情報処理学会論文誌　コンピューティングシステム　**Vol.11　No.2　1–16 (Aug. 2018)**

ソフトウェア自動チューニングにおける複数同時性能パラメタ探索手法の提案と評価

望月 大義[1]　藤井 昭宏[1,a)]　田中 輝雄[1,b)]

受付日 2017年12月27日, 採録日 2018年4月20日

概要：ソフトウェア自動チューニングは，プログラムの性能を決定する複数の性能パラメタの最適な組合せを自動的に推定することにより，ユーザのプログラムの性能を向上させる．我々はソフトウェア自動チューニングにおいて，性能パラメタを推定する手法として，離散スプライン関数（d-Spline）を用いた標本点逐次追加型性能パラメタ推定法を提案している．さらに，多次元性能パラメタ推定のために，多次元 d-Spline を用いた標本点逐次追加型性能パラメタ推定法を行ってきた．しかし，多次元 d-Spline の計算コストは 1 次元 d-Spline よりはるかに高いため，性能パラメタの数が増えるにつれて計算コストが大幅に増加する課題があった．そこで本論文では，計算量の少ない 1 次元 d-Spline 探索を繰り返すことにより，多次元性能パラメタの最適値を推定する方法を提案する．この 1 次元 d-Spline 探索を繰り返す方法をテスト関数と実アプリケーションの 4 次元性能パラメタに適用し，評価を行う．実アプリケーションの性能パラメタのすべての組合せの 65,536 通りの初期点から推定を行った．推定した値と真の最適値とのずれの平均を 1.6%に抑えることができた．また，65,536 個の性能パラメタの組合せのうち 0.3%（185.77 個）の組合せの実測のみで推定を行うことができた．推定にかかった時間は実アプリケーションの実行時間に比べ，無視できる時間に抑えることができた．実験的に多次元性能パラメタ推定に反復 1 次元 d-Spline 探索が効率的であることを示した．

キーワード：ソフトウェア自動チューニング，性能パラメタ推定，実行時チューニング

Proposal and Evaluation of Multiple Performance Parameter Search Method in Software Automatic Tuning

Masayoshi Mochizuki[1]　Akihiro Fujii[1,a)]　Teruo Tanaka[1,b)]

Received: December 27, 2017, Accepted: April 20, 2018

Abstract: Software automatic tuning automatically improves the performance of user's program by estimating the optimal combination of multiple performance parameters that determine the performance of the program. In automatic software tuning of software, we propose the incremental performance parameter estimation method using a discrete spline function (d-Spline) as a method to estimate performance parameters. Furthermore, in order to estimate multidimensional performance parameters, we have performed the incremental performance parameter estimation method using multidimensional d-spline in conventional research. However, the computational cost of multidimensional d-Spline is much higher than that of one-dimensional d-Spline. Therefore it becomes a difficult problem as the number of parameters increased. In this paper, we propose a method to estimate the optimal value of multidimensional performance parameters by repeating one-dimensional d-Spline search with small calculation amount. Estimation was made from 65,536 initial points of combinations of performance parameters of real applications. The average of the deviation between the estimated value and the true optimum value could be suppressed to 1.6%. In addition, estimation was completed with 0.3% (185.77) measured combinations out of 65,536 performance parameter combinations in estimation. The time required for the estimation could be kept very small compared with the execution time of the real application. Iterative one-dimensional d-spline search demonstrates high efficiency for multidimensional performance parameter estimation.

Keywords: software automatic performance tuning, performance parameter estimation, dynamic tuning

1. はじめに

近年，計算機性能の向上は，複数の計算ノードからなる高並列計算機やメニーコアなど，多岐にわたる計算機アーキテクチャにより実現されている．そのため，プログラムの性能チューニングは，それぞれのユーザの使用する計算機環境に合わせて対応する必要がある．この課題に対応するために，ソフトウェア自動チューニング技術の研究が進められている [1], [2], [3]．ソフトウェア自動チューニングは，対象とするユーザ・プログラムの性能に影響を与える複数のパラメタ（性能パラメタ）を組み込み，この性能パラメタのとりうる値から最適となる組合せを自動的に選択し，ユーザ・プログラムの実行時間を最短にする．ここで，その性能パラメタのとりうる値の集合を性能パラメタ空間と呼ぶ．性能パラメタのとりうる値は連続値でもよいが，ここでは，すべて離散値を前提とする．したがって，性能パラメタ空間は離散点からなる空間とする．

本研究では，ユーザ・プログラムの実行時にソフトウェア自動チューニングを行う実行時自動チューニングを対象とする．実行時自動チューニングでは，ユーザ・プログラムの反復 1 回ごとにチューニング処理を行う．そのため，推定コスト（特に推定時間）を抑え，ユーザ・プログラムに影響を与えないようにする必要がある．我々は性能パラメタの推定方式として，標本点逐次追加型性能パラメタ推定法（IPPE 法）を提案している [4], [5]．この推定法は，近似関数による最適値の推定に必要な標本点を，最低限の数の標本点から推定を始めて，必要な標本点を逐次的に自動選択・追加する特徴がある．ここで，標本点とは，自動的に選択した性能パラメタのとりうる値の組合せであり，そのときのユーザ・プログラムのチューニング対象とする区間の実行時間がその値となる．近似関数は，標本点を追加するたびに繰り返し更新される．また，各繰り返しにおいて新しい標本点によって更新されなければならない．したがって，近似関数は，(a) 計算に時間がかからないことと，(b) データの動きに柔軟に追随する補間の 2 つの特性を持つ必要がある．これらの条件を満たすために，滑らかさのみを仮定した d-Spline と呼ぶスプライン系の関数を IPPE 法の近似関数として使用する．

近似関数 d-Spline では，「滑らかさ」としてステンシルを用いる．この「滑らかさ」により補間を行う．これまでの研究では，2 次元性能パラメタ空間では 2 次元ステンシルを，3 次元性能パラメタ空間では 3 次元ステンシルを用いてきた．そのため，3 次元性能パラメタ空間では，d-Spline の計算量が増大し，推定のみの時間は推定にかかった全体の時間のうち 1.6%を占めており，無視できない時間となった [6]．したがって，d-Spline による近似は，3 次元が限界と考えられる．なお，2 次元，つまり 2 つの性能パラメタの同時推定を行う IPPE 法を自動チューニング機能付きのプログラムを生成する基盤である ppOpen-AT [7], [8], [9] に実装している [10]．

本研究の目的は，ソフトウェア自動チューニングの計算コストを抑えつつ，より多次元性能パラメタ空間での性能パラメタ推定を可能とする方式を実現することである．そこで本研究では，多次元性能パラメタ空間に対して，1 次元 d-Spline 探索を繰り返し適用する方法を提案する．1 次元 d-Spline の計算は，2.3 節に後述するように 1 つの性能パラメタのとりうる値を n とすると $O(n)$ で計算することが可能である．そのため，一般の大規模行列を扱うような性能チューニングを必要とするユーザ・プログラムに対して，ほとんど影響を与えない．

以下，2 章ではソフトウェア自動チューニングの概説と自動チューニングの手法の標本点逐次追加型性能パラメタ推定法と関連研究について示す．次に，3 章では多次元性能パラメタ空間における 1 次元 d-Spline 探索を繰り返し用いる手法について示す．4 章では反復 1 次元 d-Spline 探索を用いてテスト関数と実アプリケーションに適用し評価を行う．最後に 5 章でまとめと今後の課題について述べる．

2. これまでの研究と課題

2.1 ソフトウェア自動チューニング

ソフトウェア自動チューニングを行うタイミングはインストール時と実行時の 2 カ所ある．自動チューニングのタイミングを図 1 に示す．図はユーザ・プログラム中の自動チューニングの対象を数値計算ライブラリとした場合を想定する．

(1) インストール時自動チューニング
アプリケーションを利用する前にあらかじめ（アプリケーションのインストール時，実行時前など）アプリ

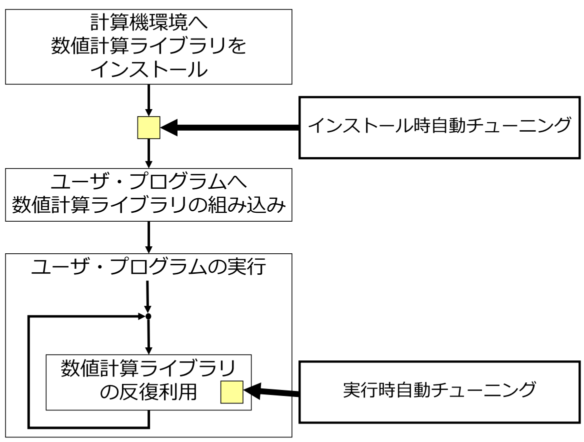

図 1　自動チューニングのタイミング
Fig. 1　Automatic tuning timing.

