特集 科学技術計算におけるソフトウェア自動チューニング

< ソフトウェア自動チューニング技術の応用 >

6 自動チューニング機能付き数値計算ライブラリ

黒田 久泰 愛媛大学大学院理工学研究科
直野 健 (株)日立製作所 中央研究所
岩下 武史 京都大学学術情報メディアセンター

ライブラリは，汎用性の高い複数のプログラムを再利用できる形でひとまとまりにしたものである．身近な例として，C 言語でよく使われる printf や malloc 関数などは C 言語標準ライブラリ関数と言われているものである．ライブラリといってもファイル入出力，メモリ管理，通信に関するものなど数多く存在するが，ここでは特に数値計算ライブラリを取り上げる．そして，自動チューニング技術がどのように数値計算ライブラリで適用されているのかについて紹介する．本稿では，まず自動チューニング機能を取り入れていない従来型の数値計算ライブラリについて紹介し，その後，自動チューニング機能付き数値計算ライブラリについて紹介する(図 -1)．

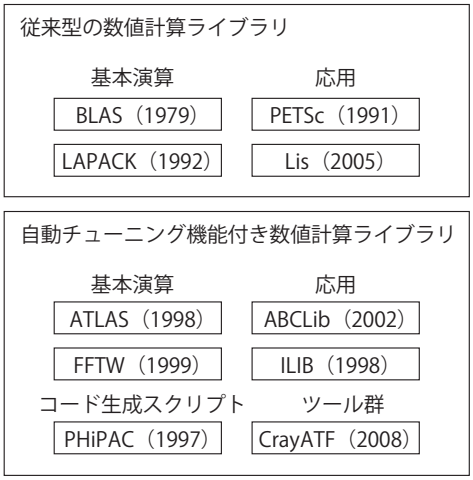

図 -1 本稿で紹介する数値計算ライブラリ
(括弧内の数字は最初に発表された年)

従来型の基本数値計算ライブラリ

最初に，数値計算ライブラリにおける最も有名なサイトとして知られる Netlib（http://www.netlib.org/）について簡単に触れておく．Netlib は 1985 年に電子メール経由で配布が開始されたところから始まり，1994 年から今のように Web サイトでサービスを行っている．ベル研究所，テネシー大学，オークリッジ国立研究所，および，世界各地の有志によって，管理維持されている．この Netlib では自由に利用できるソフトウェア，ドキュメント，データベースなどが公開されており，ほとんどの数値解法のプログラムが Netlib で手に入るといっても過言ではない．

■ BLAS（http://www.netlib.org/blas/）

基本線形計算副プログラム BLAS（Basic Linear Algebra Subprograms）はベクトルと行列に関する基本演算を集めたライブラリであり，細かくは下記の 3 つに分類される．

BLAS1（1979 年）・・・ベクトル同士の演算
BLAS2（1986 年）・・・行列とベクトルの演算
BLAS3（1988 年）・・・行列と行列の演算

BLAS1 をレベル 1BLAS，BLAS2 をレベル 2BLAS，BLAS3 をレベル 3BLAS という書き方をすることも多い．

この BLAS の API（アプリケーション・プログラミング・インタフェース）が，ベクトルと行列に関する基本演算の命名規則のデファクトスタンダードとなっており，同様な演算を行う数値計算ライブラリの多くがこの BLAS と同一の API を備えている．

■ LAPACK（http://www.netlib.org/lapack/）

LAPACK（Liner Algebra PACKage）は，連立 1 次方程式，線形最小二乗問題，固有値問題，特異値分解を解くためのライブラリである．1992 年に LAPACK 1.0 が公開され，2009 年 4 月には LAPACK 3.2.1 が公開されている．LAPACK は内部で BLAS ライブラリを呼び出しているため，LAPACK の性能を考える場合，BLAS の性能が重要

