数値計算ポリシー入力型グラムシュミット直交化ライブラリの異種混合計算機環境における性能評価

直野 健[†] 猪貝光祥[††] 木立啓之[††]

報告者らは，数値計算ポリシー入力型の行列計算ライブラリ方式を提案している．本方式は，行列計算処理において，同じ機能を持つが異なる計算精度，計算時間となるアルゴリズムを同時に実行し，数値計算ポリシーとして与えられたユーザの要求精度を満たす最短処理時間での計算結果をユーザに引き渡す．本報告では，本方式の異種混合計算機環境における有効性を評価した．特に，ユーザがプログラム開発環境としての PC から大規模計算実行環境としてのスーパコンピュータにプログラム実行環境を移した際に，満足な性能を得られていなかったグラムシュミット直交化処理ついて本方式を適用した結果，本方式は最大約 4 倍の性能向上を達成することができた．

Performance Evaluation of the Gram-Schmidt Orthogonalization Library with Numerical Policy Interface on Heterogeneous Platforms

Ken Naono,[†] Mitsuyoshi Igai[††] and Hiroyuki Kidachi[††]

We have proposed the method of the matrix library with numerical policy interface. The method executes multiple algorithms that have the same function but have a different accuracy and a different computation time, and returns the result that is executed fastest and satisfies the required accuracy of a numerical policy given by user. In this paper, we evaluate the method on the heterogeneous platforms. We especially treat the Gram-Schmidt orthogonalization that was originally executed in small size computations on PCs and that, however, often attains poor performance in large scale computations on supercomputers. Performance evaluation shows that the method of the matrix computation library with numerical policy achieves about 4 times better performance on a supercomputer than the method only tuned in small size computations on a PC.

1. はじめに

近年，スーパコンピューティングの利用環境に関するパラダイム・シフトが起こりつつある．従来は，特定の計算センタにあるスーパコン単体を利用する形態が主流であった．これに対して，近年では，LAN 環境において複数種類の計算機を導入する形で，異種混合計算機環境（キャンパス・グリッド）を提供する計算センタが増えつつある．このような計算センタを利用するユーザは，最初，小規模な計算を PC クラスタで行い，その後，大規模計算をスーパコンピュータで実行するケースが多い．

このようなケースでは，PC クラスタ上で作成したプログラムが，そのままスーパコンピュータ上でも稼動するよう，PC クラスタ上とスーパコンピュータ上とで同じインタフェースを有する行列計算ライブラリが強く望まれる．

しかし，従来の行列計算ライブラリには，PC 上で最適な計算性能を満たすアルゴリズムが，そのままスーパコンピュータ上でも最適な計算性能を満たすとは限らない，という問題がある．特に，計算精度と計算時間にトレードオフがあるケースでは，ユーザの数値計算に対するポリシー（計算精度と計算時間に対応する要求など）を PC 上で満足させたアルゴリズムが，スーパコンピュータ上において，当該ポリシーを満たすとは限らない．たとえば，「計算精度は 10^{-8} 程度でもかまわないが，計算時間はできる限り短くしたい」という数値計算ポリシーを満たすように，PC 上で選択したアルゴリズムを，そのままスーパコンピュータ上で実行させた場合，上記の数値計算ポリシーを満たすとは限らず，改めてより高速に稼動するアルゴリズムを選択する必要が生じる場合がある．

† 株式会社日立製作所中央研究所
Central Research Laboratory, Hitachi, Ltd.
†† 株式会社日立超 LSI システムズ
Hitachi ULSI Systems Co., Ltd.

この問題は，特にスーパコンピュータにおいて並列計算を適用する場合に顕著である．その主要な原因として，並列化による計算順序の変更によって計算精度が異なってしまう点があげられる．反復法を用いている連立一次方程式や固有値計算では，この計算精度の変化が処理中の反復回数に影響を与え，結果として想定したほどの並列化による性能向上が得られない場合がある．

そこで，本研究では，文献 1) で提案した「数値計算ポリシー入力型」の数値計算ライブラリ構成方法を利用し，上述の問題を解決することを図る．特に，本報告では，計算精度と処理性能の調整が難しく，かつ疎行列固有値の反復解法で多用されるグラムシュミットの直交化処理に対して上記の方法を適用する．

2. 従来の直交化ライブラリの異種混合計算機環境向け実装上の問題点

2.1 従来の直交化ライブラリにおけるアルゴリズム

複数のベクトルを互いに直交化させ，直交基底を作る「直交化処理」は，固有ベクトル計算，連立一次方程式の反復法におけるクリロフ部分空間の計算処理など，線形計算において非常に多く利用されている．これらの数値計算の精度，時間は，直交化処理の計算精度，計算時間によって大きく影響を受ける．したがって，直交化処理の高速化，高精度化は非常に重要である．

直交化処理は一般に＜プログラム 1 ＞のように表現される．ここでは，直交化の対象となるベクトル群を $v_j(j = 1, \cdots, JMAX)$ とする．ただし JMAX は直交化の対象となるベクトルの本数である．また，ベクトル v_j の i 番目の要素を $V(i, j); i = 1, \ldots, N$ と書くことにする．ただし N はベクトルの長さである．

