FIBER：汎用的な自動チューニング機能の付加を支援するソフトウエア構成方式

片桐孝洋[†,††] 吉瀬謙二[†,††]
本多弘樹[†] 弓場敏嗣[†]

本稿では，汎用的な自動チューニング機能の付加を支援する新しいソフトウエア構成方式である FIBER を提案する．FIBER では汎用的な自動チューニングを実現するため，パラメタ最適化を行うタイミングをインストール時，実行起動前，および実行時の 3 種に分ける．また FIBER では，ループアンローリングに代表されるコード生成とパラメタ登録とを必要とする処理について，その処理に特化した機能を持つことでユーザによるコード開発を支援する．

FIBER: A Software Framework To Support Automatically Addition of Generalized Auto-tuning Facilities

Takahiro KATAGIRI,[†,††] Kenji KISE,[†,††] Hiroki HONDA[†]
and Toshitsugu YUBA [†]

In this report, we propose a new software framework, named FIBER, which can support an addition of generalized auto-tuning facilities. To archive the generalized auto-tuning facilities, there are three kinds of optimization layers in FIBER — installation, before execution-invocation, and run-time layers. FIBER also provides a loop unrolling function, which needs code generation and parameter registration processes, to support code development by users.

1. はじめに

近年，ATLAS[1)]，FFTW[2)]，および ILIB[3)] に代表される，いわゆる「自動チューニング機能付き」の数値計算ソフトウエア（以降，自動チューニング機能付きソフトウエア，Software with Auto Tuning Facilities (SATF) とよぶ）が多数開発されるようになってきた．この理由は，(1) 利用しやすいライブラリインターフェース構築のため引数を削減する必要があるが，性能劣化が起きない機構の開発が必要である；(2) 並列計算機などの複雑な計算機システムでのチューニング作業のコストが著しく増加し，処理の自動化の要求が生じた；などの理由がある．

現在 SATF の性能評価が多くなされており，自動チューニング機能の有効性が周知のものとなりつつある[3)]．ところが SATF の汎用性という観点では，各パッケージごとにその機能が限定されている．すなわち各 SATF パッケージで採用されている自動チューニング方式は，その自動チューニング対象の処理のみしか有効でない．例えば数値計算ライブラリの場合，一般的に行列の直接解法ルーチン，反復解法ルーチン，密行列用解法ルーチン，および疎行列用解法ルーチンが含まれているが，それらのルーチン全てに適用可能な自動チューニングの汎用的枠組がない．

そこで我々は，SATF に幅広く適用できる基盤ソフトウエアの確立を目指し，(1) ライブラリインストール時に行うインストール時最適化階層 (Installation Optimization Layer, IOL)，(2) ユーザが指定するパラメタ (たとえば問題サイズなど) が固定した地点で行う実行起動前最適化階層 (Before Execution-invocation Optimization Layer, BEOL)，そして (3) 実際にライブラリが実行された地点で行う実行時最適化階層 (Run-time Optimization Layer, ROL)，の 3 種の最適化階層に分けて SATF を構成するソフトウエア構成方式を提案する．このソフトウエア構成方式のことを，FIBER (Framework of Installation, Before Execution-invocation, and Run-time optimization layers，ファイバー)[4)] とよぶ．

2. FIBER による自動チューニングの枠組

2.1 FIBER のソフトウエア構成

FIBER は，以下に示す 2 機能を有するソフトウエア構成方式である．

- コード開発支援機能：ユーザが記述したライブラリ，サブルーチン，およびプログラムの一部分に

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 科学技術振興事業団 戦略的創造研究推進事業 さきがけプログラム「情報基盤と利用環境」領域
"Information Infrastructure and Applications", PRESTO, Japan Science and Technology Corporation(JST)

