥
  object/subINIT.o ¥
  object/subINPUT.o ¥
  object/subLHS.o ¥
  object/subLOOPDIV.o ¥
  object/subMKOUT.o ¥
  object/subMVOUT.o ¥
  object/subOBJ.o ¥
  object/subREADC.o ¥
  object/subSECOND.o ¥
  object/subSORT.o ¥
  object/subWEIGHT.o ¥
  object/fctnWALSH.o ¥
  object/subCHLSKY.o ¥
  object/subGCOPULA.o ¥
  object/subMVNRND.o

OBJ= ¥
  object/mdlILHS.o ¥
  object/mdlINIT.o ¥
  object/mdlOBJ.o ¥
  object/mdlPROC.o ¥
  $(code)

ilhs_bmf : $(OBJ)
	$(FC) $(OPT) -o ilhshm_v$(VER) $(OBJ) $(LIB)

### code ###
object/mainOPT.o : code/mainOPT.f90 ¥
                object/subILHS.o ¥
                object/subINPUT.o ¥
                object/subMKOUT.o ¥
```

```
                object/subSECOND.o
        $(FC) $(OPT) -c $<
        mv mainOPT.o object/

object/subCDF.o : code/subCDF.f90
        $(FC) $(OPT) -c $<
        mv subCDF.o object/

object/subCHLSKY.o : code/subCHLSKY.f90
        $(FC) $(OPT) -c $<
        mv subCHLSKY.o object/

object/subENT.o : code/subENT.f90
        $(FC) $(OPT) -c $<
        mv subENT.o object/

object/subGAMMA.o : code/subGAMMA.f90
        $(FC) $(OPT) -c $<
        mv subGAMMA.o object/

object/subGCOPULA.o : code/subGCOPULA.f90
        $(FC) $(OPT) -c $<
        mv subGCOPULA.o object/

object/subILHS.o : code/subILHS.f90 ¥
                object/subCDF.o ¥
                object/subENT.o ¥
                object/subGAMMA.o ¥
                object/subGCOPULA.o ¥
                object/subLHS.o ¥
                object/subMVNRND.o ¥
                object/subINIT.o ¥
                object/subLOOPDIV.o ¥
                object/subMVOUT.o ¥
                object/subOBJ.o ¥
                object/subSORT.o ¥
                object/subWEIGHT.o
        $(FC) $(OPT) -c $<
        mv subILHS.o object/

object/subINIT.o : code/subINIT.f90 ¥
                object/subREADC.o ¥
                object/fctnWALSH.o
        $(FC) $(OPT) -c $<
        mv subINIT.o object/

object/subINPUT.o : code/subINPUT.f90 ¥
                 object/subREADC.o
        $(FC) $(OPT) -c $<
        mv subINPUT.o object/

object/subLHS.o : code/subLHS.f90 ¥
               object/subGCOPULA.o ¥
               object/subMVNRND.o
        $(FC) $(OPT) -c $<
        mv subLHS.o object/

object/subLOOPDIV.o : code/subLOOPDIV.f90
```

```
	$(FC) $(OPT) -c $<
	mv subLOOPDIV.o object/

object/subMKOUT.o : code/subMKOUT.f90
	$(FC) $(OPT) -c $<
	mv subMKOUT.o object/

object/subMVNRND.o : code/subMVNRND.f90 ¥
		     object/subCHLSKY.o
	$(FC) $(OPT) -c $<
	mv subMVNRND.o object/

object/subMVOUT.o : code/subMVOUT.f90
	$(FC) $(OPT) -c $<
	mv subMVOUT.o object/

object/subOBJ.o : code/subOBJ.f90 ¥
		  object/subSECOND.o ¥
		  object/subREADC.o
	$(FC) $(OPT) -c $<
	mv subOBJ.o object/

object/subREADC.o : code/subREADC.f90
	$(FC) $(OPT) -c $<
	mv subREADC.o object/

object/subSECOND.o : code/subSECOND.f90
	$(FC) $(OPT) -c $<
	mv subSECOND.o object/

object/subSORT.o : code/subSORT.f90
	$(FC) $(OPT) -c $<
	mv subSORT.o object/

object/subWEIGHT.o : code/subWEIGHT.f90
	$(FC) $(OPT) -c $<
	mv subWEIGHT.o object/

object/fctnWALSH.o : code/fctnWALSH.f90
	$(FC) $(OPT) -c $<
	mv fctnWALSH.o object/

### module ###
object/mdlILHS.o : code/mdlILHS.f90
	$(FC) $(OPT) -c $<
	mv mdlILHS.o object/

object/mdlINIT.o : code/mdlINIT.f90
	$(FC) $(OPT) -c $<
	mv mdlINIT.o object/

object/mdlOBJ.o : code/mdlOBJ.f90
	$(FC) $(OPT) -c $<
	mv mdlOBJ.o object/

object/mdlPROC.o : code/mdlPROC.f90
	$(FC) $(OPT) -c $<
	mv mdlPROC.o object/
```

```
clean :
	rm -f module/*.mod object/*.o ilhshm_v$(VER)
```

- mainOPT.f90

```
Program main
!=========================================================
! Iterative Latin Hypercube Samplings for history matching of Iwanohara-field data
!=========================================================

  !### write by Goda 2011.
  !### modify by PT 2015.04.18
  !### modify by Tanaka 2015.07.30

  !! input parameters
! ndim     : dimension                                (former np)
! npop     : sample size                              (former nn)
! itermax  : maximum iteration steps                    (former nt)
! ntrial   : No. of repeat count for each optimization run (former nr)
! nobjf    : No. of objective function
! seed_self: random seed indicator
! ngpu     : No. of GPU (<= 2)
! nhq      : No. of thread for each GPU (No. of Hyper-Q)

! nwm      : weighting model
! entmax   : Max. value of entropy
! nsm      : sampling model
! ncut     : No. of sample cut
! reent    : convergnece criterion for entropy
! nconvg   : permissible no. of entropy's convergence

  !! output parameters
! ymin_avg : average value of objective functions
! ymin_end : Min. value of objective functions at the end of optimization
! ymin_sd  : standard deviation of objective functions

! nunit    : device number

  !! module for parameter arrays
  use param_label
  !### modify by Tanaka 2015.04.05 --->
  use input1_module, only: ndim, npop, itermax
  use input2_module
  use input3_module
  use gpu_module
  !### modify by Tanaka 2015.04.05 <---
  !### modify by Tanaka 2015.04.06 --->
  use param_seed
  use param_time
  !### modify by Tanaka 2015.04.06 <---

  !### add by Tanaka 2015.04.08 --->
  use param_hm
  use param_domain
  use param_basisfunc
  use param_inj
  !### add by Tanaka 2015.04.08 <---

  !! module for subroutines
  !### modify by Tanaka 2015.04.03 --->
  use ilhs_module
```

```
  use input_module
  use mkout_module
  use second_module
  use param_t2out
  !### modify by Tanaka 2015.04.03 <---
  use mpi_module !### add by PT 2015.04.18
  implicit none

  !! other parameters
  integer(kind = i4) :: i, nunit

  !### add by PT --->
!  character*64 :: dname,fname
!  character*64 :: scr_job,scr_mc,scr_t2c,scr_t2g
!  character*64 :: path
!  namelist /execore/ dname,fname,scr_job,scr_mc,scr_t2c,scr_t2g,path
!  character*128 :: command
!  character*128, allocatable :: dir(:)
!  character*64 :: inputfile
!  logical:: ex

!  open(10,file="execore.cfg",action="read")
!  read(10,execore)
!  close(10)
!!  print *,dname,fname
!!  print *,scr_job,scr_mc,scr_t2c,scr_t2g
!!  print *,path
!  inquire(file=trim(scr_job),exist=ex)
!  inquire(file=trim(scr_mc),exist=ex)
!  inquire(file=trim(scr_t2c),exist=ex)
!  inquire(file=trim(scr_t2g),exist=ex)
!  if(ex==.false.) then
!    write(*,*)
!    write(*,*) "  error :: not exist scr file"
!    write(*,*) "          ",trim(scr_job)," ",trim(scr_mc)," ",trim(scr_t2c)," 
",trim(scr_t2g)
!    write(*,*)
!    stop
!  end if
  !### add by PT <---

  !### add by PT 2015.04.18 --->
  call mpi_init(ierr)
  call mpi_comm_size(mpi_comm_world,nprocs,ierr)
  call mpi_comm_rank(mpi_comm_world,myrank,ierr)
  !### add by PT 2015.04.18 <---

  !### add by Tanaka 2015.04.03 --->
  call random_seed (size = seedsize)
  allocate (seed(seedsize))
  call random_seed (get = seed)
  !### add by Tanaka 2015.04.03 <---

  if( myrank == 0 ) then   !### add by PT 2015.04.18
    !### modify by Tanaka 2015.01.13 --->
    !! open output files
    open (18, file = 'out_SORT.dat',    status = 'replace')
    open (19, file = 'out_XPOP.dat',    status = 'replace')
```

```
    !### modify by Tanaka 2015.04.27 --->
    open (20, file = 'out_ENTROPY.dat', status = 'replace')

!    open (21, file = 'out_RANGE.dat',   status = 'replace')
    open (22, file = 'out_ROT1.dat',    status = 'replace')
    open (23, file = 'out_NRNDM.dat',   status = 'replace')
    open (24, file = 'out_MNRNDM.dat',  status = 'replace')
    open (25, file = 'out_GCOPULA.dat', status = 'replace')
    open (26, file = 'out_CHLSKY.dat',  status = 'replace')
    !### modify by Tanaka 2015.04.27 <---
    !### modify by Tanaka 2015.01.13 <---

    !! set start time
!    call second (tzero) !### add by Tanaka 2015.04.03

    !! write start time
    call date_and_time (exedate, exetime, exezone, exeiv)
    write (18, '(a12, i5, 2(a1, i2.2), 3(a1, i2.2))') &
   & "Start Time :", exeiv(1), "/", exeiv(2), "/", exeiv(3), " ", &
   &               exeiv(5), ":", exeiv(6), ":", exeiv(7)
  endif !### add by PT 20115.04.18
  call mpi_barrier(mpi_comm_world,ierr) !### add by PT 20115.04.18

  !! read input files
  call input !### modify by Tanaka 2015.04.08

  do i = 1, seedsize
    seed(i) = seed(i) + seed_self
  end do
  call random_seed(put = seed)

  !### modify by PT 20115.04.18 --->
  if( myrank == 0 ) then
    !! set time to read input
    call second (exeiv,elt1)
  endif
!  call second (elt1)
!  elt1 = elt1 - tzero
  !### modify by PT 20115.04.18 <---

  !! make directories for output files of tough2/eco2n
  !! prepare for name change of output files 1
  call mkout                          !### modify by Tanaka 2015.04.08
  call mpi_barrier(mpi_comm_world,ierr) !### modify by PT 20115.04.18

  !! main part of program
  call ilhs !### modify by Tanaka 2015.04.08

  if( myrank == 0 ) then !### modify by PT 20115.04.18
    !### add by Tanaka 2014.09.06 --->
    write (18, *) !### add by Tanaka 2013.03.15
    write (20, *) !### add by Tanaka 2013.12.09
    write (21, *) !### add by Tanaka 2014.04.09
!    write (23, *) !### add by Tanaka 2014.04.09

