自動チューニング機能付き並列数値計算ライブラリの開発
— 知識発見手法の適用 —

片桐 孝洋¶　　黒田 久泰¶　　金田 康正‖
¶ 東京大学大学院理学系研究科情報科学専攻
‖ 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

概要

本稿では，自動チューニング機能付き並列数値計算ライブラリの開発とその性能について述べる．自動チューニング機能を付加することで，ユーザが指定しなくてはならないパラメタが少なくなるばかりか，高性能な並列数値計算ライブラリが構築可能となる．またこれらライブラリを構築するためには，チューニング情報からの知識発見が必要になることを述べる．

1　はじめに

我々は，自動チューニング機能を含む性能に関する最適化をユーザが直接指定することなしにライブラリ自体が行うという，新しい概念のライブラリ開発を行っている．特に数値計算においては，以下に示す機能を有する数値計算ライブラリの開発を目指している．

- 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（並列計算機を用いる場合）
- 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

我々はこの思想に基づき，直接法（LU 分解，固有値計算）や反復法（GMRES 法などの疎行列を係数行列とする連立 1 次方程式の解法）などの ライブラリ群を現在開発中である．

分散メモリ型並列計算機向きの自由入手可能なライブラリとして ScaLAPACK[1] などが既に存在する．しかしながらこれらのライブラリはユーザが定義しなくてはいけないパラメタが非常に多く，我々の思想とは異なる．一方で，数値計算において自動チューニングを行うソフトウエアとして PHiPAC[2] や ATLAS[3] などがある．しかしながらこれらのソフトウエアはまだ研究段階であり，並列計算を考慮に入れていなかったり，行列積などの比較的簡単な計算の最適化のみを行うことができる．そこで我々は，実用的な並列数値計算ライブラリを念頭において並列対称密行列固有値ライブラリ [4] や並列疎行列連立 1 次方程式ライブラリ [5] の開発を行ってきた．本稿ではまず，これら実用的なライブラリ群に自動チューニング機能を付加したライブラリの開発とその性能について述べる．最後に，自動チューニング効率を向上させるには知識発見手法の適用が必要であることを述べる．

2　並列ライブラリの機能

2.1　連立 1 次方程式

我々が開発している並列連立 1 次方程式ライブラリ [5] は，式 (1) に示される連立 1 次方程式の解ベクトル x を求めることができる．

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

ここで，係数行列 $A \in \Re^{n \times n}$ は 実数の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，反復法の 1 種である GMRES 法などの反復解法を用いている．

2.2 固有値問題

現在開発中の並列固有値ライブラリ [4] は，式 (2) に示される標準固有値問題の固有分解を行うことができる．

$$AX = \Lambda X. \tag{2}$$

ここで，係数行列 $A \in \Re^{n\times n}$ は実数の対称密行列である．また 固有値 λ_i $(i = 1, 2, ..., n) \in \Re$ から構成される行列を $\Lambda = \mathrm{diag}(\lambda_i)$ とする．さらに固有ベクトル 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．解法として，直接法と反復法を混合した Householder-bisection 法を用いている．

3 自動チューニング機能の説明

3.1 並列密行列ライブラリ

3.1.1 並列密行列ライブラリの自動チューニング項目

並列密行列ライブラリ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) 演算カーネルにおけるアンローリング段数

(ii) データ分散方式

(iii) 通信処理の実装方式

密行列を係数行列にもつライブラリの場合，多くの並列システムでは演算時間が通信時間に対してネックとなる場合が多い[1] ．この理由から，演算カーネルの自動チューニング機能 (i) が密行列ライブラリでは特に重要であるといえる．

また並列処理においては，(ii) の係数行列 A をどのように PE(Processor Element) に分散させるかが性能に大きく影響するので，適する方式を自動的に選択する必要がある．

最後に固有値問題のように密行列でありながら通信回数が多いライブラリに関しては，(iii) の通信処理の実装方式が性能に大きく影響することがあるので，この自動選択も必要となる．

