情報処理学会論文誌　コンピューティングシステム　Vol. 3　No. 2　39–47 (June 2010)

OpenATLib：数値計算ライブラリ向け自動チューニングインタフェース

櫻井隆雄[†1]　直野健[†1]　片桐孝洋[†2]
中島研吾[†2]　黒田久泰[†2,†3]

低コストで高性能なソフトウェアを開発するために，自動チューニングの実装の再利用性（RIAT）が重要となる．本稿では数値計算ライブラリ上で RIAT を実現するために OpenATLib という名の自動チューニングインタフェースを提案する．OpenATLib は実行時におけるリスタート周期と疎行列ベクトル積の最適化を行う機能を数値計算ライブラリに提供する．その有効性を確認するため，OpenATLib を利用して疎行列反復解法ライブラリ Xabclib_LANCZOS と Xabclib_GMRES を開発した．T2K オープンスパコン（東大版）1 ノード（16 コア）上で様々なフロリダ大学の行列を用いて OpenATLib の自動チューニング機能を評価した結果，Xabclib_LANCZOS は最大で 22.4 倍，Xabclib_GMRES は最大 3.5 倍の速度向上を達成した．

OpenATLib: A General Auto-tuning Interface for Numerical Solvers

Takao Sakurai,[†1] Ken Naono,[†1]
Takahiro Katagiri,[†2] Kengo Nakajima[†2]
and Hisayasu Kuroda[†2,†3]

Reusability for Implementation of Automatic Tuning facility (RIAT) is needed to develop low-cost construction of high performance software. In this paper, we propose an auto-tuning interface named OpenATLib to realize RIAT for numerical libraries. OpenATLib provides to numerical libraries with restart frequency adjustment and sparse matrix-vector multiplication auto-tuning functions. To evaluate the effectiveness of OpenATLib, we have developed sparse iterative solvers named Xabclib_LANCZOS and Xabclib_GMRES with OpenATLib. Performance evaluation of OpenATLib using several U. Florida matrices with the T2K Open Supercomputer (Todai Combined Cluster) on 1 node (16CPU) indicated that the maximum speedup established 22.4x (Xabclib_LANCZOS) and 3.5x (Xabclib_GMRES).

1. はじめに

科学技術計算などに利用される数値計算ライブラリでは，一部の入力パラメータの値や積算などの重要な処理におけるアルゴリズムの選択によりその演算時間が大幅に変化する．一方で，短い時間で演算できる良好なパラメータやアルゴリズムは入力行列によって異なり，かつ事前予測が難しいものが存在する．そのため，数値計算ライブラリのユーザが良好な条件でライブラリを利用することが困難となっていた．

これらの問題を解決する手法として，入力パラメータの決定やアルゴリズム選択の支援をする自動チューニング方式（Auto-tuning method，AT 方式）が注目されている．これまでに様々な研究機関から多くの AT 方式が提案され，それらを備えた数値計算ライブラリが開発されてきた．たとえば PHiPAC[1)]，ATLAS[2)]，FFTW[3)]，I-LIB[4)]，ABCLib[5)]，OSKI[6)] などである．これらにより数値計算ライブラリのユーザがパラメータやアルゴリズムの選択に悩まずに良好な条件でライブラリを利用できるようになってきた．

しかし，既存のライブラリでは対応していない特殊な行列を処理する場合や，いまだライブラリ化されていない新しい解法を使うために数値計算ライブラリを自作する場合も増えている．これらの自作ライブラリに対し，その作成者が AT 方式を適用する場合，AT 方式によって解法が複雑化し，プログラム記述のためにコード量が増加する場合がある．これらの問題のため，作成者が多数の AT 方式を自作ライブラリに適用するには非常に多くの労力を必要とする．そこで，これまで個別に提案されてきた自動チューニングの実装に関して再利用性（RIAT: Reusability for Implementation of Automatic Tuning facility）を高める手段が必要となる．

この課題に対し，筆者らは AT 方式の共通部分を関数として提供する AT インタフェース OpenATLib を提案する．それと同時に OpenATLib を利用した疎行列ライブラリを作成し，その性能を評価することで提案方式の有効性を検証する．

†1 株式会社日立製作所中央研究所
Central Research Laboratory, Hitachi, Ltd.
†2 東京大学情報基盤センタースーパーコンピューティング研究部門
Supercomputing Research Division, Information Technology Center, The University of Tokyo
†3 愛媛大学大学院理工学研究科
Graduate School of Science and Engineering, Ehime University

2. AT インタフェース OpenATLib とそれを用いた疎行列ライブラリ

筆者らは，RIAT の実現のため，Fortran で記述された疎行列反復法ライブラリ向けの汎用 AT インタフェース OpenATLib を提案・開発している[7]．OpenATLib は API（Application Programming Interface）を静的ライブラリの形式で提供し，主にライブラリ作成者が自作ライブラリに AT 方式を実装するために利用する．OpenATLib が提供する AT 方式を用いることでライブラリは疎行列反復解法において実行前に予測が困難なパラメータやアルゴリズムの最適値を実行中に探索し，1 回の演算で安定して高い演算性能を提供することが可能となる．本章では OpenATLib とそれを用いて開発した疎行列ライブラリについて述べる．

