Vol.2019-HPC-168 No.14
2019/3/6

ソフトウェア自動チューニングにおける 反復 2 次元 d-Spline 探索法の提案と評価

范谷瑛[†1]　関直人[†1]　多部田敏樹[†1]　藤井昭宏[†1]　田中輝雄[†1]

ソフトウェア自動チューニングは計算機環境に合わせて，自動的に最適値となる性能パラメタの値の組み合わせを見つける技術である．我々は効率の良い探索手法について研究してきた．研究のひとつとして反復 1 次元 d-Spline 探索法を提案している．これは，はじめに性能パラメタの取り得る値の組み合わせから基点を選び，その周辺を実測して，その中で性能の良い値を選び，その値を含む方向に 1 次元 d-Spline で推定最適値を探し，その推定最適値をまた初期点とし，繰り返し探索する手法である．しかし，1 次元 d-Spline では，高次元性能パラメタのカバーリングが困難であるため，探索反復回数が多くなってしまう．そこで，我々はベースとなる d-Spline を 2 次元化し，より高次元性能パラメタをカバーリングできる反復 2 次元 d-Spline 探索法を提案する．AMGS に適用して，推定の正確性と対象プログラムに影響を与えない推定時間であることを示した．

1. はじめに

1.1 背景と目的

近年，計算機の性能は複数の計算ノードからなる高速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装されたことにより向上している．そのため，プログラムの性能の影響をパラメタ化したものである性能パラメタをそれぞれの計算機環境にあわせてチューニングするのはより困難を極めている．この課題に対して，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]．ソフトウェア自動チューニングは，チューニング対象のプログラムの性能パラメタの取り得る値から効率よく最適な組み合わせを自動的に設定する技術である．また，性能パラメタの取り得る値の集合を性能パラメタ空間と呼ぶこととする．

本研究では，ユーザのプログラムの実行時にチューニングを行う，実行時自動チューニングを対象としている．実行時自動チューニングでは，ユーザのプログラムの 1 反復ごとにチューニングが行なわれる．そのため，性能パラメタの最適値を推定する際の推定時間などのコストは，ユーザの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，ソフトウェア自動チューニングの手法のひとつとして，標本点逐次追加型性能パラメタ推定法（IPPE 法）[3][4]を研究してきた．この手法は，はじめにいくつか標本点を選択し，標本点とする．この標本点から近似関数を生成し，近似関数上で最適な性能パラメタを選択する．選択された性能パラメタを標本点に追加し，近似関数を更新する．この手順を繰り返し，連続で同じ性能パラメタの値が選択されたら推定を終了する．ここで用いられる近似関数は d-Spline と呼ぶスプライン系の関数である．d-Spline は生成・更新の計算量が少ない関数である．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[5]．しかし，性能パラメタの次元を上げるにつれて，d-Spline の計算量が増大してしまい，ユーザのプログラムに対して推定コストが無視できない時間となってしまう．そこで，我々は反復 1 次元 d-Spline 探索を提案した[6]．多次元性能パラメタ空間において，基本となる点の周囲の点を計測し，軸方向の決定と，1 次元 d-Spline 探索を繰り返し行うことによって，最適な性能パラメタの値の組み合わせを見つける手法である．この手法によって計算コストを抑えることができ，実際に複数次元性能パラメタ空間に対して有用性を示してきた．しかし，この手法は性能パラメタ空間がより高次元になるにつれて，1 次元 d-Spline 探索では，広域な性能パラメタの値をカバーリングすることは困難となってしまい，探索の反復回数が増加し，探索効率が下がってしまう．

そこで本研究では，1 次元 d-Spline に代わり，ユーザのプログラムに対して，推定コストが無視でき，より広域をカバーリングできる 2 次元 d-Spline 探索を繰り返し適応する手法を実現する．これを反復 2 次元 d-Spline 探索法と呼ぶことにする．反復 1 次元 d-Spline 探索法の線の探索に代わり，面の探索を用いることでより広域をカバーリングできる推定手法であることを示す．

1.2 関連研究

パラメタ推定方法において，性能モデルとして空間的相関にもとづいて構成される Kriging モデルを用いた Surrogate Assisted Tuning(SAT)法が提案されている[7]．他にも pragma を追加するだけで自動チューニング機能付きプログラムを生成する自動チューニング基盤 ppOpen-AT[8][9][10]や実測の錯乱効果の影響を軽減できるベイズ統計を用いた自動チューニングの数理ソフトウェアの ATMathCoreLib が提案されている[11][12][13].

†1 工学院大学
Kogakuin University.

実際の推定では，計算機環境の外乱やキャッシュの競合などが要因となり，ユーザのプログラムの実行時間には，ブレが含まれることがある．このブレにより，性能パラメタ推定は実時間のブレなどに影響をうけるため誤った推定結果が得られる可能性がある[14]．しかし，本研究では，考慮しないものとする．

2. 標本点逐次追加型性能パラメタ推定法

2.1 概要

本節では，標本点逐次追加型性能パラメタ推定法（IPPE 法）について説明する．図 2.1 に IPPE 法のアルゴリズムを示す．はじめにいくつか標本点を選択し，実測を行い，初期標本点とする．初期標本点から近似関数を生成し，近似関数の最小値を最適値と考え，性能パラメタの値を推定する．推定された最適値を推定点とする．このとき，推定点がすでに実測されていた場合，標本点でない近似関数の各データの前後の 2 階差分の値を求める．近似関数上で前後の 2 階差分の値の差が大きいほど性能の変化が激しいことを表す．前後の 2 階差分の値の差がもっとも大きい性能パラメタの値に対して，最適値になり得ると考え，推定点とし，標本点に追加する．推定された性能パラメタの値を標本点に追加し，近似関数を更新する．この手順を繰り返し，連続で同じ性能パラメタの値が選択されたら推定を終了する．つまり，以下の 3 項目を繰り返すことで性能パラメタの推定を行う．

