Vol.2018-HPC-164 No.2
2018/5/7

T-S 行列の疎性を保つ直交分解の計算とその反復改良について

村上 弘[1,a)]

概要：いま A は極めて縦に細長い行列であるとする．そのとき A の QR 分解 $A = QR$ あるいは特異値分解 $A = Q\Sigma W^T$ では，A が疎でも分解で得られる Q は一般には密になる．そのため A が大規模な疎行列で，その疎性を用いて記憶量を節約することで格納を可能にしている場合には，記憶量の面から密になる Q を陽な形で保持することは困難である．そこで元の行列 A を保持し，Q は陰的に表現して保持しないようにすることで Q の格納に必要な記憶量を削減する．また大規模問題の場合には Q の列直交性が丸め誤差の累積の影響により悪化するが，それに対する改良処理を数回程度を限度として適応的に反復することを試みる．

キーワード：縦長行列，特異値分解，疎行列，陰的な表現，記憶量

1. はじめに

いま $m \times n$ 行列 A は $m \gg n$ で，極めて縦に細長い（tall skinny）形状であるとする．そのとき特異値分解 $A \to Q\Sigma W^T$（**図 1**）を以下の方法を用いて求める．

- まず $C \leftarrow A^T A$ を計算し，固有値分解 $C \to UDU^T$ を求めて，$A' \leftarrow AU$ を作る（**図 2**，**図 3**）．
- すると $A'^T A' = D$ であるから，行列 A' は列直交である．それゆえ $Q \leftarrow A'(\sqrt{D})^\dagger$ は列正規直交である（右肩のダガー † は一般化逆を表わす）．
- こうして，特異値分解 $A \to Q\,\Sigma\,W^T$ が得られる（図 1）．ただし $W \leftarrow U$，$\Sigma \leftarrow \sqrt{D}$ であり，$Q \leftarrow A'\,\Sigma^\dagger$ である．

行列 A が密であれば演算のほとんどを記憶参照の局所性が高い BLAS3 が占めるので効率の良い計算ができる．

しかし**数値計算**では（m が大きい場合には特に）計算 $C \leftarrow A^T A$ の過程で累積した丸め誤差の影響により A' の列直交性が崩れている可能性がある．そこで $C' \leftarrow A'^T A'$ を計算する（それは数学的には D に等しいはずである）．そうして C' が**ほとんど対角である**が成立しなければ，A' の列直交性はまだ不完全なので，今度は C' の固有値分解 $C' \to U'D'U'^T$ を求めて，$A'' \leftarrow A'U'$ を作ると，A'' は A' よりも列直交性が改良されている（と期待できる）．

そうならば $Q \leftarrow A''(\sqrt{D'})^\dagger$ の列直交性は改良されて，

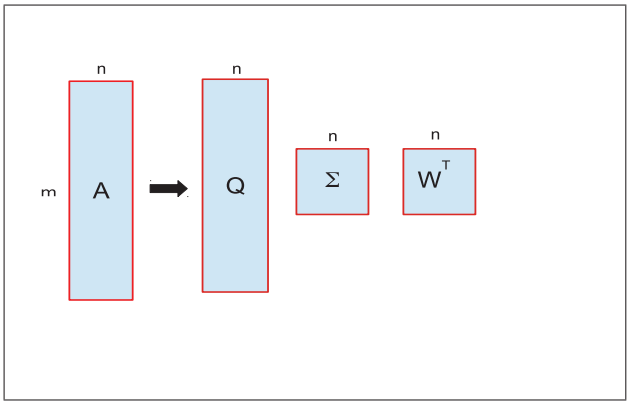


図 1 極めて縦に長い（$m \gg n$）行列 A の特異値分解

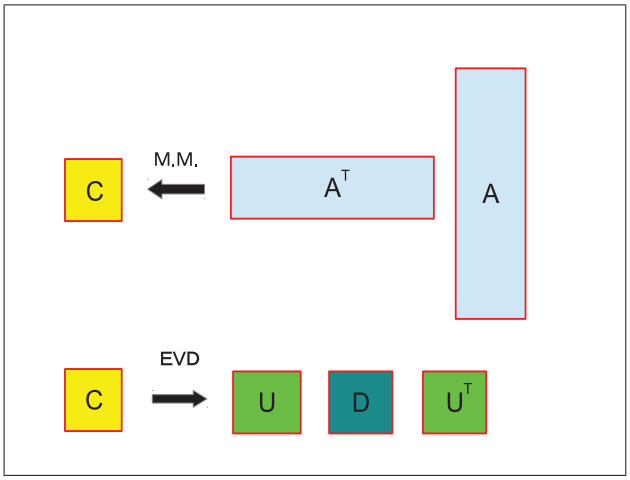


図 2 $C \leftarrow A^T A$ の計算と固有値分解 $C \to UDU^T$

[1] 首都大学東京・数理科学専攻
Department of Mathematical Sciences, Tokyo Metropolitan University
[a)] mrkmhrsh@tmu.ac.jp