    !! calculate elapsed time
    !### modify by PT 20115.04.18 --->
    call second (exeiv,elt)
```

```
!    call second (elt)
!    elt = elt - tzero
    !### modify by PT 20115.04.18 <---
    eltc = elt - elt1

    !! write elapsed time
    nunit = 18 !### modify by Tanaka 2015.04.15
    write (nunit, '(a27)') " End of ILHS Simulation Run "
    write (nunit, '(a23, f9.2, a5)') " ------ Elapsed Time = ", elt,  " sec "
    write (nunit, '(a23, f9.2, a5)') " -- Calculation Time = ", eltc, " sec "
    write (nunit, '(a23, f9.2, a5)') " --- Data Input Time = ", elt1, " sec "

    call date_and_time (exedate, exetime, exezone, exeiv)
    write (nunit, '(a12, i5, 2(a1, i2.2), 3(a1, i2.2))') &
   & "End time :",  exeiv(1), "/", exeiv(2), "/", exeiv(3)," ", &
   &             exeiv(5), ":", exeiv(6), ":", exeiv(7)

    !! close output files
    do nunit = 18, 20
!    do nunit = 18, 21
      close (nunit)
    end do
!    close (23) !add by Tanaka 2014.04.24
    !### add by Tanaka 2014.09.06 <---
  endif !### add by PT 2015.04.18
  !### add by PT 2015.04.18 --->
  call mpi_barrier(mpi_comm_world,ierr)
  call mpi_finalize(ierr)
  !### add by PT 2015.04.18 <---

  stop

end program main
```

- fctnWALSH.f90

```
module walsh_module
  implicit none
contains

!  function walsh (x, n, nmax)
  function walsh(x, n)
    !! walsh function defined in [-1, 1]

    !### write by Goda 2011.
    !### modify by Tanaka 2015.11.16
!
    !! Parameters
!   x      : Primary variable of Walsh Function, x
!   n      : Sequency, Coefficient of Walsh Function, k
!   ixdum(j): Exponent of j digit in Binary Expansion, x(r+1)
!   ndum(j) : Coefficient in Binary Expansion. k(r)


    use param_label !### add by Tanaka 2015.09.13
    implicit none
!###    implicit double precision(a-h, o-z)

    integer(kind = i4), parameter :: nmax = 1e+04
    integer(kind = i4) :: imod, iy, ixdum(nmax), j, m, n, nn, ndum(nmax)
    real   (kind = r8) :: x, xx, walsh

    !! Calculate Coefficients in Binary Expansion, ndum
    nn = n
    do j = 1, nmax
      ndum(j) = imod(nn, 2)    !set exponent of j digit
      nn = (nn - ndum(j)) / 2   !move to next digit
      if ( nn == 0 ) then      !end of binary expansion
        m = j
        goto 10
      end if
    end do
 10 continue

    !! Calculate Exponents in Binary Expansion, ixdum
    xx = x
    iy = 0
    do j = 1, m
      if ( xx >= 0.5d0 ) then
        ixdum(j) = 1
      else
        ixdum(j) = 0
      end if
      xx = 2.d0 * xx - dble(ixdum(j))
      iy = iy + ndum(j) * ixdum(j)
    end do

    walsh = (-1.d0)**iy

    return

  end function walsh
```

```
end module walsh_module
```

- mdlILHS.f90

```
!! mdlILHS.f90

!### write by Tanaka 2015.01.28
!### modify by Tanaka 2015.11.04

module param_label
  implicit none

  integer, parameter :: i4 = 4, &
                        r8 = 8
end module param_label


module input1_module
  use param_label
  implicit none

  integer(kind = i4) :: ndim, npop, &
                        ntrial, itermax, nobjf, &
                        itrial, iter, iobjf
end module input1_module

module input2_module
  use param_label
  implicit none

  integer(kind = i4) :: nwm, nsm
  real   (kind = r8) :: entmax
end module input2_module

module input3_module
  use param_label
  implicit none

  integer(kind = i4) :: ncut, nconvg
  real   (kind = r8) :: reent
end module input3_module


!### add by PT 2015.04.18 --->
!module cpu_module
!  use param_label
!  implicit none

!  integer(kind = i4) :: nt2c, nt2c_c
!  character(len=15)  :: tough2_c
!end module cpu_module
!### add by PT 2015.04.18 <---

!### modify by PT 2015.04.18 --->
!module gpu_module
!  use param_label
!  implicit none

!  integer(kind = i4) :: ngpu, nhq, nt2g, nt2g_c, ndvc(2)
!  real   (kind = r8) :: gpu_ratio
```

```
!  character(len=15)  :: tough2_g
!end module gpu_module
!### modify by PT 2015.04.18 <---

!### add by PT 2015.04.18 --->
!module mpi_module
!  use param_label
!  implicit none
!  include 'mpif.h'

!  integer(kind = i4) :: ista, iend, istate
!  integer(kind = i4), allocatable :: ista_list(:), iend_list(:)
!  integer(kind = i4) :: ierr, nprocs, myrank
!end module mpi_module
!### add by PT 2015.04.18 <---


module param_obj
  use param_label
  implicit none

  real   (kind = r8), allocatable :: xpop(:,:), yopt_mat(:)

contains

  subroutine alloc_obj
    use input1_module, only: ndim, npop
    implicit none

    allocate( xpop(npop, ndim), yopt_mat(npop) )
  end subroutine alloc_obj
end module param_obj


module param_ymin
  use param_label
  implicit none

  real   (kind = r8) :: ymin
end module param_ymin


module param_rank
  use param_label
  use input1_module, only: npop
  implicit none

  integer(kind = i4), allocatable :: irank(:)

contains

  subroutine alloc_rank
    implicit none

    allocate( irank(npop) )
  end subroutine alloc_rank

end module param_rank
```

```
module param_lhs
  use param_label
  use input1_module, only: npop
  implicit none

  real   (kind = r8), allocatable :: xbdy(:), xpop_new(:), xbdy_new(:)

contains

  subroutine alloc_lhs
    implicit none

    allocate( xbdy(npop), xpop_new(npop), xbdy_new(npop) )
  end subroutine alloc_lhs

end module param_lhs


module param_wgt
  use param_label
  use input1_module, only: npop
  implicit none

  real   (kind = r8)            :: sum_wgt1
  real   (kind = r8), allocatable :: wgt(:), wgt_mat(:)

contains

  subroutine alloc_wgt
    implicit none

    allocate( wgt(npop), wgt_mat(npop) )
  end subroutine alloc_wgt

end module param_wgt


module param_ent
  use param_label
  use input1_module, only: ndim, npop
  implicit none

  real   (kind = r8)            :: gam, beta, ent_yopt, ent_tmp !### modify
by Tanaka 2015.07.28
  real   (kind = r8), allocatable :: xside(:,:)

contains

  subroutine alloc_ent
    implicit none

    allocate( xside(npop, ndim) )
  end subroutine alloc_ent

end module param_ent
```

```
!### add by Tanaka 2015.07.30 --->
module param_dent
  use param_label
  use input1_module, only: ndim
  implicit none

  integer(kind = i4)            :: ndent
  real   (kind = r8), allocatable :: ent_xpop(:), bef_xpop(:), &
                                dent_xpop(:), dent_min(:), dent_lim(:)

contains

  subroutine alloc_dent1
    implicit none

    allocate( ent_xpop(ndim), bef_xpop(ndim), &
             dent_xpop(ndim), dent_min(ndim) )
  end subroutine alloc_dent1

  subroutine alloc_dent2
    implicit none

    allocate( dent_lim(ndim) )
  end subroutine alloc_dent2

end module param_dent
!### add by Tanaka 2015.07.30 <---


module param_others
  use param_label

  use input1_module
  implicit none

  integer(kind = i4), allocatable :: nent(:), norder_mat(:,:)
  real   (kind = r8), allocatable :: xorder_mat(:,:), &
                                xbdy_mat(:,:), xpop_mat(:,:), &
                                ymin_end(:), xopt(:), x_dum(:)

contains

  subroutine alloc_others
    implicit none

    allocate( nent(ndim), norder_mat(npop, ndim), &
             xorder_mat(npop, ndim), &
             xpop_mat(npop, ndim), xbdy_mat(npop, ndim), &
             xopt(ndim), &
             ymin_end(ntrial), &
             x_dum(0:npop) )
  end subroutine alloc_others

end module param_others


!### add by Tanaka 2015.04.27 --->
```

```
module param_copula
  use param_label
  implicit none

  integer(kind = i4)            :: nrows, ncols
  real   (kind = r8)            :: rho, tau
  real   (kind = r8), allocatable :: u(:,:), a(:,:), &
                                rdum_mat(:,:,:)
!                                rdum_gc(:,:,:)

contains

  subroutine alloc_copula
    use input1_module, only: itermax, ndim, npop
    implicit none

    allocate( u(nrows, ncols), a(ncols, ncols), &
             rdum_mat(ncols, ndim, nrows) )
!             rdum_gc(npop, ndim, itermax) )
  end subroutine alloc_copula

end module param_copula
!### add by Tanaka 2015.04.27 <---

module param_seed
  use param_label
  implicit none

  integer(kind = i4), allocatable :: seed(:)
  integer(kind = i4)            :: seedsize, seed_self
end module param_seed


module param_time
  use param_label
  implicit none

  !! variables for elapsed time measurement
  real(kind = r8)    :: tzero, elt, elt1, eltc
  character(len=10)  :: exedate, exetime, exezone
  integer(kind = i4) :: exeiv(1:8)
end module param_time
```

- mdlINIT.f90

  TOUGH2 に関わる内容のため省略

- mdlOBJ.f90