1. 推定点に対して，実測し，標本点に追加
2. 追加した標本点をもとに，近似関数を更新
3. 近似関数から最適値を推定する

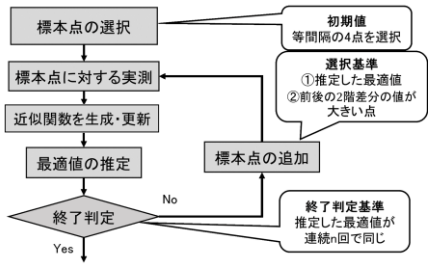
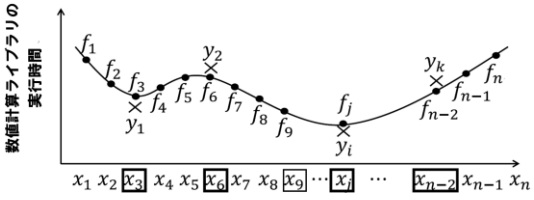


図 2.1 IPPE 法のアルゴリズム

2.2 1 次元性能パラメタ空間推定

本節では，1 次元性能パラメタ空間の IPPE 法で用いられる近似関数について説明する．一般に数値計算ライブラリの性能パラメタでの実行時間で構成される関数形は，滑らかさや凸性は保障されない．また，標本点を選択・追加するたびに近似関数を更新する必要がある．そこで，計算コストが低く，微分連続性を用いない d-Spline を用いる．d-Spline は，実測値の動きに柔軟に追従し，さらに，少ない



□：性能パラメタの取り得る値（内，□：標本点）
y_i：標本点に対する数値計算ライブラリの実測値$1 \le i \le k,\ k < n$)

図 2.2 近似関数fと実測値yの関係

$$E = \begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & 0 & 1 & & \\ & & 0 & 1 & \\ 0 & & & \ddots & \\ & & & 0 & 1 \end{pmatrix},\quad D = \begin{pmatrix} 1 & -2 & 1 & & & 0 \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & 1 & -2 & 1 \end{pmatrix}$$

図 2.3 E，Dの構造

$$Z = \begin{pmatrix} E^tE \\ \alpha D \end{pmatrix},\quad b = \begin{pmatrix} E^t\mathrm{y} \\ 0 \end{pmatrix}$$

図 2.4 Z，$\boldsymbol{b}$の構造

標本点でも安定に解が得られる近似関数である．d-Spline は近似関数$f(x)$として，n個の離散点x_j上の値$f_j = f(x_j)$，$1 \le j \le n$とベクトル$\boldsymbol{f} = (f_1, f_2, f_3, \cdots, f_j, \cdots, f_n)^t$で表現する．滑らかさを出すために，$n$は性能パラメタの取り得る値の個数$N$より十分大きく取る$(N < n)$．性能パラメタの取り得る$N$個の値から$k$個の実測値$(k \le N)$が得られているならば，それを$y_i(1 \le i \le k)$とし，$\boldsymbol{y} = (y_1, y_2, y_3, \cdots, y_i, \cdots, y_k)^t$とする．近似関数$\boldsymbol{f}$と実測値$\boldsymbol{y}$，および性能パラメタの取り得る値との関係を図 2.1 に示す．横軸は，近似関数の定義域である．図中の四角で囲んだ値が性能パラメタの取り得る値となるが，すべてが標本点となるわけではない．縦軸は，近似関数$\boldsymbol{f}$の形状と，標本点に対する実測値y_iであり，単位は実行時間となる．$\boldsymbol{y}$のみでは，個数は$\{y_i\}$より$\{f_j\}$が多いため，$\boldsymbol{f}$を規定するために$\boldsymbol{f}$の滑らかさを 2 階差分$|f_{j-1} - 2f_j + f_{j+1}|$，$2 \le j \le n-1$で表す．この近似関数$\boldsymbol{f}$を評価式$\min(\|\boldsymbol{y} - E\boldsymbol{f}\|^2 + \alpha^2\|D\boldsymbol{f}\|^2)$で選ぶ．ここで$E$，$D$は図 2.2 に示すような行列であり，サイズはそれぞれ$k \times n$，$(n-2) \times n$である．Eは実測値と近似関数の対応を表し，Dは近似関数の滑らかさを定義している．つまり，評価関数の第 1 項は実測値$\boldsymbol{y}$と近似関数$\boldsymbol{f}$との距離であり，第 2 項は近似関数の滑らかさを表す．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実測値に追従し，大きくするほど直線になる．評価関数$\min(\|\boldsymbol{y} - E\boldsymbol{f}\|^2 + \alpha^2\|D\boldsymbol{f}\|^2)$を解くためには，次の最小二乗問題$\min(\|\boldsymbol{b} - Z\boldsymbol{f}\|^2)$を $\boldsymbol{f}$ について解けばよい．Zと$\boldsymbol{b}$は図 2.3 のように表す．fill-in(零要素が非零要素になること)を抑えるために，E^tをEと$\boldsymbol{y}$の左側から掛けるよう変形を行っている．