2.1 AT インタフェース OpenATLib の概要

一般にライブラリ作成者は数値計算ライブラリの作成時には，工数を削減するために一部の演算量の多いルーチン（行列–ベクトル積，直交化など）を対象のハードウェア構成向けに最適化された BLAS や Lapack といったサブルーチン群を用いることが多い．そのため，課題を解決するシステムにおいても BLAS などと同様にサブルーチン群の形式で提供すればライブラリ作成者が違和感なく使用できると考えられる．

そこで，AT 方式の解法に依存しない共通部分を切り出し，それらを Lapack などと同様に関数の形で提供するサブルーチン群とそれら特有のパラメータを管理する機構を提供することで，容易に AT 方式を自作ライブラリに組み込むことが可能となる．

このサブルーチン群を自動チューニングインタフェース「OpenATLib」と名づけ，その基本構成を**図 1** に示した．本インタフェースは，①パラメータ自動選択機能，②アルゴリズム自動選択機能，③自動チューニングパラメータ管理機能，から構成される．本インタフェースを構成する各機能を以下で説明する．

①パラメータ自動選択機能は，数値計算ライブラリの実行時にリスタート周期のようなパラメータの最適値を推定する AT 方式を実装している．ライブラリ作成者が所望の AT 方式を自作ライブラリに実装する場合に関数として提供する．

②アルゴリズム自動選択機能は，行列–ベクトル積や直交化処理といったサブルーチンにおいてそれぞれ複数のアルゴリズムを実装している．そして，入力行列やライブラリが使用可能なメモリ量に応じて最適なアルゴリズムを選択する AT 方式を実装している．ライブラリ作成者が所望の AT 方式を適用したサブルーチンを自作ライブラリに実装する場合にそれを関数として提供する．

③自動チューニングパラメータ管理機能は，AT 方式により新たに生じるパラメータ（AT パラメータ）のデフォルト値を設定し，大域変数として管理する．普段はデフォルト値で動作するが，AT 方式に知識のあるライブラリ作成者が AT パラメータの変更を望む場合は大域変数にアクセスすることでデフォルト値からの変更を可能とする．

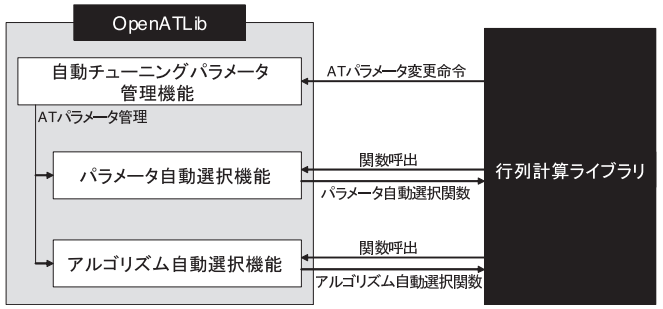

図 1　OpenATLib の基本構成
Fig. 1　The architecture of OpenATLib.

2.2 OpenATLib が備える AT 方式

OpenATLib はパラメータ自動選択機能として Krylov 部分空間法のリスタート周期が小さいかを判断する OpenATI_DAFRT を備えており，アルゴリズム自動選択機能として疎行列–ベクトル積の最適実装方式を判断する OpenATI_DSRMV，OpenATI_DURMV を備えている．以下で各機能の詳細を述べる．

2.2.1 OpenATI_DAFRT

疎行列解法として代表的な Krylov 部分空間法は計算機上で実行する場合，使用するメモリを実行前に確定させるため Krylov 部分空間の次元数を一定数に制限し，得られた最新の近似解を新たな初期値としてリスタートさせて新たな Krylov 部分空間を生成する．この Krylov 部分空間の最大次元数をリスタート周期と呼ぶ．

最適なリスタート周期の推定は実行前には困難であるが，実行中であれば残差の履歴を監視し，相対残差の停滞を検出することで推定が可能となる．

相対残差の停滞を示す指標として収束判定 $s-t$ 回目から s 回目の残差の中での最大値を最小値で割った値を「残差 Max-Min 比」とし，以下では「MM 比」[8] と表記する．s 回目の収束判定時の i 番目の相対残差 r_i に対する過去 t 回分の MM 比 $R_i(s,t)$ を式で表したものが以下である．

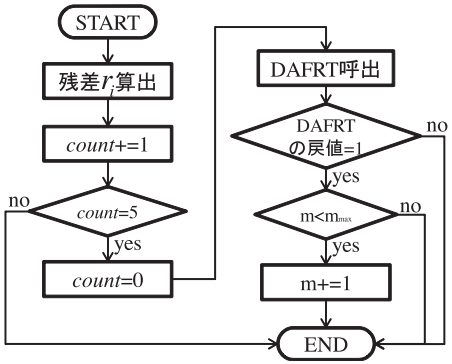

図 2　OpenATI_DAFRT を利用したリスタート周期最適化方式
Fig. 2　The method of restart frequency auto-tuning using OpenATI_DAFRT.