専用言語などを用いて指示を与えることで，自動チューニングを保証するコードの自動生成，およびそのコードに関するパラメタ化とその登録とを自動的に行う機能

- 3 階層のパラメタ最適化機能：パラメタチューニング階層 (Parameter Tuning Layer, PTL) で行う，3 階層のパラメタ最適化機能

図 1 に FIBER のソフトウエア構成を示す．

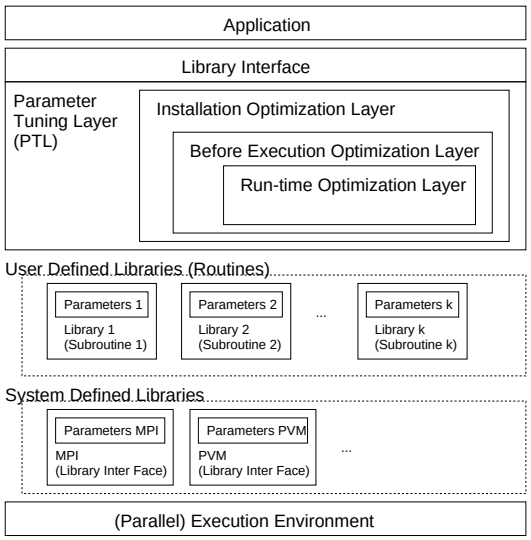

図 1　FIBER のソフトウエア構成

図 1 の FIBER のソフトウエア構成が示すとおり，MPI などの計算機環境を構成するシステムライブラリや，OS などであらかじめ用意されているライブラリについても，ライブラリインターフェースさえ周知であればパラメタに関する記述を行うことで PTL にパラメタ情報を引き渡すことができる．

2.2　パラメタ最適化のタイミング

図 2 は，PTL における 3 種の最適化階層による最適化手順について示している．図 2 から分かるように PTL には，各ライブラリやルーチンからユーザが指定したパラメタが引き渡されるが，それらは IOL に引き渡すパラメタ (IOP)，BEOL に引き渡すパラメタ (BEOP)，そして ROL に引き渡すパラメタ (ROP)，というように指定されている．このとき各最適化階層によるパラメタ最適化は，以下に示す状況，かつ順序で実行される．

(1 番目)IOL : ライブラリインストール時
(2 番目)BEOL : ユーザが指定する特定パラメタ (たとえば問題サイズなど) の指定後
(3 番目)ROL : IOL，BEOL でのパラメタ最適化が終了し，かつ対象のライブラリやルーチンの実行時

各最適化階層で最適化されたパラメタは，パラメタ情報ファイル (Parameter Information File) に保存され，より下位の最適化階層で参照される．PTL で

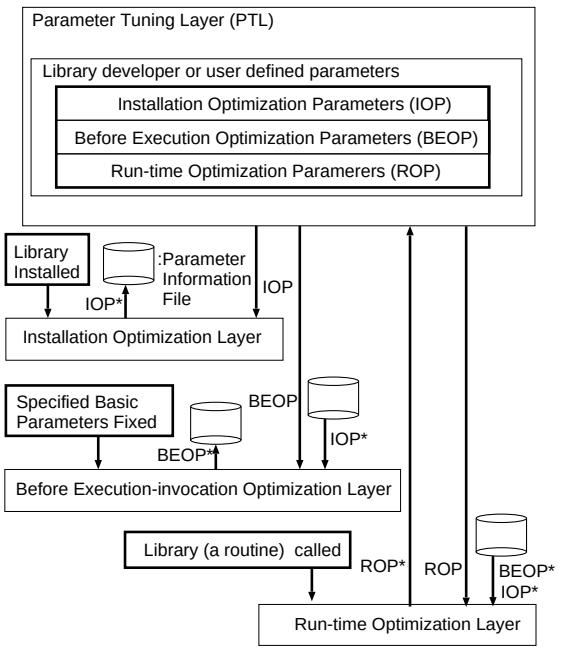

図 2　FIBER における 3 種の最適化階層の処理手順

は引き渡されたパラメタの最適化に関し概述の 3 階層を有するが，最適化されたパラメタ情報の伝達に関して図 2 のような制約がある．すなわち，IOL で最適化されたパラメタ情報は BEOL と ROL で参照可能であるが，BEOL で最適化されたパラメタ情報は ROL でしか参照できない．この主な目的は，下位層で決まるパラメタの精度を高めるためである．