この最小二乗問題を解く手法として，Zを直行行列Qと上三角行列Rの積に分解する QR 分解を適応する．QR 分解に

は Givens 変換法を用いる．$\|Q\| = \|Q^t\| = 1$に注意すると，$\|\boldsymbol{b} - Z\boldsymbol{f}\|^2$は$\|Q^t\boldsymbol{b} - R\boldsymbol{f}\|^2$と変換することができるため，後退代入により$\boldsymbol{f}$が求まる．このとき，標本点を追加するたび$Z$に戻って Givens 変換を施す必要はない．$k-1$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 行にのみ Givens 変換を施せばよい．このように，近似関数 d-Spline は行列Rを再利用することで$\boldsymbol{O}(n)$の計算量で求めることができる．なお，この 1 次元 d-Spline を用いた IPPE 法を 1 次元 d-Spline 探索と呼ぶ．

2.3 多次元性能パラメタ空間推定

本節では，d-Spline を用いた多次元性能パラメタ空間の推定について述べる[5]．2 次元性能パラメタ空間での性能パラメタ推定は，d-Spline を 2 次元に拡張することによって実現し，評価式も 2.2 節で述べた$\min(\|\boldsymbol{y} - E\boldsymbol{f}\|^2 + \alpha^2\|D\boldsymbol{f}\|^2)$を用いる．

性能パラメタ空間における実測データ $\boldsymbol{y}$ は，2 次元離散格子点上の$y_{i1,i2}$であり，関数 d-Spline $f(x_{j_1,j_2})$ は$f = (f_{1,1}, f_{2,1}, \dots, f_{j_1,j_2}, \dots, f_{n_1,n_2})^t$と離散点$x_{j1,j2}, (1 < j_1 < n_1, 1 < j_2 < n_2)$で表す．2 次元 d-Spline$\boldsymbol{f}$ の滑らかさは，表 2-1 のような差分の式で定義する．

2 次元性能パラメタの$(n_1 \times n_2 - 4) \times (n_1 \times n_2)$の滑らかさを表す行列 D を図 2.5 に示す．2 次元 d-Spline の形状を図 2.6 に示す．なお，この 2 次元 d-Spline を用いた IPPE 法を 2 次元 d-Spline 探索と呼ぶ．性能パラメタの領域はj_1, j_2で表現され性能パラメタの取り得る値を表す．縦軸は対象プログラムの実行時間を表す．赤い線が実際のデータであり，青い線が実際に生成された 2 次元 d-Spline の形状を表している．同様に 3 次元性能パラメタ空間の推定は，d-Spline を$n_1 \times n_2 \times n_3$個の離散点上の値$x_{j_1,j_2,j_3}(1 \le j_1 \le n_1, 1 \le j_2 \le n_2, 1 \le j_3 \le n_3)$による直方体領域で表現する．その領域内部では滑らかさを$\left|f_{j_1-1,j_2,j_3} + f_{j_1,j_2-1,j_3} + f_{j_1,j_2,j_3-1} - 6f_{j_1,j_2,j_3} + f_{j_1+1,j_2,j_3} + f_{j_1,j_2+1,j_3} + f_{j_1,j_2,j_3+1}\right|, (2 \le j_1 \le n_1 - 1, 2 \le j_2 \le n_2 - 1, 2 \le j_3 \le n_3 - 1)$のように 7 点差分で表す．

表 2-1 2 次元 d-Spline$\boldsymbol{f}$の境界条件式

f_{j_1,j_2}	1	$[2, n_2 - 1]$	n_2
1	$\lvert -2f_{1,1} + f_{1,2} + f_{2,1}\rvert$	$\lvert f_{1,j_2-1} - 3f_{1,j_2} + f_{1,j_2+1}\rvert$	$\lvert f_{1,n_2-1} - 2f_{1,n_2} + f_{2,n_2}\rvert$
$[2, n_1 - 1]$	$\lvert f_{j_1-1,1} - 3f_{j_1,1} + f_{j_1+1,2}\rvert$	$\lvert f_{j_1-1,j_2} + f_{j_1,j_2-1} - 4f_{j_1,j_2} + f_{j_1,j_2+1} + f_{j_1+1,j_2}\rvert$	$\lvert f_{j_1-1,n_2} - 3f_{j_1,n_2} + f_{j_1+1,n_2}\rvert$
n_1	$\lvert f_{n_1-1,1} - 2f_{n_1,1} + f_{n_1,2}\rvert$	$\lvert f_{n_1,j_2-1} - 3f_{n_1,j_2} + f_{n_1,j_2+1}\rvert$	$\lvert f_{n_1-1,n_2} + f_{n_1,n_2-1} - 2f_{n_1,n_2}\rvert$

D =

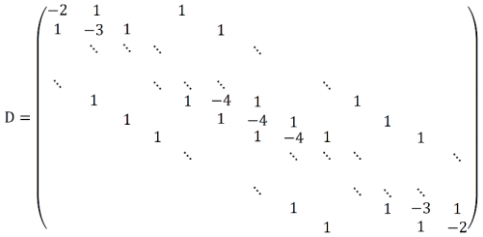


図 2.5 2 次元性能パラメタに対する行列 D の構造

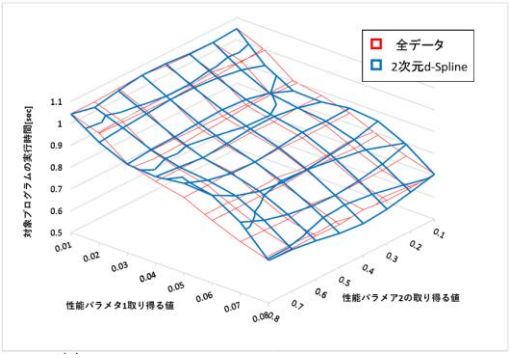


図 2.6 2 次元 d-Spline の形状

2.4 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．m 次元の性能パラメタ空間の 2 階差分式は，ひとつの次元に対して左右に隣接した接点との差分をとることで，$\left|\sum_{i=1}^{m} f_{j_1,\dots,j_i-1,\dots,j_m} - 2mf_{j_1,\dots,j_i,\dots,j_m} + f_{j_1,\dots,j_1+1,\dots,j_m}\right| (1 \le j_i \le n_1, 1 \le i \le m)$ のように表現される