となってくる．

BLAS や LAPACK のソースプログラムは，Netlib の Web ページからダウンロードすることができる．しかし，これらのソースプログラムを単純にコンパイルしてライブラリを構築したのでは，ほとんどの場合，高い性能は期待できない．通常は，ハードウェアベンダ各社から提供されている高度に最適化されたライブラリを利用するのが一般的である．たとえば，AMD 社の ACML（AMD Core Math Library）や Intel 社の IMKL（Intel Math Kernel Library）などは，BLAS と同一の API を備えており BLAS ライブラリとして利用できる．特に BLAS や LAPACK といったライブラリは，他のライブラリやアプリケーションからも利用されることが多いため，アプリケーション全体の性能に大きく影響を及ぼすことが多い．

従来型の応用数値計算ライブラリ

1980 年代以降，線形方程式の解法の分野は著しい発展を遂げている．たとえば，大規模疎行列を係数行列とした連立 1 次方程式の反復解法では新たな解法や前処理の手法が数多く提案されている．一方，ハードウェアについても，1990 年頃から分散メモリ型の並列計算機などが広く普及し，これらの計算機に対応した並列数値計算ライブラリが広く求められるようになってきている．ここでは，線形方程式の解法に関連する数値計算ライブラリとして，PETSc と Lis について紹介する．

■ PETSc（http://www.mcs.anl.gov/petsc/）

PETSc（The Portable Extensible Toolkit for Scientific Computation）はアルゴンヌ国立研究所で開発されている数値計算ライブラリである．1991 年に公開されて以来，バージョンアップが繰り返され，2008 年 12 月にはバージョン 3.0.0 が公開されている．MPI（Message Passing Interface）を用いた並列化が行われているが，ユーザは MPI を用いることなく，複数ノードに跨ったベクトル処理や行列処理を PETSc が提供している API を用いることで行うことができる．そのため，PETSc を用いた逐次プログラムを並列化する場合，プログラムの変更は最小限で済むといった利点がある．また，線形方程式の解法に関しては，実に多くの反復解法や前処理に対応している．線形方程式の解法以外にも，非線形方程式の解法，微分方程式の解法に対応している．

■ Lis（http://www.ssisc.org/lis/）

Lis（a Library of Iterative Solvers for linear systems）は日本国内において大規模シミュレーション向け基盤ソフトウェアの開発を行っている SSI（Scalable Software Infrastructure for Scientific Computing）プロジェクトで開発された線形方程式および固有値問題を反復解法で解くための数値計算ライブラリである．2005 年にバージョン 1.0.0 が公開され，2009 年 3 月にバージョン 1.2.6 が公開されている．MPI や OpenMP などの並列化に対応しており，PETSc 同様，並列環境への移行に際して，プログラムの変更は最小限で済む．11 種の疎行列格納形式，22 種の線形方程式の解法，7 種の固有値解法，10 種の前処理に対応している．また，4 倍精度演算もサポートしており，とても機能が充実している [1]．

広く利用されている数値計算ライブラリは言うまでもなく信頼性が高い．そのため，これらの数値計算ライブラリを利用することで，バグの存在個所をユーザプログラムの部分に限定させることができ，開発時間の大幅な短縮が見込める．しかし，公開されている数値計算ライブラリのすべてが，アーキテクチャを考慮した最適化が十分に行われているわけではない点に注意する必要がある．また，各パラメータ省略時の設定が，安全側に倒れていることも多い．その場合，高い性能を出せないことになる．たとえば PETSc の初期のバージョンでは，ベクトル直交化の処理に関して IRCGS（CGS を 2 回行って誤差を低く抑える）といった方法がとられており，直交化に多くの時間を費やしていた．

従来型の応用数値計算ライブラリの利用に関する問題点の 1 つに，ユーザが高い性能を得るための正しい使い方をするのが非常に困難であるという点が挙げられる．現在のほとんどの数値計算ライブラリにおいて，通常，ユーザは問題を解くのに適した解法を自ら選択し，必要ならば，解の精度や実行時間に影響を与えるパラメータの値を自ら決定しなくてはならない．しかし，解法の選択やパラメータの値を間違えてしまうと，必要以上に多くの計算時間を要することとなる．たとえば，線形方程式の反復解法で用いられる GMRES（m）法では，リスタート周期の設定を行う必要がある☆1．PETSc では省略すると 30 が設定されるが，多くの場合これは最適な値ではない．このリスタート周期の最適な値は，行列や適用する前処理さらには計算機によっても違ってくる．そのため，ユーザ自身が最適な値を設定するのは非常に困難である．

