ハイパフォーマンス
コンピューティング 76－3
（1999．5．14）

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

黒田久泰† 金田康正††

GMRES(m) 法は係数行列が大規模で非対称疎行列である連立一次方程式を解くための非定常反復解法の一つである．我々は GMRES(m) 法を用いた分散メモリ型並列計算機上で動作するライブラリを開発した．本ライブラリでは，実行時に自動的にパラメータ調整を行い，早く解を導き出すための最適な手法を選択するように構成されている．本稿では，本ライブラリの性能について述べる．

Performance of Automatically Tuned Parallel Sparse Linear Equations Solver

Hisayasu Kuroda† and Yasumasa Kanada††

The GMRES(m) method is one of the iterative methods to solve large and sparse nonsymmetric linear equations. We developed a software library which uses the GMRES(m) methods in distributed memory machines. This software library can adjust some parameters automatically and select the optimal way to find the faster solution. We show the performance of this software library.

1. はじめに

偏微分方程式の境界値問題を解く場合など理工学のあらゆる分野で，大規模疎行列を係数行列とする連立一次方程式を解く必要が生じる．

連立一次方程式の係数行列が大規模で疎という性質を持つとき，多くの場合，反復法を用いて解くことになる．代表的な反復法として係数行列が正定値対称行列であるときは，共役勾配法 (CG 法) が挙げられる．係数行列が非対称の場合には，多くの解法が存在している[1]が，並列計算機上で効率良く動作する解法についての考察はまだ十分になされているとは言えない．そこで，我々は数多くある反復法の中から GMRES 法 (Generalized Minimal RESidual method) の変形である GMRES(m) 法を取り上げ分散メモリ型並列計算機上で実装を行った．GMRES 法は Krylov 部分空間法の 1 つで，各反復で残差ノルムを最小化するように近似解を更新する方法である．GMRES(m) 法は，GMRES 法を m 回の反復を周期としてリスタートを行う．リスタートを行うことで，必要となる計算時間と記憶容量を抑えている．

† 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
†† 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, the University of Tokyo

2 章では GMRES(m) 法を利用したアルゴリズムについて示し，3 章では自動設定されるパラメータについて述べる．4 章で性能評価，5 章で考察，6 章でまとめを示す．

2. GMRES(m) 法のアルゴリズム

ここでは連立一次方程式の係数行列を A とし，未知ベクトルを x，右辺ベクトルを b とする．

前処理行列を利用した GMRES(m) 法のアルゴリズムを図 **1** に示す．ここで，前処理行列を K とし，収束条件判定の便利さから右前処理行列を適用したアルゴリズムを示す．また，r, v, w はベクトルを表し，右下に添字がある場合，その添字が異なれば別のベクトルを表す (ベクトル内の要素を指しているのではない)．m はリスタートを行う反復の回数である．

2.1 GMRES(m) 法の並列化について

要素の配置は行ブロック分割による分散を行う．

各 Processor Element (以下 PE) に分散させる行列およびベクトルは，係数行列 A，解ベクトル x，右辺ベクトル b，作業用ベクトル v ($m+1$ 個必要)，直交化ベクトル w である．行列 H，ベクトル s, c は全ての PE で同じ内容を保持する．

なお，x, v は集積通信 (gather オペレーション) が必要となるので，行列サイズ n の長さに等しいテンポ

1: 初期推定値 x_0 を与える
2: $r = b - Ax_0$
3: for j=1,2,$\cdots$
4: $v_0 = r/||r||$
5: $e_0 = ||r||$
6: for i=0,1,$\cdots$,$m-1$
7: $w = AK^{-1}v_i$
8: for k=0,1,$\cdots$,i
9: $h_{k,i} = (w, v_k)$
10: $w = w - h_{k,i}v_k$
11: end
12: $h_{i+1,i} = ||w||$
13: $v_{i+1} = w/h_{i+1,i}$
14: for k=0,$\cdots$,$i-1$
15: $\begin{bmatrix} h_{k,i} \\ h_{k+1,i} \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} h_{k,i} \\ h_{k+1,i} \end{bmatrix}$
16: end
17: $c_i = \sqrt{\frac{h_{i,i}^2}{h_{i,i}^2+h_{i+1,i}^2}}$
18: $s_i = -\frac{h_{i+1,i}}{h_{i,i}}\sqrt{\frac{h_{i,i}^2}{h_{i,i}^2+h_{i+1,i}^2}}$
19: $e_{i+1} = -s_i e_i$
20: $e_i = c_i e_i$
21: $h_{i,i} = c_i h_{i,i} + s_i h_{i+1,i}$
22: $h_{i+1,i} = 0.0$
23: e_{i+1} が十分小さければ x_k を更新して終了
24: end
25: $y_k = H_k^{-1}(e_0, e_1, \cdots, e_k)^T$
26: $x_k = x_0 + K^{-1}\sum_{i=0}^{k} y_i v_i$
27: $r = b - Ax_k$
28: $||r||/||b||$ が十分小さければ終了
29: $x_0 = x_k$
30: end

