Vol.2010-HPC-126 No.42
2010/8/5

# 行列計算ライブラリ向け 数値計算ポリシーインターフェースの提案

櫻井隆雄[†] 直野健[†] 片桐孝洋[††]
中島研吾[††] 黒田久泰[‡,††] 猪貝光祥[§]

科学技術計算等で利用される行列計算ライブラリは高い演算性能が得られるパラメータの選択や入力に多大な手間が必要なため，それを自動的に設定する方式が求められている．この課題に対し，筆者らは要求する精度や時間，メモリ量をポリシーとして入力可能な行列計算ライブラリ向け数値計算ポリシーインターフェースを提案する．提案方式は入力されたポリシーを元に行列計算ライブラリのパラメータの探索範囲を決定し，その中で最適な値を自動選択し演算を実行することを特徴とする．提案方式を既存の行列計算ライブラリに適用し評価した結果，適用前と比べ引数を 6 割以上削減した上で，ユーザの要求を満たすようにパラメータを選択していると確認できた．

# Proposal of Numerical Policy Interface for Sparse Matrix Solver

Takao Sakurai[†] Ken Naono[†] Takahiro Katagiri[††]
Kengo Nakajima[††] Hisayasu Kuroda[‡,††] Mitsuyoshi Igai[§]

Matrix libraries have many parameters as inputs by the user. They include problem parameters what are difficult to set values and the approach of automatically setting them is needed. In this paper, we propose matrix libraries with numerical computation policy interface. By using these libraries, users can specify their numerical policy, such as minimizing computation time, memory saving, and accuracy requirement for the residual of solution without difficult parameter setting. Numerical experiments show the proposed approach can reduce over 60% of parameters and satisfy user needs.

## 1. はじめに

近年，計算機環境の複雑化（超並列化，メニーコア化）により行列計算ライブラリを利用して高い演算性能が得ることが困難になっている．

この原因の一つとして，行列計算ライブラリを使用する上で入力しなければならない引数の設定が困難であることが考えられる．行列計算ライブラリには「行列の次元数」,「解の要求精度」や「エラーコードの入力先」など 20 以上の引数を設定する必要があり，これがユーザへの大きな負担となっていた．さらに，これらの引数の中には，メモリ使用量や演算時間に大きく影響を与えるものも含まれているが，それらに対し，直接「メモリ使用量 100MByte」「演算時間 100 秒」といった形で入力するインターフェースが用意されていなかった．それらを指定する場合は,「作業領域の配列の大きさ」「最大反復回数」という引数にユーザ側で要求を満たすための値を計算，もしくは予想し，間接的に入力しなければならなかった．

そこで，筆者らが開発している自動チューニングインターフェース OpenATLib の新たな機能として，ユーザの数値計算ポリシーの入力を受け付け，それを満たすようにライブラリの引数を自動的に設定する数値計算ポリシーインターフェースを開発した．本機能は数値計算ライブラリを使用する上で入力する引数の種類を極力削減した上で，引数の入力を従来の「最大反復回数 1000 回」,「作業配列サイズ 10000」といった形式から「最大演算時間 100 秒」「最大メモリ使用量 2Gbyte」といったユーザにとって分かりやすい形式にすることを目的としている．

本稿では数値計算ポリシーインターフェースを利用した行列計算ライブラリにおける引数の削減効果と，T2K オープンスーパーコンピュータ上で実施した手動で引数を設定した場合との性能比較評価の結果について述べる．

## 2. AT インターフェース OpenATLib の概要

行列計算ライブラリを利用して高い演算性能が得るために，それらの引数の一部である入力パラメータ値や使用アルゴリズムの選択を支援する自動チューニング方式（AT 方式）が注目されており，これまでに様々な方式が提案されてきた．例えば PHiPAC[1], ATLAS[2], FFTW[3], I-LIB[4], ABCLib[5], OSKI[6]などである．これらによ

† 日立製作所 中央研究所
Central Research Laboratory, Hitachi, Ltd.
†† 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
Supercomputing Research Division, Information Technology Center, The University of Tokyo
‡ 愛媛大学 大学院理工学研究科
Graduate School of Science and Engineering, Ehime University
§ 日立超 LSI システムズ
Hitachi ULSI Systems Co., Ltd.

り良好な条件でライブラリを使用するのが容易になっている．

しかし，自作のライブラリに対し，その作成者が AT 方式を適用する場合，プログラム記述のためにコード量が増加する場合がある．そのため，作成者が多数の AT 方式を自作ライブラリに適用するには非常に多くの労力を必要とする．そこで，これまで個別に提案されてきた自動チューニングの実装に関して再利用性（RIAT: Reusability for Implementation of Automatic Tuning facility）を高める手段が求められている．

この課題に対し，筆者らは，RIAT の実現のため，Fortran で記述された疎行列反復法ライブラリ向けの汎用 AT インターフェース OpenATLib を提案している[7-9]．これは主にライブラリ作成者が自作ライブラリに実行時にパラメータや実装方式を自動的に選択する AT 方式を実装するために利用することを想定している．OpenATLib は AT 方式を備えた疎行列ベクトル積や直交化演算のサブルーチンを API (Application Programming Interface) の形式で提供する．これらの AT 方式により疎行列反復解法ライブラリにおいて実行前に予測が困難なパラメータやアルゴリズムの最適値を実行中に探索し，1 回の演算で安定して高い演算性能を提供することが可能となる．