2.3　自動チューニングの目的

ここでは FIBER の処理の詳細を説明する前に，自動チューニングの目的とは何かを説明する．具体的には，以下の固有値計算インタフェースをあげる．

[例 1] 従来型のライブラリインターフェース

```
call PEigVecCal( A, x, lambda, n, ...(i)
     nprocs, myid, iDistInd,      ...(ii)
imv, iud, ihit, icomm, kbi, kort, ...(iii)
           MAXITER, deps)         ...(iv)
```

従来型のライブラリでは，例 1 のようなインタフェースをとらざるを得なかった．ここで各パラメタに関し，(i) は行列情報などの基本情報パラメタ，(ii) は並列制御パラメタ，(iii) はアンローリング段数などの性能に影響する性能情報パラメタ，(iv) はアルゴリズムに影響する解法情報パラメタ，と呼ぶことにする．一般的に，(ii) 並列制御パラメタはライブラリ設計の工夫で，(iv) 解法情報パラメタは解法の数値解析をおこなうことで除去可能である．しかし自動チューニング機能のない従来型のライブラリでは，(iii) 性能情報パラメタの除去が本質的にできない．

そこで SATF の搭載により，性能を劣化させずに (iii) の性能情報パラメタの除去を行い

```
call PEigVecCal(A, x, lambda, n)
```

のようなインタフェースの実現を目指す．

2.4 自動チューニングの定義

以下定義に入るが，この定式化はほぼ直野と山本による定義[5] と同じである☆．

いまライブラリ引数全体のパラメタ集合を AP, (i) 基本情報パラメタのような引数パラメタ集合を BP，その他の引数のパラメタ集合を $OP \equiv AP/BP$ とする．すなわち

$$AP = BP \cup OP\ , \tag{1}$$

ここで $BP \cap OP = \phi$ である．

以下，パラメタ集合 AP，BP の定義をする．

[定義 1] 集合 AP (All Parameters)

ライブラリを構成するすべてのプログラムにおいて，入出力や性能に影響するパラメタすべてからなる集合をさす．□

[定義 2] 基本パラメタ集合 BP (Basic Parameters)

AP のうち (並列) 数値計算プログラムを利用する場合における，基本情報 (行列サイズなど)，および計算機環境 (プロセッサ台数や計算機構成方式など) に関連するパラメタの集合をさす．すなわち，ユーザにより実行起動前に与えられるパラメタ，およびインストールすべき計算機の構成方式により決まるパラメタの集合である．□

ここで，以下の仮定をおく．すなわち，プログラムの実行時間 t は，引数パラメタ集合 AP からの写像 F で定義できるとする．この仮定より，

$$t = F(m)\ , \tag{2}$$

である．ここで $m \subseteq AP$ である．

$PP \subseteq OP$ である集合 PP を例 1 の (iii) 性能情報パラメタ集合とする．PP は以下のように定義される．

[定義 3] 性能パラメタ集合 PP (Performance Parameters)

パラメタ集合 OP のうちで BP を固定させた場合に，ライブラリ全体の性能に大きく影響するパラメタ集合をさす．すなわちユーザにより実行起動前に与える必要はないが，ライブラリの性能を決定するパラメタ集合である．□

ここでパラメタ集合 $OPP \equiv OP/PP$ とする．すなわち，

$$OP = PP \cup OPP\ , \tag{3}$$

ここで $PP \cap OPP = \phi$ とする．

この OPP に関して，以下の仮定をおく．すなわち，パラメタ集合 OPP は，BP にのみ影響するパラメタであると仮定する．

いま，パラメタ集合 BP を固定 ($\bar{l} \subseteq BP$) する．仮定より BP を固定すると，ライブラリ全体の実行時間 t は，

$$t = F(\bar{l}, g) \tag{4}$$

となる．ここで $g \subseteq PP$ である．

以上のことから，自動チューニングとは以下のように定義される．

[定義 4] 自動チューニング

自動チューニングとは，パラメタ集合 AP のうち，パラメタ集合 BP を固定した場合において，実行時間 t に関する関数 F を最小化する最適化問題を解く処理である．

