自動チューニング機能付き疎行列反復解法ソルバーXabclib

の公開について

片桐孝洋、黒田久泰（愛媛大学、兼務)、中島研吾
東京大学情報基盤センター

１．はじめに

1.1　概要

（１）Xabclib とは

本センターでは､自動チューニング機能付き疎行列反復解法ソルバーXabclib(eXtended ABCLib)の試験公開を開始いたします。Xabclib は、以下の 3 機能を提供する数値計算ライブラリです。

- エンドユーザ向け機能
  1. 標準固有値問題 LANCZOS 法ライブラリ
     Xabclib_LANCZOS
  2. 連立一次方程式 GMRES 法ライブラリ
     Xabclib_GMRES
- 数値計算ライブラリ開発者向け機能
  3. 汎用自動チューニング機能インターフェースライブラリ
     OpenATLib

本試験提供版はα版です。α版は、OpenMP でスレッド並列化がされており、1 ノード内での並列実行が可能です。分散メモリ版（MPI 並列版）は現在未対応です。本センターに設置された T2K オープンスパコン（東大版）の HITACHI HA8000 クラスタシステムでは、1 ノードは 16 コアありますので、16 並列実行まで可能な並列数値計算ライブラリです。

本稿は、エンドユーザ向け機能の説明を行うものです。α版では、予告しないインターフェースの変更、機能向上があります点、あらかじめご了承ください。今後の開発方針として、OpenATLib における機能の高度化（自動チューニングポリシー機能の開発）を予定しております。

（２）OpenATLib とは

OpenATLib は任意の行列計算ライブラリに様々なパラメータの最適値を推定する機能や行列-ベクトル積の最適な実行方式を判定する機能などの自動チューニング機能を提供するインターフェース（Application Programming Interface, API）です。OpenATLib は自動チューニング機能を備えた副プログラムであるオブジェクトファイルとそれらの副プログラムで使用する幾つかのパラメータをブロックデータとして宣言するインクルードファイルから構成されます。

1.2　対象言語

OpenATLib は Fortran90 で記述されたプログラムから呼び出されることを前提としております。

1.3　ライブラリが提供する自動チューニング機能

OpenATLib で実現される自動チューニング機能は以下のものです。

- 実行時自動チューニング機能
  実行時自動チューニング機能はプログラム中で実際にライブラリが呼ばれた際に決定されるパラメータを自動チューニングします。

２．共通自動チューニングインターフェース OpenATLib

2.1　OpenATLib の機能および使用方法

本節では共通自動チューニングインターフェース OpenATLib の機能および仕様について説明します。

（１）機能

OpenATLib が提供する自動チューニング機能を以下の表 2-1 に示しました。

表 2-1　OpenATLib が提供する自動チューニング機能

機能名称	機能内容
OpenATI_DAFRT	Krylov 部分空間のリスタート周期を大きくするべきかを判定する
OpenATI_DSRMV	CRS (Compressed Row Storage)形式で格納された倍精度対称疎行列-ベクトル積の最適な実行方式を判定する
OpenATI_DURMV	CRS 形式で格納された倍精度非対称疎行列-ベクトル積の最適な実行方式を判定する
OpenATI_BLDATA	自動チューニングパラメータデフォルト値設定（Fortran の Block data 文）

（２）インクルードファイル“OpenAT.inc”

OpenAT.inc をインクルードしたプログラム内では，以下の変数を宣言なしで参照・更新できます。更新結果は以降の OpenATI 関数でパラメータとして設定されます。各々のパラメータ値については，各機能の仕様を参照してください。

(a) OpenATI_DAFRT_IPARM_1
　　M/M 比による自動チューニングを実施するかどうかのフラグ。

(b) OpenATI_DAFRT_RPARM_1
　　MM 比の閾値

(c) OpenATI_DSRMV_IPARM_1
　　対称の行列ベクトル積のアルゴリズム検索範囲のパラメータ。

(d) OpenATI_DURMV_IPARM_1
　　非対称の行列ベクトル積のアルゴリズム検索範囲のパラメータ。

(e) OpenATI_DURMV_IPARM_2
　　非対称の行列ベクトル積の性能評価における反復回数。

（３）使用方法

自作ライブラリで OpenATLib を使用する場合，以下の手順を実行してください。

①OpenAT のインクルードファイル“OpenAT.inc”と OpenATLib のオブジェクト“libOpenAT.a”を任意の場所に置く。

②自作ライブラリのプログラムソース内で図 2-1 に示すように“OpenAT.inc”をインクルードする宣言を行なう。

③自作ライブラリのプログラムソース内で OpenATLib の機能をコールする。

④make ファイル中で“libOpenAT.a”をリンクする。

```
INCLUDE “OpenAT.inc”
```

図 2-1　OpenATLib 使用の際の宣言例

2.2 OpenATI_ DAFRT

2.2.1 機能概要

疎行列の固有値問題解法の Lanczos 法や連立 1 次方程式解法の GMRES 法などの Krylov 部分空間法は計算機上で実行する場合，使用するメモリを実行前に確定させるため Krylov 部分空間の次元数を一定数に制限します。それより大きくなった場合は得られた最新の近似解を新たな初期値としてリスタートさせて新たな Krylov 部分空間を生成します。この Krylov 部分空間の最大次元数をリスタート周期と呼びます。