```
!! mdlOBJ.f90

!### write by Tanaka 2015.04.08
!### modify by Tanaka 2015.11.04

module param_hm
  use param_label
  implicit none

  integer(kind = i4)            :: npdat
  !### modify by Tanaka 2015.09.26 --->
  real   (kind = r8)            :: yopt, time1, wtobj(5), co2dat(2)
!  real   (kind = r8)            :: yy, time1, wtobj(3)
  !### modify by Tanaka 2015.09.26 <---
  real   (kind = r8), allocatable :: tdat(:), pdat1(:), pdat4(:)
!  real   (kind = r8), allocatable :: tdat(:), pdat1(:), pdat4(:), co2dat(:)

contains

  subroutine alloc_hm
    implicit none

    allocate( tdat(npdat+1), pdat1(npdat+1), pdat4(npdat+1) )
!           allocate(   tdat(npdat+1),   pdat1(npdat+1),   pdat4(npdat+1),
co2dat(npdat+1) )
  end subroutine alloc_hm

end module param_hm


module param_domain
  use param_label
  implicit none

  real   (kind = r8), allocatable :: xmin(:), xmax(:)

contains

  subroutine alloc_domain
    use input1_module, only: ndim
    implicit none

    allocate( xmin(ndim), xmax(ndim) )
  end subroutine alloc_domain

end module param_domain


module param_basisfunc
  use param_label
  implicit none
```

```
  integer(kind = i4)            :: nbssf
  real   (kind = r8), allocatable :: igycod(:,:)

contains

  subroutine alloc_basisfunc
    implicit none

    allocate( igycod(2, nbssf) )
  end subroutine alloc_basisfunc

end module param_basisfunc


!### add by Tanaka 2015.11.04 --->
module param_distr
  use param_label
  use input1_module, only: npop
  implicit none

  integer(kind = i4)            :: idistr
  real   (kind = r8), allocatable :: rho_mat(:)

contains

  subroutine alloc_distr
    implicit none

    allocate( rho_mat(npop) )
  end subroutine alloc_distr

end module param_distr
!### add by Tanaka 2015.11.04 <---


module param_inj
  use param_label
  implicit none

  integer(kind = i4) :: nw, nrot

end module param_inj


module param_t2out
  use param_label
  implicit none

  character(len= 15) :: fnm(16)
  integer(kind = i4) :: ifnm(16)

end module param_t2out


module param_beta
  use param_label
  implicit none
```

```
  real   (kind = r8), allocatable :: beta1(:)

contains

  subroutine alloc_beta
    use input1_module, only: ndim
    implicit none

    allocate( beta1(ndim) )
  end subroutine alloc_beta

end module param_beta


!### add by Tanaka 2015.09.11 --->
module param_output
  use param_label
  implicit none

  real   (kind = r8), allocatable :: time_mat(:)

contains

  subroutine alloc_output
    use input1_module, only: npop
    implicit none

    allocate( time_mat(npop) )
  end subroutine alloc_output

end module param_output
!### add by Tanaka 2015.09.11 <---
```

- mdlPROC.f90

```
!! mdlPROC.f90

!### write by Tanaka 2015.08.28
!### modify by PT 2015.04.18

!### add by PT 2015.04.18 --->
module cpu_module
  use param_label
  implicit none

  integer(kind = i4) :: nt2c, nt2c_c
  character(len=15)  :: tough2_c
end module cpu_module
!### add by PT 2015.04.18 <---

!### modify by PT 2015.04.18 --->
module gpu_module
  use param_label
  implicit none

  integer(kind = i4) :: ngpu, nhq, nt2g, nt2g_c, ndvc(2)
  real   (kind = r8) :: gpu_ratio
  character(len=15)  :: tough2_g
end module gpu_module
!### modify by PT 2015.04.18 <---

!### add by PT 2015.04.18 --->
module mpi_module
  use param_label
  implicit none
  include 'mpif.h'

  integer(kind = i4) :: ista,iend,istate
  integer(kind = i4),allocatable :: ista_list(:),iend_list(:)
  integer(kind = i4) :: ierr,nprocs,myrank
end module mpi_module
!### add by PT 2015.04.18 <---
```

- subCDF.f90

  A.1と同じ
- subCHLSKY.f90

  A.1と同じ
- subENT.f90

  A.1と同じ
- subGAMMA.f90

  A.1と同じ
- subGCOPULA.f90

  A.1と同じ

- subILHS.f90

```
module ilhs_module
  implicit none
contains

  subroutine ilhs

    !### write by Tanaka 2015.04.03
    !### modify by PT 2015.04.18
    !### modify by Tanaka 2015.11.05

    !! input parameters
!   xpop     : value of sorted sample point [0, 1] (former xlhs)

    !! output parameters
!   xopt     : value of sorted sample point [0, 1]
!   yopt     : value of objective function (former yout)
!   yopt_mat  : value of objective function (former yopt)

!   ymin_avg  : average value of objective functions
!   ymin_end  : Min. value of objective functions at the end of optimization
!   ymin_sd   : standard deviation of objective functions

    !! other parameters

!   norder_mat: order of sampling point                  [1, npop] (former
norder1)
!   norder    : order of sampling point                  [1, npop] (former
norder2)
!   xpop_mat  : value of sampling point                     [0, 1] (former
xlhs_mat, xc)
!   xbdy_mat  : boundary value of sample space              (0, 1] (former
xc2)
!   xbdy     : boundary value of sample space              (0, 1] (former
xdum2)
!   xpop_new  : value of sample point in next step           [0, 1] (former
xlhs_new, xdum3)
!   xbdy_new  : boundary value of sample space in next step    (0, 1] (former
xdum4)
!   xorder_mat: random number to decide sample order         [0, 1] (former
xdum)
!   wgt      : weighting depending on objective function value       (former
ydum)
!   wgt_mat   : weighting depending on objective function value      (former
ydum2)
!   sum_wgt1  : sum of weighting depending on objective function value (former
c1)
!   ymin     : Min. value of objective function
!   ymin_tmp  : Min. value of objective function for each step
!   irank    : rank of objective function value          [1, npop]
!   xside    : length of sample space side                [0, 1] (former
range_pv)
!   ent_yopt  : normalized entropy of yopt                        (former
ent_yout, ent1)
!   ent_xpop  : normalized entropy of xpop                        (former
ent_xlhs, ent2)
!   bef_xpop  : binary entropy function of xpop
```

```
!   dent_xpop : deference between normalized entropy and binary entropy
function of xpop
!   dent_min  : Min. value of deference between normalized entropy and binary
entropy function of xpop
!   dent_lim  : lower limit value of deference between normalized entropy and
binary entropy function of xpop
!   gam      : gamma, exponential constant in Zipf's law
!   beta     : exponential constant in canonical distribution
!   nent     : No. of entropy convergence
!   rdum     : random number

!   ndim     : dimension                                          (former np)
!   npop     : sample size                                        (former nn)
!   itermax   : maximum iteration steps                           (former nt)
!   ntrial    : No. of repeat count for each optimization run         (former
nr)
!   nobjf    : No. of objective function
!   seed_self : random seed indicator
!   ngpu     : No. of GPU (<= 2)
!   nhq      : No. of thread for each GPU (No. of Hyper-Q)

!   nwm      : weighting model
!   rho      : Sperman's rho
!   tau      : Kendall's tau
!   nrows    : No. of rows of random numbers depending on copula
!   ncols    : No. of columns of random numbers depending on copula
!   entmax   : Max. value of entropy
!   nsm      : sampling model
!   ncut     : No. of sample cut
!   reent    : convergnece criterion for entropy
!   nconvg   : permissible no. of entropy's convergence

!   nunit    : device number

   !! modules for parameter arrays
   use param_label !### add by Tanaka 2015.04.06
   !### add by Tanaka 2015.04.05 --->
   use input1_module, only: ndim, npop, itermax, iter, itrial
   use input2_module
   use input3_module

   use cpu_module !### add by PT 2915.04.18
   use gpu_module

   use param_obj
   use param_ymin
   !### add by Tanaka 2015.04.05 <---
   !### add by Tanaka 2015.04.06 --->
   use param_rank
   use param_lhs
   use param_wgt
   use param_ent
   use param_dent   !### add by Tanaka 2015.07.30
   use param_others
   use param_copula !### add by Tanaka 2015.04.27
   !### add by Tanaka 2015.04.06 <---

   !### add by Tanaka 2015.04.08 --->
```

```
    use param_hm
    use param_domain
    use param_basisfunc
    use param_distr  !### add by Tanaka 2015.11.04
    use param_inj
    use param_t2out
    !### add by Tanaka 2015.04.08 <---
    use param_beta   !### add by Tanaka 2015.04.09
    use param_output !### add by Tanaka 2015.09.11

    !### modify by Tanaka 2015.04.03 --->
    !! modules for subroutines
    use cdf_module
    use ent_module
    use gamma_module !### add by Tanaka 2015.04.08
    use lhs_module
    use obj_module
    use sort_module
    use weight_module
    !### modify by Tanaka 2015.04.28 --->
    use gcopula_module
    use mvnrnd_module
    !### modify by Tanaka 2015.04.28 <---

    use init_module
    use mvout_module
    use mpi_module !### add by PT 2015.04.18
    use loopdiv_module
    !debug
    use param_seed
    !### modify by Tanaka 2015.04.03 <---
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, im, nunit
    real  (kind = r8) :: rdum
    real  (kind = r8) :: tmin, ymin2
    integer(kind = i4) :: ient(ndim)
    real  (kind = r8) :: ymin_each
    real  (kind = r8) :: elt_tmp1, elt_tmp2 !### add by Tanaka 2015.08.26
    integer(kind = i4) :: nn1, nn2, mod1
    character(len=128) :: command

    !### add by PT 2015.04.18 --->
    integer :: access, ymin_i
    character(len=15) :: input_files_dir='input_files/'
    integer,parameter :: input_files_num=7
    character(len=30), dimension(input_files_num) :: input_files
    character(len=13), parameter :: out_dir_head='output_files_'
    character(len=30) :: output_files,dir
    !### add by PT 2015.04.18 <---
    integer(kind = i4) :: ireq1, ireq2, mstatus(MPI_STATUS_SIZE) !### add by
Tanaka 2015.09.16

    !! allocate variables
    !### moidfy by Tanaka 2015.04.06 --->
    call alloc_rank
    call alloc_lhs
```

```
    call alloc_wgt
    call alloc_ent
    call alloc_dent1  !### add by Tanaka 2015.07.08
    call alloc_others
    !### moidfy by Tanaka 2015.04.06 <---
    call alloc_obj    !### add by Tanaka 2015.04.08
    call alloc_beta   !### add by Tanaka 2015.04.09
    call alloc_output !### add by Tanaka 2015.09.11
    call alloc_distr  !### add by Tanaka 2015.11.04

    myrank1: if( myrank == 0 ) then !### add by PT 2015.04.18
      !! computation of exponent
      !! set zipf's law parameter, gamma
      if ( nwm == 1 ) call gamma !### modify by Tanaka 2015.04.08

      !! set random numbers using copula function
      !### modify by Tanaka 2015.04.27 --->
      if ( nsm == 2 ) then
        nrows = itermax
        ncols = npop
        call alloc_copula
        call mvnrnd
        do i = 1, ndim
          call gcopula
          do j = 1, itermax
            rdum_mat(:,i,j) = u(j,:)
          end do
        end do
      end if

      !! optimazation start
      !! set values of sampling parameters
      iobjf  = nobjf
      itrial = ntrial
!     do iobjf = 1, nobjf
!       do itrial = 1, ntrial
!         print *, iobjf, itrial !debug
      !### modify by Tanaka 2015.04.27 <---

      !! Parameter Initialization
      !### add by PT 2015.04.18 --->
      ymin  = huge(ymin)
      ymin2 = huge(ymin)
      !### add by PT 2015.04.18 <---

      if ( nwm == 2 ) then
        ent_yopt = entmax
      end if
      !### add by Tanaka 2015.07.31 --->
      irank(1:npop) = 0
      nent(1:ndim)  = 0
      ent_xpop = 1.d0
      bef_xpop = 0.d0
      dent_min = 1.d0
      wgt      = 1.d0
      !### add by Tanaka 2015.07.31 <---
      rho_mat  = 1.d0 !### add by Tanaka 2015.11.05
```

```
      !debug
      call random_seed(put=seed)

