# 連立 1 次方程式の求解における 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 の性能 — チューニング情報からの知識発見 — *

片桐 孝洋¶ † 黒田 久泰‖ 大澤 清¶ 金田 康正‖
¶ 東京大学大学院理学系研究科情報科学専攻
† 日本学術振興会特別研究員
‖ 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**

この報告では，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．本稿では特に連立 1 次方程式の解法ルーチンを，分散メモリ型並列計算機である HITACHI SR2201，HITACHI SR8000，NEC SX-4/64M2，および Fujitsu VPP800/63 を用いた性能評価の結果から，高性能を達成するために必要なチューニングにおける知識の発見を検討し，パラメタ探索に発見した知識を適用することを考察する．

## 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．

## 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．

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

*この原稿は，並列処理シンポジウム JSPP2000 (2000 年 5 月 30 日，早稲田国際会議場) で発表された「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」，および SWoPP2000 (2000 年 8 月 3 日，松山市総合コミュニティセンター) で発表された「異機種並列環境における連立一次方程式ライブラリの性能評価」の原稿をもとに，加筆修正・削除を行ったものである．

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．

# 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．

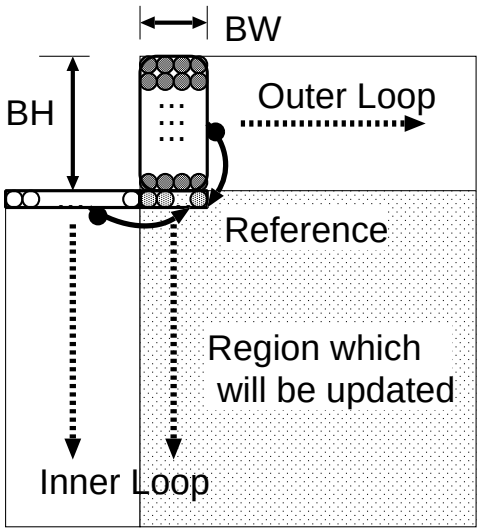


図 1: 外積形式ガウス法におけるブロック幅

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．

# 4 性能評価

## 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

ここでは HITACHI SR2201[1] ，HITACHI SR8000[2] の各並列計算機上に，密行列 LU 分解ルーチン ILIB_LU を実装し，性能評価した結果を示す．

[1] 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．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] 東京大学情報基盤センターが所有している 128 ノードの SR8000 のうち 16 ノードを使用した．各ノードは 8 つの IP から構成され，各 IP の理論ピーク性能は 1GFlops(1 ノードでは 8GFlops), 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 1Gbyte/秒，双方向 2Gbyte/秒である．

**[問題設定]**
性能評価に用いた行列は，人工心臓の流れ解析で必要となる疎行列である．この行列の非零要素の分布を図 2 に示す．なお図 2 の行列は非対称疎行列であるが，GMRES($m$) 法などの反復解法では非常に収束が悪い．したがって，反復解法は精度の観点から適用が困難である．

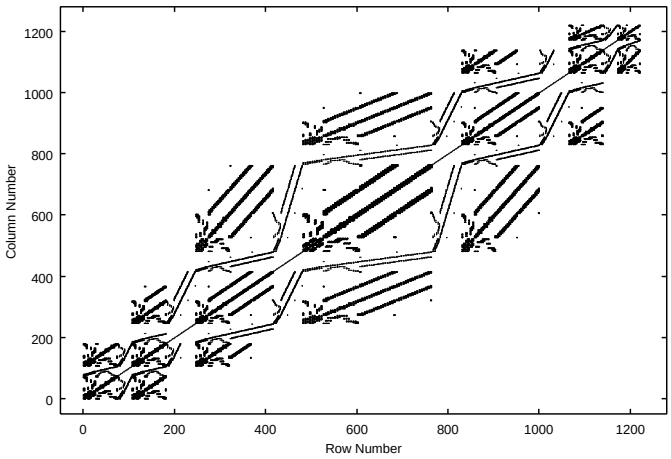


図 2: LU 分解ルーチンの性能評価に用いた行列 (非対称行列，実際の 1/10 モデル)

**[結果]**
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] (3.7 [時間]) |
| 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] (20.5 [分]) |
| 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．一方チューニングに要した時間は，2 ノードの時で 3.7 時間である。実用上この時間は無視できない時間であり，今後いかにチューニングに要する時間を減少させるのかが重要になる。また表 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: 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 |