リスタート周期は小さすぎると近似解の真の解との残差の減少が停滞し，収束するまでに非常に多くの反復を要します。また，逆に大きすぎる場合は Krylov 部分空間の生成に多大な演算を要するため，大きすぎても小さすぎても演算時間が大きく増加します。一方，最適なリスタート周期は入力行列や条件によって大きく異なり，実行前の最適な値の推定は困難です。そこで，リスタート周期の推定を実行中に行なう機能が求められています。

OpenATI_ DAFRT は Krylov 部分空間のリスタート周期を現在の値からより大きな値とするべきか判定するための機能です。

2.2.2 自動チューニング方式の概要

最適なリスタート周期の推定は実行前には困難ですが，実行中であれば残差の履歴を監視し，残差の停滞を検出することで推定を可能とする方式が文献[1]で提案されています。

残差の停滞を示す指標として収束判定*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**」と表記します。*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*)を以下に示します。

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

リスタート周期が十分に大きいときは残差が大きく減少するため MM 比は大きくなり，リスタート周期が小さいときは残差が停滞するため MM 比は小さくなります。

そこで，リスタート周期の大きさを自動的に最適化する場合は MM 比を監視し，MM 比が一定値を下回ったときにリスタート周期を大きくします。

2.2.3 引数の内容とエラーコード

（1）引数の内容

引数	型	種別	引数の説明
NSAMP	Integer	INPUT	サンプリング値の個数
SAMP (NSAMP)	Double	INPUT	サンプリング値
IRT	Integer	OUTPUT	0：リスタート周期を大きくする必要は無い。 1：リスタート周期を大きくした方がよい。
INFO	Integer	OUTPUT	エラーコード

（2）”OpenAT.inc”上で設定されているパラメータ

変数名	型	初期値	変数の説明
OpenATI_DAFRT_IPARM_1	Integer	1	1：MM 比の大きさによりリスタート周

			期を大きくすべきか判定する。
OpenATI_DAFRT_RPARM_1	Double	100.0	判定の閾値。

（3）エラーコード

値	内容
0	正常終了したことを表す。

2.2.4 使用例

5 反復に 1 回リスタート周期の大きさを判定し，大きくする必要があるときは 1 ずつ大きくするようにする場合，図 2-2 のように記述します。

```
//パラメータ宣言部
INCLUDE “OpenAT.inc”  // OpenAT.inc のインクルード
MSIZE=1                    //最初のリスタート周期
I=5                        //判定の頻度
                                  ～中略～
IF RSDID < TOL    RETURN   //収束判定

SAMP (K)=RSDID      //SAMP(K)に残差を入力

IF K = I THEN      //I 回に 1 回 DAFRT を CALL
        IRT=0
        CALL    OpenATI_ DAFRT (I, SAMP,IRT,INFO)

        IF IRT= 1    MSIZE=MSIZE+1    //リスタート周期を大きく
        K=0
END IF

K=K+1
                                  ～以下略～
```

図 2-2　OpenATI_ DAFRT の記述例

2.3 OpenATI_DSRMV，OpenATI_DURMV

2.3.1 機能概要

疎行列-ベクトル積は反復解法において多くの回数が実行され, その演算時間の合計は解法の中で大きな割合を占めます。疎行列-ベクトル積のプログラム実装方式は様々なものが存在しますが，実行する環境や行列の特性などにより適切な実装方式は変化するため，一概にどの実装方式が最適とは言えません。そのため，実行時に環境や行列に応じて最適な実装方式を選択する方式が求められています。

OpenATI_DSRMV は倍精度型実対称疎行列-ベクトル積，OpenATI_DURMV は倍精度型実非対称疎行列-ベクトル積の最適な実行方式を判定し実行する機能です。

2.3.2 自動チューニング方式の概要

本機能では文献[2]で提案されている解法開始直後の最初の疎行列-ベクトル積の実行時に複数の実装方式を順番に実行し, 処理時間の最も短かった方式で以後の疎行列-ベクトル積を実行することで最適な