---

[1] 工学院大学
Kogakuin University, Shinjuku, Tokyo 163–8677, Japan
[a] fujii@cc.kogakuin.ac.jp
[b] teru@cc.kogakuin.ac.jp

ケーションの最適値を選択する．ユーザ・プログラムが対象とする問題のサイズが特定できないため，複数の問題のサイズに対して数値計算ライブラリを実行させる必要がある．そのため，最適化に膨大な時間がかかる．

(2) 実行時自動チューニング
アプリケーションの実行時にアプリケーションの最適値を選択する．実行時自動チューニングは，ユーザ・プログラムの反復1回ごとに性能評価を行う．実行時自動チューニングでは実行時にすでに問題のサイズが決定されていると考えられるため，インストール時自動チューニングと比べて探索空間は小さい．また，疎行列の形状も実行時に決まるので実行時自動チューニングが必須となる．しかし，自動チューニングに要する時間はアプリケーションに推定コストとして追加される．したがって，時間のかかる自動チューニングはかえってターゲットプログラムの性能を落とすことになるため，自動チューニングのための計算コストが少ないことが必要なる．

PHiPAC [11], ATLAS [12], FFTW [13] などは，インストール時自動チューニングを採用しており，Active Harmony, Autopilot などは，実行時自動チューニングを採用している．さらに，FIBER [14] は両方のタイミングで自動チューニングを行う．

本論文では，実行時自動チューニングについて行う．実際の推定では実時間のブレなどに影響を受けると考えられる [15].

性能パラメタのとりうる値の中から最適値を見つける探索アルゴリズムとして，すべての性能パラメタを調べる全探索や，少数の性能パラメタを標本点とし，標本点による実測データから多項式近似を用いて調べる標本点推定がある．全探索はすべての性能パラメタを調べるため，自動チューニングの時間が長くなる．標本点推定は実測するパラメタ数が全探索より少ないため，自動チューニングの時間が抑えられる．しかし，多項式による関数近似の性能パラメタの組合せごとの数値計算ライブラリの実行時間で構成される関数形は，滑らかさや凸性が保証されないため，よりよい近似を求めることが難しい．実測する標本点が事前に固定されることもあり，近似の確からしさには課題がある．また，標本点を選択・追加するたびに近似関数を再計算する必要がある．

そこで，我々は標本点推定において，実測データの動きに柔軟に追随する近似関数を導入し，必要に応じて標本点を追加する推定手法として，標本点逐次追加型性能パラメタ推定法を提案している．近似関数には，微分連続性を用いない d-Spline と呼ぶスプライン系の関数を用いる．最適な性能パラメタを推定する確率が高く，最適値を選択できなかった場合においても最適値に近い性能パラメタを推定できる．

2.2　標本点逐次追加型性能パラメタ推定法（IPPE 法）

この節では，d-Spline を使用した IPPE 法について説明する．2.2.1 項は 1 次元パラメタ空間における IPPE 法の概要について示し，2.2.2 項はこれを多次元パラメタ空間に拡張した方法について示す．

2.2.1　1 次元推定

IPPE 法は，いくつかの標本点から最適な性能パラメタを推定する．この方法では，すべての性能パラメタの組合せを実測しない．推定はいくつかの標本点から推定を開始し，最適な性能パラメタが安定するまで点を追加する．標本点とその性能値が追加されるたびに，パラメタ空間上の近似関数が更新され，この更新された関数を使用して最適なパラメタを推定する．

IPPE 法の次の標本点を選択する基準は 2 つある．推定した最適値がまだ選択されていない場合，そのパラメタを次の標本点として選択する．そうでなければ，近似関数内で変化の大きい点として，前後の勾配の差が最大の性能パラメタを選択する．これにより，関数の形状が実質的に変更される．

また，終了条件として，同じ性能パラメタが指定回数連続で最適な性能パラメタとして選択され続けたときとする．この終了条件を強める（回数を多く設定する）と多くの標本点が必要だが，良好な性能パラメタが見つかる可能性が高い．一方，終了条件を弱めると少数の標本点で推定を終了するが，最適でない性能パラメタが推定される可能性が増える．

IPPE 法は新しい標本点の性能値を追加するたびに近似関数を更新する必要があるため，近似関数は次の特性を持たなければならない：

(a) 計算に時間がかからない
(b) データの動きに柔軟に追随する良好な補間

これらの条件を満たす近似関数として d-Spline を使用する．d-Spline $\boldsymbol{f}$ は n 個の離散点 x_j 上の値 $f_j = \boldsymbol{f}(x_j)$, $1 \leq j \leq n$ で表される．ここで，t は転置を表す．

$$\boldsymbol{f} = (f_1, f_2, \cdots, f_j, \cdots, f_n)^t \tag{1}$$

d-Spline のモデルを簡略化するために，離散点 x_j の値は j に対して単調に増加すると仮定し，その区間 $|x_j - x_{j-1}|$ は $1 \leq j \leq n-1$ とする．IPPE 法の実行中は x_j と n の両方が固定される．実測データは y_i, $1 \leq i \leq N$ として表す．

$$\boldsymbol{y} = (y_1, y_2, \cdots, y_i, \cdots, y_N)^t \tag{2}$$

後述するように，$\boldsymbol{f}$ は 2 階差分を定義しているので，$\boldsymbol{f}$ の連続する 4 つの要素で 3 次式を構成する．実測データから滑らかな $\boldsymbol{f}$ を推定するために測定値 y_i の 2 つの点の間には少なくとも 2 つの点 f_j が必要である．したがって，n は N

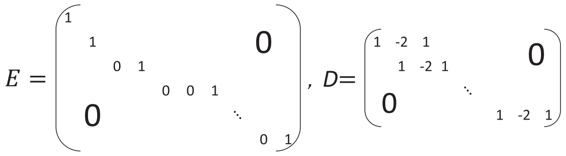

図 2　行列 E, D の構造
Fig. 2　Structure of matrix E (left) and matrix D (right).

よりも大きく設定する．また，任意の実測データ y_i に対して，ある節点 f_j が対応している．これは y_i の d-Spline による推定値が f_j であることを意味している．この対応関係を行列 E を用いて表す．ある y_i が標本点 f_j における測定値である場合 $E_{i,j} = 1$ となる．そうでなければ，$E_{i,j} = 0$ となる．$\boldsymbol{y}$ と $\boldsymbol{f}$ との間の距離は，$||\boldsymbol{y} - E\boldsymbol{f}||^2$ で表せる．

d-Spline $\boldsymbol{f}$ の滑らかさは次の二階差分で計測する．

$$|f_{j-1} - 2f_j + f_{j+1}|, (2 \le j \le n-1) \tag{3}$$

d-Spline $\boldsymbol{f}$ の滑らかさは，$||D\boldsymbol{f}||^2$ で表せる．**図 2** に $N \times n$ の行列 E と $(n-2) \times n$ の行列 D の構造を示す．

d-Spline $\boldsymbol{f}$ は次の評価関数で選択する．

$$min(||\boldsymbol{y} - E\boldsymbol{f}||^2 + \alpha^2||D\boldsymbol{f}||^2) \tag{4}$$

第 1 項は実測データ $\boldsymbol{y}$ と d-Spline $\boldsymbol{f}$ との間の距離を表し，第 2 項は d-Spline $\boldsymbol{f}$ の滑らかさを表す．α は第 1 項と第 2 項のバランスを調整するスカラー値であり，小さく設定するほど近似関数は実測データに追随し，大きくするほど直線となる．また，式 (4) は以下のように最小二乗問題に帰着する．

$$min||\boldsymbol{b} - Z_\alpha \boldsymbol{f}||^2 \tag{5}$$

Z_α と $\boldsymbol{b}$ は以下のように表すことができる．fill-in を抑えるために，E^t を E と $\boldsymbol{y}$ の左側から掛けるように変形を行っている．

$$Z_\alpha = \begin{pmatrix} E^tE \\ \alpha D \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} E^t\boldsymbol{y} \\ 0 \end{pmatrix} \tag{6}$$

この最小二乗問題 (5) を解く手法として，Z を直交行列 Q と上三角行列 R の積に分解する QR 分解を適用する．QR 分解には Givens 変換法を用いる．よって，$||\boldsymbol{b} - Z_\alpha\boldsymbol{f}||^2$ は $||Q^t\boldsymbol{b} - R\boldsymbol{f}||^2$ と変換することができるため，d-Spline $\boldsymbol{f}$ は後退代入により求めることができる．

このとき，標本点を追加するたびに Z_α に戻って Givens 変換法を施す必要はない．$k+1$ 個目の標本点を追加する場合は，k 個の標本点から得られた行列 R に 1 行追加して，この 1 行のみ Givens 変換法を施せばよい．y_i が f_j に追加されると，R' と $\boldsymbol{b'}$ は次のように構成される．

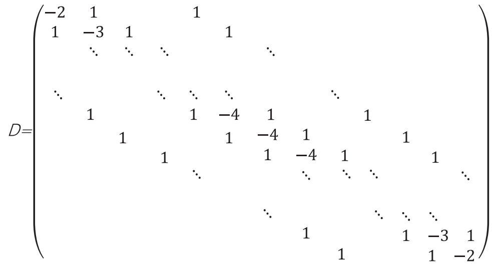

図 3　2 次元パラメタの行列 D の構造
Fig. 3　Structure of matrix D for 2D parameter.