このことから，m 次元の性能パラメタ空間のための 2 階差分は$2m+1$点差分となり，これに対応する 2 階差分要素 D のうち非零要素が格納されている帯の本数も$2m+1$となる．

次に行列 D を QR 分解するときの計算コストについて考える．例えば，性能パラメタが 2 つの場合，評価関数$\min(\|\boldsymbol{y} - E\boldsymbol{f}\|^2 + \alpha^2\|D\boldsymbol{f}\|^2)$の第 2 項の$D\boldsymbol{f}$は図 2.7 のような構造となる．行列D$\boldsymbol{f}$の構造は，$n_2 \times n_2$のブロック行列が行と列方向にそれぞれ$n_1$個並んだ形となる．ただし，1 番上と下のブロック行列は$(n_2 - 2) \times (n_2)$である．

行列 D の i 行の下三角部分の要素の消去を考える．第 1 列から$i-1$列の要素は，i 行目で消去する必要がある．前節の 2.1 節で述べたが，すでに d-Spline が生成されており，QR 分解が完了して行列 R が得られているとき，新たな標本点から d-Spline を更新することを考えると行列 E の 1 行を行列 R に，実測データ $\boldsymbol{y}$ を b に追加する．よって，1 行のみに QR 分解を行えばよい．

行列 D の帯幅は$2n_2$であるため，第 1 列から$i - n_2$の要素はすでに零であり，行ごとに消去させる要素の数はn_2である．また，第 1 列から$i - n_2$列までの要素は零であり，1 行ごとに消去させる要素はn_2である．Givens 法で要素を消去させるために 2 行のベクトル 12 演算を行う．したがって，計算コストは各行について$\boldsymbol{O}(n_2) \sim \boldsymbol{O}(2n_2)$までである．そのため，QR 分解の計算コストは$\boldsymbol{O}(2n_2)$となる．また，後退代入にかかるコストは$n_1 \times n_2$となる．よって，QR 分解の更新のためにかかる計算コストは$\boldsymbol{O}(n_2^3 \times n_1)$となる．

多次元性能パラメタ推定は QR 分解による fill-in が増加するため計算コストが膨大になる．多次元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 2.1 に示す．ここで，$n_k (k = 1 \dots 4)$は性能パラメタの取り得る

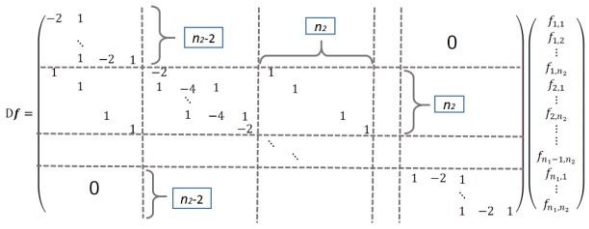


図 2.7 2 次元の d-Spline の滑らかさ Df の構造

表 2-2 各次元での d-Spline の計算量

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

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．同様に反復 2 次元 d-Spline 探索のn_1, n_2は次元数によって変化しない．性能パラメタが増加するほど d-Spline による計算の複雑さが少ないという利点が減ってしまう．

3. 反復 1 次元 d-Spline 探索法

本研究では，複数性能パラメタによって構成される離散点上のデータで複数パラメタ空間の最適値の推定を行う．複数の性能パラメタ推定は 2.2 節で述べたように，滑らかさを表す行列 D は性能パラメタの次元が上がるにつれて複雑になる．そのため，自動チューニングのコストが増加してしまう．それに対して，計算量が$O(n)$である 1 次元 d-Spline 探索を繰り返し用いた反復 1 次元 d-Spline 探索が提案されている．図 3.1 に推定の流れを示す．反復 1 次元 d-Spline 探索の手順は以下の 2 項目を繰り返し行う．

1. 探索方向を決定

   推定点の周囲の点を計測し，その中で最小値を選択して，その点ともとの標本点の 2 点を含んだ直線を決定する．性能パラメタが 2 つの場合は，x,y 軸と，軸と斜め 45° 方向である 2 つの合計 4 方向である．図 3.2 に探索する 4 方向を示す．太線になっている部分が探索方向であり，赤い点が推定点であり，青い点が探索方向を決定するために実測する離散点 8 点を示している．

2. 1 次元 d-Spline 探索

   推定手順について，2 次元性能パラメタ推定の 1 例を用いて図 3.3 に示す．これは，右側に山 1 つ，左側に谷 2 つの Frank 関数[15]の関数値を表した等高線図である．それぞれ性能パラメタの取り得る値を 31 個とする．