3.1.2 並列密行列ライブラリの自動チューニングの特長

密行列ライブラリの場合は係数行列 A が密であり規則的であるので，チューニングの要因が係数行列 A に依らないことが多い．このことは，ライブラリ実行前に一度チューニングを行えばその情報を何度も利用できることを意味している．すなわち，この場合自動チューニングは静的（ライブラリ実行前）に行えることに注意する．

3.2 並列疎行列ライブラリ

3.2.1 並列疎行列ライブラリの自動チューニング項目

並列疎行列ライブラリ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) 演算カーネルにおける通信方式

(ii) 前処理アルゴリズムの決定

(iii) その他の計算アルゴリズムの決定

疎行列ライブラリは密行列ライブラリと大きく異なり，演算時間よりも通信時間が一般に多くなる．そのため (i) の演算カーネルにおける通信方式を自動選択することが性能の観点では非常に重要である．

[1] 専用並列機を用いて数千台規模の超並列処理を行う場合や計算機クラスタなどの通信処理が演算性能に対して極端に悪い並列システム，また問題サイズが極めて小さい場合は，通信時間が無視できなくなることに注意する．

また 疎行列の解法は反復解法が用いられることが多く，この場合係数行列 A を反復に入る前に反復処理が収束しやすい行列に変換することがよく行われる．これを前処理と呼ぶが，疎行列反復解法では多くの前処理が提案されている．ところがどの前処理を適用したら最も効率が良いのかは問題に依存しており，一般には実際に実行してみないとわからない．この理由から本ライブラリでは，実際に実行した情報から前処理の方式を自動選択する．

さらに，反復解法では反復処理中に並列性能に影響する計算処理 (iii) が必要なことがある．例えば直交化処理がその一例である．これらの反復解法に必要な計算アルゴリズムも自動選択をする必要がある．

3.2.2 並列疎行列ライブラリの自動チューニングの特長

疎行列ライブラリの場合は係数行列 A が疎であり，一般的に非ゼロ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 の非ゼロ要素の位置に依存することが多い．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ている．結果として自動チューニングは動的（ライブラリ実行時）に行わざるを得ないことになる．

4 実験結果

この章では，本稿で示した自動チューニングの機能を実装した並列数値計算ライブラリを日立 SR2201 および日立 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[2]．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 PE8 台の共有メモリ構成となっている．ノード間は三次元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[3]．

4.1 密行列固有値問題

現時点では対称密行列の固有分解を行う並列ライブラリ全体について，自動チューニング機能は実装されていない．そのため密行列の固有分解において必須な Householder 法による三重対角化ルーチンに自動チューニング機能を付加した結果を報告する．三重対角化ルーチン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, ···, 8 段, 16 段 }
- 行列更新のアンローリング = { なし，2 段, 3 段, ···, 8 段, 16 段 }

の 3 種類である．

表 1 は我々の三重対角化ルーチン (以降 Our TRD) 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 1 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 時間 – 約 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，8PE 対 SR8000 4 ノード) でチューニングパラメタが異なることが分かる．

表 2 は自動チューニングによる実行時間の差を示している．ここで Our TRD が十分に高速であることを示すため，コードが公開されている ScaLAPACK[1] の 三重対角化ルーチン (以降 SLP TRD) との比較ものせた．本評価における SLP TRD は，日立製作所がチューニングして提供した PVM 版 SLP Version 1.2 [6] を用いた．

さて SLP TRD は， ブロックサイズ BL が性能に大きな影響を与える．文献 [6] によると SR2201 では，問題サイズ n が 4000 以下ならば $BL = 60$，問題サイズ n が 4000 より大きいなら $BL = 100$ という指

[2] 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 32PE 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．

[3] 東京大学情報基盤センターが所有している 128 ノード (8PE/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．

表 1: 密行列ライブラリにおける自動チューニングの結果とチューニング時間

(a-1) $PE = 4$ の場合 (SR2201)

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
チューニング時間	118401 [秒]	(32.8 [時間])	

(a-2) $PE = 32$ の場合 (SR2201)

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
チューニング時間	15555 [秒]	(4.3 [時間])	

(b-1) 1 ノード, 8PE 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
チューニング時間	16325 [秒]	(4.5 [時間])	

(b-2) 4 ノード 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
チューニング時間	4443 [秒]	(1.2 [時間])	