＜プログラム 1 ＞

```
do M = 1, JMAX
   call single_orthogonal(V(1,M), V, M)
enddo
```

＜プログラム 1 ＞での single_orthogonal は，ベクトル 1 本分の直交化であり，この部分は，古典グラムシュミット（Classical Gram-Schmidt，以下 CGS）アルゴリズム，修正グラムシュミット（Modified Gram-Schmidt，以下 MGS）アルゴリズム，Daniel-Gragg-Kaufman-Stewart 型グラムシュミット（以下 DGKS）アルゴリズム[2] という 3 種類のアルゴリズムがある．他にもハウスホルダ直交化アルゴリズムなどが知られているが，従来，上記 3 種類のグラムシュミットアルゴリズムがよく利用されてきた．以下にそれぞれのアルゴリズムとその計算速度と計算精度の特徴を説明する．

＜プログラム 2 ＞に CGS アルゴリズムを示す．

＜プログラム 2 ＞

```
subroutine single_orthogonal(W,V,M)
do j = 1, M-1
  s = 0.0d0
  do i = 1, N
    s = s + V(i, j)*w(i)
  enddo
  hr(j) = s
enddo
do i = 1, N
  s = 0.0d0
  do j = 1, M-1
    s = s + V(i, j)*hr(j)
  enddo
  w(i) = w(i) − s
enddo
```

このアルゴリズムは，前半の 2 重ループが BLAS2 アルゴリズムであるためキャッシュチューニングが有効である．したがって計算速度上，有効なアルゴリズムである．しかし，前半の 2 重ループにおける s の内積計算が丸め誤差を蓄積しやすい構造となっているため，数値的に不安定である．したがって計算精度上は不利なアルゴリズムである．

＜プログラム 3 ＞に MGS アルゴリズムを示す．

＜プログラム 3 ＞

```
subroutine single_orthogonal(W,V,M)
do j=1, M-1
  s = 0.0d0
  do i =1, N
    s = s + V(i, j)*w(i)
  enddo
  if (s .ne. 0.0d0) then
    do i=1, N
      w(i) = w(i) − s*V(i, j)
    enddo
  endif
enddo
```

このアルゴリズムは，計算精度が高く安定しており，従来から，様々なライブラリに利用されてきた．これ

はsの内積演算での丸め誤差が，直後の「w(i) = w(i) − s*V(i, j)」によって緩和されるためである．しかし，BLAS 演算としては BLAS1 でしかなく，キャッシュチューニングの効果が出にくいため，計算速度上は不利である．

＜プログラム 4＞に DGKS アルゴリズムを示す．このアルゴリズムは CGS アルゴリズムと同様にキャッシュチューニングに向く．ただし条件 (1) によって繰返し計算を行うため，CGS ほどは計算速度上，有利ではない．しかし，一方で条件 (1) によって CGS よりも計算精度は高くなる．

＜プログラム 4＞

- 第 1 に，CGS を実施する．
- 第 2 に，以下の (1) を満たすまで CGS を繰り返し実行する．

$$||w||_2 < \eta ||Vw||_2 \qquad (1)$$

ただし，実装上，Vw のノルムは，＜プログラム 2＞の hr(i) のノルムとする．これによって，計算量を削減できる．なお，3 章に述べる数値実験においては，文献 3) での本アルゴリズムに関する検討結果を参考にして $\eta = \frac{1}{\sqrt{2}}$ とした．

2.2 異種混合計算機環境向け実装上の問題点

一般に，数値計算ライブラリの計算速度と計算精度に対するユーザのニーズは様々である．概算的に計算結果を得たい場合であれば，計算精度が低くても，短時間での処理が求められる．逆に，悪条件の問題を計算する場合では，時間がかかっても高精度な計算が求められる．計算時間と計算精度の組に対するこのようなユーザのニーズを「数値計算ポリシー」[1] と呼ぶことにする．

直交化処理を含む従来の数値計算ライブラリは，計算精度を最優先した設計になっており，かつ，利用する直交化アルゴリズムは 1 種類に限定されている．特に，上記 3 種類の直交化アルゴリズムの中で，比較的高精度かつ安定であるという理由から，MGS を利用するケースが多い．しかし文献 1) において，単一 CPU の PC 上での数値実験によって，CGS，MGS，DGKS の 3 種類の直交化アルゴリズムの中で最適なものが，入力されるデータ，および数値計算ポリシーに応じて変わってくることが明らかになった．

さらに，**図 1** に示す異種混合計算機環境においては，計算プラットフォームのアーキテクチャに応じたアルゴリズム選択も必要になってくる．図 1 に示すとおり，ユーザが最初は PC で小規模な計算を実行し，

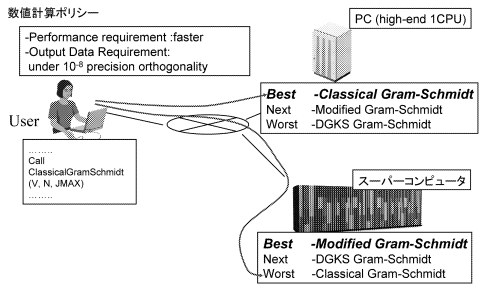

図 1 異種混合計算機環境における従来の直交化ライブラリの利用状況
Fig. 1 Use case of the conventional orthogonal libraries on heterogeneous platforms.