   はじめに，任意に初期点を設定する．次に，探索方向の決定のために右図の赤い点の周り 8 点の青い点を実測し，赤い矢印の 4 方向の中から探索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 3.4 に示す．図中の（1），（2）は図 3.4 の（1），（2）に対応し，横軸は性能パラメタの取り得る値，縦軸は関数値を示している．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-Spline 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．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パラメタの推移となる．2 度目以降に推定する直線を決定するときには 1 度探索をした直線を探索しないために直前に探索をした直線方向の 2 点を除く推定点の周り 6 点を実測する．6 点の中から最小値を求め，その点と推定点を含む直線を決める．また，推定は 1 次元 d-Spline 探索によって異なる 3 方向において連続で最適と推定された場合によって終了する．

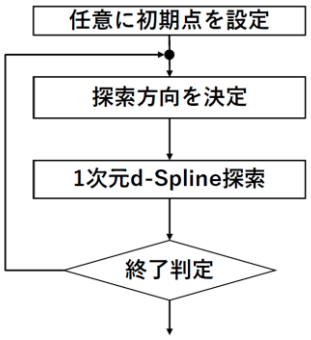


図 3.1 反復 1 次元 d-Spline 探索のアルゴリズム

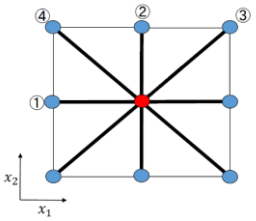


図 3.2 2 次元パラメタ空間における探索する 4 方向

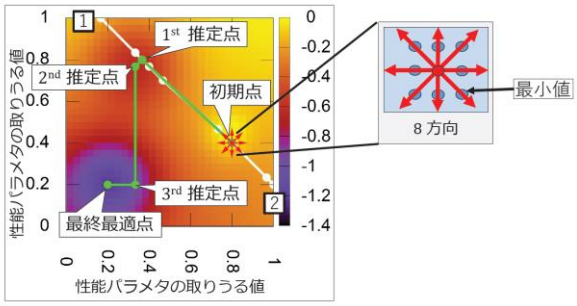


図 3.3 反復 1 次元 d-Spline 探索での探索経路

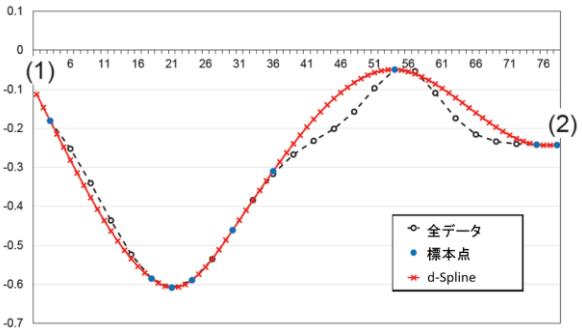


図 3.4 d-Spline の形状(図 3.3 中の白い線)

4. 提案手法（反復 2 次元 d-Spline 探索法）

本研究では，より高次元な複数性能パラメタ空間に対して，最適値の推定を行う．1 次元 d-Spline 探索だとより広範囲のカバーリングが困難になってしまい，最適値を探索するのに反復回数がより増加してしまう．それに対して，ベースとなる d-Spline を直線で探索する 1 次元 d-Spline 探索から面で探索する 2 次元 d-Spline に切り替える．それにより，広範囲をカバーリングする反復 2 次元 d-Spline 探索法を提案する．ここで，反復 2 次元 d-Spline 探索法に，(1) 性能パラメタ推定の正確性，(2) 推定にかかる時間コストの 2 つが課題として挙げられる．

図 4.1 に推定の流れを示す．反復 2 次元 d-Spline 探索の手順は以下の 2 項目を繰り返し行う．

1. 探索面を決定

   推定点の周囲の点を実測し，その中で比較的小さい値を含む軸を 2 つ選択し，探索面を決定する．例えば，性能パラメタ 3 つの場合，図 4.2 の右のような方法で探索面が決定される．太線がそれぞれ性能パラメタの *x,y,z* 軸であり，赤い点が推定点，青い点が標本点である．緑の点は，軸ごとの最小値を決定し，決定された軸ごとの最小値の中でも最も小さい 2 つの標本点を表している．この場合だと *x* 軸と *y* 軸が決定され *xy* 面について探索を行う．比較として用いる反復 1 次元 d-Spline 探索では，方向決定に斜め方向を加えているが，本研究で提案する反復 2 次元 d-Spline 法では，実装難易度から斜め方向の軸を含んだ面の探索は行っていない．

2. 2 次元 d-Spline 探索

   決定された探索面に対して 2 次元 d-Spline 探索を行う．図 4.2 と 4.3 の 3 次元性能パラメタの推定を例として説明する．図中の斜め左下方向は *x* 軸を，斜め右下方向は *y* 軸を，縦方向 *z* 軸を表している．緑の点は初期標本点を表し，青，赤の点は推定点を表し，赤い点は最適値でもある．緑，青，赤の面は，探索された面を表しており，黒い矢印は推定点の推移を示している．図の赤い点のように複数面で連続に推定点に選ばれるとその推定点を最適値として探索は終了する．