$$R' = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & r_{j,j} & r_{j,j+1} & r_{j,j+2} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & r_{n-2,n} \\ & & & & & \ddots & r_{n-1,n} \\ & & & & & & r_{n,n} \\ 0 & \dots & 1 & & \dots & & 0 \end{pmatrix},$$

$$\boldsymbol{b'} = \begin{pmatrix} b_1 \\ \vdots \\ b_j \\ \vdots \\ \vdots \\ \vdots \\ b_n \\ y_i \end{pmatrix}$$

2.2.2　2 次元・3 次元化

2 つの性能パラメタの推定は，d-Spline の 2 次元化により実現する．性能パラメタ空間における実測データ $\boldsymbol{y}$ は，2 次元離散格子点 y_{i_1,i_2}（$1 < i_1 < N_1$，$1 < i_2 < N_2$）で表せられ，d-Spline $\boldsymbol{f}$ は離散点 x_{j_1,j_2}（$1 \le j_1 \le n_1$，$1 \le j_2 \le n_2$）で表される．

$$\boldsymbol{f} = (f_{1,1}, f_{2,1}, \cdots, f_{j_1,j_2}, \cdots, f_{n_1,n_2})^t \tag{7}$$

2 次元の d-Spline $\boldsymbol{f}$ の滑らかさは，次のように表される．

$$|f_{j_1-1,j_2} + f_{j_1,j_2-1} - 4f_{j_1,j_2} + f_{j_1,j_2+1} + f_{j_1+1,j_2}|, \quad (2 \le j_1 \le n_1 - 1, 2 \le j_2 \le n_2 - 1) \tag{8}$$

図 3 は，2 次元性能パラメタに対する $(n_1 \times n_2 - 4) \times (n_1 \times n_2)$ の行列 D を示す．2 次元性能パラメタ空間では，D の帯域幅は 1 次元 d-Spline $\boldsymbol{f}$ の帯域幅よりも大きい．行列 D の構造は，サイズが $n_2 \times n_2$ のブロック行列が行と列方向そ

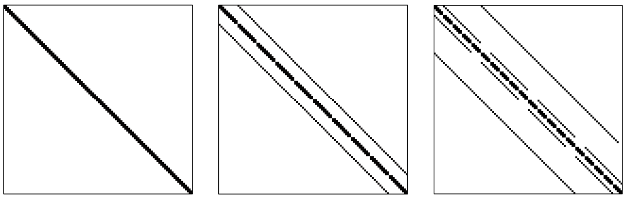

図 4　行列 D の形状　1 次元 (左)，2 次元（中央），3 次元（右）
Fig. 4　Shape of matrix D for 1D d-Spline (left), 2D d-Spline (center), and 3D d-Spline (right).

れぞれに n_1 個並んだ形となる．ただし，1 番上と下のブロックの行列は $(n_2-2)\times n_2$ である．これは，2 次元化した d-Spline のうち，四隅の点については滑らかさを定義していないためである．

同様に，性能パラメタが 3 つの推定は，3 次元 d-Spline を適用することによって実現する．ここで，d-Spline $\boldsymbol{f}$ は，離散点 x_{j_1,j_2,j_3}（$1\le j_1\le n_1$，$1\le j_2\le n_2$，$1\le j_3\le n_3$）で表す．3 次元 d-Spline $\boldsymbol{f}$ の滑らかさは，次のように表される．

$$|f_{j_1-1,j_2,j_3}+f_{j_1,j_2-1,j_3}+f_{j_1,j_2,j_3-1}-6f_{j_1,j_2,j_3} + f_{j_1,j_2,j_3+1}+f_{j_1,j_2+1,j_3}+f_{j_1+1,j_2,j_3}|, \quad (2\le j_1\le n_1-1, 2\le j_2\le n_2-1, 2\le j_3\le n_3-1) \tag{9}$$

性能パラメタが増加するにつれて，d-Spline $\boldsymbol{f}$ の滑らかさの計算が複雑になる．

2.3　d-Spline の計算量

滑らかさを表す方程式は，性能パラメタが増加するにつれてより複雑になる．k 次元の性能パラメタ空間の二階差分は，$2k+1$ の近傍点の値を用いて計算することができる．したがって，D も 1 行につき $2k+1$ 個の非ゼロ要素からなる．

性能パラメタが 1 次元から 3 次元までの行列 D の形状を図 4 に示す．IPPE 法では，性能パラメタが増加すると，d-Spline の滑らかさを表す行列 D の形状がより複雑になる．そのため，多次元性能パラメタ推定の計算コストが高くなる．たとえば，2 次元性能パラメタは，図 4 の中央の行列に QR 分解を実行する．この行列の帯域幅は $2n_1$ である．多次元性能パラメタ空間推定と後述する反復 1 次元 d-Spline 探索の計算量を**表 1** に示す．ここで，n_k（$k=1,2,3,4$）は性能パラメタのとりうる数を表す．反復 1 次元 d-Spline 探索の n は次元数によって変化しない．

2.4　関連研究

性能パラメタ推定方法として，Surrogate Assisted Tuning（SAT）法が提案されている [16]．IPPE 法は d-Spline 近似関数を性能モデルとして用いる．近似関数に対する仮定が少ないため，様々なデータに柔軟に対応できる．SAT 法では空間的相関を用いて構成される Kriging モデル（ガウス確率過程回帰）を性能モデルに用いる．Kriging モデルは実測データの重み付き線形和で表現される．各性能パラメタ間の距離を測るカーネル関数を用いることで実測データとの重みを計算することができる．また，新たに標本点を選択する基準として，SAT 法では，Efficient Global Search（EGO）[17] に基いて，実測されていない点の期待値を計算する．現在の最適パラメタから性能を改善する期待値を計算することで局所探索と大域探索のバランスをとる．SAT 法の機構はこの手法に基づいており，EGO の自動チューニングへの適用であるといえる．SAT 法では，(1) 性能モデルのパラメタの最尤推定と (2) 各候補点との暫定の最適パラメタより高い性能を示す期待値を計算する必要がある．(1) は，新たに標本点を追加するたびに発生する．(2) は，(1) で生成と更新された性能モデルのパラメタをもとに次に実測する性能パラメタを決定するために必要となる．この方法は，性能パラメタ空間全体での推定を行うため，IPPE 法と同様に性能パラメタが増加すると計算量も増加する．

また，正規分布に基づく自動チューニングのための数理ソフトウェアとして ATMathCoreLib が提案されている [18], [19], [20]．ベイズ統計による性能推定を行う．特徴は，実際の実測のばらつき（撹乱効果）に着目し，これをモデル化することで実測の擾乱を最小化する．したがって，実測する各標本点の統計情報にのみ着目している．

表 1　d-Spline の計算量
Table 1　Computational complexity for d-Spline function.

次元数	多次元性能パラメタ推定	反復 1 次元 d-Spline 探索
1	$O(n_1)$	$O(n)$
2	$O({n_2}^3\times n_1)$	$O(n)$
3	$O({n_2}^3\times {n_3}^3\times n_1)$	$O(n)$
4	$O({n_2}^3\times {n_3}^3\times {n_4}^3\times n_1)$	$O(n)$

3.　提案手法

2.3 節で説明したように，滑らかさを表す行列 D は，性能パラメタの数が増えるにつれて複雑になるため，自動チューニングの推定コストが増加する．それに対して，計算量が $O(n)$ である 1 次元パラメタ空間における軽量 IPPE 法を繰り返し用いた性能パラメタ推定を提案する．

繰り返し探索する手法として，一般的に連続関数の空間での関数の最小値を探索する最急降下法がある．この方法では，関数値が最も減少する方向を関数の微分によって選択する．つまり，最も傾きが大きい方向を選び，決定した方向の直線を探索する．方向の決定と探索を繰り返し行い最適値を推定する．一方，本研究で必要とする推定値は，離散点上からなる性能パラメタ空間上で行う．離散点からなる空間の推定には最急降下法のように関数の微分は計算

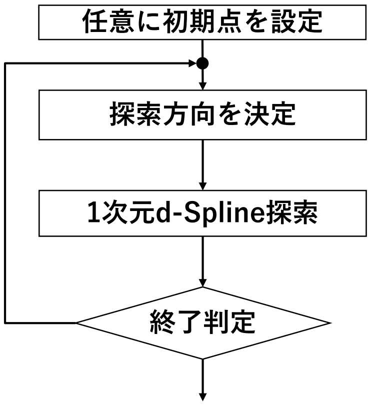

図 5　推定のアルゴリズム
Fig. 5　Estimation algorithm.

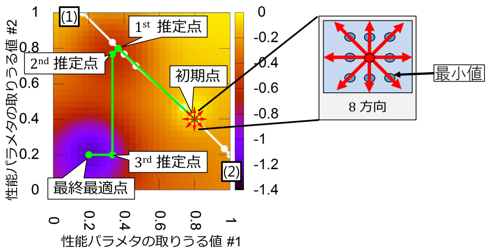

図 6　Franke 関数の反復 1 次元 d-Spline 探索の推定パス
Fig. 6　Path of iteration of 1D d-Spline search for Franke function.

