数理解析研究所講究録
第1733巻 2011年 40-46

疎行列反復解法ライブラリにおける自動チューニング機能の開発

Development of Auto-tuning Facility for Sparse Matrix Iterative Libraries

東京大学・情報基盤センター　片桐　孝洋(Takahiro Katagiri)
Information Technology Center,
The University of Tokyo

1. はじめに

数値計算ライブラリの性能自動チューニング（以降，ATと記載）技術は，密行列やFFTライブラリで成功を収めてきた．これは，階層キャッシュに代表されるハードウェアの複雑化と，数値計算ライブラリ特有の複雑なチューニング技術による開発コスト増大という工学的な背景がある．

近年，計算機ハードウェアの複雑化はさらに進み，数十コアに及ぶマルチコア化や多階層キャッシュ化が進んでいる．さらに，GPU（Graphics Processing Unit）などの演算アクセレータとマルチコアCPUの混載型が登場し，非均質（ヘテロジニアス）環境が普通となりつつある．このような状況の中，疎行列ライブラリの実行時ATの機能開発を目的に，AT機能付き数値計算ライブラリXabclib[1]を研究開発してきた．

本講演では，ATの概要と定式化に始まり，既存研究の紹介と問題点，および，ATソフトウェアの一つの実現例として平成22年開発の疎行列反復解法ライブラリXabclibβ版について紹介する．

2. 自動チューニング機能とは

2.1 AT機構と構成要素

図1は本研究が想定するAT機構である．図1のAT機構が対象とするものは，（1）計算機ハードウェアやシステムソフトウェア（OSなど）；（2）プログラムやアルゴリズム上に現れる性能に影響を及ぼすパラメタ；が対象となる．

AT機構を構成する要素は，（1）調整機構；（2）性能モニタ機構；（3）つまみ自動生成機構；（4）性能蓄積機構；である．ここで，つまみ自動生成機構とは，任意のプログラムにAT機構が利用できるようにする機構であり，多くの場合はソースコード変換ツール（プリプロセッサ）で実現される．

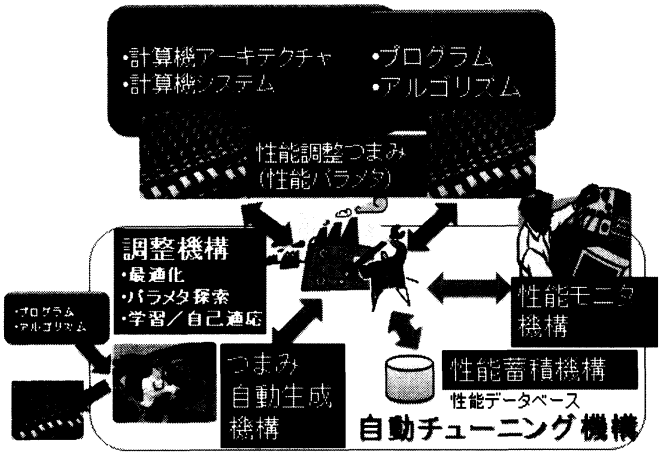

図1　AT機構の概要

AT機構のうち，もっとも数理的な基盤を必要とする要素は，調整機構である．パラメタの最適化，探索，学習，自己適応を担当する．

AT機構の適用対象は複数のコンパイラを含む．コンパイラ最適化とAT機構の差異は，コンパイラ最適化は，ある特定計算機を強く意識して最適化をするのに対し，AT機構は複数のコンパイラを含む幅広い計算機環境に対し，汎用的に適用できる最適化技術である点が異なる．

2.2 定式化

文献[2]では，ソフトウェア自動チューニング（以降AT）を以下の図 ように定義している．

図2のi～vの処理すべてが自動化されていることが望ましい．しかし，一部にユーザが介在する半自動化のATも，自動チューニングと呼ぶ．

i. プログラム上の対象個所を決定する.
ii. 対象個所中において，性能に影響を及ぼすパラメタ集合 Xを抽出する.
iii. チューニングする評価基準となる関数 Fを パラメタ集合 Xで定義する.
iv. 関数$F: X \rightarrow Y$ を，何らかの基準で最小とするパラメタ集合 Xの値を，プログラムを実行しながら見つける.（これを、解パラメタ集合 X*とする）
v. 解パラメタ集合 X*でプログラムを実行する.

