2007－HPC－111 (30)
2007／8／2

リスタート付ランチョス法における実行時パラメータ自動チューニング方式の提案

櫻井隆雄，直野健，恵木正史，（株）日立製作所中央研究所
猪貝光祥，木立啓之，（株）日立超LSIシステムズ

要旨

行列計算ライブラリは，ユーザからの入力として様々なパラメータを持っている。その中には誤った値を入力すると100倍以上性能が劣化し，かつ実行前に最適値を予測するのが難しいものが存在し，これらを自動的に最適化する仕組みが求められている。この課題に対し，本稿では疎行列固有値解法のリスタート付ランチョス法における射影行列次元数 m を，実行時に自動チューニングする方式を提案した。本方式は演算中に残差履歴から最適な m を探索する。従来方式と本方式を比較した結果，最大で127倍高速化していることを確認した。この性能は手動で最適な m を入力した場合と同等であり，行列計算ライブラリにおける実行時パラメータ自動チューニングの有効性が確認できた。

Proposal on Runtime Parameter Auto Tuning Approach for Restarted Lanczos Method

Takao Sakurai, Ken Naono, Masashi Egi, Central Research Laboratory, Hitachi, Ltd.
Mitsuyosi Igai, Hiroyuki Kidachi, Hitachi ULSI Systems Corp.

Abstract

Matrix libraries have many parameters as inputs by the user. They include problem parameters what are difficult to predict the best values before runtime and the approach of automatically optimizing them during runtime is needed. In this paper, we propose a runtime automatic tuning approach for deciding the size of projection matrix (we demote "m") in Restarted Lanczos Method. This approach searches the best "m" with history of residual value at runtime. Numerical experiments show the proposed approach performs 127 times faster than the conventional method in the best case. The result implies the automatic tuning during runtime is effective to iterative matrix solvers.

1. はじめに

近年，行列計算プログラムのチューニングの自動化，すなわち自動チューニング方式が盛んに研究されている。計算機アーキテクチャの複雑化のため，高性能なプログラムの記述に多大な工数が必要となっているためである。

自動チューニング方式とは，演算を行う計算機環境や演算対象となる入力行列に応じて，ループアンローリングやキャッシュブロッキング，アルゴリズム選択といった性能チューニングをプログラムが自動で行う方式を指す。

既存の自動チューニング方式として，インストール時にループアンローリングやキャッシュブロッキングの最適化を行うATLAS[4][5]，インストール時と実行時にループアンローリングとアルゴリズム選択を行うI-LIB[6][7]，ソースコード上にディレクティブを挿入することで実行時にループアンローリング段数を最適化するABCLibScript[8]などが提案されている。これらの方式により，少ない工数で計算機環境，入力行列に合わせた高性能なプログラムの記述が可能になる。

一方で，行列計算プログラムには入力するパラメータの値がほんの少し変更されただけで性能が大幅に変動するものが存在しており，そのパラメータの決定を支援する機能の実現が求められている。

疎行列固有値解法の一種であるリスタート付Lanczos法もそのようなパラメータを持っている。リスタート付Lanczos法は次元数 N の入力行列を次元数 m(<<N)の三重対角行列に射影し，元の行列の固有値の近似値を得る反復解法である。この射影三重対角行列の次元数 m はユーザの入力するパラメータの1つであり，m が小さい場合は収束までの反復回数が多大になり性能が劣化し，逆に大きい場合も演算量が増加するため性能が劣化する。最適な m の値は入力行列の値によって変わるため，ユーザが演算前に最適値を予測するのは困難である。

そこで本稿では，リスタート付 Lanczos 法に演算を実行しながら自動的に m の値を変化させ，最適な値を探索する実行時パラメータ自動チューニング方式を実現することを目的とする。

2. リスタート付Lanczos法

本研究のチューニング対象となるリスタート付 Lanczos 法[1]は疎行列固有値解法の Lanczos 法を計算機向けに固定の作業領域での演算を実現した計算手法である。この手法はWuらのThick-restart Lanczos法[2]をベースにしており，そのアルゴリズムは図1に示すとおりである。

前章で述べたようにリスタート付 Lanczos 法では次元数

Nの入力行列Aを次元数mの三重対角行列Tに射影し，Ritz値と呼ばれるTの固有値を，Aの固有値とした場合の残差を求める。その残差が所望の値以下の場合は演算を終了し，そうでなければTを元により強くAの要素を反映した新たなTを作成し，残差が所望の値を下回るまでこれを繰り返す。