☆1 GMRES 法（Generalized Minimal RESidual method）は，残差をクリロフ部分空間において最小化することにより近似解を計算する方法であり，GMRES（m）法は計算量と使用メモリ量の増大を防ぐために反復 m 回ごとに現在の近似解を初期値として最初から計算をやり直すといった方法である．この m の値をリスタート周期と呼んでいる．

また，コンパイラが扱う最適化の範囲は非常に限定的で，ソースコード内の局所的な速度向上にしか対応しきれていないというのが現状である．ソースコードが提供されているライブラリでは，計算機ごとにコンパイルを行ってライブラリを構築する．しかし，線形方程式の解法などのライブラリの場合には，コンパイルの段階では与えられる行列のサイズや性質，実行プロセス数といった情報はまったく未知であるため，問題に特化するような高度なチューニングは一切行うことができない．そのため，高い性能を持ったライブラリを実現するためには，ライブラリ実行時に何らかのチューニングの仕組みが備わっていなければならないことになる．そこで，登場してきたのが自動チューニングという概念である．

自動チューニング機能付きライブラリ

数値計算ライブラリの中には，自動チューニングの仕組みを持つものがいくつか存在している．それはベクトルや行列の基本演算を対象としたものから，線形方程式の解法といった高い階層で自動チューニングを取り入れたものまである．ここではそういったライブラリの中から 6 つを紹介する．

■ PHiPAC

PHiPAC[2] は Blimes らにより提案された，ANSI C で記述された行列計算ループを高性能化するコーディング方法である．文献 2）では，このコーディング方法に基づいた行列乗算ループ（DGEMM）をチューニングするスクリプト言語が提案され，PHiPAC を自動チューニング付きライブラリとして呼ぶ場合，このスクリプトを指す．コーディング方法としての PHiPAC 自体は，9 つのテクニックから構成される．

(1) 依存関係を解消するためのローカル変数の利用

たとえば，

```
a[i] = b[i] + c;
a[i+1] = b[i+1] * d;
```

のコードは，b[i+1] をロードする前に強制的に a[i] をストアしようとする．a[i] のアドレスと b[i+1] のアドレスが異なるとは想定しておらず，両者のアドレスが同じ場合にも正しい動作を行わせる必要があるためである．これをローカル変数 f1，f2 を利用し，次のように記述すると，

```
float f1, f2;
f1 = b[i]; f2 = b[i+1];
a[i] = f1 + c;  a[i+1] = f2 * d ;
```

遂次性が解消され，速度が向上する．

(2) 多くの整数レジスタおよび浮動小数点レジスタの活用

(3) オフセットアドレスを利用することによるポインタ更新の最小化

(4) 複数のローカル変数を用いることによる演算レイテンシの隠蔽

(5) 命令比率の調整（浮動小数点乗算 1 回に対して，加算 1，ロードまたはストアを 1 ～ 2 回といったように調整する）

(6) キャッシュ性能を上げるためのメモリアクセスの局所性向上

(7) ループ内に現れる整数の乗算を整数の加算に変更

(8) 分岐命令や大小比較命令の削減

(9) ループアンローリングの明示的記述

これらのテクニック自体は提案された 1997 年頃のマイクロプロセッサアーキテクチャとコンパイラの性質に依存する部分が多いが，行列計算という単なる数学上の計算からどのようにコンパイラの性質に合わせて行列計算のプログラムコードを記述するかという観点では，現在の自動チューニング研究の発端として非常に意義深い．

さて，上記 9 つのテクニックは ANSI C の範囲でポータブルであるが，当然，可読性を犠牲にすることになる．そこで，科学技術計算で汎用的に利用される行列計算ライブラリ向けに上記のテクニックがパッケージされた「自動チューニング機能付きライブラリ」が提供されることが望まれる．そこで文献 2）では，行列乗算に限って「自動チューニング機能付きライブラリ」を提案し，Sun Sparc-20/61，HP712/80i，IBM RS/6000 590 などのマシン上で，ベンダ提供の行列乗算とほぼ互角の性能を達成している．

