平成 26 年度 共同利用研究報告書

平成　27　年　1 月　29 日

九州大学 マス・フォア・インダストリ研究所長 殿

所属・職名　　京都大学大学院情報学研究科数理工学専攻・特定准教授

提案者 氏名(ふりがな)　木村欣司(きむらきんじ)

下記の通り共同研究の報告をいたします。　　記

		※整理番号			
1.研究計画題目	高速・高精度な特異値分解および正定値対称行列の固有値分解を実現するソフトウェアの開発				
2.種別（○で囲む）	a. 研究集会Ⅰ　b. 研究集会Ⅱ　c.短期共同研究　ⓓ.短期研究員				
3.研究代表者	氏名(ふりがな)	木村欣司(きむらきんじ)			
	所属部局名	京都大学大学院情報学研究科数理工学専攻	職名	特定准教授	
	連絡先				
	e-mail		TEL		
4.研究実施期間	平成　26 年　9 月 22 日（ 月 曜日）～平成　26　年 9　月 28　日（ 日 曜日）				

5.参加者数・参加者リスト（＊別紙「共同利用研究報告書作成上の注意」参照）

（a および b の場合、参加者数のみ記入し、集会参加者リストを添付。 c および d の場合は下記欄に記入。）

参加者数：　1　人

氏名(ふりがな)	所属	職名	氏名(ふりがな)	所属	職名
木村欣司(きむらきんじ)	京都大学大学院情報学研究科	特定准教授			

6.本研究で得られた成果の概要

はじめに, 密行列向けの計算アルゴリズムとその実装について, 得られた成果を述べる。速度優先で固有値と固有ベクトルの計算を行うソフトウェアとして, LAPACK において公開されている 2 分法と逆反復法の改良を行った。そのコードは, 改良前のコードに比べて高速であることを確認した。速度優先で特異値計算を行うソフトウェアとして, LAPACK において公開されている DQDS 法の改良を行った。そのコードは, 改良前のコードに比べて, 乱数行列のような一般的な行列においては高速であり, かつ, 相対誤差の意味での計算精度も高い。DQDS 法については, LAPACK の改良に止まらず, 独自の実装も行い, それを公開した。精度優先で特異値分解を行うソフトウェアとして, OQDS 法を実装した。そのコードを用いると, LAPACK の QR 法に比べて, 計算時間は長くなるが, 相対誤差の意味での高い精度を持つ特異値と高精度な特異値分解を達成する特異ベクトルを得られる。

次に, 疎行列向けの計算アルゴリズムとその実装について, 得られた成果を述べる。こちらについては, 速度優先で開発を行った。大次元疎行列の一部の固有対を計算できる著名なソフトウェア ARPACK に着目し, コードをマルチコア計算機の性質に特化した並列実装に変更した。改良前のコードに比べて高速であることを確認した。さらに, その改良した ARPACK のソースコードを基にして, 特異対を計算するためのソースコードも公開した。

より詳細な内容については, 添付の書類に記載する。

平成 26 年度 共同利用研究報告書
研究課題「高速・高精度な特異値分解および正定値対称行列の固有値分解を実現するソフトウェアの開発」

木村欣司 (Kinji Kimura)
京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University

1 はじめに

平成 26 年 9 月 22 日 (月) より平成 26 年 9 月 28 日 (日) まで, 九州大学 MI 研究所に滞在し共同研究を行った. その際に得られた成果について, この報告書の中で詳細に解説する.

2 成果物

以下の URL において, この共同研究によって得られた成果 (ソフトウェアのソースコード) を公開している.
http://www-is.amp.i.kyoto-u.ac.jp/kkimur/LAPROGNC/LAPROGNC.html

3 固有値分解のためのソフトウェア

実数の密対称行列の固有値と固有ベクトルを計算する一般的な方法として, (1) 密対称行列を実数の対称 3 重対角行列へ変換, (2) 実数の対称 3 重対角行列の固有値と固有ベクトルを計算, (3)(2) で得られた実数の対称 3 重対角行列の固有値と固有ベクトルから元の密対称行列の固有値と固有ベクトルを得るための逆変換の 3 つのステップを経て計算を行う手法が存在する.

