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

直野 健[†] 猪貝光祥[††] 木立啓之[††]

計算精度と計算時間の調整が課題であったグラムシュミット直交化処理において，与えられた最低計算精度の範囲内で最速計算を実行する方法を提案した．本方法では，性質の異なる古典グラムシュミット，修正グラムシュミット，DGKS 型グラムシュミットの処理を同時に実行し，ユーザが与えた精度と速度の要求に最も適合する計算結果を選択する徒競走方式を採用した．PC 上で提案法を実行し，その効果を検証した結果，従来手法では直交性誤差が 10^{-8} を満たさなかったケースでその誤差を 10^{-14} に低減できた．また，計算時間についても最大約 4.8 倍の高速化を達成し，精度と速度の両面で提案法が有効であることが明らかになった．

Orthogonalization Library with a Numerical Computation Policy Interface

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

We propose an orthogonalization library that can execute faster computations under the conditions of required accuracy to control the balance between the orthogonal accuracy of the obtained vectors and the computation time. We design the library of the orthogonalization by race-based method, which execute in the same time the classical Gram-Schmidt, the modified Gram-Schmidt, and the DGKS type Gram-Schmidt, and take the fastest result that satisfies the required accuracy. The experiments of the method for three different types of vectors on PC (Pentium4, 3.2 GHz) show that, the method can achieves better accuracy even in the case that the single orthogonalization fails in over 10^{-8} orthogonality, and that the method achieves about 4.8 times faster in the best case.

1. はじめに

近年，スーパー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ー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，複数のスーパーコンあるいは PC クラスタを連携させて利用する傾向になりつつある．従来，行列ライブラリの開発においては，特定スーパーコン向けの性能チューニングに多くの工数を費やしてきた．しかし今後，PC クラスタを含めた GRID コンピューティング環境向けには，行列ライブラリの機能として以下の 3 つが新たに必要になってくる．

第 1 に，行列ライブラリの自動チューニング機能である．多数の機種に対するチューニングを従来のように計算プラットフォームごとにハンドコーディングすると莫大な工数を要してしまう．そこで計算機プラットフォームに自動的に適合したチューニングを行う，自動チューニング技術が必要になってくる．このような技術に関して，すでに多くの研究[1)~6)] が行われている．

第 2 に，ユーザ要求性能に対するライブラリ性能保証機能である．上記の自動チューニングでは，特定の行列サイズ N（たとえば 1000，2000，3000 など）の条件下でのピンポイント的な測定データからチューニングパラメータの値を決定していた．ところがそうして得られたチューニングコードを用いても，行列サイズ N の値によっては大幅な性能劣化を引き起こす場合があることが明らかになった[7),8)]．このような結果を受けて，ライブラリ性能を保証する機能が必要であることが認識されるようになった．この分野では，我々がいち早くその重要性を認知し，性能保証ラインを算出する方式[9)] や性能保証ラインを向上させる方法[10)] を提案した．

第 3 にユーザの計算ポリシーを反映する機能である．従来，数値処理アルゴリズムでは精度評価と性能

† 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.

評価が独立に論じられることが多かった．しかし，現実には，精度はある程度悪くなっても高速な処理が必要である場合，あるいは時間がかかっても高精度な計算結果が必要である場合など，ユーザごとに計算上のポリシーが異なる．しかし，このような計算精度と計算時間のバランスを考慮した数値計算ライブラリがなく，そのため，ユーザは計算時間と計算精度の調整に悩まされることが少なくない．このような課題を解決するための機能に関する研究分野は我々の知る限り，未開拓の領域である．

そこで，本研究では，この第 3 の機能を有する行列ライブラリの開発を目的とする．

2. 従来の直交化アルゴリズムと直交化処理を含む数値計算ライブラリの問題点

2.1 従来の直交化アルゴリズム

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．当然，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 $V(i, j); i = 1, ..., N$ 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，次の 3 種類のアルゴリズムによって違いがある．