することができない．

そこで，我々は標本点の周りの複数の離散点を実測し，その中で最小値を選択して，その点ともとの標本点の 2 点含んだ直線を決定する．その直線上に 1 次元 d-Spline 探索を適応する．

2 次元性能パラメタ空間の場合，x，y 軸と軸と 45° の斜め方向である 2 つの合計 4 方向を考える．探索する方向に斜め方向を加えた．理由としては，性能パラメタ間に相関がある可能性が考えられるためである．ここで，個々の性能パラメタを p_k と表すとする．たとえば，円の方程式 ${p_1}^2+{p_2}^2$ のように 2 つのパラメタ間に相関がなければ，p_1，p_2 ごとに独立に最適値を見つけることができる．一方，2 つの性能パラメタが相互依存している場合，p_1，p_2 は独立に最適値を見つけることができない．したがって，p_1，p_2 の線形結合した式（一般に斜めの直線）が必要となる．

推定の流れを**図 5** に示す．推定の手順について 2 次元性能パラメタ推定の 1 例を用いて**図 6** に示す．図 6 は，谷が 2 つ，山が 1 つの Franke 関数 [21] の関数値の等高線図である．Franke 関数の値を性能パラメタの値として，それぞれの性能パラメタのとりうる値は 31 個の離散値と仮定する．最終目標は，Franke 関数上で一番低い値の性能パラメタの組合せを推定することである．はじめに，任意に初期点を設定する．次に，探索方向決定のために右図の青い 8 つの点を実測し，赤い矢印の 4 方向の中から探索

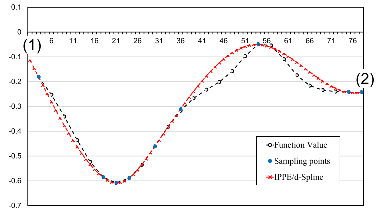

図 7　d-Spline の形状
Fig. 7　Shape of d-Spline.

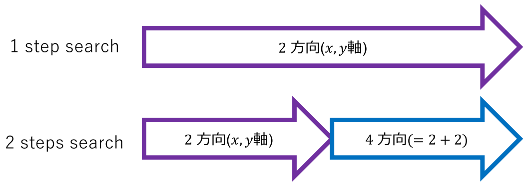

図 8　2 次元性能パラメタの推定手法
Fig. 8　Estimation method of two dimensional performance parameter.

する直線を選択する．選択した直線が等高線図中の白い線（(1)–(2)）となる．選択した直線に対して 1 次元 d-Spline 探索を用いて推定を行う．1 次元 d-Spline 探索を適応したときの d-Spline の形状を**図 7** に示す．図 7 の (1)，(2) は，図 6 の (1)，(2) に対応し，横軸は性能パラメタの値，縦軸は関数値を示す．それぞれ，黒い線は実際の Franke 関数の値，青色の点は，1 次元 d-スプライン探索の標本点，赤い線は最終推定結果の d-Spline 関数の形状を示す．図において直線内で最適値と推定した点を緑の点として，緑の点を結ぶ線が最適な性能パラメタの推移となる．また，推定は 1 次元 d-Spline 探索によって異なる 3 方向で連続で最適と推定された場合に終了する．推定結果は，最適値を正しく推定した．

次に，探索方向の決定について考察する．探索方向を決定するために必要な標本点の数は，性能パラメタの数によって決まる．

- 2 性能パラメタ：8（$3 \times 3 - 1$）点
- 3 性能パラメタ：26（$3 \times 3 \times 3 - 1$）点
- 4 性能パラメタ：80（$3 \times 3 \times 3 \times 3 - 1$）点

次元が高くなると，決定するために必要な標本点の数が指数的に増加する．

一方，1 次元 d-Spline 探索を用いた IPPE 法では標本点は最大でも性能パラメタのとりうる数であり，性能パラメタ空間の次元数によらず，一定である．

方向決定に用いる標本点数を削減するために探索を段階的に行う方法について考える．ここでは，新たに multi-steps search を提案する．比較のためにもとの方法を 1 step search とする．2 次元の場合の推定手法について**図 8** に示す．1 step search は，推定の最初から探索方向が 4 方向（2 つの軸方向と 2 つの斜め方向）に固定した探索を行う．一方，提案する 2 steps search は，最初に 2 方向（軸方向）

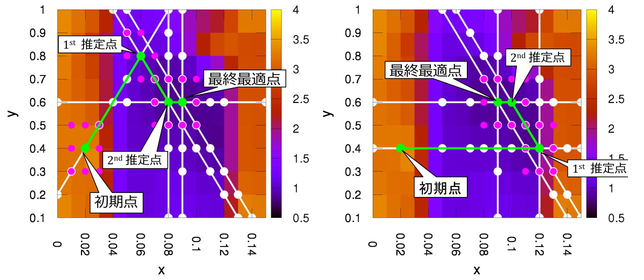

図 9　推定の探索経路（左：1 step search，右：2 steps search）
Fig. 9　Path of reiteration (Left: 1 step search, Right: 2steps search).

のみを探索し，終了条件が満たされた後に，軸方向および斜め方向で行う探索を行う．

図 9 に同じ初期点から推定を行ったときの 2 つの方式それぞれの探索経路を示す．それぞれの性能パラメタのとりうる値は横軸が 16 個，縦軸が 10 個である．赤い点は，探索方向の決定に用いた標本点，白い点は 1 次元 d-Spline の推定に用いた標本点を表す．特に赤い点は 1 step search で 23 点だったが，2 steps search では 17 点に減少した．2 steps search は，最初に探索方向を制限することによって探索方向の決定に必要な標本点を減らすことができる．性能パラメタの数が増加するにつれて，方向決定に用いる点数は増加していくため，方向決定に用いる点数の削減の効果はより大きくなる．

4.　数値実験

本項では，反復 1 次元 d-Spline 探索を用いて多次元性能パラメタ推定の数値実験の結果と考察を行う．4.1 節では，性能パラメタが 3 つの場合について従来手法である多次元 d-Spline と反復 1 次元 d-Spline 探索について比較を行う．4.2.1 項では，性能パラメタが 4 つの場合について提案手法の特性評価のためテスト関数に適用し，関数形状が単純な問題と局所解が複数存在する問題について網羅的に推定結果について評価する．4.2.2 項では，4 次元性能パラメタ推定を実問題に適用し，高次元性能パラメタ推定での提案手法の有用性を評価する．

4.1　性能パラメタが 3 つの場合

3 次元性能パラメタすなわち性能パラメタが 3 つの場合について推定を行う．探索方向決定のために実測する点数は $3^3 - 1 = 26$ 点となる．3 次元性能パラメタ推定の探索方向は 13（$= \frac{26}{2}$）方向となる．3 次元性能パラメタ推定について 1 次元 d-Spline を繰返して推定する 2 つの手法の結果を**図 10** に示す．1 step search は，最初から最後まで探索を 13 方向で行う．2 steps saerch は，はじめに軸方向である 3 方向の探索を行い，次に 10 方向を加えた 13 方向の 2 段階で探索を行う．反復 1 次元 d-Spline 探索との比較対象として，従来手法である 3 次元 d-Spline 探索も計測する．

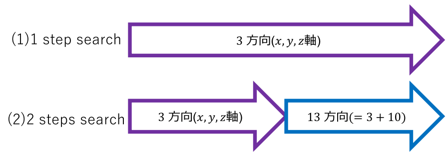

図 10　3 次元性能パラメタの推定手法
Fig. 10　Estimation method of three-dimensional performance parameter.

表 2　使用した性能パラメタの条件（3 次元）
Table 2　Conditions of performance parameters used (Three-dimensional).

種類	*Para*.1	*Para*.2	*Para*.3
間隔	0.01	0.1	0.1
パターン数	16	10	9
性能パラメタの組み合わせ総数		1,440 (= 16 × 10 × 9)	

対象として，ユーザ・プログラムの BiCGStab 法の前処理として代数的マルチグリッド法（AMG 法）[22] を適用したときの多次元性能パラメタ推定を行う．我々は，AMG 法のプログラムを数値計算ライブラリ（AMG ライブラリ）として提供しており [23], [24]，ユーザ・プログラムにこの AMG ライブラリを適用することにより，効率よく AMG 法を適用することができる．AMG 法は性能パラメタの種類が多く，自動チューニングの好適な対象となる [24], [25]．対象とする問題は，3 次元立方体の Poisson 方程式とし，拡散係数は等方性で，メッシュサイズは 120 × 120 × 120 で行った．問題は 12 ノードの 192 プロセスに分解された MPI モデルを使用した．ここでは，実験環境として東京大学に設置された富士通 FX10 スーパコンピュータシステム [26] を使用した．推定する AMG ライブラリの性能パラメタは，強連結成分の判定のための閾値である *strong connection threshold*（*Para*.1）とレベルが粗くなったときの *strong connection threshold* の削減率である θ *reduction rate*（*Para*.2）と定数補間を滑らかにするための減速ヤコビ法の減速係数 *dump jacobi coefficient*（*Para*.3）である．**表 2** に性能パラメタの条件を示す．

性能パラメタのとりうるすべての組合せを初期点として，1,440 回の試行を行った．反復 1 次元 d-Spline 探索（(1) 1 step search，(2) 2 steps search），(3) 3 次元 d-Spline 探索の比較結果を**表 3** に示す．比較項目を下記に説明する．

- 標本点数：「方向決定」と「1 次元探索」に用いた標本点の合計数
- 方向決定：探索方向を決定するために，実測した標本点の数
- 1 次元探索：方向決定で求めた直線上を探索するときに追加した標本点の数
- 最適値とのずれ：推定した最適値と，真の最適値とのずれ

表 3　3 つの性能パラメタ推定結果

Table 3　Three-dimensional performance parameter estimation results.