この行列乗算向け「自動チューニング機能付きライブラリ」では，コードジェネレータ mm_gen を提供している．この mm_gen は，レジスタ，L1 キャッシュ，L2 キャッシュといったメモリ階層に応じた行列積の C のソースコードを複数生成する．サーチスクリプトは，生成されたソースコードの中で，実行するマシンに最適なレジスタブロック数（＝アンローリング段数）を定め，次に最適な L1 キャッシュブロック数，最適な L2 キャッシュブロック数と順次定めていく．この順番で最適なパラメータ値を決定できる保証はないが，経験上，良好な結果を得ることが分かっている．

文献 2）では，mm_gen によって出力されるソースコードの例が 1 つ示されている．単純な行列乗算実装は 3 重ループなので 7 行で済むが，出力されたソースコードはループ部分で 31 行，ローカル変数定義を含めると 44 行にもなる．これは 3 つのブロック数のパラメータ

値がすべて2の場合の行数であり，計算機によってはこれがもっと長くなる．しかし，単純な実装と比較して，およそ2～5倍もの性能を達成している．マイクロプロセッサアーキテクチャや，メモリ階層が複雑化する計算機プラットフォームでの高性能達成には，ナイーブな実装(手で書くだけ)では，もはや難しいことが実感できる．

■ ATLAS（http://math-atlas.sourceforge.net/）

ATLAS（Automatically Tuned Linear Algebra Software）[3]は，1998年に米国テネシー大学で開発された自動チューニング機能を備えた行列計算ライブラリである．BLASをフルサポートし，LAPACKの一部の機能にも対応している．PHiPACと同様に手動チューニングの作業量が爆発的に増えたことに対する解決策として検討が始まった．ただ，PHiPACとは異なり，自動チューニング機能付きライブラリを含むソフトウェア性能チューニング技術をAEOS（Automated Empirical Optimization of Software）と呼ばれる実証的自動最適化の概念として抽象化している．

このAEOSは，1990年代後半に提案されていた“self-tuning libraries”（適応型チューニングライブラリ）や“adaptive software”（適応型ソウトウェア）あるいは“empirical compilation”（経験的コンパイル実行）や“iterative compilation”（反復的コンパイル実行）など，自動化方法と経験的アプローチの両者による性能最適化技術が共通して持っている性質をまとめて表現した総称である．その共通の性質とは以下のようなものである．

- 最適パラメータや最適ソースコードなどの決定が何らかの方法で自動化されている．
- しかし，どれが最適なものかを決定するのは，経験的に，実測データに基づいて定め，単なるプロファイリング情報だけで決定しない．
- 対象のソフトウェアを，計算機環境に適合させることができる．

そこで，AEOSを踏襲するライブラリに求められる基本要件として，以下の4点を挙げた．

① 性能に影響するコードをサブルーチンとして分離し，最適なサブルーチンを選択実行できるようにすること
② 異なる環境にソフトウェアを適合させる方法（性能に影響するパラメータを持たせたり，ソースコードジェネレータを提供したりすること)が提供されること
③ 実行中のプログラムの正確な実行時間を計測するためのタイマーを持つこと
④ 最適なサブルーチンを自ら選択し，かつ，早く選択できるような探索方法が実装されていること

ATLASでは，特に②について，2つのソフトウェア適合方法（Methods of Software Adaptation）が利用されている．第1に，パラメータ適合（parameterized adaptation）である．これは，行列計算特有のブロック化アルゴリズムでのブロック長を決定する際に利用される．第2に，ソースコード適合（source code adaptation）である．ソースコード適合は，細かくは，多重実装（multiple implementation）とソース生成(source generation)がある．前者は，パラメータ化できないパターンを実装するためや，ATLAS以外で発見された実装と接続可能なように用意されている．後者は，計算機によって異なる命令キャッシュサイズや同時実行可能な演算命令の組合せなどいくつものパターンが考えられる際に必要なソースを自動生成するもので，この中の1つが④の探索方式と③のタイマーによる実測で選択される．