すなわち $\bar{l} \subseteq BP$ という与えられた制約のもと，以下の関数 F を最小化する $g \subseteq PP$ となる解集合 g を求める処理である：

$$\min_g F(\bar{l}, g) = \min_g \bar{F}(g)\ .\quad \square \tag{5}$$

ここで関数 F は，上記の定義ではライブラリの実行時間を定義する関数とした．しかし一般的なパラメタチューニングは，ライブラリ実行時間の最小化だけする処理ではない．たとえば，メモリ量や計算機利用料金などの要因も考慮したい状況がありうる．このことから関数 F は，最適化問題のコストを定義する関数であると拡張定義し，以降関数 F をコスト 定義関数とよぶ．

2.5 FIBER における自動チューニングの定義

FIBER では，性能パラメタ集合 PP を以下の 3 つに分ける．すなわち，

$$PP = IOP \cup BEOP \cup ROP\ . \tag{6}$$

ここで IOP，$BEOP$，および ROP は，それぞれ，インストール時，実行起動前，および実行時のタイミングで最適化される PP である．このとき，FIBER の自動チューニングは以下のように定義される．

[定義 5] FIBER の自動チューニング

FIBER の自動チューニングとは，パラメタ集合 AP のうち，パラメタ集合 BP の一部を固定した場合において，インストール時，実行起動前，および実行時の 3 種のタイミングでパラメタ PP の推定を行う処理のことである．具体的には，ユーザにより指定された各タイミングでの最適化処理対象ごとに定義されるコスト定義関数 F について，各タイミングごとに定まるパラメタ集合 BP を固定した上で，関数 F を最小化するパラメタ PP を求める処理である．□

定義 5 から FIBER のインストール時最適化とは，インストールされる計算機環境が確定した状況で定まる一部の BP を固定した上で，PP を推定する処理といえる．また実行起動前最適化は，処理対象の知識を利用してユーザが定めた BP を固定した上で，PP を推定する処理といえる．ゆえに実行起動前最適化のほうが PP の推定精度が高い．この 2 階層で定まらない BP について，実行時に確定する BP を固定した上で PP を推定する処理が，実行時最適化といえる．

☆ 直野と山本による定義[5] では，性能に影響するパラメタを CP (Critical Parameter) とし，CP のうちライブラリインタフェースに現れるパラメタを UCP (Users' Critical Parameter)，ライブラリインタフェースに現れないパラメタを ICP(Internal Critical Parameter) と定義している．

3. FIBER による自動チューニング適用例

3.1 ユーザによる性能パラメタ指定方法

FIBER では，ユーザがソースプログラム中に性能パラメタ PP とチューニングの範囲 (チューニング領域とよぶ) の指定をする．以降例をあげて説明する．

3.1.1 指 定 形 式

ソースプログラムにおいて，!ABCLib$で始まる行は，FIBER による自動チューニングのための指示行とみなす．この表記法の概略を図 3 に示す．

```
!ABCLib$ ⟨ 自動チューニング種類 ⟩ ⟨ 機能名 ⟩
        [ (対象変数) ]  region  start
[ !ABCLib$ ⟨ 機能の詳細 ⟩[ sub region start ]]
     チューニング領域
[ !ABCLib$ ⟨ 機能の詳細 ⟩[ sub region end ]]
!ABCLib$ ⟨ 自動チューニング種類 ⟩ ⟨ 機能名 ⟩
        [ (対象変数) ]  region  end
```

図 3　FIBER における自動チューニング指示形式

図 3 における自動チューニングの種類や機能名のことを指定子とよぶ．指定子である ⟨ 自動チューニング種類 ⟩ には，インストール時 (install)，実行起動前 (static)，および実行時 (dynamic) の指定をする．また指定子である ⟨ 機能名 ⟩ には，コード処理機能や自動チューニング方式の詳細を指定する．