手法	反復 1 次元 d-Spline 探索 (1) 1 step search	(2) 2 steps saerch	(3) 3 次元 d-Spline 探索
標本点数	126.1	104.8	94
方向決定	89.1	46.0	
1 次元探索	37.0	58.8	
最適値とのずれ [%]	0.0003	0.0	7.5
最適値とのずれ 1%以内（最適値を推定した個数）	1,439 (1,439)	1,440 (1,440)	—
推定時間 [s]	0.59×10^{-3}	0.71×10^{-3}	3.6
実行時間（AMG）[s]	204.7	150.9	227.9

- 最適値とのずれ 1%以内：ずれが 1%未満の最適値を推定した探索の試行回数
- 推定時間：探索に要した時間
- 実行時間（AMG）：探索で標本点として実測した AMG 法の実行時間

なお，数字は，「最適値とのずれ 1%以内」以外は 1,440 回の試行の平均値を示す．反復 1 次元 d-Spline 探索の (1) 1 step search と (2) 2 steps search の比較を行う．

「標本点数」は，(1) に対して (2) は 16.9%（$= 1 - \frac{104.8}{126.1}$）削減した．これは，探索を multi-steps（ここでは，2 steps）にしたことにより，方向決定のための標本点数が大幅に削減されたためである．一方，全体の標本点数は少なくなったが，1 次元探索に要する標本数は (2) の方が多く，これは，(2) の方が 1 次元探索による大域的な探索に多くの標本点が用いられていることを示している．上記の理由もあり，(2) では「最適値とのずれ」が 0%，つまり，すべての初期点から最適値を推定することができた．「推定時間」は「実行時間」に比べて，(1)，(2) ともに 10^5 のオーダで小さくなっており，推定時間はほとんど無視できる大きさであることが分かる．(1)，(2) は (3) に比べて大幅に改善した．これは，(3) の探索の計算量が $O({n_2}^3 \times {n_3}^3 \times n_1)$ に対して，(1)，(2) の探索の計算量が $O(n)$ を繰り返して用いるためである．(3) は性能パラメタの種類が増えるほど計算量が指数的に増えるため，さらに，(1)，(2) の優位性は高くなる．

4.2　性能パラメタが 4 つの場合

4.1 節で，性能パラメタが 3 つの場合，従来の多次元 d-Spline 探索に対して，2 steps 反復 1 次元 d-Spline 探索が有効な推定を行うことができるとともに，探索時間がとても小さく，さらなる高次元性能パラメタ推定での可能性が分かった．4.2 節では，性能パラメタが 4 つの場合，つまり，4 次元性能パラメタ推定における multi-steps 反復 1 次元 d-Spline 探索について評価を行う．**図 11** に，最初から最後まで 40 方向行う 1 step search と推定を 4 つの段階に分けた 4 steps search の方法について示す．4 steps search の方向の切り替えのタイミングとしては，以下の 2 つがある．

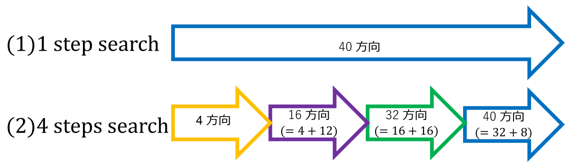

図 11　4 次元性能パラメタの推定手法

Fig. 11　Estimation method of four-dimensional performance parameter.

(1)「40 方向」以外の場合
　標本点 P において方向探索を行う場合，調べた周囲の点がすべて標本点 P より大きな値の場合，探索方向の追加を行う．

(2)「最後の 40 方向」の場合
　調べた周囲の点の中で最小の値の点の方向を選択し，1 次元探索を行い，再度，最適値として，標本点 P が選択されることが 3 回連続して行われる段階で終了する．

4.2.1　テスト関数への適用

反復 1 次元 d-Spline 探索手法の特性評価のために，テスト関数による人工データを用いた最適化実験を行う．テスト関数は計算機モデルを含む実験の設計と分析に対する新しい方式を評価するための関数とデータセットの提供を行っている文献 [27] から問題を選択し，評価を行った．16 個のテスト関数に対して適用した．この 16 個のテスト関数の名称，特性，および，離散化した性能パラメタのとりうる値の数などを付録表 A·1～A·6 に示した．テスト関数の変数が 4 つ未満の場合には，独立な変数（${p_k}^2$）を加えて性能パラメタが 4 つになるようにした．テスト関数の形状は，単峰性と多峰性の 2 つがある．単峰性とは，谷が 1 つだけの単純な関数形状をしている問題である．多峰性とは，谷が複数存在し，大域的な最適解を求めることが困難な問題である．16 個のテスト関数のうち，Sphere，Booth，Matyas（付録表 A·1）が単峰性であり，残りの 13 個が多峰性である．それぞれの関数の 2 変数における 3 次元グラ

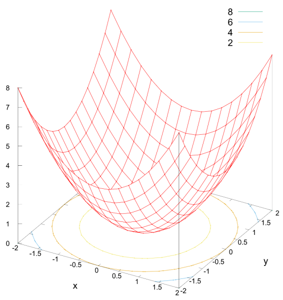

図 12　単峰性関数の形状（Sphere 関数）
Fig. 12　Shape of unimodal function (Sphere function).

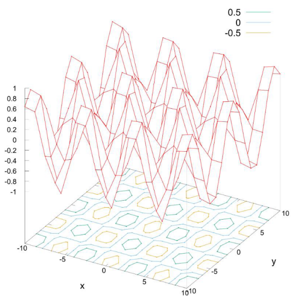

図 13　多峰性関数の形状（Griewank 関数）
Fig. 13　Shape of multimodal function (Griewank function).

フの例を**図 12**，**図 13** に示す．

なお，それぞれの関数ごとに，その関数の形状を表すように離散化を行い，性能パラメタとしてのとりうる値を決めた．そのため，関数ごとに，性能パラメタの組合せの数が異なる（付録表 A·1～A·6 を参照のこと）．

提案手法の有効性を示すために，局所探索手法を定義し，評価を行う．ここで用いた局所探索手法の手順を示す．

(1) 初期点を 1 点決める．
(2) その点を基準点として周りの点を実測する．
(3) 実測した中での最小値が基準点より小さいときは，その最小値を次の基準点とする．
(4) (2) から (3) を周りの点がすべて基準点より大きくなるまで繰り返す．
(5) 最後の基準点を最適値とする．

4.1 節と同様に，性能パラメタ空間上のすべての性能パラメタの組合せの点を初期値として，実測を行う．実測結果として，テスト関数ごとに最適値および最適値に近い値を推定した割合を**図 14** に示す．横軸はテスト関数の種類，縦軸は初期点から最適値を推定した割合を表す．青色の積立グラフが局所探索，緑色の積立グラフが 1 step search，赤色の積立グラフが 4 steps search を表す．それぞれの濃い色の部分が最適値を推定した割合を示す．薄い色の部分が下記の式に当てはまる値の割合を表す．

$$EOP < 0.1(f_{Max} - f_{min}) + f_{min} \tag{10}$$

EOP は推定した最適値，f_{Max} は関数の最大値，f_{min} は関数の最小値を表す．式 (10) は，関数値の最大値と最小値の差を取り，その差の 10%以内の値を推定した割合を表している．単純な関数形状をしている単峰性関数の 3 つの問題では，すべての初期点から最適値を推定した．

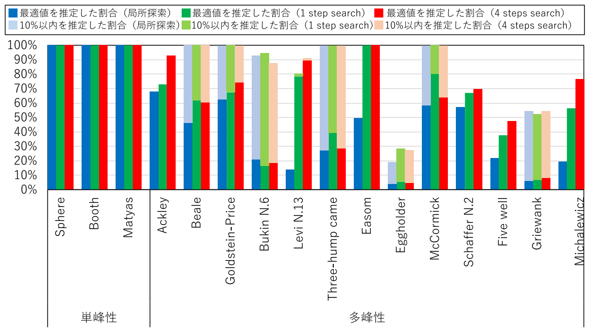

図 14　最適な値を推定した割合
Fig. 14　Percentage at which the optimum value was estimated.

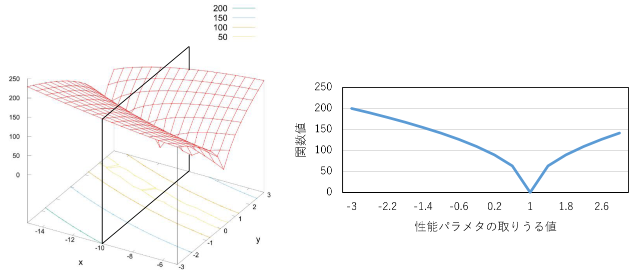

図 15　Bukin N.6 関数の関数形状
Fig. 15　Function shape of Bukin N.6 function.