図 2 ATの定義

2.3 パラメタ集合とユーザ知識

プログラム上の i番目の対象領域の性能パラメタ変数を x_i とする. 変数 x_i の値のとりうる範囲を，以下とする.

$$x_i \in [IS_{x_i}, IE_{x_i}]. \quad \ldots(1)$$

式(1)のとりうる範囲$[IS_{x_i}, IE_{x_i}]$を，パラメタ x_i の定義域と呼ぶことにする. 一般に変数 x_i の取りうる値は整数であることが多いが，実数になる場合もある.

すべてのパラメタ x_i の値 $\bar{x}_i$ を元として構成される集合 X

$$X = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m\} \quad \ldots(2)$$

は，プログラム上の複数の対象個所のパラメタについて，なんらかの操作で設定した値の集合である. すなわち，

$$\bar{x}_i = f(x_i), i = 1,2,\ldots,m, \quad \ldots(3)$$

で関数 fは，AT 方式による操作を意味する. 通常，定義域を縮小する操作（探索枝刈り操作）が入る.

パラメタ変数間，たとえば x_i, x_j などは，プログラムの実行速度などのチューニング対象において，影響するかもしれないし，影響しないかもしれない. また影響する場合，どのパラメタ変数間で影響するか，いくつのパラメタ変数に影響を及ぼすか，はプログラム上の対象個所の位置に依存する. ユーザは，どのパラメタ変数について影響するか知っており，事前情報として与えることができる.

2.4 コスト定義関数と探索時間

パラメタ集合 Xを入力とし，集合 Zを出力とする関数 Fを定義する. この関数 Fを**コスト定義関数**と呼ぶ. コスト定義関数は，以下になる.

$$F: X \rightarrow Z. \quad \ldots(4)$$

また，パラメタ集合 X を，最適化時に固定される集合 BP (Basic Parameter，基本パラメタ)と，最適化の際に定義域の範囲内で変動させ，その都度コスト定義関数を評価するパラメタ変数の値の集合 PP (Performance Parameter，性能パラメタ) の2つに分けて表記する流儀もある. すなわち，

$$X = BP \cup PP, BP \cap PP = \phi. \quad \ldots(5)$$

式(5)から，AT とは条件付き最適化問題

$$\min F(PP)\ s.t.\ BP = \{\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_n\} \quad \ldots(6)$$

である. ここで，$\bar{p}_i \in \Re$ である.

コスト定義関数を定義する者は，ユーザである場合もあるし，計算機システムであることもある. 一般に，実行速度をコスト定義関数として与えられることが多いが，メモリ量，演算精度，または電力量が与えられることもある.

コスト定義関数の特徴が数理解析できるのであれば，最適化時間の短縮につながり有効である. しかし一般には自明ではない. 連続関数として扱うと特異点があり，確率的に扱うと解の品質が悪くなる場合がある.

式(6)の最適性を工学上意味がある程度に保証し(たとえば，最適値より（多くは経験的だが）10%品質が劣化するなど)，集合 PP の元に関する変数 x_i の定義域を縮小させるような条件 Pを見つけること，すなわち

$$L = \{l_i \in [IS_{x_i}, IE_{x_i}] \mid P\}(i = 1,\ldots,m), L \subset PP \quad \ldots(7)$$

は意味がある. 数値計算でよく現れる演算を対象にし（たとえば，BLAS の DGEMM 関数)，条件 Pを与えることが経験的技能（職人芸）である.

一般的にコスト定義関数の特徴や条件 P がうまく抽出できないので，パラメタ集合 Xの直積集合 $X \times X$を関数 Fの定義域として関数 Fを調べる方法（全探索）が良く用いられる.

2.5 自動チューニングのタイミング

AT を行うタイミングは一つではない. FIBER[3] では，インストール時，実行起動前時，実行時の3通りを定めている. パラメタ集合 Xも，AT タイミングで3通りが定められている. それぞれのパラメタ集合を，*IP*, *BEP*, *RP*で記載すると

$$X = IP \cup BEP \cup RP \quad \ldots(8)$$

となる. ここで，*IP*, *BEP*, *RP*に重複して入るプログラム上の対象個所があってもよいが，パラメタ変数は別となる. AT 適用の時間的順序は決まっており，①インストール時；②実行起動前時；③実行時；であり，順番が逆転することはない. 事前 AT で最適化したパラメタ集合は，事後の AT で集合

BPとして固定される.