(1) 古典グラムシュミット（Classical Gram-Schmidt，以下 CGS）アルゴリズム
(2) 修正グラムシュミット（Modified Gram-Schmidt，以下 MGS）アルゴリズム
(3) Daniel-Gragg-Kaufman-Stewart 型グラムシュミット（以下 DGKS）アルゴリズム[11)]

ほか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について説明する．

2.1.1 古典グラムシュミット（CGS）アルゴリズム

〈プログラム 2〉に CGS アルゴリズムを示す．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 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
```

2.1.2 修正グラムシュミット（MGS）アルゴリズム

〈プログラム 3〉に MGS アルゴリズムを示す．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 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
```

2.1.3 Daniel-Gragg-Kaufman-Stewart 型グラムシュミット（DGKS）アルゴリズム

〈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) によって繰り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〉
- 第一に，CGS を実施する．
- 第二に，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 12) 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 直交化処理を含む従来の数値計算ライブラリの問題点

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．ここでは，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」と呼ぶことにする．

これに対して，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．実際，CGS を利用すると，直交基底の十分な計算精度が得られない場合があるため，それを利用した反復解法で数値計算が破綻する事例も知られている．一方，文献 12) によって，MGS を利用した場合であっても CGS と同様に計算が破綻するように，直交化処理対象のベクトル群を原理的に構成可能なことが明らかになった．そのため，疎行列向け固有値ライブラリである ARPACK [13] では，高精度な直交化処理として DGKS を採用している．

いずれにしても，従来の数値計算ライブラリでは，1 種類の直交化アルゴリズムしか利用できないため，ユーザの数値計算ポリシーにそぐわない状態での計算処理となる場合があり，問題であった．

3. 従来のグラムシュミット直交化の計算時間と計算精度に関する数値実験

本章では，数値計算ポリシーを入力とする直交化ライブラリを構成するための基礎データの採取を目的として，PC 上において上記 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，例題として，以下の 3 種類のベクトル群 $v_j(j = 1, \cdots, JMAX)$ を用いた．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きを元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**

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

- **例題 2**

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

- **例題 3**

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

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 20 あるいは 100 とした．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．

$$Ortho = ||V^tV - I||_F \qquad (2)$$

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）

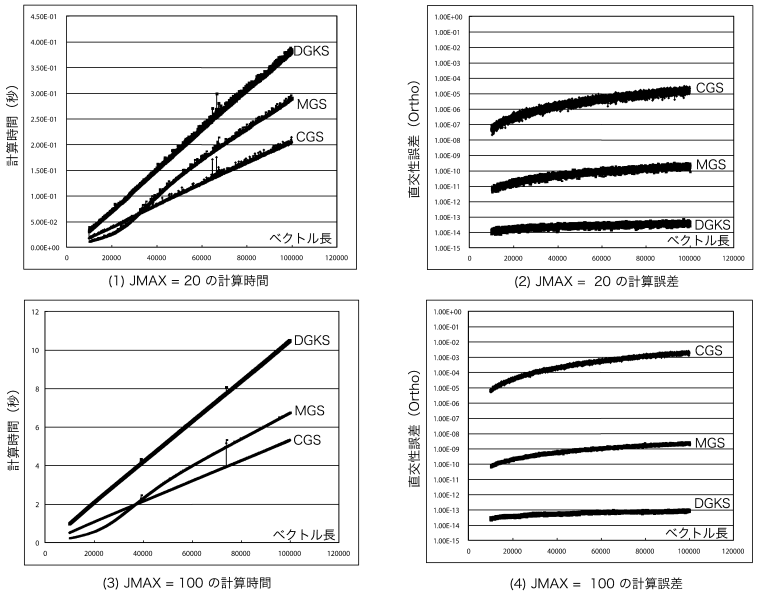

図 1 例題 1 における CGS，MGS，DGKS の計算時間および計算誤差
Fig. 1 Execution time and orthogonality error of CGS, MGS and DGKS for Sample 1.