      !! construct initial LHS
      call lhs_ini !### modify by Tanaka 2015.11.03

    endif myrank1 !### add by PT 2015.04.18

    !### add by PT 2015.04.18 --->
    call mpi_barrier(mpi_comm_world,ierr)

    nt2g_c = ngpu * nhq
    nn1    = nt2c_c + nt2g_c
    nn2    = (npop - 1) / nn1 + 1
!    mod1   = mod(npop, nn1)

    if ( nprocs /= nn1 ) then
      write(*,'(2(a,i0))') "error: nprocs = ",nprocs,", nn1 = ",nn1
      call MPI_Abort(MPI_COMM_WORLD, 99, ierr)
    endif

    !### modify by Tanaka 2015.09.08 --->
    nt2g = ceiling((gpu_ratio * nt2g_c) / ((gpu_ratio * nt2g_c) + nt2c_c) *
npop)
!    nt2g = (gpu_ratio * nt2g_c) / ((gpu_ratio * nt2g_c) + nt2c_c) * npop
    !### modify by Tanaka 2015.09.08 <---
    nt2c = npop - nt2g

    if ( myrank < nt2g_c ) then
      call loopdiv(nt2g,myrank,nt2g_c,ista,iend)
    else
      call loopdiv(nt2c,myrank-nt2g_c,nt2c_c,ista,iend)
      ista = ista + nt2g
      iend = iend + nt2g
    endif

    if ( myrank == 0 ) then
      allocate(ista_list(nprocs))
      allocate(iend_list(nprocs))
    endif
    call mpi_barrier(mpi_comm_world,ierr)

    !### modify by Tanaka 2015.09.11 --->
    call
mpi_gather(ista,1,MPI_INTEGER,ista_list(1),1,MPI_INTEGER,0,mpi_comm_world
,ierr)
    call
mpi_gather(iend,1,MPI_INTEGER,iend_list(1),1,MPI_INTEGER,0,mpi_comm_world
,ierr)
!                                                                       call
mpi_gather(ista,1,MPI_INTEGER,ista_list(1),1,MPI_INTEGER4,0,mpi_comm_worl
d,ierr)
!                                                                       call
mpi_gather(iend,1,MPI_INTEGER,iend_list(1),1,MPI_INTEGER4,0,mpi_comm_worl
d,ierr)
    !### modify by Tanaka 2015.09.11 <---
    call mpi_barrier(mpi_comm_world,ierr)
```

```
    !debug
    if( myrank == 0 ) then
      write(*,*) "ista_list = ",ista_list
      write(*,*) "iend_list = ",iend_list
    endif
    call mpi_barrier(mpi_comm_world,ierr)

    input_files(1) = 'CO2TAB'
    input_files(2) = 'GENER'
    input_files(3) = 'MESH'
    input_files(4) = 'MESHA'
    input_files(5) = 'MESHB'
    input_files(6) = 'PCAP.csv'
    input_files(7) = 'RELP.csv'

    !check input files existence
    do i = 1, input_files_num
      input_files(i)                                                    =
trim(adjustl(input_files_dir))//trim(adjustl(input_files(i)))
    enddo

    if( myrank == 0 ) then !### add by PT 2015.04.18
      do i = 1, input_files_num
        if( access(input_files(i),' ') /= 0 ) then
          write(*,*) 'error: not exists ',input_files(i)
          stop
        end if
      end do
    end if
    call mpi_barrier(mpi_comm_world,ierr)
    !### add by PT 2015.04.18 <---

!    irank(1:npop) = 0

    !! calculate objective functions
    do iter = 1, itermax !### moidfy by Tanaka 2015.04.05
      call cpu_time (elt_tmp1) !debug, add by Tanaka 2015.08.26
      !### comment out by Tanaka 2015.11.02 --->
!     myrank2: if( myrank == 0 ) then !### add by PT 2015.04.18

        !debug
!       call random_seed(put=seed)

        !! set value of LHS
        !### modify by Tanaka 2015.04.27 --->
!       if ( nsm == 1 ) then
!         do j = 1, ndim
!           do i = 1, npop
!             call random_number(rdum)
!             xorder_mat(i, j) = rdum
!           end do
!         end do
!       else if ( nsm == 2 ) then
!         xorder_mat(:,:) = rdum_mat(iter,:,:)
!       end if
        !### modify by Tanaka 2015.04.27 <---

        !! set order of LHS
```

```
!       do j = 1, ndim !### add by Tanaka 2015.04.27
!         do i = 1, npop
!           norder(i) = 1
!           do im = 1, npop
!             if ( xorder_mat(i, j) > xorder_mat(im, j) ) then
!               norder(i) = norder(i) + 1
!             end if
!           end do
!           norder_mat(i, j) = norder(i)
!           xpop(i, j)   = xpop_mat(norder(i), j)
!         end do
!       end do
        !### add by Tanaka 2014.09.05 <---
!      end if myrank2 !### add by PT 2015.04.18
      !### comment out by Tanaka 2015.11.02 <---

      !### modify by PT 2015.04.27 --->
      call mpi_barrier(mpi_comm_world,ierr)

      call mpi_bcast(xpop(1,1),npop*ndim,MPI_REAL8,0,mpi_comm_world,ierr)
      call mpi_bcast(ymin,1,MPI_REAL8,0,mpi_comm_world,ierr)

      do i = ista, iend
        !! debug
!        print *, myrank !### add by Tanaka 2015.08.26

        write (command, '(a, i0)') 'rm -rf task',myrank
        call system (trim(command))
        write (command, '(a, i0)') 'mkdir -p task',myrank
        call system (trim(command))

        write          (command,          '(3a,i0)')          'ln          -s
`pwd`/',trim(tough2_c),' ./task',myrank
        call system (trim(command))
        write          (command,          '(3a,i0)')          'ln          -s
`pwd`/',trim(tough2_g),' ./task',myrank
        call system (trim(command))
        do j = 1, input_files_num
          write          (command,          '(3a,i0)')          'ln          -s
`pwd`/',input_files(j),' ./task',myrank
          call system (trim(command))
        end do

        print *, iter, i !debug
        do j = 1, ndim
          beta1(j) = xpop(i, j) * ( xmax(j) - xmin(j) ) + xmin(j)
        end do
        call init(myrank) !### modify by Tanaka 2015.04.09


        if( myrank < nt2g_c ) then
          write (command, '(a,i0,2a,i0,a)') 'cd task',myrank,' && ./', &
                        tough2_g,ndvc(mod(myrank,ngpu)+1),' < flow.inp > OUT'
          !! debug
!          print *, myrank, command !### add by Tanaka 2015.09.02
          call system (command)
        else
          write (command, '(a,i0,3a)') 'cd task',myrank,' && ./', &
```

```
                    tough2_c,' < flow.inp > OUT'
         !! debug
!         print *, myrank, command !### add by Tanaka 2015.09.02
          call system (command)
        endif

        call obj(myrank) !### modify by Tanaka 2015.04.08

        !### add by Tanaka 2015.08.26 --->
        call cpu_time (elt_tmp2)
        print *, "Calculation End", iter, i
        print *, "Calculation Time = ", elt_tmp2 - elt_tmp1, "sec"
        !### add by Tanaka 2015.08.26 <---
        yopt_mat(i) = yopt  !### modify by Tanaka 2015.09.26
        time_mat(i) = time1 !### modify by Tanaka 2015.09.16

        !### modify by Tanaka 2015.09.26 --->
        if( yopt < ymin ) then
!        if( yopt(i) < ymin ) then
        !### modify by Tanaka 2015.09.26 <---
          write(command,'(2a,i0)') 'mkdir ',out_dir_head,i
          call system(command)
          do j = 1, 16
            if ( ifnm(j) >= 1 ) then
              write (command, '(a, i0, 4a, i0, 2a)') &
              'mv            -f            ./task',myrank,'/',trim(fnm(j)),'
',out_dir_head,i,'/',trim(fnm(j))
              call system(command)
            end if
          end do
        end if
      end do
      call mpi_barrier(mpi_comm_world,ierr)

      !### modify by Tanaka 2015.09.17 --->
      !! Merge time data for history matching, time_mat
      if ( myrank /= 0 ) then
        call mpi_isend(time_mat(ista),iend-ista+1,MPI_REAL8,0,myrank, &
       &            mpi_comm_world,ireq1,ierr)
        call mpi_wait( ireq1,mstatus,ierr )
      else
        do i = 2,nprocs
          call  mpi_irecv(time_mat(ista_list(i)),iend_list(i)-ista_list(i)+1,
&
         &           MPI_REAL8,i-1,i-1,mpi_comm_world,ireq2,ierr)
          call mpi_wait( ireq2,mstatus,ierr )
        end do
      endif

      !! Merge objective function value data, yopt_mat
      if ( myrank /= 0 ) then
        call mpi_isend(yopt_mat(ista),iend-ista+1,MPI_REAL8,0,myrank, &
       &            mpi_comm_world,ireq1,ierr)
        call mpi_wait( ireq1,mstatus,ierr )
      else
        do i = 2,nprocs
          call  mpi_irecv(yopt_mat(ista_list(i)),iend_list(i)-ista_list(i)+1,
&
```

```
        &            MPI_REAL8,i-1,i-1,mpi_comm_world,ireq2,ierr)
         call mpi_wait( ireq2,mstatus,ierr )
        end do
      endif
!     if ( myrank /= 0 ) then
!                                                                       call
mpi_send(yopt_mat(ista),iend-ista+1,MPI_REAL8,0,myrank,mpi_comm_world,ier
r)
!     else
!       do i = 2,nprocs
!         call mpi_recv(yopt_mat(ista_list(i)),iend_list(i)-ista_list(i)+1,
&
!        &            MPI_REAL8,i-1,i-1,mpi_comm_world,istate,ierr)
!       end do
!     endif
      !### modify by Tanaka 2015.09.17 <---

!     if( myrank == 0 ) then
!                                                                       call
mpi_gather(MPI_IN_PLACE,iend-ista+1,MPI_REAL8,yopt_mat(1),iend-ista+1,MPI
_REAL8,0,mpi_comm_world,ierr)
!     else
!                                                                       call
mpi_gather(yopt_mat(ista),iend-ista+1,MPI_REAL8,yopt_mat(1),iend-ista+1,M
PI_REAL8,0,mpi_comm_world,ierr)
!     endif
!     call mpi_barrier(mpi_comm_world,ierr)

      myrank3: if( myrank == 0 ) then

       !debug
!       write(*,*) "yopt = ",yopt_mat

       nunit = 19
       do i = 1, npop
         write (nunit, '(2i5, 2e15.7)', advance = 'no') itrial, iter,
time_mat(i), yopt_mat(i)

         do j = 1, ndim
           write (nunit, '(e15.7)', advance = 'no') xpop(i, j)
         end do

         write (nunit, '(e15.7)', advance = 'no') wgt(i)
         do j = 1, ndim
           write (nunit, '(2e15.7)', advance = 'no') xpop_mat(i,j),
xbdy_mat(i,j)
         end do
         write (nunit, *)
       end do