針が示されている．そこで ブロックサイズ $BL = \{40, 60, 80, 100, 120\}$ の 5 通りを全て実行し，最も高速であった実行時間をのせた．また Our TRD について not auto-tuned と表記された実行時間は，パラメタを適当な値である 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 2(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 倍程度あることがわかる．さらに SLP TRD と比べると PE 数が増加すると Our TRD の方が高速になってくることや，十分問題サイズが小さい場合には Our TRD の方が 2–5 倍程度高速なことがわかる．一方，表 2(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 倍程度あることがわかる．

4.2 疎行列連立 1 次方程式

疎行列反復解法の 1 種である GMRES 法を用いたライブラリ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) = { 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) = { なし，2 段, 3 段, 4 段, 8 段 }
- 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 = { 古典的 Gram-Schmidt(CGS), 修正 Gram-Schmidt(MGS)}
- 前処理行列 = { なし, 行列多項式前処理，ブロック不完全 LU 分解前処理 }

の 6 種類である．また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．

表 2: 密行列ライブラリにおける性能比較．単位は秒.

(a-1) $PE = 4$ の場合 (SR2201)

問題サイズ	SLP TRD (Grid, BL)	Ours1 (not auto-tuned)	SLP /Ours1	Ours2 (auto-tuned)	Ours1 /Ours2	SLP /Ours2
100	0.02 (1×4, 100)	0.056 (2×2)	0.35	0.056 (2×2)	1.0	0.35
200	0.48 (1×4, 100)	0.131 (2×2)	3.6	0.133 (2×2)	0.98	3.6
400	1.73 (1×4, 40)	0.435 (2×2)	3.9	0.475 (2×2)	0.91	3.6
800	6.01 (1×4, 40)	3.732 (2×2)	1.6	2.454 (2×2)	1.5	2.4
1000	9.32 (2×2, 40)	3.817 (2×2)	2.4	3.785 (2×2)	1.0	2.4
2000	41.90 (2×2, 40)	28.165 (2×2)	1.4	26.937 (2×2)	1.0	1.5
4000	231.10 (2×2, 40)	411.666 (2×2)	0.56	242.010 (2×2)	1.7	0.95
8000	1422.69 (2×2, 100)	3589.175 (2×2)	0.39	1962.512 (2×2)	1.8	0.72

(a-2) $PE = 32$ の場合 (SR2201)

問題サイズ	SLP TRD (Grid, BL)	Ours1 (not auto-tuned)	SLP /Ours1	Ours2 (auto-tuned)	Ours1 /Ours2	SLP /Ours2
100	0.09 (4×8, 100)	0.108 (4×8)	0.83	0.106 (4×8)	1.01	0.84
200	0.87 (2×16, 100)	0.250 (4×8)	3.4	0.240 (4×8)	1.04	3.6
400	2.33 (2×16, 60)	0.514 (4×8)	4.5	0.516 (4×8)	0.99	4.5
800	6.27 (2×16, 60)	1.207 (4×8)	5.1	1.228 (4×8)	0.98	5.1
1000	8.28 (2×16, 60)	1.654 (4×8)	5.0	1.687 (4×8)	0.98	4.9
2000	22.18 (4×8, 40)	5.930 (4×8)	3.7	5.886 (4×8)	1.00	3.7
4000	72.74 (4×8, 40)	32.961 (4×8)	2.2	32.124 (4×8)	1.02	2.2
8000	313.25 (4×8, 40)	427.267 (4×8)	0.73	254.937 (4×8)	1.6	1.2

(b-1) 1 ノード, 8PE の場合
(SR8000, ノード内並列，共有メモリ構成)

問題サイズ	Ours1 (not auto-tuned)	Ours2 (auto-tuned)	Ours1 /Ours2
100	0.024 (2×4)	0.022 (2×4)	1.09
200	0.053 (2×4)	0.049 (2×4)	1.08
400	0.162 (2×4)	0.145 (2×4)	1.11
800	0.678 (2×4)	0.587 (2×4)	1.15
1000	1.155 (2×4)	0.988 (2×4)	1.16
2000	7.098 (2×4)	5.595 (2×4)	1.26
4000	50.451 (2×4)	39.263 (2×4)	1.28
8000	389.297 (2×4)	308.307 (2×4)	1.26