3. 関連研究

汎用的なAT手法の開発を目的に，数理的な研究が行われているものは少ない．多くは，パラメタの定義域を縮小する条件Pを定めるものであり，以下に例を挙げる.

- 測定（試行）回数を少なくするためBays法を利用する方法（須田[4]）
- 密行列ソルバの実行時間予測を，多項式近似（主に線形3次）により推定する方式（片桐ら[5]）
- 非連続な実行時間（実測データ）の形状から，d-Spline近似により，適切なパラメタ変数の定義域の自動設定を行う方法（田中ら[6]）
- 密行列ソルバの小規模サイズの測定データをもとに，動的計画法で大規模サイズの実行時間を予測する方式（深谷ら[7]）

ここでは，著者が直接関連した2つについて概要を説明する.

3.1 FIBER方式[5]

FIBER方式は，以下の特徴をもつパラメタ推定方式である.

1. インストール時，実行起動前時，実行時の3つの自動チューニングのタイミングで最適化する.
2. 実行前に，基本パラメタおよび性能パラメタの定義域内から抽出した値（**標本点**とよぶ）を作る.
3. 標本点について，線形多項式を用いて定義域すべてについて性能を予測する．予測タイミングは以下の2種である.
   (3a) 性能パラメタの全定義域の予測
   (3b) (3a)の予測値を新たな標本点とした場合の基本パラメタの全定義域の予測

性能パラメタx_iの定義域（たとえば，アンローリング段数）を整数とし，[1, 2,..., m]とする．定義域内の標本点として，[$s_1, s_2, \ldots, s_n$] をとる.

【操作(I)】性能パラメタx_iに対する実行時間の関数を，q次多項式と仮定する．すなわち，

$$g_{x_1}(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_q x^q . \quad \ldots(9)$$

ここで，標本点[$s_1, s_2, \ldots, s_n$]に対応する実行時間を対象を実行し測定する．これを，$\{\bar{t}_{s_1}, \bar{t}_{s_2}, \ldots, \bar{t}_{s_n}\}$とする．すなわち，$g_{x_1}(s_1) = \bar{t}_{s_1}$である．このとき式(9)の係数を，各標本点の実測時間$\{\bar{t}_{s_1}, \bar{t}_{s_2}, \ldots, \bar{t}_{s_n}\}$をもとに，最小二乗法で決定することができる.

係数決定後，式(9)の関数が定まるので，標本点の実測値$\{\bar{t}_{s_1}, \bar{t}_{s_2}, \ldots, \bar{t}_{s_n}\}$に加え，標本点に含まれないパラメタの実行時間を式(9)から計算（予測）できる．すなわち性能パラメタx_iの定義域全てについて，実測値もしくは推定実行時間が定まる．全定義域の実行時間集合$\{\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_m\}$が定まる.

【操作(II)】 基本パラメタ（通常は行列サイズNなので変数 Nとする）に対する標本点を取る．いま，[$N_1, N_2, \ldots, N_o$]とする．このo個の標本点（つまり問題サイズを固定したもの）それぞれについて，操作(I)の処理を行う．すなわち，o 個の関数$g_{x_i 1}, g_{x_i 2}, \ldots, g_{x_i o}$を，測定実行時間を用いて最小二乗法で係数を定める.

【操作(III)】操作(II)の実行により，定義域に対する$o \times m$個の，標本点の測定時間と推定時間による実行時間の集合を得ることができる．すなわち，

$$\{\{\bar{t}_{11}, \bar{t}_{12}, \ldots, \bar{t}_{1m}\}, \{\bar{t}_{21}, \bar{t}_{22}, \ldots, \bar{t}_{2m}\}, \ldots, \{\bar{t}_{o1}, \bar{t}_{o2}, \ldots, \bar{t}_{om}\}\} \quad \ldots(10)$$

ここで，式(10)の実行時間集合を，基本パラメタの新たな標本点とする.