ATLAS 3.8.0[4]のサポート範囲は，LAPACKのLU分解などの10機能（GESV, GETRF, GETRS, GETRI, TRTRI, POSV, POTRF, POTRS, POTRI, LAUUAM）とBLASの全機能である．LAPACK向けには，従来のLAPACKではブロックサイズが固定化されていたのに対し，ATLASではブロックサイズが動的に決定されるため，より高性能になっている．また，BLAS2をBLAS3に置き換える実装も行っている（もちろんBLAS3への置き換えが常に行われるのではなく，うまくいく場合に限られる）．BLAS向けには，以下のようになっている．BLAS3向けには，30ルーチン（realが6つ，complexが9つ，singleとdoubleで倍の計30）が用意されているが，すべてgemmKと呼ばれる1つのカーネルが利用されている．ただし，一部の実装にアセンブラを組み込んでいる．この理由として，Whaleyは，コンパイラが変わっても性能を維持するため，そして，特にSIMD（Single Instruction Multiple Data)命令を最適化するため，と述べている．現在，x86，x86-64，PowerPC，PA-RISC，MIPSの32bit，64bitのアセンブラに対応しているが，gemmKはPentium 4，Pentium 4 E，Efficion，core2Duo，Opteron，PowerPCでは高度に最適化されている．BLAS2向けには，66ルーチンが用意され，GEMV（行列ベクトル積）とGER（ランク1アップデート）など複数のカーネルが用意されている．BLAS1向けには，ATLASはあまりチューニングされておらず，ベンダBLASほどの性能がない場合もある．LAPACK，BLAS3，BLAS2は，上記2つのソフトウェア適合方法が適用されているが，BLAS1向けには多重実装（multiple implementation）のみ適用されている．

■ FFTW（http://www.fftw.org/）

FFTW（Fastest Fourier Transform in the West）は1999年に米国MITで開発された自動チューニング機能を備えた高速フーリエ変換FFT（Fast Fourier Transform）ライブラリである．FFTでは，通常，入力要素の数値がどういう値であっても演算順序や演算回数に変化がないため，入力要素数が決まれば実行時間も決まる．FFTWの特長の1つとして，インストール時だけではなく，ライブラリ実行時にもチューニングを行わせることができる点が挙げられる．これは，インストール時にあらゆる入力要素数に対して最適な計算方法を決定するのは困難であるため，実行時に入力要素数に応じた最適な計算方法を選択することができるようになっている．具体的には，FFTでは，基底と呼ばれる一度に処理する要素数を増やすことで，メモリへのアクセス回数や演算量を削減して高速化を行うことができる．しかし，浮動小数点レジスタの個数は限られているため，基底を大きくしすぎてもよくない．FFTでは大規模な要素数に対して，2基底，4基底，8基底といった複数の基底を組み合わせて用いることになるが，これらの組合せを実行時に決定するというものである．同じ入力要素数に対するFFTを繰り返し行う場合に，この自動チューニングの効果がとても大きくなる．

■ CrayATF

CrayATF（Cray Auto Tuning Framework）は2008年からCray社が開発している自動チューニングされたライブラリ生成のためのツール群である．テスト行列とライブラリソースコードの自動生成を行い，それらを用いて最適なパラメータ値および最適なコンパイラオプションの組合せを見つけることでライブラリを構築する．バッチ処理や性能解析部分も含め，すべてが自動化されている．膨大なパラメータ探索空間の中から最適なものを見つけることになるが，対象をCray社のスーパーコンピュータXTシリーズに絞っているために探索空間を限定でき，ライブラリ構築時間を短縮できる．CrayATFを用いたライブラリとして，CASKとCRAFFTの2つがリリースされている．

CASK（Cray Adaptive Sparse Kernels）は線形方程式の反復解法で多くの時間を費やす疎行列–ベクトル積を対象としている．PETScと組み合わせて利用することが可能である．CASKでは，ライブラリ構築時にさまざまなテスト疎行列と多数の疎行列–ベクトル積のソースコードの生成を行い，その後，最適なコンパイラオプションを見つけるという一連のチューニング作業をCrayATFで自動的に行っている．また，ライブラリ実行時には短時間で最適なコードを自動選択する．同様にCRAFFT（CRay Adaptive FFT）はFFTを対象としたライブラリである．