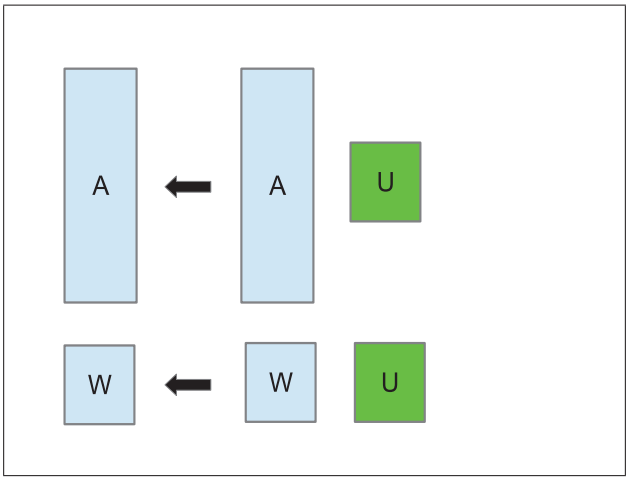


図 3 A および W の右側からの直交行列 U による回転

特異値分解は $A \to Q\Sigma W^T$ となる．ここで $W \leftarrow UU'$，$\Sigma \leftarrow \sqrt{D'}$ であり，$Q \leftarrow A''\Sigma^\dagger$ である．

もしも A'' の列直交性がまだ不十分であれば，同様に改良のための処理を繰り返す．

2. 極めて縦長の密行列に対する陽的な SVD

元の A が密行列の場合には，A と同じ場所に途中の A' や A'' および Q を陽に表現して重ね書きする．元の A の値は失われる．この方法の擬似コードを以下の**リスト 1** に示す．

リスト 1：極めて縦長の密行列 A の陽的な SVD

```
mxit ← 3 /* 最大反復数を 3 に */
W ← I /* 次数 n の単位行列 */
for it を 1 から mxit まで do begin
  C ← A^T A /* カーネル 1 */
  もしも it > 1 で C がほぼ対角 なら繰返しから脱出
  C → U D U^T /* 固有値分解（固有値の降順）*/
  W ← WU /* 小次数 n の回転の蓄積 */
  A ← AU  /* カーネル 2 */
end
Σ ← √D /* 注：D の負の対角は零に置換 */
/* 相対精度 ε での有効階数 r を決める */
r ← 0
もしも σ_1 = 0 なら処理終了/* A が零行列の場合 */
r ← 1
for k を 2 から n まで do begin
    もしも σ_k < ε σ_1 なら繰返しから脱出.
    r ← k
end do
for k を 1 から r まで do /* Q の第 k 列を作る*/
    A の第 k 列に 1/σ_k を乗じる.
```

ここで**カーネル 1** の計算 $C \leftarrow A^T A$ は，A を縦方向にブロック幅 b で区切ってブロックごとに行なう．第 p ブロックの計算 $C^{(p)} \leftarrow {A^{(p)}}^T A^{(p)}$ には BLAS3 の `xSYRK` を用いて，$C^{(p)}$ の累和で C を作る．

カーネル 2 の計算 $A \leftarrow AU$ も，A を縦方向にブロック幅 b で区切ってブロックごとに行なう．第 p ブロックに対する計算 $A^{(p)} \leftarrow A^{(p)}U$ には BLAS3 の `xGEMM` を用いる．

反復の停止には A の列直交性を C がほとんど対角であるかどうかを調べて判断する．それに用いる条件は，すべての非対角要素 $c_{i,j}$（$i < j$）に対して $c_{i,j}^2 \le \epsilon_{\rm mac}\, c_{i,i}\, c_{j,j}$ であるとした（相対比が閾値 ε 以下の特異値は切断されるので，そのことを考慮すればこの条件は弱めることができるであろう）．

3. 極めて縦長の疎行列に対する陰的な SVD

いま縦長の $m \times n$ 行列 A（ただし $m \gg n$）は疎行列で，主記憶上にその非零要素とその位置の情報だけがたとえば CRS 形式（Compressed Row Storage format）を用いて格納されているとする．特異値分解 $A \to Q\Sigma W^T$ で得られる行列 Q は行列 A が極めて疎であっても一般には密になる．しかし行列 A を書き換えないで特異値分解に現れる小次数の行列 Σ と W を構成するなら，関係 $Q = AW\Sigma^\dagger$ を用いて Q を陰的に表現すれば，密な Q を陽な形で保持せずに済むのでその分だけ記憶量が減らせる．

陰的な Q の表現からは，各特異値に対する Q の列（A の特異ベクトル）を必要に応じて容易に構成できる．また他の行列と Q の積を求める場合も，Q を陽に構成せずに陰的な表現を用いれば A の疎性が利用できて効率の良い計算ができる．