(1)初期乱数ベクトルq_0を選ぶ。
　$\beta_0 = \|q_0\|$　とする。
(2)m-step分の直交化付Lanczos反復を行う。
　$AQ_m = Q_m T_m + \beta_m q_{m+1} e_m^T$
(3)三重対角行列T_mの固有値$\theta^{(m)}$,固有ベクトル$S^{(m)}$を求める。
(4)残差をチェックする。
　$r_i = (AQ_m - Q_m T_m) S_i^{(m)}$
(5)リスタートベクトルの選択。
　(4)の残差を見て,適当な固有ベクトルを
　$S^{(m)}$からk本選択する。これをYとする。
(6)リスタート行列$\hat{T}_{k+1}$の作成
　①$\hat{T}_k$の対角成分にYに対応するRitz値$\hat{\theta}_k$を入れる。
　②Ritzベクトル$\hat{Q}_k = Q_m Y$を計算する。
　③Y^Tの第m列をgとして$\hat{T}_k$を1行1列拡大し($\hat{T}_{k+1}$),
　　$\hat{T}_{k+1}$の第k+1行（対角除く）に$\beta_m g$を入れる。
　④$\hat{r}_{k+1} = A\hat{q}_{k+1} - (\hat{q}_{k+1}^T \hat{p})\hat{q}_{k+1} - \sum_{j=1}^{k} \beta_m g_j \hat{q}_j$を計算する。
　⑤$\hat{\beta}_{k+1} = \|\hat{r}_{k+1}\|$を計算する。
(7)k+2以降l-step分の直交化付Lanczos反復を行う。
　$A\hat{Q}_i = \hat{Q}_i \hat{T}_i + \hat{\beta}_i \hat{q}_{i+1} e_i^T$,　$i = k+2, \cdots, k+2+(l-1)$
(8)$L = k+2+(l-1)$として，$\hat{T}_L$を三重対角化後，
　固有値$\theta^{(L)}$，固有ベクトル$Z^{(L)}$を求める。
(9)残差をチェックする。（(4)同様）
(10)収束性チェック
　所望の数Eの固有値が所望の残差で得られた→(11)へ
　それ以外→(5)へ
(11)終了処理　$r = Ax - \lambda x$,　$\lambda = \theta^{(L)}$,　$x = Q_L Z^{(L)}$

図1 リスタート付 Lanczos 法のアルゴリズム

このとき，Tの次元数mは，値によって演算で使用する記憶領域が変動するため，ユーザの与えるパラメータとして設定されている。しかし，mは記憶領域だけではなく演算性能にも影響を与える。mが小さい場合，TがAの要素を十分強く反映するまで非常に多い回数のLanczos反復回数を必要とし，演算性能を劣化させる。逆にmが大きい場合，直交化や三重対角化の演算時間が大きくなり，やはり演算性能を劣化させる。従って，最良の性能でリスタート付Lanczos法を実行するためには大きすぎず小さすぎない最適なmの値m_{bh}を設定する必要がある。しかし，このm_{bh}は入力行列により変化するため，事前に予測するのは難しい。

次章でリスタート付Lanczos法の入力行列の違いによりm_{bh}が変動すること，m_{bh}を外れることにより性能が大幅に劣化すること，の2点を実際の演算結果と共に詳しく述べる。

3. リスタート付Lanczos法における最適な射影行列の次元数

3.1 入力行列および演算条件

リスタート付Lanczos法における最適な射影行列の次元数m_{bh}の入力行列による変化を実際に測定した。表1に測定環境，リスタート付Lanczos法のパラメータから成る演算条件，表2に測定に用いた入力行列を示す。表2に示した行列は数値計算アルゴリズムの研究用に線形代数のテストデータを提供している例題サイトMatrixMarket[3]より入手した。

表1 演算条件

CPU	Pentium®D820 2.8GHz,FSB:800MHz,L1:16KB,L2:1MB
主記憶	2GB(DDR2-667)
コンパイラ名	Intel® Fortran Compiler Ver8.0
コンパイルオプション	-nologo, -threads, -c
固有値数 E	降順に 10 個
射影行列次元数 m	21～101 (2 刻みで変化させる) ※m の最小値は 2E+1
リスタート数 k	(m-1)/2
要求精度 EPS	1.0D-08