は JMAX とする．

3.2 数値実験結果

各例題につき，ベクトル長 N を 10000 から 100000 まで 100 おきに設定し，以下の実験環境で計算時間と直交性の誤差を評価した．

- **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）
- **コンパイラ**
  Intel FORTRAN Compiler Version 7.1
- **コンパイルオプション**
  `-O3 -prefetch -unroll -align -lowercase -nodps -fpp2 -tpp7 -xW -vec_report3 -opt_report`

3.2.1 例題 1 の結果

図 1 に例題 1 における CGS，MGS，DGKS の計算時間および計算精度をそれぞれ示す．

まず，計算時間については，図 1 (1) および (3) の結果が示すとおり，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．また，この領域ではどのアルゴリズムでもほぼ線形に計算時間が増加していくことも確認された．最も計算時間のかかる DGKS は CGS のほぼ 2 倍弱の計算時間となることも確認できる．

次に計算精度については，図 1 (2) および (4) の結果が示すとおり，CGS は非常に悪く，直交性の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，特に JMAX = 20 の場合，CGS では直交性の誤差が，ベクトル長 N の微小な変化に対して 3 倍程度の幅を持って変化していることが分かる．

3.2.2 例題 2 の結果

図 2 に例題 2 における CGS，MGS，DGKS の計算時間および計算誤差をそれぞれ示す．まず，計算時間については，図 2 の (1) および (3) が示すとおり，例題 1 とほぼ同様の結果が得られ，N = 10000 から 30000 の領域では MGS が最も高速であり，N = 30000 を超える領域では CGS が最も高速であった．また，この領域ではベクトル長 N にほぼ線形に計算時間が増加することが確認できた．

次に，計算精度については，図 2 の (2) および (4) が示すとおり，CGS が最も低いものの，この例題では直交性の誤差が単精度分はほぼ保たれていることが分かる．しかし，ベクトル長 N の微小変化に対して約 10 倍程度と大きくぶれる現象が確認された．一方，MGS の精度は良好な水準でありかつぶれ幅も小さく，DGKS はさらに高精度かつ安定していることが分かる．

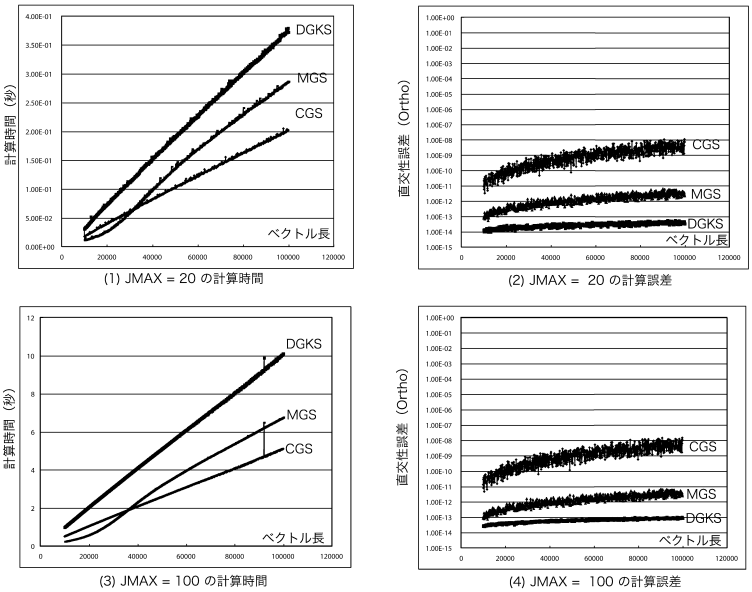

図 2 例題 2 における CGS，MGS，DGKS の計算時間および計算誤差
Fig. 2 Execution time and orthogonality error of CGS, MGS and DGKS for Sample 2.