図 1 GMRES(m) 法

ラリ用のベクトルを各 PE 内で用意する．

並列化する際，通信が発生するのは，行列ベクトル積 (2,7,26,27 行目) とベクトル内積 (9 行目) である．

ギブンズ回転行列を用いて QR 分解を行うところ (14～22 行目) では，各 PE でヘッセンベルグ行列 H の内容を全ての PE で共通に持たせて，全ての PE で全く同じ更新処理を行うことにする．

この QR 分解を行う部分では，リスタートする反復回数を m としたとき，ヘッセンベルグ行列のサイズは $(m+1) \times m$ となる．m の値は実用的には高々100ぐらいまでとなるので，行列 H を分散させて処理する必要はないと考えられる．全ての PE で同じ更新処理を行うオーバヘッドについては，考察のところで述べる．

3. 自動チューニング

我々は既に，CG 法を基にした自動チューニング機能付きライブラリ [2] を開発している．ここでは，本ライブラリで実装されている自動チューニングについて説明する．

3.1 行列格納形式の変換

次に示す係数行列をメモリに格納する場合，**図 2** のように行圧縮格納形式を使うことが多い．

$$A = \begin{bmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{bmatrix}$$

```
rp[5]={0,2,5,8,10}; /* 各行の index 開始位置 */
cval[10]={0,1,0,1,2,1,2,3,2,3}; /* index */
aval[10]={a,b,c,d,e,f,g,h,i,j}; /* 要素 */
```

図 2 行圧縮格納形式

各行に非零要素が均等に散らばっている場合，格納形式を**図 3** に示すように行毎のサイズを固定になるよう変換した方がループアンローリングの効果で高速化が期待できる．

```
cval[12]={0,1,0, 0,1,2, 1,2,3, 2,3,0}; /* index */
aval[12]={a,b,0, c,d,e, f,g,h, i,j,0}; /* 要素 */
nsize[4]={2,3,3,2};          /* 各行の要素数 */
csize=4;                     /* 行の幅 */
```

図 3 アンローリング用格納形式

本ライブラリでは，それぞれの格納形式で，実際に行列ベクトル積を行い，実行時間の短い格納形式を選択する．

3.2 行列ベクトル積のループアンローリング

各行の非零要素の個数が 9 個以下である場合，それぞれの個数に対して，ループを全て展開したルーチンを用意しておく．これにより行方向についてすべて展開されたコードになる．そしてさらにもう一段，行列の列方向に対しても 1,2,3,4,8 回展開したものをそれぞれ用意する．

ループを展開する際，配列要素の間接参照を無くすようプリフェッチされたものと，プリフェッチしないもの (この場合プリフェッチの処理はコンパイラに委ねる) とをそれぞれ用意する．**図 4** にこの例を示す．

本ライブラリには，1 行あたりの非零要素の個数が，ある特定の値に対し，プリフェッチあり 4 種類，プリフェッチなし 5 種類の計 9 個のルーチンがある．さらに非零要素の個数が 2 - 9 のものはそれぞれ個別のルーチンを持っているので，その部分だけで合計 72 個のルーチンを持っていることになる．

```
プリフェッチなし
  do i=1,10
    s = s + a(i) * b(ind(i))
  end do
プリフェッチあり
  m=ind(1)
  do i=1,9
    s = s + a(i) * b(m)
    m=ind(i+1)
  end do
  s = s + a(10) * b(m)
```

図 4　プリフェッチコードの例

以上のルーチン群を利用して，反復開始前に実際に単一 PE 上で係数行列 A に対する行列ベクトル積を行う．この結果から，実行時間の最も早かったループアンローリングの方式を自動選択する．