以降，典型的な機能名に関する指定子である，アンローリング指定子 (unroll)，および選択指定子 (select) について指定例を示す．

3.1.2 アンローリング指定子の例

以下に，アンローリング指定子の例を示す．

```
!ABCLib$ install unroll (j) region start
!ABCLib$ varied (j) from 1 to 16
!ABCLib$ fitting polynomial 5
!ABCLib$     sampled (1-4,8,16)
 do j=0, local_length_y-1
    tmpu1 = u_x(j)
    tmpr1 = mu * tmpu1 - y_k(j)
    do i=0, local_length_x-1
       A(i_x+i, i_y+j) = A(i_x+i, i_y+j)
          + u_y(i)*tmpr1 - x_k(i)*tmpu1
    enddo enddo
!ABCLib$ install unroll (j) region end
```

上記の例では，region start ~ region end で囲まれたチューニング領域について，j ループをアンローリングするコードを自動生成し，その段数をパラメタ化して PP とする．またインストール時最適化をする．

各指定子の機能の詳細を指定する指定子のことを補助指定子とよぶ．定義域については，varied 補助指定子で定義されており，{1,..,16}である．コスト定義関数について，fitting 補助指定子でその種類を指定できる．ここでは 5 次の線形多項式を指定している．また標本点については，sampled 補助指定子で指定でき，ここでは{1-4,8,16}である．

3.1.3 選択指定子の例

以下に，選択指定子の例を示す．

```
!ABCLib$ static select region start
!ABCLib$ parameter (in CacheS, in NB, in NPrc)
!ABCLib$   select sub region start
!ABCLib$   according estimated
!ABCLib$    (2.0d0*CacheS*NB) / (3.0d0*NPrc)
             対象処理 1
!ABCLib$   select sub region end
!ABCLib$   select sub region start
!ABCLib$   according estimated
!ABCLib$    (4.0d0*CacheS*dlog(NB))/(2.0d0*NPrc)
             対象処理 2
!ABCLib$   select sub region end
!ABCLib$ static select region end
```

上記の例では，sub region start ~ sub region end で囲まれた複数のチューニング領域に対して，according estimated 補助指定子で指定される数式の値が小さい処理を選択する．その数式中で指定される変数は，parameter 補助指定子で指定されており，in 補助指定子で入力であることを指定する．この例では実行起動前最適化を指定しているので，これらの変数はインストール時最適化を指定したチューニング領域中で parameter 補助指定子により指定した変数において，out 補助指定子を用いてパラメタ保存ファイルに出力しておく必要がある．

この例では，対象処理 1 が選択されるか，対象処理 2 が選択されるか，という値をパラメタ化して PP とする点に注意する．

3.2 インストール時最適化層 (IOL) の予備実験

ここでは FIBER による自動チューニングの適用例として，実数対称密行列における多くの固有値，固有ベクトルを求める場合に用いられる Householder-二分-逆反復法のライブラリに FIBER のインストール時最適化を適用した場合について予備評価する．

Householder-二分-逆反復法のライブラリインターフェースは，例 1 の PEigVecCal のようになっているとする．この主要なパラメタは以下のようになる．

- $PP \equiv$ { imv, iud, icomm, ihit, kbi, kort }
- $BP \equiv$ { n, nprocs }

本実験での PP，すなわち IOP，を iud とする．この定義域と処理は，以下の通りであるとする．なおコード，およびパラメタの定義域などは，3.1.2 節のアンローリング指定子の例と同一である．

- iud $\equiv$ { 1,2,...,16 } : Householder 三重対角化で必要な行列更新処理 (2 重ループ, BLAS2) のうち，最外ループアンローリング段数を指定．

なおここでは実行時間に関し最適化するので，コスト定義関数は実行時間で定義する．また計算機は，東

京大学情報基盤センタの HITACHI SR8000/MPP を利用する☆

3.2.1 パラメタiud のインストール時最適化方法

いまインストール対象の計算機環境は，プロセッサ台数が 8 台とする．またパラメタ集合 BP を固定するためのパラメタiud に関しての標本点は { 1,2,3,4,8,16 }，問題サイズn に関しての標本点は，{ 200, 400, 800, 2000, 4000, 8000 } とする．HITACHI SR8000/MPP による，これらの標本点の測定結果を表 **1** にのせる．