CrayATFは，HPL（High Performance Linpack）のチューニングにも使用されている．このHPLは，スーパーコンピュータの性能ランキングであるTOP500で利用されるベンチマークソフトウェアである．2008年11月，米国オークリッジ国立研究所のスーパーコンピュータJaguar（Cray XT5）において，一般の科学者向けに開放されたコンピュータでは初めてHPLによる実効性能で1PFlopsを超え，同月発表のTOP500ランキングでは世界第2位を記録するなど，システムのチューニング作業コストの削減に貢献している．

ここまで紹介したライブラリは，数値計算の中でも基本演算部分に該当する．これらは利用頻度がかなり高いためアプリケーション全体の速度向上に大きく貢献することは言うまでもない．しかし，自動チューニングの対象範囲が限定的であるため，数値計算応用の分野における広範囲な問題を解決できているとは言い難い状況である．一般に数値計算応用の分野では，BLASなどの基本演算をもとにした階層（基本演算階層）での処理を，基本構成要素として包含する．しかし，数値計算応用における高性能化，高安定性の要請を考えると，対象を基本演算階層に限定しない，より高い演算階層での高性能化や高安定性を達成できる方式が必要となる．ここからは，高い階層で自動チューニングを行うライブラリとしてILIBとABCLibの2つを紹介する．

■ ILIB

ILIB（Intelligent LIBrary）[5]は，1998年から当時東京大学にいた片桐・黒田らによって開発されてきた自動チューニング機能付きライブラリである．1998年並列処理シンポジウム（JSPP'98）に合わせて開催された並列ソフトウェアコンテスト「疎行列連立一次方程式の並列解法」が開発のきっかけとなっている．ここで，参加者には事前に知らされていない未知の問題に対応するため，いろいろな行列（主に係数行列の非零要素の分布に着目）に適したプログラム実行コードをあらかじめ用意しておき，実行時になって，最速と考えられる実行コードを自動選択するという手法が取られていた．また，複数の計算機（当コンテストでは4機種あった）に対応するために，同じ計算を行うにしても複数のソースコードを用意していた．

ILIBでは，次の設計思想に基づき開発が行われている．

1. ユーザが指定するパラメータが少ないこと

2. 演算カーネルに関する自動チューニング機能があること
3. 通信処理に関する自動チューニング機能があること
4. 利用するアルゴリズムに関する自動選択機能があること

そして，ILIB では次の解法が提供されている．

- ILIB_GMRES（GMRES（m）法による疎行列反復解法）
- ILIB_LU（LU 分解による密行列直接解法）
- ILIB_RLU（疎行列を帯行列と見なして LU 分解を行う）
- ILIB_DRSSED（実対称行列の標準固有値問題の解法）

さらに ILIB_DRSSED は次の 3 つに分かれる．

- ILIB_TriRed（相似変換における 3 重対角化ルーチン）
- ILIB_MGSAO（固有ベクトル計算の直交化ルーチン）
- ILIB_HouseInv（HouseHolder 逆変換ルーチン）

■ ABCLib（http://abc-lib.org/）

ABCLib（Automatically Blocking and Communication-adjustment Library）は，2002 年から当時電気通信大学にいた片桐らによって開発されてきた数値計算ライブラリである．キャッシュサイズに関連するパラメータチューニングを自動化する機能（自動ブロック化），および，通信実装方式の選択を自動化する機能（通信最適化）が実装されている．通信最適化をするため，実行時にしか判明しない入力行列の特徴を利用することから，実行時自動チューニング機能が充実している．この ABCLib の開発においては，自動チューニング機能のソースコードが約 1 万行にもなり，かつ手動でソースコードの記述を行っていたので，開発コストの増大が問題となっていた．そこで，自動チューニング機能の付加を自動化するための記述言語 ABCLibScript（本特集 4 番目の記事）が開発されたという経緯がある．ABCLib は 2009 年度，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」プロジェクトの支援のもと再整備され，名称を Xabclib に変更する．また，汎用的な自動チューニングインタフェース OpenATLib の仕様が公開される予定である．また同時に，OpenATLib を用いた，標準固有値問題を求解する LANZOS 法ライブラリ Xabclib_LANCZOS，連立 1 次方程式を求解する GMRES 法ライブラリ Xabclib_GMRES が，フリーウェアとして提供されることになっている．