表2 入力行列一覧

#	名称	次元数	非零要素数	備考
1	H2O	67024	11441880	分子構造
2	Ga41As41H72	268096	9378286	分子構造
3	SiO2	155331	5719417	分子構造
4	SHIPSEC1	140874	3977139	船体構造
5	c-73	169422	724348	非線形最適化
6	cfd2	123440	1605669	圧力分布

3.2 最適な射影行列次元数の測定結果と本研究の課題

図2,図3にGa41As41H72とSHIPSEC1の各mで収束するまでの演算時間を示した．横軸はm，縦軸は演算時間である．また，表3に各行列のm_{bh}と，m_{bh}の演算時間を1としたときのm=21とm=101のときの演算時間の比を示した。

図2，図3を比較すると,図2のGa41AS41H72ではmがE＝10における最小値の21のときは非常に演算時間が長くなっており,m=33までで急速に演算時間が短くなる。m=33からm_{bh}=53までは演算時間はあまり変化せず，それより大きくなると緩やかに演算時間が長くなり，最終的にm=101になるとm_{bh}と比べて演算時間が約2倍になる。一方で，図3のSHIPSEC1の場合は，m=21のときに演算時間はm_{bh}=35と比べ2割増程度の水準であり，mが大きくなると演算時間は増加し，m=101のときはm_{bh}と比べて約2倍となる。

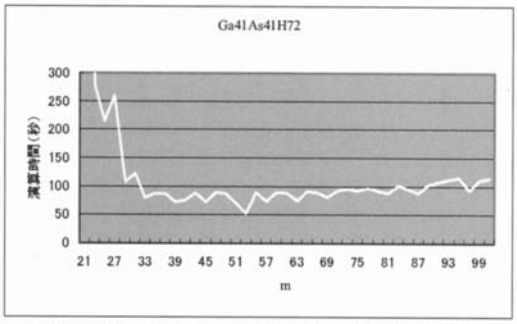

図 2 Ga41As41H72 におけるmと演算時間の関係

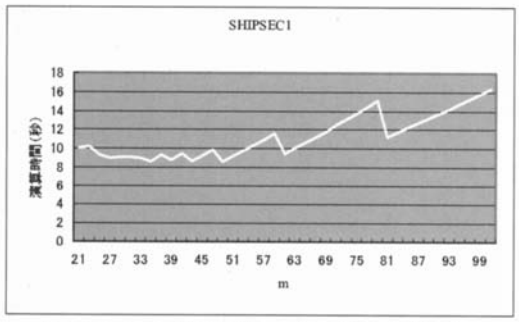

図 3 SHIPSEC1 におけるmと演算時間の関係

表3 各行列のm_{th}と値が離れたときの演算時間比

#	名称	m_{th}	m=21 との比	m=101 との比
1	H2O	29	28.33	2.05
2	Ga41As41H72	53	170.63	2.17
3	SiO2	33	13.64	1.75
4	SHIPSEC1	35	1.17	1.90
5	c-73	25	58.07	1.88
6	cfd2	31	1.34	2.41

表3からは，行列によりm_{th}は大きく変化し，mが小さいときに非常に演算時間が長くなるものとそうでないものが存在すること，mが101のように大きくなるとm_{th}と比べて演算時間が約2倍となることが読み取れる。また，表2に示した各行列の次元数や非零要素数とm_{th}の値やmが小さいときに演算時間が特に長くなるかには相関が無く，入力行列からm_{th}を予測するのは容易ではないとわかった。

以上からリスタート付Lanczos法の射影行列次元数mは次の2つの特徴があることがわかった。1つ目はリスタート付Lanczos法における射影行列の次元数mはユーザの入力するパラメータであるが，入力行列ごとに異なる最適値m_{th}が存在し，値の選択を間違えるとm_{th}で演算を行った場合と比べ，最悪で100倍以上性能が劣化することである。2つ目はm_{th}の値は次元数や非零要素数などの行列の特徴からは判断できず，演算実行前にm_{th}の値を予測するのは難しいことである。

しかし，リスタート付Lanczos法は反復法であるため，実行前にmの妥当性が判断できなくても，実行時に収束判定時に算出される演算の途中経過を用いることで入力されたmが高い性能が得られるか判断するのは可能であると考えられる。