表 **1** HITACHI SR8000/MPP 8PE による実測結果 [秒]

n\iud	1	2	3	4	8	16
200	.0628	.0628	.0629	.0623	.0621	.0625
400	.1817	.1784	.1763	.1745	.1723	.1719
800	.7379	.6896	.6638	.6550	.6369	.6309
2000	7.535	6.741	6.333	6.240	6.013	5.846
4000	54.06	48.05	44.85	44.36	42.89	41.19
8000	413.2	366.5	349.2	344.1	327.6	315.5

本実験では，まず基本情報パラメタであるn を固定して，パラメタiud に関する関数 $f_n(iud)$ を推定する．ここで $f_n(iud)$ は k 次の線形多項式 $f_n(iud) = a_1 \cdot iud^k + a_2 \cdot iud^{k-1} + a_3 \cdot iud^{k-2} + \cdots + a_k \cdot iud + a_{k+1}$ であると仮定すると，適当な最適化手法により係数 $a_1, .., a_{k+1}$ を決定できる．ここでは最小二乗法を選択したとする．ここで最小二乗法の解法には，Householder 法による QR 分解を用いて正規方程式を解く方法を用いる．

表 1 の標本点を用いて k 次の線形多項式によるコスト定義関数の係数を決定し得た関数の値☆☆が最も小さい推定パラメタによる実行時間と，定義域すべてについて実測して得た最適パラメタによる実行時間との相対誤差を各問題サイズn の標本点ごとに求め，その相対誤差を相加平均した値を表 **2** にのせる．

表 **2** 線形多項式によるコスト定義関数を用いた推定パラメタの実行時間と最適パラメタの実行時間との相対誤差の相加平均値 (HITACHI SR8000/MPP 8PE)

0 次	1 次	2 次	3 次	4 次	5 次
19.2	0.51	1.73	0.87	0.82	0.23

表 2 から，この例では 5 次線形多項式の場合が最も相加平均値が小さく，よりよい推定方式であるといえる．そこで 5 次の線形多項式をコスト定義関数に採用するとする．いま表 **3** に，表 1 のデータを標本点として最小二乗法により 5 次の線形多項式の係数を決定した結果をのせる．

表 **3** コスト定義関数の推定結果 (問題サイズ固定)

f_n	a_1	a_2	a_3	a_4	a_5	a_6
f_{200}	-2.2e-6	6.7e-5	-6.5e-4	2.4e-3	-3.8e-3	6.4e-2
f_{400}	-1.3e-6	4.2e-5	-4.6e-4	2.3e-3	-7.8e-3	1.8e-1
f_{800}	9.6e-6	-2.3e-4	7.9e-4	1.1e-2	-8.4e-2	8.1e-1
f_{2000}	2.4e-4	-6.3e-3	3.5e-2	1.1e-1	-1.3e-0	8.6e-0
f_{4000}	3.0e-3	-8.3e-2	6.1e-1	-4.9e-1	-7.7e+0	6.1e+1
f_{8000}	-1.3e-2	5.0e-1	-7.0e+0	4.5e+1	-1.4e+2	5.1e+2

表 3 から，iud の定義域 { 1,2,...,16 } で最小となるiud の値を，問題サイズn に関する標本点上で決定できる．表 3 を係数にもつ関数 $f_n(iud)$ と，観測点の概略を図 **4** に示す．

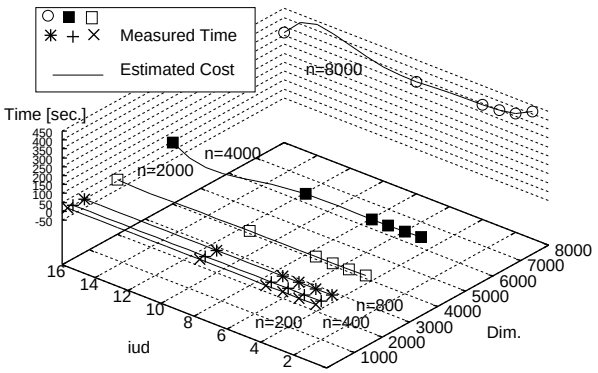

図 **4** 最適パラメタiud の推定 (問題サイズ標本点で固定)