       !! optimum or not
       ymin_i = minloc(yopt_mat(1:npop),dim=1)
       if ( yopt_mat(ymin_i) < ymin ) then
         ymin = yopt_mat(ymin_i)
         tmin = time_mat(ymin_i)
         write(*,*) "update optimal value, ymin = ",ymin !debug

         call mvout(ymin_i)
```

```
          do j = 1, ndim
            !### add by Tanaka 2015.09.19 --->
            !! Binary Entropy Function
            bef_xpop(j)  = - (xopt(j) * log(xopt(j)) + (1.d0 - xopt(j)) &
           &           * log(1.d0 - xopt(j))) / log(dble(npop))
            !### add by Tanaka 2015.09.19 <---
            xopt(j) = xpop(ymin_i, j)
          end do
        end if
        do i = 1,npop
          irank(i) = i
        end do
        command = 'rm -rf '//out_dir_head//'*'
        call system(command)
        !### modify by PT 2015.04.27 <---

        !! rank transformation
        call sort !### modify by Tanaka 2015.04.08

        !### add by Tanaka 2015.07.31 --->
        do j = 1, ndim
          !! Binary Entropy Function
!         bef_xpop(j)  = - (xopt(j) * log(xopt(j)) + (1.d0 - xopt(j)) &
!         &           * log(1.d0 - xopt(j))) / log(dble(npop))
          !! Difference between Normalized Entropy and Normalized BEF
          dent_xpop(j) = ent_xpop(j) - bef_xpop(j)

          if ( dent_xpop(j) < dent_min(j) .and. dent_xpop(j) >= 0.d0 ) then
            dent_min(j) = dent_xpop(j)
          end if
        end do

        nunit = 20
        write (nunit, '(2i5)', advance = 'no') itrial, iter
        do j = 1, ndim
          write (nunit, '(e15.7)', advance = 'no') ent_xpop(j)
        end do
        do j = 1, ndim
          write (nunit, '(e15.7)', advance = 'no') dent_min(j)
        end do
        write (nunit, *)
        !### add by Tanaka 2015.07.31 <---

        !! Set Weight Parameter
        call weight !### modify by Tanaka 2015.04.08

        !debug
        call random_seed(put=seed)

        !! set value and order of new LHS
        do j = 1, ndim
          do i = 1, npop
!             xdum(norder_mat(i, j))    = xpop(i, j) !### add by Tanaka
2014.09.07
            xbdy(i)             = xbdy_mat(i, j)
            wgt_mat(norder_mat(i, j)) = wgt(i)
          end do
```

```
        !### modify by Tanaka 2015.04.08 --->
        !! Set Cumulative Density Function
        call cdf

        !! calculate probability density function (pdf, [0,1]) to calculate
Normalized Entropy
        x_dum(0)     = 0.d0
        x_dum(1:npop) = xbdy_new(1:npop)
        !### modify by Tanaka 2015.04.08 <---
        do i = 1, npop
          xpop_mat(i, j) = xpop_new(i)
          xbdy_mat(i, j) = xbdy_new(i)
          xside(i, j)    = x_dum(i) - x_dum(i-1)       !### add by Tanaka
2015.04.08
        end do

        !### add by Tanaka 2014.09.07 --->
        !! calculate Normalized Entropy
        call ent (xside(1:npop, j)) !### modify by Tanaka 2015.04.08
        ent_xpop(j) = ent_tmp
!         ent_xpop(iter, j) = ent_tmp

!         write (20, '(e15.7)', advance = 'no') ent_tmp !### modify by Tanaka
2015.04.27
        !### add by Tanaka 2014.09.07 <---
       end do

!       write (20, *) !### modify by Tanaka 2015.04.27

       !! write output data
       !### add by Tanaka 2014.11.05 --->
       write (19, *)

       ymin_each = minval(yopt_mat, mask = yopt_mat > 0)
       nunit = 18
       !### modify by Tanaka 2015.09.19 --->
       write (nunit, '(2i5, 3e15.7)', advance = 'no') &
      & itrial, iter, tmin, ymin, ymin_each
!       write (nunit, '(i4, 2x, i3, 2x, i4, 3e15.7)', advance = 'no') &
!      & iter, itrial, iter*npop, tmin, ymin, ymin_each
       !### modify by Tanaka 2015.09.19 <---

       do j = 1, ndim
         write (nunit, '(e15.7)', advance = 'no') xopt(j)
       end do

       if ( nwm == 1 ) then
         write (nunit, '(2e15.7)', advance = 'no') ent_yopt, gam
       else if ( nwm == 2 ) then
         write (nunit, '(2e15.7)', advance = 'no') ent_yopt, beta
       end if
       write (nunit, *)

       !### add by Tanaka 2015.11.03 --->
       do j = 1, ndim
         if ( dent_min(j) >= dent_lim(j) ) exit
         goto 1111
```

```
end do

!! set order of new LHS
!### modify by Tanaka 2015.11.09 --->
if ( idistr == 2 ) then
  do j = nbssf + 1, ndim - nbssf
    do i = 1, npop

      !! set order of LHS
      call lhs_ord (i, j, iter + 1)

    end do

    !! make sample set
    call lhs_set (j)

  end do

  !! set Sperman's rho for each sample set
  rho_mat(1:npop) = xpop(1:npop, ndim - nbssf)

  do j = 1, nbssf
    !! primary variables about distribution data
    do i = 1, npop

      !! set order of LHS
      call lhs_ord (i, j, 1)
    end do

    call lhs_set (j)
    call lhs_set (j + ndim - nbssf)
  end do

  !! debug
  do j = 1, ndim
    write (96, '(i5)', advance = 'no') iter+1
    write (97, '(i5)', advance = 'no') iter+1
    write (98, '(i5)', advance = 'no') iter+1
    do i = 1, npop
      write (96, '(2e15.7)', advance = 'no') xpop(i, j)
      write (97, '(2e15.7)', advance = 'no') xpop_mat(i, j)
      write (98, '(2e15.7)', advance = 'no') xorder_mat(i, j)
    end do
    do i = 96, 98
      write (i, *)
    end do
  end do
  do j = 96, 98
    write (j, *)
  end do

else

  do j = 1, ndim

    do i = 1, npop
      call lhs_ord (i, j, 1) !### modify by Tanaka 2015.11.03
    end do
```

```
           call lhs_set (j)        !### modify by Tanaka 2015.11.03

         end do

        end if
        !### modify by Tanaka 2015.11.09 <---
        !### add by Tanaka 2015.11.03 <---

      end if myrank3
      call mpi_barrier(mpi_comm_world,ierr) !### add by PT 2015.04.18

    end do

1111 call system("rm -rf task*")
!    call system("rm -rf task*")

    !### add by Tanaka 2015.04.27 --->
!     end do
!    end do
    !### add by Tanaka 2015.04.27 <---

    return

  end subroutine ilhs

end module ilhs_module
```

- subINIT.f90

  TOUGH2 に関わる内容のため省略

- subINPUT.f90

```
module input_module
  implicit none
contains

  subroutine input

    !### write by Tanaka 2014.12.26
    !### modify by Tanaka 2015.11.11

    !! output parameters
!   ndim    : dimension (former np)
!   npop    : sample size (former nn)
!   itermax  : maximum iteration steps (former nt)
!   ntrial   : No. of repeat count for each optimization run (former nr)
!   nobjf   : No. of objective function
!   seed_self: random seed indicator
!   ngpu    : No. of GPU (<= 2)
!   nhq     : No. of thread for each GPU (No. of Hyper-Q)

!   nwm     : weighting model
!   nconvg   : flag of entropy convergnece
!   ndent   : No. of dent_lim
!   dent_lim : lower limit value of dent
!   rho     : Sperman's rho
!   tau     : Kendall's tau
!   entmax   : Max. value of entropy
!   nsm     : sampling model
!   ncut    : No. of sample cut

    !! other parameter
!   nunit   : device number

    !! module for parameter arrays
    !### modify by Tanaka 2015.04.08 --->
    use param_label
    use param_seed, only: seed_self
    !### modify by Tanaka 2015.04.08 <---
    !### add by Tanaka 2015.04.05 --->
    use input1_module
    use input2_module
    use input3_module
    use cpu_module
    use gpu_module
    use param_dent                !### add by Tanaka 2015.07.31
    use param_copula, only: rho, tau !### add by Tanaka 2015.04.28
    !### add by Tanaka 2015.04.05 <---

    !### add by Tanaka 2015.04.08 --->
    use param_hm
    use param_domain
    use param_basisfunc
```

```
    use param_distr, only: idistr   !### add by Tanaka 2015.11.04
    use param_inj
    use param_t2out, only: ifnm
    !### add by Tanaka 2015.04.08 <---

    !! module for subroutines
    use readc_module
    use mpi_module !### add by PT 2015.04.18
    implicit none

    !! other parameters
    integer(kind = i4) :: i, nunit
    real   (kind = r8) :: dent_tmp !### add by Tanaka 2015.07.31
    character(len= 20) :: file_name!### add by PT 2015.04.18
    character(len=128) :: line !### text line buffer added by Tanaka 2015.04.16


    !! read input file
    !### modify by PT 2015.04.18 --->
    call get_command_argument(1,file_name)
    nunit = 60
    open(nunit,file=file_name,status='old')
!    nunit = 5
    !### modify by PT 2015.04.18 <---

    !### modified by PT --->
    call readc (nunit)                    !skip comments
    !### modify by Tanaka 2015.04.28 --->
    read (nunit, *) ndim, nbssf, npop, itermax, ntrial, nobjf !dimension, sample
size, maximum iteration steps, no. of trials
!    read (nunit, *) ndim, nbssf, npop       !number of parameters, basis
function, sample size per one iteration
    !### modify by Tanaka 2015.04.28 <---

    !### add by Tanaka 2015.10.29 --->
    call readc (nunit)
    read (nunit, *) idistr
    !### add by Tanaka 2015.10.29 <---
    !### add by Tanaka 2015.04.08 --->
    call alloc_domain
    call alloc_basisfunc
    !### add by Tanaka 2015.04.08 <---

    call readc (nunit)
    do i = 1, nbssf
      read (nunit, *) xmin(i), xmax(i), igycod(1, i), igycod(2, i)
      !### add by Tanaka 2015.11.11 --->
      if ( idistr == 2 ) then
        xmin(ndim - nbssf + i) = xmin(i)
        xmax(ndim - nbssf + i) = xmax(i)
      end if
      !### add by Tanaka 2015.11.11 <---
    end do
    !### modify by Tanaka 2015.11.11 --->
    do i = nbssf + 1, ndim - nbssf
!    do i = nbssf + 1, ndim
    !### modify by Tanaka 2015.11.11 <---
      read (nunit, *) xmin(i), xmax(i)
```

```
    end do

    call readc (nunit)
    read (nunit, *) npdat                   !pressure data for history matching

    call alloc_hm !### add by Tanaka 2015.04.08

    call readc (nunit)
    do i = 1, npdat
      read (nunit, *) tdat(i), pdat1(i), pdat4(i)
    end do
    !### modify by Tanaka 2015.01.05 --->
    call readc (nunit)
    read (nunit, *) tdat(npdat+1)
!    tdat(npdat+1) = 1.d+20
    !### modify by Tanaka 2015.01.05 <---
    call readc (nunit)
    !### add by Tanaka 2015.09.22 --->
    read (nunit, *) co2dat(1), co2dat(2)     !CO2 arrival time
!    read (nunit, *) co2dat(2), co2dat(4)     !CO2 arrival time

    call readc (nunit)
    read (nunit, *) (wtobj(i), i = 1, 5)     !weighting for objective function
!     read (nunit, *) wtobj(1), wtobj(2)         !weighting for objective
function
    !### add by Tanaka 2015.09.22 <---