現在開発中の OpenATLib の $\beta$ 版が備える機能は以下の 4 つである．これらの機能の命名規則は次のとおりである．まず，“OpenATI_”で機能名が始まる．続いて，“_”後の最初の 1 文字は演算精度（S:単精度，D:倍精度）であり，2 文字目と 3 文字目について，補助機能の場合は“AF”とし，演算機能の場合は，2 文字目は行列形状（S:対称，U:非対称），3 文字目は行列の格納形式（R:CRS，C:CCS）を表す．4 文字目と 5 文字目以降は機能名となる．

● リスタート周期最適化機能：**OpenATI_DAFRT**
Krylov 部分空間法において，相対残差の履歴から算出した停滞を判断する指標を用いてリスタート周期の最適値を実行時に判断する．

● 疎行列ベクトル積機能：**OpenATI_D{S|U}RMV**
複数用意された疎行列ベクトル積のアルゴリズムの中で，与えられた行列と実行する計算機環境において最も高い性能が得られる方式を判断する．**OpenATI_DSRMV** が対称行列向けの関数であり，**OpenATI_DURMV** が非対称行列向けの関数である．

● Gram-Schmidt 直交化機能：**OpenATI_DAFGS**
複数用意された Gram-Schmidt 直交化の実装方式の中で, 与えられた行列と実行する計算機環境において最も高い性能が得られる方式を判断する．

● 数値計算ポリシーを反映するためのメタインターフェース：
**OpenATI_LINEARSOLVE | OpenATI_EIGENSOLVE**
ユーザが要求する AT の方針（ポリシー）に基づいてライブラリに演算を実行させるためのメタインターフェース．

本稿ではこれらの機能の中で，数値計算ポリシーを反映するためのメタインターフェースと，それを用いて開発した数値計算ポリシー入力型行列計算ライブラリに関して，機能の詳細を述べるとともに，性能評価を実施する．

## 3. 数値計算ポリシー入力型行列計算ライブラリ

### 3.1 行列計算ライブラリの利便性向上のための要件

行列計算ライブラリが提供する解法は自動チューニング機能の有無に関わらず 20～30 程度の多くの引数を持ち，それら全てに適切な値を与えなければ正しく動作しない．前節で述べたようにいくつかの引数は AT 方式により自動設定されており，他のいくつかの引数は計算対象の行列や連立一次方程式の右辺ベクトル，求める固有値の個数，といったユーザがライブラリに計算を要求する上で不可欠なデータである．一方で，他の引数としてライブラリ開発者により定められた作業領域のサイズや反復回数といったものが存在する．これらは，ライブラリが使用するメモリ量や許容される演算時間に影響するため，自動設定のし難いものだが，ユーザにとって直感的に分かりやすい「最大メモリ使用量 1GByte」「最大演算時間 100 秒」といった形式ではなく，「配列長 10000」「最大反復回数 10000 回」といった形式で与えなければならない．そのため，引数を入力する際にユーザが許容できるメモリ使用量や演算時間からこれらの値を計算して与えなければならない．

また，別の問題として前述のメモリ使用量の削減と演算時間の短縮，それに加えて得られた解の精度の 3 者はトレードオフの関係にあり，ユーザはそれらのいずれかを重要視する場合，それに合わせて引数を設定する必要がある．例えば演算時間の短さを重要視する場合，演算に使用するメモリ量を最大限取り，パラメータの選択範囲を広げることや，解の要求精度を緩くして，より短い反復回数で収束させなければならない．このような引数の設定の手間もユーザにとって大きな負担となる．例えば，OpenATLib を使用し作成した連立一次方程式解法ライブラリ Xabclib_GMRES は 26 個の引数を入力しなければならない．

さらに，上記の課題を解決するには，引数の設定の処理やメモリ使用量の算出など従来は必要としなかった新たな処理が発生し，その結果ライブラリの演算性能が低下する恐れがある．それを防ぐためには，ライブラリ全体の演算時間の半分以上の時間を占める重要な処理である行列ベクトル積を高速化することが有効と考えられる．

以上から，ユーザにとって扱い易いライブラリを作成するために解決すべき課題は

以下の 3 点と考えられる．

①ユーザが必ず入力する引数は最小限とし，ライブラリを動作させるための工数を削減する．
②ユーザが要求する最大の演算時間やメモリ使用量は引数以外の手段で簡易に入力可能とし，演算を実行する上でそれぞれトレードオフの関係にある演算時間，メモリ使用量，解の精度はどれを最も重視するかのポリシーをユーザが指定できるようにする．
③上記 2 課題の解決によりライブラリの演算性能が低下するのを防ぐためにより高速な行列ベクトル積の実装方式を考案する．

これらの課題を解決するため，ライブラリに入力する引数の種類を調査した．調査対象は Xabclib_GMRES と同じく OpenATLib を使った固有値解法ライブラリ Xabclib_LANCZOS である．その結果，ライブラリに入力する引数は**表 1** のように分類できると分かった．