ここで問題サイズn に関する標本点と同じ値を，実行時にユーザが指定する保証はない．したがって図 4 方式による推定値が必ず利用できるわけではない．標本点以外の値を実行時にユーザが指定する場合を想定し，FIBER では以下の方式にてパラメタ推定をする．

まず，表 3 により各標本点上でiud の全定義域について推定値を得ることができる．すなわちiud 定義域 { 1,2,...,16 } 全てにおいて，問題サイズn の標本点 { 200, 400, 800, 2000, 4000, 8000 } だけ変化させた推定値を計算できる．そこでこれらの推定値を新たな標本点とみなし，関数 $f_{iud}(n)$ が最小二乗法により決定できる．いま $f_{iud}(n)$ も 5 次多項式 $f_{iud}(n) = \bar{a_1} \cdot n^5 + \bar{a_2} \cdot n^4 + \bar{a_3} \cdot n^3 + \bar{a_4} \cdot n^2 + \bar{a_5} \cdot n + \bar{a_6}$ と仮定して，関数 $f_{iud}(n)(iud = 1, ..., 16)$ を推定した結果の概略を図 **5** に示す．

図 5 から，n の関数 $f_{iud}(n)$ がiud に関する全定義域 { 1,2,...,16 } で定まったので，実行時にユーザが指定したn を代入することで，関数値が最小となる推定パラメタiud の値を決定できる．これが FIBER による IOL のパラメタ最適化方式である．

3.2.2 推定値と最適値との比較

上記の方法で，ユーザが実行時にn = 123, 1234, 9012 の値を指定する場合を評価した．その結果，推定したiud とその実行時間 (秒) は，それぞれ，14 (0.0333

☆ HITACHI SR8000/MPP の各ノードは，8PE を有し，理論性能は 14.4GFLOPS，メモリは 16 GB である．通信網のトポロジは 3 次元ハイパークロスバ網であり，最大通信性能は片方向で 1.6 Gbytes/s，双方向で 3.2Gbytes/s である． 本実験では，HITACHI Optimized Fortran90 V01-04 コンパイラが使われている．コンパイラオプションは *-opt=4 -parallel=0* である．通信ライブラリは，日立製作所が提供している MPI (Message Passing Interface) を用いている．

☆☆ 0 次から 5 次まで．ここで 0 次の多項式とは，最も値の小さなパラメタによる値を常に返す関数とする．たとえばこの例の場合は，iud が 1 のときの測定時間を返す．

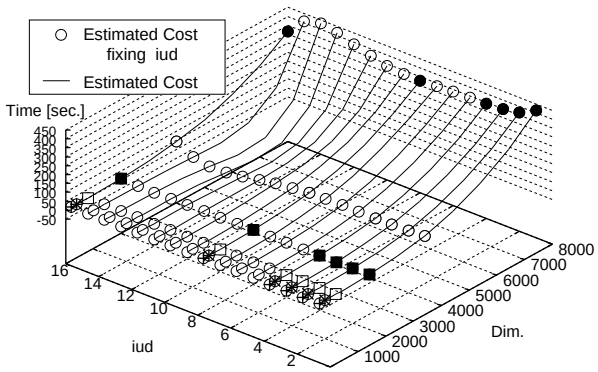

図 5 最適パラメタiud の推定 (問題サイズ不定)

秒)，14 (1.668 秒)，および 16 (440.6 秒) であった．iud の全定義域を測定し得た最適値は，それぞれ，6 (0.0333 秒)，16 (1.662 秒)，および 16 (440.6 秒) であった．したがってこの例では，推定値と最適値との実行時間の差はほとんどなかった．

