I-LIB：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

片桐 孝洋[†,††] 黒田 久泰[†††]
大澤 清[†] 金田 康正[†††]

この論文では，自動チューニング機能を付加した並列数値計算ライブラリ I-LIB (Intelligent LIBrary) の機能と性能について報告する．現在開発されている I-LIB のルーチンとしては，連立 1 次方程式の求解における密行列 LU 分解ルーチン (ILIB_LU)，疎行列 GMRES(m) 法ルーチン (ILIB_GMRES)，および固有値問題の求解における密対称行列の三重対角化ルーチン (ILIB_TriRed) がある．分散メモリ型並列計算機である HITACHI SR2201 と HITACHI SR8000 を用いた性能評価の結果，自動チューニングにより ILIB_LU では理論性能に対し 80%以上の効率，ILIB_GMRES ではより柔軟なパラメタの決定により約 52 倍の高速化，ILIB_TriRed では約 1.8 倍の高速化，が得られることが分かった．

I-LIB: An Automatically Tuned Parallel Numerical Library and Its Performance Evaluation

Takahiro KATAGIRI,[†,††] Hisayasu KURODA,[†††]
Kiyoshi OHSAWA[†] and Yasumasa KANADA[†††]

In this paper, we report functions and performance for I-LIB (Intelligent LIBrary) which is parallel numerical library with an auto-tuning facilities. Already developed I-LIB routines include linear equation solvers which are LU factorization routine (ILIB_LU) for dense matrices, GMRES(m) routine (ILIB_GMRES) for sparse matrices, and tridiagonalization routine (ILIB_TriRed) for dense symmetric eigenproblem. The HITACHI SR2201 and HITACHI SR8000 both of which are distributed memory parallel machines are used in our performance evaluation. From the experimental results, we found that ILIB_LU routine attains 80% or more efficiency to theoretical peak performance, ILIB_GMRES attains 52 times speed-up by setting flexible parameters, and ILIB_TriRed attains 1.8 times speed-up by using our auto-tuning facilities.

1. はじめに

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

今回我々は，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 1 次方程式の解法）などの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 I-LIB（Intelligent LIBrary）を開発した．I-LIB は性能に関する各種パラメタの指定を，ユーザが直接行うのではなくライブラリ自体が行うという概念を実現したライブラリであり，今後その適用範囲を広げる予定である．

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ならないパラメタが非常に多く，我々の設計思想

† 東京大学大学院理学系研究科情報科学専攻
Department of Information Science, Graduate School of Science, the University of Tokyo
†† 日本学術振興会特別研究員
Research Fellow of the Japan Society for the Promotion of Science
††† 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

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だけである．そこで我々は，真に実用となる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．

本稿の構成は以下の通りである．まず 2 章で，今回開発した自動チューニング機能付き並列数値計算ライブラリ I-LIB の機能について説明する．次に 3 章で，具体的に自動チューニングの方法を説明する．4 章では，分散メモリ型並列計算機である HITACHI SR2201 と HITACHI SR8000 を用いて I-LIB の性能を評価した結果を示す．さらに 5 章で，現在開発されている自動チューニング機能付き数値計算ライブラリとの違いを説明する．最後に，本論文のまとめを述べる．

2. 現在の I-LIB が提供する機能

2.1 連立 1 次方程式ライブラリ

我々が開発している並列連立 1 次方程式ライブラリは，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．

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

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．我々の並列ライブラリにおいては

- 直接解法：密行列の LU 分解に基づくルーチン (`ILIB_LU`)
- 反復解法:疎行列反復法の一種である GMRES(m) 法に基づくルーチン [5] (`ILIB_GMRES`)

において既に自動チューニング機能を実装している．

2.2 固有値問題ライブラリ

現在開発中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．

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

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 Householder-bisection 法を用いている [4]．

今回自動チューニング機能を実装した部分は

- 対称実数密行列の相似変換における三重対角化ルーチン (`ILIB_TrdRed`)

であり，行列 A を三重対角行列 T に Householder 法を用いて相似変換をするルーチンである．この三重対角化処理は全固有値を計算する場合，全体の処理の 90% 以上を占めることが知られており [4]，高速実装が望まれるルーチンである．

3. 自動チューニング機能の説明

3.1 連立 1 次方程式ライブラリ

3.1.1 `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（同時多段多列消去法における 段数 BH と列数 BW）
(ii) 通信実装方式
(iii) データ分散方式

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 [6] 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 1** 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．