よって，本研究の課題は，リスタート付Lanczos法において，演算を実行しながら最適なmを探索し，高性能な演算を行う実行時自動チューニング機能の実現，とした。

4. 実行時自動チューニング方式の提案

4.1 残差履歴による射影行列次元数の妥当性判断手法

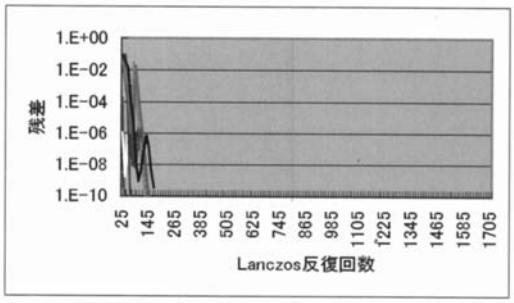

図 4 SHIPSEC1, m=25 の残差履歴

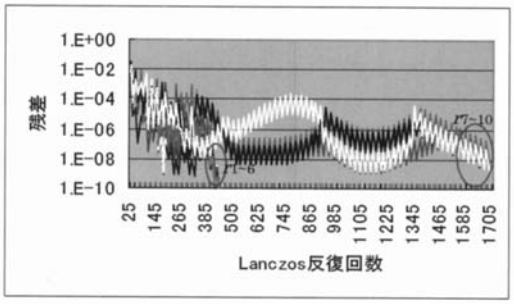

図 5 Ga41As41H72, m=25 の残差履歴

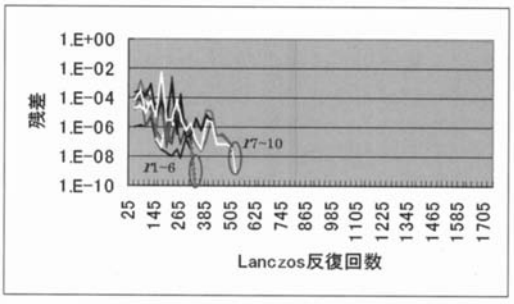

図 6 Ga41As41H72, m=49 の残差履歴

リスタート付Lanczos法では，Tが作られ，Ritz値を求める

際に，E個それぞれのRitz値の残差r_i(i=1～E, i:絶対値の大きい固有値の順序)を調べることで，収束/未収束の判定を行っている。この残差r_iを記録し，収束が早い場合と遅い場合を比較した。図4にSHIPSEC1，m=25，図5にGa41As41H72，m=25，図6にGa41As41H72，m=49のそれぞれの残差の履歴を示した。横軸がLanczos反復回数，縦軸が残差である。

図4と図5の比較により，以下が分かる。図4では全てのr_iがTの求まるたびに1000倍～10000倍のオーダで変動し，180回のLanczos反復回数で収束条件を満たす。一方で，図5では，r_1からr_8は約400回と比較的少ない反復で収束条件を満たすが，それ以外のr_9からr_{10}は，10倍～100倍程度と1回のT作成あたりの変動幅が小さく，収束条件を満たすまでに1500回以上の反復回数を必要とする。しかし，図6に示したように，同じGa41As41H72が入力行列でも，mが十分に大きい49では，図5の約3分の1である500回強の反復で全てのr_iが収束している。

ここで，収束の遅い図5において，r_1からr_8が収束した後のr_9からr_{10}の動きについて，図4や図6と違う特徴的な点がないか考える。これらの残差は収束判定ごとに上下に振動しており，例えばあるi回目の収束判定での残差r_i (s)をs-1回目の収束判定での残差r_i (s-1)で割った値は，収束の早い他の部分のそれと比べて特に大きい，小さい，といった違いがあるわけではない。しかし，収束判定s-t回目(tは2や4)からx回目の残差の中での最大値を最小値で割った値は100未満が多く，他の部分で同様に最大値を最小値で割った値である1000～10000というオーダの値と比べると非常に小さい。

この収束判定s-t回目からs回目の残差の中での最大値を最小値で割った値を「残差Max-Min比」とし，以下では「**MM比**」と表記する。s回目の収束判定時のi番目の残差r_iに対する過去t回分のMM比$R_i(s,t)$を式で表したものが以下である。

$$R_i(s,t)=\frac{\max_z\{r_i(z);z=s-t+1,\cdots,s\}}{\min_z\{r_i(z);z=s-t+1,\cdots,s\}}$$