4. 関 連 研 究

パラメタ自動チューニングに関する方式は，以下の 2 種に分類できる．まず計算機システムソフトウエアにおける実行時最適化層の枠組がある．たとえば，パラメタ (IO バッファサイズなど) の決定を実行時に行うソフトウエアとして Active Harmony[6]，Autopilot[7] がある．次に数値計算ライブラリにおけるインストール時最適化層の枠組がある．例えば，PHiPAC[8]，ATLAS[1] および経験的手法を用いた枠組の AEOS[9]，FFTW[2]，である．また ILIB[3] では，さらに実行時最適化層を付加している．しかしいずれも FIBER 方式のように，処理の汎用性やパラメタの推定精度を高める目的で，(1) 実行起動前最適化層，および (2) 3 階層に分け最適化済パラメタを参照する枠組，をもつ方式はない．ゆえにこの 2 点が独創的な点である．また自動チューニングの定式化では，SIMPL[5] においてインストール時最適化層の定式化を行った．

5. お わ り に

本論文では，インストール時，実行起動前，および実行時の最適化階層を有する自動チューニングソフトウエア構成方式である FIBER を提案した．FIBER の枠組では各階層におけるコスト定義関数 F を，最適化したいライブラリ，サブルーチン，およびプログラムの一部分の特性に応じて，どのように定義するのかが大きな問題となる．今後の課題として，コスト定義関数の情報をユーザ指定にさせるのか，コードの特性を参照した上で何らかの高度な関数推定法を導入しシステム側が自動推定するのか，などコスト定義関数決定法について議論と評価とを行う必要がある．

現在，FIBER 機能の一部を実装した並列固有値ソルバのプロトタイプ ABCLibDRSSED を開発中である．そのα 版は WWW サーバ上 (http://www.abc-lib.org/) でソースコードとマニュアルを含めて公開されている．また 3.1 節で示した自動チューニングのためのコード処理，パラメタ指定，およびその登録を支援するプリプロセッサ ABCLibScript[10] も開発中である．

謝辞： FIBER について有益なコメントや議論をして頂いた，東京大学大学院情報理工学系研究科 須田礼仁 助教授に感謝いたします．本研究は，科学技術振興事業団 戦略的創造研究推進事業 さきがけプログラムの助成による．

参 考 文 献

1) ATLAS project; http://www.netlib.org/atlas/index.html.
2) Frigo, M.: A Fast Fourier Transform Compiler, *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, Atlanta, Georgia, pp. 169–180 (1999).
3) 片桐孝洋, 黒田久泰, 大澤清, 工藤誠, 金田康正: 自動チューニング機構が並列数値計算ライブラリに及ぼす効果, 情報処理学会論文誌 : ハイパフォーマンスコンピューティングシステム, Vol.42, No.SIG 12 (HPS 4), pp. 60–76 (2001).
4) 片桐孝洋: プログラム、記録媒体およびコンピュータ, 日本国特許出願, 特願 2003-022792 (平成 15 年 1 月 30 日).
5) 直野健, 山本有作: 単一メモリ型インターフェイスを有する自動チューニング並列ライブラリの構成方法, 情報処理学会研究報告, No. 2001-HPC-87, pp. 25–30 (2001).
6) Tapus, C., Chung, I.-H. and Hollingsworth, J. K.: Active Harmony : Towards Automated Performance Tuning, *Proceedings of High Performance Networking and Computing (SC2002)*, Baltimore, USA (2003).
7) Ribler, R. L., Simitci, H. and Reed, D. A.: The Autopilot Performance-Directed Adaptive Control System, *Future Generation Computer Systems, special issue (Performance Data Mining)*, Vol. 18, No. 1, pp. 175–187 (2001).
8) Bilmes, J., Asanović, K., Chin, C.-W. and Demmel, J.: Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, *Proceedings of International Conference on Supercomputing 97*, pp. 340–347 (1997).
9) Whaley, R., Petitet, A. and Dongarra, J. J.: Automated Empirical Optimizations of Software and the ATLAS project, *Parallel Computing*, Vol. 27, pp. 3–35 (2001).
10) 片桐孝洋: 計算装置、計算方法、プログラムおよび記録媒体, 日本国特許出願, 特願 2003-092592 (平成 15 年 3 月 28 日).