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

片桐孝洋[†1,†2] 黒田久泰[†3] 大澤清[†2,☆]
工藤誠[†4,☆☆] 金田康正[†3]

数値計算において高性能を達成するため，現在でも多くの技法を用いてチューニングがなされる．この作業自体が時間を浪費するものであったり，また利用する計算機に対する高度な知識が必要となる．本論文ではこのチューニング作業を自動化するため，並列数値計算ライブラリに自動チューニング機構を付加することを提案する．従来の機構との違いは，(1) 分散メモリ型計算機を対象とする ；(2) 標準化された計算のみの最適化ではない ；(3) ライブラリ全体の最適化が可能である ；の 3 点である．提案機構を付加した並列数値計算ライブラリ ILIB (Intelligent LIBrary) を，日立の分散メモリ型並列計算機 HITACHI SR2201, HITACHI SR8000 で評価した．その結果，密行列 LU 分解ルーチンでは理論性能に対して 89%の効率，疎行列 LU 分解ルーチンでは 72%の効率という高い性能を達成した．また疎行列反復解法では，標準的にもちいられているパラメタに対する速度向上が 52 倍にもおよぶことが明らかとなった．ブロック化アルゴリズムにおけるブロック幅といったような演算と通信に影響するパラメタも，ILIB では最適化可能である．このブロック幅の最適化の結果，2.3 倍の速度向上が得られる場合があった．

Impact of Auto-tuning Facilities for Parallel Numerical Library

Takahiro KATAGIRI,[†1,†2] Hisayasu KURODA,[†3]
Kiyoshi OHSAWA,[†2,☆] Makoto KUDOH[†4,☆☆]
and Yasumasa KANADA[†3]

To attain high performance in numerical computations, one has to tune one's programs by using several techniques. The tuning work is a time-consuming work, and it needs special knowledge of the target architectures. To automate the tuning work, we propose an auto-tuning facility for parallel numerical libraries in this paper. The different points between conventional facility and our facility are: (1) our facility focuses on distributed memory parallel machines; (2) our facility is not the limited optimization facility to standard computations; (3) our facility can optimize the all area of the target library. We evaluate the ILIB (Intelligent LIBrary) which contains the proposed auto-tuning facility on the HITACHI's distributed memory parallel machines of the HITACHI SR2201 and HITACHI SR8000. The evaluation shows that the ILIB can attain the efficiency of 89% in a dense LU decomposition routine and that of 72% in a sparse LU decomposition to theoretical peak performance. In the sparse iterative method routine, we obtained the speedup factor of 52 times in comparison with the execution time by the default parameters. On the other hand, the proposed facility can tune a block factor for blocked algorithm, which can affect computations and communications. For the optimization of the block factor, we found a case that the speed-up factor was 2.3 times.

1. はじめに

多くの数値計算において，計算カーネルをチューニングすることは時間の浪費であるばかりでなく，利用する計算機に関する高度な知識を必要とする．高性能を達成するため，いまだに多くのコーディング技法を

†1 日本学術振興会特別研究員
Research Fellow of the Japan Society for the Promotion of Science
†2 東京大学大学院理学系研究科情報科学専攻
Department of Computer Science, Graduate School of Science, The University of Tokyo
†3 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
Computer Centre Division, Information Technology Center, The University of Tokyo
†4 東京大学理学部情報科学科
Department of Computer Science, Facility of Science, The University of Tokyo
☆ 現在，信陽ビジネスサービス株式会社
Presently with Shinyo Business Service Co., Ltd.
☆☆ 現在，東京大学大学院情報理工学系研究科コンピュータ科学専攻
Presently with Department of Computer Science, Graduate School of Information Technology Science, The University of Tokyo

施している．このチューニング作業を減らす目的で，線型計算プログラムの中には計算機メーカが提供する基本線型副プログラム BLAS (Basic Linear Algebra Subprograms)[1] を用いて作成されることがある．

ところがこの BLAS 自体の性能が低ければ，全体の性能は低下する．このような実装の問題を解く鍵は，BLAS に関する自動チューニングソフトウエアの利用にある．たとえば PHiPAC[2] や ATLAS[3] などの自動チューニングソフトウエアが既に知られている．BLAS をチューニングすることで，それを用いたすべてのソフトウエアのチューニングが可能なことから，これらのソフトウエアを本論文では汎用型自動チューニングソフトウエアとよぶ．

この一方で，BLAS を利用していなかったり，利用できないソフトウエアの自動チューニングは難しい．このようなソフトウエアでは，それぞれのソフトウエアごとに何らかの自動チューニング機構を持つことになる．たとえば，離散フーリエ変換のソフトウエアである FFTW[4] や疎行列反復解法ライブラリ[5] における自動チューニング機構などがその例である．これらのソフトウエアを本論文では特化型自動チューニングソフトウエアとよぶ．

この論文の目的は，数値計算ソフトウエアにおける特化型自動チューニング機構を提案することと，その有効性を実験的に示すことである．特化型自動チューニングに注目した理由は，以下の 3 つの事項による．

(1) 現在，並列処理に関する自動チューニング機構を持つソフトウエアはほとんど利用できない．
(2) ソフトウエア利用の観点から，自動チューニング機構はソフトウエアごとに含むべきである．
(3) 高性能を達成するには，広域最適化が必須となる．

(1) に関して：現在，分散メモリ型並列計算機向きのライブラリのチューニングを統一的に自動化するソフトウエアの枠組は皆無であるので，ソフトウエアごとに自動チューニング機構を付加するしかない．

(2) に関して：もし自動チューニング機構がその対象となるソフトウエアから分離されていると，利用者は高性能を達成するために自動チューニング用のソフトウエアを自分の環境にインストールしなくてはならず，容易な利用を妨げる．さらに自動チューニングに関する時間に関しても，不必要なルーチンまでチューニングをしてしまえば，チューニング時間が膨大になる恐れがある．(たとえば，BLAS の全てのルーチンを最適化することを考えてみよ．)

(3) に関して：BLAS のみに限定した汎用型自動チューニングソフトウエアでは，対象となるライブラリ全体から見ると局所最適化を行っているといえる．一般的により高い性能を得るには，プログラム全体を最適化する大域最適化が必要である．したがって大域最適化が行える特化型自動チューニングソフトウエアの方がさらに性能を引き出せる可能性がある．並列処理では通信処理と演算処理が存在するが，この通信処理と演算処理はお互いに影響していることが多い．このことから計算の高速化のために，両方の処理に関連する要因の最適化が必要と考えられる．この点で従来の局所最適化の方法では，十分に性能が引き出されない可能性がある．

以上の理由から，並列数値計算ライブラリには特化型自動チューニング機構を含むべきである．図 **1** に，本論文で提案する特化型自動チューニング機構の概念を示す． 図 1 の概念では，まずライブラリの機能を

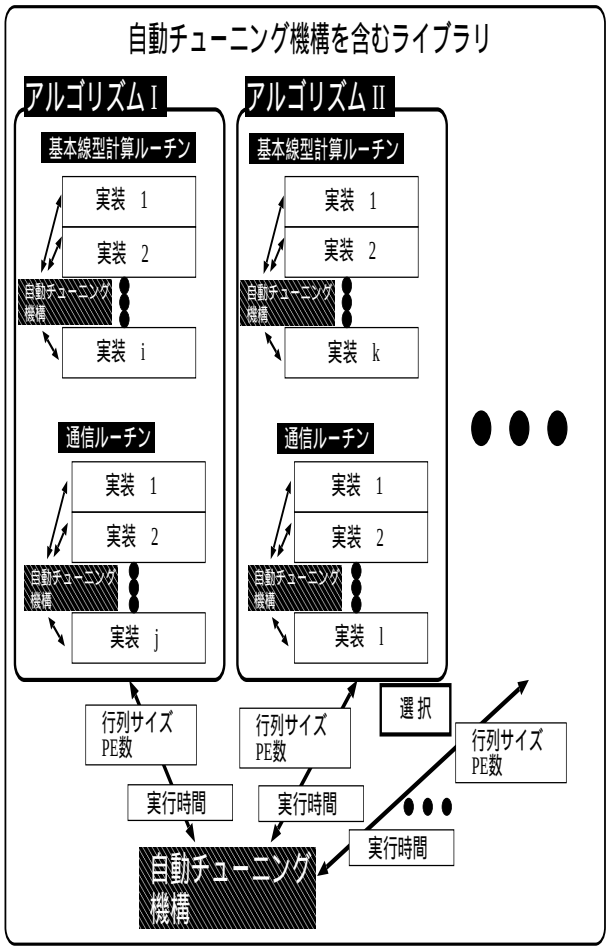


図 **1** 自動チューニング機構の概念
Fig. 1 The concept of auto-tuning facilities.