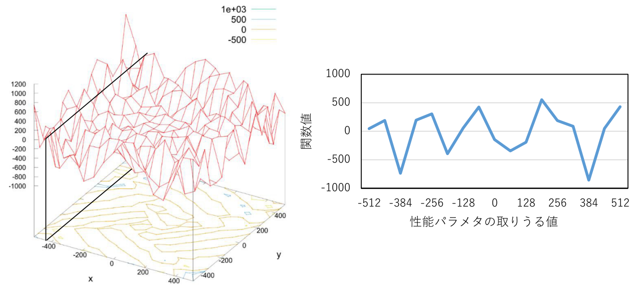

図 16　Eggholder 関数の関数形状
Fig. 16　Function shape of Eggholder function.

多峰性の問題において，10%以内を推定した割合が大きく増加した関数（Bukin N.6 関数）とほとんど増加しない関数（Eggholder 関数）を比較する．**図 15** と**図 16** に Bukin N.6 関数と Eggholder 関数の 3 次元グラフと 3 次元グラフ上の黒い線での x 軸を固定したときの関数の断面図を示す．Bukin N.6 関数は空間全体としては局所解が存在するが x 軸を固定したときの関数形状は単純な形状をしている．しかし，Eggholder 関数のように x 軸を固定したときの関数形状も局所解が複数存在し，最適値を推定することが困難な問題のためほとんど増加しない結果となった．

局所探索と比較すると，単峰性の問題では，どの手法でも最適値を得ることができた．多峰性の問題では，最適値からのずれの視点から局所探索より，反復 1 次元 d-Spline

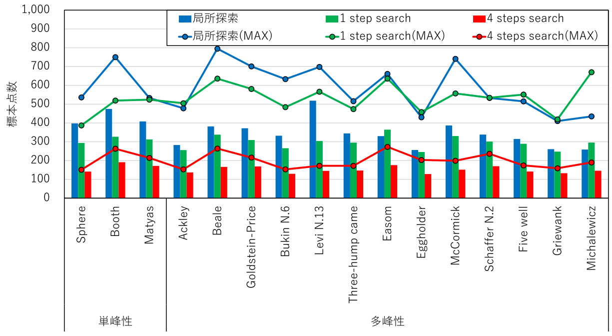

図 **17**　推定に使用した標本点数

Fig. 17　Sampling points used for estimation.

表 **4**　実験環境

Table 4　Experiment environment.

CPU	Intel Xeon E5-2695v4
周波数	2.1 [GHz]
CPU の数	2
コア数	18
メモリ	256 [GB]

表 **5**　推定する性能パラメタの条件

Table 5　Condition of estimated performance parameter.

種類	*strong con threshold* *Lv1*	*strong con threshold* *Lv2*	*dump jacobi coef* *Lv1*	*dump jacobi coef* *Lv2*
範囲	0.01 ～ 0.05	0.01 ～ 0.05	0.1 ～ 1.0	0.1 ～ 1.0
パターン数	16	16	16	16
性能パラメタの組み合わせ総数	65,536 (= 16 × 16 × 16 × 16)			

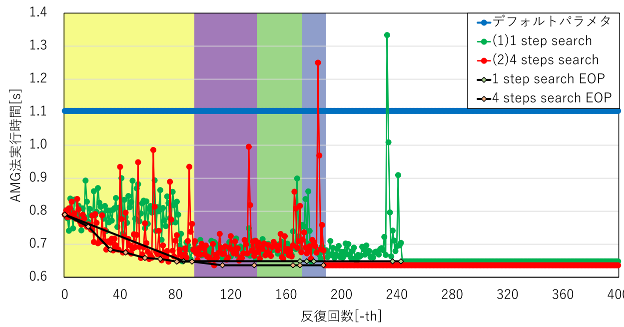

図 **18**　推定の最適値の推移

Fig. 18　Transition of estimated optimum value.

探索手法の方が有効であることが示された．これは，反復 1 次元 d-Spline 探索は大域的な探索ができているためと考えられる．

すべての性能パラメタの組合せに対して，推定に使用した標本点数を図 **17** に示す．横軸はテスト関数の種類を表し，縦軸はすべてのパラメタの組合せに対する標本点数を表す．折れ線グラフはそれぞれの手法における標本点数の最大値を表す．4 steps search の標本点数は，すべてのテスト関数に対して 1 step search と局所探索より削減した．また，すべての関数で実測した標本点数の割合は 0.3%を下回る結果となった．反復 1 次元 d-Spline 探索において単峰性の標本点数が多峰性よりも多い場合があるのは，多峰性の問題で早い段階で局所解を推定した後に推定点が変化しないで推定を終了したためである．それぞれのテスト関数の推定結果の詳細は付録 A.1 に示す．

4.2.2　AMG 法への適用

4.1 節では，AMG 法プログラムを題材に 3 つの性能パラメタ推定を行った．ここでは，対象のユーザ・プログラムとして CG 法を用いて評価を行う．数値計算ライブラリは，ユーザ・プログラムの CG 法の前処理として 4.1 節と同じ AMG ライブラリを用いて 4 つの性能パラメタ推定を行う．問題は，拡散係数が 10^{-5} から 10^5 と場所によって不連続に変化する $100 \times 100 \times 100$ の三次元 Poisson 方程式である．計算機環境は東京大学に設置されている Reedbush-U スーパコンピュータシステムを使用した [28]．1 ノードにおける仕様を表 **4** に示す．ここでは，2 ノードを使い，各ノードで 16 プロセス ×2 スレッドで実行した．推定する AMG ライブラリの性能パラメタは，強連結成分の判定のための閾値である *strong connection threshold*（*strong con threshold*）と定数補間を滑らかにするための減速ヤコビ法の減速係数 *dump jacobi coefficient*（*dump jacobi coef*）である．それぞれの性能パラメタはマルチグリッドのレベル 1 とレベル 2 で異なる値が設定できるようにした．表 **5** に性能パラメタの条件を示す．

性能パラメタを変化させたときの AMG ライブラリの実行時間の最大値は 1.83 [s]，最小値は 0.64 [s]，平均値は 0.92 [s]，分散は 0.02 となっている．最適な性能パラメタの組合せは *strong con threshold Lv1* = 0.031，*Lv2* = 0.021，*dump jacobi coef Lv1* = 0.94，*Lv2* = 0.94 となる．

ユーザ・プログラムの実行時の反復ごとの性能パラメタ推定の推移をみるために，ある 1 つの初期値からはじめて 400 回 AMG 法プログラムを反復したときの結果を図 **18** に示す．横軸が反復回数，縦軸が AMG 法プログラムの実行時間を表す．赤と緑の折れ線グラフがそれぞれの (1) 1 step search および (2) 4 steps search にて，それぞれ探索中の性能パラメタの組合せで実測した AMG ライブラリの実行時間を表す．黒はそれぞれの手法の推定最適値の推移を表す．黄，紫，緑，青のそれぞれの領域は，4 steps search における 4 段階のそれぞれのステップを表す．青の線は，文献 [24] で示されている AMG ライブラリのデフォルトの性能パラメタ設定を用いた場合である．AMG ライブラリのデフォルトの性能パラメタは固定されており，問題により設定されるのではないので，この問題に対してはあまり良い設定になっていない．このことからも問題ごとの実行時の自動チューニングが重要であることが分かる．

(1) は 400 反復までの実行時間の合計は 280.7 [s] となった．(2) は 400 反復までの実行時間の合計は 270.4 [s] となった．デフォルトパラメタの 400 反復までの合計時間は

表 6　4 つの性能パラメタ推定結果
Table 6　Four-dimensional performance parameter estimation results.

手法	(1) 1 step search	(2) 4 steps search
標本点数	333.53	185.77
方向決定	293.13	128.21
1 次元探索	38.40	57.56
1 次元探索回数	8.56	12.96
4 方向	—	4.93
16 方向	—	2.33
32 方向	—	1.54
40 方向	—	4.16
推定時間 [s]	0.24×10^{-3}	0.39×10^{-3}
実行時間 (AMG)[s]	2.56×10^{2}	1.37×10^{2}
最適値とのずれ [%]	1.60	1.62
最適値とのずれ 10%以内（最適値を推定した個数）	65,433（5,940）	65,529（5,023）
ランダム探索による最適値とのずれ [%]	3.92	5.73

442.6 [s] であった．デフォルトパラメタの実行時間の合計は (2) と比較して 38%削減した．従来使用されていた性能パラメタで実測するよりもソフトウェア自動チューニングを行うことで実行時間の短い性能パラメタの組合せを推定した．(1) の反復回数は，243 反復で最適値から 1.9%外れた値を推定した．(2) の反復回数は，187 反復で最適値を推定することができた．(2) は (1) より少ない反復で推定を終了した．(2) はパラメタの組合せの合計数の 0.3%の点数を実測して推定を終了した．ソフトウェア自動チューニングは，AMG ライブラリの性能パラメタチューニングとして有用であることを示した．

次に，全体の推定の傾向をみるために性能パラメタの組合せのすべての点を初期点として実測した．その平均値を**表 6** に示す．「探索回数」は，1 次元 d-Spline 探索をした回数を表す．「4，16，32，64 方向」は，4 steps search のそれぞれの段階で 1 次元 d-Spline 探索をした回数を表す．4.2.1 項では，反復 1 次元 d-Spline 探索の有効性を示すために，「局所探索」との比較を行った．ここでは，逆に，大域的探索である「ランダム探索」（モンテカルロ法）との比較を行う．比較を公平にするために，反復 1 次元 d-Spline 探索のそれぞれの方法（4 steps search，1 step search）と同じ数のランダム探索を行う．比較は，「最適値とのずれ (%)」で行う．