これらの分類の中で課題①の解決の要件となるユーザが入力しなければならない必要最小限の引数を抽出する．(a)，(c)の 2 つはユーザに入力させなければライブラリが何を演算するのか判断できず，演算結果を出力できない．また，(g)がなければ入力値の不正など発生した問題をユーザに通知できない．その他の引数は適切なデフォルト値を与えておけばライブラリを動作させることが可能である．以上から，(a)「解くべき問題」と(e)「演算結果の出力先」，(g)「エラーコード」の 3 つが必要最小限の引数と考えられる．そこで，課題①は従来のライブラリの上に一段上の関数を重ね，その関数の入力は上記 3 種類のみとし，他の引数はこの関数内で自動で設定する形式を取ることで解決できる．以後，この一段上の関数をメタインターフェースと呼称する．

次に課題②のユーザにとり重要な関心事である許容される最大の演算時間とメモリ使用量，そしてそれらのどれを重視するかのポリシーを簡易に入力する手段について検討する．従来の行列計算ライブラリでは演算の諸条件として入力していたもののうち，最大反復回数が最大の演算時間，リスタート周期がメモリ使用量に大きく影響を与えていた．この関係は定式化可能だが，その式は行列計算ライブラリの作成者により任意に決められていたため，それをライブラリユーザがマニュアルを熟読の上，設定しなければならなかった．そこで，演算時間を秒単位，メモリ使用量を GByte，MByte 単位で指定できればユーザにとって直感的に入力しやすくなる．これを解決するには課題①で考案したメタインターフェースに許容される演算時間とメモリ使用量を入力可能とする仕組みを加え，メタインターフェースはそれを満たすように関係式に従い最大反復回数やリスタート周期を設定すればよい．

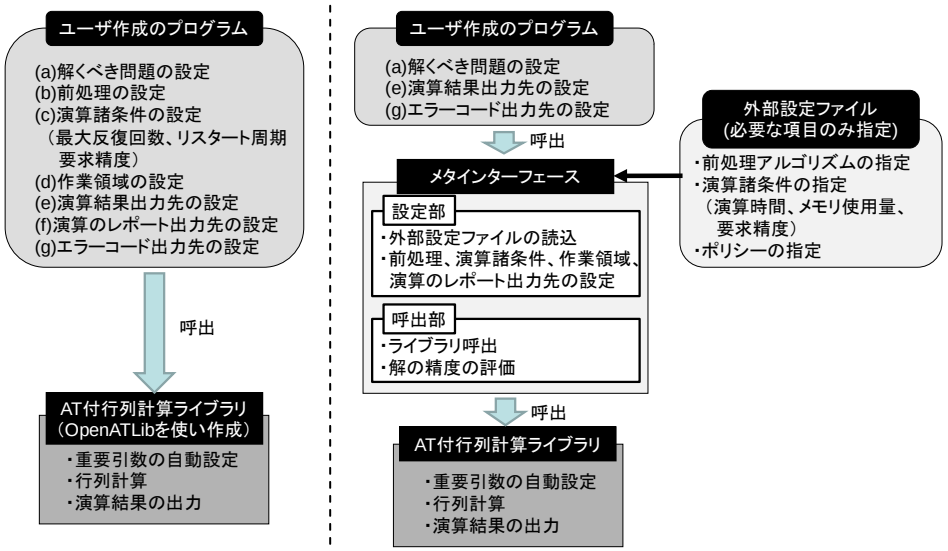
表 1　行列計算ライブラリの引数の分類

| | # | 引数の分類 | 引数の例 |
|---|---|---|---|
| 入力系 | (a) | 解くべき問題 | ・入力行列（次元数，行のポインタ，各要素の値）<br>・右辺ベクトル<br>・求める固有値の個数 |
| | (b) | 前処理の条件 | ・前処理のアルゴリズム<br>・前処理行列のサイズ<br>・前処理行列を格納する配列<br>・その他前処理行列作成のためのパラメータ |
| | (c) | 演算の諸条件 | ・反復法のリスタート周期<br>・収束判定基準となる相対残差<br>・演算打ち切り判断基準となる最大反復回数 |
| | (d) | 作業領域 | ・作業領域のサイズ<br>・作業領域のための配列 |
| 出力系 | (e) | 演算結果の出力先 | ・求めた近似解や固有値の格納する配列 |
| | (f) | 演算の諸条件のレポート | ・収束までの反復回数を格納する変数<br>・近似解の相対残差を格納する変数 |
| | (g) | エラーコード | ・エラーコードを格納する変数<br>（入力値の不正，演算打ち切りなど） |

しかし，これらの条件を引数とするとメタインターフェースそのものの引数が増加し，使いやすさが低下する．この問題を回避するため，これらの条件の入力をユーザが毎回望むわけではないことを考慮し，メタインターフェースに外部設定ファイルを読み込む仕組みを導入する．外部設定ファイルにはメタインターフェース内で設定する前処理や演算諸条件を記述可能とし，ファイルが存在する場合はメタインターフェースはその記述に従い各種設定を行い，存在しない場合はデフォルト値を設定する．

さらに，この設定ファイルを用いてユーザのポリシーを満たす方式について検討する．前章で述べた通り，行列計算ライブラリの演算時間，メモリ使用量，解の精度はそれぞれトレードオフの関係にある．そこで，その 3 つの中で最も重視するものがどれかをユーザが設定ファイルに記述できるようにする．
上記の手段で課題②のユーザが要求する最大の演算時間やメモリ使用量は引数以外の手段で簡易に入力可能とし，更に重視する項目をポリシーとして入力する仕組みは実現できる．