実装方式の選択を実現しています。

以下に示すように OpenATI_DSRMV，OpenATI_DURMV のそれぞれで 3 種の実装方式を用意しています。

- OpenATI_DSRMV
  1) 内積ループ並列，外積ループ逐次方式
  2) キャッシュ向けループ融合，逐次実行方式
  3) キャッシュ向けループ融合，リダクション並列（動的作業領域）方式。コアごとに別の作業領域を参照する。よって（共有メモリ並列数)×(ベクトルの次元数）の作業領域を要する。

- OpenATI_DURMV
  1) ループがコンパイラによりベクトル化されるよう記述した方式
  2) 2 重ループが 8-2 アンローリングになるよう明示的にディレクティブを挿入した方式
  3) ループがコンパイラによりベクトル化されないよう記述した方式

2.3.3 OpenATI_DSRMV の引数の内容とエラーコード

（1） 引数の内容

引数	型	種別	引数の説明
N	Integer	INPUT	係数行列の次元数（N≧1）
NNZ	Integer	INPUT	係数行列の非零要素の個数
IRP(N+1)	Integer	INPUT	係数行列の各行の対角要素へのポインタ。
ICOL(NNZ)	Integer	INPUT	係数行列の非零要素の列番号を圧縮型 1 次元配列に入れる。
VAL(NNZ)	Double	INPUT	係数行列の非零要素を圧縮型 1 次元配列に入れる。
X(N)	Double	INPUT	ベクトルの要素を 1 次元配列に入れる。
Y(N)	Double	OUTPUT	行列ベクトル積の演算結果のベクトルが入る。
ICASE	Integer	INPUT/ OUTPUT	OpenATI_DSRMV_IPARM_1=1 のとき，実行方式を示す番号を入力する。 OpenATI_DSRMV_IPARM_1=2，3 のとき，最適と判断された実行方式を示す番号が出力される。 実行方式を示す番号は以下の通り。 11：内積ループ並列，外積ループ逐次方式 12：キャッシュ向けループ融合，逐次実行方式 13：キャッシュ向けループ融合，リダクション並列（動的作業領域）方式。共有メモリ並列数に応じた作業領域を要することに注意。
NUM_SMP	Integer	INPUT	OpenATI_DSRMV_IPARM_1=1 かつ ICASE=13，もしくは OpenATI_DSRMV_IPARM_1=3 が選択されたとき，共有メモリ並列数を入れる。
WK(N,NUM_SMP)	Double	WORK	OpenATI_DSRMV_IPARM_1=1 かつ ICASE=13，もしくは OpenATI_DSRMV_IPARM_1=3 が選択されたとき，作業領域を入れる。
INFO	Integer	OUTPUT	エラーコード

（2）”OpenAT.inc”上で設定されているパラメータ

変数名	型	初期値	変数の説明
OpenATI_DSRMV_IPARM_1	Integer	1	1:ICASE で指定された方式で行列-ベクトル積を実行。 2：リダクション並列方式を除いた 2 方式の中で最適な実行方式を判定する。 3:3 方式の中で最適な実行方式を判定する。リダクション並列数に応じた作業領域を要することに注意。

（3）エラーコード

値	内容
0	正常終了したことを表す。
100	ICASE の値が不正であることを表す。 （OpenATI_DSRMV_IPARM_1=1 の場合のみ）
200	OpenATI_DSRMV_IPARM_1 の値が不正であることを表す。

2.3.4　OpenATI_DURMV の引数の内容とエラーコード

（1）引数の内容

引数	型	種別	引数の説明
N	Integer	INPUT	係数行列の次元数（N≧1）
NNZ	Integer	INPUT	係数行列の非零要素の個数
IRP(N+1)	Integer	INPUT	係数行列の各行の先頭要素へのポインタ。
ICOL(NNZ)	Integer	INPUT	係数行列の非零要素の列番号を圧縮型 1 次元配列に入れる。
VAL(NNZ)	Double	INPUT	係数行列の非零要素を圧縮型 1 次元配列に入れる。
X(N)	Double	INPUT	ベクトルの要素を 1 次元配列に入れる。
Y(N)	Double	OUTPUT	行列ベクトル積の演算結果のベクトルが入る。
ICASE	Integer	INPUT/ OUTPUT	OpenATI_DURMV_IPARM_1=1 のとき，実行方式を示す番号を入力する。 OpenATI_DURMV_IPARM_1=2，3 のとき，最適と判断された実行方式を示す番号が出力される。 実行方式を示す番号は以下の通り。 11：ループをベクトル化して実行 12：2 重ループを 8-2 アンローリングで実行 13：ループがベクトル化しないで実行
INFO	Integer	OUTPUT	エラーコード