(2) はすべてのパラメタの組合せの 0.3%の実測のみで推定を終了した．(1) と比較しても (2) の推定に実測したパラメタの組合せを 45%削減した．それは，方向決定に 80 点を追加する (1) よりも多段階にすることにより方向決定に使う点数を大きく削減したためである．また，(1) より (2) の方が探索回数が多く，大域探索で広い範囲を探索できたことが分かる．推定時間は実行時間（AMG）と比較してとても小さい．反復 1 次元 d-Spline 探索は，とても推定コストの軽微な推定方法となっている．推定点と真の最適値とのずれはどちらの方法もほとんどかわらない結果となった．(2) の最適値とのずれが 10%以内の個数はすべての組合せの割合に対して 99.9%であった．ランダム探索との比較でも「最適値とのずれ」の視点で，反復 1 次元 d-Spline 探索のそれぞれの方法の方が有効であることを示すことができた．

5.　おわりに

本論文では，多次元性能パラメタ空間を推定する手法として，推定点周辺の離散点のデータを測定し，性能パラメタ空間から 1 次元空間を抽出し，その 1 次元空間で d-Spline 探索を行う反復 1 次元 d-Spline 探索方法を提案した．また，推定を多段階に行うことで標本点の削減を行った．提案手法を 4 次元性能パラメタに対して適用した．

実験の結果，4 次元の性能パラメタの 65,536 個の組合せのみのうち 0.3%の実測の組合せで推定を終了した．また，すべてのパラメタの組合せの点を初期点として推定した推定値と真の最適値とのずれが 10%以内の割合は 99.9%であった．

d-Spline の推定時間は無視できるほど小さくすることができた．これは，計算量が $O(n)$ のとても小さい 1 つの性能パラメタを推定する方法を繰返し用いたためである．

この反復 1 次元 d-Spline 探索を局所探索法と大域的探索法の 1 つであるランダム法（モンテカルロ法）と比較評価し，反復 1 次元 d-Spline 探索の方がより良い推定を行うことが分かった．

今後の課題としては以下の 4 点ある．1 点目は，推定の初期点の選択方法の導入である．本論文では，初期点を任意の点から推定を行った．しかし，初期点によって 1 次元 d-Spline 探索をする回数は変動する．そのため，適切な初期点から推定を始めることで 1 次元 d-Spline 探索の回数と方向決定の回数を減らし，標本点の増大を抑えることができる．2 点目は，本論文では実問題として AMG 法での評価例のみであたったため，他のアプリケーションでの評価も重要である．3 点目は，評価値が悪くなる箇所を緩和する手法の導入である．評価値が悪い値を実測するということは推定コストが大きくなり対象プログラムの性能を落とすことになる．そのため，評価値が悪くなる箇所を実測の対象から除外し，よい実測のみで推定する手法の導入が必要になる．4 点目は，関連研究で示した SAT 法など他の推定手法と組み合わせた推定方法の提案とより性能パラメタ推定の高精度化，高速化を追求していく．

謝辞　査読者の皆さまから有益なコメントをいただきました．名古屋大学情報基盤センターの片桐孝洋教授には ppOpen-AT についてご指導いただきました．ここに感謝の意を表します．本研究の一部は 16H02823 の助成を受け

て行われた．

参考文献

[1] 片桐孝洋：ソフトウェア自動チューニング−数値計算ソフトウェアへの適用とその可能性，慧文社 (2004).

[2] 黒田久泰，直野　健，岩下武史：特集：科学技術計算におけるソフトウェア自動チューニング，情報処理，Vol.50，情報処理学会 (2009).

[3] 片桐孝洋：特集：数値計算のための自動チューニング，応用数理，Vol.20，日本応用数理学会 (2010).

[4] Tanaka, T., Katagiri, T. and Yuda, T.: d-Spline Based Incremental Parameter Estimation in Automatic Performance Tuning, *Proc. 8th International Conference on Applied Parallel Computing: State of the Art in Scientific Computing*, LNCS, Vol.4699, pp.986–995, Springer (2007).

[5] Tanaka, T., Otsuka, R., Fujii, A., Katagiri, T. and Imamura, T.: Implementation of d-Spline-based Incremental Performance Parameter Estimation Method with ppOpen-AT, *Scientific Programming*, Vol.22, No.4, pp.299–307 (2014).

[6] Murata, R., Irie, J., Fujii, A., Tanaka, T. and Katagiri, T.: Enhancement of Incremental Performance Parameter Estimation on ppOpen-AT, *Proc. MCSoC2015*, pp.203–210 (2015).

[7] Katagiri, T., Ito, S. and Ohshima, S.: Early Experiences for Adaptation of Auto-tuning by ppOpen-AT to an Explicit Method, *Proc. MCSoC2013*, pp.153–158 (2013).

[8] Katagiri, T., Ohshima, S. and Matsumoto, M.: Auto-tuning of Computation Kernels from an FDM code with ppOpen-AT, *Proc. MCSoC2014*, pp.91–98 (2014).

[9] ppOpen-HPC Project, available from ⟨http://ppopenhpc.cc.u-tokyo.ac.jp/ppopenhpc/⟩ (accessed 2017-12-26)

[10] 入江　純，村田　陸，藤井昭宏，田中輝雄，片桐孝洋：自動チューニング基盤 ppOpen-AT 上での標本点逐次追加型性能パラメータ推定機能の実現，研究報告ハイパフォーマンスコンピューティング（HPC），Vol.148, No.27, pp.1–8 (2015).

[11] Bilmes, J., Asanovic, K., Chin, C.-W. and Demmel, J.: Optimizing matrix multiply using PHiPAC: A portable, high-perfomance, *ANSI C Coding Methodology*, *SC97*, pp.340–347 (1997).

[12] Whaley, R.C., Petitet, A. and Dongarra, J.J.: Automated Empirical Optimization of Software and the ATLAS Project, *Parallel Computing*, Vol.27, pp.3–35 (2001).

[13] Frigo, M. and Johnson, S.G.: FFTW: An Adaptive Software Architecture for the FFT, *ICASSP '98*, Vol.3, pp.1381–1384 (1998).

[14] Katagiri, T., Kise, K., Honda, H. and Yuba, T.: A General Framework for Auto-Tuning Software, *ISHPC-V*, pp.146–159 (2003).

[15] Fan, G., Mochizuki, M., Tanaka, T., Fujii, A. and Katagiri, T.: D-Spline Performance Tuning Method Flexibly Responsive to Execution Time Perturbation, *Proc. PPAM2017*, pp.1–10 (2017).

[16] Chen, J., Chen R.-B., Fujii, A., Suda, R. and Wang, W.: Timing Performance Surrogates in Auto-Tuning for Qualitative and Quantitative Factors, *SIAM Conference on Parallel Processing and Scientific Computing* (*PP14*) (2014).

[17] Jones, D.R., Schonlau M. and Welch, W.J.: Efficient Global Optimization of Expensive Black-box Functions, *Jounal of Global Optimization*, Vol.13, pp.455–492 (1998).

[18] Suda, R.: A Bayesian Method for Online Code Selection: Toward Efficient and Robust Methods of Automatic Tuning, *2nd International Workshop on Automatic Performance Tuning* (*iWAPT2007*), pp.23–31 (2007).

[19] 須田礼仁：オフライン自動チューニングの数理手法，研究報告ハイパフォーマンスコンピューティング（HPC），Vol.125, pp.1-9 (2010).

[20] 須田礼仁：自動チューニング数理基盤ライブラリ AT-MathCoreLib，研究報告ハイパフォーマンスコンピューティング（HPC），Vol.129, pp.1-12 (2011).

[21] Franke, R.: A critical comparison of some methods for interpolation of scattered data, Naval Postgraduate School Technical Report, NPS-53-79-003 (1979).

[22] Vanek, P., Mandel, J. and Brezina, M.: Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems, Technical Report, UCD-CCM-036 (1995).

[23] 藤井昭宏，西田　晃，小柳義夫：領域分割による並列 AMG アルゴリズム，情報処理学会論文誌コンピューティングシステム，Vol.44, SIG6(ACSI), pp.9-17 (2003).

[24] 藤井昭宏：SMAC 法による流体解析を対象とした AMG ライブラリの自動チューニング方式，電子情報通信学会論文誌 D，Vol.J96-D, No.5, pp.1321–1329 (2013).

[25] Chan, C, Ansel, J. and Wong, Y.L., Amarasinghe, S. and Edelman, A.: Autotuning multigrid with PetaBricks, *SuperComputing '09*, New York, USA (2009).

[26] FX10 Supercomputer System, available from ⟨http://www.cc.u-tokyo.ac.jp/system/fx10/⟩（参照 2018-03-10）.

[27] Virtual Library of Simulation Experiments: Test Functions and Datasets, available from ⟨https://www.sfu.ca/~ssurjano/index.html⟩（参照 2017-12-26）.