その後，大規模な計算をスーパコンピュータで実行する際には，計算プラットフォームのアーキテクチャによって，ユーザの数値計算ポリシーを満たすアルゴリズムが変わってくる可能性がある．特に，計算機アーキテクチャが並列になると，上記 3 つのアルゴリズムの並列化効率がそれぞれ異なること，および，並列化にともなう演算順序の違いから計算精度も変わってくること，の 2 点が予想されるため，その傾向は顕著になると考えられる．

このように，ベクトル群の直交化を行う計算処理において，数値計算ポリシーを満たすためには，入力データだけでなく，計算プラットフォームごとに最適なアルゴリズムをうまく選択する必要がある．しかし，従来の直交化ライブラリにはそのような機能がないため，数値計算ポリシーを満たす計算が実行できていなかった．

3. 異種混合計算機環境における直交化アルゴリズムの性能および精度の数値実験

本章では，後に説明する「ポリシー入力型グラムシュミット直交化ライブラリ」の異種混合計算機環境での適合性を評価するための基礎データの採取を目的として，PC，およびスーパコンピュータ SR11000 の 1 CPU，16 CPU において上記 3 種類の直交化処理に関する数値実験を行い，それぞれのアルゴリズムの計算精度および計算時間の評価を行う．

3.1 数値実験に用いた例題，および評価指標

本数値実験では，直交性にばらつきのある以下の 2 種類のベクトル群 $v_j(j = 1, \cdots, JMAX)$ を例題として用いた．各例題とも直交化の対象となるベクトル要素 $v_j(i)$ の生成において，合同乗算法による一様乱数 $x(k)$ を用いた．なお，これらの例題は，乱数のばらつき，cos 関数の連続的な値の変化，およびインデックスの倍数分のばらつきという 3 種類の外的なばらつ

表 1 数値実験に用いた計算プラットフォームのスペック
Table 1 Platform specification for numerical experiments.

環境属性	PC	SR11000（スーパコンピュータ）
ハードウェア プラットフォーム	Hitachi FLORA370DG CPU：Pentium 4（HT）3.2 GHz L1 キャッシュ：8 KB, L2 キャッシュ：512 KB 主記憶：512 MB DDR-SDRAM	Hitachi SR11000 model J1 1node（16 CPU），CPU：Power5 1.9 GHz L1 キャッシュ：32 KB，L2 キャッシュ：1.875 MB L3 キャッシュ：288 MB，主記憶：128 GB
コンパイラ	Intel FORTRAN Compiler Version 8.0	Hitachi FORTRAN for SR11000, 01-01
コンパイル オプション	-O3 -prefetch -unroll -align -lowercase -nodps -fpp2 -tpp7 -xW -vec_report3 -opt_report	f90 -64 -Os -noscope -parallel （1 CPU, 16 CPU 共通．実行時，1 CPU の場合は %>a.out -F'PRUNST(THREADNUM(1))' 16 CPU の場合は %>a.out -F'PRUNST(THREADNUM(16))' となる．）

きを基に形式的に構成した．以下，例題ごとにベクトル v_j の i 番目の要素を記載する．

- **例題 1**

$$v_j(i) = x(k) * j + cos\left(\frac{i*j}{N+1}\right) + i * 0.01$$

- **例題 2**

$$v_j(i) = x(k) + i * j * 0.01$$

ただし，$i = 1, ..., N$ および $k = i + (j-1) * N$ である．

いずれの例題でもベクトルの本数（JMAX）を 128 とした．これは，グラムシュミットの直交化アルゴリズムがランチョス法に利用される場合に比較的使われると想定される数値である．なお，本例題は，文献 1) における 3 つの例題のうち，数値のばらつきが大きい 2 つの例題を採用したものである．残りの 1 つの例題は，計算時間は上記 2 つの例題とほぼ同じであり，かつ計算精度は，上記 2 つよりも計算誤差が小さかったため，以下，上記 2 つの例題に関して評価結果を述べることとする．

また，計算速度の評価指標として，直交化処理の実行時間とし，また，計算精度の評価指標として，式 (2) に定義するベクトル群の直交性誤差 Ortho を用いた．

$$Ortho = ||V^t V - I||_F \quad (2)$$

ただし I は単位行列であり，$||*||_F$ はフロベニウスノルムである．また，V は j 番目の列がベクトル v_j である行列であり，列数（すなわち固有ベクトル本数）は JMAX とする．

3.2 数値実験結果

各例題につき，**表 1** に示す実験環境で計算時間と直交性誤差（計算精度）を評価した．なお，ベクトル長 N については，PC 上では 10000 から 100000 まで 10000 おきに，SR11000 上では，10000 から 300000 まで 10000 おきに値を変えて計算した．

図 2 に PC 上での CGS，MGS，DGKS による各例題の計算時間をそれぞれ示す．いずれの例題もベクトル長 N=10000 から 30000 の場合には，MGS が最も高速であり，次に CGS，DGKS となったが，ベクトル長が 30000 を超える領域では CGS が最も高速であり，次に MGS，DGKS と順番が入れ替わることが分かった．これは，CGS および DGKS がベクトル長に対してほぼ線形に計算時間が増加するのに対し，MGS では，ベクトル長が 40000 において計算時間が増加してしまっているためである．MGS の BLAS1 演算処理が，PC の L2 キャッシュサイズを超えたためと考えられる．