（2）”OpenAT.inc”上で設定されているパラメータ

変数名	型	初期値	変数の説明
OpenATI_DURMV_IPARM_1	Integer	1	1:ICASE で指定された方式で行列-ベク

			トル積を実行。 2,3 : 3 方式の中で最適な実行方式を判定する。
OpenATI_DURMV_IPARM_2	Integer	1	非対称の行列ベクトル積の性能評価における反復回数

（3）エラーコード

値	内容
0	正常終了したことを表す。
100	ICASE の値が不正であることを表す。 （OpenATI_DURMV_IPARM_1=1 の場合のみ）
200	OpenATI_DURMV_IPARM_1 の値が不正であることを表す。

2.3.5　使用例

解法の最初の行列ベクトル積で最適アルゴリズムを探索し，以後の行列ベクトル積ではその結果を元に最適な行列ベクトル積を用いて演算する場合，図 2-3 のように記述します。

```
//パラメータ宣言部
INCLUDE “OpenAT.inc”            // OpenAT.inc のインクルード
OpenATI_DSRMV_IPARM_1=3   //DSRMVのパラメータ
ICASE=0                           // DSRMV のパラメータ

                                  ～中略～

//最初の行列ベクトル積
CALL    OpenATI_ DSRMV (N,NNZ,IRP,ICOL,VAL,X,Y,ICASE,
                                  NUM_SMP,WK,INFO)
OpenATI_DSRMV_IPARM_1=1      //以後は判定された実行方式を用いる

                                  ～中略～

//以後の行列ベクトル積　出力された結果をそのまま入力すればよい
CALL    OpenATI_ DSRMV (N,NNZ,IRP,ICOL,VAL,VEC,JPARM,
                                  IPARM,RPARM,INFO)

                                  ～以下略～
```

図 2-3　OpenATI_DSRMV の記述例

３．自動チューニング機能付き数値計算ライブラリ Xabclib

本章では、OpenATLib を使用した反復解法ライブラリである Xabclib_LANCZOS と Xabclib_GMRES について説明します。

3.1　Xabclib_LANCZOS

3.1.1　機能概要

Xabclib_LANCZOS は大規模な対称疎行列の固有値の中から値（もしくは絶対値）の大きい数個～数十個を求める機能です。

3.1.2　対象とする問題とデータ構造

（１）対象とする問題

本機能の対象とする問題は大規模疎行列の固有値・固有ベクトルを求める標準固有値問題 $Av=\lambda v$ です。（ここで A：大規模疎行列，λ：固有値，v：固有ベクトルを表す）

（２）入力行列のデータ構造

入力となる対称疎行列Aのデータ構造は図3-1に示すような行圧縮方式(Compressed Row Storage)です。

$$\begin{pmatrix} 1 & 2 & 0 & 3 & 0 \\ & 4 & 0 & 0 & 5 \\ & & 6 & 0 & 0 \\ & & & 7 & 8 \\ & & & & 9 \end{pmatrix} =$$

対角要素のポインタ

1	4	6	7	9	10

非零要素の列番号

1	2	4	2	5	3	4	5	5

非零要素の値

1	2	3	4	5	6	7	8	9

図 3-1　行圧縮方式

3.1.3　Lanczos 法のアルゴリズム

本ライブラリで提供する Lanczos 法のアルゴリズムを図 3-2 に示しました。このアルゴリズムは文献[3]で示されたものを元としています。

1. $Start\ with\ v_0 \equiv r, \beta_0 := \|r\|_2, lock = 0$

2. $For\ IR = 1,2,\cdots, maxrestart\ Do:$

3. $\quad For\ j = lock+1,\cdots, m\ Do:$

4. $\quad\quad Compute\ v_j := r/\beta_0$

5. $\quad\quad r{:=}Av_j$

6. $\quad\quad \alpha_j := (r, v_j)$

7. $\quad\quad if\ (j=1)\ then \quad r{:=}r - \alpha_j v_j$

8. $\quad\quad if\ (j \neq 1)\ then \quad r{:=}r - \alpha_j v_j - \beta_{j-1} v_{j-1}$

9. $\quad\quad r \perp V_{j-1}\ \ by\ modified\ Gram\text{-}Schmidt$

10. $\quad\quad \beta_j := \|r\|_2$

11. $\quad EndDo$

12. $Eigen\ solve\ T = S\Theta S^T\ ,\ T = \begin{bmatrix} \alpha_{lock+1} & & & \\ \beta_{lock+1} & \alpha_{lock+2} & & \\ & \cdots & \cdots & \\ & & \cdots & \cdots \\ & & & \beta_{m-1} \quad \alpha_m \end{bmatrix}$