$$R_i(s,t) = \frac{\max_z\{r_i(z) : s-t+1, ..., s\}}{\min_z\{r_i(z) : s-t+1, ..., s\}}$$

リスタート周期が十分に大きいときは残差が大きく減少するため MM 比は大きくなり，小さいときは残差の減少が停滞し MM 比は小さくなる．

OpenATI_DAFRT は入力引数として残差履歴を受け取り，その MM 比を算出する．その結果，MM 比が閾値（デフォルト値は 100，ユーザの指示により変更可能）以下であれば出力として 1 を返し，閾値より大きければ 0 を返す．パラメータ t はユーザが与えるが 3〜5 程度で良い結果が得られることが多い．

ライブラリ作成者は収束判定のための残差算出後に OpenATI_DAFRT を呼び出し，その戻値が 1 であったときにリスタート周期を大きくするように記述することで最適なリスタート周期が探索できる．たとえば 5 反復に 1 回リスタート周期 M の大きさを判定し，周期の最大値 $m_{\max}$ を超えない範囲で 1 ずつ大きくする場合，相対残差 r_i 算出時に**図 2** に示したフローに沿った処理を加えればよい．

2.2.2　OpenATI_DSRMV，OpenATI_DURMV

疎行列–ベクトル積は反復解法において多くの回数が実行され，その演算時間の合計は解法の中で大きな割合を占める．疎行列–ベクトル積のアルゴリズムは様々なものが存在するが，実行する計算機環境（特に並列数）や疎行列の内部形状などにより適切なアルゴリズムは変化する．そのため，一概にどのアルゴリズムが最適のものであるとは決め難い．そのため，実行時に環境や行列に応じて最適なアルゴリズムを選択する方式が必要となる[9]．

OpenATI_DSRMV，OpenATI_DURMV は解法開始直後の最初の疎行列–ベクトル積の実行時に複数の実装方式を順番に実行し，演算時間が最も短かった方式で以後の疎行列–ベクトル積を実行することで最適アルゴリズムの選択を実現している．DSRMV が上三角部分のみを Compressed Row Storage（CRS）形式で格納した対称行列を対象としており，DURMV が CRS 形式の非対称行列を対象としている．

以下に示すように _ のそれぞれで 3 種のアルゴリズムを用意している．

（1）　**OpenATI_DSRMV（対称疎行列用）**
S1) 上三角，下三角ループ分割方式：上三角部分のみ格納された対称行列の行列ベクトル積において，上三角部分の処理と下三角部分の処理を別のループで実行し，後で合算する方式．基本的に低速だが，並列数が大きくなったときに有効な場合がある．
S2) ループ融合方式：上三角の処理と，下三角の処理をマージし，1 つの 2 重ループで実行する方式．3 方式の中では多く使われ，安定して高い性能が得られる．
S3) ループ融合＋リダクション並列方式：S2 の上に並列実行がしやすいように OpenMP の動的作業領域を割り当てた方式．並列数が増えた際に特に有効となる場合が多いが，並列数が少ないと性能が得られない．

（2）　**OpenATI_DURMV（非対称疎行列用）**
U1) ループがコンパイラによりベクトル化されるようディレクティブ[⋆1]を挿入した方式
U2) 行列ベクトル積の 2 重ループにおいて，外側ループが 8 段，内側ループが 2 段のアンローリングになるよう明示的に UNROLL のディレクティブを挿入した方式
U3) ループがコンパイラによりベクトル化されないようディレクティブを挿入した方式

ライブラリ作成者は BLAS2（行列–ベクトル積の標準ライブラリ）と同様に行列–ベクトル積の実行タイミングで OpenATI_DSRMV，OpenATI_DURMV を呼び出せば最適な行列–ベクトル積実装方式が選択される．

2.3　OpenATLib を用いた疎行列ライブラリ Xabclib

OpenATLib を実装した疎行列ライブラリとして，標準固有値問題を求解する LANCZOS 法[10] ライブラリ Xabclib_LANCZOS，および連立一次方程式を求解する GMRES 法[11] ラ

⋆1 OpenATI_DURMV のディレクティブは Intel Fortran Compiler Version 11.0 に対応するものを挿入している．

イブラリ Xabclib_GMRES を開発した[7]．

Xabclib_LANCZOS，Xabclib_GMRES はともに OpenATLib が提供するリスタート周期最適化機能 OpenATI_DAFRT，行列–ベクトル積最適化機能 OpenATI_DSRMV，OpenATI_DURMV を実装している．また，Xabclib_GMRES は ILU(0)，Jacobi，SSOR の 3 種類の前処理を備えている．対応する疎行列圧縮形式は，ともに CRS 形式である．