(b-2) 4 ノード の場合
(SR8000, ノード間並列，分散メモリ構成)

問題サイズ	Ours1 (not auto-tuned)	Ours2 (auto-tuned)	Ours1 /Ours2
100	0.038 (2×2)	0.036 (2×2)	1.05
200	0.077 (2×2)	0.072 (2×2)	1.06
400	0.176 (2×2)	0.162 (2×2)	1.08
800	0.490 (2×2)	0.450 (2×2)	1.08
1000	0.714 (2×2)	0.648 (2×2)	1.10
2000	2.806 (2×2)	2.345 (2×2)	1.19
4000	14.957 (2×2)	11.392 (2×2)	1.31
8000	102.369 (2×2)	75.398 (2×2)	1.35

4.2.1 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 1000×1000 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 1 次方程式の次元は 1,000,000 である．R については，$R = 1$，1000 の 2 種類.

- 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 $128 \times 128 \times 128$ 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 1 次方程式の次元は 2,097,152 である．R については，$R = 1$，100 の 2 種類.

4.2.2 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 3 に示す．なお表 3 の最左欄の語句の説明は次の通りである．

合計時間： 自動チューニングも含めた時間

行列積時間： 行列ベクトル積にかかった時間

Unrolling： 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

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

前処理行列 $I - B$ 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 3 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ことがわかる．

表 3: 疎行列ライブラリにおける自動チューニング結果と性能 (SR2201)

(a) 問題 1

	γ=1					γ=2				
PE 台数	8	16	32	64	128	8	16	32	64	128
反復回数	19	20	20	20	21	323	322	322	322	322
合計時間	43.9	25.0	15.3	9.0	6.5	159.5	87.9	46.6	23.6	13.7
行列積時間	21.0	10.7	5.2	2.6	1.3	36.9	17.7	8.4	4.0	1.9
直交化時間	4.7	2.4	1.2	0.6	0.4	91.6	52.5	26.5	12.2	6.2
h, v 計算	0.9	0.5	0.2	0.1	0.1	7.3	3.6	1.9	1.0	0.7
リスタート	32	64	128	128	128	32	64	128	128	128
Unrolling	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)
通信方式	Send	Send	Send	Isend	Send	Send	Send	Irecv	Send	Send
直交化方式	MGS	MGS	MGS	CGS	CGS	MGS	MGS	MGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	BILU	なし	なし	なし	なし	なし

(b) 問題 2

	R=1					R=1000				
PE 台数	8	16	32	64	128	8	16	32	64	128
反復回数	3930	4086	3557	4278	4738	1213	1233	1452	1205	1563
合計時間	2145.9	1114.2	458.1	275.1	151.1	525.5	264.3	156.1	61.5	44.1
行列積時間	1000.0	509.9	218.7	124.1	64.7	309.3	156.1	90.2	34.7	21.4
直交化時間	1083.0	568.8	223.3	140.0	77.9	187.0	92.8	57.1	22.1	18.3
h, v 計算	20.9	13.9	6.0	4.5	4.2	6.6	3.7	2.3	1.3	1.7
リスタート	128	128	128	128	128	128	128	128	128	128
Unrolling	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)
通信方式	Send	Send	Send	Send	Send	Send	Send	Send	Send	Send
直交化方式	MGS	CGS	CGS	CGS	CGS	MGS	CGS	CGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	BILU	BILU	BILU	BILU	BILU	BILU

(c) 問題 3

	R=1					R=100				
PE 台数	8	16	32	64	128	8	16	32	64	128
反復回数	483	481	504	518	691	274	315	383	333	418
合計時間	562.8	281.2	146.2	74.8	28.1	326.3	186.3	111.5	48.8	31.9
行列積時間	404.7	199.7	102.4	52.2	11.2	234.1	132.3	78.4	34.0	20.5
直交化時間	102.6	51.9	28.2	13.9	11.7	46.4	28.3	19.0	7.4	6.1
h, v 計算	5.7	2.9	1.7	1.0	0.9	3.3	1.9	1.3	0.6	0.6
リスタート	64	128	128	128	128	64	128	128	128	128
Unrolling	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)
通信方式	Irecv	Irecv	Irecv	Irecv	Irecv	Irecv	Irecv	Irecv	Irecv	Irecv
直交化方式	MGS	MGS	CGS	CGS	CGS	MGS	MGS	CGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	$I - B$	BILU	BILU	BILU	BILU	BILU