13. $k\text{-}th\ residual\ estimate\ with\ |\beta_m S_{m,k}| / |\Theta_k|\ for\ k = lock+1, NEV$

14. $creat\ Ritz\ vectors\ \mathrm{Q_k} = V_m S_k$

15. $count-up\ \ 'new\ locked'\ Ritz\ pair$

16. $if\ (lock + 'new\ lock' \geq NEV)\ \ goto\ exit$

17. $create\ new\ starting\ Shur\ vector\ r = V_m S_{'new\ locked'+1}$

18. $deflation\ V_{lock+L} = Q_{lock+L}\ \ for\ L = 1, 'new\ lock', then\ lock = 'new\ lock'$

19. $EndDo$

図 3-2　Lanczos 法のアルゴリズム

3.1.4　引数の内容とエラーコード

（1）引数の内容

引数	型	種別	引数の説明
N	Integer	INPUT	係数行列の次元数（N≧1）
NNZ	Integer	INPUT	係数行列の上三角部分の非零要素の個数
IRP(N+1)	Integer	INPUT	係数行列の各行の対角要素へのポインタ。 注)IRP(1)=1, IRP(N+1)=NNZ+1 となるようにすること。
ICOL(NNZ)	Integer	INPUT	係数行列の上三角部分の非零要素の列番号を圧縮型１次元配列に入れる。
VAL(NNZ)	Double	INPUT	係数行列の上三角部分の非零要素を圧縮型１次元配列に入れる。
NEV	Integer	INPUT	求めたい固有値の個数。演算時間は NEV の大きさに応じて増大する。特に NEV>100 となった場合、現実的な演算時間で解けなくなることが多い。
EV(NEV)	Double	OUTPUT	K 番目に求められた固有値が EV(k)に入る。
EVEC (LDE,NEV)	Double	OUTPUT	固有値 EV(k)に対応する固有ベクトルが第 k 列に入る。
LDE	Integer	INPUT	EVEC の整合寸法(LDE≧N)

MSIZE	Integer	INPUT	リスタート周期。 MSIZE＞NEV とすること。
IPARM(10)	Integer	INPUT/ OUTPUT	Lanczos 法のオプションパラメータ（整数型） ・IPARM(1) 1：値の大きい固有値の方から固有値・固有ベクトルを求める。 2：絶対値の大きい固有値の方から固有値・固有ベクトルを求める。 ・IPARM(2) Lanczos 法の最大リスタート回数を入れる。 ・IPARM(3) 実際のリスタート回数が出力される。 ・IPARM(4)～IPARM(10) 値入力の必要なし。
RPARM(10)	Double	INPUT	Lanczos 法のオプションパラメータ（浮動小数点型） ・RPARM(1) 固有値・固有ベクトルの残差の収束判定基準値を入力する。本機能での収束判定の式を以下に示す。 $\frac{\|Ax-\lambda x\|}{\|\lambda\|}$ ・RPARM(2) 許容する演算時間の最大値(sec)を入れる。 ・RPARM(3) リスタート周期の大きさを判定するための MM 比の閾値を入力する。 （OpenATI_DAFRT の OpenATI_DAFRT_RPARM_1） ・RPARM(4)～RPARM(10) 値入力の必要なし。
IAT(10)	Integer	INPUT	自動チューニングのオプションパラメータ ・IAT(1)＝1 のとき，自動チューニング方式により最適なリスタート周期を求めて計算する。 ・IAT(2) 1：自動チューニング方式により最適な行列-ベクトル積演算方式を決定して計算する。 2：自動チューニング方式により残りのメモリ量を考慮した上で最適な行列-ベクトル積演算方式を決定して計算する。 ・IAT(3)～IAT(10) 値入力の必要なし。
WK (LWK)	Double	WORK	入力する必要なし。（作業領域）
LWK	Integer	INPUT	実数型作業領域配列 WORK のサイズ。 LWK >= (1+MSIZE)*N + 2*MSIZE*MSIZE + 7*MSIZE

			+ 5*NEV +2
IWK (LIWK)	Integer	WORK	入力する必要なし。（作業領域）
LIWK	Integer	INPUT	整数型作業領域配列 IWORK のサイズ。 LIWK >= 5*MSIZE + 3
INFO	Integer	OUTPUT	エラーコード

（2）エラーコード

値	内容
0	正常終了したことを表す。
0 未満	-i が返された場合，i 番目の引数の入力値が不正な値であることを表す。
100	零ベクトルが生成されブレイクダウンが発生したため演算途中で終了したことを表す。
200	三重対角行列の固有値が正常に求められなかったために演算途中で終了したことを表す。
300	最大リスタート回数を超過したため演算途中で終了したことを表す。
400	許容する演算時間を超過したため演算途中で終了したことを表す。