3.3　行列ベクトル積における通信方式

行列ベクトル積の際，ベクトルの集積通信が必要である．ここでは，この集積通信において下記の 5 つの方法から最も早く計算できるものを選択する．

行列 A の性質に依存しない通信手段：

(1)　MPI ライブラリにある MPI_Allreduce を使う，
(2)　1PE に集中させてからブロードキャスト通信．

行列 A の性質に依存 (通信テーブル利用)：

(3)　MPI_Isend，MPI_Irecv の順で使う，
(4)　MPI_Irecv，MPI_Isend の順で使う，
(5)　MPI_Send，MPI_Recv を使う．

ここで通信テーブルとは，ベクトルの中の要素の index と，それが参照される PE 番号との関連を持ったテーブルである．この通信テーブルを利用すると，ベクトルの要素を参照する PE にだけデータが送信されるので，通信量を最小限に止めることができる．ただし，要素の index が離れている場合には，1PE への送信が 1 回となるよう最小 index と最大 index の間にある要素をすべて送信するようにする．通信テーブルの利用により非零要素の存在範囲が狭まっている場合には，通信量を大きく減らすことができる．

(5) の MPI_Send と MPI_Recv はブロッキング通信なので，その間，他の処理も停止してしまう．しかし，通信の呼出し時間を短く押さえられ，通信量が少なくかつ PE 間の通信順序をうまく決めることができている場合には性能向上が行える．

通信方式は，反復開始前に全 PE 間にまたがる行列ベクトル積を行い，実行時間の最も早かった方式を自動選択する．

3.4　リスタート周期間隔

リスタート周期は長いほど全体の反復回数は減少する．しかし，リスタート周期を長くすると，反復回数が増えるに従って 1 回の反復にかかる時間が増大する．これは，リスタート開始から前回の反復までに計算されて出てくるベクトルに直交化するベクトルを新しく算出する必要があるためである．図 1 の 8〜11 行目が直交化の部分であるが，リスタート周期を大きくすると，その 2 乗に比例して，計算量が増えていく．

リスタート周期の決め方については，いろいろな参考文献 [3] があるが，本ライブラリにおいては，次のような方法を取る．

最大リスタート周期を m(m は偶数となるように取る) とすると，

(i) $2, 4, 6, 8, \cdots, m$ のように 2 から m になるまでリスタート周期を 2 ずつ足していく．
(ii) m になったらリスタート周期を再び 2 に戻す．
(iii) 以下，同様に繰り返す．

なお，本ライブラリでは，リスタートする m の値は，メモリの許す範囲で大きな値を取るようになっている．ただし，上限値として 128 を設定している．これは，リスタート周期を大きくとると，反復を重ねる毎にベクトルの直交化の計算量が増大してしまうためである．

3.5　グラム・シュミットの直交化方式

逐次版で動作する GMRES(m) 法においては，計算精度の向上のため修正グラムシュミット直交化法 (Modified Gram-Schmidt orthogonalization, MGS) が利用されることが多い．しかし，この方法を並列計算機上で実装した場合，同期点が多く，リスタート周期が大きくなると通信が多く発生してしまい効率が悪くなる．一方で，古典的グラムシュミット (Classical Gram-Schmidt orthogonalization, CGS) の場合は一回の通信でよいため並列化には適している．

CGS を利用する場合，図 1 の 8–11 行目は**図 5** のように変更する．

```
 8:    for k=0,1,···,i
 :       h_{k,i} = (w, v_k)
 :     end
 :     for k=0,1,···,i
 :       w = w − h_{k,i} v_k
11:    end
```

図 5　古典的グラムシュミット (CGS) への置換え

本ライブラリでは，リスタートする反復回数を m とした際，$m/2$ 個のベクトルに対し CGS にかかる時間と MGS にかかる時間を計測し，早い方を自動選択する．

単一 PE 上では，メモリアクセスの局所性のある MGS の方が一般的に速くなる．しかし，並列計算機上では，キャッシュメモリ容量，PE 台数，通信性能，直交化するベクトルの本数などのいくつもの条件が重なり合うため，実際に実行時間を測定し最適な方式を選択する方が良い．

CGS では誤差が蓄積されるという問題があるが，GMRES(m) 法では，反復をリスタートする際に誤差の蓄積はすべてクリアされる．そのため，CGS を使うことによる誤差の影響はほとんど無視できる．

しかしながら，本ライブラリでは，CGS を利用している際に，リスタートを 2 回行っても残差ノルムが減少していかないと判断される場合には，CGS の利用を取り止め，MGS の利用を強制させるようにしている．

3.6 前処理行列

前処理行列を適用すると，GMRES(m) 法の収束が速くなることが期待できる．

反復開始前に係数行列に対してスケーリングを行う．スケーリングすることで，数値的安定性が増し，また，対角要素が全て 1 に固定されることで，演算量やメモリ使用量を減らすことにもつながる．