【操作(IV)】性能パラメタx_iの全定義域 [1, 2, ..., m] に対応する，基本パラメタ Nの実行時間の予測式をq次多項式とする．すなわち，

$$\begin{aligned} h_1(N) &= a_{10} + a_{11}N + a_{12}N^2 + \ldots + a_{1q}N^q, \\ h_2(N) &= a_{20} + a_{21}N + a_{22}N^2 + \ldots + a_{2q}N^q, \\ &\ldots \\ h_m(N) &= a_{m0} + a_{m1}N + a_{m2}N^2 + \ldots + a_{mq}N^q. \end{aligned} \quad \ldots(11)$$

このとき，操作(III)で得られた標本点により，最小二乗法で式(11)のすべての係数を定めることができる．たとえば，$h_i(N)$の係数を定めるためには，標本点$\{\bar{t}_{i1}, \bar{t}_{i2}, \ldots, \bar{t}_{im}\}$を利用する.

【操作(V)】以上により，実行時にユーザが与えた任意の基本パラメタ Nの値$\bar{N}$について，式(11)の実行時間予測により最小となるパラメタ値を推定することができる．これが，推定による最適なパラメタ値となる．すなわち，

$$\min(h_1(\bar{N}), h_2(\bar{N}), \ldots, h_m(\bar{N})) \quad \ldots(12)$$

となる性能パラメタx_iの値を推定する.

FIBER方式を密対称行列の標準固有値問題ライ

ブラリに適用したところ，インストール時の最適化では最適解による実行に対し 0.6%~6.7%の性能劣化で済む場合がある．ただし，行列が小さい場合でキャッシュの影響を受けやすい計算機では319%もの性能劣化を生じる場合があった．これは，実行時間の変動が激しい問題サイズや計算機環境ではFIBER 方式では対応できないことを示唆している．

3.2 d-Spline 近似による動的な標本点追加法[6]

FIBER 方式では，実行時間の変動が激しい場合，少ない標本点で実行時間を近似できないことが問題であった．実用となる実行時間近似のためには，以下の性質をもつことが望ましい．

1. 変動が激しいデータに追従できるコスト関数の推定法であること．
2. 動的に標本点を追加できること．また，標本点の追加が，低い計算量でできること．

上記 2 点を満たすため，[6]では実行時間の推定関数に d-Spline を採用した．定義域$[1,2,..,m]$のパラメタ値による実行時間を$f_1, f_2,..., f_m$とする．このとき，滑らかさを以下で定義する：

$$\left|f_{j-1}-2f_j+f_{j+1}\right|, 2\le j\le k-1. \quad ...(13)$$

実行時間に関するコスト定義関数Fを，以下の最小化で求める．

$$\min_f \left(\|y-Ef\|^2+\alpha^2\|Df\|^2\right), \quad ...(14)$$

ここで，

$$E=\begin{pmatrix} 1 & & & & \\ & 1 & & 0 & \\ & & 01 & & \\ & & & 001 & \\ 0 & & & & \cdots \\ & & & & & 01 \end{pmatrix} \quad D=\begin{pmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & 0 \\ & & 1 & -2 & 1 \\ & & & \cdots & \\ 0 & & & 1 & -2 & 1 \end{pmatrix}$$

であり，y は，標本点による実行時間のベクトル $y=(y_1,y_2,...,y_n)^T$ である．

上記の(14)式は，以下の最小化問題

$$\min_f \left(\|b-Zf\|^2\right), \quad ...(15)$$

を求解することで解ける．ここで，

$$Z=\begin{pmatrix} E^TE \\ \alpha D \end{pmatrix}, b=\begin{pmatrix} E^Ty \\ 0 \end{pmatrix}$$

である．普通は式(15)中の Zを Givens 回転による QR 法で上三角化することで，最小二乗解を求めることができる．計算コストはZが帯行列であることを考慮すると $O(m)$で，m は定義域の要素数(通常、数十～数百）となるので計算量は少ない．

QR 分解は，行列$Q^TZ=R, Q^Tb=b^*$であり，Rは上三角行列（帯行列）となる．新たな標本点を追加する場合，標本点の定義域での対応位置情報を Rの最終行に追加し，b^*に新たな標本点での実行時間を最終行に追加し，追加部分のみに QR 法を適用するだけで式(15)の最小二乗問題が解ける．このコストは $O(1)$である．それゆえ，動的に標本点が追加できるほど計算量が少ない．