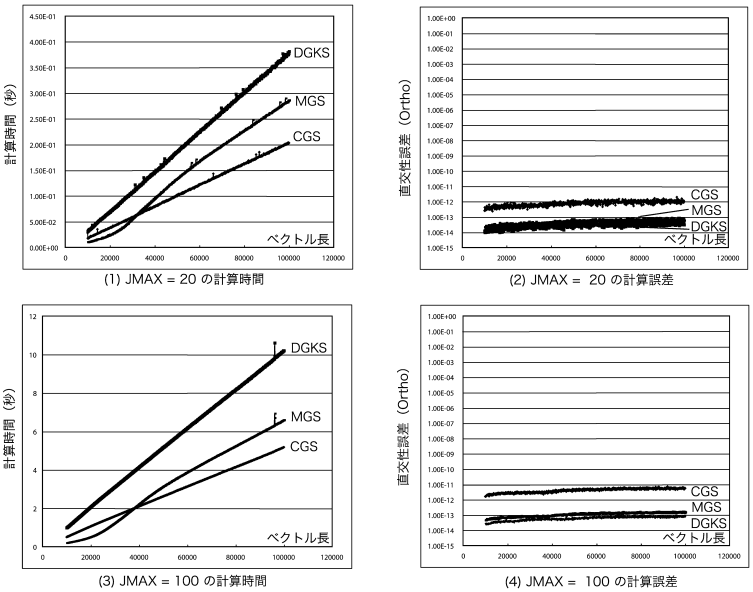

図 3 例題 3 における CGS，MGS，DGKS の計算時間および計算誤差
Fig. 3 Execution time and orthogonality error of CGS, MGS and DGKS for Sample 3.

3.2.3 例題 3 の結果

図 3 に例題 3 における CGS，MGS，DGKS の計算時間および計算誤差をそれぞれ示す．まず，計算時間については，図 3 の (1) および (3) が示すとおり，例題 1 および例題 2 とほぼ同様の結果が得られ，N = 10000 から 30000 の領域では MGS が最も高速であり，N = 30000 を超える領域では CGS が最も高速であった．また，この領域ではベクトル長 N にほぼ線形に計算時間が増加することが確認できた．次に，計算精度については，図 3 の (2) および (4) が示すとおり，他の例題の結果と同様に，CGS の誤差が最も大きく，次に MGS，DGKS の順になっているが，この例題では CGS も含めいずれのアルゴリズムでも直交性の誤差が小さく，かつ N の微小変化に対するぶれ幅も小さく安定していることが分かる．

以上の 3 つの例題から，ベクトル長が 10000，20000

と比較的短い場合には，従来から多くライブラリに採用されてきた MGS が計算時間，計算精度の両面で有効であることが確認された．しかし，ベクトル長が 40000，80000 と比較的長い場合には，実行時間では CGS < MGS < DGKS の順で長くなり，CGS が最も良いが，計算誤差は CGS > MGS > DGKS の順で小さくなり，DGKS が最も良いことが分かった．したがって，ベクトルの長さや，ユーザの数値計算ポリシーによっては，選択するべきアルゴリズムが異なってくることがいえる．

4. 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の処理方法

4.1 数値計算ポリシーの設定方法

前章で得られた各処理の計算時間および計算精度の結果を元に，数値計算ポリシーを入力とするベクトル群直交化処理ライブラリの構成方法を検討する．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．

4.2 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の処理フロー

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を徒競走方式[10] により構成する．本論文では，ポリシー精度 ε を満たすため，**図 4** 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．

図 4 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，この処理フローを以下，提案方法と呼ぶことにする．

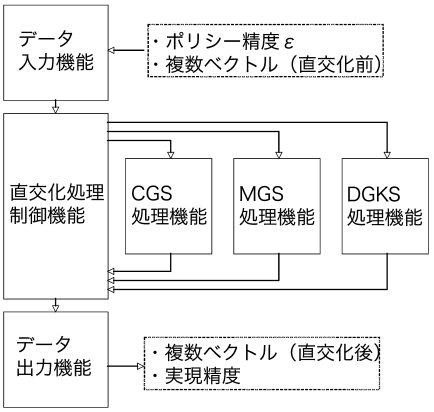

図 4 徒競走方式による数値計算ポリシー Policy(ε) を入力とする直交化ライブラリの処理フロー
Fig. 4 Processing flow of orthogonalization library with numerical policy $Policy(\epsilon)$ using race-based method.