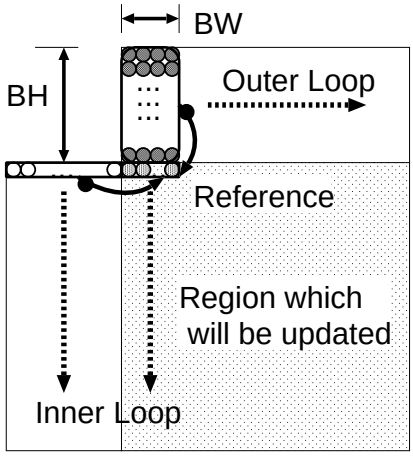

図 1 外積形式ガウス法におけるブロック幅

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の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，$BH, BW = \{1, 2, 3, 4, 5, 6, 7, 8, 16\}$ とし，この $9 \times 9 = 81$ 通りの組合せから最適な (BH_{opt}, BW_{opt}) を決定する．ここで，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，自動チューニングは静的（ライブラリ実行前）に行われる．

次に (ii) の通信実装方式とは，ピポッティング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．また，(iii) のデータ通信方式とは，行列データ A の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，これら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．

3.1.2 ILIB_GMRES の自動チューニング項目

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
(ii) 疎行列-ベクトル積における通信方式
(iii) 前処理方式
(iv) その他の各反復解法に必要とされる処理

疎行列反復解法ルーチンの場合は係数行列 A が疎であり，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．

(i) での自動チューニングでは，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) では，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) は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ことを示している．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) は，(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ことを意味している．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 に戻す方法を用いている [5]．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（古典的 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（改良された古典的 Gram-Schmidt 法，IRCGS（Iterative Refinement CGS））[7]，および並列効率が低いが精度が高い方法（修正 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．

3.2 固有値問題ライブラリ

3.2.1 ILIB_TriRed の自動チューニング項目

三重対角化ルーチンにおける自動チューニングは，密行列の LU 分解とほぼ同じである．ただし k 反復時の演算処理の自動チューニングに関して，(i) 行列-ベクトル積：$A^{(k)}u$，行列更新：$A^{(k+1)} = A^{(k)} - u(x^T - \mu u^T) - xu^T$ （ここで $u, x \in \Re^n$, $\mu \in \Re$）におけるループアンローリング段数を自動決定する点が異なる．また，(ii) 通信処理に関しては，(i) 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．LU 分解と同様に，これらの自動チューニングは静的（ライブラリ実行前）に 1 度行えばよい点に注意する．

4. 性 能 評 価

この章では，本稿で示した I-LIB 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．

☆ 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–32 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．

4.1 連立 1 次方程式ライブラリの性能

4.1.1 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, ···, 8 段, 16 段 }
- 同時消去列数 BW = {1 列，2 列, 3 列, ···, 8 列, 16 列 }

の 2 種類である．

表 1 に SR8000 における，自動チューニングにより最適化したパラメタの組を用いた実行時間と，文献 6) から得られる妥当と思われるパラメタの組 (BH, BW) = (8, 4) を設定した場合の実行時間の比較を載せる．

表 1 ILIB_LU（密行列 LU 分解）における性能比較 (SR8000，問題サイズ：11608)

(a) 2 ノード (16IP)

Auto Tuning	段数 (BH)	列数 (BW)	実行時間	GFLOPS (効率%)	最適化時間
ON	16	5	78.01 [秒]	13.36 (83.5%)	13390 [秒]
OFF	8	4	103.51 [秒]	10.07 (62.9%)	—

(b) 32 ノード (256IP)

Auto Tuning	段数 (BH)	列数 (BW)	実行時間	GFLOPS (効率%)	最適化時間
ON	16	5	9.79 [秒]	106.5 (41.6%)	1230 [秒]
OFF	8	4	10.76 [秒]	96.91 (37.8%)	—

表 1(a) 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 83%を達成している．このことから ILIB_LU は，SR8000 の性能を十分に引き出すことができたといえる．

表 2 は，SR8000 の 2 ノード（16IP）における最適パラメタ探索時の実行時間を示している．表 2 から，SR8000 では $BH \times BW$ が 80 以下の時に性能が出やすい傾向があることがわかる．

4.1.2 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．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．

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．

[**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 1 次方程式の次元は 160,000 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 $80 \times 80 \times 80$ 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 1 次方程式の次元は 512,000 である．R については，$R = 1$，100 の 2 種類．

表 2 ILIB_LU（LU 分解）における自動チューニングによる実行結果.
(SR8000, 2 ノード (16IP), 問題サイズ:11608, 単位:秒)