極めて縦長な疎行列 A に対する陰的な SVD の方法の擬似コードを以下の**リスト 2** に示す．

リスト 2：極めて縦長の疎行列 A の陰的な SVD

```
mxit ← 3 /* 最大反復数を 3 に */
W ← I  /* 次数 n の単位行列 */
for it を 1 から mxit まで do begin
  C←(AW)^T(AW) /*カーネル 3 .it=1 なら C←A^T A
*/
  もしも it > 1 で C がほぼ対角 なら繰返しから脱出
  C → U D U^T /* 固有値分解（固有値の降順）*/
  W ← WU   /* 直交回転の蓄積 */
end
Σ ← √D /* 注：D の負の対角は零で置換 */
/* 相対精度 ε での有効階数 r を求める */
ここから先は「リスト 1」と同様なので省略.
```

行列 A が疎でも $A' = AU$ は一般には密になるので，A' は作らずに，その代わりに直交変換 U を W に蓄積する．

カーネル 3 の処理 $C \leftarrow (AW)^T(AW)$ では，作業中の記憶量を抑えるために，密になる AW の全体を作ることは避ける．具体的には**図 4** が示すように，まず行列 A を縦方向にブロック分割して，各 p 番目のブロックのそれぞれに対して一時期には AW の一部分である $B^{(p)} \leftarrow A^{(p)}W$ だけを $A^{(p)}$ の疎性を利用して作る．そうして $C^{(p)} = {B^{(p)}}^T B^{(p)}$ を（BLAS3 の `xSYRK` を用いて）作り，C への加算 $C \leftarrow C + C^{(p)}$ を終えたら直ちに $B^{(p)}$ も $C^{(p)}$ も棄てる．この作業で用いる行列 $B^{(p)}$ と $C^{(p)}$ に対する記憶量は少なくて済む．複数ブロックに対する計算の大部分は（C への累和を行なう箇所を除けば）複数のスレッドを用いて並列に処理できる．

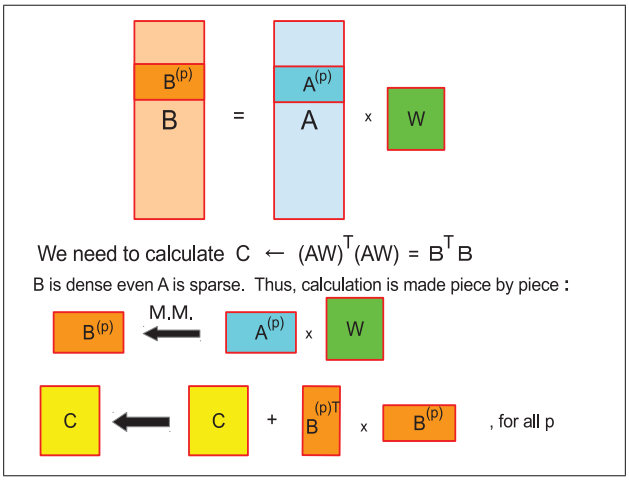


図 4 カーネル 3：$C \leftarrow A^T A$ のブロック単位での計算

行列 A が疎でも行列 Q は一般には密になるが，この計算方法では A を変更することなく特異値分解 $A \to Q\Sigma W^T$ に現れる Σ と W が求まる．すると関係 $Q = AW\Sigma^\dagger$ を用いて陰的に Q を表わすことができる（**図 5**）．なお行列 A が密の場合でも，この陰的な特異値分解の方法を適用すれば A は変更を受けず，また密行列 Q を保持する記憶場所を確保せずに済む．

4. 陰的な SVD を構成した後の処理

4.1 Q の一部あるいは全部の列を陽な形で構成すること

陰的な特異値分解を行って得られた陰的な Q の表現から，Q の列ベクトル（A の特異ベクトル）の一部あるいは全部を，陽的な形で必要に応じて構成することは容易にできる．しかし陽的な形に構成した Q の列ベクトルを保持する記憶量を確保する必要がある．もしも一時にはごく少数の列だけが参照できれば十分で，多数の列を同時期に揃える必要はなければ，ごく少数の列を陽な形で保持できる記憶場所を確保してそれを使い回すことで，必要な記憶量の増加を抑えることができる．

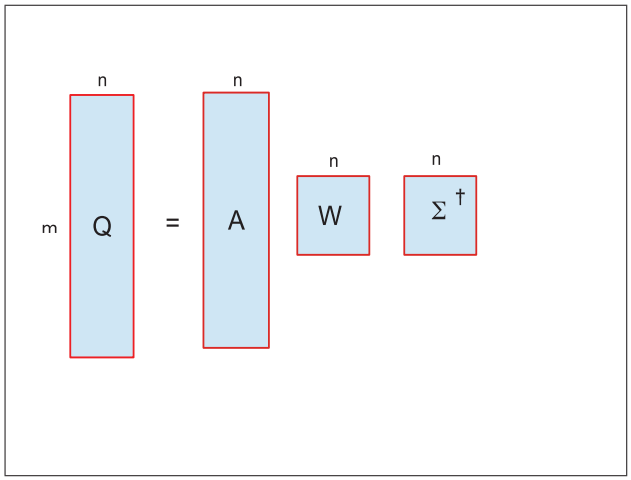


図 5 陰的な Q の表現

4.2 陰的に表現された Q と他の行列との積計算

行列 Q に任意の行列を右あるいは左から乗じたものを構成する場合も，Q を陽な形で作らずに Q の陰的表現を用いて行列 A の疎性を利用することにより必要な演算量を減らすことができる．

