5K-02

LAPACK を用いた固有値計算におけるテストシーケンスの最適化

樫村　寛大†　森崎　修司†3　片桐　孝洋‡　河合　直聡‡　永井　亨‡　星野　哲也‡
名古屋大学　情報学部コンピュータ科学科†　　名古屋大学情報基盤センター‡
名古屋大学情報学研究科†3

1. はじめに

科学技術計算において数値計算ソフトウェアのテスト実行時間の短縮は非常に重要である．本研究では典型的な数値解析ライブラリである LAPACK を取り上げ，固有値計算におけるテストシーケンスの最適化を目的とする．

本研究のテストシーケンスとは，固有値計算の解析解がわかっている問題を与え，計算結果がその範囲内に入っているかをチェックする系列である．特定のソフトウェアを前提としないテストシーケンスの最適化の研究はあるが[3]，数値計算ソフトウェアの評価はあまりない．

本研究ではまず，LAPACK のライブラリに意図的にバグを発生させ，固有値ライブラリのテスト STCollection[1]のシーケンスを入れ替える最適化によって，バグを検出するまでのテスト実行時間が短縮できるかを検討する．

2. 前提

先行研究[3]では，テストシーケンス最適化の方法として，Deja Vu ベース，Firewall ベース，Dependency ベース，Specification ベースの 4 手法が紹介されている．本研究では，数値計算分野のプログラム構造や処理の特徴から，従来のソフトウェア工学的手法との違いの検証や事例研究をすることを目的とする．

数値計算特有の処理として，以下が考えられる．(1)チェックルーチン作成時の解析解の活用；(2)プログラムの階層性．本研究で，これらの特徴をまず紹介する．

3. LAPACK とプログラム階層性

LAPACK は数値線形計算のため広く使われている数値ソフトウェアライブラリであり，線型方程式や線形最小二乗問題，固有値問題，特異値問題等を解くために利用されている．

Test Sequence Optimization for Eigenvalue Computation Using LAPACK
† Hiroto Kashimura, Department of Computer Science, School of Informatics, Nagoya University
‡ Takahiro Katagiri, Masatoshi Kawai, Toru Nagai, Tetsuya Hoshino, Information Technology Center, Nagoya University
†3Shuji Morisaki, Graduate School of Informatics, Nagoya University

LAPACK は内部で BLAS(Basic Linear Algebra Subprograms)ライブラリを呼び出し，多くの計算が BLAS で行われる実装になっている．すなわちプログラムの構造上，「階層性」が必ず存在する。

本実験ではこの BLAS の 1 つのサブルーチンである dgemm に人工的にバグを混入することで，意図的にバグを発生させる．dgemm は，式(1)の行列-行列演算を実行する．

$$C = alpha * op(A) * op(B) + beta * C \quad \cdots(1)$$

ここで $op(X)$ は $op(X)=X$, または $op(X)=X^T$ である．*alpha* と *beta* はスカラー，A, B, C は行列であり，$op(A)$ は m*k 行列，$op(B)$ k*n 行列，$op(C)$ は m*n 行列である．図 1 は，本研究で対象とする固有値計算ルーチン dgesv のコールグラフである．

dgemm の引数は *alpha*=1.0 で利用することが多いが，本研究では，dgemm ルーチン中で *alpha* の値を強制的に 0 に近づけていくことで，人工的にバグを発生させる．

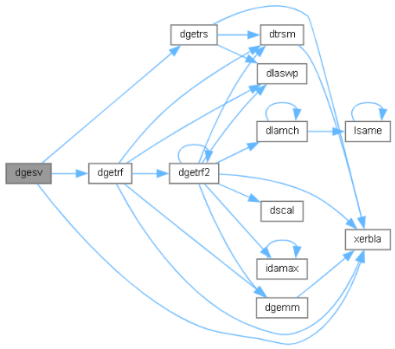

図 1 dgesv のコールグラフ（参照：LAPACK dgesv (https://www.netlib.org/lapack/)）

4. STCollection

本実験では，LAPACK の対称三重対角固有値ルーチンのテストプログラムである STCollection[1]を使用する．STCollection はテスト行列のデータセットである DATA とテストインフラストラクチャの stetester からなり，stetester には LAPACK の対称三角固有値ソルバのテストに使用された Fortran サブルーチンのセットが含まれている．