	BW								
BH	1	2	3	4	5	6	7	8	16
1	320.1	315.8	338.2	316.9	313.1	316.0	320.9	275.4	274.2
2	205.7	189.3	187.7	163.1	174.3	183.3	188.3	168.7	181.1
3	255.3	183.9	161.0	142.1	152.5	149.6	148.5	145.4	121.4
4	160.2	131.6	115.3	111.4	121.2	109.2	128.4	126.7	99.36
5	298.0	216.7	169.0	147.8	148.5	138.4	146.0	121.2	99.21
6	402.0	160.4	134.2	118.5	119.5	103.4	101.9	114.7	110.2
7	473.9	184.2	151.1	129.2	100.2	95.97	94.30	118.9	107.9
8	138.5	105.8	97.36	103.5	94.30	92.09	93.05	87.00	110.0
16	120.5	94.87	85.51	78.69	**78.00**	106.8	98.70	106.1	214.7

表 3 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．(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．

(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)
T2(itr.)	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

[**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 の主要な最左欄の語句の説明は次の通りである．

T1： PETSc[7] と同様のパラメタ（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 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．また問題の性質に依存するが，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ことがわかる．

4.2 密行列固有値問題ライブラリの性能

4.2.1 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, ···, 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．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 4 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 4 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 5 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 `ILIB_TriRed` 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[8] を用いた．

表 4 `ILIB_TriRed`（三重対角化）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 (8IP) 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 (32IP) 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])	

表 5 ILIB_TriRed（三重対角化）における性能比較．単位は秒．

(a-1) 4PE の場合 (SR2201)

問題サイズ	SLP TRD (Grid, BL)	ILIB_TriRed1 (not auto-tuned)	SLP /ILIB_TriRed1	ILIB_TriRed2 (auto-tuned)	ILIB_TriRed1 /ILIB_TriRed2	SLP /ILIB_TriRed2
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) 32PE の場合 (SR2201)

問題サイズ	SLP TRD (Grid, BL)	ILIB_TriRed1 (not auto-tuned)	SLP /ILIB_TriRed1	ILIB_TriRed2 (auto-tuned)	ILIB_TriRed1 /ILIB_TriRed2	SLP /ILIB_TriRed2
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 (8IP) の場合
(SR8000, ノード内並列，共有メモリ構成)

問題サイズ	ILIB_TriRed1 (not auto-tuned)	ILIB_TriRed2 (auto-tuned)	ILIB_TriRed1 /ILIB_TriRed2
100	0.024 (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 (32IP) の場合
(SR8000, ノード間並列，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	0.038 (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

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 8) 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 `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 5(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 数が増加するにつれ `ILIB_TriRed` 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 `ILIB_TriRed` の方が 2–5 倍程度高速なことがわかる．一方，表 5(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．

5. 関連研究

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は PHiPAC[2] である．PHiPAC は行列-行列積における最適なブロック幅の探索を自動的に行うことで，自動チューニング機能を実現した．しかしながら PHiPAC は並列処理における自動チューニングは行っていない．

一方，基本線形計算副プログラム (BLAS) を自動チューニングすることでライブラリ全体の自動チューニングを実現するという方針のライブラリとして ATLAS[3] が挙げられる．ATLAS は BLAS3 演算の自動チューニングプログラムが完成し，現在 BLAS2 および 疎行列ライブラリの自動チューニングプログラムの開発を行っている．しかしながら現在，並列処理を含めた自動チューニングは考慮していない．

また疎行列反復法における前処理の自動選択を行っているライブラリとして PSPARSLIB[9] が挙げられる．PSPARSLIB は Bi-CGStab 法などの多くの反復解法を含み，並列化をしているライブラリである．しかし我々が行ったような行列-ベクトル積における実行時最適化は現在，実装されていない．

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，高い実行性能を達成可能な並列数値計算ライブラリ I-LIB 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，連立 1 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 83%を達成し，疎行列 GMRES(m) 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た．また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 倍程度の高速化を達成した．これらの結果から，今後自動チューニングは数値計算ライブラリにおいて，重要な技法になるものと考えられる．

なお I-LIB 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) Choi, J., Demmel, J., Dhillon, I., Dongarra, J., Ostrouchov, S., Petitet, A., Stanley, K., Walker, D. and Whaley, R. C.: ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers — Design Issues and Performance, Technical Report LAPACK Working Note 95, University of Tennessee, Knoxville (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, *Proc. 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: 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 研究部門).

7) 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).

8) (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).

9) 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.