図7に収束が早い場合と遅い場合について，収束判定回数sとr_{10}に対するMM比の関係を示した。グラフ上の各線は図7～9で示したSHIPSEC1のm=25と，Ga41As41H72のm＝25，49についてであり，r_{10}を例にしたのはLanczos法では固有値の絶対値の大きいr_iから順に収束する傾向を示すからである。収束の早いSHIPSEC 1のm=25とGa41As41H72のm＝49はMM比が100以上で推移するが，収束の遅いGa41As41H72のm＝25は収束判定回数が少ない間のMM比は1000～10000であるが，途中からほぼ100以下のまま推移している。

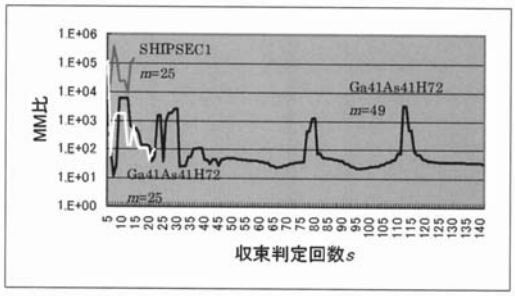

図 7 収束の早さによるMM比の変化

以上から，MM比が大きいときは収束が早い，すなわち収束を早めるためにmを大きくする必要は無く，MM比が小さい場合は収束が遅い，すなわち収束を早めるためにmを大きくする必要があるとわかる。

よって，残差履歴を利用した射影行列の次元数の実行時自動チューニングの戦略として，mを最小値で演算を開始し，何度か収束判定を行った後に未収束のr_i全てのMM比を調べ，小さいものが存在した場合は早く収束させるためにmを大きくする，という方式を提案する。

4.2 残差履歴を利用した実行時パラメータ自動チューニング方式のアルゴリズム

4.1節で示した，収束速度がMM比というパラメータで読み取れるという特徴から，図8に示すリスタート付Lanczos法における射影行列Tの次元数mを自動チューニングするアルゴリズムを考案した。

このアルゴリズムと従来法の違いは以下の2点である。

①リスタート付Lanczos法で使用する記憶領域は射影行列の次元数mが大きくなると増加するため，ユーザは従来通りmを指定する。ただし，本方式ではユーザの入力するmは許容できる最大の次元数，つまり使用可能な記憶領域を表しており，ここではm_{max}と表記する。実際に演算を行う射影行列次元数mは内部パラメータとしてm_{max}とは別に存在する。

②演算開始時にmは最小の値である2E+1に設定される。m=2E+1でTをp回求め，各Tで全ての残差r_xの履歴を記録する。p回終わったあとでmが妥当な大きさかの判断として，所望の残差が得られていないr_xについてp回分のMM比$R_i(p,p)$を求め，求めた$R_i(p,p)$の中に一定の値B以下のものがあった場合，収束が遅い，つまりmは十分な大きさでないとし，mをaだけ大きくし，反復を続ける。この判断はmがユーザの指定したm_{max}に達するか，全ての固有値が収束条件を満たすまで，Tをp回作成するたびに行う。

(1)初期乱数ベクトルq_0を選ぶ。
$\beta_0=\|q_0\|$ とする。
(2)m-step分の直交化付Lanczos反復を行う。
$AQ_m = Q_m T_m + \beta_m q_{m+1} e_m^T$
(3)三重対角行列T_mの固有値$\theta^{(m)}$,固有ベクトル$S^{(m)}$を求める。
(4)残差をチェックする。
$r_i = (AQ_m - Q_m T_m) S_i^{(m)}$
(5)リスタートベクトルの選択。
(4)の残差を見て,適当な固有ベクトルを
$S^{(m)}$からk本選択する。これをYとする。
(6)リスタート行列$\hat{T}_{k+1}$の作成
(7)k+2以降l-step分の直交化付Lanczos反復を行う。
$A\hat{Q}_i = \hat{Q}_i \hat{T}_i + \hat{\beta}_i \hat{q}_{i+1} e_i^T, \quad i = k+2, \cdots, k+2+(l-1)$
(8)$L = k+2+(l-1)$として，$\hat{T}_L$を三重対角化後，
固有値$\theta^{(L)}$，固有ベクトル$Z^{(L)}$を求める。
(9)残差をチェックする。（(4)同様）
(10)収束性チェック
所望の数Eの固有値が所望の残差で得られた→(12)へ
前のmの妥当性判定からp回目のチェック→(11)へ
それ以外→(5)へ
(11)mの妥当性判定
①$m = m_{max}$の場合→(5)へ
②所望の残差に達していない固有値の中で、
過去p回分のMM比がB以下のものが存在する場合
→$m = \min(m+a, m_{max})$
③(5)へ
(12)終了処理 $r = Ax - \lambda x, \quad \lambda = \theta^{(L)}, \quad x = Q_L Z^{(L)}$