満たす複数のアルゴリズムを用意する．それらのアルゴリズムを実現するにあたり，計算カーネルの実装方式や通信の実現方式も複数用意する．これらの複数の実装方式から，行列サイズ，Processor Element (PE) 数および実行時間を考慮し，自動チューニング機構が適するアルゴリズムと実装方式を自動選択する．これが本論文で提案する自動チューニング機構である．以降，この特化型自動チューニング機構を用いたチューニングを単に自動チューニングとよぶ．

本論文の構成は以下の通りである．まず第 2 章で，関連研究と自動チューニングの分類を説明する．次に第 3 章で，本論文で提案する自動チューニング機構を付加した並列数値計算ライブラリ ILIB (Intelligent LIBrary) の概要と機能について説明する．次に 4 章で，自動チューニング項目などの具体的な説明を行う．5 章では自動チューニング機構の有効性を示すため，分散メモリ型並列計算機である HITACHI SR2201 と HITACHI SR8000 を用いる．本論文で示す自動チューニング機構により ILIB のパラメタが実際に変化するのか，またチューニングの効果があるのか評価した結果を示す．最後に，本論文のまとめを述べる．

2. 関連研究と自動チューニングの 2 分類

2.1 関 連 研 究

数値計算ライブラリにおける自動チューニング機構の付加というアイデアそのものは新しいものではない．以下に自動チューニング機構を付加したソフトウエアの名称と特徴をまとめる．

- PHiPAC (a Portable, High-Performance, Ansi C Coding Methodology)[2]，1997 年
  数値計算ライブラリにおいて，最も初期に自動チューニングの概念を導入したのが PHiPAC である．PHiPAC では行列-行列積 (BLAS3 演算) において ANSI C の文法に従った書式を定め，その書式に従いコードの自動生成を行って最も高速である書式を選択する．またブロック化アルゴリズムを利用していることから，最も高速となるブロック幅の探索も行うことで自動チューニング機能を実現した．
- ATLAS (Automatically Tuned Linear Algebra Software)[3),6)]，1998 年
  ATLAS では PHiPAC の手法にさらに工夫を加え，BLAS の全ルーチンのチューニングを可能にしたソフトウエアである．ATLAS の主な工夫は，パラメタ探索の時間を減らすために演算カーネルの挙動をモデル化した点である．このモデル化によりキャッシュサイズの情報や数回の実行時間の測定から，最適パラメタ絞りこみと実測していない行列サイズのチューニングも可能となった．
- PSPARSLIB[7),8)]，1996 年
  PSPARSLIB は疎行列反復解法の並列計算ライブラリであるが，前処理に関するパラメタの自動選択が可能である点が自動チューニングソフトウエアの観点での主な特徴である．PSPARSLIB は Bi-CGStab 法などの多くの反復解法を含んでいる．
- FFTW[4)]，1999 年
  FFTW は離散フーリエ変換を計算するための，C 言語で書かれたサブルーチンライブラリである．任意の入力数に対応しているが，入力数の情報から利用するアルゴリズムを切替える．また演算回数を減らすための最適化も行う．このことから FFTW は，自動チューニング機構付きライブラリといえる．
- ILIB (Intelligent LIBrary)[9)~12)]，1998 年 ~
  ILIB は疎行列反復解法と密行列直接法の並列計算ライブラリである．ILIB の疎行列や直接法ライブラリでは，前処理に関するパラメタの自動選択ばかりでなく，演算カーネルの実装方式の自動選択も行う．さらに通信実装方式の自動選択も行っている．密行列直接法ではブロック化アルゴリズムを利用しているが，性能に影響するブロック幅の自動決定も行う．

2.2 自動チューニングの 2 分類

この論文では，数値計算における自動チューニングを以下のように分類分けをしてよぶことにする．

(i) 実行時自動チューニング
(ii) 実行前自動チューニング

(i) 実行時自動チューニングでは，対象となるルーチンやライブラリは実行時に最適化される．この自動チューニングは，疎行列や反復法のライブラリに適用できる．なぜならこれらのライブラリではチューニング項目が，本質的に問題の性質に依存するからである．たとえば疎行列ソルバでは，性能は行列の非零要素の位置に依存することが多い．また反復解法を用いたソルバの性能は，行列の数値的な性質によることが多い．これらの原理を利用したソフトウエアとして，黒田と金田により提案された連立一次方程式の反復解法の一つである GMRES(m) 法の実行時自動チューニング機構 [5)] があげられる．この実行時自動チューニングは，人手により行うことが困難である．その理由は，実行時にどの実装方式が最も適するものであるか人手

で選択することが困難であることによる．実行時自動チューニングの特性上，チューニング時間に関する制約が存在する点に注意する．すなわちチューニングの時間を含めてチューニングしない場合（パラメタ固定の場合）よりも，実行時間が高速となる必要がある．

(ii) 実行前自動チューニングでは，対象となるルーチンやライブラリをそれらが呼ばれる前に最適化する．この自動チューニングは，密行列ソルバに適用できる．なぜならチューニング項目は多くの場合，問題の性質に依存しないからである．たとえばLU分解などの直接法ソルバの性能に関しては，行列の次元数にのみ依存し行列の数値的特性によらないことが多い．このことからILIBのLU分解ルーチンでは，ブロック幅，ループアンローリング段数といった項目を自動チューニングする．実行前自動チューニングは人手により行うことが可能であるが，チューニング作業自体が時間を浪費するものであったり，またコードの変更の際にバグを入れる可能性があるので，このチューニング作業は自動化すべきである．なお本論文で扱う実行前自動チューニングにおいては，インストール時に代表的な問題サイズにおけるパラメタ最適化を行うことでチューニングを終了する，もしくは実行前に問題サイズが確定している時にそのサイズに対するチューニングを1度だけ行う状況を想定する．また実行前自動チューニングでは，高い性能を達成するパラメタが得られることを優先しチューニング時間に関する制約はないものと仮定する．

表1に数値計算ライブラリにおける自動チューニング機構の特徴をまとめる．

表1 数値計算ライブラリにおける自動チューニング機構の特徴のまとめ
Table 1 Summary for the characteristics of auto-tuning facilities in numerical libraries.

分類	型	最適化	分散メモリ最適化	名称
実行時	特化	局所	○	PSPARSLIB[7),8)]
		大域	○	ILIB_GMRES[5),10)]
実行前	汎用	局所	×	PHiPAC[2)]
			×（共有メモリ並列化のみ）	ATLAS[3),6)]
	特化	局所	×（通信処理最適化なし）	FFTW[4)]
		大域	○	ILIB_DRSSED[11),13)]
			○	ILIB_LU[11)]
			○	ILIB_RLU[14)]

3. ILIB：自動チューニング機能付き並列数値計算ライブラリ

3.1 開発方針

我々は科学技術計算用ライブラリ群，特に並列数値計算において，以下に示す特徴・機能を有する数値計算ライブラリの開発を目指してきた．

- ユーザが指定するパラメタが少ないこと
- 演算カーネルに関する自動チューニング機能があること
- 通信処理に関する自動チューニング機能があること
- 利用するアルゴリズムに関する自動選択機能があること

我々は，この設計思想に基づく直接法（LU分解，固有値計算における三重対角化）や反復法（GMRES法などの疎行列を係数行列とする連立一次方程式の解法）などにおいて自動チューニング機能を付加した並列数値計算副プログラム集ILIB（Intelligent LIBrary）を開発した．

3.2 連立一次方程式ライブラリ

ILIBの並列連立一次方程式ライブラリは，式(1)に示される連立一次方程式の解ベクトル x を求めることができる．

$$Ax = b \tag{1}$$

ここで係数行列 $A \in \Re^{n \times n}$ は実数の非対称密行列もしくは非対称疎行列である．また右辺ベクトル b は $b \in \Re^n$ であり，解ベクトル x は $x \in \Re^n$ である．

式(1)の解法として，直接解法と反復解法の2種が存在する．ILIBの並列ライブラリにおいては

- 直接解法（密行列用）：
  LU分解に基づくルーチン[15)]（ILIB_LU）
- 直接解法（疎行列用）：計算範囲縮小によるLU分解に基づくルーチン[14),15)]（ILIB_RLU）
- 反復解法 疎行列反復法の一種である$\mathrm{GMRES}(m)$法に基づくルーチン[5)]（ILIB_GMRES）

において自動チューニング機能を実装している．

3.3 固有値問題ライブラリ

ILIBの並列固有値ライブラリは，式(2)に示す標準固有値問題の固有分解を行うことができる．

$$A = X \Lambda X^{-1} \tag{2}$$

ここで係数行列 $A \in \Re^{n \times n}$ は実数の対称密行列であり，固有値 λ_i $(i = 1, 2, ..., n) \in \Re$ から構成される行列を $\Lambda = \mathrm{diag}(\lambda_i)$，固有ベクトル x_i $(i = 1, 2, ..., n)$ $\in \Re^n$ から構成される行列を $X = (x_1, x_2, ..., x_n)$ とする．式(2)の固有分解を行う解法として良く知られ

ている Householder-二分-逆反復法を用いている [16]．

ILIB におけるこの対称実数密行列ルーチン ILIB_DRSSED (Dense Real Symmetric Standard Eigenvalue Decomposition) は，直接解法 (Householder 三重対角化，Householder 逆変換など) と反復解法 (逆反復法) を両方必要とする．現在の ILIB において自動チューニング機能を付加している部分は，

- 直接解法 (密行列) :相似変換における三重対角化ルーチン [11] (ILIB_TrdRed)
- 直接解法 (密行列) :固有ベクトル計算に利用可能な直交化ルーチン [13] (ILIB_MGSAO)
- 直接解法 (密行列) :固有ベクトル計算に用いる Householder 逆変換ルーチン [13] (ILIB_HouseInv)

である．

4. ILIB における自動チューニング機能の説明

4.1 連立一次方程式ライブラリ

4.1.1 ILIB_LU の自動チューニング項目

並列密行列 LU 分解ルーチンでの自動チューニングとして，以下に示す項目が考えられる．

(i) ブロックアルゴリズムにおけるブロック幅 :我々の LU 分解ルーチンは外積形式ガウス法を用いてブロック化 [17] を行っているので，ブロック幅が性能に大きく影響する．図 **2** の外積形式ガウス法におけるブロック幅が示すように，ブロック幅に関する 2 つのパラメタ BH，BW がある．

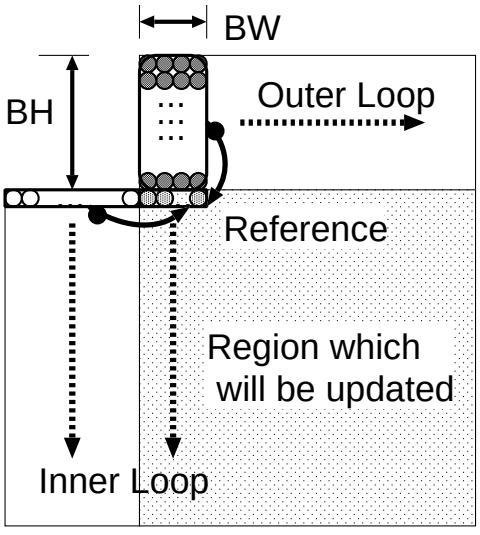


図 **2** 外積形式ガウス法における 2 つのブロック幅
Fig. 2 The two kinds of block length in outer-product formed Gaussian elimination.