（たとえばいま A と Q が $m \times n$ 行列で，W は n 次行列で，Σ は n 次の対角行列とする．X を $n \times \ell$ 行列とすると Q と X の積は $QX = A(W\Sigma^\dagger X)$ であるから，まず $X' = W\Sigma^\dagger X$ を作り，次に A の疎性を用いて AX' を計算すれば非零要素が少なければそれだけ少ない演算回数で効率良く計算ができることがわかる．

またいま Y が $\ell \times m$ 行列であるとき，Y と Q の積は $YQ = (YA)(W\Sigma^\dagger)$ であるから，A の疎性を用いてまず $Y' = YA$ を計算して，次に $Y'(W\Sigma^\dagger)$ を計算すれば A の非零要素が少なければそれだけ少ない演算回数で効率良く計算ができることがわかる．実際，いま A の非零要素の割合を s として古典的な行列積計算法により YQ を計算すると ℓmn 回の乗算と加算が必要である．他方で A の疎性を用いて $Y' = YA$ を古典的に計算すると $s\ell mn$ 回の乗算と加算が必要であり，その次に $Y'(W\Sigma^\dagger)$ の計算にはほぼ ℓn^2 回の乗算と加算が必要である．つまり ℓmn と $s\ell mn + \ell n^2$ の比較になるが，その比は $1 : (s + n/m)$ であり，いま $m \gg n$ であるから，A の疎性を利用しない方法に比べて疎性を利用する計算法の演算量はほぼ s 倍であり，非零要素の割合 s に比例して演算量が減ることがわかる．）

5. 実験

5.1 計算機システムの仕様

実験に用いた計算機システムの仕様を述べる．CPU はインテル Core i7-5960X （8 コア，クロック 3.0GHz，L3 キャッシュ 20MB，AVX2 演算）である．CPU のハイパースレッドとターボブーストの機能は BIOS の設定によりオフにしてある．主記憶はクアド・チャンネルのメモリバスに DDR4-17000 の 16GB モジュールを 8 個を用いてお

り，合計容量は 128GB である．コンパイラにはインテル Fortran v15.0.0 を使用し，コンパイルオプションとして `-fast -openmp` を指定して，OpenMP により最大 8 スレッドで並列計算を行なった．なおプログラムの中で OpenMP の各スレッドから呼び出す BLAS3 のルーチンには，インテル MKL v11.2 に含まれるシングルスレッド版のものを用いた．

5.2 実験内容の説明

計算には前述の**リスト 2** に示した方法を用いた．縦長 $m \times n$ 行列 A を書き換えずに，陰的な Q の表現を構成する．なお小さい n 次の行列 C の固有値分解には精度の面からは Jacobi 法を用いるのが良く [2]，実験には Rutishauser の Jacobi 法 [1] を使用した．

実験例の行列 A は非零要素の割合を指定し，行列内での非零要素の位置の分布は一様でランダムとした．また非零要素の値も乱数を用いて設定した．

行列 A を疎行列として扱う場合には，非零要素の位置と値を CRS（Compressed Row Storage）形式を用いて格納した．密行列として扱う場合には，通常の Fortran の二次元配列 `DIMENSION(m,n)` の形式を用いて格納した．

計算は OpenMP を用いてスレッド並列化した．用いたスレッド数を `nth` と表す．図 2 や図 3 の処理も，行列 A を縦方向にブロック幅 b で区切り，各スレッドがなるべく独立してブロック単位の処理を並行に行なうようにした．各スレッドの内部で用いた BLAS3 のルーチンは，シングルスレッド用のものである．最大値に対する相対比が ε 以下の特異値は切断して計算した．その切断により決定された行列 A の実効階数を r とする．

5.3 非零要素の割合と陰的な SVD と Q の列を求めた経過時間

5.3.1 m=10^7，n=100 の場合

行列 A のサイズが $m = 10^7$（一千万）で $n = 100$ とした場合に，スレッド数が `nth` $= 1$ と `nth` $= 8$ の二通りの場合について，陰的な SVD の構成とそれを用いて Q の列の一部あるいは全部を求める処理のそれぞれの経過時間の表（**表 1**，**表 2**）と陰的 SVD の経過時間のグラフ（**図 6**，**図 7**）を示す．

表 1 陰的な SVD と Q の列を ℓ 個求めた経過時間（単位：秒）
（m=10^7，n=100，`nth`=1）（b=100，ε=10^{-12}，r=100）