3. 性 能 評 価

3.1 性能評価環境とテスト行列

開発した OpenATLib および Xabclib_LANCZOS，Xabclib_GMRES の有効性を確認するためにこれらの性能評価を実施した．本評価は T2K オープンスパコン（東大版）1 ノードを利用した．**表 1** に計算機およびコンパイル環境を示した．テスト行列はフロリダ大学の Sparse Matrix Collection [12] よりそれぞれ 20 個ずつ選んだ．Xabclib_LANCZOS 用の行列を**表 2**，Xabclib_GMRES 用の行列を**表 3** に示した．表 2，表 3 では分野ごとに行列をまとめ，さらに次元数の少ない順に並べた．

その他の解法固有の条件として，Xabclib_LANCZOS において求める固有値の個数は 10，Xabclib_GMRES では解ベクトルは全要素が 1，初期近似解ベクトルは全要素 0 とし，前処理は ILU(0) を用いた．

3.2 リスタート周期自動チューニングの評価結果

最初にリスタート周期自動チューニング（OpenATI_DAFRT）の効果を評価した．Xabclib_LANCZOS において求める固有値の数を 10 とし，リスタート周期を 30 に固定して解いた場合と，OpenATLib のリスタート周期自動チューニング機能を用いて解いた場合の演算時間を比較した．自動チューニングにおいてリスタート周期 M の初期値を 10，最大値を 150 とし，1 反復ごとに過去 5 回分の残差から MM 比を求め，閾値 100 を下回ったときに M が小さいと判断し 5 ずつ増加させた．ここで，リスタート周期の固定値を 30 とした理由は，PETSc [13] など代表的な数値計算ライブラリにおいてデフォルト値とされていることによる．

評価の結果を**表 4** に示した．ここでは表 3 に示した 20 個の行列の中で代表的な傾向を示した 5 つの行列の結果を示している．表中の “M” はリスタート周期，“restart” はリスタート回数を表す．また，“last M” は自動チューニングによりリスタート周期を探索したときに収束した際の最終的な値を表す．また，“自動チューニング有” における時間は自動チューニングによる処理の時間を含んでいる．これは以後の表でも同様である．bmw3_2，c-71 は，リスタート回数が比較的少なく，自動チューニングにより最適なリスタート周期を判別する前に終了するため，リスタート周期を固定した方が短い時間で収束した．一方，リスタート周期を固定したときに演算終了まで 1,000 回以上の多くのリスタート回数を必要とする t3dl や dawson5 は非常に高い速度向上が得られている．また，Ga41As41H72 ではリスタート周期を 30 に固定した場合では収束しなかったが，自動チューニングを用いた場合は収束した．

表 1　計算機およびコンパイルの環境
Table 1　Computer and compiler configuration.

CPU	Quad-Core AMD Opteron(tm) 8356 2.3 GHz 16core/1node
L2 サイズ	2 MByte/4core
Memory	32 GByte（8 GByte/1Socket）
OS	Red Hat Enterprise Linux 5
コンパイラ	Intel Fortran Compiler Version 11.0
コンパイルオプション	-O3 -m64 -openmp -mcmodel=medium

表 2　Xabclib_LANCZOS 用テスト行列
Table 2　Test matrices for Xabclib_LANCZOS.

Matrix	N	NNZ	Field
vibrobox	12,328	177,578	acoustics
Lin	256,000	1,011,200	chemistry
cfd1	70,656	949,510	fluid dynamics
cfd2	123,440	1,605,669	fluid dynamics
gyro	17,361	519,260	model reduction
t3dl	20,360	265,113	model reduction
c-71	76,638	468,096	optimization
Si5H12	19,896	379,247	structural
SiO	33,401	675,528	structural
dawson5	51,537	531,157	structural
H2O	67,024	1,141,880	structural
F2	71,505	2,682,895	structural
oilpan	73,752	1,835,470	structural
shipsec1	140,874	3,977,139	structural
bmw7st_1	141,347	3,740,507	structural
SiO2	155,331	5,719,417	structural
shipsec5	179,860	5,146,478	structural
Si41Ge41H72	185,639	7,598,452	structural
bmw3_2	227,362	5,757,996	structural
Ga41As41H72	268,096	9,378,286	structural

表 3　Xabclib_GMRES 用テスト行列
Table 3　Test matrices for Xabclib_GMRES.

Matrix	N	NNZ	Field
chem_master1	40,401	201,201	2D/3D
torso2	115,967	1,033,473	2D/3D
torso1	116,158	8,516,500	2D/3D
torso3	259,156	4,429,042	2D/3D
memplus	17,758	126,150	electric circuit
ex19	12,005	259,879	fluid dynamics
poisson3Da	13,514	352,762	fluid dynamics
airfoil_2d	14,214	259,688	fluid dynamics
poisson3Db	85,623	2,374,949	fluid dynamics
viscoplastic2	32,769	381,326	materials
xenon1	48,600	1,181,120	materials
xenon2	157,464	3,866,688	materials
wang4	26,068	177,196	semiconductor device
ecl32	51,993	380,415	semiconductor device
sme3Da	12,504	874,887	structural
sme3Db	29,067	2,081,063	structural
sme3Dc	42,930	3,148,656	structural
epb1	14,734	95,053	thermal
epb2	25,228	175,027	thermal
epb3	84,617	463,625	thermal

表 4　Xabclib_LANCZOS によるリスタート周期自動チューニングの効果
Table 4　Effectiveness of restart frequency auto-tuning by Xabclib_LANCZOS.