本ライブラリでは，スケーリング以外の前処理として，次の 3 つから選択される．

(1) 前処理なし
(2) 多項式前処理 [4]
(3) ブロック不完全 LU 分解前処理 [5]

ここでスケーリング後の行列を改めて A とする．(2) の多項式前処理は，$A = I + B$ と表した際，$K^{-1} = I - B$ を前処理行列として適用する．これは，A^{-1} の近似として行列のノイマン級数展開で得られる行列多項式の第 2 項までに相当する $I - B$ を利用するということに他ならない．

行列 $I - B$ は，行列 A の対角要素以外の正負を逆にしたものに等しく，非零要素の位置は全て同じで，$I - B$ を格納するためのメモリ領域を必要としないなどの利点がある．メモリを多く必要とする GMRES(m) 法においては，特に役に立つ前処理方式である．

ただし，この前処理で効果があるのは，B のスペクトル半径 $\rho(B)$ が 1 より小さい場合に限るという点に注意を要する．$\rho(B) \geq 1$ の場合は $I - B$ は A^{-1} の近似として利用できない．

(3) のブロック不完全 LU 分解前処理 (以下 BILU) では，各プロセッサ間に依存関係が起きないよう，対角ブロック毎 (対角ブロックの幅は各 PE 数が保持する行列の幅と等しくする) に不完全 LU 分解を行う．ここでは，行列 A の非零要素の存在するインデックスのところにだけ，L, U 行列の非零要素が来るという ILU(0) を使う．

なお図 1 の 7 行目と 26 行目で，$K^{-1}r$ の計算が必要となるが，これを $q = (LU)^{-1}r$ と見て，方程式 $LUq = r$ を解くことによって行う．これは作業用ベクトル z を用いて，

$$\begin{aligned} Lz &= r \ \text{(前進代入)} \\ Uq &= z \ \text{(後退代入)} \end{aligned} \qquad (1)$$

によって計算する．

リスタート間隔 m を大きく取れた場合，前処理行列の効果はさらに大きくなる．m が小さい場合には，前処理行列を処理部分のオーバーヘッドが大きくなり，効果は小さくなる．

どの前処理を適用するかの判定は，次のようにして行う．$m/2$ 回 (上限を 16 とする) の反復を行い，残差ノルムが $||r_0||$ から $||r_k||$ になったとすると，$||r_k||/||r_0||$ が最小となる前処理を適用する．ここで計算にかかる時間については考慮しないことにする．GMRES(m) 法では m の値が大きければ大きいほど，前処理行列を適用する効果が大きくなる．反復回数が 16 回という上限を設けているため，最も効果のある方法を見付けることは困難である．本ライブラリにおいては，同じ反復回数で，最も残差ノルムを小さくする前処理方式を適用することにした．

4. 性能評価

この章では，日立 SR2201 分散メモリ型並列計算機上で本ライブラリを実装し，その性能について評価した結果を示す．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☆．

なお，数値実験は以下の条件で行った．

- 収束条件：$||r_k||/||r_0|| < 1.0 \times 10^{-12}$
- 最大反復回数：10000 回
- 初期近似解：$x_0 = (0, 0, \cdots, 0)^T$
- 計算精度：倍精度
- 使用ライブラリ：MPI ライブラリ
- 実行形式：バッチジョブによる実行

4.1 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．

(1) 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，$\gamma = 1.0, 1.5, 2.0$ の 3 種類．行列の次元は 4,000,000 とする．

(2) 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

☆ 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．ベクトル積，行列積の部分は Fortran で，他は C 言語で記述されている．日立の最適化 FORTRAN90 のコンパイラオプションとして-W0,'pvec(diag(1)),opt(o(ss),fold(1))' を，日立の最適化 C のコンパイラオプションとして+O4 -Wc,-hD1 を指定した．

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

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，$R = 1.0$，1000.0 の 2 種類．

(3) 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

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

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，$R = 1.0$，100.0 の 2 種類．

4.2 結　　果

各問題における実行時間 (単位:秒) 及び自動選択された方式を**表 1**～**表 7** に示す．図 1 の 14–23 行目にあたる QR 分解の計算にかかる時間は，すべての問題で，0.1 秒未満であった．そのため，この部分を全ての PE で共通に更新を行う時間のオーバーヘッドは無視できる．各表の最左欄の語句の説明は脚注に示す☆．

表 1 問題 1 (γ=1.0)