非零の割合	SVD	ℓ=1	ℓ=5	ℓ=10	ℓ=100
1%	6.93	0.0882	0.255	0.392	2.90
3%	8.26	0.116	0.386	0.625	4.65
10%	13.2	0.173	0.666	1.17	9.88
30%	25.2	0.376	1.37	2.55	23.5
100%	86.9	1.02	3.29	6.12	56.9
密の扱い	19.9	1.70	2.69	3.08	9.02

表 2 陰的な SVD と Q の列を ℓ 個を求めた経過時間（単位：秒）
（m=10^7，n=100，`nth` = 8）（b=100，ε=10^{-12}，r=100）

非零の割合	SVD	ℓ=1	ℓ=5	ℓ=10	ℓ=100
1%	0.930	0.0113	0.0322	0.0496	0.421
3%	1.10	0.0155	0.0488	0.0789	0.645
10%	1.74	0.0310	0.0849	0.148	1.30
30%	3.25	0.0794	0.177	0.323	3.01
100%	11.0	0.240	0.433	0.782	7.20
密の扱い	2.59	0.244	0.374	0.421	1.21

図 6 非零要素の割合に対する陰的な SVD の経過時間（単位：秒）
（m=10^7, n=100，`nth`=1）

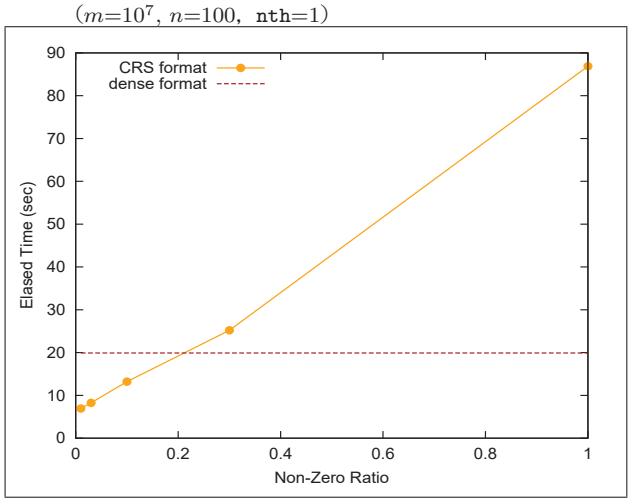
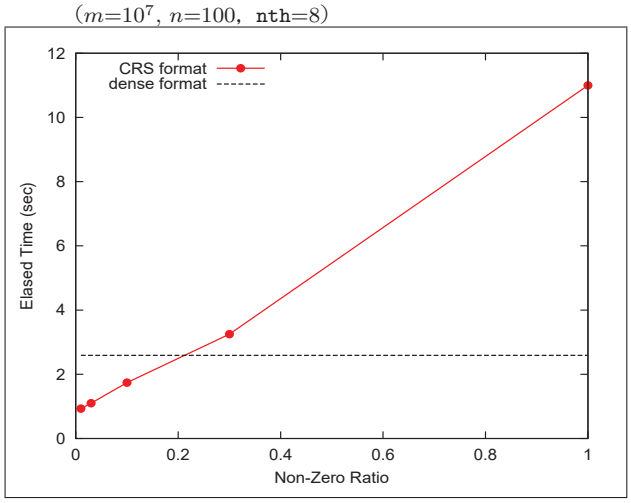


図 7 非零要素の割合に対する陰的な SVD の経過時間（単位：秒）
（m=10^7, n=100，`nth`=8）



5.3.2 m=10^7，n=300 の場合

同様に行列 A のサイズを $m = 10^7$（一千万）と $n = 300$ に変えた場合についても，スレッド数が nth = 1 と nth = 8 の二通りの場合についてそれぞれ経過時間の表（**表 3**，**表 4**）と陰的 SVD の経過時間のグラフ（**図 8**，**図 9**）を示す．

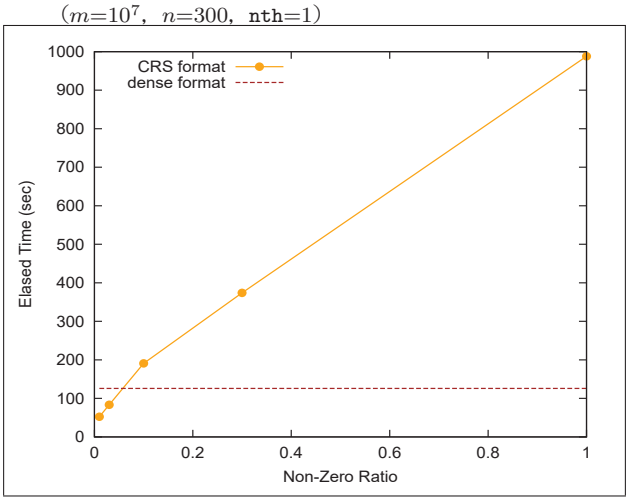
表 3 陰的な SVD と Q の列を ℓ 個を求めた経過時間（単位：秒）（m=10^7，n=300，nth=1）（b=100，ε=10^{-12}，r=300）