行列	自動チューニングなし			自動チューニングあり			自動チューニング
	M	restart	時間（秒）	last M	restart	時間（秒）	による速度向上
t3dl	30	1,878	125.62	90	33	6.69	18.79
c-71	30	4	0.70	25	17	1.38	0.51
dawson5	30	1,052	135.75	105	34	14.79	9.18
bmw3_2	30	3	4.20	25	19	12.93	0.33
Ga41As41H72	30	10,000	未収束	150	44	204.05	未収束 → 収束

続いて，Xabclib_GMRES において，リスタート周期を同じく 30 に固定した場合とリスタート周期自動チューニング機能を用いて解いた場合の演算時間を比較した．自動チューニングにおいてリスタート周期の初期値を 2，最大値を 100 とし，1 反復ごとに過去 5 回分の残差から MM 比を求め，閾値 100 を下回ったときに M を 5 ずつ増加させた．その結果を**表 5** に示した．Xabclib_LANCZOS と同様にリスタート回数の少ない poisson3Da，airfoil_2d

表 5　Xabclib_GMRES によるリスタート周期自動チューニングの効果
Table 5　Effectiveness of restart frequency auto-tuning by Xabclib_GMRES.

行列	自動チューニングなし			自動チューニングあり			自動チューニング
	M	restart	時間（秒）	last M	restart	時間（秒）	による速度向上
torso1	30	1	2.54	2	1	0.72	3.53
poisson3Da	30	3	0.48	17	7	0.54	0.89
airfoil_2d	30	7	0.73	22	14	0.83	0.88
sme3Da	30	670	215.49	100	90	101.33	2.13
sme3Db	30	1,000	未収束	100	120	377.29	未収束 → 収束

ではリスタート周期を固定した方が演算時間が短かく，収束するまでに必要なリスタート回数の多い sme3Da やリスタート周期 2 のときに 1 反復で収束する torso1 では性能が大きく向上していた．また，リスタート周期を固定した場合に未収束の sme3Db が収束するようになった．

以上から，リスタート周期の自動チューニング機能は必ずしも速度向上をもたらすわけではないが，演算時間の大幅な増加や未収束を防げると考えられる．

3.3　行列–ベクトル積自動チューニングの評価結果

続いて，行列–ベクトル積の自動チューニング（OpenATI_DSRMV，OpenATI_DURMV）の効果を評価した．本評価では複数の行列に対して並列数を 1，4，8，16 と変化させて，OpenATI_DSRMV，OpenATI_DURMV に実装されている S1〜S3，U1〜U3 の各方式の性能を測定した．本測定は同条件で 10 回動作させたときの平均値を測定値とした．

図 3 に OpenATI_DSRMV の 3 種の実装方式の演算性能を示した．評価に使用した行列は dawson5，F2，Ga41As41H72 の 3 つである．図 3 の結果では S2（ループ融合方式）は 1SMP のときは 3 つの行列すべてで最も性能が良いが，並列数を増やしたときの加速率が悪く，4SMP 以上では S3（ループ融合＋リダクション並列方式）の方が 2 倍近くの高い性能が得られていた．しかし，dawson3 のように 16SMP 時に S3 が S2 の性能を 10%程度下回っている場合もあり，このような場合に S2 を選択できるため，OpenATI_DSRMV の行列–ベクトル積実行方式選択は有効に働くと分かった．一方，S1（上三角，下三角ループ分割方式）は 12 パターンのすべてにおいて最も性能が悪く，より有意な実装方式に入れ替える必要があると分かった．

OpenATI_DURMV の 3 種の実装方式の演算性能を**図 4** に示した．この評価に使用した行列は memplus，ecl32，sme3Da の 3 つである．図 4 の結果では ecl32，sme3Da の 2 つはすべての並列数において U3（ループがコンパイラによりベクトル化されないよう記述し