3.2 Xabclib_GMRES

3.2.1 機能概要

Xabclib_GMRES は大規模な非対称疎行列の連立一次方程式を解く機能です。

3.2.2 対象とする問題とデータ構造

（1）対象とする問題

本機能の対象とする問題は連立一次方程式問題 Ax=b です。

（ここで A：大規模疎行列，x：解ベクトル，b：右辺ベクトルを表す）

（2）入力行列のデータ構造

入力となる非対称疎行列 A のデータ構造は行圧縮方式（Compressed Row Storage）です。

3.2.3 アルゴリズムの概要

本ライブラリで提供する GMRES 法のアルゴリズムを図 3-4 に示しました。このアルゴリズムは文献[4]で示されたものです。

1. $Compute\ r_0 = b - Ax_0, \beta := \|r_0\|_2, and\ v_1 := r_0 / \beta$
2. $Define\ the\ (m+1) \times m\ matrix\ \overline{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}, Set\ \overline{H}_m = 0$
3. $For\ j = 1, 2, \cdots, m\ Do:$
4. $\quad Compute\ \omega_j := Av_j$
5. $\quad For\ i = 1, \cdots, j\ Do:$
6. $\quad\quad h_{ij} := (\omega_j, v_j)$
7. $\quad\quad \omega_j := \omega_j - h_{ij} v_j$
8. $\quad EndDo$
9. $\quad h_{j+1,j} = \|\omega_j\|_2.\ If\ h_{j+1,j} = 0\ Set\ m := j\ and\ go\ to\ 12$
10. $\quad v_{j+1} = \omega_j / h_{j+1,j}$
11. $EndDo$
12. $Compute\ y_m\ the\ minimizer\ of\ \|\beta e_1 - \overline{H}_m y\|_2\ and\ x_m = x_0 + V_m y_m.$

図 3-4　GMRES 法のアルゴリズム

3.2.4　引数の内容とエラーコード

（1）引数の内容

引数	型	種別	引数の説明
N	Integer	INPUT	係数行列の次元数（N≧1）
NNZ	Integer	INPUT	係数行列の非零要素の個数
IRP(N+1)	Integer	INPUT	係数行列の各行の先頭要素へのポインタ。 注)IRP(1)=1, IRP(N+1)=NNZ+1 となるようにすること。
ICOL(NNZ)	Integer	INPUT	係数行列の非零要素の行番号を圧縮型 1 次元配列に入れる。
VAL(NNZ)	Double	INPUT	係数行列の非零要素を圧縮型 1 次元配列に入れる。
B(N)	Double	INPUT	右辺ベクトル b の要素を入れる。
X(N)	Double	INPUT/ OUTPUT	(INPUT) 初期解ベクトル x_0 の要素を入れる。 (OUTPUT) 求められた解ベクトル x の要素が入る。
KIND_PRE COND	Integer	INPUT	前処理の種類を指定する。 0：前処理無し 1：Jacobi 2：SSOR 3：ILU(0)
PRECOND (NPRE)	Double	INPUT/ OUTPUT	(INPUT) ・IPCPARM(1)=0 のとき 何も入れなくてよい。 ・IPCPARM(1)=1 のとき 既に定義されている前処理行列 M の情報を入れる。 (OUTPUT) ・IPCPARM(1)=0 のとき 前処理行列 M の情報が入る。 ・IPCPARM(1)=1 のとき