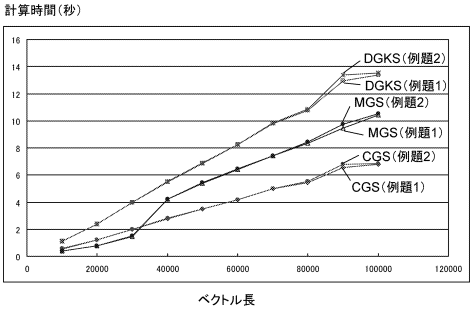

図 2 PC 上での CGS，MGS，DGKS の計算時間
Fig. 2 Execution time of CGS, MGS and DGKS on PC.

図 3 に SR11000 の 1 CPU 上での CGS，MGS，DGKS による各例題の計算時間を，**図 4** には SR11000 の 16 CPU 上での CGS，MGS，DGKS による各例題の計算時間を示す．

1 CPU の場合には，いずれの例題でも，どのベクトル長でも CGS が最も高速であり，次に MGS，DGKS の順になった．しかし，16 CPU のケースでは，ほとんどのベクトル長で DGKS が最も計算時間がかかるという点では共通しているものの，ベクトル長が 10 万未満では CGS が MGS よりも高速であり，ベクトル長が 10 万以上では MGS が CGS よりも高速になるという結果になった．また，16 万を超えたところ

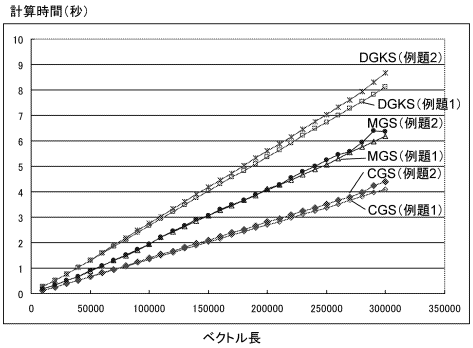

図 3 SR11000（1 CPU）上での CGS，MGS，DGKS の計算時間

Fig. 3 Execution time of CGS, MGS and DGKS on SR11000 (1 CPU).

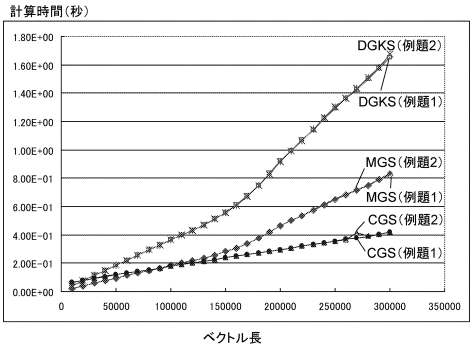

図 4 SR11000（16 CPU）上での CGS，MGS，DGKS の計算時間

Fig. 4 Execution time of CGS, MGS and DGKS on SR11000 (16 CPU).

から，CGS および DGKS は実行時間の増加度合いが増し，ベクトル長が 30 万の場合には MGS と比べて CGS は約 2 倍，DGKS は約 4 倍もの時間を要している．

この違いの理由は次のとおりである．MGS では，＜プログラム 3＞において，内側の i のループの処理が 16 CPU で並列化されている．このため，ベクトル群 $V(i,j)$ の i の範囲が 16 分割され，1 CPU あたりのベクトル群 $V(i,j)$ へのデータアクセス範囲は L3 キャッシュの容量を超えない．一方，CGS および DGKS では，＜プログラム 2＞において外側の j のループの処理が 16 CPU で分割されており，$V(i,j)$ の i の範囲は 16 分割されない．このため，1 CPU あたりのベクトル群 $V(i,j)$ へのデータアクセス範囲が L3 キャッシュの容量を超えてしまう．このように，ループ構造の違いから並列化される箇所が変わり，データアクセス範囲とキャッシュ容量の関係が異なるため上記のような性能差が生じる．

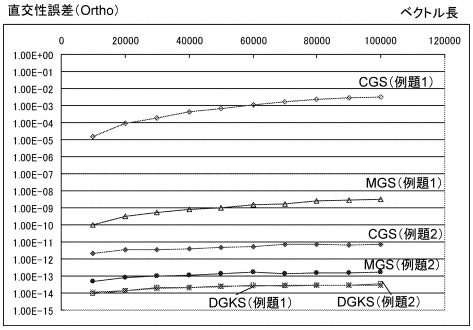

図 5 PC 上での CGS，MGS，DGKS の直交性誤差

Fig. 5 Orthogonal error of CGS, MGS and DGKS on PC.

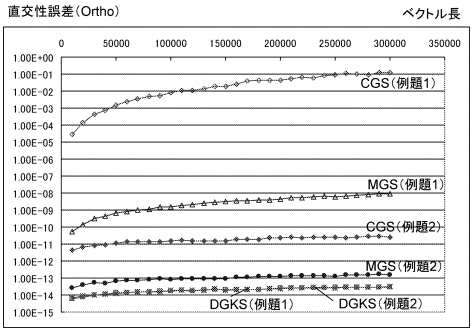

図 6 SR11000（1 CPU）上での CGS，MGS，DGKS の直交性誤差

Fig. 6 Orthogonal error of CGS, MGS and DGKS on SR11000 (1 CPU).