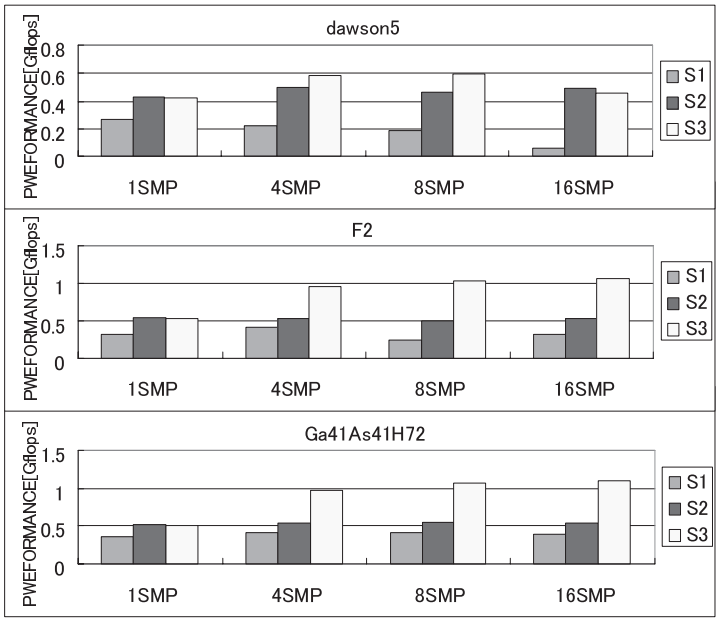

図 3　OpenATI_DSRMV の 3 方式の性能
Fig. 3　Comparison of performance of 3 methods provided by OpenATI_DSRMV.

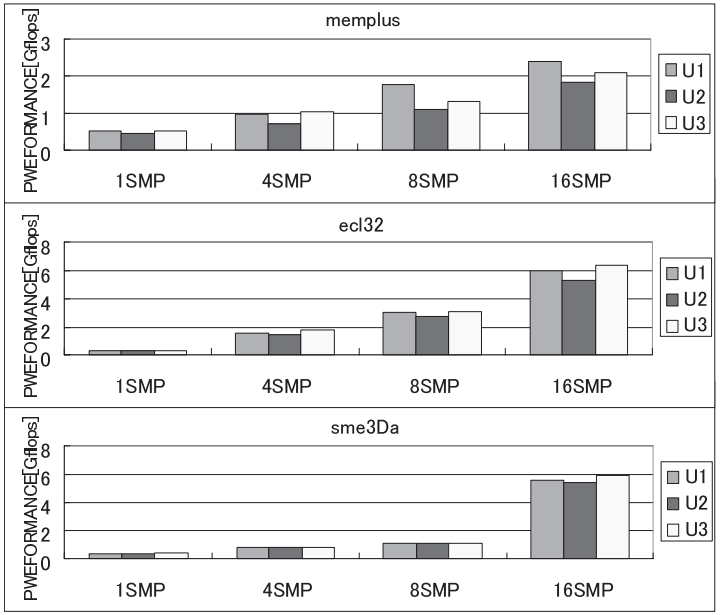

図 4　OpenATI_DURMV の 3 方式の性能
Fig. 4　Comparison of performance of 3 methods provided by OpenATI_DURMV.

た方式）で高い性能が得られた．一方，memplus はすべての並列数で U1（ループがコンパイラによりベクトル化されるよう記述した方式）が U3 より 10%程度高い性能が得られた．このように非対称行列では行列によって有効な実装方式が異なるため，実行時に最適な実装方式を探索する自動チューニングが有効であると考えられる．

以上から，行列–ベクトル積の最適実装方式を探索する自動チューニングには有効であるといえる．しかし，現状は S1 のように選択されない実装方式も含まれており，より有意な実装方式を検討する必要があることも分かった．

3.4　2 方式の自動チューニングを合わせた評価結果

最後に Xabclib_LANCZOS，Xabclib_GMRES の 2 つのライブラリにおいて OpenATI_DAFRT，OpenATI_DSRMV，OpenATI_DURMV を同時に動作させたときに，それらを使用しなかった場合と比べてどの程度の速度向上効果が得られるか評価した．

評価に用いた行列は表 2，表 3 で示したものすべてである．また，OpenAT_DAFRT を用いた場合のパラメータは 3.2 節と同様であり，自動チューニングを使わない場合の行列–ベクトル積のアルゴリズムはそれぞれ S2，U3 を用いた．

表 6 に Xabclib_LANCZOS でリスタート周期を 30，行列–ベクトル積の実装方式を S2 に固定した場合と比べ，双方を自動チューニングで最適化した場合の性能向上効果を示した．自動チューニングを使用することにより 20 個中 15 個で高い性能が得られるようになり，固定値では未収束であった行列 1 個が収束した．また，最も向上率の高かった t3dl では 22.5 倍の速度向上が得られた．

表 7 に Xabclib_GMRES でリスタート周期を 30，行列–ベクトル積の実装方式を U3 に固定した場合と比べ，双方を自動チューニングで最適化した場合の性能向上効果を示した．Xabclib_GMRES においても自動チューニングなしの場合と比べて 20 個の行列の中で 12 個において速度が向上し，未収束の行列 4 個が収束していた．特に，torso1 は最も性能向上率が高く，3.5 倍となった．

表 6　Xabclib_LANCZOS による 2 方式の自動チューニングの効果
Table 6　Effectiveness of restart frequency and sparse matrix-vector multiply auto-tuning by Xabclib_LANCZOS.