### 4.1.2 疎行列反復解法ルーチン ILIB_GMRES の性能

ここで HITACHI SR2201，HITACHI SR8000，NEC SX-4/64M2 [3] ，Fujitsu VPP800/63[4] の各並列計算機上で疎行列反復解法ルーチン ILIB_GMRES を実装し，その性能について評価した結果を示す．

ILIB_GMRES では，ベクトル演算，行列演算の部分は実行性能を考慮して Fortran 言語で記述されており，それ以外は C 言語で記述されている．疎行列ベクトル積におけるループアンローリングルーチンは，コンパイラでループアンローリングを行わないようなオプションを付けてコンパイルしている．なお，数値実験は以下の条件で行った．

- 収束条件：相対残差 $||Ax-b||/||b|| < 1.0 \times 10^{-12}$
- 初期近似解：$x_0 = (0, 0, \cdots, 0)^T$
- 計算精度：倍精度

[**問題設定**]
ここでは次の 2 つの問題で性能を測定した．

1. 問題 1　(サイズ：大，非零要素：PE 間で均等)
   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．ここでは R=1.0 とした．
2. 問題 2　(サイズ：小，非零要素：PE 間で不均等)
   Electronic circuit design の問題．これは MATRIX MARKET(http://math.nist.gov/MatrixMarket/) で提供されている問題で，セット名は HAMM，行列名は memplus である．連立 1 次方程式の次元は 17,758 で，非零要素の個数は 99,147 である．1 行当たりの平均非零要素数は 5.6 で，最大 353，最小 1 となっている．非零要素の分布を図 3 に示す．なお，右辺ベクトルの要素の値はすべて 1 とした．

[3] 大阪大学サイバーメディアセンターが所有している 64PE の SX-4/64M2 のうち 16PE を使用した．各 PE の理論ピーク性能は 2GFlops である．
[4] 京都大学大型計算機センターが所有している 63PE の VPP800/63 のうち 32PE を使用した．各 PE の理論ピーク性能は 8GFlops, PE 間はクロスバ網で結合されており，その最大転送性能は 3.2Gbyte/秒である．

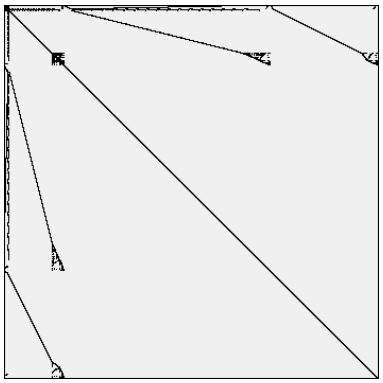

図 3: 問題 2 の非零要素の分布

[**結果**]
ここでは比較の対象として,「自動チューニングなし」の場合のパラメータとしては下記のものを選んだ.

- 行列格納形式：1 行の要素数を固定した形式
- アンローリング：なし
- 通信方式：MPI (Message Passing Interface) の関数である MPI_Allgather を使う方式
- 最大リスタート周期：30
- リスタート周期の値：固定
- 直交化：反復改良グラムシュミット
- 前処理：なし

各問題における実行時間 (単位:秒) と自動選択された方式を表 3〜表 4 に示す．表の中の語句の説明は脚注に示す[5] ．

[**考察**]
問題 1 については，まず機種が異なると選択されるアンローリング方式にも違いが出ている．SR2201 はプリフェッチコードが選択されており，SX-4 では列方向に 3 段アンローリングしたコードが選択されている．通信方式では非同期方式を使うものと同期方式を使うものに二分されたことは興味深い．直交化方式は SR2201 の PE=8 以外では，すべて CGS が選択された．前処理行列は PE 台数が増えるに従って，多項式前処理行列 (POLY) が適用される傾向にあるが，機種によってその振舞が多少異なっている．全ての機種において，前処理方式としてブロック不完全 LU 分解前処理 (BILU) が選択された場合に，速度向上比があまりよくない結果が出ている．ライブラリ側では，BILU は収束までの反復回数を大きく減らすことが見込めると判断して選択しているものの，実際には前進代入・後退代入の計算が今回のベクトル向き並列計算機には実行時間から見て不向きであるため速度向上にはいたっていない．このことから前処理方式を選ぶ基準についてはまだまだ議論の余地が残っている．

[5] 合計時間：自動チューニングにかかる時間も含めた全体の時間.
Unrolling：アンローリング方式．N はプリフェッチあり P はプリフェッチなし．() 内の数字は列方向，行方向の展開数.
通信方式：Send は Send と Recv のペアを使う．Isend は Isend → Irecv の順．Irecv は Irecv → Isend の順.
M_All は MPI_Allgather．One_A は 1PE に集約後 MPI_Bcast を使う.
直交化方式：CGS は古典的グラムシュミット．MGS は修正グラムシュミット.
前処理方式：BILU はブロック不完全前処理行列．POLY は行列多項式前処理.
速度向上比：自動チューニングなしと比べた場合の速度向上比.