3.1 2 分法と逆反復法

実数の対称 3 重対角行列の固有値と固有ベクトルを計算するための 2 分法と逆反復法について, 既存の数値計算ライブラリ LAPACK の 2 分法と逆反復法の改良を行い, それを公開した.

3.1.1 2 分法

数値計算ライブラリ LAPACK の 2 分法のサブルーチン dlaebz.f には, 並列に符号変化数を数えるコードが含まれているが, 普段, アクティブではなく, さらに, それは並列化されていない為, 実質, 有効な機能となっていない. そこで, OpenMP を用いて, 明示的に並列化し, それを公開した. 我々が公開しているコードを用いると, マルチコア計算機において, 並列化された 2 分法が実行される.

3.1.2 改良した 2 分法のコードの性能評価

実験環境は以下の通りである.

- Appro Green Blade 8000（京都大学学術情報メディアセンター）
  - CPU: Intel Xeon E5-2670 @ 2.60GHz, 16 cores (8 cores × 2)
    - L3 cache: 20MB × 2
  - RAM: DDR3-1600 64GB
  - Compiler: Intel Fortran 14.0.2
    - -O3 -xHOST -ipo-no-prec-div -mcmodel=medium -shared-intel -openmp
  - Library: Intel Math Kernel Library 11.1.2

これ以降の数値実験においては, 特に断りがない限り, すべてこの実験環境を用いる.

30000 次元の対称 3 重対角行列の全固有値を計算するための実行時間を示す.

	LAPACK の 2 分法	改良後
実行時間	449.92sec	29.27sec

3.1.3 逆反復法

固有値が密集した行列の固有ベクトルを計算する場合, 単に, 逆反復法を行うのみでは, 複数の初期ベクトルから始めた反復計算が, 同じ収束値に到達する可能性があり, さらに, 実数の密対称行列の固有ベクトルの性質である互いが直交するという性質を満足する固有ベクトルを計算することはできない. そこで, 再直交化という操作が必要になる. 再直交化の方法として, 次の 4 種類が存在する. (1) 古典グラムシュミット法 (CGS 法) による直交化, (2) 修正グラムシュミット法 (MGS 法) による直交化, (3)Givens 回転による直交化, (4)Householder 変換による直交化がある. 数値計算ライブラリ LAPACK の逆反復法のサブルーチン dstein.f は, MGS 法を採用している. しかし, 並列計算を考慮すると, CGS 法のほうが都合がよい. より正確には, 計算精度の点で, CGS 法は MGS 法に劣るため, CGS 法を 2 回繰り返す CGS2 法を利用することになる. CGS2 法は, 単に, CGS 法を 2 回繰り返す方法であるため, 以下の記述では, CGS 法を例に, 我々が行った改良について解説する.

3.2 CGS 法による逐次直交化計算

v_j を, $q_1, \cdots, q_{j-1}$ と直交化するベクトル q_j へと直交化:

$$q_j = v_j - \sum_{k=1}^{j-1} \langle q_k, v_j \rangle \, q_k, \quad 1 \leq j \leq m, m \leq n$$

上記の式は, 式変形により行列-ベクトル乗算で実装可能であり,

$$q_j = v_j - Q_{j-1} Q_{j-1}^{\top} v_j, \quad Q_{j-1} = [q_1 \ \cdots q_{j-1} \ ]$$

行列-ベクトル乗算を用いて実装を行うのが標準的な実装法である. しかし, 今回の共同研究においては, 別の実装法を採用した.