課題①，②を解決するメタインターフェースを導入した構成と従来の構成の比較を**図 1** に示した．メタインターフェースは外部設定ファイルの記述に従いライブラリの




(A)従来の構成　　(B)メタインターフェース導入後の構成

図 1　メタインターフェースの導入による引数の削減

引数を設定する**設定部**と設定された引数を用いてライブラリを呼び出し演算させ，解の精度が重視されている際にその評価を行う**呼出部**により構成される．また，**表 2** に外部設定ファイルに入力可能な項目を示す．POLICY は演算時間，メモリ使用量，解の正確さの中で最優先するものを設定する．MAXMEMORY，MAXTIME はそれぞれ許容する最大のメモリ使用量，演算量を記述する．EPS は解の要求精度，PRECONDITIONING は前処理アルゴリズムを指定する．CPU は演算に使用する CPU の数，SOLVER は将来の拡張に備えて解法を指定できるようにしている．

### 3.2 対称疎行列ベクトル積の評価結果

本節ではメタインターフェースによる各種引数設定およびライブラリ呼出の具体的な手順について検討する．メタインターフェースにより呼び出される行列計算ライブラリは以前作成した Xabclib_LANCZOS と Xabclib_GMRES である．これらのライブラリには OpenATLib による自動チューニング方式が組み込まれており，決められた範囲でリスタート周期や行列ベクトル積実装方式の最適値を探索する機能を備えている．また，再直交化演算は引数の指定により BCGS，MGS，DGKS の 3 方式から選択可能となっている．さらに，他のライブラリにない特徴として最大演算時間を引数で直接指定する仕組みを備えているため，MAXTIME による最大演算時間指定はこれを直接利用する．ここで，Xabclib_LANCZOS を呼び出すメタインターフェースを数値計算ポリシー入力型行列計算ライブラリ **OpenATI_EIGENSOLVE**，Xabclib_GMRES を呼び出すメタインターフェースを同じく **OpenATI_LINEARSOLVE** と名付ける．

表 2　外部設定ファイルに記述可能な項目名

| 項目名 | 説明 | 入力例 | デフォルト値 |
|---|---|---|---|
| POLICY | 演算時間，メモリ使用量，解の正確さの中で最優先するもの | ・TIME（時間優先）<br>・MEMORY（メモリ量優先） | TIME（演算時間優先） |
| MAX MEMORY | 最大メモリ使用量（Gbyte 単位） | ・4.0<br>・24.0 | システム情報から取得した空きメモリ量 |
| MAXTIME | 要求時間 | ・300.0 | 無し |
| EPS | 解の要求精度 | ・1.0D-08 | 1.0D-08 |
| PRECOND ITIONING | 前処理アルゴリズム | ・Jacobi<br>・ILU(0) | ILU(0) |
| CPU | 演算に使用する CPU の数 | ・8<br>・16 | 環境変数から取得（例：OMP_NUM_THREADS） |
| SOLVER | 解法 | ・GMRES<br>・LANCZOS | GMRES（連立一次の場合）<br>LANCZOS（固有値の場合） |

**図 2** にリスタート周期（図中では M）と行列ベクトル積実装方式の探索範囲を設定する設定部の動作を示した．最初にリスタート周期 M と行列ベクトル積の探索範囲は最小限に設定した際の必要メモリ量を計算し，それがユーザの指定する MAXMEMORY よりも大きければ演算不能としてエラーを出力する．小さければほとんどの場合において収束する M の値である 100 を探索範囲の上限に設定した際の必要メモリ量を計算し，MAXMEMORY と比較する．これが確保できない場合はその中で最大の M を計算し呼出部に送る．確保した場合も POLICY が MEMORY である場合は十分演算可能なパラメータであるため，必要以上にメモリ使用量を大きくしないために M の上限を 100 として呼出部に送る．それ以外の場合は行列ベクトル積の探

索範囲を広げ，さらに M の探索範囲を MAXMEMORY の値まで広げる．

続いて，**図 3** にライブラリを呼び出す呼出部の動作を示した．呼出部は POLICY と EPS の値により再直交化の実装方式とライブラリへの再投入の有無といった動作の違いがある．最直交化方式は EPS が 1.0E-10 を下回り，かつ POLICY が TIME でない場合は速度と正確さのバランスのとれた MGS を用い，それ以外の場合は速度重視の BCGS を用いる．その他の引数は設定部および外部設定ファイルの指定により設定し，ライブラリに投入する．ライブラリが解を出力した際に POLICY が ACCURACY の場合は改めて検算を行い，解の信頼性を確認する．このとき，解が要求精度を満たしていなければ収束条件を 0.1 倍し再投入する．この際，再直交化処理は速度に劣るが最も直交化の信頼性の高い DGKS を用いる．この外部反復を要求精度を満たすか最大演算時間になるまで繰り返す．

ここまでに述べたポリシー入力型メタインターフェースの導入によりユーザにとって使いやすい行列計算ライブラリである数値計算ポリシー入力型行列計算ライブラリが実現されると期待できる．