行列	自動チューニングなし			自動チューニングあり			自動チューニングによる速度向上
	M	restart	時間（秒）	last M	restart	時間（秒）	
vibrobox	30	24	1.13	35	20	1.12	1.01
Lin	30	1,047	601.65	150	54	190.48	3.16
cfd1	30	55	12.42	80	24	11.75	1.06
cfd2	30	45	18.69	70	28	21.35	0.88
gyro	30	10	1.13	35	16	0.83	1.36
t3dl	30	1,878	125.62	90	33	5.59	22.46
c-71	30	4	0.7	25	17	1.17	0.60
Si5H12	30	192	17.99	115	34	8.1	2.22
SiO	30	161	24.32	100	37	14.48	1.68
dawson5	30	1,052	135.75	105	34	14.68	9.25
H2O	30	623	168.46	130	40	35.34	4.77
F2	30	27	15.04	50	21	8.74	1.72
oilpan	30	28	10.63	45	24	9.72	1.09
shipsec1	30	26	20.89	50	30	29.73	0.70
bmw7st_1	30	1	1.06	15	5	0.85	1.25
SiO2	30	699	805.68	145	45	137.32	5.87
shipsec5	30	53	59.31	75	27	36.35	1.63
Si41Ge41H72	30	411	679.91	150	40	160.7	4.23
bmw3_2	30	3	4.2	25	19	8.63	0.49
Ga41As41H72	30	10,000	未収束	150	44	144.12	未収束 → 収束

表 7　Xabclib_GMRES による 2 方式の自動チューニングの効果
Table 7　Effectiveness of restart frequency and sparse matrix-vector multiply auto-tuning by Xabclib_GMRES.

行列	自動チューニングなし			自動チューニングあり			自動チューニングによる速度向上
	M	restart	時間（秒）	last M	restart	時間（秒）	
chem_master1	30	22	2.55	52	14	2.22	1.15
torso2	30	1	0.68	7	2	0.3	2.27
torso1	30	1	2.54	2	1	0.72	3.52
torso3	30	12	33.57	32	14	33.76	0.99
memplus	30	5	0.25	22	10	0.2	1.25
ex19	30	1,000	未収束	100	54	21.28	未収束 → 収束
poisson3Da	30	3	0.48	17	7	0.54	0.89
airfoil_2d	30	7	0.73	22	14	0.83	0.88
poisson3Db	30	7	10.38	17	14	11.04	0.94
viscoplastic2	30	19	2.93	37	15	1.74	1.68
xenon1	30	30	20.16	62	19	16.47	1.22
xenon2	30	40	92.96	72	20	64.31	1.45
wang4	30	5	0.37	17	9	0.29	1.28
ecl32	30	1,000	未収束	92	22	11.66	未収束 → 収束
sme3Da	30	670	215.49	100	87	97.48	2.21
sme3Db	30	1,000	未収束	100	120	380.39	未収束 → 収束
sme3Dc	30	1,000	未収束	100	98	531.37	未収束 → 収束
epb1	30	11	0.38	32	14	0.34	1.12
epb2	30	3	0.22	12	9	0.21	1.05
epb3	30	11	3.22	42	14	3	1.07

以上の結果から，OpenATLib によって提供される自動チューニングにより演算時間の大幅な増加や未収束を防げ，多くの場合で性能向上が得られると分かった．

4. おわりに

本稿では，数値計算ライブラリに対して自動チューニングの実装の再利用性を実現するため，自動チューニングインタフェース OpenATLib を提案した．さらに，それを利用して，疎行列反復解法ライブラリ Xabclib_LANCZOS，Xabclib_GMRES を開発した．

これらの効果を評価するために，開発したライブラリの性能評価を T2K オープンスパコン（東大版）上でフロリダ大学の Sparse Matrix Collection から得た 20 個の行列を用いて実施した．

評価の結果，8 割の行列で OpenATLib により演算時間の大幅な増加や未収束を防げており，その有効性が確認できた．特に Xabclib_LANCZOS では最大で性能が 22.5 倍，Xabclib_GMRES では最大で 3.5 倍となっていた．

しかし，残りの 2 割では OpenATLib の使用により性能が劣化していたこと，Xabclib_GMRES において行列–ベクトル積の自動チューニングの効果が見られないなどの問題も明確になった．また，CPU 数やメモリ量といった計算機環境によって OpenATLib がつねに最適な実装方式を選択できるかどうかの評価を行う必要がある．これらの解決が今後の課題となる．

謝辞　本研究は文部科学省「e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」の支援を受けている．

参考文献

1) Bilmes, J., Asanovic, K., Chin, C.-W. and Demmel, J.W.: Optimizing Matrix Multiply Using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology,