			変更無し。
NPRE	Integer	INPUT	配列 PRECOND のサイズ。 KIND_PRECOND が 1 のとき NPRE≧0 KIND_PRECOND が 2，3 のとき NPRE≧N
IPCPARM (10)	Integer	INPUT	前処理行列 M 作成時のオプションパラメータ（整数型） ・IPCPARM(1) 0：前処理行列 M を求める。 1：入力された前処理行列 M を使用する ・IPCPARM(2)～IPCPARM(10) 値入力の必要なし。
RPCPARM (10)	Double	INPUT	前処理行列 M 作成時のオプションパラメータ（浮動小数点型） ・RPCPARM(1) KIND_PRECOND=2 のとき、 SSOR 前処理のパラメータ ω を入れる。 KIND_PRECOND=3 のとき、 ILU(0)前処理行列作成時に breakdown したか否かを判定する閾値を入れる。 ・RPCPARM(2)～RPCPARM(10) 値入力の必要なし。
MSIZE	Integer	INPUT	リスタート周期。
IGRPARM (10)	Integer	INPUT/ OUTPUT	GMRES 法のオプションパラメータ（整数型） ・IGRPARM(1) GMRES 法の最大リスタート回数を入れる。 ・IGRPARM(2) 実際のリスタート回数が出力される。 ・IGRPARM(3)～IGRPARM(10) 値入力の必要なし。
RGRPARM (10)	Double	INPUT	GMRES 法のオプションパラメータ（浮動小数点型） ・RGRPARM(1) 解の収束判定基準値を入力する。本機能での収束判定式を以下に示す。 $\frac{\left\|M^{-1}(b-Ax)\right\|}{\left\|M^{-1}b\right\|}$ ・RGRPARM(2) 計算時間の最大値(sec)を入れる。 ・RGRPARM(3) リスタート周期の大きさを判定するための MM 比の閾値を入力する。 （OpenATI_DAFRT の OpenATI_DAFRT_RPARM_1） ・RGRPARM(4)～RGRPARM(10) 値入力の必要なし。
IAT(10)	Integer	INPUT	自動チューニングのオプションパラメータ

			・IAT(1)＝1 のとき，自動チューニング方式により最適なリスタート周期を求めて計算する。 ・IAT(2)＝１のとき，自動チューニング方式により最適な行列-ベクトル積を決定して計算する。 ・IAT(3)～IAT(10) 値入力の必要なし。
WK (LWK)	Double	WORK	入力する必要なし。（作業領域）
LWK	Integer	INPUT	実数型作業領域配列 WORK のサイズ。 LWK >= (MSIZE+2)*N + (MSIZE+1)*(MSIZE+1) + (N-1)/2+1
INFO	Integer	OUTPUT	エラーコード

（2）エラーコード

値	内容
0	正常終了したことを表す。
0 未満	-i が返された場合，i 番目の引数の入力値が不正な値であることを表す。
100	前処理行列の作成に失敗したため演算途中で終了したことを表す。
200	ブレイクダウンが発生したため演算途中で終了したことを表す。
300	OpenAT_DAFRT の入力値が不正であるため演算途中で終了したことを表す。
400	許容する演算時間を超過したため演算途中で終了したことを表す。
500	最大リスタート回数を超過したため演算途中で終了したことを表す。

４．T2K オープンスパコン（東大版）(HITACHI HA8000 クラスタシステム）での利用法

4.1　ライブラリのインストール先

本ライブラリ（静的ライブラリ）は以下の場所にあります。コンパイル時にリンクしてお使いください。

なお、ログインノードのみライブラリが配布されております。qsub するジョブスクリプトファイル中にコンパイルコマンドを記述されますと、計算ノードでのコンパイルとなります。したがって、本リリース版はコンパイルができません。

- Xabclib 静的ライブラリ一式
  /opt/itc/lib/xabcliblanczos.a　：Xabclib_LANCZOS ソルバー
  /opt/itc/lib/xabclibgmres.a　：Xabclib_GMRES ソルバー
  /opt/itc/lib/libOpenAT.a　：OpenATLib ライブラリ

- Xabclib インクルードファイル
  /opt/itc/include/OpenAT.inc

使い方の一例（Intel Fortran コンパイラ）

```
$ ifort -O3 -m64 –openmp -mcmodel=medium -o hoge hoge.o /opt/itc/lib/xabcliblanczos.a /opt/itc/lib/libOpenAT.a
```

4.2　サンプルプログラム

Xabclib_GMRES、および Xabclib_LANCZOS のサンプルプログラムは以下の場所にあります。T2K オープンスパコンにログイン後、コピーしてお使いください。

- /opt/itc/Xabclib/Xabclib_samples200907.tar.gz

サンプルプログラムのコピーおよび展開方法

```
$　cp /opt/itc/Xabclib/Xabclib_samples200907.tar.gz ./
$　gzip -d Xabclib_samples200907.tar.gz
$　tar xvf Xabclib_samples200907.tar
```

上記で展開され作成されるディレクトリは 2 つあります。その内容は、以下の通りです。

- testLANCZOS
  ：Xabclib_LANCZOS のサンプルプログラム一式
- testGMRES
  ：Xabclib_GMRES のサンプルプログラム一式

4.3　コンパイルのしかた

展開されたディレクトリに入り make することでコンパイルが可能です。

α版サンプルプログラムでは Intel コンパイラでの利用を想定しています。したがって、Intel コンパイラへの切り替え作業が必要です。

以下に、Xabclib_LANCZOS のサンプルプログラムのコンパイル方法を載せます。

```
$　source /opt/intel/Compiler/11.0/074/bin/ifortvars.sh intel64
$　cd testLANCZOS/
$　make
```