!    call getarg(1,inputfile)
!    inputfile=adjustl(inputfile)
!    inquire(file=trim(inputfile),exist=ex)
!    if(ex==.false.) then
!      write(*,*)
!      write(*,*) "  error :: not exist input file"
!      write(*,*)
!      stop
!    end if
!    open(10,file=trim(inputfile),action="read")

!    call random_seed()
!    read(10,*) ndim                      !number of parameters
!    read(10,*) npop                      !sample size per one iteration
!    do i=1,ndim
!      read(10,*) xmin(i), xmax(i)
!    end do
!    read(10,*) itermax                    !iteration times

!     read(10,*) npdat                          !pressure data for history
matching
!    do i=1,npdat
!      read(10,*) tdat(i),pdat1(i),pdat4(i)
!    end do
!    tdat(npdat+1)=1.e+20
!    read(10,*) co2dat(2),co2dat(4)          !co2 arrival time

!    close(10)
    !### modified by PT <---

    call readc (nunit)
```

```
    !### modify by Tanaka 2015.07.31 --->
    read (nunit, *) nw, nrot                !number of wells and inj./pro.
rotation
    call readc (nunit)
    read (nunit, '(a)') line
    read (line, *) nwm, entmax, nconvg
    call alloc_dent2
    if ( nconvg == 1 ) then
      read (line, *) nwm, entmax, nconvg, dent_tmp
      dent_lim(1:ndim) = dent_tmp
    else if ( nconvg == 2 ) then
      read (nunit, *) (dent_lim(i), i = 1, ndim)
    end if
!    read (nunit, *) nwm, entmin, entmax      !weighting model and range of
normalized entropy
    !### modify by Tanaka 2015.07.31 <---

    call readc (nunit)
    !### modify by Tanaka 2015.07.31 --->
    read (nunit, '(a)') line
    read (line, *) nsm                     !sampling model
    if (nsm == 2) then
      read (line, *) nsm, rho, tau          !sampling model, Sperman's rho,
Kendall's tau
    end if
!    call readc (nunit)
!    read (nunit, *) ncut, reent, nconvg      !sampling model and number of
sample cut
    !### modify by Tanaka 2015.07.31 <---

    call readc (nunit)
    do i = 1, 16
      read (nunit, *) ifnm(i)
    end do
    call readc (nunit)
    read (nunit, *) seed_self
    call readc (nunit)
    read (nunit, *) tough2_c, tough2_g
    call readc (nunit)
    read (nunit, *) gpu_ratio
    call readc (nunit)
    read (nunit, *) nt2c_c
    !### modify by Tanaka 2015.01.13 --->
    call readc (nunit)
    read (nunit, *) ngpu, nhq, ndvc(1), ndvc(2)
    !### add by Tanaka 2015.04.18 --->
    !debug
    call mpi_barrier(mpi_comm_world,ierr)
    if( myrank == 0 ) then
      print *, "(tough2_c, tough2_g) = (",tough2_c, ",", tough2_g, ")"
      print *, "(nt2c_c, ngpu, nhq) = (", nt2c_c, ",", ngpu, ",", nhq, ")"
      print *, "(ndvc(1), ndvc(2)) = (", ndvc(1), ",", ndvc(2), ")"
    endif
    !### add by Tanaka 2015.04.18 <---
    !### modify by Tanaka 2015.01.13 <---

    return
```

```
  end subroutine input

end module input_module
```

- subLHS.f90

```
module lhs_module
  implicit none
contains

  subroutine lhs_ini

    !### write by Tanaka 2015.11.03
    !### modify by Tanaka 2015.11.05

    !! this subroutine makes initial sample sets generated by Latin Hypercube
Sampling

    !! input parameters
!   ndim     : dimension
!   npop     : number of sampling points
!   nsm      : sampling model

    !! output parameters
!   norder_mat: order of sampling point                  [1, npop]
!   xpop_mat  : value of sampling point                    [0, 1]
!   xbdy_mat  : boundary value of sample space              (0, 1]
!   xorder_mat: random number to decide sample order          [0, 1]

    !! other parameters
!   norder    : order of sampling point                 [1, npop]
!   rdum      : random number

    !! module for parameter arrays
    use param_label
    use input1_module,   only: ndim, npop
    use param_obj,      only: xpop
    use param_others,    only: norder_mat, xorder_mat, xbdy_mat, xpop_mat
    use param_copula

    use param_basisfunc, only: nbssf
    use param_distr

    use gcopula_module
    use mvnrnd_module
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, im
    real   (kind = r8) :: rdum


    !### modify by Tanaka 2015.11.03 --->
    !! permeability and porosity case
    if ( idistr == 2 ) then
      do j = nbssf + 1, ndim - nbssf
        do i = 1, npop

          !! set value of LHS
          call random_number(rdum)
          xpop_mat(i, j) = (dble(i) - rdum) / dble(npop)
          xbdy_mat(i, j) = dble(i) / dble(npop)
```

```
          !! set order of LHS
          call lhs_ord (i, j, 1)

        end do

        !! make sample set
        call lhs_set (j)

      end do

      !! set Sperman's rho for each sample set
!      rho_mat(1:npop) = 0.99d0
      rho_mat(1:npop) = xpop(1:npop, ndim - nbssf)

      do j = 1, nbssf
        !! primary variables about distribution data
        do i = 1, npop

          !! for permeability distribution
          xpop_mat(i, j) = (dble(i) - rdum) / dble(npop)
          xbdy_mat(i, j) = dble(i) / dble(npop)

          !! for porosity distribution
          xpop_mat(i, j + ndim - nbssf) = (dble(i) - rdum) / dble(npop)
          xbdy_mat(i, j + ndim - nbssf) = dble(i) / dble(npop)

          !! set order of LHS
          call lhs_ord (i, j, 1)
        end do

        call lhs_set (j)
        call lhs_set (j + ndim - nbssf)
      end do

      !! debug
      do j = 96, 98
        write (j, *) ndim, npop
      end do
      do j = 1, ndim
        write (96, '(a5)', advance = 'no') "1"
        write (97, '(a5)', advance = 'no') "1"
        write (98, '(a5)', advance = 'no') "1"
        do i = 1, npop
          write (96, '(2e15.7)', advance = 'no') xpop(i, j)
          write (97, '(2e15.7)', advance = 'no') xpop_mat(i, j)
          write (98, '(2e15.7)', advance = 'no') xorder_mat(i, j)
        end do
        do i = 96, 98
          write (i, *)
        end do
      end do
      do j = 96, 98
        write (j, *)
      end do

    else
      do j = 1, ndim
```

```
       do i = 1, npop
         call random_number(rdum)
         xpop_mat(i, j) = (dble(i) - rdum) / dble(npop)
         xbdy_mat(i, j) = dble(i) / dble(npop)
       end do
     end do
   end if
!    do j = 1, ndim
!      do i = 1, npop

       !! set value of LHS
!       call random_number(rdum)
!       xpop_mat(i, j) = (dble(i) - rdum) / dble(npop)
!       xbdy_mat(i, j) = dble(i) / dble(npop)

       !! set order of LHS
!       call lhs_ord (i, j, 1)

!     end do

!     call lhs_set (j)

!    end do
   !### modify by Tanaka 2015.11.03 <---

   return

  end subroutine lhs_ini


  subroutine lhs_ord (i, j, k)

    !### write by Tanaka 2015.11.03
    !### modify by Tanaka 2015.11.04

    !! this subroutine set random values to decide order of Latin Hypercube
Samples

    !! input parameters
!   nsm      : sampling model

    !! output parameters
!   xorder_mat: random number to decide sample order          [0, 1]

    !! other parameters
!   rdum     : random number

    !! module for parameter arrays
    use param_label
    use input1_module,   only: ndim, npop
    use input2_module,   only: nsm
    use param_others,    only: xorder_mat
    use param_copula

    use param_basisfunc, only: nbssf
    use param_distr

    use gcopula_module
```

```
    use mvnrnd_module
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, k
    real   (kind = r8) :: rdum


    !### modify by Tanaka 2015.11.04 --->
    if ( idistr == 2 ) then

      if ( j >= nbssf + 1 .and. j <= ndim - nbssf ) then
        call random_number(rdum)
        xorder_mat(i, j) = rdum

      else if ( j >= 1 .and. j <= nbssf ) then
        nrows = npop
!       nrows = nbssf
        ncols = 2
!       nrows = 2
!       ncols = nbssf
        call alloc_copula
        rho = rho_mat(i)

        !! generate random number matrix
        call mvnrnd
        call gcopula

        !! for permeability distribution
        xorder_mat(i, j) = u(i, 1)

        !! for porosity distribution
        xorder_mat(i, j + ndim - nbssf) = u(i, 2)
      end if

    else

      if ( nsm == 1 ) then
        call random_number(rdum)
        xorder_mat(i, j) = rdum
      else if ( nsm == 2 ) then
        xorder_mat(i, j) = rdum_mat(i, j, k)
      end if

    end if
!    if ( nsm == 1 ) then
!      call random_number(rdum)
!      xorder_mat(i, j) = rdum
!    else if ( nsm == 2 ) then
!      xorder_mat(i, j) = rdum_mat(i, j, k)
!    end if
    !### modify by Tanaka 2015.11.04 <---

    return

  end subroutine lhs_ord
```

```
  subroutine lhs_set(j)

    !### write by Tanaka 2015.11.03

    !! this subroutine makes sample sets generated by Latin Hypercube Sampling

    !! input parameters
!   npop     : number of sampling points
!   xorder_mat: random number to decide sample order          [0, 1]

    !! output parameters
!   norder_mat: order of sampling point                    [1, npop]
!   xpop_mat  : value of sampling point                      [0, 1]

    !! other parameters
!   norder   : order of sampling point                    [1, npop]

    !! module for parameter arrays
    use param_label
    use input1_module, only: npop
    use param_obj,    only: xpop
    use param_others,  only: norder_mat, xorder_mat, xbdy_mat, xpop_mat
    implicit none

    !! input parameters
    integer(kind = i4) :: j

    !! other parameters
    integer(kind = i4) :: i, im, norder(npop)

    do i = 1, npop
      norder(i) = 1
      do im = 1, npop
        if ( xorder_mat(i, j) > xorder_mat(im, j) ) then
          norder(i) = norder(i) + 1
        end if
      end do
      norder_mat(i, j) = norder(i)
      xpop(i, j)      = xpop_mat(norder(i), j)
    end do

    !! debug
!    do i = 1, npop
!!     write (97, '(i5)', advance = 'no') norder(i)
!     write (97, '(2e15.7)', advance = 'no') xpop_mat(norder(i), j)
!!     write (97, '(2e15.7)', advance = 'no') xpop_mat(i, j)
!    end do
!    write (97, *)

    return

  end subroutine lhs_set


end module lhs_module
```

- subLOOPDIV.f90

```
module loopdiv_module
  implicit none
contains

  subroutine loopdiv(n,myrank,nprocs,ista,iend)