PE 台数	8	16	32	64	128
反復回数	19	20	20	20	21
合計時間	43.9	25.0	15.3	9.0	6.5
行列積時間	21.0	10.7	5.2	2.6	1.3
直交化時間	4.7	2.4	1.2	0.6	0.4
h, v 計算	0.9	0.5	0.2	0.1	0.1
リスタート	32	64	128	128	128
Unrolling	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)
通信方式	Send	Send	Send	Isend	Send
直交化方式	MGS	MGS	MGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	BILU

5. 考　　察

反復回数については，同じ問題でも γ, R の値によって変化することがわかる．前処理方式として BILU が選択されると前処理方式なしに比べて 1/3 から 1/2 になる．また，$I - B$ が選択されると，前処理方式なしに比べて 1/2 程度になることを注記しておく．

合計時間は PE 台数が増えれば，確実に減少していることがわかる．台数効果を図 **6** に示す．

行列積時間は前処理方式で BILU が選択されると前進代入，後退代入部分も含まれるため相対的に大きくなる．BILU では反復回数が少なくなった分，行列積時間に多くを費されることがわかる．

直交化時間は，前処理方式なしの場合には，相対的に行列積時間に比べて，計算時間の占める割合が大きくなることがわかる (表 2，表 3)．

ループアンローリングは，すべての問題でプリフェッチありが選択されており，1 行当たりの非零要素数が 3 の場合，列方向にも 2 回展開，非零要素数が 5,7 の場合には，列方向には展開しない方が良いという結果

表 2 問題 1 (γ=1.5)

PE 台数	8	16	32	64	128
反復回数	51	94	94	94	94
合計時間	75.0	30.2	17.9	10.1	6.9
行列積時間	43.5	7.8	3.7	1.7	0.8
直交化時間	9.6	9.8	4.9	2.4	1.2
h, v 計算	1.6	1.2	0.6	0.3	0.2
リスタート	32	64	128	128	128
Unrolling	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)
通信方式	Send	Send	Send	Irecv	Send
直交化方式	MGS	MGS	MGS	CGS	CGS
前処理行列	BILU	なし	なし	なし	なし

表 3 問題 1 (γ=2.0)

PE 台数	8	16	32	64	128
反復回数	323	322	322	322	322
合計時間	159.5	87.9	46.6	23.6	13.7
行列積時間	36.9	17.7	8.4	4.0	1.9
直交化時間	91.6	52.5	26.5	12.2	6.2
h, v 計算	7.3	3.6	1.9	1.0	0.7
リスタート	32	64	128	128	128
Unrolling	P(2,3)	P(2,3)	P(2,3)	P(2,3)	P(2,3)
通信方式	Send	Send	Irecv	Send	Send
直交化方式	MGS	MGS	MGS	CGS	CGS
前処理行列	なし	なし	なし	なし	なし

表 4 問題 2 (R=1.0)

PE 台数	8	16	32	64	128
反復回数	3930	4086	3557	4278	4738
合計時間	2145.9	1114.2	458.1	275.1	151.1
行列積時間	1000.0	509.9	218.7	124.1	64.7
直交化時間	1083.0	568.8	223.3	140.0	77.9
h, v 計算	20.9	13.9	6.0	4.5	4.2
リスタート	128	128	128	128	128
Unrolling	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)
通信方式	Send	Send	Send	Send	Send
直交化方式	MGS	CGS	CGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	BILU

☆ 合計時間：自動チューニングの時間も含めた全体の時間
行列積時間：行列ベクトル積にかかる時間 (7 行目)
直交化時間：直交化ベクトルの計算時間 (8–11 行目)
h, v 計算：h, v を計算する時間 (12–13 行目)
リスタート：リスタートする反復回数の最大値
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縦 2 つ・横 3 つに展開するという意味
通信方式：Send は Send-)Recv の順に使う，Isend は Isend -)Irecv の順に使う，Irecv は Irecv-)Isend の順に使う

表 5　問題 2 (R=1000.0)

PE 台数	8	16	32	64	128
反復回数	1213	1233	1452	1205	1563
合計時間	525.5	264.3	156.1	61.5	44.1
行列積時間	309.3	156.1	90.2	34.7	21.4
直交化時間	187.0	92.8	57.1	22.1	18.3
h, v 計算	6.6	3.7	2.3	1.3	1.7
リスタート	128	128	128	128	128
Unrolling	P(1,5)	P(1,5)	P(1,5)	P(1,5)	P(1,5)
通信方式	Send	Send	Send	Send	Send
直交化方式	MGS	CGS	CGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	BILU