Proc. International Conference on Supercomputing 97, pp.340–347 (1997).
2) Whaley, R.C., Petitet, A. and Dongarra, J.J.: Automated Empirical Optimizations of Software and the ATLAS Project, *Parallel Computing*, Vol.27, pp.3–35 (2001).
3) Frigo, M.: A Fast Fourier Transform Compiler, *Proc. 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, Atlanta, Georgia, pp.169–180 (1999).
4) 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ソフトウエアに及ぼす効果，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).
5) Katagiri, T., Kise, K., Honda, H. and Yuba, T.: ABCLib_DRSSED: A Parallel Eigensolver with an Auto-tuning Facility, *Parallel Computing*, Vol.32, No.3, pp.231–250 (2006).
6) Vuduc, R., Demmel, J.W. and Yelick, K.: OSKI: A Library of Automatically Tuned Sparse Matrix Kernels, *Proc. SciDAC 2005, Journal of Physics: Conference Series* (2005).
7) Xabclib プロジェクトホームページ．http://www.abc-lib.org/Xabclib/index-j.html
8) 櫻井隆雄，直野　健，恵木正史，猪貝光祥，木立啓之：リスタート付ランチョス法における実行時パラメータ自動チューニング方式の提案，第 111 回ハイパフォーマンスコンピューティング研究会，情報処理学会研究報告 2007-HPC-111，pp.173–178 (2007).
9) 工藤　誠，黒田久泰，片桐孝洋，金田康正：並列疎行列ベクトル積における最適なアルゴリズム選択の効果，第 147 回アーキテクチャ研究会，情報処理学会研究報告 2002-ARC-147，pp.151–156 (2002).
10) Hernandez, V., Roman, J.E. and Tomas, A.: Evaluation of Several Variants of Explicitly Restarted Lanczos Eigensolvers and Their Parallel Implementations, *High Performance Computing for Computational Science–VECPAR 2006*, pp.403–416 (2007).
11) Saad, Y.: *Iterative Methods for Sparse Linear Systems* (*2nd ed.*), SIAM, Philadelphia (2003).
12) Davis, T.A.: Sparse Matrix collection. http://www.cise.ufl.edu/research/sparse/matrices/
13) Balay, S., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L.C., Smith, B. and Zhang, H.: PETSc User's Manual Revision 3.0.0. http://www.mcs.anl.gov/petsc/petsc-as/

(平成 21 年 10 月 2 日受付)
(平成 22 年　1 月 7 日採録)

櫻井　隆雄（正会員）

1980 年生．2003 年東京大学工学部電子情報工学科卒業．2005 年同大学院情報理工学研究科電子情報学専攻修士課程修了．同年（株）日立製作所入社．以来，並列計算機向け行列ライブラリ，自動チューニング，およびクライアントモニタリングシステムの研究開発に従事．

直野　　健（正会員）

1968 年生．1994 年京都大学大学院理学研究科数理解析専攻修士課程修了．同年（株）日立製作所中央研究所入社．現在，（株）日立製作所中央研究所主任研究員，博士（工学）．並列計算機 SR2201，SR8000，SR11000，SR16000 向け行列計算ライブラリの研究開発に従事．また，自動チューニング，業務モニタリングシステムの研究開発に従事．2000 年 JSPP 最優秀論文賞受賞，2006 年日本計算工学会奨励賞受賞，2009 年関東地方発明賞発明奨励賞受賞．日本応用数理学会，日本金融・証券計量・工学学会，日本計算工学会各会員．

片桐　孝洋（正会員）

東京大学情報基盤センター特任准教授．1994 年豊田工業高等専門学校情報工学科卒業．1996 年京都大学工学部情報工学科卒業．2001 年東京大学大学院理学系研究科情報科学専攻博士課程修了．博士（理学）．2001 年 4 月日本学術振興会特別研究員 PD，12 月科学技術振興機構研究者，2002 年 6 月電気通信大学大学院情報システム学研究科助手，2005 年 3 月から 2006 年 1 月米国カリフォルニア大学バークレー校コンピュータサイエンス学科訪問学者を経て，2007 年 4 月より現職．超並列数値計算アルゴリズム，およびソフトウェア自動チューニングの研究に従事．2002 年情報処理学会山下記念研究賞受賞．2007 年 Microsoft INNOVATION AWARD 2007 アカデミック部門最優秀賞受賞．日本ソフトウェア科学会，日本応用数理学会，ACM，IEEE-CS，SIAM 等各会員．

中島　研吾（正会員）

東京大学情報基盤センター教授（スーパーコンピューティング研究部門）博士（工学）．専門は計算力学・数値線形代数・並列アルゴリズム．東京大学工学部航空学科卒業（1985 年），テキサス大学オースティン校大学院航空宇宙工学・応用力学科修士課程修了（1993 年）．三菱総合研究所，東京大学理学系研究科を経て 2008 年より現職．日本応用数理学会，日本計算工学会，SIAM，AIAA，IEEE 各会員．平成 20 年度山下記念研究賞受賞．

黒田　久泰（正会員）

1970 年生．1993 年名古屋大学理学部物理学科卒業．1995 年京都大学大学院工学研究科応用システム科学専攻修士課程修了．2000 年東京大学大学院理学系研究科情報科学専攻博士課程単位取得退学．同年同大学情報基盤センター助手．2007 年同助教．2009 年より愛媛大学大学院理工学研究科電子情報工学専攻准教授．高性能計算，自動チューニングに関する研究に従事．ACM，IEEE，SIAM，日本応用数理学会，人工知能学会各会員．