以上の手順で、lload という実行ファイルが作成されればコンパイルは終了です。

4.4　ジョブの実行と実行結果

本サンプルプログラムには、debug キューにジョブを投げるジョブスクリプトが添付してあります。Xabclib_LANCZOS サンプルプログラムでは、go-lanczos.bash です。
実行のためには、以下を入力してください。

```
$ qsub go-lanczos.bash
notice: '-lm 2gb' is automatically set.
Request 255679.batch1 submitted to queue: debug.
```

ジョブが実行されますと、結果が OUT ファイルに書き込まれます。OUT ファイルの中身において、以下の結果が書き込まれていれば正しい動作となります。

```
$ cat OUT
 ================================================
 ============== Xabclib_LANCZOS START ==============
 ================================================
 ++++++++++++++ Input Parameter List +++++++++++++++++
 ===> INPUT FILE NAME IS
 ./H2O.rsa
 PARSEC real-space pseudopotential. Zhou, Saad, Tiago, Chelikowsky, UMN
 H2O
 + Matrix Info. N=           67024    NZ=         1141880
 + IPARM(1)= 2 ,IPARM(2)= 5000 ,NEV= 10 ,MSIZE= 120
 + Convergence Criteria= 1.00000E-08    Upper Limit of CPU TIME= 5.000E+03 [sec]
 + OpenMP Number of MAX. Threads=                16
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 <<< Xabclib LANCZOS SUCCESSFUL EXIT >>>
 <<<                 RESULT                      >>>
 NO. of Restart=              40
 CPU_TIME =     35.7481918334961          IN SMPs=              16
 IC=               1 E=     235.244914026497        RES=   5.494705092632958E-009
 IC=               2 E=     234.961735203665        RES=   1.736900498828370E-009
 IC=               3 E=     234.613864053549        RES=   3.065643332002172E-009
 IC=               4 E=     234.609054174968        RES=   6.568143055415601E-010
 IC=               5 E=     234.259760865316        RES=   3.161664366613420E-010
 IC=               6 E=     234.202141466430        RES=   1.333826393176566E-009
 IC=               7 E=     234.194351079241        RES=   2.134069564030512E-009
 IC=               8 E=     233.772844199167        RES=   3.969445024251797E-010
 IC=               9 E=     233.738533772319        RES=   7.810067875798472E-009
 IC=              10 E=     233.733870107103        RES=   1.830964543402003E-009
 !!! OK !!! EIGENVECTOR NORMALIZED
 ================================================
 === ORTHOGONALITY=    1.548306770879186E-015
 ================================================
```

4.5　ライブラリコールのしかた

　具体的なライブラリコールの方法につきましては、サンプルプログラム中のソースコードをご参照ください。Xabclib_LANCZOS は tp_lanczos.f、Xabclib_GMRES は TEST_GMRES.f　が相当します。

4.6　Xabclib プロジェクトの情報

　Xabclib プロジェクトの情報は、[5]のホームページで入手可能です。ご興味のある方はご覧ください。

4.7　ユーザサポートについて

　Xabclib に関するご質問・ご要望は、soudan@cc.u-tokyo.ac.jp までお願いします。

　効率良くユーザサポートを行うため、メールの subject に “[Xabclib]”とご記入くださいますよう、よろしくお願いいたします。

謝辞

本開発は、文部科学省「e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」、シームレス高生産・高性能プログラミング環境、の支援により行われました。また本開発は、日立製作所中央研究所の櫻井隆雄氏、直野健博士との共同研究における成果の一部を利用しています。ここに感謝の意を表します。

参 考 文 献

[1] 櫻井 隆雄, 直野 健, 恵木 正史, 猪貝 光祥, 木立 啓之 :「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」, 第 111 回ハイパフォーマンスコンピューティング研究会、情報処理学会研究報告 2007-HPC-111, pp.173-178, 2007.

[2] 工藤 誠, 黒田 久泰, 片桐 孝洋, 金田 康正 :「並列疎行列ベクトル積における最適なアルゴリズム選択の効果」, 第 147 回アーキテクチャ研究会、情報処理学会研究報告 2002-ARC-147, pp.151-156, 2002.

[3] V. Hernandez, J. E. Roman, and A. Tomas: “Evaluation of Several Variants of Explicitly Restarted Lanczos Eigensolvers and Their Parallel Implementations”, High Performance Computing for Computational Science - VECPAR 2006, pp.403-416, 2007.

[4] Y. Saad: “Iterative methods for sparse linear systems”, 1996.

[5] Xabclib プロジェクトホームページ
http://www.abc-lib.org/Xabclib/index-j.html