4.3 提案方法の評価

提案方法に対して，ベクトル本数 JMAX = 100 として，ベクトル長 N = 10000，20000，40000，80000 の 4 パターンによって，それぞれポリシー精度 $\varepsilon = 10^{-8}$，10^{-10}，10^{-12} の 3 パターンを評価する．ポリシー精度をこの 3 パターンとしたのは，倍精度計算で単精度分の精度を最低必要とするユーザと，得られた直交基底をさらに利用するため単精度よりもやや高精度な計算を必要とするユーザを想定したためである．なお，提案方法は本来，図 4 での直交化処理制御機能を組み込んだ実装になるが，ここでは 3 種類の直交化処理機能の実現精度と実行時間のみに注目した評価を行った．また，評価環境は，3.2 節の冒頭に記載した実験環境と同じである．

表 1 に例題 1 の場合の結果を示す．それぞれのポリシー精度に対して，選択された処理，その実現精度と実行時間をそれぞれ第 3 列から第 5 列に示す．さらに，表の第 6 列から第 8 列には CGS，MGS，DGKS についてポリシー精度を満たさなかった場合を NG と

表 1 例題 1（JMAX = 100）における選択処理とその実現精度および実行時間，ならびに他の処理の精度，計算時間倍率の結果
Table 1 Accuracy and execution time of selected algorithm and of other algorithms for Sample 1 data (JMAX = 100).

ベクトル長	ポリシー精度	選択処理	実現精度 ($Ortho$)	実行時間 (sec)	対 CGS	対 MGS	対 DGKS
10000	1.00E-08	MGS	7.70E-11	0.2247	NG(7.43E-06)	–	4.331
	1.00E-10	MGS	7.70E-11	0.2247	NG(7.43E-06)	–	4.331
	1.00E-12	DGKS	2.63E-14	0.9731	NG(7.43E-06)	NG(7.70E-11)	–
20000	1.00E-08	MGS	2.17E-10	0.5695	NG(4.49E-05)	–	3.644
	1.00E-10	DGKS	3.53E-14	2.0752	NG(4.49E-05)	NG(2.17E-10)	–
	1.00E-12	DGKS	3.53E-14	2.0752	NG(4.49E-05)	NG(2.17E-10)	–
40000	1.00E-08	MGS	6.34E-10	2.3134	NG(1.86E-04)	–	1.809
	1.00E-10	DGKS	5.94E-14	4.1852	NG(1.86E-04)	NG(6.34E-10)	–
	1.00E-12	DGKS	5.94E-14	4.1852	NG(1.86E-04)	NG(6.34E-10)	–
80000	1.00E-08	MGS	1.67E-09	5.3774	NG(1.08E-03)	–	1.552
	1.00E-10	DGKS	7.45E-14	8.3461	NG(1.08E-03)	NG(1.67E-09)	–
	1.00E-12	DGKS	7.45E-14	8.3461	NG(1.08E-03)	NG(1.67E-09)	–

表 2 例題 2（JMAX = 100）における選択処理とその実現精度および実行時間，ならびに他の処理の精度，計算時間倍率の結果
Table 2 Accuracy and execution time of selected algorithm and of other algorithms for Sample 2 data (JMAX = 100).

ベクトル長	ポリシー精度	選択処理	実現精度 ($Ortho$)	実行時間 (sec)	対 CGS	対 MGS	対 DGKS
10000	1.00E-08	MGS	8.08E-14	0.2344	2.130	–	4.188
	1.00E-10	MGS	8.08E-14	0.2344	2.130	–	4.188
	1.00E-12	MGS	8.08E-14	0.2344	2.130	–	4.188
20000	1.00E-08	MGS	2.29E-13	0.5801	1.782	–	3.533
	1.00E-10	MGS	2.29E-13	0.5801	1.782	–	3.533
	1.00E-12	MGS	2.29E-13	0.5801	1.782	–	3.533
40000	1.00E-08	CGS	8.05E-10	2.0653	–	1.117	1.985
	1.00E-10	MGS	6.67E-13	2.3077	NG(8.05E-10)	–	1.776
	1.00E-12	MGS	6.67E-13	2.3077	NG(8.05E-10)	–	1.776
80000	1.00E-08	CGS	3.22E-09	4.0741	–	1.320	1.978
	1.00E-10	MGS	3.22E-12	5.3763	NG(3.22E-09)	–	1.499
	1.00E-12	DGKS	8.00E-14	8.0574	NG(3.22E-09)	NG(3.22E-12)	–

表 3 例題 3（JMAX = 100）における選択処理とその実現精度および実行時間，ならびに他の処理の精度，計算時間倍率の結果
Table 3 Accuracy and execution time of selected algorithm and of other algorithms for Sample 3 data (JMAX = 100).