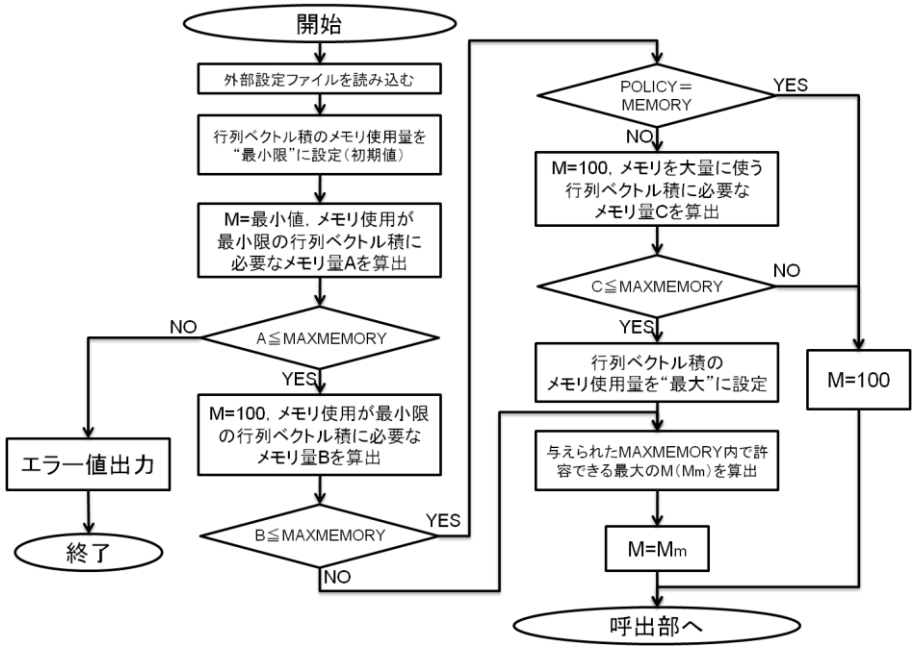



図 2　メタインターフェースの設定部の動作

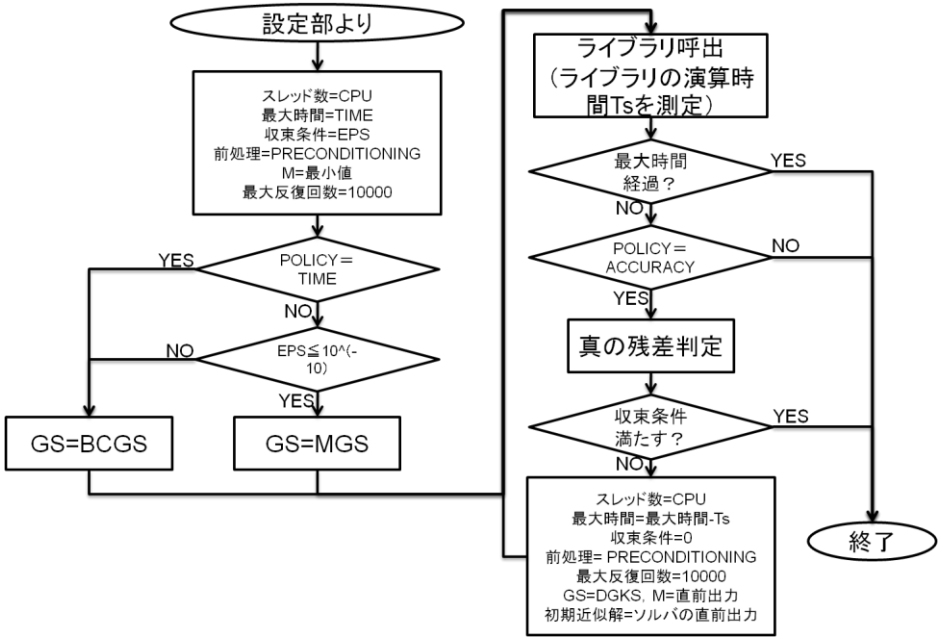



図 3　メタインターフェースの呼出部の動作

## 4. 数値計算ポリシー入力型行列計算ライブラリの評価

### 4.1 メタインターフェースの導入による必須パラメータの削減効果の評価

本節では，提案した数値計算ポリシー入力型行列計算ライブラリによる引数の削減効果を評価する．本評価ではメタインターフェース導入前の Xabclib の 2 ライブラリの引数の種類と，それにメタインターフェースを導入して作成した OpenATI_EIGENSOLVE, OpenATI_LINEARSOLVE の引数の種類を比較した．**表 3** にメタインターフェース導入による各ライブラリの引数削減効果を示した．メタインターフェースの導入によりユーザが入力する引数が固有値解法で 60%，連立一次方程式解法で 69%削減されていた．メタインターフェース導入後の引数は全て解くべき問題や得られた解を出力するための配列，エラーコード用の変数であり，ユーザが値の設定に手間が必要な引数はすべて削減されている．

以上の結果から，提案したメタインターフェースにより行列計算ライブラリの引数の設定を大幅に削減可能であると考えられる．

表 3　メタインターフェース導入による引数削減効果

| 解法 | 引数の種類 | | 削減率 (%) |
|---|---|---|---|
| | メタインターフェース導入前 | メタインターフェース導入後 | |
| 連立一次方程式解法 | 26 | 8 | 69 |
| 固有値解法 | 25 | 10 | 60 |