さて (i) の演算性能をチューニングするためには，BH, BW を自動的に変更して最も高速となりうる組 (BH_{opt}, BW_{opt}) を決定すればよい．ところがこの組は，使用する並列計算機のアーキテクチャやコンパイラの最適化性能などに依存し，実際にプログラムを実行してみないことには最適な組合せを決定できない．このことから我々のライブラリでは現在，適度に小さい問題サイズの行列を自動生成し，(BH,BW) の全ての組合せを全探索して最適な組を決定する方式を実装している．具体的には，$BH, BW = \{1, 2, 3, ..., 20\}$ とし，この $20 \times 20 = 400$ 通りの組合せから最適な (BH_{opt},BW_{opt}) を決定する．この自動チューニングはライブラリをインストールする時に 1 度行えば，利用するプロセッサ台数や並列計算機環境が変わらなければ再度する必要がないことに注意しておく．すなわち実行前自動チューニングを行うことができる．

(ii) 通信実装方式 ：枢軸選択処理の際の通信方式を同期的にするか，非同期的にするかということである．

(iii) データ通信方式 ：行列 A のデータ分散をどの様にするのかということである．

現在のバージョンでは，(ii)(iii) の自動チューニングは実装されておらず，(ii) は同期的に通信を行い，(iii) では列サイクリック分散，で固定されている．

4.1.2 ILIB_RLU の自動チューニング項目

ILIB_RLU は帯行列を LU 分解するルーチンであるが，内部で行列の非零構造や零ブロックを考慮しており，全体としては疎行列 LU 分解ルーチンとみなすことができる．

ILIB_RLU の自動チューニングとして，以下に示す項目が考えられる．

(i) 非零構造からの演算範囲限定 :構造解析の分野でしばしば用いられる疎行列は対角要素に非零要素が集中していることが多い．その場合対角要素以外の要素に対して乗算を行うとほとんどの場合零要素との乗算になり，無駄な演算を行っていることになる．そこで非零要素の構造を特定するインデックスとして，各行，各列の非零要素の終端位置を表す配列を定義する．具体的には $jlast(i)$ = (第 i 行の非零要素の最大列番号)，$ilast(j)$ = (第 j 列の非零要素の最大行番号) とする．実際には計算誤差を抑えるための枢軸選択処理により計算が進むにつれて非零要素の構造は変化し，その際のインデックスの変化を最小にするために $jlast(i) \geq jlast(i-1)$, $ilast(j) \geq ilast(j-1)(i,\ j = 2, 3, \ ...$,行列サイズ $n)$ とする．$jlast(i)$, $ilast(j)$ で示される要素が第 i 行，第 j 列の計算対象領域の終端となる (図 **3**)．枢軸選択処理に

より第 i 行と第 k 行が交換される場合，$jlast(I) = jlast(k)(I = i+1,\ i+2,\ \ldots,k-1)$ と更新する．$ilast(j)$ についてはこのような更新を行う必要はない．以上のようにして無駄な領域の演算を行わないようにする．この手法は実行時自動チューニングに分類される．

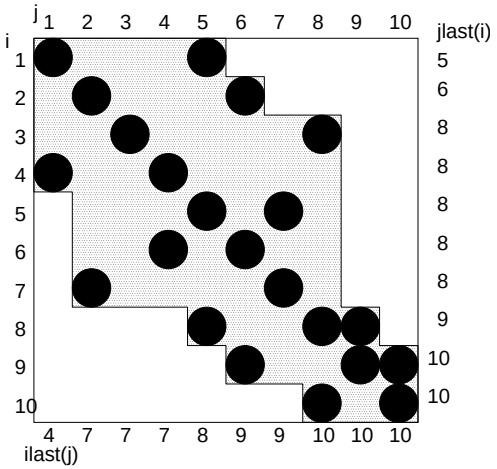


図 3 非零構造からの演算範囲限定
Fig. 3 The restriction of update area based on the location of non-zero elements.

非対称行列のサイズを n とすると，通常の LU 分解では $2 \times n^3/3$ の演算量を必要とするが，演算範囲を限定した場合，用いる行列を帯行列とみなして半帯幅 $\max_i(jlast(i) - i)$ を hbw とすると，演算量は最悪でも $4 \times hbw^2 \times n$ となる．たとえば $hbw = n/5$ とすると，その演算量は $4 \times n^3/25$ に抑えられ，通常の場合の 24% にまで減少する．

(ii) 零要素による更新演算の省略 :疎行列において，ある行の対角要素とその右端の要素との間に非零要素がほとんど存在しない場合がある．このときブロックアルゴリズムで LU 分解を行うと，ブロック行 (Block Rows) 内の $BH \times BW$ の領域がすべて 0 になり，更新演算がすべて零要素との乗算になるため更新領域 (Update Region) 内の演算が不要になる場合がある (図 4)． ILIB_RLU では，更新演算を開始する前に図 4 の $BH \times BW$ の長方形ブロックがすべて 0 であるかどうかを調べて，すべて 0 の場合には更新領域内の演算を行わないようにして速度向上を図る．この手法も実行時自動チューニングとなる．

4.1.3 ILIB_GMRES の自動チューニング項目

疎行列反復解法ルーチンの場合は係数行列 A が疎

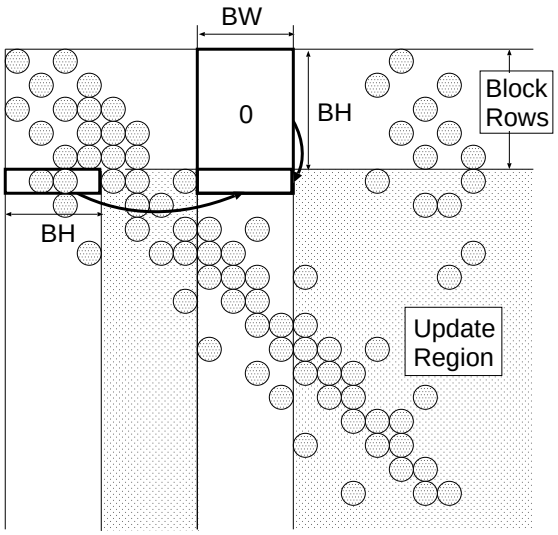


図 4 零要素による更新演算の省略
Fig. 4 Skipping the update computations based on the area of non-zero elements.

であり，かつ一般的に非零要素の位置が不規則的であるので，チューニングの要因が係数行列 A の非零要素の位置に依存する．このことはライブラリ実行時にならないとチューニングを行えないことを意味し，結果として実行時自動チューニングを行わざるを得ない．また疎行列反復法ではデータ圧縮方式が性能を左右することが多いが，ここでは行方向圧縮方式でデータが格納されているものと仮定する．

並列疎行列反復解法ルーチンでの自動チューニング項目は大まかには以下に示す通りである．

(i) 疎行列-ベクトル積におけるアンローリング段数 : 行方向圧縮形式用の演算カーネルにおけるループ展開数を決める．具体的には反復中は非零要素の位置は変化しないので，ループアンローリングされたカーネルを複数用意しておき，最初の反復処理を実行する前に 1 度だけ全てのカーネルの速さを検査することで最適なカーネルが決定できる．

(ii) 疎行列-ベクトル積における通信方式 :1 対 1 通信を用いて通信回数を最少にする実装方式か，同期的に全体全通信をする実装方式かを自動選択する．この自動チューニングも最初の反復処理を実行する前に 1 度だけ行えばよい点に注意する．

(iii) 前処理方式 :一般に反復解法で広く用いられている，収束の加速技法である前処理の方式を自動的に決定する．最適な前処理方式は，行列の性質や通信処理の性能で変わるので，自動的な決定が困難である．今回実装したチューニング手法では，複数用意した前

処理方式をそれぞれ異なる各反復で 1 回づつ適用して，最も残差ベクトルを減少させた方式を自動選択する．

(iv) その他の各反復解法に必要とされる処理 :(i)–(iii) の他に各反復解法に特化した性能パラメタを自動調整する必要がある．具体的に GMRES(m) 法では，(1) リスタート周期 ;(2) 直交化の方式 ;などが全体の実行時間に大きく影響する．我々が実装した自動チューニング手法では，(1) に関しては 2,4,6,8 と m になるまで各反復で増加，m になると 2 に戻す方法 [5] を用いている．この周期 m はメモリ残量から自動的に決定される．(2) に関しては，並列効率が高いが精度が悪い方法 (Classical Gram-Schmidt 法，CGS)，CGS よりも数倍の実行時間は増加するが精度が良い方法 (改良された Classical Gram-Schmidt 法，IRCGS (Iterative Refinement CGS))[18]，および並列効率が低いが精度が高い方法 (Modified Gram-Schmidt 法，MGS) の 3 種を実装している．我々の適用方針は，まず反復に入る前に CGS と MGS の実行時間を調べ速い方を選択する．反復を開始した後，誤差の減少がある値 (ここでは 0.5%) 以下になった場合，IRCGS を強制的に選択する．なおこれらの自動チューニングが最適であり，必ず収束するという理論的な保証は無いことに注意しておく．