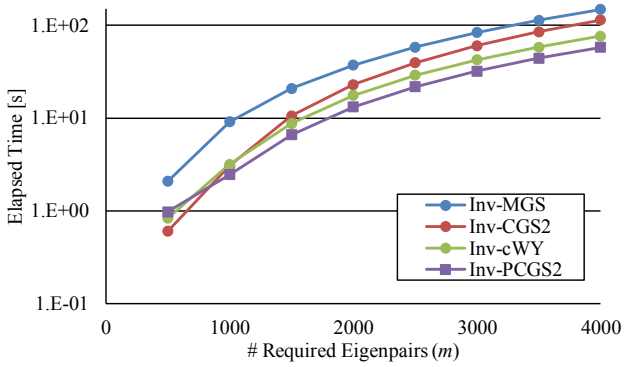
Algorithm 1 Pseudocode of `PCGS`

```
1: function PCGS(v_j, Q_{j-1})
2:     q_j = v_j
3:     #omp parallel for private(s) reduction(+:q_j)
4:     for k = 1 to j − 1 do
5:         s = −⟨q_k, v_j⟩                          ▷ Call DOT
6:         q_j = q_j + s q_k                        ▷ Call AXPY + array reduction
7:     end for
8:     #omp end parallel for
9:     return q_j
10: end function
```

我々の実装法である PCGS 法は, 各スレッドで $\boldsymbol{v}_j$, $\boldsymbol{q}_j$, $\boldsymbol{q}_k$ のみしか使用しないため, キャッシュの再利用性が高いという特徴がある. さらに, 同期は配列リダクションのときのみ行われる. なお, 実験で用いる PCGS2 法は, PCGS 法を 2 回繰り返すものである.

3.2.1 改良した逆反復法のコードの性能評価

一部の固有ベクトル計算における実行時間 (行列サイズ $n = 5000$) を示す.



- m が大きい: Inv-PCGS2 が高速
- m が小さい: Inv-CGS2 が高速

✓ $n = 10000,\ 15000,\ 20000$ でも同様の傾向 (10% で未満で性能が逆転)

今回の共同研究において公開しているソースコードには, PCGS2 法が採用されている.

3.3 ブロック逆反復法

我々は, 逆反復法よりも計算機の性能を引き出すことが容易なアルゴリズムとしてブロック逆反復法という固有ベクトルの計算法も提案しているが, 論文を執筆中であるため, 公開していない.

4 特異値分解のためのソフトウェア

実数の密行列の特異値と特異ベクトルを計算する一般的な方法として, (1) 密行列を実数の上 2 重対角行列へ変換, (2) 実数の上 2 重対角行列の特異値と特異ベクトルを計算, (3)(2) で得られた実数の上 2 重対角行列の特異値と特異ベクトルから元の密行列の特異値と特異ベクトルを得るための逆変換の 3 つのステップを経て計算を行う手法が存在する.

4.1 特異値のみを高速に計算するアルゴリズム DQDS 法

4.1.1 DQDS 法の概要

LAPACK には, 特異値のみを高速に計算するアルゴリズムとして DQDS 法が採用されている. DQDS 法の最後の S は, シフトというアルゴリズム収束を加速するための方法が採用されていることをを意味している. そのシフト計算の部分を改良したものを公開する.

長方 (正方) 形行列 Y は, 特異値を変えずに, 対角, 上副対角共に 0 でない上 2 重対角行列 $B = B^{(0)}$ に変換される (対角, 上副対角に, 0 が現れたならば, 対処する方法が存在する).

$$B = B^{(0)} = \begin{pmatrix} B^{(0)}_{1,1} & B^{(0)}_{1,2} & & 0 \\ & \ddots & \ddots & \\ & & \ddots & B^{(0)}_{N-1,N} \\ 0 & & & B^{(0)}_{N,N} \end{pmatrix}.$$

PQDS 法

$$\left(B^{(i+1)}\right)^{\top} B^{(i+1)} = B^{(i)} \left(B^{(i)}\right)^{\top} - s^{(i)} I$$