非零の割合	SVD	ℓ=1	ℓ=5	ℓ=10	ℓ=300
1%	52.5	0.116	0.385	0.625	15.4
3%	83.4	0.165	0.628	1.10	28.5
10%	191	0.378	1.38	2.55	71.1
30%	374	0.970	3.25	6.02	167
100%	988	3.02	10.2	19.2	540
密の扱い	126	3.24	6.35	7.10	58.8

表 4 陰的な SVD と Q の列を ℓ 個を求めた経過時間（単位：秒）（m=10^7，n=300，nth=8）（b=100，ε=10^{-12}，r=300）

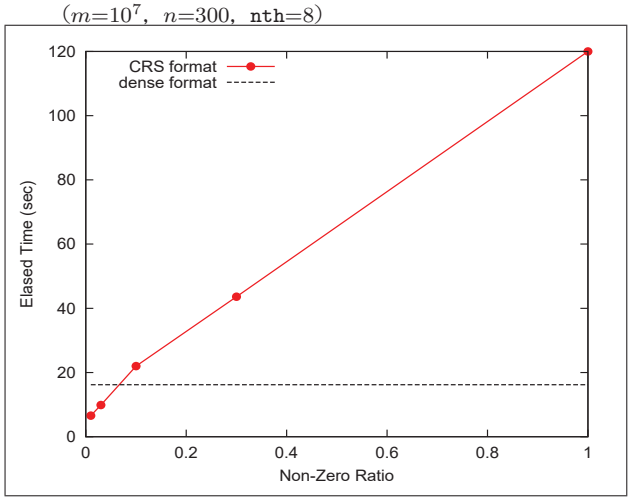
非零の割合	SVD	ℓ=1	ℓ=5	ℓ=10	ℓ=300
1%	6.58	0.0156	0.0487	0.0790	2.00
3%	9.89	0.0273	0.0799	0.139	3.65
10%	22.0	0.0835	0.180	0.323	8.98
30%	43.6	0.219	0.423	0.766	21.0
100%	120	0.701	1.28	2.26	61.8
密の扱い	16.2	0.576	0.867	0.952	7.21

図 8 非零要素の割合に対する陰的な SVD の経過時間（単位：秒）（m=10^7，n=300，nth=1）



行列の非零要素の割合が小さければ陰的な SVD の計算の経過時間が減少することがグラフからわかる．いずれのグラフでも陰的な SVD を行列 A を密行列として扱って求めた場合の処理の経過時間を水平な点線を用いて示している．密行列に対しては BLAS3 を用いると非常に効率の良い実装ができるので，（現時点でのコードでは）$n = 100$ の場合には非零要素の割合が 30%以上であれば，また $n = 300$ の場合には割合が 10%以上であれば，それぞれ A を密行列として扱って BLAS3 を用いた場合の方が疎行列として扱かって処理する場合よりもむしろ経過時間が短縮できていることがわかる．

図 9 非零要素の割合に対する陰的な SVD の経過時間（単位：秒）（m=10^7，n=300，nth=8）



各表の中では，陰的な特異値分解を既に求めた後に，行列 Q の列ベクトルを陽な形でその一部または全部を構成するのに掛かった経過時間も示している．陽な形で Q の列ベクトルを ℓ 個構成すると，それらの格納には新たに ℓ に比例する記憶量（$\ell \times m$ 語）が必要になる．そうして個数 ℓ が多ければそれだけ構成に掛かる経過時間も増加する．

5.4 陰的な SVD の経過時間とスレッド数

5.4.1 m=10^7 で，n=100 あるいは n=300 の場合

$m \times n$ 行列 A を m を 10^7 として n は 100 あるいは 300 とした二通りの場合について，行列 A の非零要素の割合を 10%として，陽な形で Q を作らない陰的な SVD の処理の経過時間をスレッド数 nth を 1 から 8 までについて測定した結果を**表 5** と**表 6** に示す．**図 10** と**図 11** のグラフはそれらの結果について，スレッド数 nth を横軸にとり，経過時間の逆数を縦軸にとってプロットしたものである（スレッド数に正比例する理想的な場合をグラフ中の黒い点線で示している）．スレッド数が 8 以下での結果ではあるが，このグラフから経過時間の逆数（つまり処理性能）はスレッド数にほぼ正比例していることがわかる（もしも行列 A の非零要素の位置が今回の実験のようにランダムな一様分布ではなくて著しい集中や偏りを持つ分布である場合には，高い性能を得るためには A の縦方向のブロック分割の幅を一定にせずに分布に応じて調整するなどの対応が必要であろう）．

5.4.2 m=10^8 で，n=100 あるいは n=300 の場合

また同様に，m を 10^8（一億）として n を 100 あるいは 300 とした二通りの場合について，行列 A の非零要素の割合を 3%として，陰的な SVD の処理の経過時間をスレッド数 nth を 1 から 8 までについて測定した結果を**表 7** と**表 8**