STCollection は，固有値問題のチェックルーチンに，固有値問題の解析解がわかる問題[2]を採用している．そのため，計算結果の正当性が，

演算誤差を含めて「数学的」に保証される．この特徴は，数値計算プログラムのいくつかで保障できる特性であり，この点で，一般のプログラムとは異なる．本研究では，この解析解による検証が，テストルーチンで利用できるという前提を活用する．

本実験では STCollection で用意されているデータセット test_easy.in を使用する．図 2 に，test_easy.in の一部を示す．test_easy.in を入力データとして，実行することで実行時間，残差ベクトルのノルム(RESD)，および，固有ベクトルの直交性(ORTH)を得る．この 2 つの値により，演算結果が正しいかチェック可能となる．

```
% Cases 01-04:
%
VALUES                  % Sets built-in eigenvalue distributions
       3  4             % Distribution 3, EIG(i)=COND**(-(i-1)/(N-1))
      1000 1500             % Dimensions of the matrices to be generated
%
% Cases 05-16:
%
MATRIX  T               % Sets built-in matrices
        2   3           % Matrix type 2 and 3
       2000:2005            % Dimensions of the matrices to be generated
%
% Cases 17-18:
%
GLUED                   % Sets glued matrices
     1      2     1     % If 1, set eigenvalues; if 2, set matrix
     1      2     3     % Eigenvalue distribution or matrix type
     1000   1100   1200    % Dimensions
     0.001  0.002       % Glue factors
```

図 2 test_easy.in の一部

5. 実験

5. 1 実験環境

名古屋大学情報基盤センター設置のスーパーコンピュータ「不老」（以降、スパコン「不老」）クラウドシステムを利用した（表 1）．

表 1　スパコン「不老」クラウドシステムの性能

機種名		HPE ProLiant DL 560
計算ノード	CPU	Intel Xeon Gold 6230 × 4
	メインメモリ	DDR4 384 GiB（メモリバンド幅：563.136 GB/s）
ノード数		100
総理論演算性能		537.6TFLOPS
総メモリ容量		37.5 TiB
冷却方式		空冷

5.2 実験結果

dgemm の *alpha* の値を 1.0 にした場合と 0.01 にした場合の実行結果を表 2 に示す．表 2 は case17 の分割統治法(dgesv)で固有値問題を解いた場合の結果である．表 2 より，残差ベクトルのノルム，直交性の値が極端に大きくなっており，テストによりバグが検出できることを確認できた．

表 2　Case17 における精度

Alpha 値	実行時間	RESD	ORTH
1.0（正常）	6.32	5.00×10^{-3}	1.97×10^{-2}
0.01	5.09×10^{-1}	6.82×10^{11}	1.36×10^{12}

次にテストシーケンス入替による最適化の効果を検証する．表 3 は，case17 実行時間（最適化実施例）と case17 までの実行時間（デフォルトの STCollection によるテスト実行シーケンスによりバグを検出した時間）を示す．

表 3　テストシーケンス最適化の効果

Case17 を最初に行う(a)	Case17 迄累計時間(b)	高速化率 (b/a)
73.13 [秒]	385.61 [秒]	5.27 倍

表 3 から，テストシーケンスの入替による最適化で，case17 を先頭としてテストを実行することにより 5.27 倍の高速化を達成できる可能性が明らかとなった．

6. まとめと今後の課題

本研究では LAPACK を用いて固有値計算機能のチェックルーチン STCollection のテストシーケンスの最適化可能性を検討した．dgesv で使われる下位のサブルーチン dgemm に意図的にバグを発生させる数値実験を行った．その結果，テストシーケンスを入れ替えることで 5.27 倍のテスト時間の短縮ができる可能性を示した．

今後の課題は，dgemm のバグをいれるとなぜ case17 のみでバグを検出できるのかを，プログラム構造や数値解析上の理由から分析することで，より汎用的なテスト手法を提案することがある．

謝辞

本研究は，学際大規模情報基盤共同利用・共同研究拠点（JHPCN）の支援による(課題番号: jh230005).

参考文献

[1] STCollection, https://github.com/oamarques/STCollection［閲覧日:2024 年 1 月 3 日］

[2] O. A. Marques, et al., “A Testing Infrastructure for Symmetric Tridiagonal Eigensolvers” ACM Transactions on Mathematical Software, Vol. 35, No. 1, Article 8, (July 2008).

[3] E. Engström, P. Runeson, M. Skoglund, “A Systematic Review on Regression Test Selection Techniques”, Information and Software Technology, Vol. 52, pp.14-30 (2010).