### 4.2 ポリシー効果の評価環境とテスト行列

ポリシーの設定により，演算時間，メモリ使用量と解の精度がそれぞれ優先されるかを確認するため，評価を実施した．評価内容は演算時間優先時とメモリ使用量優先時の実際の演算時間とメモリ使用量の比較と，演算時間優先時と解の正道優先時の実際の演算時間と得られた解の精度の比較である．本評価は T2K オープンスパコン（東大版）1 ノードを利用した．T2K オープンスパコン（東大版）1 ノードの仕様およびコンパイル環境を**表 4** に示した．

テスト行列はフロリダ大学の Sparse Matrix Collection[10]よりそれぞれ 5 個ずつ選出した．テスト用の行列を**表 5** に示した．その他の解法の条件として，固有値解法において求める固有値の個数は 10，連立一次方程式解法では解ベクトルは全要素が 1，初期近似解ベクトルは全要素 0 とし，前処理として不完全 LU 分解を用いた．収束条件は共に 1.0E-8 とした．

また，ポリシー入力として MAXTIME は 1000 秒，MAXMEMORY は主記憶サイズの半分の 16GByte と設定し，従来の Xabclib の自動チューニング機能としてリスタート周期 M の探索範囲は固有値解法では 10~150，連立一次解法では 2~100 とした．

### 4.3 ポリシー効果の評価結果

最初に，ポリシーに MEMORY（メモリ使用量優先）を設定した場合と TIME（演算時間優先）を設定した場合の演算時間とメモリ使用量を比較した．**表 6**，**表 7** にその結果を示す．8 割弱の行列では両者の速度は 5%以内の差に収まった．しかし，残りの 2 割ではメモリを削減するために M の探索範囲の上限を 100 にしたことの影響を受け，MEMORY の方が大きく性能が劣化しており，特に表 6 の Ga41As41H72 では 6.12 倍となっており，平均すると固有値解法で 1.28 倍，連立一次方程式解法で 1.09 倍に性能が向上していた．一方，演算に使用したメモリ量は TIME の場合は多くの行列で上限の 16GByte まで使用していたが，MEMORY の場合は全ての行列で 0.4Gbyte 以下に収まっており，平均すると固有値解法で 207 分の 1，連立一次解法で 253 分の 1 のメモ

表 4　T2K オープンスパコン（東大版）1 ノードの仕様およびコンパイル環境

| | |
|---|---|
| CPU | Quad-Core AMD Opteron(tm) Processor 8356<br>2.3GHz 16core/1node |
| L2 サイズ | 2MByte/4core |
| 主記憶サイズ | 32GByte（8GByte/1Socket） |
| OS | Red Hat Enterprise Linux 5 |
| コンパイラ | Intel Fortran Compiler Professional Version 11.0 |
| コンパイル<br>オプション | –O3 –m64 –openmp –mcmodel=medium |

表 5　テスト用行列

| 固有値解法向け対称行列 | | |
|---|---|---|
| Matrix | N | NNZ |
| vibrobox | 12328 | 177578 |
| Lin | 256000 | 1011200 |
| cfd1 | 70656 | 949510 |
| cfd2 | 123440 | 1605669 |
| gyro | 17361 | 519260 |
| t3dl | 20360 | 265113 |
| c-71 | 76638 | 468096 |
| Si5H12 | 19896 | 379247 |
| SiO | 33401 | 675528 |
| dawson5 | 51537 | 531157 |
| H2O | 67024 | 1141880 |
| F2 | 71505 | 2682895 |
| oilpan | 73752 | 1835470 |
| shipsec1 | 140874 | 3977139 |
| bmw7st_1 | 141347 | 3740507 |
| SiO2 | 155331 | 5719417 |
| shipsec5 | 179860 | 5146478 |
| Si41Ge41H72 | 185639 | 7598452 |
| bmw3_2 | 227362 | 5757996 |
| Ga41As41H72 | 268096 | 9378286 |

| 連立一次解法向け非対称行列 | | |
|---|---|---|
| Matrix | N | NNZ |
| chem_master1 | 40401 | 201201 |
| torso2 | 115967 | 1033473 |
| torso1 | 116158 | 8516500 |
| torso3 | 259156 | 4429042 |
| memplus | 17758 | 126150 |
| ex19 | 12005 | 259879 |
| poisson3Da | 13514 | 352762 |
| airfoil_2d | 14214 | 259688 |
| poisson3Db | 85623 | 2374949 |
| viscoplastic2 | 32769 | 381326 |
| xenon1 | 48600 | 1181120 |
| xenon2 | 157464 | 3866688 |
| wang4 | 26068 | 177196 |
| ecl32 | 51993 | 380415 |
| sme3Da | 12504 | 874887 |
| sme3Db | 29067 | 2081063 |
| sme3Dc | 42930 | 3148656 |
| epb1 | 14734 | 95053 |
| epb2 | 25228 | 175027 |
| epb3 | 84617 | 463625 |