4.2 固有値問題ライブラリ

本論文で示す固有値問題ライブラリの各ルーチンは，全て反復解法ではない．したがって，これらの自動チューニングは実行前自動チューニングとなる☆．

4.2.1 ILIB_DRSSED の自動チューニング項目

並列密対称標準固有値解法ルーチン ILIB_DRSSED は，以下の 3 つのサブルーチンから構成される．これらのサブルーチンの主な演算カーネルは，BLAS1, BLAS2, および BLAS3 とよばれる異なる種類の構成をもつ．

- 三重対角化ルーチン ILIB_TriRed (BLAS2)
- MGS 直交化ルーチン ILIB_MGSAO (BLAS3)
- Householder 逆変換ルーチン ILIB_HouseInv (BLAS1)

ILIB_DRSSED の自動チューニング項目は，大まかには以下に示す通りである．

(i) ベクトル操作 (BLAS1 演算) のアンローリング段数 :Householder 逆変換ルーチン ILIB_HouseInv における主演算 $x = x - \mu u$ (ここで $u, x \in \Re^n$, $\mu \in \Re$) のループアンローリング段数を自動決定する．

(ii) 行列-ベクトル積 (BLAS2 演算) のアンローリング段数 :三重対角化ルーチン ILIB_TriRed における k 反復時の主演算 (1) 行列-ベクトル積 $A^{(k)}u$;(2) 行列更新 $A^{(k+1)} = A^{(k)} - u(x^T - \mu u^T) - xu^T$;におけるループアンローリング段数を自動決定する．

(iii) 行列-行列積 (BLAS3 演算) のアンローリング段数 :MGS 直交化ルーチン ILIB_MGSAO における主演算 $A = A - \mu A$ におけるループアンローリングの段数を自動決定する．

(iv) 通信実装方式の選択 :三重対角化ルーチン ILIB_TriRed における処理 (ii) 行列-ベクトル積を実行する為に必要な，ベクトルのリダクション演算の実装方式を自動選択する．

(v) ブロック幅の選択 :MGS 直交化ルーチン ILIB_MGSAO はブロック化アルゴリズムを用いていることから，ブロック幅を自動調整する．このブロック幅は，演算処理である (iii) 行列-行列積 (BLAS3 演算) と通信回数に影響を及ぼすパラメタである点に注意する．

5. 性能評価

この章では，本稿で示した ILIB ルーチン群を HITACHI SR2201 および HITACHI SR8000 を用いて評価した結果を記す．

本評価で使用した SR2201 の各 PE の理論ピーク性能は 300MFLOPS，PE 間は三次元ハイパクロスバ網で結合されており，その最大転送性能は 300Mbyte/秒である☆☆．また SR8000 の各 PE の理論ピーク性能は 8GFLOPS である．その各ノードは 1GFLOPS の IP(Instruction Processor) 8 台の共有メモリ構成となっている．ノード間は三次元ハイパクロスバ網で結合されており，その最大転送性能は片方向で 1Gbyte/秒，双方向で 2Gbyte/秒である☆☆☆．なお，通信ライブラリとしては MPI (Message Passing Interface) を利用する．

☆ ただし逆反復法ルーチン (固有ベクトル計算時) などを最適化する場合には，実行時自動チューニングが不可欠であることに注意する．

☆☆ 東京大学情報基盤センターが所有している 1024PE の SR2201 のうち 128PE を使用した．またコンパイラとして FORTRAN では，日立の最適化 FORTRAN90 V02-06-/D，オプションとしては -rdma -W0,'OPT(O(SS))' を指定した．C では +O4 -Wc,-hD1 を指定した．

☆☆☆ 東京大学情報基盤センターが所有している 128 ノード (8IP/1 ノード) の SR8000 のうち 1 ノード–4 ノードを使用した．また，コンパイラとして 日立の最適化 FORTRAN90 V01-00，オプションとしては ノード内並列版 (共有メモリ構成) に関しては -W0,'opt(o(ss)),mp(p(0))'，ノード間並列版 (分散メモリ構成) は -W0,'opt(o(ss)),mp(p(4))' を指定した．

5.1 連立一次方程式ライブラリの性能

5.1.1 密行列 LU 分解ルーチン ILIB_LU の性能

ILIB_LU での自動チューニングのパラメタは，

- 同時消去段数
  $BH = \{$ 1 段，2 段，3 段，$\cdots$，20 段 $\}$
- 同時消去列数
  $BW = \{$ 1 列，2 列，3 列，$\cdots$，20 列 $\}$

の 2 種類である．

表 2 に SR8000 における，(1) ブロック化の効果を無くし ($(BH, BW) = (1, 1)$)，すべての演算をコンパイラに最適化させた場合；(2) 文献 17) から得られる妥当と思われるパラメタの組 ($(BH, BW) = (8, 4)$: SR8000，$(BH, BW) = (5, 2)$:SR2201) を設定した場合；(3) 自動チューニングにより最適化して得られたパラメタの組を用いた場合；の実行時間の比較を示す．

表 2 ILIB_LU (密行列 LU 分解) における性能
Table 2 Performance of ILIB_LU (LU decomposition for dense matrices).

(a) SR2201，16PE，問題サイズ :11608

自動チューニング	段数 (BH)	列数 (BW)	実行時間 [秒]	MFLOPS (効率%)
なし (1)	1	1	1157.88	900 (18.8%)
なし (2)	5	2	333.00	3131 (65.3%)
あり (3)	6	2	319.22	3266 (68.1%)

(b) SR8000，2 ノード (16IP)，問題サイズ :41296

自動チューニング	段数 (BH)	列数 (BW)	実行時間 [秒]	GFLOPS (効率%)
なし (1)	1	1	25278.84	1.85 (11.6%)
なし (2)	8	4	5343.38	8.78 (54.9%)
あり (3)	10	9	3286.19	14.2 (89.3%)

表 2(b) から自動チューニング機能により，ILIB_LU は SR8000 の 2 ノード (16IP) におけるピーク性能 (16GFLOPS) に対する効率 89% を達成している．このことから自動チューニングにより性能を十分に引き出すことができたといえる．

5.1.2 疎行列用 LU 分解ルーチン ILIB_RLU の性能

ILIB_RLU は，疎行列の零要素の位置により演算範囲が動的に変化する．したがって性能は行列の形状に依存する．ここでは，人工心臓の流体解析 [19] に用いられた係数行列を性能評価の対象にする．表 3 および図 5 に使用した係数行列の特徴を示す．また ILIB_RLU ではある程度帯幅の大きな行列を対象とするため，一般的なスパースソルバのように疎行列圧縮方式を利用しない．すなわち密行列としてデータを所有する点に注意する．

表 3 係数行列の情報
Table 3 Information of the coefficient matrix.

(a) 使用した係数行列，次元数 $n = 41296$

	非零要素の幅 (対角要素からの距離)	一行当たりの非零要素数
平均	2334.0	67.6
最大	4007	107
最小	43	7

(b) 使用した係数行列，次元数 $n = 11608$

	非零要素の幅 (対角要素からの距離)	一行当たりの非零要素数
平均	1254.2	63.3
最大	2179	107
最小	33	7

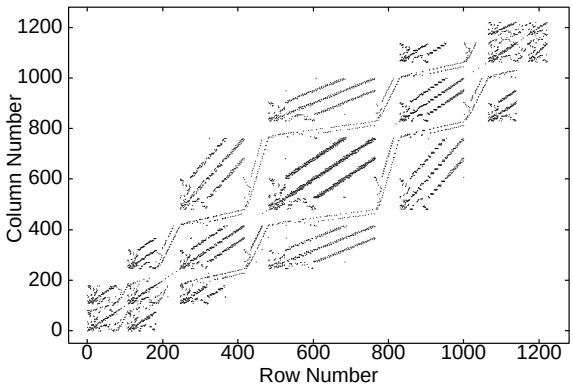


図 5 係数行列の形状 (非対称行列, 問題サイズ :1222)
Fig. 5 The location of non-zero elements for the coefficient matrix (non-symmetric matrix, problem size:1222.)

ILIB_RLU での演算カーネルに対する自動チューニングのパラメタは，

- 演算範囲限定
- 更新演算の省略

の 2 種類である．

ここではまず，演算範囲限定の効果を評価するため ILIB_LU ルーチンにおいて自動チューニング済の演算パラメタに対して，演算範囲限定の自動チューニングを施した場合の結果を表 4 に示す．

表 4(b) から演算範囲限定の自動チューニング機能により，最大で 39 倍の速度向上が達成できる場合がある．したがってこの機能は速度向上の観点で有用である．ところでスパースソルバとよばれる疎行列用の LU 分解ルーチンでは，無駄な演算を省くことで演算量と計算時間，およびメモリ領域の削減を図っている．このことから単純に比較はできないが，高性能な並列スパースソルバでも理論性能に対する効率 30%程度しか達成できない [20] ことを考慮すると，ILIB_RLU で得られた理論性能に対する効率 72.5%という値は，きわめて高い性能を達成できることを意味している．

表 4 密行列 LU 分解に対する ILIB_RLU (疎行列用 LU 分解) における演算範囲限定の効果
Table 4 Effect of the restriction of update region in ILIB_RLU (LU decomposition for sparse matrices) with comparison to dense LU decomposition.