は, $B^{(i)}$ の対角成分が特異値に収束するアルゴリズムである. $s^{(i)}$ はシフト項であり, $s^{(i)} < \lambda_{\min}(B^{(i)} \left(B^{(i)}\right)^{\top})$, かつ, できるだけ最小固有値に近い値を採用すると, $(B^{(i)})_{N-1,N} \to 0$ となる収束が加速される (0 に近くなったら行列を分割する). 最小固有値のシャープな下界は,

よいシフトを与え, 特異値の計算精度の向上に貢献する.

$$(B^{(i)}_{j,j})^2 = q^{(i)}_j, j = 1, \cdots, N, (B^{(i)}_{j,j+1})^2 = e^{(i)}_j, j = 1, \cdots, N-1$$

とおく. d 変数という補助変数を導入すると, DQDS 法が得られる.

$$d^{(i+1)}_1 = q^{(i)}_1 - s^{(i)}, q^{(i+1)}_1 = d^{(i+1)}_1 + e^{(i)}_1, e^{(i+1)}_1 = e^{(i)}_1 q^{(i)}_2 / q^{(i+1)}_1, \cdots$$

4.1.2 シフト戦略

現在公開を行っているコードのシフト計算は, 以下の数理に基づいている.

1. M 次一般化 Laguerre 型下界:$A = B^{(i)}(B^{(i)})^\top$, $J_L \equiv \mathrm{Tr}(A^L)$ として,

$$\Phi_M \equiv \frac{N}{J_{-M} + \sqrt{(N-1)\left(NJ_{-2M} - J^2_{-M}\right)}} < (\lambda_{\min}(A))^M$$

が成立する. $(\Phi_M)^{1/M}$ を一般化 Laguerre 型下界と呼ぶ. $M = 1$ の場合が, Laguerre 型下界として, 代数方程式の Laguerre による解法に対応する. 数値計算では, 常に計算誤差が入るため, $NJ_{-2M} - J^2_{-M}$ が負になる場合もあり, そのような場合には, この下界を利用することはできない.

2. M 次一般化 Newton 型下界:$\Theta_M = (J_{-M})^{-\frac{1}{M}}$
$M = 1$ の場合が, Newton 型下界として, 代数方程式の Newton 法に対応する. この下界は常に用いることが可能である.

3. Kato-Temple の不等式
開区間 $(\underline{\lambda}, \overline{\lambda})$ には $N \times N$ の対称行列 A の固有値 λ が 1 個だけと ρ とが含まれているということが知られているとする. このとき

$$\rho = u^\top Au, \quad \varepsilon = ||Au - \rho u||_2, \quad \rho - \frac{\varepsilon^2}{\overline{\lambda} - \rho} \leq \lambda \leq \rho + \frac{\varepsilon^2}{\rho - \underline{\lambda}}$$

が成り立つ. A が非負定値対称行列の場合には, $\overline{\lambda}$ として, $N-1 \times N-1$ の首座小行列 A' の最小固有値の下界 $\lambda_{\min}(A')$ を利用する.

4. ゲルシュゴリンの定理:$A = (a_{i,j}), R_i \equiv \sum_{k \neq i} |a_{i,k}|$ とおくと, すべての固有値は, 複素平面において, N 個の $a_{i,i}$ を中心とする半径 R_i の円のユニオンに存在する.