[6]の方法は，以下で構成される．

1. 定義域について，初期値として均等な間隔で標本点 4 点を選ぶ．
2. d-Spline で実行時間近似する．
3. 以下の 2 基準のどちらかで，新たな標本点を選ぶ．
   (3a) d-Spline で推定した最小値となる点．
   (3b) 2 階の微分係数
   $$\max_j \left|f_{j-1}-2f_j+f_{j+1}\right|, 2\le j\le k-1$$
   が最大となる点．（変動が激しい点）
4. 3 で選ばれた標本点が，過去に c 回選ばれたら終了．そうでなければ，標本点に追加し，2 へ戻る．

上記の方法を固有値ソルバで性能評価した．3 次多項式，5 次多項式，3 次スプラインによる近似法を比べた結果，d-Spline による近似法が最も良い予測精度を示した．また，疎行列 - ベクトル積演算への適用も行っている[6]．

4. 疎行列反復解法ライブラリへの適用

4.1 疎行列反復解法への AT 適用の特徴

密行列解法に対し，疎行列反復解法に AT を適用する場合，インストール時 AT は利用できるものの，実行時 AT がより重要となる．理由は，数値計算ライブラリレベルでは，疎行列形状情報や数理特性が実行時まで把握できないことによる．実行時 AT が効果を奏す機構を実現すること，すなわち，計算量が少ない方式を採用しなくてはならない．

4.2 必要な AT 機能

必要な AT 機能は，数値計算ライブラリ中の階層で異なる．まず，メタ・ソルバ階層機能は，アプリケーションプログラムに最も近い機能群である．外

部ソルバ階層機能は，メタ・ソルバ階層中で使われている機能群である．最後に内部ソルバ階層機能は，外部ソルバ階層で使われている機能群である．

- **メタ・ソルバ階層**

1. エンドユーザによる最適化方針の指定．たとえば，(i)実行時間，(ii)メモリ量，(iii)利用する CPU 数などの計算機資源，(iv)演算精度，のうちどれを選ぶか，もしくは複数の方針を設定した場合はその比重の指定．
2. AT を行うタイミングおよび頻度の指定．
3. エンドユーザによるヒント．たとえば，疎行列形状情報の指定．
4. 反復解法の選択．たとえば，GMRES(m)法や IDR(s)法などの数値解法の指定．

- **外部ソルバ階層**

1. 高性能な疎行列 BLAS の実装選択．特に，疎行列 - ベクトル積（SpMxV）の実装．アンローリング段数．7 点差分など，特定の疎行列形状に特化した実装のためのデータ構造への変換．
2. 前処理方式の選択．たとえば，ILU や SPAI の選択．前処理に付属するパラメタも対象となる．たとえば，フィルインの深さ，零とみなす許容値．
3. 数値解法上の基本演算の選択．たとえば，Gram-Schmidt 直交化の実装方式．

- **内部ソルバ階層**

1. 解法に特有のパラメタの設定．たとえば，リスタート周期の値．
2. 疎行列データ構造に依存する実装方式．たとえば，CRS 形式用か JDS 形式用か．

4.3 実現例：Xabclib と OpenATLib

OpenATLib は，数値計算ライブラリで必要となる汎用的な AT 機能を API（Application Programming Interface）化し，その参照実装の提供を目的に開発された．現在，疎行列反復解法のうち，Krylov 部分空間法による解法に特化した AT 機能を実現している．

OpenATLib の AT 方式は実行時全探索である．反復解法の特徴を利用し，反復の開始時に数回の試行を行い，その後の反復では試行情報をもとに実装を固定する（たとえば SpMxV 部分）．解法特有のパラメタのチューニングは，反復ごとに AT を行う（たとえば，リスタート周期）．

平成 22 年に公開された OpenATLib β 版は，以下の新機能がある．

- SpMxV：*OpenATI_D{S/U}RMV*

1. 非零要素均等化方式：非零要素数を考慮し，並列実行時の計算負荷をバランス化させる方式（対称・非対称行列用の双方）
2. 作業行列の零要素について，並列実行時の加算を省く方式（対称行列用のみ）
3. 新規開発方式の Branchless Segmented Scan(BSS)方式（非対称行列用のみ）