(a) SR2201, 16PE, 問題サイズ :11608

段数 (BH)	列数 (BW)	実行時間	MFLOPS (効率%)	速度向上
6	2	33.73 [秒]	2374 (49.5%)	9.5

(b) SR8000, 2 ノード (16IP), 問題サイズ :41296

段数 (BH)	列数 (BW)	実行時間	GFLOPS (効率%)	速度向上
10	9	84.18 [秒]	11.6 (72.5%)	39.0

次に更新演算の省略の効果を評価するため, ILIB_LU ルーチンにおいて自動チューニング済の演算パラメタに対して, 更新演算の省略の自動チューニングを施した場合の結果を表 5 に示す.

表 5 ILIB_RLU (疎行列用 LU 分解) における更新演算省略の効果
Table 5 Effect of the skipping update calculation in ILIB_RLU (LU decomposition for sparse matrices).

(a) SR2201, 16PE, 問題サイズ :11608

段数 (BH)	列数 (BW)	実行時間	速度向上 (演算範囲限定に対する)
6	2	32.64 [秒]	1.033

(b) SR8000, 2 ノード (16IP), 問題サイズ :41296

段数 (BH)	列数 (BW)	実行時間	速度向上 (演算範囲限定に対する)
10	9	83.81 [秒]	1.004

表 5 から更新演算の省略の効果は, このテスト行列の場合は高々1.033 倍とわずかである. 表 5 の結果のように更新演算の省略の自動チューニングは行列の非零構造に依存するので, 適用する/しないを実行時に判断する必要がある.

5.1.3 疎行列反復解法ルーチン ILIB_GMRES の性能

ILIB_GMRES での自動チューニングのパラメタは,

- 行列-ベクトル積の種類 = { プリフェッチなし, プリフェッチあり }
- 行列-ベクトル積のアンローリング (行方向) 段数 = { なし, 2 段, 3 段, ···, 9 段 }
- 行列-ベクトル積のアンローリング (列方向) 段数 = { なし, 2 段, 3 段, 4 段, 8 段 }
- 行列-ベクトル積の通信方式 = { MPI_ALLGATHER, 1PE 集約 → MPI_BCAST ,MPI_ISEND → MPI_IRECV, MPI_IRECV → MPI_ISEND ,MPI_SEND → MPI_RECV }
- 直交化方式 = { CGS, IRCGS, MGS }
- 前処理方式 = { なし, 行列多項式前処理 [5], ブロック不完全 LU 分解前処理 [5] }

の 6 種類である. ここで "プリフェッチあり" とは, 行列-ベクトル積のコードにおいてループ中の依存関係をフロー依存からループ伝搬フロー依存に変更し, パイプライン処理向きにしたコードのことを意味している. このような 2 種のコードを用意した理由は, 疎行列-ベクトル積ではループ長が短く, かつ間接参照が必要であることによる. したがってプリフェッチ向きコードによる性能向上が, 密行列の場合に比べ期待できる. また MPI_ALLGATHER とは, MPI における同期全対全のベクトル収集関数, MPI_BCAST は同期 1 対全放送関数, MPI_SEND は同期 1 対 1 送信関数, MPI_RECV は同期 1 対 1 受信関数, MPI_ISEND は非同期 1 対 1 送信関数, そしてMPI_IRECV は非同期 1 対 1 受信関数を意味している.

一般に疎行列反復解法ではその実行時間は係数行列 A の性質に依存するので, 以降に示す 3 つの問題について性能を評価する.

[問題設定]

1 行当たりの非零要素の個数の最大値が 3, 5, 7 の 3 つの問題で性能を測定した.

- 問題 1
  Toeplitz 行列

$$A = \begin{bmatrix} 2 & 1 & & & \\ 0 & 2 & 1 & & \\ \gamma & 0 & 2 & 1 & \\ & \gamma & 0 & 2 & \cdots \\ & & \cdots & \cdots & \cdots \end{bmatrix}$$

  を係数行列とし, 右辺は $b = (1, 1, \cdots, 1)^T$ とする. γ については, $\gamma = 1$, 2 の 2 種類. 行列の次元は 4,000,000 とする.
- 問題 2
  領域 $\Omega = [0,1] \times [0,1]$ における楕円型偏微分方程式の境界値問題

$$\begin{aligned} -u_{xx} - u_{yy} + Ru_x &= g(x,y) \\ u(x,y)|\partial\Omega &= 1 + xy \end{aligned}$$

  右辺は厳密解が $u = 1 + xy$ となるように定める. 領域を 400×400 のメッシュで 5 点中心差分によって離散化した. 得られた連立一次方程式の係数行列の次元は 160,000 である. R については, $R = 1$, 1000 の 2 種類.

表 6 ILIB_GMRES (GMRES(m) 法) における自動チューニング結果による性能．単位は秒．(HITACHI SR2201) ここで，itr. は反復回数，Ort.T は直交化にかかる時間，h, v.T は最小二乗法の求解時間，max m はリスタート間隔，Ort.M は直交化方式を意味している．

Table 6 Performance of ILIB_GMRES (GMRES(m) method) with auto-tuned parameters. The unit is in second. (HITACHI SR2201) The notations of itr., Ort.T, h, v.T, max m and Ort.M show that the number of iterations, time for orthogonalization, time for solving a least squares problem, the re-start frequency and the orthogonalization methods, respectively.

(a) 問題 1

γ	1.0				2.0			
PE	8	16	64	128	8	16	64	128
T1(itr.)	49.6(43)	36.7(43)	20.9(43)	22.0(43)	353.3(337)	277.6(323)	153.9(323)	143.9(323)
T2itr.)	40.5(18)	24.0(19)	8.3(19)	5.3(20)	124.9(332)	86.6(321)	22.8(321)	13.7(321)
M.V.T	21.7	11.0	2.6	1.3	38.0	18.1	3.9	1.9
Ort.T	4.7	2.4	0.8	0.4	56.6	52.1	12.6	6.2
h, v.T	0.9	0.5	0.1	0.1	7.5	3.6	1.0	0.7
max m	16	64	128	128	16	64	128	128
Unroll	N(1,3)	N(1,3)	N(3,3)	P(3,3)	N(1,3)	N(1,3)	N(3,3)	P(3,3)
Com.M	Send	Send	Irecv	Send	Send	Send	Send	Send
Ort.M	MGS	MGS	CGS	CGS	MGS	MGS	CGS	CGS
Pre.M	BILU	BILU	BILU	BILU	None	None	None	None
T1/T2	1.22	1.52	2.51	4.15	2.82	3.20	6.75	10.5

(b) 問題 2

R	1.0				1000.0			
PE	8	16	64	128	8	16	64	128
T1 (itr.)	1205.7 (21842)	769.3 (21842)	520.3 (21842)	413.1 (21842)	90.4 (1597)	55.0 (1650)	34.3 (1646)	36.1 (1618)
T2(itr.)	90.7(1349)	44.3(1328)	14.1(1596)	7.9(1497)	23.2(384)	12.4(415)	3.7(460)	2.4(466)
M.V.T	52.7	24.4	6.2	2.8	15.2	7.8	1.8	0.9
Ort.T	33.1	17.1	6.4	3.8	5.4	3.2	1.2	0.8
h, v.T	1.3	0.8	0.8	0.8	0.4	0.3	0.2	0.3
max m	128	128	128	128	128	128	128	128
Unroll	P(3,5)	P(3,5)	P(2,5)	P(2,5)	P(3,5)	P(3,5)	P(3,5)	P(2,5)
Com.M	Send	Send	Send	Send	Send	Send	Send	Send
Ort.M	CGS	CGS	CGS	CGS	CGS	CGS	CGS	CGS
Pre.M	BILU	BILU	BILU	BILU	BILU	BILU	BILU	BILU
T1/T2	13.2	17.3	36.9	52.2	3.89	4.43	9.27	15.0

(c) 問題 3

R	1.0				100.0			
PE	8	16	64	128	8	16	64	128
T1(itr.)	250.6(1265†)	149.4(†)	90.8(†)	80.9(†)	124.2(626††)	72.1(††)	44.7(††)	39.3(††)
T2(itr.)	81.9(288)	42.5(300)	7.3(417)	4.7(417)	45.7(176)	25.3(199)	11.3(306)	4.3(272)
M.V.T	52.0	26.4	2.9	2.0	32.5	17.8	5.9	2.4
ort.T	20.9	11.5	3.0	2.0	5.5	3.4	3.8	1.0
h, v.T	0.9	0.5	0.3	0.3	0.6	0.3	0.3	0.2
max m	128	128	128	128	128	128	128	128
Unroll	P(2,7)	P(2,7)	P(1,7)	P(1,7)	P(2,7)	P(1,7)	P(1,7)	P(1,7)
Com.M	Isend	Isend	Isend	Isend	Isend	Isend	Irecv	Isend
Ort.M	MGS	CGS	CGS	CGS	MGS	CGS	CGS	CGS
Pre.M	BILU	BILU	$I+B$	$I+B$	BILU	BILU	BILU	BILU
T1/T2	3.05	3.51	12.4	17.2	2.71	2.84	3.95	9.13

- 問題 3
  領域 $\Omega = [0,1] \times [0,1] \times [0,1]$ における楕円型偏微分方程式の境界値問題

$$\begin{aligned} -u_{xx} - u_{yy} - u_{zz} + Ru_x &= g(x,y,z) \\ u(x,y)|\partial\Omega &= 0.0 \end{aligned}$$