4.1.3 J_{-1} と J_{-2} の計算法

$$\beta^{(i)}_1 = \frac{1}{q^{(i)}_1}, \ \beta^{(i)}_j = \frac{1}{q^{(i)}_j} + \left(\frac{e^{(i)}_{j-1}}{q^{(i)}_j}\right)\beta^{(i)}_{j-1}, j = 2, \cdots, N, \ J_{-1} = \sum_{j=1}^{N} \beta^{(i)}_j$$

$$\gamma^{(i)}_1 = \left(\beta^{(i)}_1\right)^2, \ \gamma^{(i)}_j = \left(\beta^{(i)}_j\right)^2 + \left(\frac{e^{(i)}_{j-1}}{q^{(i)}_j}\right)\left[\gamma^{(i)}_{j-1} + \left(\beta^{(i)}_{j-1}\right)^2\right], j = 2, \cdots, N, \ J_{-2} = \sum_{j=1}^{N} \gamma^{(i)}_j$$

4.1.4 CPU の丸めモードと J_{-1} と J_{-2} の計算

J_{-1} と J_{-2} の計算には, 減算が含まれない. よって, CPU の丸めモードを正の無限大の方向へ丸めるように設定することにより, 真値よりも必ず大きい値をとる J_{-1} と J_{-2} を得ることができる. この性質と M 次一般化 Newton 型下界に基づく下界を利用すると, 丸め誤差を含む計算においても, 行列 A の最小固有値の下界を常に得ることができる.

4.1.5 シフト計算を変更した DQDS 法の性能評価

1. 対角 = 1, 非対角 = 1 の上 2 重対角行列の特異値計算の実行時間

次元	LAPACK の DQDS 法	改良後の DQDS 法	独自実装の DQDS 法
10000	2.80sec	3.44sec	3.89sec
100000	248.30sec	315.24sec	359.30sec
1000000	23454.64sec	32231.97sec	36667.08sec

2. 対角 = 1, 非対角 = 1 の上 2 重対角行列の特異値計算の相対誤差の平均

次元	LAPACK の DQDS 法	改良後の DQDS 法	独自実装の DQDS 法
10000	1.29×10^{-15}	1.06×10^{-15}	1.38×10^{-15}
100000	4.50×10^{-15}	2.46×10^{-15}	3.07×10^{-15}
1000000	1.09×10^{-14}	5.36×10^{-15}	5.98×10^{-15}

3. 乱数の上 2 重対角行列の特異値計算の実行時間

次元	LAPACK の DQDS 法	改良後の DQDS 法	独自実装の DQDS 法
10000	2.40sec	2.35sec	2.44sec
100000	155.67sec	137.24sec	156.58sec

4. 乱数の上 2 重対角行列の特異値計算の相対誤差の平均 (真値として, 2 分法を用いて計算した値を採用する)

次元	LAPACK の DQDS 法	改良後の DQDS 法	独自実装の DQDS 法
10000	5.48×10^{-15}	3.08×10^{-15}	3.04×10^{-15}
100000	5.14×10^{-14}	7.90×10^{-15}	8.38×10^{-15}

4.2 特異値と特異ベクトルを高精度に計算するための OQDS 法

4.2.1 OQDS 法の概要

LU ステップ

$$G^{(k)} \begin{bmatrix} L^{(k)} \\ t^{(k-1)} I \end{bmatrix} = \begin{bmatrix} U^{(k)} \\ t^{(k)} I \end{bmatrix}, t^{(k)} = \sqrt{(t^{(k-1)})^2 + (s^{(k)})^2} \tag{1}$$

UL ステップ

$$\begin{bmatrix} I & O \\ O & (H^{(k)})^\top \end{bmatrix} \begin{bmatrix} U^{(k)} \\ t^{(k)} I \end{bmatrix} H^{(k)} = \begin{bmatrix} L^{(k+1)} \\ t^{(k)} I \end{bmatrix} \tag{2}$$

- $L^{(k)}$: 下 2 重対角行列, $U^{(k)}$: 上 2 重対角行列, $s^{(k)}$: シフト量
- $G^{(k)}$: 直交行列 (Givens 変換 $n-1$ 回と一般化 Givens 変換 n 回)
- $H^{(k)}$: 直交行列 (Givens 変換 $n-1$ 回)
- 左特異ベクトルは左からかけられた直交行列の積
- 右特異ベクトルは右からかけられた直交行列の積
- シフト量 $s^{(k)}$ のとれる範囲は $L^{(k)}$ の最小特異値未満の非負の実数