    !### write by PT 2015.04.18

    use param_label

    integer(kind = i4) :: n,nprocs,myrank
    integer(kind = i4) :: ista,iend,iwork1,iwork2

    iwork1 = n/nprocs
    iwork2 = mod(n,nprocs)
    if(myrank < (nprocs-iwork2)) then
      ista = iwork1*(myrank)+1
      iend = ista+(iwork1-1)
    else
      iend = n-(iwork1+1)*(nprocs-(myrank+1))
      ista = iend-iwork1
    end if
  end subroutine loopdiv
end module
```

- subMKOUT.f90

```
module mkout_module
  implicit none
contains

  subroutine mkout

    !### write by Tanaka 2014.12.24
    !### modify by PT 2015.04.18.
    !### modify by Tanaka 2015.09.19

    !! input parameters
!   ndim    :   number of primary variables
!   npop    :   number of sampling points
!   itermax :   number of iteration steps
!   nw      :   number of wells
!   nrot    :   number of inj./pro. rotation
!   ifnm    :   number of file name

    !! output parameters
!   fnm     :   file name

    !! module for parameter arrays
    use param_label
    use param_seed, only: seed_self
    !### add by Tanaka 2015.07.31 --->
    use input1_module, only: ndim, npop, itermax, ntrial
    use input2_module, only: nwm
    !### add by Tanaka 2015.07.31 <---
    !### add by Tanaka 2015.04.08 --->
    use param_inj
    use param_t2out
    !### add by Tanaka 2015.04.08 <---
    use mpi_module, only: myrank !### add by PT 2015.04.18
    implicit none

    !! other parameters
    integer(kind = i4) :: i, j, nunit !### modify by Tanaka 2015.04.28
    character(len=128) :: command    !### add by Tanaka 2014.12.23


    !! prepare for name change of output files 1
    write (fnm( 1), '(a5)') 'INCON'
    write (fnm( 2), '(a6)') 'CO2TAB'
    write (fnm( 3), '(a5)') 'GENER'
    write (fnm( 4), '(a4)') 'MESH'
    write (fnm( 5), '(a4)') 'INIT'
    write (fnm( 6), '(a3)') 'OUT'
    write (fnm( 7), '(a4)') 'SAVE'
    write (fnm( 8), '(a5)') 'TABLE'
    write (fnm( 9), '(a3)') 'ffi'
    write (fnm(10), '(a4)') 'MINC'
    write (fnm(11), '(a4)') 'VERS'
    write (fnm(12), '(a4)') 'FOFT'
    write (fnm(13), '(a4)') 'GOFT'
    write (fnm(14), '(a4)') 'COFT'
    write (fnm(15), '(a5)') 'LINEQ'
```

```
   write (fnm(16), '(a4)') 'MASS'

   if(myrank == 0) then !### add by PT 2015.04.18
     !! creat directories for output files of tough2/eco2n
     do i = 1, 16
       write (command, '(2a)') 'rm -rf file_', trim(fnm(i))
       call system (trim(command))

       if (ifnm(i) >= 1) then
         write (command, '(2a)') 'mkdir -p file_', trim(fnm(i))
         call system (trim(command))
       end if
     end do

     !! write headers of output files
     !### add by Tanaka 2015.07.31 --->
     do i = 18, 20
       write (i, '(a36, 4(i4, a1))') &
        "(ndim, npop, itermax, ntrial) = (", ndim, ",", npop, ",", itermax,
",", ntrial, ")"
       write (i, '(a14, i9)') "seed_self =", seed_self
       write (i, *)
     end do
     !### add by Tanaka 2015.07.31 <---

     !! out_sort.dat
     nunit = 18
!     write (nunit, '(a12, 2(i4, a1))') "(npop, itermax) = (", npop, ",",
itermax, ")"
!     write (nunit, '(a14, i9)') "seed_self =", seed_self
!     write (nunit, *)
     !### modify by Tanaka 2015.09.11 --->
     write (nunit, '(2a5, 3(a15))', advance = 'no') &
    & "ntri", "iter", "Time", "Ymin", "ymin_in_nn"
!    & "ll", "l", "l*npop", "Time", "Ymin", "ymin_in_nn"

     do i = 1, ndim
       write (command, '(a11, i0, a1)', advance = 'no') "xopt(", i, ")"
       write (nunit, '(a15)', advance = 'no') trim(command)
     end do
     !### modify by Tanaka 2015.09.11 <---

     if ( nwm == 1 ) then
       write (nunit, '(2(a15))', advance = 'no') "NormalizedEnt","Gamma"
     else if ( nwm == 2 ) then
       write (nunit, '(2(a15))', advance = 'no') "NormalizedEnt", "Beta"
     end if
     write (nunit, *)


     !! out_xpop.dat
     nunit = 19
!     write (nunit, '(a12, 2(i4, a1))') "(npop, itermax) = (", npop, ",",
itermax, ")"
!     write (nunit, '(a14, i9)') "seed_self =", seed_self
!     write (nunit, *)
     !### modify by Tanaka 2015.07.31 --->
     write (nunit, '(2a5, 2(a15))', advance = 'no') "ntri", "iter", "Time",
```

```
"Yout"
!     write (nunit, '(a4, a5, 2(a15))', advance = 'no') "ll", "l", "Time",
"Yout"
     !### modify by Tanaka 2015.07.31 <---

     do i = 1, ndim
       write (command, '(a7, 2(i0, a1))', advance = 'no') "xpop(", npop, ",",
i, ")"
       write (nunit, '(a15)', advance = 'no') trim(command)
     end do

!     do i = 1, nw
!       write (nunit, '(2(a4, i1))', advance = 'no') "x", i, "y", i
!     end do

!     do i = 1, nw
!       do j = 1, nrot
!         write (nunit, '(a7, 2(i3, a1))', advance = 'no') "xrate(", i, ",",
j, ")"
!       end do
!     end do

     write (nunit, '(a15)', advance = 'no') "Weighting"
     do i = 1, ndim
       write (command, '(a11, i0, a1)', advance = 'no') "xpop_mat(", i,")"
       write (nunit, '(a15)', advance = 'no') trim(command)
       write (command, '(a11, i0, a1)', advance = 'no') "xbdy_mat(", i, ")"
       write (nunit, '(a15)', advance = 'no') trim(command)
     end do
     write (nunit, *)


     !! out_entropy.dat
     nunit = 20
!     write (nunit, '(a12, 2(i4, a1))') "(npop, itermax) = (", npop, ",",
itermax, ")"
!     write (nunit, '(a4, a5)', advance = 'no') "l", "ll"

     write (nunit, '(2a5)', advance = 'no') "ntri", "iter"
     do i = 1, ndim
       write (command, '(a11, i0, a1)', advance = 'no') "NormEnt(", i,")"
       write (nunit, '(a15)', advance = 'no') trim(command)
     end do
     do j = 1, ndim
       write (command, '(a10, i0, a1)') "DifEntMin(", j, ")"
       write (nunit, '(a15)', advance = 'no') trim(command)
     end do
     write (nunit, *)


     !! out_range.dat
     !### modify by Tanaka 2014.11.05 --->
!     nunit = 21
!     write (nunit, '(a12, 2(i4, a1))') "(npop, itermax) = (", npop, ",",
itermax, ")"
!     write (nunit, '(a3, a5)', advance = 'no') "ll", "ndim"

!     do i = 1, npop
```

```
!       write (command, '(a11, i0, a1)', advance = 'no') "xdum4_old(", i,")"
!       write (nunit, '(a15)', advance = 'no') trim(command)
!     end do

!     do i = 1, npop
!       write (command, '(a11, i0, a1)', advance = 'no') "xdum4_new(", i,")"
!       write (nunit, '(a15)', advance = 'no') trim(command)
!     end do
!     write (nunit, *)

!     write (nunit, '(3(a3))', advance = 'no') "ll", "l", "i"
!     write (nunit, '(a4)', advance = 'no') "lhs"
!     do i = 1, npop
!       write (nunit, '(a2, i2, a3)', advance = 'no') "(", i, ",i)"
!     end do
!     do i = 1, npop
!       write (nunit, '(a9, i2, a4)', advance = 'no') "yout(", i, ", i)"
!     end do
!     do i = 1, npop
!       write (nunit, '(a10, i2, a4)', advance = 'no') "yout(lhs(", i, ",i))"
!     end do
!     write (nunit, '(5(a9))', advance = 'no') &
!    & "irank1", "spml_min", "spml_max", "nx_min", "nx_max"
!     do i = 1, npop
!       write (nunit, '(a9, i2, a4)', advance = 'no') "yout1(", i, ", i)"
!     end do
!     do i = 1, npop
!      write (nunit, '(a14, i2, a2)', advance = 'no') "yout1(i,lhs3(", i, "))"
!     end do
!     write (nunit, *)
      !### modify by Tanaka 2014.11.05 <---

    end if !### add by PT 2015.04.18

  end subroutine mkout

end module mkout_module
```

- subMVNPND.f90
- A.1と同じ

- subMVOUT.f90

```
module mvout_module
  implicit none
contains

  subroutine mvout(dirnum)

    !### write by tanaka 2015.02.28
    !### modify by PT 2015.04.18
    !### modify by tanaka 2015.04.28

    !! input parameters
!   itrial :   current number of simulation run
!   iter   :   current number of iteration step
!   ifnm   :   number of file name
!   fnm    :   file name

    !! module for parameter arrays
    !### add by Tanaka 2015.04.06 --->
    use param_label
    use input1_module, only: itrial, iter
    use param_t2out
    !### add by Tanaka 2015.04.06 <---

    implicit none

    !! other parameters
    integer(kind = i4) :: dirnum !### add by PT 2015.04.18
    integer(kind = i4) :: i, j
    character(len=128) :: command !### add by Tanaka 2014.12.23

    character (len=30) :: dir_name !### add by PT 2015.04.18

    !! move output files
    do i = 1, 16
      if (ifnm(i) >= 1) then
        !! mv <file_name>  file_<file_name>/<file_name>itrial-iter.dat
        !### modify by PT 2015.04.18 --->
        write(dir_name,'(a,i0)')'output_files_',dirnum
        write (command, '(8a, 2(i0, a))') &
       & 'mv -f ',trim(dir_name),'/',trim(fnm(i)), ' file_', trim(fnm(i)), '/',
&
       & trim(fnm(i)), itrial, '-', iter, '.dat'
        !### modify by PT 2015.04.18 <---
        call system (trim(command))
      end if
    end do

  end subroutine mvout

end module mvout_module
```

- subOBJ.f90

```
module obj_module
  implicit none
contains

  subroutine obj(dirnum)

    !### write by Goda 2011.
    !### modify by PT 2015.04.18
    !### modify by Tanaka 2015.09.26

    !! module for parameter arrays
    use param_label           !### add by Tanaka 2015.04.06
    use dfm_module, only: timax !### add by Tanaka 2015.04.01
    use param_hm             !### add by Tanaka 2015.04.08

    !! module for subroutines
    use readc_module
    use second_module         !### add by Tanaka 2015.08.04