  右辺は厳密解が $u = e^{xyz}\sin(\pi x)\sin(\pi y)\sin(\pi z)$ となるように定める．領域を 80 × 80 × 80 のメッシュで 7 点中心差分によって離散化した．得られた連立一次方程式の係数行列の次元は 512,000 である．R については，$R = 1$，100 の 2 種類．

[結果]

SR2201 における，各問題の実行時間 (単位は秒) 及び自動選択された方式を表 **6** に示す．なお表 6 の主要な最左欄の語句の説明は次の通りである．

T1 ： 自由入手ができる並列反復解法ライブラリ PETSc[18] が標準で採用しているのと同じパラメタ (プリフェッチなし，リスタート :30，`MPI_Allgather`，IRCGS, 前処理なし) を設定した場合の実行時間．

T2 ： 自動チューニングを行う場合の実行時間．(自動チューニング時間を含む)

M.V.T ： 行列-ベクトル積にかかった時間．

Unroll ： アンローリング方式．例えば P(2,3) はプリフェッチありで行列の列方向 2，行方向 3 展開するという意味．

Com.M ： 通信方式．Send は`MPI_Send` → `MPI_Recv` の順に使う，Isend は `MPI_Isend` → `MPI_Irecv` の順に使う，Irecv は `MPI_Irecv` → `MPI_Isend` の順に使う．

Pre.M 前処理方式．$I - B$ は行列多項式前処理，BILU はブロック不完全 LU 分解前処理．

表 6 から，各問題の性質，PE 数，問題サイズなどの要因から適するパラメタが自動選択されている．また問題の性質に大きく依存するが，自動チューニングしない場合に対して最大で 52 倍の速度向上が得られる場合がある．

5.2 密行列固有値問題ライブラリの性能

5.2.1 三重対角化ルーチン `ILIB_TriRed` の性能

`ILIB_TriRed` での自動チューニングのパラメタは，

- 通信方式 = { トーナメント方式 ,`MPI_ALLREDUCE` }
- 行列-ベクトル積のアンローリング
  = { なし，2 段，3 段，· · · ，6 段，8 段，16 段 }
- 行列更新のアンローリング
  = { なし，2 段，3 段，· · · ，6 段，8 段，16 段 }

の 3 種類である．ここで “トーナメント方式” とは，1 対 1 通信を用いた二分木通信形態でベクトルリダクションを実現する方式であり，`MPI_ALLREDUCE` は MPI の関数を利用する方式である．

表 **7** は `ILIB_TriRed` において，自動チューニングの結果得られたパラメタと自動チューニングの実行時間を示している．表 7 から，チューニング時間は並列計算機の性能に依存し 約 1 – 33 時間であり，並列計算機の構成 PE 台数 (4PE 対 32PE)，ハードウエアの違い (SR2201 対 SR8000)，および 共有メモリ構成か分散メモリ構成か (SR8000, 1 ノード (8IP) 対 SR8000, 4 ノード (32IP)) でチューニングパラメタが異なることが分かる．表 **8** は自動チューニングによる実行時間の差を示している．

また `ILIB_TriRed` について， not auto-tuned と表記された実行時間は，パラメタを SR2201 における我々の利用経験から得られた適当な値である通信方式 = トーナメント方式， 行列-ベクトル積=8 段，行列更新=6 段 と固定して測定したものである．

表 8(a-1)，(a-2) から，SR2201 では 問題サイズが十分に大きくなると自動チューニングの効果が 1.6 – 1.8 倍程度ある．また表 8(b-1)，(b-2) から SR8000 において 共有メモリ構成でも分散メモリ構成でも問題サイズが大きくなると， 自動チューニングの効果が 1.2 – 1.3 倍程度ある．

5.2.2 直交化ルーチン `ILIB_MGSAO`

`ILIB_MGSAO` はブロック化を施した MGS 法を用いている．MGS 法はブロック化を施すことで，演算カーネルは BLAS3 演算，すなわち 3 重ループをなす．ここではループ長を短くしない目的で，最内ループの展開を行わない．またブロックアルゴリズムの特性から，計算に必要な直交化をブロック幅の個数だけ先に行いその他の直交化を一度にまとめて行うが，この計算に必要な直交化を行っている PE を枢軸 PE とよぶ．このことから計算カーネルは，枢軸 PE 用とその他の PE 用の 2 種が存在する．

以上のことから自動チューニングパラメタは

- ブロック幅 = { 1，2，3，· · · ，6，8，16 }
- 枢軸 PE の最外ループのアンローリング
  = { なし，2 段，3 段，4 段 }
- 枢軸 PE の第 2 ループのアンローリング
  = { なし，2 段，3 段，· · · ，6 段，8 段，16 段 }
- 他 PE の最外ループのアンローリング
  = { なし，2 段，3 段，4 段 }
- 他 PE の第 2 ループのアンローリング
  = { なし，2 段，3 段，· · · ，6 段，8 段，16 段 }

の 5 種類となる．

表 7 HITACHI SR2201 , SR8000 におけるチューニング済パラメタ (三重対角化ルーチン)
Table 7 The auto-tuned parameters on the HITACHI SR2201 and SR8000. (Tri-diagonalization routine)

(a-1) $PE = 4$ (PE Grid:2 × 2) の場合 (SR2201 , 分散メモリ構成)

問題サイズ	通信方式	行列-ベクトル積	行列更新
100	MPI_ALLREDUCE	6 段	3 段
200	トーナメント方式	8 段	4 段
300	トーナメント方式	8 段	6 段
400	トーナメント方式	5 段	2 段
500	トーナメント方式	8 段	5 段
600	トーナメント方式	5 段	6 段
700	トーナメント方式	8 段	6 段
800	トーナメント方式	3 段	3 段
900	トーナメント方式	8 段	4 段
1000	トーナメント方式	5 段	5 段
2000	トーナメント方式	5 段	6 段
3000	トーナメント方式	5 段	5 段
4000	トーナメント方式	3 段	3 段
5000	MPI_ALLREDUCE	5 段	5 段
6000	MPI_ALLREDUCE	5 段	5 段
7000	MPI_ALLREDUCE	5 段	5 段
8000	MPI_ALLREDUCE	3 段	2 段
Tuning Time	118401 [秒]	(32.8 [時間])	

(b-1) 1 ノード (8IP) (PE Grid:2 × 4) の場合 (SR8000, ノード内並列 , 共有メモリ構成)

問題サイズ	通信方式	行列-ベクトル積	行列更新
100	MPI_ALLREDUCE	なし	なし
200	MPI_ALLREDUCE	4 段	なし
300	MPI_ALLREDUCE	8 段	なし
400	MPI_ALLREDUCE	4 段	なし
500	MPI_ALLREDUCE	5 段	なし
600	MPI_ALLREDUCE	6 段	3 段
700	MPI_ALLREDUCE	6 段	なし
800	MPI_ALLREDUCE	6 段	3 段
900	MPI_ALLREDUCE	6 段	なし
1000	MPI_ALLREDUCE	6 段	3 段
2000	MPI_ALLREDUCE	6 段	なし
3000	MPI_ALLREDUCE	6 段	なし
4000	MPI_ALLREDUCE	6 段	なし
5000	MPI_ALLREDUCE	4 段	なし
6000	MPI_ALLREDUCE	4 段	なし
7000	MPI_ALLREDUCE	6 段	なし
8000	MPI_ALLREDUCE	6 段	なし
Tuning Time	16325 [秒]	(4.5 [時間])	

(a-2) $PE = 32$ (PE Grid:4 × 8) の場合 (SR2201 , 分散メモリ構成)

問題サイズ	通信方式	行列-ベクトル積	行列更新
100	MPI_ALLREDUCE	6 段	16 段
200	MPI_ALLREDUCE	4 段	5 段
300	MPI_ALLREDUCE	4 段	4 段
400	MPI_ALLREDUCE	6 段	3 段
500	MPI_ALLREDUCE	6 段	4 段
600	MPI_ALLREDUCE	6 段	4 段
700	MPI_ALLREDUCE	8 段	3 段
800	MPI_ALLREDUCE	5 段	3 段
900	MPI_ALLREDUCE	4 段	3 段
1000	MPI_ALLREDUCE	5 段	3 段
2000	MPI_ALLREDUCE	5 段	5 段
3000	MPI_ALLREDUCE	8 段	5 段
4000	MPI_ALLREDUCE	5 段	5 段
5000	MPI_ALLREDUCE	8 段	5 段
6000	MPI_ALLREDUCE	5 段	5 段
7000	MPI_ALLREDUCE	5 段	5 段
8000	MPI_ALLREDUCE	3 段	3 段
Tuning Time	15555 [秒]	(4.3 [時間])	

(b-2) 4 ノード (32IP) (Node Grid:2 × 2) の場合 (SR8000, ノード間並列 , 分散メモリ構成)

問題サイズ	通信方式	行列-ベクトル積	行列更新
100	トーナメント方式	なし	2 段
200	トーナメント方式	なし	なし
300	トーナメント方式	なし	2 段
400	トーナメント方式	なし	なし
500	トーナメント方式	なし	なし
600	トーナメント方式	なし	なし
700	トーナメント方式	なし	なし
800	トーナメント方式	4 段	なし
900	トーナメント方式	4 段	なし
1000	トーナメント方式	6 段	なし
2000	MPI_ALLREDUCE	6 段	4 段
3000	MPI_ALLREDUCE	6 段	4 段
4000	MPI_ALLREDUCE	4 段	16 段
5000	MPI_ALLREDUCE	4 段	16 段
6000	MPI_ALLREDUCE	4 段	16 段
7000	MPI_ALLREDUCE	6 段	16 段
8000	MPI_ALLREDUCE	6 段	16 段
Tuning Time	4443 [秒]	(1.2 [時間])	