4.2.2 一般化 Givens 変換

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} \sqrt{f^2 - h^2} \\ \sqrt{g^2 + h^2} \end{bmatrix}, c^2 + s^2 = 1$$

従来の計算法・実装

- DROTG3[von Matt(1997)], 除算を多く含み, 数値的不安定性の原因となっている

提案する計算法・実装

- 通常の Givens 回転の回転行列を求める問題に帰着させる

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} f' \\ g' \end{bmatrix} \Rightarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} f \cdot f' + g \cdot g' \\ g \cdot f' - f \cdot g' \end{bmatrix} = \begin{bmatrix} f^2 + g^2 \\ 0 \end{bmatrix}, c^2 + s^2 = 1$$

4.2.3 シフト戦略

現在公開を行っているコードのシフト計算は, 以下の数理に基づいている.

1. Rutishauser シフト:

   定理

   A を, 実正定値対称 3 重対角行列とする, $b_i \neq 0$ と仮定する.

$$A = \begin{pmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & a_{n-1} & b_{n-1} \\ & & b_{n-1} & a_n \end{pmatrix}.$$

A と固有値の等しい $A' = Z^{-1}AZ$ を定義する,

$$A' = \begin{pmatrix} a_1 & b_1^2 & & \\ 1 & \ddots & \ddots & \\ & \ddots & a_{n-1} & b_{n-1}^2 \\ & & 1 & a_n \end{pmatrix},\ Z = \begin{pmatrix} 1 & & & \\ & b_1 & & \\ & & \ddots & \\ & & & \prod_{i=1}^{n-1} b_i \end{pmatrix}.$$

適当な λ^M について, $A' - \lambda^M I = LU$ 分解する,

$$L = \begin{pmatrix} \hat{q}_1 & & & \\ 1 & \ddots & & \\ & \ddots & \hat{q}_{n-1} & \\ & & 1 & \hat{q}_n \end{pmatrix}, U = \begin{pmatrix} 1 & \hat{e}_1 & & \\ & \ddots & \ddots & \\ & & 1 & \hat{e}_{n-1} \\ & & & 1 \end{pmatrix},$$

$\hat{q}_i > 0, \hat{e}_i \geq 0, i = 1, \ldots, n-1, \hat{q}_n < 0$ であるならば, $\hat{q}_n + \lambda^M < \lambda_{\min}(A)$ である.

2. Collatz シフト:
すべての成分が正の行列 A とすべての成分が正のベクトル v について,

$$\lambda_{\max}(A) \leq \max_i \frac{(A^k v)_i}{(A^{k-1}v)_i}$$

が成立する. ただし, $(x)_i$ は, ベクトル x の第 i 成分を表す. $\hat{L}^{(k)}$ は, $L^{(k)}$ の副対角成分の符号をすべて負に変更したものとする. $A = (\hat{L}^{(k)})^{-1}(\hat{L}^{(k)})^{-T}$ とすると,

$$\frac{1}{\lambda_{\min}((\hat{L}^{(k)})^T\hat{L}^{(k)})} = \lambda_{\max}((\hat{L}^{(k)})^{-1}(\hat{L}^{(k)})^{-T}) \leq \max_i \frac{(A^k v)_i}{(A^{k-1}v)_i},$$
$$\min_i \frac{(A^{k-1}v)_i}{(A^k v)_i} \leq \lambda_{\min}((\hat{L}^{(k)})^T\hat{L}^{(k)}) = \lambda_{\min}((L^{(k)})^T L^{(k)}).$$

4.2.4 OQDS 法と QR 法の比較実験

1. 対角 = 1, 非対角 = 1 の上 2 重対角行列に対する実行時間

次元	OQDS 法	QR 法
10000	837.45sec	535.30sec
20000	7711.80sec	5247.60sec
30000	29647.85sec	19297.40sec