図8 実行時自動チューニング方式のアルゴリズム

5. 評価実験

5.1 演算条件と自動チューニングのパラメータ

4章で考案したアルゴリズムの有効性を確認する為に3章の測定と同一の条件で評価実験を行った。表4に提案手法で新たに必要となったパラメータの値を示す。

表4 提案手法で新たに必要になったパラメータ

m_{max}	21～101（2刻みで測定）
a	2
p	5回
B	1.0E+02

5.2 評価結果

図9，図10にGa41As41H72とSHIPSEC1について従来のもの(Non-AT)と提案した自動チューニング方式(AT)の演算時間を比較したグラフを示す。横軸はm(自動チューニングを使った場合m_{max})，縦軸は演算時間である。また，図11に3章の表3で得たm_{th}の演算時間を1としたときのm=21,101及び提案方式で十分な大きさのm_{max}を指定した場合との比較のグラフを示した。

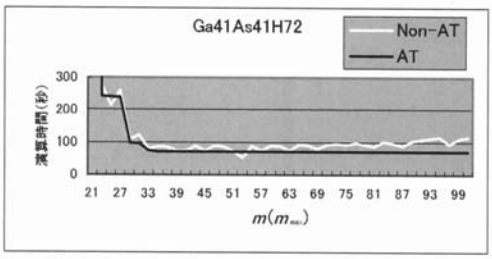

図9 Ga41As41H72における提案法と従来法の比較

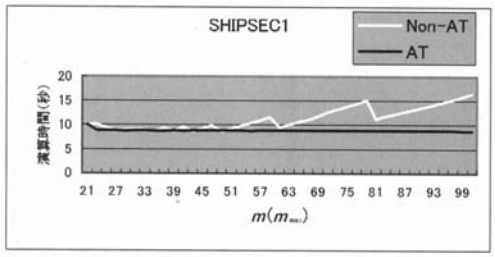

図10 SHIPSEC1における提案法と従来法の比較

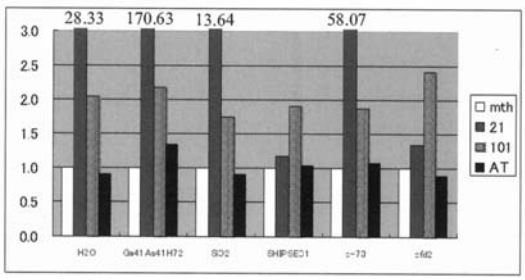

図11 m_{th}とm=21,101及び提案方式(AT)の演算時間の比

図9，10，11から以下の事柄が読み取れる。

従来法であるリスタート付Lanczos法はmが最適値m_{th}と比べ，大きくても小さくても性能が劣化するため，値の選択が難しかった。しかし，図9,10に示したように，提案手法で新たな入力パラメータであるm_{max}を大きくすることは，自動チューニングにおける探索範囲を広めるだけで，最適な次元数がm_{max}より小さければソルバはその最適な次元数を自動で選択して演算を行うため，m_{max}を大きくすることによる性能劣化は起こらないことが確認できた。一方で，提案手法でも指定されたm_{max}が小さい場合，探索できる範囲が狭いため，従来法同様に非常に性能が悪くなる。このことから，提案方式を用いる場合は可能な限りm_{max}を大きくすることで高い性能が得られると分かった。

図11から，従来法と比較した場合の具体的な高速化倍率が読み取れる。従来法でm=21,101を入力した場合と，提案方式で十分な大きさのm_{max}を入力した場合を比較し