表 8 HITACHI SR2201，SR8000 での実行時間．単位は秒．(三重対角化ルーチン)
Table 8 Execution time on the HITACHI SR2201 and SR8000. Unit is in second. (Tri-diagonalization routine)

(a-1) 4PE (PE Grid:2×2) の場合
(HITACHI SR2201，分散メモリ構成)

問題サイズ	ILIB_TriRed1 (not auto-tuned)	ILIB_TriRed2 (auto-tuned)	ILIB_TriRed1 /ILIB_TriRed2
100	0.056	0.056	1.0
200	0.131	0.133	0.98
400	0.435	0.475	0.91
800	3.732	2.454	1.5
1000	3.817	3.785	1.00
2000	28.165	26.937	1.04
4000	411.666	242.010	1.7
8000	3589.175	1962.512	1.8

(a-2) 32PE (PE Grid:4×8) の場合
(HITACHI SR2201，分散メモリ構成)

問題サイズ	ILIB_TriRed1 (not auto-tuned)	ILIB_TriRed2 (auto-tuned)	/ILIB_TriRed1 /ILIB_TriRed2
100	0.108	0.106	1.01
200	0.250	0.240	1.04
400	0.514	0.516	0.99
800	1.207	1.228	0.98
1000	1.654	1.687	0.98
2000	5.930	5.886	1.00
4000	32.961	32.124	1.02
8000	427.267	254.937	1.6

(b-1) 1 ノード (8IP) (PE Grid:2×4) の場合
(HITACHI SR8000, ノード内並列，共有メモリ構成)

問題サイズ	TriRed1 (not auto-tuned)	TriRed2 (auto-tuned)	TriRed1 /TriRed2
100	0.024	0.022	1.09
200	0.053	0.049	1.08
400	0.162	0.145	1.11
800	0.678	0.587	1.15
1000	1.155	0.988	1.16
2000	7.098	5.595	1.26
4000	50.451	39.263	1.28
8000	389.297	308.307	1.26

(b-2) 4 ノード (32IP) (Node Grid:2×2) の場合
(HITACHI SR8000, ノード間並列，分散メモリ構成)

問題サイズ	TriRed1 (not auto-tuned)	TriRed2 (auto-tuned)	TriRed1 /TriRed2
100	0.038	0.036	1.05
200	0.077	0.072	1.06
400	0.176	0.162	1.08
800	0.490	0.450	1.08
1000	0.714	0.648	1.10
2000	2.806	2.345	1.19
4000	14.957	11.392	1.31
8000	102.369	75.398	1.35

表 9 HITACHI SR2201 におけるチューニング済パラメタ (MGS 直交化ルーチン)
Table 9 The auto-tuned parameters on the HITACHI SR2201. (MGS orthogonalization routine)

(a) 4PE の場合

問題サイズ	ブロック幅	枢軸 PE 最外ループ	枢軸 PE 第 2 ループ	他 PE 最外ループ	他 PE 第 2 ループ
100	6	3 段	4 段	2 段	6 段
200	6	なし	なし	2 段	6 段
300	6	なし	なし	2 段	6 段
400	6	なし	なし	2 段	3 段
500	6	なし	なし	2 段	6 段
600	6	なし	なし	2 段	6 段
700	6	なし	なし	2 段	6 段
800	6	なし	なし	2 段	3 段
900	6	なし	なし	2 段	6 段
1000	6	なし	なし	2 段	6 段
2000	6	なし	6 段	2 段	3 段
3000	6	なし	16 段	2 段	6 段
4000	6	なし	なし	なし	3 段
Tuning Time	91407 [秒]	(25.3 [時間])			

(b) 16PE の場合

問題サイズ	ブロック幅	枢軸 PE 最外ループ	枢軸 PE 第 2 ループ	他 PE 最外ループ	他 PE 第 2 ループ
100	5	4 段	8 段	4 段	2 段
200	5	なし	なし	2 段	5 段
300	5	なし	なし	なし	4 段
400	5	なし	なし	2 段	5 段
500	6	なし	なし	2 段	6 段
600	6	なし	なし	2 段	6 段
700	5	なし	なし	2 段	5 段
800	5	なし	なし	2 段	5 段
900	6	なし	なし	2 段	6 段
1000	5	なし	なし	2 段	5 段
2000	6	なし	8 段	2 段	3 段
3000	6	なし	6 段	2 段	6 段
4000	6	なし	8 段	2 段	6 段
Tuning Time	26607 [秒]	(7.3 [時間])			

表 10 HITACHI SR2201 での実行時間 . 単位は秒 . (MGS 直交化ルーチン)
Table 10 Execution time on the HITACHI SR2201. Unit is in second. (MGS orthogonalization routine)

(a) 4 PE の場合

問題サイズ	MGSAO1 (not AT)	MGSAO2 (AT)	MGSAO1 /MGSAO2
100	0.017	0.010	1.70
200	0.103	0.051	2.01
400	0.733	0.389	1.88
800	6.379	3.174	2.00
1000	12.598	5.378	2.34
2000	108.665	48.879	2.22
4000	892.796	420.207	2.12

(b) 16 PE の場合

問題サイズ	MGSAO1 (not AT)	MGSAO2 (AT)	MGSAO1 /MGSAO2
100	0.014	0.012	1.16
200	0.047	0.033	1.42
400	0.294	0.136	2.16
800	1.932	0.859	2.24
1000	3.865	1.668	2.31
2000	32.407	14.310	2.26
4000	260.168	119.003	2.18

表 **9** は SR2201 での自動チューニングの結果得られたパラメタである．表 9 では，問題サイズ 100，2000–4000 で最適なパラメタが大きく変動するが，これは自動チューニング機構が問題サイズの違いから生じるループ長やハードウエア特性の変化による最適な実装方式の違いを吸収した結果と思われる．

表 **10** は SR2201 での実行時間である．ここで，図中の表記 MGSAO1(not AT) は適当なパラメタ (ブロック幅=4，枢軸 PE の最外ループ=4 段，枢軸 PE の第 2 ループ=8 段，他 PE 最外ループ=4 段，他 PE 第 2 ループ=8 段) を設定した場合の実行時間である．

表 10 の結果から，適当な固定パラメタを設定した場合に対して自動チューニングにより，約 1.1 倍 – 2.3 倍の速度向上が達成できる場合がある．このことから ILIB_MGSAO のようなブロック化されたルーチンにおいて性能パラメタを固定して実装する従来の方式は，性能の観点から好ましくないことが推察できる．

5.2.3 Householder 逆反復法ルーチン ILIB_HouseInv の性能

ILIB_HouseInv の演算カーネルは BLAS1 演算であるが，逆変換すべきベクトルの数だけ BLAS1 演算があるので，2 重ループの構造をなす．ここではこの 2 重ループの最外ループに対してアンローリングを行う．したがって自動チューニングのパラメタとしては，

- 行列更新のアンローリング
  = { なし，2 段，3 段，$\cdots$，6 段，8 段，16 段 }

の 1 種類である．

表 **11** は SR2201 における自動チューニング済のパラメタリストである．

表 **12** は SR2201 での実行時間の結果を示している．ここで図中の表記 HIT(not AT) は，(行列更新=なし) とした場合，すなわちこの最適化をコンパイラに任せた場合の実行時間であり，HIT(AT) は自動チューニングによりパラメタを決定した場合の実行時間である．

表 12 の結果から，自動チューニングを行うことで約 1.2 倍 – 1.4 倍の速度向上を達成できる場合がある．

表 **11** HITACHI SR2201 におけるチューニング済パラメタ (Householder 逆変換ルーチン)
Table 11 The auto-tuned parameters on the HITACHI SR2201. (Householder inverse transformation routine)

(a) 4 PE の場合

問題サイズ	行列更新
100	6 段
200	3 段
300	5 段
400	3 段
500	5 段
600	6 段
700	5 段
800	3 段
900	5 段
1000	6 段
2000	3 段
3000	6 段
4000	3 段
Tuning Time	6246 [秒] (1.73 [時間])

(b) 16 PE の場合

問題サイズ	行列更新
100	6 段
200	4 段
300	4 段
400	3 段
500	3 段
600	6 段
700	8 段
800	3 段
900	5 段
1000	6 段
2000	3 段
3000	6 段
4000	3 段
Tuning Time	3129 [秒] (0.86 [時間])

6. おわりに

本論文では並列数値計算ライブラリにおける特化型自動チューニング機構の提案を行い，また実際にその機能を付加した並列数値計算ライブラリ ILIB の性能評価を行った．従来の汎用型自動チューニング機構との違いは，最適化が局所的 (演算カーネルのみ) を対象としているのではなく，ライブラリ全体を対象にした広域最適化ができることである．この広域最適化により，通信粒度と計算性能の両方に影響するブロック幅などのパラメタ調整も可能となる．

自動チューニング機能を付加することにより，連立一次方程式ライブラリでは密行列 LU 分解ルーチン ILIB_LU において理論性能に対する効率 89%を達成した．さらに更新領域を自動調節する疎行列用 LU 分解ルーチン ILIB_RLU においては，従来のスパースソルバと比較しても高い性能である理論性能に対する効率 72%を達成した．また疎行列反復解法の一つである GMRES(m) 法ルーチン ILIB_GMRES ではより柔軟なパラメタ選択が可能となり，その結果 52 倍の高速化が達成できる場合があることがわかった．性能向上に関し，固有値問題ライブラリにおける三重対角化ルーチン ILIB_TriRed では 1.8 倍程度の高速化，直交化ルーチン ILIB_MGSAO では 2.3 倍程度の高速化，そして Householder 逆変換ルーチン ILIB_HouseInv では 1.4 倍程度の高速化が達成される場合があることが判