表 5 陰的な SVD のスレッド数に対する経過時間（単位：秒）
（m=10^7，n=100，非零要素の割合 10%，b=100）

スレッド数 nth	経過時間
1	13.2
2	6.67
3	4.53
4	3.39
5	2.76
6	2.29
7	1.96
8	1.74

表 6 陰的な SVD のスレッド数に対する経過時間（単位：秒）
（m=10^7，n=300，非零要素の割合 10%，b=100）

スレッド数 nth	経過時間
1	191
2	88.6
3	57.4
4	42.9
5	34.6
6	29.1
7	25.3
8	22.0

図 10 陰的な SVD のスレッド数に対する経過時間の逆数
（単位：1/秒）（m=10^7，n=100，非零要素の割合 10%）

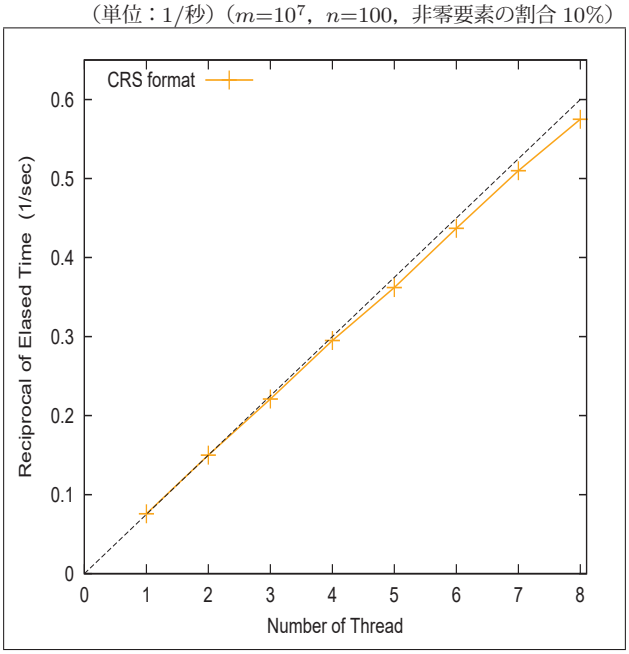


図 11 陰的な SVD のスレッド数に対する経過時間の逆数
（単位：1/秒）（m=10^7，n=300，非零要素の割合 10%）

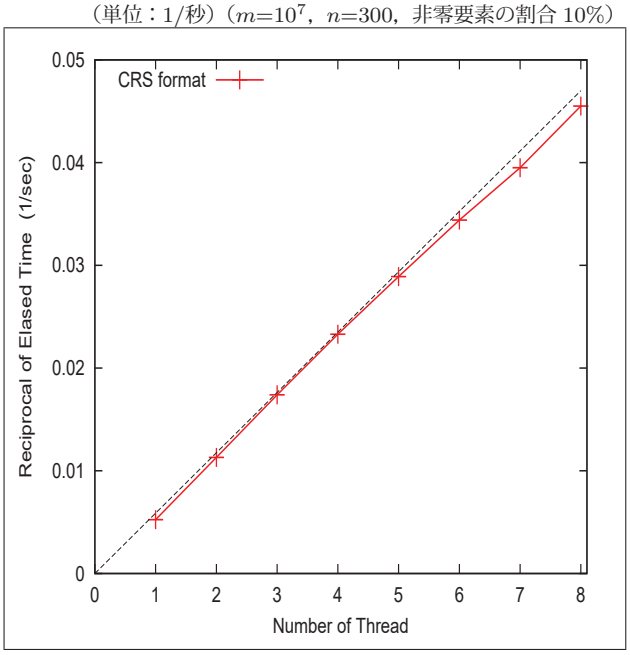


表 7 陰的な SVD のスレッド数に対する経過時間（単位：秒）
（m=10^8，n=100，非零要素の割合 3%，b=100）

スレッド数	経過時間
1	83.2
2	41.8
3	28.0
4	21.0
5	16.8
6	14.0
7	12.1
8	10.5

表 8 陰的な SVD のスレッド数に対する経過時間（単位：秒）
（m=10^8，n=300，非零要素の割合 3%，b=100）

スレッド数 nth	経過時間
1	831
2	413
3	259
4	191
5	153
6	128
7	110
8	95.8

に示す．グラフ**図 12** と**図 13** はそれらの結果について，スレッド数を横軸にとり，経過時間の逆数を縦軸にとってプロットしたものである（スレッド数に正比例する理想的な場合をグラフ中の黒い点線で示している）．スレッド数が 8 以下での結果ではあるが，このグラフから経過時間の逆数がスレッド数にほぼ正比例していることがわかる．

この場合，要素が 64bit 倍精度浮動小数点数である $m \times n$ 行列を通常の密行列の形式で 1 個保持するのに必要な記憶量は，n が 100 の場合には 80Gbyte であり，n が 300 の場合には 240Gbyte である．実験に用いた計算機の主記憶の容量は 128Gbyte なので，密行列の形式を用いたのでは n が 300 の場合には行列 1 個分たとえば Q を陽な形では主記憶に収められないわけである．ところが A を疎行列形式で表し，Q は陰的に表すことで，A の非零要素の割合が少なければ（今の場合 3%）それだけ要求する記憶の量が減って主記憶の容量以下に納まったために計算が実行できた．