!debug
use param_seed

    implicit none

    !! other parameters
    integer(kind = i4) :: dirnum !### add by PT 2015.04.18
    integer(kind = i4) :: j, j1, jobs2, jobs3, jobs4, npass
    real  (kind = r8) ::  c1,  c2,  c3,  c4,  c5,  c6,  c7,  c8,  c9, c10, &
                        c11, c12, c13, c14, c15, c16, c17, c18, c19, c20, &
                        c21, c22, c23, c24, c25, c26, c27, c28, c29, c30, &
                        c31, c32, c33, c34, c35, c36, c37, c38, c39, c40, &
                        c41, c42, c43, c44, c45, c46, c47, c48, c49, c50, &
                        c51, c52, c53, c54, c55, c56, c57, c58, c59, c60, &
                        c61, c62, c63, c64, c65, c66, c67, c68, c69, c70, &
                        c71, c72, c73, c74, &
                        csim2, csim3, csim4, psim4, rdum
    real  (kind = r8) :: yopt_term(5) !### add by Tanaka 2015.09.26
    character*30 file_name !### add by PT 2015.04.18

!   tmax = 63162000 !end-time of TOUGH2 simulation
    yopt = 0.d0
    yopt_term = 0.d0 !### add by Tanaka 2015.09.26

    !! Read pressure profile data at IW-1
    !### modify by PT 2015.04.18 --->
    write(file_name,'(a,i0,a)') 'task',dirnum,'/GOFT'
    open (22, file = file_name, status = 'old')
!   open (22, file = 'GOFT', status = 'old')
    !### modify by PT 2015.04.18 <---
    read (22, *)
    npass = 0
    j  = 0
    j1 = 1
 10 j  = j + 1
    read (22, *, end = 100) c1, c2, &               !kcyc, Time
   &                    c3, c4, c5, c6, c7, c8, c9 !IW-1: 2J27
```

```
   !! Compare time in simulation output and sampled observation data
   time1 = c2
 11 if ( time1 > tdat(j1)*0.999d0 .and. time1 < tdat(j1)*1.001d0 ) then
     !! Calculate objective fuction of pressure defference
     !! between simulation and sampled observation data
     !### modify by Tanaka 2015.09.26 --->
     yopt_term(1)  =  yopt_term(1)  +  wtobj(1)  *  ((c9  -  pdat1(j1))  /
pdat1(j1))**2.d0
!     yopt = yopt + wtobj(1) * ((c9 - pdat1(j1)) / pdat1(j1))**2.d0
     !### modify by Tanaka 2015.09.26 <---
     j1 = j1 + 1
     npass = npass + 1
     goto 10
   else if ( time1 <= tdat(j1)*0.999d0 ) then
     goto 10
   else if ( time1 >= tdat(j1)*1.001d0 ) then
!     yopt_term(1) = yopt_term(1) + wtobj(1)
     j1 = j1 + 1
     goto 11
   end if
100 continue
   close (22)
   print *, "npass = ", npass !debug
   if (j1 /= npdat + 1) goto 151


   !! Read pressure profile data at OB-4 and CO2 arrival at OB-2, OB-3 & OB-4
   !### modify by PT 2015.07.31 --->
   write(file_name,'(a,i0,a)') 'task',dirnum,'/FOFT'
!   write(file_name,'(a,i1,a)') 'task',dirnum,'/FOFT'
   open (23, file = file_name, status = 'old')
!   open (23, file = 'FOFT', status = 'old')
   !### modify by PT 2015.07.31 <---
   read (23, *)
   j  = 0
   j1 = 1
   jobs2 = 0
   jobs3 = 0
   jobs4 = 0
!   npass = 0
 20 j = j + 1
   read (23, *, end = 200) c1, c2, &              !kcyc, Time
  &                     c3, c4, c5, c6, c7, c8, & !OB-2: 2K29
  &                     c9,c10,c11,c12,c13,c14, & !OB-2: 3E29
  &                    c15,c16,c17,c18,c19,c20, & !OB-2: 4829
  &                    c21,c22,c23,c24,c25,c26, & !OB-3: 2H13
  &                    c27,c28,c29,c30,c31,c32, & !OB-3: 3A95
  &                    c33,c34,c35,c36,c37,c38, & !OB-3: 4495
  &                    c39,c40,c41,c42,c43,c44, & !OB-4: 2C53
  &                    c45,c46,c47,c48,c49,c50, & !OB-4: 3653
  &                    c51,c52,c53,c54,c55,c56   !OB-4: 4053

   !! Record Time and CO2 saturation
   time1 = c2
   csim2 = max( c6, c12, c18)
   csim3 = max(c24, c30, c36)
   csim4 = max(c42, c48, c54)
```

```
   !! Compare time in simulation output and sampled observation data
   !### modify by Tanaka 2015.04.08 --->
    if ( time1 > tdat(j1)*0.999d0 .and. time1 < tdat(j1)*1.001d0 ) then
!    if ( c2 > tdat(j1)*0.999d0 .and. c2 < tdat(j1)*1.001d0 ) then
    !### modify by Tanaka 2015.04.08 <---

      !! Calculate pressure at OB-4
      !### modify by Tanaka 2015.08.10 --->
      psim4 = c46
!      psim4 = (c40 + c46 + c52) / 3.d0
      !### modify by Tanaka 2015.08.10 <---
      !### modify by Tanaka 2015.09.26 --->
      yopt_term(2) = yopt_term(2) + wtobj(2) * ((psim4 - pdat4(j1)) /
pdat4(j1))**2.d0
!      yopt = yopt + wtobj(2) * ((psim4 - pdat4(j1)) / pdat4(j1))**2.d0
      !### modify by Tanaka 2015.09.26 <---

      !! OB-3, CO2 saturation comparison
      !### add by Tanaka 2015.09.27 --->
      if ( csim3 > 0.d0 ) then
        yopt_term(4) = yopt_term(4) + wtobj(4)
      end if
      !### add by Tanaka 2015.09.27 <---
      j1 = j1 + 1
!      npass = npass + 1
    else if ( time1 >= tdat(j1)*1.001d0 ) then
!      yopt_term(2) = yopt_term(2) + wtobj(2)
      j1 = j1 + 1
    end if

    !! Calculate objective fuction of CO2 arrival time difference
    !! between simulation and sampled observation data
    !! OB-2, CO2 arrival time comparison
    !### modify by Tanaka 2015.09.24 --->
    if ( jobs2 == 0 .and. csim2 > 0.d0 ) then
      jobs2 = 1
      yopt_term(3) = yopt_term(3) + wtobj(3) * ((time1 - co2dat(1)) /
co2dat(1))**2.d0
    end if
    !! OB-2, CO2 saturation comparison @ end point
!    if ( jobs2 == 0 .and. csim2 > 0.d0 .and. time1 <= co2dat(1) ) then
!      jobs2 = 1
!      yopt_term(3) = yopt_term(3) + wtobj(3)
!    end if

    !! OB-4, CO2 arrival time comparison
    if ( jobs4 == 0 .and. csim4 > 0.d0 ) then
      jobs4 = 1
      yopt_term(5) = yopt_term(5) + wtobj(5) * ((time1 - co2dat(2)) /
co2dat(2))**2.d0
    end if
    !! OB-4, CO2 saturation comparison @ end point
!    if ( jobs4 == 0 .and. csim4 > 0.d0 .and. time1 <= co2dat(2) ) then
!      jobs4 = 1
!      yopt_term(5) = yopt_term(5) + wtobj(5)
!    end if
```

```
   goto 20
200 continue
   close (23)

   !! Calculate objective fuction of CO2 saturation difference
   !! between simulation and sampled observation data @ OB-3
   !! OB-3, CO2 saturation comparison @ end point
!    if ( csim3 > 0.d0 ) then
!      yopt_term(4) = yopt_term(4) + wtobj(4)
!    end if
   !### modify by Tanaka 2015.09.24 <---


   !! Not all pressure data are referred
   !! add by Tanaka 2015.09.26 --->
   if ( npass < npdat) then
     yopt_term(1) = yopt_term(1) * real(npdat) / real(npass)
     yopt_term(2) = yopt_term(2) * real(npdat) / real(npass)
     yopt_term(4) = yopt_term(4) * real(npdat) / real(npass)
   end if
   !! add by Tanaka 2015.09.26 <---

   !! No CO2 arrives @ OB-2 or OB-4
   !### modify by Tanaka 2015.09.26 --->
   if ( jobs2 == 0 ) yopt_term(3) = yopt_term(3) + wtobj(3) * ((timax -
co2dat(1)) / co2dat(1))**2.d0
   if ( jobs4 == 0 ) yopt_term(3) = yopt_term(5) + wtobj(5) * ((timax -
co2dat(2)) / co2dat(2))**2.d0
!    if ( jobs2 == 0 ) yopt = yopt + wtobj(3) * ((timax - co2dat(1)) /
co2dat(1))**2.d0
!    if ( jobs4 == 0 ) yopt = yopt + wtobj(5) * ((timax - co2dat(2)) /
co2dat(2))**2.d0
   !### modify by Tanaka 2015.09.26 <---

   yopt = sum(yopt_term) !### add by Tanaka 2015.09.26

   !! Set tentative objective function value when it is zero
   !### modify by Tanaka 2015.09.11 --->
   if ( yopt == 0.d0 .or. time1 < timax * 0.9d0 ) then
!    if ( yopt == 0.d0 ) then
   !### modify by Tanaka 2015.09.11 <---
   !### modify by PT 2015.04.18 --->
  151 continue
!  151 call random_seed()
   !### modify by PT 2015.04.18 <---

     !! debug
     call random_seed(put=seed)

     yopt = 1.d+20
   end if
   close (23)

   print *, "yopt_term =", yopt_term       !debug
   print *, "time =", time1, "yopt =", yopt !debug

   return
```

```
  end subroutine obj

end module obj_module
```

- subREADC.f90
- A.1と同じ

- subSECOND.f90

```
module second_module
  implicit none
contains

  subroutine second (s_values,elt)

    !### write by Tanaka 2014.
    !### modify by PT 2015.04.18

    implicit none

    integer :: s_values(8),e_values(8)
    double precision :: elt

    integer :: icall, mclock
    integer :: tvalues(8)

    character(len=  8) :: e_date
    character(len= 10) :: e_time
    character(len= 10) :: e_zone
    character(len= 19) :: s_time_s, e_time_s
    double precision   :: time_r
    character(len= 10) :: log_time_name = "log_time_t"
    character(len=123) :: command

    save icall
    data icall/0/
    icall = icall + 1

    call date_and_time(e_date,e_time,e_zone,e_values)

    write(s_time_s,'(i0,5(a,i2))')
s_values(1),'-',s_values(2),'-',s_values(3),' ', &
                         s_values(5),':',s_values(6),':',s_values(7)
    write(e_time_s,'(i0,5(a,i2))')
e_values(1),'-',e_values(2),'-',e_values(3),' ', &
                         e_values(5),':',e_values(6),':',e_values(7)

    write(command,*) 't1="',s_time_s,'" && t2="',e_time_s, &
                   '" && echo $(expr `date -d"$t2" +%s` - `date -d"$t1" +%s`)>
'// &
                   log_time_name

    call system(command)
    open (80,file=log_time_name)
    read (80,*) elt
    close(80)
    call system("rm "//log_time_name)

    elt = elt+dble(e_values(8)-s_values(8))/1000.d0

    return

  end subroutine second

end module second_module
```

- subSORT.f90
- A.1と同じ

- subWEIGHT.f90
- A.1と同じ