表 3: 問題 1(楕円型偏微分方程式) における数値実験結果

(a) HITACHI SR2201

| PE 台数 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| 自動チューニングなし | | | | | |
| 反復回数 | 1265 | 1265 | 1265 | 1265 | 1265 |
| 実行時間 | 187.6 | 112.7 | 87.6 | 73.3 | 67.3 |
| 自動チューニングあり | | | | | |
| 反復回数 | 288 | 300 | 417 | 417 | 417 |
| 合計時間 | 72.1 | 37.3 | 12.6 | 7.4 | 5.1 |
| Unrolling | N(1,7) | P(1,7) | P(1,7) | P(1,7) | P(1,7) |
| 通信方式 | Isend | Isend | Isend | Isend | Isend |
| 直交化方式 | MGS | CGS | CGS | CGS | CGS |
| 前処理方式 | BILU | BILU | POLY | POLY | POLY |
| 速度向上比 | 2.6 | 3.0 | 7.0 | 9.9 | 13.2 |

(b) HITACHI SR8000

| IP 台数 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| 自動チューニングなし | | | | | |
| 反復回数 | 1265 | 1265 | 1265 | 1265 | 1265 |
| 実行時間 | 78.0 | 48.4 | 31.6 | 27.3 | 26.2 |
| 自動チューニングあり | | | | | |
| 反復回数 | 288 | 300 | 417 | 417 | 417 |
| 合計時間 | 37.9 | 20.0 | 6.3 | 4.1 | 3.9 |
| Unrolling | N(1,7) | N(1,7) | N(1,7) | N(1,7) | N(1,7) |
| 通信方式 | Irecv | Irecv | Isend | Isend | Irecv |
| 直交化方式 | CGS | CGS | CGS | CGS | CGS |
| 前処理方式 | BILU | BILU | POLY | POLY | POLY |
| 速度向上比 | 2.1 | 2.4 | 5.0 | 6.7 | 6.7 |

(c) NEC SX-4/64M2

| PE 台数 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|
| 自動チューニングなし | | | | |
| 反復回数 | 1265 | 1265 | 1265 | 1265 |
| 実行時間 | 729.1 | 553.1 | 287.7 | 147.7 |
| 自動チューニングあり | | | | |
| 反復回数 | 279 | 417 | 288 | 300 |
| 合計時間 | 457.6 | 28.7 | 119.3 | 61.0 |
| Unrolling | N(3,7) | N(3,7) | N(3,7) | N(3,7) |
| 通信方式 | Irecv | Send | Send | Send |
| 直交化方式 | CGS | CGS | CGS | CGS |
| 前処理方式 | BILU | POLY | BILU | BILU |
| 速度向上比 | 1.6 | 19.3 | 2.4 | 2.4 |

(d) Fujitsu VPP800/63

| PE 台数 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| 自動チューニングなし | | | | | |
| 反復回数 | 1265 | 1265 | 1265 | 1265 | 1265 |
| 実行時間 | 261.1 | 132.7 | 68.6 | 36.4 | 20.6 |
| 自動チューニングあり | | | | | |
| 反復回数 | 279 | 417 | 288 | 300 | 417 |
| 合計時間 | 107.9 | 7.6 | 27.5 | 14.1 | 1.1 |
| Unrolling | N(1,7) | N(1,7) | N(1,7) | N(1,7) | N(1,7) |
| 通信方式 | Isend | Send | Send | Send | Send |
| 直交化方式 | CGS | CGS | CGS | CGS | CGS |
| 前処理方式 | BILU | POLY | BILU | BILU | POLY |
| 速度向上比 | 2.4 | 17.5 | 2.5 | 2.6 | 18.7 |