5 知識発見手法の適用

本稿で示した 2 種の並列ライブラリにおいてもなお，パラメタの自動決定が困難な状況が生じる．この場合パラメタやチューニングの際の情報を基に，知識発見のプロセスを踏むことで解決をする必要があるものと考えられる．以降にこの問題を述べる．

(I) チューニングされていない項目におけるパラメタの推定

この問題は，例えば問題サイズ n が 10000, 20000 のみ自動チューニングされたパラメタがあり $n = 15000$ のパラメタを推定する場合や，$n = 10000$ までしか自動チューニングができない場合に $n = 50000$ のパラメタを推定する場合である．このような状況は，実行時間の都合からよく生じるものと考えられる．解決法として，補間や補外による推定など統計的手法を用いることが必要であろう．

(II) パラメタ相互の依存性の発見

この問題は，各パラメタ間に依存性が有るか無いかわからない場合に生じる．現在の実装では，ライブラリ作成者の経験や知識から各パラメタが独立であると仮定して自動チューニングをしている．しかし実際はパラメタ間に依存関係があるかもしれない．その場合は最適な性能を引き出すことができなくなる．この解決法として，各パラメタに対して統計的手法である主因子分析を実行し依存があるかどうか知識発見することなどが挙げられる．

(III) パラメタの探索の最適化

この問題は (II) から拡張された問題である．すなわちパラメタ間の依存解析を行った後，パラメタ間で相関の無いものについては独立にパラメタ探索をする．このことで全探索に対する探索時間を大幅に削減できる．

(IV) パラメタの探索手法

この問題も (II) からの拡張問題であるが，もし依存するパラメタが多数になる場合，全探索は問題サイズが極めて小さい場合以外では無理である．また全探索にならない場合でも，問題サイズが大きいと実行時間も大きくなるので，この理由からチューニングのための実行回数に制限がある．このような状況では推定や適当な初期値を設定した上で，遺伝的アルゴリズムやランダムアルゴリズムを適用して近似値を得る方法を利用するのが有効と考えられる．

(V) 並列計算機の特徴の知識発見

以上のような自動チューニングの結果から，各並列計算機の特徴を知識発見できる可能性が大いにある．例えば，特定の通信処理が高性能であるとか，演算性能を引き出すのに適する演算カーネルの形，さらには特定の計算機に向くアルゴリズムなどである．この得られた知識はユーザにとって性能を引き出すコーディングの雛型となり得る情報であり，非常に有用なものになりうる．

6 おわりに

本稿ではユーザによるパラメタ設定の手間を削減させると同時に，高性能が達成可能な並列数値計算ライブラリの開発とその性能について述べた．自動チューニング機能を付加することにより，密行列ライブラリでは 1.8 倍程度の高速化，疎行列ライブラリではより柔軟なパラメタ選択が可能となった．これら自動チューニング機能付き数値計算ライブラリの性能向上や，よりパラメタを減らして処理をさらに自動化するためにはチューニング情報からの知識発見が必要であると考えられる．

参考文献

[1] Choi, J., Dhillon, I., Dongarra, J., Ostrouchv, S., Petitet, A., Stanley, K., Walker, D. and Whaley, R.: LAPACK Working Note 95 : ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance, Technical report (1995).

[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*, 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] 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).

[5] 黒田久泰, 金田康正: 自動チューニング機能付き並列疎行列連立一次方程式ソルバの性能, *IPSJ SIG Notes, 99-HPC-76*, pp. 13–18 (1999).

[6] (株) 日立製作所: HITACHI SR2201 用 ScaLAPACK, PBLAS ライブラリーのご紹介, 東京大学大型計算機センター センターニュース, Vol. 30, No. 2, pp. 36–58 (1998).