表 6　問題 3 (R=1.0)

PE 台数	8	16	32	64	128
反復回数	483	481	504	518	691
合計時間	562.8	281.2	146.2	74.8	28.1
行列積時間	404.7	199.7	102.4	52.2	11.2
直交化時間	102.6	51.9	28.2	13.9	11.7
h, v 計算	5.7	2.9	1.7	1.0	0.9
リスタート	64	128	128	128	128
Unrolling	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)
通信方式	Irecv	Irecv	Irecv	Irecv	Irecv
直交化方式	MGS	MGS	CGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	$I-B$

表 7　問題 3 (R=100.0)

PE 台数	8	16	32	64	128
反復回数	274	315	383	333	418
合計時間	326.3	186.3	111.5	48.8	31.9
行列積時間	234.1	132.3	78.4	34.0	20.5
直交化時間	46.4	28.3	19.0	7.4	6.1
h, v 計算	3.3	1.9	1.3	0.6	0.6
リスタート	64	128	128	128	128
Unrolling	P(1,7)	P(1,7)	P(1,7)	P(1,7)	P(1,7)
通信方式	Irecv	Irecv	Irecv	Irecv	Irecv
直交化方式	MGS	MGS	CGS	CGS	CGS
前処理行列	BILU	BILU	BILU	BILU	BILU

になっている．

通信方式については．今回の問題は全て非零要素の位置が対角部分に密集しているため，通信テーブルを利用する方式が選択されている．特に問題 1 と問題 2 では対角成分の近くに非零要素が集中するため，MPI_Send と MPI_Recv だけを用いた通信方式が選択されている．送信データ量が多くなると，MPI_Irecv，MPI_Isend の順で使う方式が選択されている．

直交化方式では，PE 台数が多くなると，自動的に MGS から CGS に切り替わっていることがわかる．問題 1 で PE=64，問題 2 で PE=16，問題 3 で PE=32 で，CGS に切り替わっている．

前処理方式では，BILU が多くの問題で選択された．表 3 では，PE=128 のとき BILU ではなく，$I-B$ が選択されているが，これは BILU ではプロセッサ台数

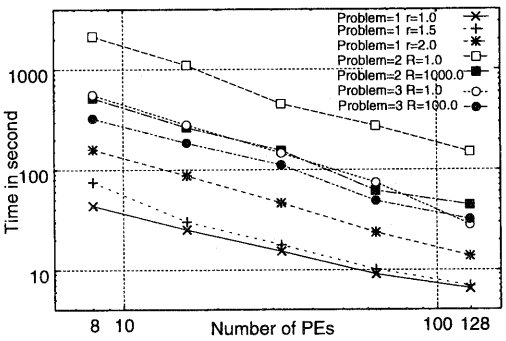

図 6　各問題における台数効果

が増えるにしたがって，ブロック化していることの欠点が露出して来ることによる．実際，問題サイズをもう少し小さく取った場合，$I-B$ が選択される場合が多く出てくると思われる．これは，前処理方式 $I-B$ の場合には，PE 数が増えても反復回数が増加することがないためである．

6. ま と め

本ライブラリは，http://www.cc.u-tokyo.ac.jp/経由で公開される予定である．専門知識のない利用者にとっても，多くのパラメータを自動設定してくれるために，容易にこれらのライブラリが利用できるようになる．また，問題の性質を理解している利用者にはついては，それぞれのパラメータを個別に設定することも可能である．今後，他のタイプの分散メモリ型並列計算機上でも性能を評価していく予定である．

参 考 文 献

1) J.J.Dongarra, I.S.Duff, D.C.Sorensen, H.A.van der Vorst: Numerical Linear Algebra for High-Performance Computers. SIAM (1998).
2) 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 — 対称疎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 回全国大会講演論文集 (1), pp.1-10 – 1-11 (1998)
3) 津野直人，野寺隆: 早期リスタートによる GMRES(m) 法の高速化. 情報処理学会論文誌. Vol.40,No.4,pp.1760–1773 (1999).
4) O.G.Johnson, C.A.Micchelli, G.Paul: Polynomial Preconditioners for Conjugate Gradient Calculations. SIAM J. Numer. Anal., Vol.20,No.2 (1983).
5) J.S.Kowalik, S.P.Kumar: An Efficient Parallel Block Conjugate Gradient Method for Linear Equations. Proc. 1982 Int. Conf. Par. Proc., pp. 47–52 (1982).