以上のように，計算時間の短いアルゴリズムは，ベクトル長および計算プラットフォームによって異なるという結果になった．

図 5 に PC 上での CGS，MGS，DGKS の各例題における直交性誤差をそれぞれ示す．例題 1 に関して，CGS は非常に悪く，直交性誤差が単精度レベル（10^{-8}）を大幅に上回ってしまっている．一方，MGS はおおよそ単精度レベルを下回っており，DGKS は 10^{-13} 以下と非常に良好な精度を保っていることが分かる．また，例題 2 に関して，CGS は 10^{-11} 以下となっており，十分な精度で計算できているといえる．MGS および DGKS はさらに高い精度で計算できていることが分かる．さらに，CGS や MGS では例題による差が大きいのに対して，DGKS の計算精度が安定して高いことも分かる．

次に，**図 6** に SR11000（1 CPU）上での CGS，MGS，DGKS の各例題における直交性誤差をそれぞれ示す．例題 1 に関して，CGS は非常に悪く，また MGS も今回の計測の範囲ではおおよそ単精度（10^{-8}）は確保されたが，ベクトル長が 30 万より大きいと単精度レベルを超えてしまうと予測される．しかし，DGKS

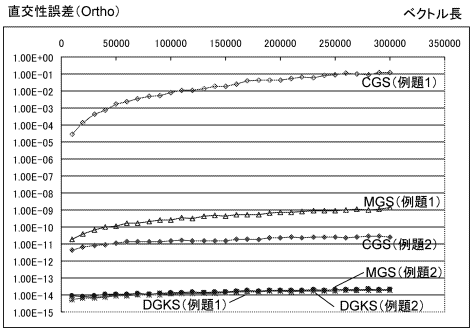

図 7 SR11000（16 CPU）上での CGS，MGS，DGKS の直交性誤差
Fig. 7 Orthogonal error of CGS, MGS and DGKS on SR11000 (16 CPU).

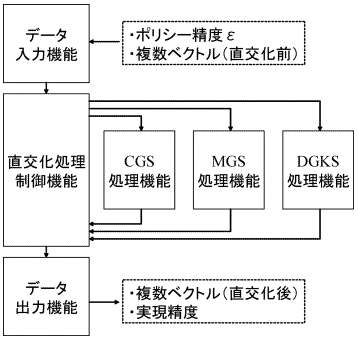

図 8 数値計算ポリシー Policy(ε) を入力とする直交化ライブラリの処理フロー
Fig. 8 Flow of orthogonalization library with numerical policy Policy(ε).

では 10^{-13} 以下と非常に良好な精度を保っていることが分かる．また，例題 2 に関して，CGS は 10^{-10} 以下となっており，十分な精度で計算できているといえる．MGS および DGKS はさらに高い精度で計算できていることが分かる．

図 7 に SR11000（16 CPU）での結果を示す．ここでは，MGS の精度が 1 CPU の場合よりも 1 桁程度高精度になることが分かった．それ以外の結果については，SR11000（1 CPU）のときと同様の結果であった．計算精度の結果は，MGS のみが並列化の効果によって若干変化するものの，どのプラットフォームにおいても CGS < MGS < DGKS の順に高精度になっていき，また，CGS および MGS は例題による差が大きく，DGKS は，例題によらず高精度であることが分かった．

4. 数値計算ポリシー入力型直交化ライブラリの異種混合計算機環境における評価

4.1 数値計算ポリシー入力型直交化ライブラリの処理フロー

本節では，数値計算ポリシー入力型の直交化ライブラリの処理フローにつき説明する．まず，本研究での数値計算ポリシーは次のように設定した．

Policy(ε)：
「直交化精度 Ortho が ε よりも小さくなる範囲において，実行時間が最も短い直交化処理を実行したい」

この数値計算ポリシーが反映されると，必要以上に高精度な計算処理を得るために長い時間待たなくて済むことが期待される．以下，Policy(ε) の ε をポリシー精度と呼ぶことにする．

本直交化ライブラリは，通常の直交化ライブラリのインタフェースに，ポリシー精度 ε（入力）および実現精度 Ortho（出力）が追加される．

次に，本直交化ライブラリの処理フローを徒競走方式によって以下のように構成する．本方式では，ポリシー精度 ε を満たすため，**図 8** に示すように，CGS，MGS，DGKS の 3 つの直交化機能による徒競走方式として処理フローを構成する[4]．図 8 に従い，本処理フローを説明する．