自動チューニング機能付きライブラリの基盤技術

先に述べたように，線形方程式の解法，なかでも線形反復法は数値計算ライブラリの中で応用分野において最も頻繁に使用されるルーチンである．線形反復法には多数の方法が存在し，またこれらの個々の方法においてパラメータを内含することが多い（各種の反復法の詳細はたとえば，「反復法 Templates」朝倉書店に詳しい）．一方，応用分野のシミュレーションにおいてこれらの線形反復法ライブラリを利用する場合，対象としている問題に応じて，適切な解法やパラメータ値が大きく変化するという問題がある．そこで，理想的には対象とする問題に応じてこれらの解法選択やパラメータチューニングを自動的に行うライブラリが望まれる．しかし，現在の技術水準では，与えられた問題に対して多数の解法を短時間で評価する技術は確立されておらず，多数の解法に対応した自動解法選択付き反復解法ライブラリは存在していないのが実状である．一方，特定の解法に限定した場合，当該の解法に関連したパラメータ等を自動的に評価する方法がいくつか提案されており，将来の自動チューニング機能付きライブラリの基盤となる技術としてその一例を紹介する．

ICCG（Incomplete Cholesky Conjugate Gradient）法は不完全コレスキー分解を前処理とする共役勾配法である．ここで，不完全コレスキー分解とは対称行列に対する LLT 分解において元の行列の要素が 0 となっている個所は強制的に 0 として分解する方法である．不完全コレスキー分解により得られる下三角／上三角行列はそれらの行列積が元の行列と一致しないが，これを前処理行列として使用することで反復解法の収束性を加速することができる．ICCG 法は正値・対称な係数行列を持つ連立 1 次方程式の反復解法として一般的なもので，たとえば有限要素法による数値シミュレーション等で頻繁に用いられる代表的な線形反復法である．ICCG 法は他の反復法と同様に十分な近似解を得るために必要な反復回数が求解時間に大きな影響を及ぼす．ICCG 法においてこの反復回数に影響を及ぼす要因の 1 つにオーダリングがある．ここで，オーダリングとは解くべき方程式の未知変数に振られた番号のことである．この未知変数の順序を変えた場合，係数行列の形状が変化し，その結果 ICCG 法の反復回数が変化するということが生ずる．そこで，対象としている問題において適切なオーダリングを自動的に選択できれば，反復回数を低減し，求解時間を短縮することができる．そこで，京都大学の岩下らは ICCG 法における反復回数は前処理効果の程度により決まることに着目し，前処理効果を簡易に評価する P.R.I.（Precise Remainder Index）と呼ぶ評価指標を提案した．本評価指標の導出の詳細は文献 6）に詳しいが，付加的なメモリを使用することなく計算することが可能で

ある．図 -2 はインターネット上で多数の係数行列データを公開しているサイト Matrix Market（http://math.nist.gov/MatrixMarket/）のデータを用いた 50 個のランダムオーダリングにおける反復回数と評価指標の関連性を示したものである．この図に見られるように本評価指標は反復回数と高い相関を持っており，相関係数も 0.86 という値を得ている．このように，評価指標 P.R.I. を用いることにより，前処理効果におけるオーダリングの評価が可能となることが分かる．そこで，本指標を用いることにより，たとえば多色順序付け法において色数をどのように与えるか，前処理効果を考慮した上で決定することができる．このような技術を発展させることにより，適切なオーダリングを自動的に選択する自動チューニング付き ICCG ソルバライブラリを提供することが可能となると考えられる．