表 4: 問題 2(Electronic circuit design) における数値実験結果

(a) HITACHI SR2201

| PE 台数 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| 自動チューニングなし | | | | | |
| 反復回数 | 775 | 754 | 803 | 715 | 819 |
| 実行時間 | 101.0 | 49.9 | 27.5 | 13.6 | 9.4 |
| 自動チューニングあり | | | | | |
| 反復回数 | 286 | 286 | 286 | 286 | 286 |
| 合計時間 | 55.1 | 27.9 | 14.8 | 8.2 | 5.2 |
| Unrolling | N(1,3) | N(1,7) | N(1,3) | N(1,6) | N(1,7) |
| 通信方式 | Irecv | Send | Send | M_All | M_All |
| 直交化方式 | CGS | CSG | CGS | CGS | CGS |
| 前処理方式 | POLY | POLY | POLY | POLY | POLY |
| 速度向上比 | 1.8 | 1.8 | 1.9 | 1.7 | 1.8 |

(b) HITACHI SR8000

| IP 台数 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| 自動チューニングなし | | | | | |
| 反復回数 | 821 | 799 | 769 | 808 | 769 |
| 実行時間 | 191.0 | 57.5 | 76.4 | 47.5 | 23.4 |
| 自動チューニングあり | | | | | |
| 反復回数 | 286 | 286 | 286 | 286 | 286 |
| 合計時間 | 89.4 | 42.4 | 62.4 | 36.6 | 19.1 |
| Unrolling | N(1,2) | N(1,1) | N(1,1) | N(1,6) | N(1,6) |
| 通信方式 | Irecv | Irecv | Irecv | Irecv | M_All |
| 直交化方式 | CGS | CGS | CGS | CGS | CGS |
| 前処理方式 | POLY | POLY | POLY | POLY | POLY |
| 速度向上比 | 2.1 | 1.4 | 1.2 | 1.3 | 1.2 |

(c) NEC SX-4/64M2

| PE 台数 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|
| 自動チューニングなし | | | | |
| 反復回数 | 859 | 715 | 821 | 835 |
| 実行時間 | 44.8 | 17.8 | 12.1 | 7.7 |
| 自動チューニングあり | | | | |
| 反復回数 | 286 | 286 | 286 | 286 |
| 合計時間 | 29.6 | 16.7 | 9.1 | 4.9 |
| Unrolling | N(1,8) | N(1,8) | N(1,8) | N(1,8) |
| 通信方式 | Irecv | Irecv | M_All | One_A |
| 直交化方式 | CGS | CGS | CGS | CGS |
| 前処理方式 | POLY | POLY | POLY | POLY |
| 速度向上比 | 1.5 | 1.1 | 1.3 | 1.6 |

(d) Fujitsu VPP800/63

| PE 台数 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| 自動チューニングなし | | | | | |
| 反復回数 | 803 | 819 | 807 | 824 | 758 |
| 実行時間 | 7.5 | 4.0 | 2.2 | 1.3 | 0.94 |
| 自動チューニングあり | | | | | |
| 反復回数 | 286 | 286 | 286 | 286 | 286 |
| 合計時間 | 7.0 | 3.8 | 2.1 | 1.2 | 0.79 |
| Unrolling | N(1,1) | N(1,1) | N(1,1) | N(1,1) | N(1,1) |
| 通信方式 | Isend | Send | Send | Send | Send |
| 直交化方式 | CGS | CGS | CGS | CGS | CGS |
| 前処理方式 | POLY | POLY | POLY | POLY | POLY |
| 速度向上比 | 1.1 | 1.1 | 1.0 | 1.1 | 1.2 |

問題 2 については，アンローリング方式に関しては問題 1 と同様に機種によって違いが出ている．一般にアンローリング段数は大きいほど効率が良いと考えられるが，疎行列ベクトル積においては，ループ長はそれほど長くはならないため (1 行あたりの要素数は数十以下である)，係数行列の非零要素の分布によって最適なオブジェクトコードを選択した方が良い．通信方式では MPI_Allgather を使った方式が 4 箇所で表れている．本ライブラリでは，非零要素の分布を調べてできるだけ通信量が少なくなるような通信方法を利用しているが，それでも，メーカー提供の全対全通信である MPI_Allgather の方が選択されることがある．これは，本ライブラリではネットワークの結合網の形状や通信性能まで考慮していないことによる．

どの機種においても，問題 2 のような比較的小さいサイズの問題では，自動チューニングの効果はあまり出てこない．しかし，問題 1 のような大きな問題では，自動チューニングの効果が大きくなることがわかる．

# 5 チューニング済パラメタからの知識発見

この章では，I-LIB の各ルーチンにおけるチューニング済のパラメタ情報から，どのような知識が発見できるか検討する。またその得られた知識を，どのように自動チューニングルーチンに実装するかを検討する。