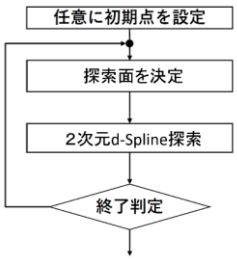


図 4.1 反復 2 次元 d-Spline 探索のアルゴリズム

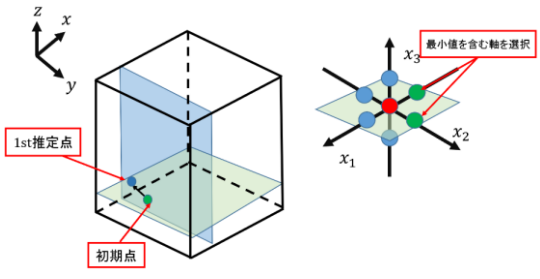


図 4.2 反復 2 次元 d-Spline 探索の経路

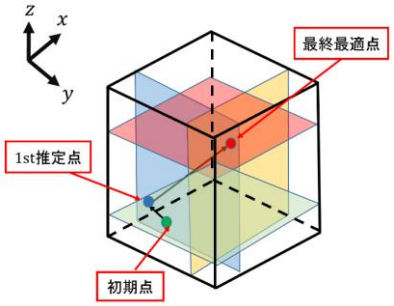


図 4.3 最終最適点までの経路

表 4-1 探索面数

次元数	探索面数
3	${}_3C_2 = 3$
4	${}_4C_2 = 6$
5	${}_5C_2 = 10$
m	${}_mC_2 = \frac{1}{2}m(m-1)$

性能パラメタ数ごとの探索面の個数について説明する．例えば 3 つの性能パラメタより，探索面を決定する場合，表 4-1 ように組み合わせの式で表すことができる．

2 次元 d-Spline を生成するときの初期標本点について説明する．まず 2 次元 d-Spline の形状を保つために必要最低限である四隅の 4 点を初期標本点にする．性能パラメタのそれぞれの取り得る値をn_1, n_2としたとき，残りの初期標本

点を性能パラメタの端を除いた，$(n_1 - 2 \times n_2 - 2)$の領域内で，直線と斜め直線に重ならないように初期標本点を選択する．

5. 実問題への適用

5.1 AMG 法

本研究では代数的マルチグリッド（AMG）法のライブラリである AMGS の性能パラメタに対して適用する[16][17][18]. AMGS で用いられている AMG 法は連立一次方程式を高速に解く手法のひとつであり，構築部と解法部に分かれる．構築部では，問題行列の情報からより小さいサイズ（粗いレベル）の連立一次方程式を生成し，この工程を繰り返すことで複数の階層構造を構成する．解法部では，構築部で作られた粗いレベルの問題を利用し効率よく解く．解法部の構造を図 5.1 に示す．V 字形の階層構造をしているため V-cycle と呼ぶ. 与えられた問題行列は階層 1 に設置され，粗いレベルになるほど小規模の行列が設置される．階層移動には補間行列と制限行列が使用されるが，これらの行列は構築部で生成される．

AMGS では，構築部において，粗いレベルの行列や補間行列などを生成する Smoothed Aggrigation に基づく AMG（SA-AMG ）法を用いている[19]. SA-AMG 法では与えられた問題行列をグラフ構造で考える．つまり，問題行列の各行は節点に対応し，非ゼロ要素が節点同士を結ぶ辺となる．グラフ構造として考えた節点全体を以下のルールにもとづいてアグリゲートと呼ばれる節点集合に分解する．
生成されたアグリゲートの情報から補間行列と制限行列が作られる．これらの要素を再帰的に集約することでより未知数の少ない（粗い）問題を生成し，マルチレベルを構成する．表 5-1 に AMGS の性能パラメタの一覧を示す．表のように AMGS はその性能に大きな影響を与える性能パラメタを多数持つ Damped_Jacobi と Strong_Connection，GS iteration#は実数値である．そのため，性能パラメタの組み合わせを考えるとその数は膨大となる．このことより AMGS は複数次元での性能パラメタ推定が必要になる．本研究では，以下の 3 階層ごとの 2 性能パラメタについてチューニングを行う．

●Damped_Jacobi

AMG 法における解法部の各レベルでの緩和法の加速度係数である．この値により緩和法を減速させることで収束性が安定することが知られている

●Strong_Connection

SA-AMG 法における構築部の強連結成分のしきい値である．アグリゲート生成の際に弱連結成分が含まれると収束性が悪くなることがある．この値を用いて，アグリゲートのサイズや弱連結成分をどの程度含めるか調整する

本研究では，4 つと 5 つの性能パラメタの推定を行った．

表 5-1 AMG 法の性能パラメタ一覧

AMGの性能パラメタ	階層1	階層2	階層3
Strong Connection	[0.01,0.08] 8通り	[0.01,0.08] 8通り	[0.01,0.08] 8通り
Damped Jacobi	[0.1,0.8] 8通り	[0.1,0.8] 8通り	[0.8,1.5] 8通り
Aggregation	Coupled aggregation or Independent aggregation		
GS iteration#	1 or 2	1 or 2	1 or 2
GS acceleration	[0.6,1.1] 6通り	[0.6,1.1] 6通り	[0.6,1.1] 6通り
MG Cycle	V or W		

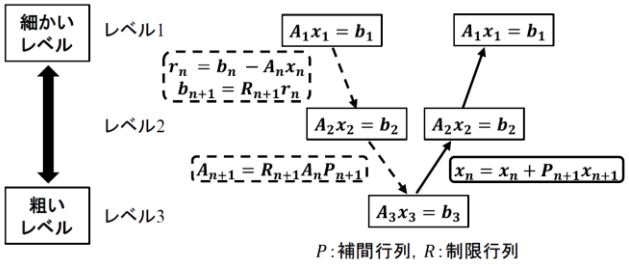


図 5.1 解法部(V-cycle)

5.2 4 次元性能パラメタへの適用

4 つ性能パラメタについて，適用実験を行う．表 5-3 に性能パラメタの詳細を示す．計算機環境としては，東京大学に設置されている Reedbush-U スーパーコンピュータシステムを使用した[20]. 1 ノードにおける使用を表 5-2 に示す．今回は 2 ノード使い，各ノードで 16 プロセス，スレッドで実行した．実験では，最適値を出力するまでを 1 試行とし，それぞれ性能パラメタ 4096 個を初期標本点として，4096 回試行を行った．反復 1 次元 d-Spline 探索法と反復 2 次元 d-Spline 探索法を比較する. 表 5-4 に実験結果を示す.