ベクトル長	ポリシー精度	選択処理	実現精度 ($Ortho$)	実行時間 (sec)	対 CGS	対 MGS	対 DGKS
10000	1.00E-08	MGS	4.61E-14	0.2044	2.446	–	4.817
	1.00E-10	MGS	4.61E-14	0.2044	2.446	–	4.817
	1.00E-12	MGS	4.61E-14	0.2044	2.446	–	4.817
20000	1.00E-08	MGS	7.34E-14	0.5163	2.051	–	4.023
	1.00E-10	MGS	7.34E-14	0.5163	2.051	–	4.023
	1.00E-12	MGS	7.34E-14	0.5163	2.051	–	4.023
40000	1.00E-08	CGS	2.61E-12	2.0995	–	1.064	1.973
	1.00E-10	CGS	2.61E-12	2.0995	–	1.064	1.973
	1.00E-12	MGS	9.72E-14	2.2348	NG(2.61E-12)	–	1.853
80000	1.00E-08	CGS	5.99E-12	4.1279	–	1.271	1.979
	1.00E-10	CGS	5.99E-12	4.1279	–	1.271	1.979
	1.00E-12	MGS	1.30E-13	5.2472	NG(5.99E-12)	–	1.557

記載した．また，ポリシー精度を満たしたが，選択処理よりも時間がかかった場合にはその倍率を記載した．なお，選択された処理には「-」を記載してある．

また，例題 1 と同様に，例題 2 および例題 3 についても，3 通りのポリシー精度に対する提案法の評価を行った．その結果を例題 1 の場合と同様の形式で**表 2**，**表 3** に示す．

表 1 から，例題 1 は CGS がいずれも精度が低く，

どのポリシー精度にも満たないことが分かる．ベクトル長が比較的短い N = 10000，20000 のケースでは MGS がやや良好な精度を保っているものの，ベクトル長が長くなると，DGKS でなければ十分な精度を保てないことが分かる．しかし，N = 10000 かつポリシー精度が低い場合には MGS が有効であり，DGKS に対して 4.3 倍も高速であることが分かる．

例題 2 および例題 3 では，ベクトル長が 10000，20000 の場合において，MGS が高性能かつ高精度な処理であることが確認できた．最も計算時間のかかる DGKS に対して最大 4.8 倍の性能となり，また実現精度も 10^{-14} のオーダであり，十分な計算精度を保持しているといえる．

しかし，例題 2 のベクトル長が 80000 の場合では，ポリシー精度によって処理がそれぞれ異なる結果となり，CGS，MGS，DGKS の 3 種類による徒競走方式から構成した提案方法が有効であることが明らかになった．例題 3 では，いずれの場合も DGKS は必要ではなく，CGS でも直交性の誤差が 10^{-10} 以下など十分な精度を達成するケースもあった．その場合，最も高精度な DGKS の実行時間に比べて最大約 2.0 倍の高速化を達成し，提案方法の有効性を確認できた．

5. 関 連 研 究

GRID コンピューティング向け行列ライブラリに対する 3 つの要件に関連する研究動向について述べる．

第 1 の要件である自動チューニングに関する分野については，国内外で研究がさかんに行われている．国内では，電通大の片桐らがプラットフォームに依存しない可搬性のある自動チューニング・フレームワーク[2)~4)] を提唱しており，また，電通大の今村らが性能の安定化技術[8)] を提案している．これらは，いずれも先進的な取り組みであり，技術的に優れている．海外では，米テネシー大の Dongarra らが SANS（Self-Adapting Numerical Software）[5)] という GRID コンピューティング環境での自己適合型の行列ライブラリチューニング技術および，ATLAS[6)] という PC 上の高品質なライブラリを発表している．