[28] Reedbush-U　スーパーコンピュータシステム，入手先 ⟨https://www.cc.u-tokyo.ac.jp/system/reedbush/⟩（参照 2017-12-26）.

付　　録

A.1　テスト関数推定結果

テスト関数の推定結果の詳細を表 **A·1**，表 **A·2**，表 **A·3**，表 **A·4**，表 **A·5**，表 **A·6** に示す．

表 A·1　テスト関数推定結果 (1)

Table A·1　Test function estimation result (1).

関数名	Sphere 関数		Booth 関数		Matyas 関数	
パラメタの組合せ	83,521（$= 17 \times 17 \times 17 \times 17$）		99,225（$= 21 \times 21 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）	
手法	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search
標本点数	293.7	141.3	326.7	191.3	312.6	171.4
方向決定	256.0	101.4	288.9	131.2	275.9	121.0
1 次元探索	37.7	39.9	37.8	60.1	37.8	60.1
1 次元探索回数	7.0	10.8	7.5	12.6	7.3	10.9
4 方向	——	4.8	——	5.7	——	4.0
16 方向	——	1.0	——	1.9	——	1.9
32 方向	——	1.0	——	1.0	——	1.0
40 方向	——	4.0	——	4.0	——	4.0
推定時間 [s]	0.26×10^{-3}	0.41×10^{-3}	0.26×10^{-3}	0.49×10^{-3}	0.25×10^{-3}	0.40×10^{-3}
最適値とのずれ	0.0	0.0	0.0	0.0	0.0	0.0
最適値とのずれ 10%以内（最適値を推定した個数）	83,521（83,521）	83,521（83,521）	99,225（99,225）	99,225（99,225）	65,025（65,025）	65,025（65,025）
ランダム探索 [%]	0.6	0.8	0.3	4.43	0.1	0.4

表 A·2　テスト関数推定結果 (2)

Table A·2　Test function estimation result (2).

関数名	Ackley 関数		Beale 関数		Goldstein-Price 関数	
パラメタの組合せ	83,521（$= 17 \times 17 \times 17 \times 17$）		81,225（$= 19 \times 19 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）	
手法	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search
標本点数	255.7	136.8	337.9	166.3	309.2	168.7
方向決定	220.3	95.2	220.3	95.2	271.2	115.9
1 次元探索	35.4	41.7	43.81	55.4	37.9	52.8
1 次元探索回数	6.81	10.6	7.8	11.7	7.4	11.0
4 方向	——	4.6	——	5.1	——	4.1
16 方向	——	1.0	——	1.6	——	1.9
32 方向	——	1.0	——	1.0	——	1.0
40 方向	——	4.0	——	4.0	——	4.0
推定時間 [s]	0.26×10^{-3}	0.43×10^{-3}	0.28×10^{-3}	0.51×10^{-3}	0.25×10^{-3}	0.40×10^{-3}
最適値とのずれ	5.6	1.5	1.3	2.4	18.7	19.9
最適値とのずれ 10%以内（最適値を推定した個数）	60,857（60,857）	77,625（77,625）	50,150（50,150）	49,047（49,047）	43,899（43,899）	48,375（48,375）
ランダム探索 [%]	14.8	16.1	0.39	0.21	0.01	0.08

表 A·3　テスト関数推定結果 (3)

Table A·3　Test function estimation result (3).

関数名	Bukin N.6 関数		Levi N.13 関数		Three-hump camel 関数	
パラメタの組合せ	61,200（$= 17 \times 16 \times 15 \times 15$）		119,025（$= 23 \times 23 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）	
手法	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search
標本点数	265.3	129.4	304.0	145.0	295.1	147.1
方向決定	231.3	91.2	266.7	101.5	255.6	100.2
1 次元探索	34.0	38.3	266.7	101.5	39.5	46.9
1 次元探索回数	7.0	10.4	7.4	11.2	7.2	10.9
4 方向	—	4.4	—	5.2	—	4.9
16 方向	—	1.0	—	1.0	—	1.0
32 方向	—	1.0	—	1.0	—	1.0
40 方向	—	4.0	—	4.0	—	4.0
推定時間 [s]	0.22×10^{-3}	0.37×10^{-3}	0.22×10^{-3}	0.37×10^{-3}	0.27×10^{-3}	0.45×10^{-3}
最適値とのずれ	16.1	16.2	34.4	16.2	0.3	0.4
最適値とのずれ 10%以内（最適値を推定した個数）	10,004（10,004）	11,250（11,250）	93,634（93,634）	106,650（106,650）	25,652（25,652）	18,675（18,675）
ランダム探索 [%]	0.02	18.7	0.4	0.5	0.4	0.5

表 A·4　テスト関数推定結果 (4)

Table A·4　Test function estimation result (4)

関数名	Easom 関数		Eggholder 関数		McCormick 関数	
パラメタの組合せ	90.000（$= 20 \times 20 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）	
手法	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search
標本点数	364.3	175.4	245.5	128.1	330.6	151.3
方向決定	315.2	118.7	213.4	88.5	293.3	105.7
1 次元探索	49.2	56.7	32.1	39.6	37.3	45.5
1 次元探索回数	8.4	11.5	6.8	10.1	7.6	11.6
4 方向	—	4.7	—	3.8	—	5.4
16 方向	—	1.7	—	1.3	—	1.1
32 方向	—	1.1	—	1.0	—	1.0
40 方向	—	4.0	—	4.0	—	4.0
推定時間 [s]	0.26×10^{-3}	0.43×10^{-3}	0.26×10^{-3}	0.43×10^{-3}	0.26×10^{-3}	0.43×10^{-3}
最適値とのずれ	0.0	0.0	0.4	0.4	0.3	0.6
最適値とのずれ 10%以内（最適値を推定した個数）	90.000（90.000）	90.000（90.000）	6,469（3,637）	8,100（2,925）	52,120（52,120）	41,625（41,625）
ランダム探索 [%]	0.3	0.3	0.5	0.6	0.1	0.2

表 A·5　テスト関数推定結果 (5)

Table A·5　Test function estimation result (5).

関数名	Schaffer 関数		Five-Well 関数		Griewank 関数	
パラメタの組合せ	65,025（$= 17 \times 17 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）		65,025（$= 17 \times 17 \times 15 \times 15$）	
手法	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search	(1) 1 step search	(2) 4 steps search
標本点数	301.1	170.0	289.2	141.8	247.3	132.6
方向決定	263.0	121.2	253.8	101.3	215.2	94.7
1 次元探索	38.1	48.8	35.5	40.5	32.1	37.9
1 次元探索回数	7.3	11.2	7.0	10.8	6.7	10.3
4 方向	——	3.8	——	4.8	——	4.3
16 方向	——	1.7	——	1.0	——	1.0
32 方向	——	1.0	——	1.0	——	1.0
40 方向	——	4.6	——	4.0	——	4.0
推定時間 [s]	0.26×10^{-3}	0.44×10^{-3}	0.26×10^{-3}	0.43×10^{-3}	0.26×10^{-3}	0.43×10^{-3}
最適値とのずれ	0.1	0.1	0.3	0.2	0.1	0.1
最適値とのずれ 10%以内 （最適値を推定した個数）	43,475 （43,475）	45,225 （45,225）	24,489 （24,489）	30,825 （30,825）	33,964 （4,381）	35,325 （5,175）
ランダム探索 [%]	0.03	0.2	0.05	0.09	0.02	0.07

表 A·6　テスト関数推定結果 (6)

Table A·6　Test function estimation result (6).

関数名	Michalewicz 関数	
パラメタの組合せ	65,025（$= 17 \times 17 \times 15 \times 15$）	
手法	(1) 1 step search	(2) 4 steps search
標本点数	295.7	145.8
方向決定	256.4	100.2
1 次元探索	39.3	45.6
1 次元探索回数	7.2	10.7
4 方向	——	4.6
16 方向	——	1.1
32 方向	——	1.0
40 方向	——	4.0
推定時間 [s]	0.26×10^{-3}	0.43×10^{-3}
最適値とのずれ	0.2	0.1
最適値とのずれ 10%以内 （最適値を推定した個数）	36,730 （36,730）	49,725 （49,725）
ランダム探索 [%]	0.03	0.43

望月 大義　（正会員）

1993 年生．2016 年工学院大学情報学部コンピュータ科学科卒業．2018 年工学院大学大学院工学研究科情報学専攻修士課程修了．

藤井 昭宏　（正会員）

1975 年生．1999 年東京大学理学部情報科学科卒業．2004 年同大学大学院情報理工学系研究科コンピュータ科学専攻博士課程修了．博士（情報理工学）．2004 年工学院大学 CPD センタ講師となり，2015 年 4 月同大学情報学部コンピュータ科学科准教授，現在に至る．ハイパフォーマンスコンピューティング，階層的なアルゴリズム，不規則疎行列に関する並列処理等の研究に従事．IEEE-CS，電子情報通信学会各会員．

田中 輝雄　（正会員）

1958 年生．1983 年電気通信大学大学院情報数理工学研究科修士課程修了．2007 年同大学院情報システム学研究科博士課程修了．博士（工学）．1983 年（株）日立製作所中央研究所入所．2011 年工学院大学情報学部教授．専門は，大規模数値計算アルゴリズム．日本応用数理学会，IEEE-CS 各会員．