表 6　ポリシー選択による固有値解法の演算時間とメモリ使用量の変化

| 行列名 | POLICY=TIME | | POLICY=MEMORY | | TIME による速度向上率 | MEMORY によるメモリ使用量削減比 |
|---|---|---|---|---|---|---|
| | 演算時間 (秒) | メモリ (Gbyte) | 演算時間 (秒) | メモリ (Gbyte) | | |
| Vibrobox | 0.45 | 3.65 | 0.44 | 0.01 | 0.98 | 365.00 |
| Lin | 178.44 | 16.02 | 253.04 | 0.24 | 1.28 | 66.75 |
| cfd1 | 7.79 | 16.01 | 7.74 | 0.07 | 0.99 | 228.71 |
| cfd2 | 14.39 | 16.02 | 14.39 | 0.13 | 1.00 | 123.23 |
| Gyro | 0.37 | 7.24 | 0.39 | 0.02 | 1.05 | 362.00 |
| t3dl | 2.41 | 9.95 | 2.35 | 0.02 | 0.98 | 497.50 |
| c-71 | 0.79 | 16.01 | 0.86 | 0.07 | 1.08 | 228.71 |
| Si5H12 | 3.10 | 9.51 | 3.03 | 0.02 | 0.98 | 475.50 |
| SiO | 5.73 | 16.01 | 5.54 | 0.04 | 0.97 | 400.25 |
| dawson5 | 15.11 | 16.01 | 19.12 | 0.05 | 1.27 | 320.20 |
| H2O | 36.01 | 16.02 | 32.29 | 0.07 | 0.89 | 228.86 |
| F2 | 7.39 | 16.03 | 7.50 | 0.10 | 1.02 | 160.30 |
| Oilpan | 6.69 | 16.02 | 6.44 | 0.09 | 0.96 | 178.00 |
| shipsec1 | 20.35 | 16.06 | 20.23 | 0.17 | 0.99 | 94.47 |
| bmw7st_1 | 0.43 | 16.05 | 0.44 | 0.17 | 1.02 | 94.41 |
| SiO2 | 103.18 | 16.07 | 105.81 | 0.21 | 1.03 | 76.52 |
| shipsec5 | 36.99 | 16.07 | 37.86 | 0.22 | 1.02 | 73.05 |
| Si41Ge41H72 | 152.19 | 16.09 | 156.11 | 0.26 | 1.03 | 61.88 |
| bmw3_2 | 3.59 | 16.08 | 3.59 | 0.27 | 1.00 | 59.56 |
| Ga41As41H72 | 151.90 | 16.13 | 929.36 | 0.35 | 6.12 | 46.09 |
| 平均 | | | | | 1.28 | 207.05 |

表 7 ポリシー選択による連立一次方程式解法の演算時間とメモリ使用量の変化

| 行列名 | POLICY=TIME | | POLICY=MEMORY | | TIME による速度向上率 | MEMORY によるメモリ使用量削減比 |
|---|---|---|---|---|---|---|
| | 演算時間 (秒) | メモリ (Gbyte) | 演算時間 (秒) | メモリ (Gbyte) | | |
| chem_master1 | 1.50 | 16.00 | 1.52 | 0.04 | 1.00 | 400.00 |
| torso1 | 0.73 | 16.11 | 0.76 | 0.20 | 1.04 | 80.55 |
| torso2 | 0.30 | 16.02 | 0.30 | 0.11 | 1.00 | 145.64 |
| torso3 | 30.80 | 16.06 | 30.64 | 0.28 | 0.99 | 57.32 |
| memplus | 0.19 | 5.05 | 0.19 | 0.02 | 1.02 | 252.50 |
| ex19 | 12.70 | 2.31 | 25.94 | 0.01 | 2.04 | 231.00 |
| poisson3Da | 0.53 | 2.93 | 0.53 | 0.02 | 1.00 | 146.50 |
| poisson3Db | 10.37 | 16.03 | 10.42 | 0.10 | 1.01 | 160.30 |
| airfoil_2d | 0.77 | 3.24 | 0.77 | 0.02 | 1.00 | 162.00 |
| viscoplastic2 | 1.55 | 16.00 | 1.46 | 0.03 | 0.94 | 533.33 |
| xenon1 | 14.98 | 16.01 | 14.98 | 0.06 | 1.00 | 266.83 |
| xenon2 | 58.47 | 16.05 | 58.56 | 0.18 | 1.00 | 89.17 |
| wang4 | 0.25 | 10.88 | 0.25 | 0.02 | 1.00 | 544.00 |
| ecl32 | 9.03 | 16.01 | 9.15 | 0.06 | 1.01 | 320.20 |
| sme3Da | 91.44 | 2.51 | 99.70 | 0.02 | 1.09 | 125.50 |
| sme3Db | 256.40 | 13.55 | 388.47 | 0.05 | 1.52 | 271.00 |
| sme3Dc | 319.21 | 16.03 | 393.60 | 0.07 | 1.23 | 229.00 |
| epb1 | 0.30 | 3.48 | 0.30 | 0.01 | 1.00 | 348.00 |
| epb2 | 0.19 | 10.19 | 0.19 | 0.02 | 1.00 | 509.50 |
| epb3 | 2.29 | 16.01 | 2.26 | 0.08 | 0.99 | 200.13 |
| 平均 | | | | | 1.09 | 253.62 |

表 8　ACCURACY の選択による性能の変化と解の信頼性の向上（要求精度 1.0E-8）

| 行列名 | POLICY=TIME | | POLICY=ACCURACY | |
|---|---|---|---|---|
| | 演算時間 (秒) | 解の 相対残差 | 演算時間 (秒) | 解の 相対残差 |
| torso3 | 30.80 | 2.28E-08 | 34.77 | 2.59E-09 |
| airfoil_2d | 0.77 | 5.17E-05 | 1.32 | 3.03E-10 |
| viscoplastic2 | 1.55 | 3.67E-05 | 2.59 | 2.02E-09 |
| Ga41As41H72 | 151.90 | 1.52E-08 | 290.51 | 9.30E-09 |