一方，第 2，第 3 の要件である性能保証，およびユーザの計算ポリシーの反映に関わる分野については，自動チューニング技術の発展形として今後検討が進んでいくと考えられる．本研究では特に第 3 の要件である計算ポリシーの反映について検討した．

6. まとめと今後の課題

6.1 ま と め

本研究では，従来から計算精度と計算時間の調整が課題であったグラムシュミットの直交化処理において，与えられた最低計算精度の範囲内で最速計算を実行する方法を提案した．本方法では，古典グラムシュミット（CGS），修正グラムシュミット（MGS），DGKS 型のグラムシュミットの処理を同時に実行し，ユーザが入力した数値計算ポリシーに最も適合する出力につき結果を採用する徒競走方式とした．性質の異なる 3 種類のベクトル群を対象に PC（Pentium4，3.2 GHz，HT）上で上記 3 種類の直交化処理の計算時間および計算精度を評価したところ，以下の結果が得られた．

- ベクトル長が 10000，20000 と比較的短い場合には，従来から多くライブラリに採用されてきた MGS が計算時間，計算精度の両面で有効であることが確認された．
- ベクトル長が 40000，80000 と比較的長い場合には，実行時間では CGS < MGS < DGKS の順で長くなり，CGS が最も良いが，計算誤差は CGS > MGS > DGKS の順で小さくなり，DGKS が最も良いことが確認された．

このうえで，ポリシー精度 ε による数値計算ポリシー「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」を入力とし，CGS，MGS，DGKS の 3 種類による徒競走方式から構成した直交化処理方法を提案した．提案方法を上記のベクトル群に対して評価したところ，

- 従来良好な精度といわれてきた MGS においても，ユーザのニーズが高いと思われる 10^{-10} あるいは 10^{-12} の直交性の誤差を超えてしまうケースがあった．しかし，高精度な DGKS によってこれらの精度は達成することができた，
- 一方，問題によっては，CGS でも直交性の誤差が 10^{-10} 以下など十分な精度を達成するケースもあった．その場合，最も高精度な DGKS の実行時間に比べて最大約 4.8 倍の高速化を達成した，

という結果が得られ，計算精度，および計算速度の両面において，CGS，MGS，DGKS の 3 種類による徒競走方式が有効であることが明らかになった．

6.2 今後の課題

以下に今後の課題を列挙する．

- **並列計算機での数値実験**
  今回の評価実験では PC における 1CPU での評価を対象とした．今後，並列計算機での数値実験

を行い，より高性能な直交化機能を開発する．

- **各ソルバーへの実装**
  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．
- **他の指標を加えたポリシーの検討**
  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．使用するメモリ量など他の指標も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．

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, 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).
2) 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, LNCS 2281, pp.628–639, Springer (2002).
3) 自動ブロック化・通信最適化ライブラリ ABCLIB. http://www.abc-lib.org/
4) 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).
5) Dongarra, J.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 (Summer 2003).
6) ATLAS project. http://www.netlib.org/atlas/index.html
7) 直野 健，今村俊幸：自動チューニング型固有値ソルバについて，情報処理学会研究報告，2002-HPC-91 (SWoPP2002), pp.49–54 (2002).
8) 今村俊幸，直野 健：性能安定化を目指した自動チューニング型固有値ソルバーについて，*SACSIS* (*Symposium on Advanced Computing Systems and Infrastructures*) *2003*, pp.145–152 (2003).
9) 直野 健，今村俊幸，恵木正史：GRID コンピューティング環境における行列ライブラリ向け性能保証方式の検討，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG6 (ACS6), pp.105–112 (2004).
10) 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).
11) 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).
12) Lehoucq, R.: Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration, Ph.D. thesis, Rice University, Houston, TX (1995).
13) ARPACK Homepage. http://www.caam.rice.edu/software/ARPACK/

(平成 16 年 9 月 28 日受付)
(平成 17 年 1 月 17 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．