たとき，6つの行列の平均でm=21に対しては38倍，m=101に対しては2.0倍高速化していた。特にGa41As41H72でm=21を選択した場合と比べたとき，提案方式は127倍高速化していた。また，従来法でm_{th}を入力した場合と提案方式を比べたとき，SHIPSEC1やc-73，Ga41As41H72では提案方式が2.3%，7.6%，34%遅く収束し，H2O，SiO2，cfd2では提案方式が9.7%，9.6%，12%速く収束した。結果は3勝3敗であり，提案方式により，従来法でm_{th}を入力して演算した場合と同等の演算性能が得られると言える。

Ga41As41H72では，図9で示すように次元数53が周囲の値より50%早く収束する特異点となっていた。しかし，提案手法は次元数37で十分に収束が早いと判断するため，それより大きい次元数を探索しなかった。このことが従来法のm_{th}と比較してGa41As41H72の性能劣化が顕著だった理由と考えられる。

以上から，提案方式では，できる限りm_{max}を大きく指定することで，m_{th}で演算した場合と同等の高い性能が得られることが分かった。

6. まとめ

本報告では行列計算ライブラリの機能の1つであるリスタート付Lanczos法において，値が小さい場合は収束が遅く，逆に値が大きい場合は演算量が増加するため，値の選択により性能が大幅に劣化するパラメータである射影行列次元数mの自動チューニング方式を提案した。提案方式は過去一定回数の最大—最小残差の比である「MM比」を判断指標として演算中の射影行列次元数の大きさが妥当であるか判断し，自動で最適な次元数に近づくよう次元数を最適化しながら演算を行うものである。本方式の適用によりユーザは使用できるメモリ量から許容できるm_{max}を指定すれば，あとはソルバが自動で最適な次元数を探索することにより，最適な次元数をあらかじめ指定して演算を行った場合と同等の性能が得られるようになる。

本方式を適用した結果，ユーザの入力したm_{max}が最適な値であるm_{th}と比べて十分大きい場合でも，内部では最適な次元数で演算が行われるため，性能の劣化が起こらなかった。従来のリスタート付Lanczos法で射影行列次元数をm_{th}として最初から最後まで演算した場合と比べても，最低で25%の性能劣化，最高で12%の性能向上であり，また，今回の測定の範囲ではパラメータ選択に失敗し最も性能が劣化する場合と比較したときと比べると127倍高速化しているとわかった。

今後の課題としては以下が考えられる。

今回の実験で用いた1回の判断で大きくする次元数a，次元数妥当性の判断頻度pなど自動チューニング方式のパラメータが最適なものだという検討が不十分であり，特に，求める固有値の数Eが10個の場合以外でも十分な性能が得られるかは不明である。今後，Eが10個以外の実験等を行い現状のパラメータの妥当性を評価する必要がある。

Ga41As41H72において次元数53は特別に収束の早い値であるが，提案方式では次元数が37で十分に収束が早いと判断するため，それ以上次元数を大きくしない。このような特別な値を探索の対象とすべきか，対象とするのならばどのように探索するか，について検討する。

上記の解決後，本稿で提案した技術を応用し，反復法のさまざまなパラメータを実行中に自動チューニングする手法について検討を進める。

参考文献

[1] 猪貝 光祥，直野 健，木立 啓之，“大規模疎行列固有値計算における行列ベクトル積の並列性能向上方法，” 日本応用数理学会論文誌 Vol.15, No.2, 2005, pp.117～128, (2005).

[2] Kesheng Wu; “Thick-restart Lanczos Method for Large Symmetric Eigenvalue Problems,” 2000 SIAM J. MATRIX ANAL. APPL, Vol.22, No.2, pp.602-616, (2000).

[3] MatrixMarket; “http://math.nist.gov/MatrixMarket/.”

[4] R. Whaley, A. Petitet and J. Dongarra, “Automated empirical optimizations of software and the ATLAS project,” Parallel Computing, 27, pp.3-35 (2001).

[5] ATLAS project, “http://www.netlib.org/atlas /index.html.”

[6] H. Kuroda, T. Katagiri, and Y. Kanada, “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 Springer 2002, pp.628-639 (2002).

[7] Project I-LIB, “http://www.super-computing.org/~kuroda/nadia.html.”

[8] ABC-LIB Homepage, “http://www.abc-lib.org/.”