リ使用量で演算していた．以上から，メモリ使用量を抑えたい場合は MEMORY，演算時間を優先する場合は TIME を選択するのが有効であるとポリシー選択の効果が確認できた．最後に，ポリシーに ACCURACY を選択した場合の解の信頼性向上効果を評価した．一度のライブラリの呼び出しで得られた解が要求精度を満たした場合，ACCURACY 選択時の動作は TIME と変化しないように設計している．そのため，本評価では一度目の呼び出しで要求精度を満たさなかった 4 つの行列についてポリシーを TIME に設定した場合と ACCURACY に設定した場合の演算時間と得られた解の相対残差を比較した．**表 8** にその結果を示す．

表 8 の結果によると，TIME を選択した場合，ACCURACY と比べ 13~91%演算性能が高いが，得られた解が要求精度に満たない場合があると分かった．一方，ACCURACY では常に解が要求精度を満たしていた．この結果から，ACCURACY を選択した場合は外部反復により演算性能が低下するが，解が常に要求精度を満たすため，解の信頼性を重視する場合に効果があるとわかった．

以上から，提案したメタインターフェースを適用した数値計算ポリシー入力型行列計算ライブラリは従来の行列計算ライブラリから性能を落とすことなくユーザの引数入力の手間を大幅に削減でき，さらにポリシーの入力を備えることによりユーザの重視する要件に合わせた演算を実行できると確認できた．

## 5. おわりに

本稿では，行列計算ライブラリの使用における引数の入力の手間を削減するために，数値計算ポリシー入力型のメタインターフェースを提案し実装した．提案方式は行列計算ライブラリの多くの引数をユーザの要求する最大演算時間，最大メモリ使用量，要求精度の 3 点を満たすように設定する．さらに，その 3 点の中で特にユーザが重視するものがポリシーとして入力されると他の 2 点が上限を超えない範囲でそれが最良となるように引数を設定する．

提案方式の有効性を確認するため，メタインターフェースを行列計算ライブラリ Xabclib に適用し，数値計算ポリシー入力型行列計算ライブラリを開発し，評価した．その結果，提案方式によりユーザの入力する引数の種類を従来から最大で 18 個（69%）削減でき，さらに，ユーザの入力するポリシーによって，演算時間，メモリ使用量，解の信頼性のそれぞれが最適化されることが確認できた．

今回の評価結果ではメモリ使用量が重視された場合と演算時間が重視された場合に，8 割程度の行列では演算時間に大きな差が見られなかった．今後の課題として，メモリ使用量が多く高速な行列ベクトル積を実装するなど，重視項目ごとの差が大きく見えるような方式を検討する．

なお，OpenATLib とそれを用いて開発された疎行列反復法ライブラリ Xabclib のソースコードは，PC クラスタコンソーシアム経由で配布されている．

**謝辞**　本研究は文部科学省「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，の支援を受けている．

## 参考文献

1)　Bilmes, J., Asanovic, K., Chin, C.-W., and Demmel, J.W.: Optimizing Matrix Multiply Using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, Proceedings of International Conference on Supercomputing 97, pp.340-347 (1997).
2)　Whaley, R.C., Petitet, A., and Dongarra, J.J.: Automated Empirical Optimizations of Software and the ATLAS Project, Parallel Computing, 27, pp.3-35 (2001).
3)　Frigo, M.: A Fast Fourier Transform Compiler, Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation, Atlanta, Georgia, pp.169-180 (1999).
4)　片桐孝洋，黒田久泰，大澤清，工藤誠，金田康正: 自動チューニング機構が並列数値計算ソフトウエアに及ぼす効果，情報処理学会論文誌 : ハイパフォーマンスコンピューティングシステム， 42, SIG 12 (HPS 4), pp.60-76 (2001).
5)　Katagiri, T., Kise, K., Honda, H., and Yuba, T: ABCLib_DRSSED: A Parallel Eigensolver with an Auto-tuning Facility, Parallel Computing, 32, 3, pp.231-250 (2006).
6)　Vuduc, R., Demmel, J.W., and Yelick, K.: OSKI:A Library of Automatically Tuned Sparse Matrix Kernels, Proceedings of SciDAC 2005, Journal of Physics: Conference Series (2005).
7)　片桐孝洋，櫻井隆雄，黒田久泰，直野健，中島研吾，OpenATLib:汎用的な自動チューニングインターフェースの設計と実装，情報処理学会研究報告:ハイパフォーマンスコンピューティング, 2009-HPC-121(3) (2009).
8)　櫻井隆雄，直野健，片桐孝洋，中島研吾，黒田久泰，OpenATLib を利用した疎行列ライブ

Vol.2010-HPC-126 No.42
2010/8/5

ラリの開発と評価, 情報処理学会研究報告:ハイパフォーマンスコンピューティング, 2009-HPC-121(17) (2009).
9) Xabclib プロジェクトホームページ, http://www.abc-lib.org/Xabclib/index-j.html
10) T. A. Davis: Sparse Matrix collection, http://www.cise.ufl.edu/research/sparse/matrices/