## 5.1 密行列 LU 分解ルーチン ILIB_LU

ILIB_LU での自動チューニング済パラメタとその実行時間の表 2 から，上位 10 パラメタを抽出した．それを表 5 に示す．

表 5: 自動チューニング済パラメタ (表 2) における上位 10 パラメタ

| 順位 | BH | BW | 時間 [秒] | BH × BW |
|---|---|---|---|---|
| 1 | 16 | 5 | 78.00 | 80 |
| 2 | 16 | 4 | 78.69 | 64 |
| 3 | 16 | 3 | 85.51 | 48 |
| 4 | 8 | 8 | 87.00 | 64 |
| 5 | 8 | 6 | 92.09 | 48 |
| 6 | 8 | 7 | 93.05 | 56 |
| 7 | 8 | 5 | 94.30 | 40 |
| 7 | 7 | 7 | 94.30 | 49 |
| 9 | 7 | 6 | 95.97 | 42 |
| 10 | 8 | 3 | 97.36 | 24 |

表 5 では，SR8000 では $BH \times BW$ が 80 付近の時に性能が出やすい傾向があることがわかる．また $BH \times BW$ が 80 以上になると性能向上が見込めないこともわかる．

この性能向上の上限値 80 という値は，ブロック LU 分解の演算カーネルのレジスタ割り当てに依存することが考えられる．なぜならばこのアルゴリズムでは，ブロック幅 $BH, BW$ を大きくする程，レジスタに固定される，消去演算中で参照のみされるデータ数が増加するからである．このことから，この上限値は計算機の物理的なレジスタ数とコンパイラのレジスタ割り当ての仕方から決まる値であるといえる．したがってどのような計算機システムでも，このような上限値を持つことがアルゴリズムの性質上予想される．

以上のアルゴリズムの特性とチューニング済のパラメタ情報から，以下のような知識を発見できる．

- ブロック LU 分解ルーチンでは，性能に関する $BH \times BW$ の上限値が存在する．
- その上限値を越える場合，性能向上は見込めない．
- 問題の性質によらず，この上限値は一定であることが予想される．

この知識からチューニング時間を減らすために，まず自動チューニングを行う場合に比較的小さなサイズの問題を実行して $BH \times BW$ の上限値を発見し，次に大きなサイズの問題をチューニングする際にその上限値の周辺のみ調べる手法が考えられる．

### 5.2 疎行列反復解法ルーチン ILIB_GMRES

疎行列反復解法ルーチンにおいて，アルゴリズムの特性とチューニング済のパラメタ情報から，以下のような知識を発見できる．

- PE 台数を増やした時にスーパーリニアな性能向上が得られると判明した場合，実行時に最適であると判断した前処理が最適でない可能性がある．スーパーリニア性能を達成する前処理がより最適な前処理であるといえる．
- 同一の問題を解く場合，プロセッサー台数に関係なくアンローリングの方式はほぼ同一である．
- 非零要素の分布が PE 間で均等な場合は通信に１対１通信が用いられることが多く，不均等な場合で PE 台数が多くなると１対全通信が用いられる傾向がある．

これらの知識からチューニングのためのパラメタの探索を絞りチューニング時間の削減が行える．またチューニング済の実行時間情報の記録から，より最適なパラメタに修正できる可能性がある．

## 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．密行列 LU 分解ルーチン ILIB_LU を人工心臓の流体解析に用いられる，反復解法が適用困難な行列に対して適用した．その結果として，理論性能に対する効率 83%を達成した．また疎行列 GMRES($m$) 法ルーチン ILIB_GMRES では，偏微分方程式や実際の回路設計に用いられる実用的な行列に対して適用を行い，その結果としてより柔軟なパラメタ選択が可能であることが判明した．またその結果として，最高で約 20 倍程度の速度向上が得られた例も確認された．

この一方で自動チューニングをおこなったパラメタ情報から，直接解法や反復解法に関する特化された知識も発見できることが判明した．この得られた知識は自動チューニング時間の短縮ばかりでなく，精度の向上にも利用できる．またこの知識は，アルゴリズムや適用する問題の特質を熟知している人間にとっては自明のことである．しかしながら最適化されたパラメタのみから推論を行っても，人間にとっては発見可能である．これらのことから我々の自動チューニングの手続きをさらに一般化して，汎用的なチューニングの手続きを実装できる可能性があるといえよう．

またより洗練された自動チューニングを行うためには，コンピュータアーキテクチャの特性情報 (レジスタ数，キャッシュサイズなど) が必要になる．そのため，これらの特性情報を正確に測定できる手法を確立することが将来的な課題となるだろう．

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).