- Gram-Schmidt 直交化：*OpenATI_DAFGS*

1. 古典 Gram-Schmidt (CGS)
2. Daniel-Gragg-Kaufman-Stewart 型の Gramm-Schmidt (DGKS)
3. 修正 Gram-Schmidt (MGS)
4. ブロック化古典 Gram-Schmidt (BCGS)

OpenATLib β 版の性能パラメタ・基本パラメタと定義域を記述すると以下になる．

$$
\begin{aligned}
&PP = RP = \{DSRMV, DURMV, DAFGS, DAFRT\}\\
&BP = \{N\}\\
&DSRMV \in \{S1, S2, S3\}\\
&DURMV \in \{U1, U2, U3, U4\}\\
&DAFGS \in \{CGS, DGKS, MGS, BCGS\},\\
&DAFRT \in [1, MAXM]
\end{aligned}
$$

OpenATLib β 版におけるコスト定義関数 F は，エンドユーザが定めるポリシーにより，関数が異なる．それを以下に示す。

$$
\begin{aligned}
&F_T(PP) = \text{実測時間}\\
&F_M(PP) = \text{利用メモリ量}\\
&F_A(PP) = \{\text{実測時間} \mid \text{精度} < \text{要求精度} \}\\
&F(PP) = \{F_T, F_M, F_A \mid \text{ユーザの要求} \}\\
&\min F(PP)\ s.t.\ BP = \{\overline{N}\}
\end{aligned}
$$

以上のポリシーは**数値計算ポリシー**と呼ばれ，エンドユーザにより定義される．数値計算ポリシーは，OpenATLib の以下の関数により実装されている．

- 数値計算ポリシーのメタ・インターフェース：*OpenATI_LINEARSOLVE* と *OpenATI_EIGENSOLVE*

Xabclib は，数値計算ポリシーが実装された OpenATLib β 版を利用し，疎行列に対する対称標準固有値問題の解法であるリスタート付き Lanczos 法，および，連立一次方程式の解法である

GMRES(m)法を実装して，数値計算ライブラリ化したものである．すなわち Xabclib とは，OpenATLib を利用して開発された AT 機能付き数値計算ライブラリの一実装とみなすことができる．

4.4 自動チューニングの効果

OpenATLib による AT 効果，特にβ版で新たに実装された疎行列 - ベクトル積の AT 効果を示すため，非零要素均等化方式と BSS 方式が実装されていないα版 OpenATLib を利用し，実行速度ポリシーの効果を比較した．測定行列はフロリダ行列から対称行列 21 種，非対称行列 22 種を，Xabclib が収束するものから選んだ．

計算機は，T2K オープンスパコンの 1 ノード(16 コア，4 ソケットの AMD Quad Core Opteron 8356 (2.3GHz))を利用している．なお，OpenATLib は OpenMP で並列化されている．コンパイラは Intel Fortran Compiler Professional Version 11.0，コンパイラオプションは -O3 -m64 -openmp -mcmodel=medium である．

実行速度ポリシーを指定し，要求精度 1.0e-8 を設定した．固有値問題については，絶対値の大きい固有値から 10 個について，固有値，固有ベクトルの計算をしている．

図 3 に連立一次方程式の求解インターフェース OpenATI_LINEARSOLVE，図 4 に標準固有値問題求解インターフェース OpenATI_EIGENSOLVE の AT 効果を示す．