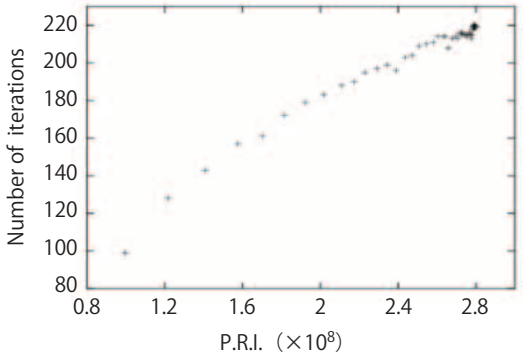

図 -2　評価指標 P.R.I. と反復回数の相関関係
（有限要素法による構造解析のデータにおいて）

自動チューニング機能付きライブラリの今後

自動チューニング機能を持つライブラリの問題点の 1 つに，自動チューニングに要する時間をいかに減らすかといったことがある．まず，パラメータ値の決定がインストール時に行えるもの，すなわち，プロセッサの特性（命令セット，レジスタの個数，キャッシュサイズ等）を利用するものであれば，インストール時にできるだけ時間をかけて性能パラメータ値を決めることで，高い性能を持ったライブラリを構成できる．あるいは，パラメータ推定範囲などの決定を行い，実行時の自動チューニングに要する時間を削減することも可能となる．

しかし，性能パラメータ値やアルゴリズムの組合せが膨大な数になったり，測定データに大きなバラつきが生じたりすることもよくある．この場合には，インストール時とはいえ膨大な時間をかけることはできないので，自動チューニングに関する数理基盤を利用して，最適な性能パラメータ値を決定することも必要になる．

次に，実行時にしか行えないパラメータ値の決定やアルゴリズムの選択をどのように行うかという問題もある．密行列同士の積，LU 分解，FFT 等では，入力データの各数値がどういう値になっていても実行時間には直接影響を及ぼさないという特徴がある．しかし，線形方程式の反復解法のように，入力行列の特性が変わると大きく実行時間に影響するものもある．実行時の自動チューニングにかかる時間はできるだけ小さい方が望ましいが，そのためには，選択するアルゴリズムの候補を利用者ごとにあらかじめ絞っておくなどの工夫が必要になると考えられる．たとえば，利用するアルゴリズムとして選択されれば大幅な高速化に繋がるものでも，適用可能かどうかの判定に時間がかかるものや，適用される可能性が低いものは，選択候補から外しておかなければならないようなことも起こり得る．これらは，利用者が解こうとしている行列問題や計算機環境によっても大きく左右されるため，利用者や計算機環境ごとに自動チューニングの選択結果とそのときの性能などをデータベース化しておき，次回からの自動選択の際に活用するなどの工夫が必要になるだろう．

参考文献

1) 小武守恒，藤井昭宏，長谷川秀彦，西田　晃：反復法ライブラリ向け 4 倍精度演算の実装と SSE2 を用いた高速化，情報処理学会論文誌：コンピューティングシステム，Vol.1, No.1, pp.73-84 (2008).
2) Blimes, J., Asanovic, K., Chin, C. -W. and Demmel, J. : Optimizing Matrix Multiply using PHiPAC : A Portable, High-performance, ANSI C Coding Methodology, Proceedings of International Conference on Supercomputing 97, pp.340-347 (1997).
3) Whaley, R. C., Petitet, A. and Dongarra, J. J. : Automated Empirical Optimization of Software and the ATLAS Project, Parallel Computing, 27 (1-2) : 3-35 (2001).
4) Whaley, R. C. : ATLAS Version 3.8 : Overview and Status, 2nd International Workshop for Automatic Performance Tuning 2007, Tokyo, Japan (2007).
5) 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ライブラリに及ぼす効果，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，Vol.42, No.SIG 12 (HPS 4), pp.60-76 (2001).
6) Iwashita, T., Nakanishi, Y. and Shimasaki, M. : Comparison Criteria for Parallel Orderings in ILU Preconditioning, SIAM Journal on Scientific Computing, Vol.26, No.4, pp.1234-1260 (2005).

（平成 21 年 4 月 9 日受付）

黒田　久泰（正会員） kuroda@cs.ehime-u.ac.jp
愛媛大学大学院理工学研究科電子情報工学専攻准教授．

直野　健（正会員） ken.naono.aw@hitachi.com
（株）日立製作所中央研究所 主任研究員．

岩下　武史（正会員） iwashita@media.kyoto-u.ac.jp
京都大学学術情報メディアセンター准教授．