2. 対角 = 1, 非対角 = 1 の上 2 重対角行列に対する特異値計算の相対誤差の平均

次元	OQDS 法	QR 法
10000	2.27×10^{-16}	9.69×10^{-16}
20000	2.89×10^{-16}	9.83×10^{-16}
30000	2.75×10^{-16}	9.89×10^{-16}

3. 対角 = 1, 非対角 = 1 の上 2 重対角行列に対する特異値分解の精度

次元	OQDS 法	QR 法
10000	3.44×10^{-12}	3.94×10^{-12}
20000	6.83×10^{-12}	7.70×10^{-12}
30000	1.01×10^{-11}	1.14×10^{-11}

4. 乱数の上 2 重対角行列に対する実行時間

次元	OQDS 法	QR 法
10000	656.24sec	369.43sec
20000	4485.99sec	2904.56sec
30000	13067.80sec	8341.36sec

5. 乱数の上 2 重対角行列に対する特異値計算の相対誤差の平均

次元	OQDS 法	QR 法
10000	6.54×10^{-15}	1.35×10^{-14}
20000	8.67×10^{-15}	2.21×10^{-14}
30000	1.02×10^{-14}	2.51×10^{-14}

6. 乱数の上 2 重対角行列に対する特異値分解の精度

次元	OQDS 法	QR 法
10000	1.68×10^{-12}	1.76×10^{-12}
20000	3.05×10^{-12}	3.35×10^{-12}
30000	4.32×10^{-12}	4.74×10^{-12}

5 部分固有対計算のためのソフトウェア ARPACK の改良

大次元疎行列の一部の固有対 (絶対値の大きいほうから r 個の固有値と付随する固有ベクトルなど) を計算できる代表的なアルゴリズムとして implicitly restarted Arnoldi 法というものが存在する. implicitly restarted Arnoldi 法を実装しているソフトウェア ARPACK に着目し, コードをマルチコア計算機の性質に特化した並列実装に変更する. ARPACK のベクトルの直交化計算ならびにベクトルの再直交化計算を行っている部分は, 行列-ベクトル乗算を用いた CGS2 法の実装が採用されている. それを, PCGS2 法に置き換えたコードを公開する.

5.1 implicitly restarted Arnoldi 法

1: Set m: an upper limit, and set l: the number of the desired eigenpairs
2: Input: Arnoldi decomposition $AV_m = V_m H_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^\top$

3: **for** $i := 1, 2, \ldots$ **do**

4: Compute all eigenvalues of H_m: $\lambda_1, \ldots, \lambda_m$

5: Divide eigenvalues: $\lambda_1, \ldots, \lambda_l$ and $\lambda_{l+1}, \ldots, \lambda_m$

6: Implicitly shifted QR step

$$Q^+ = Q_1 Q_2 \ldots Q_{m-l}, \quad V_m^+ = V_m Q^+, \quad H_m^+ = (Q^+)^\top H_m Q^+$$

7: $\boldsymbol{v}_{m+1}^+ := \boldsymbol{v}_{m+1}^+ / h_{m+1,m}$

8: $V_l^+ := V_m^+(:, 1:l), H_l^+ := H_m^+(1:l, 1:l)$

9: $m - l$ step Arnoldi algorithm starting with

$$AV_l^+ = V_l^+ H_l^+ + h_{l+1,l} \boldsymbol{v}_{l+1}^+ \boldsymbol{e}_l^\top$$