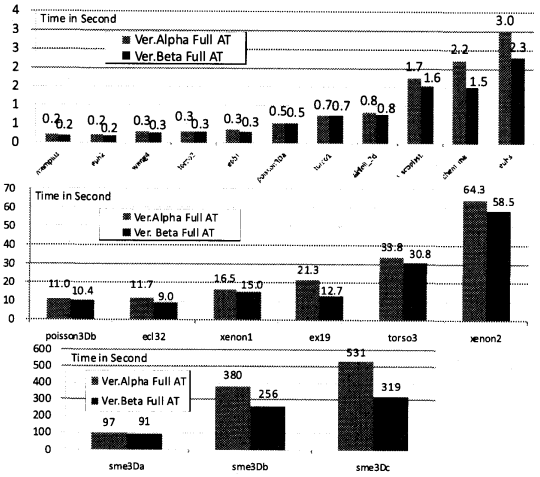

図 3 OpenATI_LINEARSOLVE の AT 効果

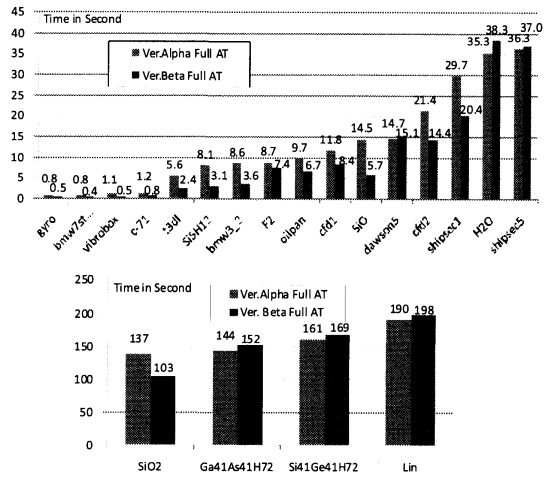

図 4 OpenATI_EIGENSOLVE の AT 効果

図 3 から，連立一次方程式の求解（非対称行列）では，22 種類の速度向上は平均 1.2 倍で，最大の速度向上は 1.6 倍（行列名：sme3Dc）であった．図 4 から，固有値問題の求解（対称行列）では，21 種類の速度向上は平均 1.6 倍で，最大の速度向上は 2.6 倍(行列名：Si5H12)であった．

以上より，従来のα版に対し，β版ではさらなる速度向上が得られる AT 方式といえる．特に対称行列の固有値問題の求解で AT 効果が高いのは，α版では疎行列 - ベクトル積の AT 機構で，作業行列の零要素について並列実行時の加算を省く方式が無く，この方式の効果が大きいことを示唆している．

5. おわりに

本原稿では，数値計算ライブラリにおける自動チューニング(AT)の概略と，疎行列反復解法へ AT 機構を実装する場合の特徴を説明した．

より汎用的な AT 機構を構築するための問題が山積している．特に，パラメタ調整機構における最適化方式に数理を適用し，探索空間の縮小を行うことが必須である．AT の研究分野に，多くの数理研究者の参入を期待する．

Xabclib は LGPL ライセンスのフリーソフトウェアとして，PC クラスタコンソーシアムから配布されている[8]．

謝辞: Xabclib プロジェクトの諸氏，東京大学情報基盤センターの大島聡史助教，伊藤祥司特任准教授，黒田久泰准教授（兼任，愛媛大学），中島研吾

教授，および日立製作所中央研究所の櫻井隆雄氏，猪貝光祥氏，直野健博士に感謝いたします．また，d-Spline による AT 方式の資料の提供について，日立製作所の田中輝雄博士に感謝いたします．

参 考 文 献

[1] 櫻井隆雄，直野健，片桐孝洋，中島研吾，黒田久泰，"OpenATLib： 数値計算ライブラリ向け自動チューニングインターフェース"，情報処理学会論文誌：ACS, Vol.3, No.2, pp.39-47, 2010.

[2] 片桐孝洋，ソフトウエア自動チューニング—数値計算ソフトウエアへの適用とその可能性—，慧文社，2004.

[3] T., Katagiri, K., Kise, H., Honda and T., Yuba, FIBER: A General Framework for Auto-Tuning Software, The Fifth International Symposium on High Performance Computing (ISHPC-V), Springer LNCS 2858, pp.146-159, 2003.

[4] 須田礼二，ソフトウェア自動チューニングの数理，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, No.6, pp.487-493, June 2009.

[5] T., Katagiri, K., Kise, H., Honda, and T., Yuba, ABCLib_DRSSED: A Parallel Eigensolver with an Auto-tuning Facility, Parallel Computing, Vol.32, Issue 3, pp.231-250, March 2006.

[6] 田中輝雄，片桐孝洋，弓場敏嗣，ソフトウェア自動チューニングにおける標本点逐次追加型性能パラメタ推定法の疎行列計算への適用，情報処理学会論文誌：コンピューティングシステム，Vol.48, No. SIG13 (ACS 19)，pp. 223-234, 2007.

[7] 深谷猛，山本有作，張紹良，ブロックハウスホルダーQR 分解の並列計算における自動チューニング手法の検討，情報処理学会研究報告，Vol.2009-HPC-121，No.18, 2009.

[8] PC クラスタコンソーシアム HP. http://www.pccluster.org/