(1) データ入力機能では，ポリシー精度 ε および直交化対象のベクトル群を入力する．
(2) 直交化処理制御機能および付随する 3 つの直交化処理機能では，以下の処理を行う．
  (a) CGS 処理機能，MGS 処理機能，DGKS 処理機能は，同時に各々の直交化処理を実行する．
  (b) 上記 3 つの直交化処理機能は，処理が終了したら，直交化処理制御機能に直交化されたベクトル群を返す．
  (c) 直交化処理制御機能は，ベクトル群の直交性 Ortho を計算し，ε を超えた場合には計算結果として採用せず，ε 以下の場合には計算結果として採用する（この Ortho を「実現精度」，採用された処理を「選択処理」と呼ぶ）．採用すると同時に，他の直交化処理を中断する．どの直交化機能の Ortho も ε を超えた場合は，Ortho が最小であるベクトル群を返す．
(3) データ出力機能では，計算結果である直交化されたベクトル群と実現精度を出力する．

なお，本報告では，グリッドミドルウェアなどによる実装がなされたと仮定したうえでの，異種分散環境での数値計算ポリシーの有効性を検討する．

表 2 例題 1 に対する SR11000（16 CPU）における方法①/方法②による選択処理の差
Table 2 Difference between selections with method 1 and with method 2 on the SR11000 (16 CPU) for test 1.

ベクトル長	ポリシー精度	方法①による選択処理	方法②による選択処理	SR11000（16 CPU における両選択処理の差）	
				実行時間の差	実現精度（直交性誤差）の差
100000	1.00E-08	MGS	MGS	-	-
	1.00E-09	DGKS	MGS	0.363 秒→ 0.179 秒	1.11E-14 → 2.68E-10
	1.00E-10	DGKS	DGKS	-	-
300000	1.00E-08	DGKS（予測）	MGS	1.652 秒→ 0.419 秒	1.98E-14 → 1.38E-09
	1.00E-09	DGKS（予測）	DGKS	-	-
	1.00E-10	DGKS（予測）	DGKS	-	-

表 3 例題 2 に対する SR11000（16 CPU）における方法①/方法②による選択処理の差
Table 3 Difference between selections with method 1 and with method 2 on the SR11000 (16 CPU) for test 2.

ベクトル長	ポリシー精度	方法①による選択処理	方法②による選択処理	SR11000（16 CPU における両選択処理の差）	
				実行時間の差	実現精度（直交性誤差）の差
100000	1.00E-08	CGS	MGS	0.183 秒→ 0.179 秒	1.55E-11 → 1.36E-14
	1.00E-09	CGS	MGS	0.183 秒→ 0.179 秒	1.55E-11 → 1.36E-14
	1.00E-10	CGS	MGS	0.183 秒→ 0.179 秒	1.55E-11 → 1.36E-14
	1.00E-11	CGS	MGS	0.183 秒→ 0.179 秒	1.55E-11 → 1.36E-14
300000	1.00E-08	CGS（予測）	MGS	0.837 秒→ 0.418 秒	2.48E-11 → 2.19E-14
	1.00E-09	CGS（予測）	MGS	0.837 秒→ 0.418 秒	2.48E-11 → 2.19E-14
	1.00E-10	CGS（予測）	MGS	0.837 秒→ 0.418 秒	2.48E-11 → 2.19E-14
	1.00E-13	DGKS（予測）	MGS	1.674 秒→ 0.418 秒	1.96E-14 → 2.19E-14

4.2 数値実験結果

本節では，ユーザがプログラム上でグラムシュミット直交化ライブラリを呼び出しており，そのプログラムを PC から SR11000（16 CPU）へと利用環境を移したという想定のもと，数値計算ポリシー入力型直交化ライブラリ方式の性能，精度の評価を行う．

そのため，まず，ユーザが PC 上のみで数値計算ポリシーを満たすアルゴリズムを探し，そのアルゴリズムをそのままスーパコンでも実行する方法を，①「PC 上において徒競走方式によって選択された直交化アルゴリズムを，SR11000（16 CPU）上でそのまま SR11000 の 16 CPU で計算する方法」とする．数値計算ポリシー入力型直交化ライブラリ方式，すなわち②「SR11000（16 CPU）でも徒競走を行い改めて選択し直した直交化アルゴリズムによって計算する方法」の性能と精度を①のそれらと比較することで，本ライブラリ方式を評価する．

表 2 に例題 1 に対する上記の比較結果を，**表 3** に例題 2 に対する上記の比較結果をそれぞれまとめる．例題 2 ではベクトル長 10 万および 30 万のそれぞれに対し，ポリシー精度 ε を 10^{-8}，10^{-9}，10^{-10} と変化させて比較した．ベクトル長が 30 万の方法①による選択処理（方法①で選択される直交化処理）としては，図 2 および図 5 の結果から採用されると予測される直交化処理を選択した．評価の結果，方法①と②で同じ直交化処理が選択される場合もあったが，異なる直交化処理が選択された，ベクトル長が 30 万，ポリシー精度 ε が 10^{-8} の場合に，実行時間が 1.652 秒から 0.419 秒となり，約 3.96 倍の性能向上が得られた．