10: **end for**

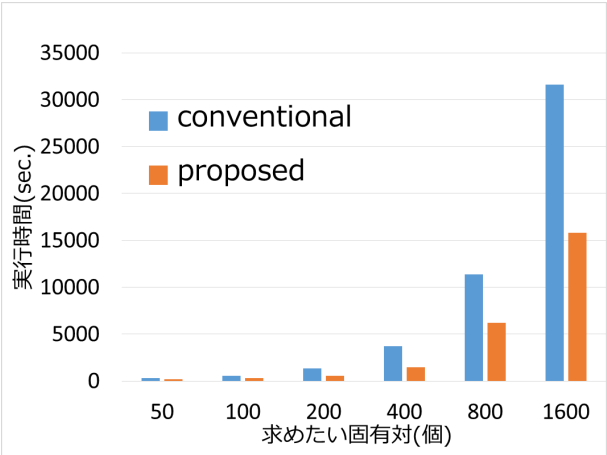
2, 9 行目の Arnoldi 分解の部分で, 直交化計算, 再直交化計算を必要とする. その部分を, 著者らが開発した PCGS2 法に置き換えた.

5.2 ARPACK のコードを修正したコードの性能評価

実験環境は以下の通りである.

- Appro Green Blade 8000 (京都大学学術情報メディアセンター)
  - CPU: Intel Xeon E5-4650L @ 2.60GHz, 32 cores (8 cores×4)
  - RAM: DDR3-1066 1.5TB
  - Compiler: Intel C++/Fortran Compiler 14.0.2
  - Options: -O3 -xHOST -ipo -no-prec-div -mcmodel=medium -shared-intel
  - Software: Intel Math Kernel Library 11.1.2
  - runtime option: numactl -i all

122500 次元の実対称行列において, 絶対値の大きいほうからいくつかの固有値と固有ベクトルを計算する.



求めたい固有対の個数に関わらず, 提案実装が高速であることが確認される.

6　部分特異対計算のためのソフトウェアの開発

$$Av = u\sigma_i, \quad A^\top u = v\sigma_i$$

を満たす正のスカラ値 σ_i を特異値, ベクトル u, v をそれぞれ, 左, 右に特異ベクトルと呼ぶ. この (σ_i, u, v) を特異対として, 値の大きいほうから r 個の特異値と付随する特異ベクトルを計算できるコードを公開した. それは, 以下の性質に基づいている.

$$A^\top Av = v\sigma_i^2, \quad ||Av||_2 = \sigma_i, \quad u = \frac{Av}{\sigma_i}$$

7　まとめ

はじめに, 密行列向けの計算アルゴリズムとその実装について, 得られた成果を述べる. 速度優先で固有値と固有ベクトルの計算を行うソフトウェアとして, LAPACK において公開されている 2 分法と逆反復法の改良を行った. そのコードは, 改良前のコードに比べて高速であることを確認した. 速度優先で特異値計算を行うソフトウェアとして, LAPACK において公開されている DQDS 法の改良を行った. そのコードは, 改良前のコードに比べて, 乱数行列のような一般的な行列においては高速であり, かつ, 相対誤差の意味での計算精度も高い. DQDS 法については, LAPACK の改良に止まらず, 独自の実装も行い, それを公開した. 精度優先で特異値分解を行うソフトウェアとして, OQDS 法を実装した. そのコードを用いると, LAPACK の QR 法に比べて, 計算時間は長くなるが, 相対誤差の意味での高い精度を持つ特異値と高精度な特異値分解を達成する特異ベクトルを得られる.

次に, 疎行列向けの計算アルゴリズムとその実装について, 得られた成果を述べる. こちらについては, 速度優先で開発を行った. 大次元疎行列の一部の固有対を計算できる著名なソフトウェア ARPACK に着目し, コードをマルチコア計算機の性質に特化した並列実装に変更した. 改良前のコードに比べて高速であることを確認した. さらに, その改良した ARPACK のソースコードを基にして, 特異対を計算するためのソースコードも公開した.

8　今後の課題

目的と期待される成果の中の「さらに, ユーザーの利便性を意識し, マニュアルを充実させる. 加えて, ユーザーインターフェースの構築も行う. 」を共同研究の時間内に達成することができなかった為, 今後の課題とし, ソースコードを公開している web ページに随時アップロードする予定である.