表 12 HITACHI SR2201 での実行時間．単位は秒．(Householder 逆反復法ルーチン)
Table 12 Execution time on the HITACHI SR2201. Unit is in second. (Householder inverse transformation routine)

(a) 4PE の場合

問題サイズ	HIT1 (not AT)	HIT2 (AT)	HIT1 /HIT2
100	0.010	0.007	1.42
200	0.067	0.051	1.31
400	0.473	0.375	1.26
800	3.469	2.837	1.22
1000	6.624	5.200	1.27
2000	49.46	39.51	1.25
4000	384.9	312.9	1.23

(b) 16PE の場合

問題サイズ	HIT1 (not AT)	HIT2 (AT)	HIT1 /HIT2
100	0.005	0.004	1.25
200	0.034	0.026	1.30
400	0.234	0.189	1.23
800	1.735	1.436	1.20
1000	3.332	2.660	1.25
2000	24.81	19.86	1.24
4000	192.96	158.93	1.21

明した．これらの 3 種のルーチンの演算カーネルは BLAS の分類において異なっていることから，科学技術計算におけるある種のベンチマークとみなせる．したがってその他の計算についても，同様の高速化が得られる可能性があるといえよう．

得られた実験的な結果は，当然のことながらコンパイラの最適化能力に大きく依存している．最適化が十分になされるならば，本論文で行ったような演算カーネルに対する自動チューニング (アンローリング段数の決定) による速度向上はほとんど得られないはずである．このことから，コンパイラによる最適化は現状では不十分であるといわざるを得ない．ライブラリ全体の性能はコンパイラの最適化性能，キャッシュ容量等のハードウエア特性，および OS による処理に依存し，きわめて「複雑な」振舞いを示す．したがって実際の計測なしで最適な性能パラメタを推定すること—すなわち測定なしで最適なコードを生成すること—は不可能であろう．不十分なコンパイラの最適化，複雑なハードウエア機構，および異なる OS，で構成される並列計算機環境において，できるだけ高性能なライブラリを構築する方法の一つとして本論文で示した自動チューニングは重要な技法になるものと考えられる．

今後の課題として，(1) 提案した自動チューニング機構と同等の機能が得られる言語やトランスレータを実現して適用範囲を広げること ;(2) 自動チューニングに関する時間を削減するための計算モデルや手法の開発 ;が必要である．

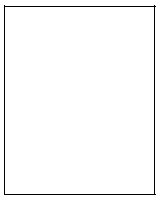
なお ILIB に関する情報は，http://www.hints.org/ から入手可能である．

参 考 文 献

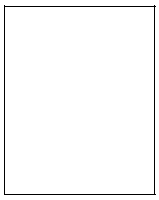
1) Dongarra, J. J., Croz, J. D., Hammarling, S. and Duff, I.: A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Transactions on Mathematical Software*, Vol. 16, No. 1, pp. 1–17 (1988).
2) Bilmes, J., Asanović, K., Chin, C.-W. and Demmel, J.: Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, *Proceedings of International Conference on Supercomputing 97*, Vienna, Austria, pp. 340–347 (1997).
3) Whaley, R. C. and Dongarra, J. J.: Automatically Tuned Linear Algebra Software, ATLAS project, http://www.netlib.org/atlas/index.html.
4) 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).
5) Kuroda, H. and Kanada, Y.: Performance of Automatically Tuned Parallel Sparse Linear Equations Solver, *IPSJ SIG Notes, 99-HPC-76*, pp. 13–18 (1999). in Japanese.
6) 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).
7) Saad, Y., Malevsky, A. V., Lo, G. C., Kuznetsov, S., Sosonkina, M. and Moulitsa, I.: PSPARSLIB Project. http://www.cs.umn.edu/Research/arpa/p_sparslib/psp-abs.html.
8) Saad, Y. and Lo, G.: PSPARSLIB Users Manual: A Portable Library of parallel Sparse Iterative Solvers (1996).
9) 黒田久泰, 片桐孝洋, 佃良生, 金田康正: 自動チューニング機能付き並列数値計算ライブラリ構築の試み—対称行列用の連立一次方程式ソルバを例にして—, 情報処理学会第 57 回全国大会予稿集, Vol.1, pp. 1–10 – 1–11 (1998).
10) Kuroda, H., Katagiri, T. and Kanada, Y.: Performance of Automatically Tuned Parallel GMRES(m) Method on Distributed Memory Machines, *Proceedings of Vector and Parallel Processing (VECPAR) 2000*, Porto, Portugal, pp. 251 – 264 (2000).
11) Katagiri, T., Kuroda, H. and Kanada, Y.: A Methodology for Automatically Tuned Paral-

lel Tri-diagonalization on Distributed Memory Parallel Machines, *Proceedings of Vector and Parallel Processing (VECPAR) 2000*, Porto, Portugal, pp. 265 – 277 (2000).
12) 片桐孝洋, 黒田久泰, 大澤清, 金田康正: ILIB :自動チューニング機能付き並列数値計算ライブラリとその性能評価, JSPP'2000 論文集, pp. 27–34 (2000).
13) Katagiri, T.: *A Study on Large Scale Eigensolvers for Distributed Memory Parallel Machines*, Ph.D Thesis, The University of Tokyo (2000).
14) 大澤清, 片桐孝洋, 黒田久泰, 金田康正: ILIB_RLU 疎行列を密行列として扱う自動チューニング機能付き LU 分解ルーチンの性能評価, *IPSJ SIG Notes, 00-HPC-82*, pp.25–30 (2000).
15) 大澤清: 自動チューニング機能付きスパースダイレクトソルバ ILIB_RLU の性能評価, スーパーコンピューティングニュース, Vol. 2, No. 5, pp. 23–36 (2000). 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部門).
16) 片桐孝洋, 金田康正: 並列固有値ソルバーの実現とその性能, *IPSJ SIG Notes, 97-HPC-69*, pp. 49–54 (1997).
17) (株) 日立製作所: スーパーテクニカルサーバ HITACHI SR8000 LINPACK 性能の ご紹介, スーパーコンピューティングニュース, Vol. 1, No. 2, pp. 72–79 (1999). 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部門).
18) Balay, S., Gropp, W., McInnes, L. and Smith, B.: *PETSc 2.0 Users Manual*, ANL-95/11 – Revision 2.0.24, Argonne National Laboratory (1999).
19) Zhang, Q. and Hisada, T.: Analysis of Fluid-Structure Interaction Problems under Large Domain Changes by ALE Finite Element Method, *Theoritical and Applied Mechanics*, Vol. 47, pp. 167–174 (1998).
20) 山本有作, 猪貝光祥, 直野健: 分散メモリ型並列計算機向けスパース対称行列ソルバの開発と評価, 情報処理学会論文誌, Vol. 41, No. 5, pp. 1567–1576 (2000).

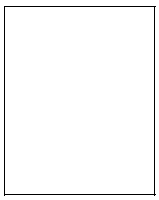
(平成？年？月？日受付)
(平成？年？月？日採録)



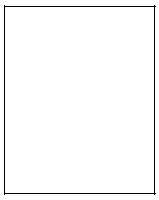
片桐 孝洋 (正会員)
1973 年生．1994 年 豊田工業高等専門学校情報工学科卒業．1996 年 京都大学工学部情報工学科卒業．2001 年 東京大学大学院理学系研究科情報科学専攻博士課程修了．博士 (理学)．2001 年 4 月より日本学術振興会特別研究員-PD．並列計算機による行列計算アルゴリズムの研究に従事．大規模固有値問題，並列数値計算に興味を持つ．日本応用数理学会，SIAM 各会員．



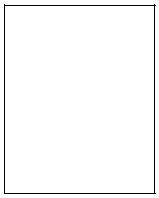
黒田 久泰 (正会員)
1970 年生．1993 年 名古屋大学理学部物理学科卒業．1995 年 京都大学大学院工学研究科応用システム科学専攻修士課程修了．2000 年 東京大学大学院理学系研究科情報科学専攻博士課程満期退学．2000 年 4 月より 東京大学情報基盤センタースーパーコンピューティング研究部門 助手．ACM，SIAM，人工知能学会各会員．



大澤 清
1974 年生．1998 年 東京工業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修士課程修了．修士 (理学)．2001 年 4 月より信陽ビジネスサービス勤務．在学時は分散メモリ型並列計算機による並列数値計算の研究に従事．



工藤 誠 (学生会員)
1978 年生．2001 年 東京大学理学部情報科学科卒業．2001 年 4 月より東京大学大学院情報理工学系研究科コンピュータ科学専攻修士課程入学．並列数値計算，特に連立一次方程式の解法に興味を持つ．



金田　康正（正会員）

1949 年生．1973 年東北大学理学部物理第二学科卒業．1978 年東京大学理学系研究科博士課程修了．理学博士．1978 年名古屋大学プラズマ研究所助手．1981 年東京大学大型計算機センター助教授，教授を経て現在東京大学情報基盤センター教授．その間英国ケンブリッジ大学計算機研究所客員研究員．昭和 58 年度及び平成 10 年度情報処理学会論文賞，平成 6 年度情報処理学会 Best Author 賞受賞．著書「π のはなし」（東京図書），編著「Trends in Supercomputing」（World Scientific）．日本応用数理学会，プラズマ・核融合学会，ACM，SIAM 各会員．研究テーマは「大規模数値計算」および「研究の研究」．