例題 2 ではベクトル長 10 万に対し，ポリシー精度 ε を 10^{-8}，10^{-9}，10^{-10}，10^{-11} と変化させ，ベクトル長 30 万に対し，ポリシー精度 ε を 10^{-8}，10^{-9}，10^{-10}，10^{-13} と変化させて比較した．ベクトル長が 30 万の方法①による選択処理としては，図 2 および図 5 の結果から採用されると予測される直交化処理を選択した．評価の結果，ベクトル長が 30 万，ポリシー精度 ε が 10^{-13} の場合に実行時間が 1.674 秒から 0.418 秒となり，約 4.00 倍の性能向上が得られた．また，ベクトル長が 10 万であり，ポリシー精度 ε が 10^{-11} の場合には，方法①による選択処理での結果が 1.55×10^{-11} となり，ポリシー精度 10^{-11} を満足しなくなったが，本ライブラリ方式では，1.36×10^{-14} となり，ポリシー精度 10^{-11} を満たすようにすることができた．

5. 関 連 研 究

本研究で扱った異種混合計算機環境における行列ライブラリについて，以下，関連研究について述べる．第 1 に，国内の事例として，東大および電通大のグループがそれぞれ，I-LIB [5),6)]，ABC-LIB [7),8)] という自動

チューニング型ライブラリの研究を行っており，様々な計算機向けに多重ループのアンローリングを自動化するツールを開発している．第2に，海外の事例として，テネシー大の Dongarra らが SANS（Self-Adapting Numerical Software）[9] という PC クラスタ向け自己適合型の行列ライブラリ構成方式および，ATLAS [10] という PC 上の高品質なライブラリを発表している．また，カリフォルニア大の Frigo らが適合型の FFT に関する研究[11] を行い，FFTW [12] というライブラリを発表している．また，応用分野を限定しつつもさらに適合度を高める FFT の研究[13] などもあり，適合型ライブラリの研究はプロトタイプソフトウェアが発表される段階にある．ただし，ソフトウェア製品として発表されているものはまだない．

報告者らは，自動チューニングのインタフェース形式について研究[14] を行っており，研究[1] では，ユーザの計算精度と処理性能に対する数値計算ポリシーを反映させる入力方式を提案した．本報告では，異種混合環境で実際に数値計算ポリシーを入力とする方式が有効であることを示した．これらは，特に以下の2点において，特長があると考えている．第1に，インタフェース実装方式について，他の研究では，計算性能のみを考慮しているのに対し，本研究では，ポリシー入力における計算精度と計算性能というインタフェースを提案している点である．第2に，チューニングの方式について，他の研究では単一アルゴリズムのパラメータ最適化にとどまっているのに対し，本研究では，徒競走方式を用いたアルゴリズム選択を含めた最適化を行っている点である．

6. まとめと今後の課題

6.1 ま と め

数値計算ポリシーを入力とする行列計算ライブラリ方式について異種混合計算機環境における有効性を評価した．本方式は，行列計算処理において，同じ機能を持つが異なる計算精度，計算時間となるアルゴリズムを同時に実行し，数値計算ポリシーとして与えられたユーザの要求精度を満たす最短処理時間での計算結果をユーザに引き渡す．

本報告では，特に，ユーザがプログラム開発環境としての PC から大規模計算実行環境としてのスーパコンピュータにプログラム実行環境を移した際に，満足な性能を得られていなかったグラムシュミット直交化処理について本方式を適用した．

数値計算ポリシー入力型のグラムシュミット直交化処理では，CGS，MGC，DGKS の3種類のアルゴリズムを徒競走方式で同時に実行し，ユーザが要求する最低限の直交性誤差 ε（ポリシー精度）以内で最短の時間で処理した結果をユーザに引き渡す．

本方式の評価データを得るため，まず，PC（1 CPU），スーパコンピュータ SR11000（1 CPU，16 CPU）上において，上記の 3 種類の直交化アルゴリズムの計算時間，計算精度を評価したところ以下のことが分かった．

(1) 計算時間については，直交化対象のベクトル群におけるベクトル長，および計算プラットフォームによって，最適なアルゴリズムが大きく変わってくる．特に，CGS と MGS は，ベクトル長，および計算プラットフォームが並列か否かで，実行時間が互いに入れ替わることが多く，選択が容易ではない．

(2) 計算精度については，MGS のみが並列化の効果によって若干変化するものの，どの計算プラットフォームにおいても CGS < MGS < DGKS の順に高精度になっていき，また，CGS および MGS は扱う入力データの違いによる差が大きく，DGKS は，扱う入力データによらず高精度であることが分かった．

これらの数値実験の結果を用いて，数値計算ポリシーを入力とするグラムシュミット直交化ライブラリについて，PC から SR11000（16 CPU）へと利用環境を移したという想定のもと，次のように，性能，精度の評価を行った．

まず，ユーザが PC 上のみで数値計算ポリシーを満たすアルゴリズムを探し，そのアルゴリズムをそのままスーパコンでも実行する方法を，①「PC 上において徒競走方式によって選択された直交化アルゴリズムを，SR11000（16 CPU）上でそのまま SR11000 の 16 CPU で計算する方法」とする．そのうえで，数値計算ポリシー入力型直交化ライブラリ方式，すなわち②「SR11000（16 CPU）でも徒競走を行い改めて選択し直した直交化アルゴリズムによって計算する方法」の性能と精度を①のそれらと比較することで，本ライブラリ方式を評価した．

評価の結果，以下のことが分かった．

(1) 方法①と②で同じ直交化処理が選択される場合もあったが，異なる直交化処理が選択された，ベクトル長が 30 万，ポリシー精度 ε が 10^{-13} のあるケースで，実行時間が 1.674 秒から 0.418 秒となり，約 4.00 倍の性能向上が得られる例があった．