相対誤差 10%以下とは，推定した最適値$vEOP$と真の最適値vOPとの相対誤差$\frac{|vEOP-vOP|}{|vOP|} < 0.1$である試行数を示している．課題で上げていた推定の正確性について，反復 1

表 5-2 Reedbush-U の仕様

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

表 5-3 4 次元性能パラメタの詳細

性能パラメタ	Strong_Connection		Damped_Jacobi		
階層	1	2	1	2	3
範囲	[0.01,0.08]	[0.01,0.08]	[0.1,0.8]	[0.1,0.8]	[0.8,1.5]
データの間隔	0.01	0.01	0.1	0.1	0.1
パラメタの取り得る個数	8	8	8	8	8
パラメタの組合わせ総数	32,768(=8×8×8×8×8)				

表 5-4 4 次元性能パラメタにおける実験結果

			反復2次元d-Spline探索法(提案手法)	反復1次元d-Spline探索法
相対誤差10%以下を推定した試行数			4096	3857
標本点数	全体	平均	112.21	225.9
		最大	263	578
		最小	45	72
	方向決定	平均	22.38	79.22
		最大	48	152
		最小	4	26
	d-Spline生成	平均	89.83	146.66
		最大	215	476
		最小	41	46
ATの探索反復回数		平均	7.03	26.76
		最大	19	48
		最小	3	16
ATを含んだAMGSの平均時間[sec]			102.2	91.5
ATの平均時間			6.46.E-02	9.67.E-04

次元 d-Spline 探索法と比較すると提案手法の方がより正確に推定できていることがわかる．また，標本点の数についても，全体で約 50%削減できている．その内訳も方向決定に用いられる標本点を約 70%削減できている．反復回数についても，平均で 26.76 回が 7.06 回で約 74%削減できており，提案手法の方がより効率よく性能パラメタ空間を探索できていることがわかる．推定の時間については，反復 1 次元 d-Spline 探索は，全体の時間が 91.5[sec]に対して，推定に使われた時間が9.67×10^{-4}[sec]であり，全体の1.06×10^{-3}である．提案手法は，全体の時間が 102.2[sec]に対して，推定に使われた時間が6.46×10^{-2}[sec]であり，全体の6.32×10^{-2}である．反復 2 次元 d-Spline 探索法は推定時間が増えているが，全体時間に対して推定に使われた時間は無視できるほど小さいことがわかる．したがって，課題に挙げていた推定時間についても問題ないことがわかる．

5.3 5 次元性能パラメタでの実験結果

5 つ性能パラメタについて，適用実験を行う．表 5-5 に性能パラメタの詳細を示す．計算機環境としては，Reedbush-U スーパーコンピュータシステム[19]を使用し，2 ノード使い，各ノードで 16 プロセス，スレッドで実行した．実験では，それぞれ性能パラメタ 32768 個を初期標本点として，32768 回試行を行った．反復 1 次元 d-Spline 探索法と提案手法を比較する．表 5-6 に実験結果を示す．

課題に挙げていた推定の正確について，どちらも正確に推定できていることがわかる．また，標本点の数についても，全体で約 35%削減できている．その内訳も方向決定に用いられる標本点を約 61%削減できている．反復回数についても，反復 1 次元 d-Spline 探索法は，平均で 25.43 回だが，提案手法は 8.46 回で約 67%削減できており，提案手法の方がより効率よく性能パラメタ空間を探索できていることがわかる．推定の時間については，反復 1 次元 d-Spline 探索は，全体の時間が 131.3[sec]に対して，推定に使われた

表 5-5 5 次元性能パラメタの詳細

性能パラメタ	Strong_Connection		Damped_Jacobi	
階層	1	2	1	2
範囲	[0.01,0.08]	[0.01,0.08]	[0.1,0.8]	[0.1,0.8]
データの間隔	0.01	0.01	0.1	0.1
パラメタの取り得る個数	8	8	8	8
パラメタの組合わせ総数	4,096(=8×8×8×8)			

表 5-6 5 次元性能パラメタにおける実験結果

			反復2次元d-Spline探索法(提案手法)	反復1次元d-Spline探索法
相対誤差10%以下を推定した試行数			32768	32768
標本点数	全体	平均	233.27	358.71
		最大	386	839
		最小	107	61
	方向決定	平均	33.46	86.28
		最大	70	202
		最小	7	26
	d-Spline生成	平均	199.81	272.43
		最大	316	637
		最小	100	35
ATの探索反復回数		平均	8.46	25.43
		最大	18	38
		最小	3	4
ATを含んだAMGSの平均時間[sec]			147.57	131.28
ATの平均時間			3.86E-02	9.32.E-03

時間が9.32×10^{-3}[sec]であり，全体の7.10×10^{-3}である．提案手法は，全体の時間が 147.6[sec]に対して，推定に使われた時間が3.86×10^{-2}[sec]であり，全体の2.62×10^{-2}である．反復 2 次元 d-Spline 探索法の推定時間は増えているが，全体時間に対して推定に使われた時間は無視できるほど小さいことがわかる．したがって，課題に挙げていた推定時間についても問題ないことがわかる．