(2) 計算精度の面では，ベクトル長が 10 万であり，

ポリシー精度 ε が 10^{-11} のあるケースで，方法①による直交化処理での結果が 1.55×10^{-11} となり，ポリシー精度 10^{-11} を満足しなくなったが，本ライブラリ方式では，1.36×10^{-14} となり，ポリシー精度 10^{-11} を満たすようにできた例があった．

6.2 今後の課題

以下に今後の課題を列挙する．

(1) 今回検討した直交化処理は，疎行列反復解法や，固有値解法において重要な役割を果たす．提案方法をこれらの解法に実装する方式を検討する．

(2) 今回検討したユーザの計算ポリシーは，計算時間と計算精度のみであった．使用するメモリ量や CPU 数など計算機資源の量を加えたポリシーについて入力できるライブラリ構成方法を検討する．

(3) 今回の評価では，徒競走方式に関し，グリッドミドルウェアなどによる実装がなされたと仮定したうえでの，異種分散環境での数値計算ポリシーの有効性の評価であった．今後は，徒競走方式をグリッドミドルウェアなどによる実装を含めたうえでの性能評価を行っていく．

謝辞 本研究に至るうえで共同研究を通じ重要な数多くの寄与をいただいた電気通信大学の今村俊幸講師に謝意を表します．自動チューニング研究に関し多くの示唆をいただいた電気通信大学の弓場敏嗣教授，片桐孝洋助手，東京大学の須田礼仁助教授，名古屋大学の山本有作講師に謝意を表します．

参 考 文 献

1) 直野 健，猪貝光祥，木立啓之：数値計算ポリシーを入力とするベクトル群の直交化ライブラリの構成方法，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG7(ACS10), pp.35–43 (2005).
2) Daniel, J., Gragg, W.B., Kaufman, L. and Stewart, G.W.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, *Mathematics of Computation*, Vol.30, pp.772–795 (1976).
3) Lehoucq, R.: Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration, Ph.D. thesis, Rice University, Houston, TX (1995).
4) 直野 健，猪貝光祥，木立啓之：徒競走型の性能保証レベル向上方法の検討，情報処理学会研究報告 2004-HPC-99 (SWoPP2004)，pp.97–102 (2004).
5) Kuroda, H., Katagiri, T. and Kanada, Y.: Knowledge Discovery in Auto-tuning Parallel Numerical Library, Progress in Discovery Science, *Final Report of the Japanese Discovery Science Project*, Lecture Notes in Computer Science 2281, pp.628–639, Springer (2002).
6) project I-LIB. http://www.super-computing.org/ kuroda/nadia.html
7) Katagiri, T., Kise, K., Honda, H. and Yuba T.: FIBER: A General Framework for Auto-Tuning Software, *5th International Symposium on High Performance Computing* (*ISHPC-V*), LNCS 2858, pp.146–159, Springer (2003).
8) 自動ブロック化・通信最適化ライブラリ ABC-LIB. http://www.abc-lib.org/
9) Dongarra, J. and Eijkhout, V.: Self-adapting numerical software for next generation applications, *International Journal of High Performance Computing Applications*, Vol.17, No.2, pp.125–131 (2003).
10) ATLAS project. http://www.netlib.org/atlas/index.html
11) Frigo, M. and Johnson, S.: FFTW: An adaptive software architecture for the FFT, *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing 3*, pp.1381–1384 (1998).
12) FFTW Homepage. http://www.fftw.org/
13) Sundararajan, E., Premaratne, M., Karunasekera, S. and Harwood, A.: Algorithmic-Parameter Optimization of a Parallelized Split-Step Fourier Transform Using a Modified BSP Cost Model, *Proc. 2nd Symposium on Parallel and Distributed Processing and Applications* (*ISPA-2004*), LNCS 3358, pp.245–256, Springer (2004).
14) 直野 健，山本有作：単一メモリ型インターフェイスを有する自動チューニング並列ライブラリの構成方法，情報処理学会研究報告 2001-HPC-87 (SWoPP2001)，pp.25–30 (2001).

(平成 17 年 1 月 6 日受付)
(平成 17 年 4 月 28 日採録)

直野　健（正会員）

1968 年生．1994 年京都大学大学院理学研究科数理解析専攻修士課程修了．同年（株）日立製作所中央研究所入所．以来，並列計算機 SR2201，SR8000，SR11000 向け行列計算ライブラリの研究開発に従事．現在の主たる研究テーマは，HPC の研究，並列固有値計算ライブラリ，HPC の金融工学への応用．日本応用数理学会，日本金融・証券計量・工学学会各会員．

猪貝　光祥（正会員）

1963 年生．1987 年横浜市立大学理学部物理学課程卒業．同年（株）日立超 LSI システムズ入社．以来，並列計算機用数値計算ライブラリ，数値シミュレーション関連ソフトウェアの設計開発に従事．

木立　啓之

1977 年生．2000 年室蘭工業大学情報工学科卒業．同年（株）日立超 LSI システムズ入社．以来，並列計算機向け行列計算ライブラリの開発に従事．