6. まとめ

本研究ではベースとなる d-Spline を 2 次元化し，より高次元性能パラメタをカバーリングできる反復 2 次元 d-Spline 探索法を提案した．反復 2 次元 d-Spline 探索法の評価実験として，代数的マルチグリッド法(AMG 法) に適用し，4 次元と 5 次元の性能パラメタについて実際にチューニングを行い，反復 1 次元 d-Spline 探索法と比較を行った．そのとき，課題として（1）性能パラメタ推定の正確性，（2）推定にかかる時間コストの 2 つが挙げた．

結果として，推定の正確性については，最適値の推定について，提案手法は反復 1 次元 d-Spline 探索法と同じ程度推定できていた．標本点数についても，最大で約 50%削減できた．その内訳も方向決定に用いられた標本点を最大で 70%削減できた．反復回数についても最大で 74%削減できている．推定時間についても全体時間に比べて，推定に使われた時間は無視できる程度であり，課題で上げた推定時間についても問題ないことがわかる．

したがって，反復 2 次元 d-Spline 探索法は，反復 1 次元 d-Spline 探索法と同じ程度の精度で最適値を推定でき，反復

回数，標本点数を削減し，より効率よく性能パラメタ空間を探索できることを示した．

今後の課題としては，斜め方向を考慮した実装とより良い初期標本点の選択方法の考案があげられる．

謝辞

本研究の一部は JSPS 科研費 JP17K00164，JP16H02823，JP18K19782，JP18K11340 の助成を受けて行われた．

参考文献

1) J. Bilmes, K. Asanovic, C.-W. Chin and J. Demmel, Opti-Mizing Matrix Multiply using PHiPAC : a Portable, High-Performance, ANSI C Coding Methodology, in: Proceedings of the 11th international conference in Supercomputing, Vol.97, pp. 340-347 (1997).
2) R. Clint Whaley, A. Petitet and j.J. Dongarra, Automated Empirical Optimization of Software and ATLAS project, Parallel Computing, Vol. 27, pp. 3-35 (2001).
3) T. Tanaka, T. Katagiri and T. Yuda, d-Spline Based Incremental Parameter Estimation in Automatic Performance Tuning, in: Proceedings of the 8th International Conference on Applied Parallel Computing: State of the Art in Scientific Computing, LNCS, Vol. 4699, Springer, pp. 986-995 (2007).
4) T. Tanaka, R. Otsuka, A. Fujii, T. Katagiri and T. Imamura, Implementation of d-Spline-based Incremental Performance Parameter Estimation Method with ppOpen-AT, Scientific Programming 22, pp. 299-307 (2014).
5) R. Murata, J. Irie, A. Fujii, T. Tanaka and T. Katagiri, Enhancement of Incremental Performance Parameter Estimation on ppOpen-AT, in: proc. Of MCSoC2015, pp. 203-210 (2015).
6) M. Mochizuki, A. Fujii, T. Tanaka,Fast Multidimensional Performance Parameter Estimation with Multiple One-dimensional d-Spline Parameter Search, in Proc, IEEE International Parallel and Distributed Processing SymposiumWorkshop 2017 (IPDPSW2017), pp.1399-1407, 6.2.2017. Abstract One approach of software automatic
7) J. Chen, R. Chen, A. Fujii, R. Suda, and W. Wang, Timing Performance Surrogates in Auto-Tuning for Qualitative and Quantitative Factors, in SIAM Conference on Parallel Processing and Scientific Computing (PP14) (2014).
8) T. Katagiri, S. Ohshima, M. Matsumoto, Auto-tuning of Cpmputation Kernels from an FDM code with ppOpen-AT, in: Proceedings of MCSoC2014, pp. 91-98 (2014).
9) ppOpen-HPC Project, http://ppopenhpc.cc.u-tokyo.ac.jp/ppopenhpc(accessed 2019-2-2).
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, HPC 研究発表会,Vol.148-27, pp. 1-8 (2015).
11) R. Suda, A Bayesian Method for Online Code Selection: Toward Efficient and Robust Methods of Automatic Tuning, in the Second International Workshop on Automatic Performance Tuning (iWAPT2007), pp. 23-31 (2007).
12) 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).
13) 須田礼仁，自動チューニング数理基盤ライブラリ ATMathCore-Lib，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).
14) G. Fan, M. Mochizuki, T. Tanaka, A. Fujii, and T. Katagiri, D-Spline Performance Tuning Method Flexibly Responsive to Execution Time Perturbation, in: proc. of PPAM2017, pp.1-10 (2017).
15) R. Franke, A critical comparison of some methods for interpolation of scattered data, Naval Postgraduate School Tech. Rep, NPS-53-79-003, (1979).
16) 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).
17) A. Fuji, Osni MARQUES, Axis Communication Method for Algebraic Multigrid Solver, IEICE TRANSACTIONS on Information and Systems, Vol.E97-D, No.11, pp.2955-2958
18) AMGS: 代数的マルチグリッド（ ＡＭＧ ）法 のライブラリ http://hpcl.info.kogakuin.ac.jp/lab/software/amgs(accessed 2019-2-4)
19) P. Vanek, J. Mandel and M. Brezina, Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems, Technical Report UCD-CCM-036 (1995).
20) Reedbush-U スーパーコンピュータシステム, https://www.cc.u-tokyo.ac.jp/system/reedbush/(accessed 2019-2-2)