図 12 陰的な SVD のスレッド数に対する経過時間の逆数（単位：1/秒）（m=10^8, n=100，非零要素の割合 3%）

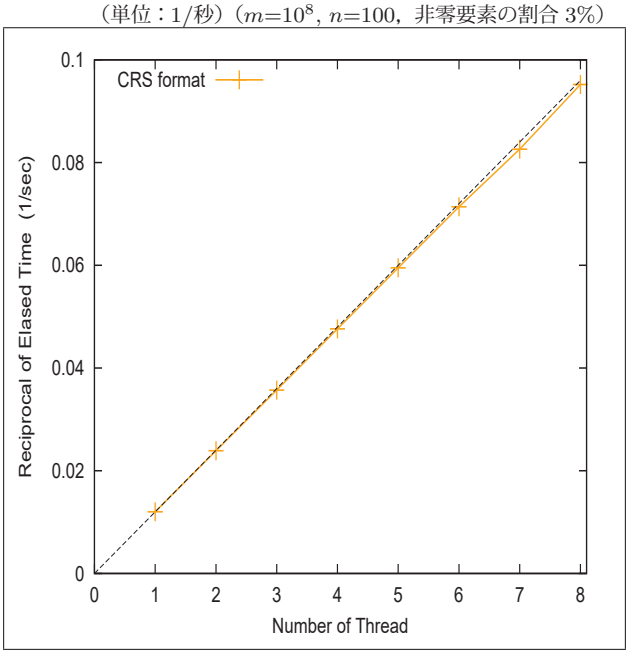
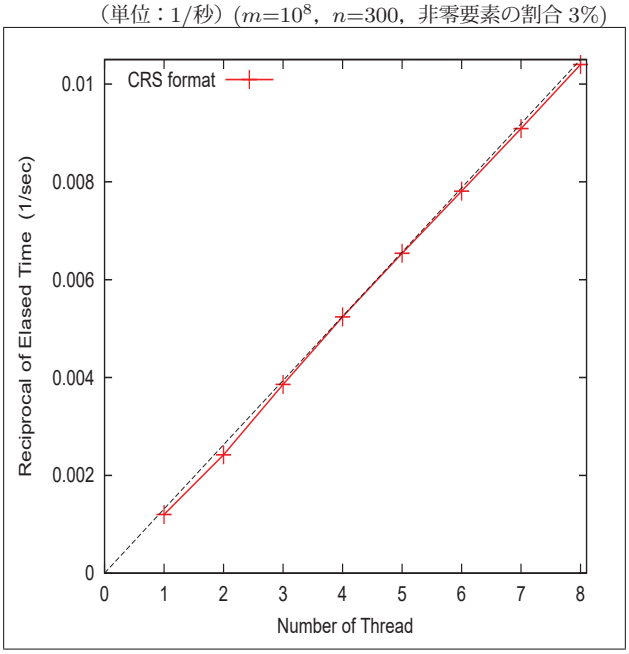


図 13 陰的な SVD のスレッド数に対する経過時間の逆数（単位：1/秒）（m=10^8，n=300，非零要素の割合 3%）



6. おわりに

極めて縦に細長い $m \times n$ 行列 A（$m \gg n$）の特異値分解 $A \rightarrow Q\Sigma W^T$ を構成する一つの計算法について調べた．大規模で m の大きい問題の場合には，計算 $C \leftarrow A^T A$ の際の丸め誤差の累積の影響により Q の列直交性が悪くなるので，それを改良する処理を適応的に繰り返すことにする（ただし，今回の実験の乱数を用いて構成された行列はどれも実際には改良の処理が必要なかった．これについては，今後に数値的な性質の悪い大規模な疎行列を作る方法を調べて適用し，実験をしてみる必要がある）．

行列 A が疎でも行列 Q は一般には密であるが，A を変更せずに小さい行列 W と対角行列 Σ だけを構成すれば，関係 $Q = AW\Sigma^{\dagger}$ を用いて Q を陰的に表現することにより Q 全体を格納する記憶を確保する必要はなくなる．陰的な Q の表現からは Q の列の全体あるいは一部を簡単な計算により陽な形で構成することができる．

この「極めて縦に細長い疎行列に対する陰的な特異値分解」の計算をいくつかの例について実験し，その結果を紹介した．元の疎行列 A を変更せずに，その疎性を用いて計算量を抑え，Q を陽には作らずに陰的な表現を用いることで，必要な記憶量を大幅に削減することが可能であることを示した．

参考文献

[1] Heinz Rutishauser:“The Jacobi method for real symmetric matrices”, *Numer. Math.*,Vol.9,No.1(1966),pp.1–10.

[2] James Demmel and Krešimir Veselić: “Jacobi’s method is more accurate than QR”, *SIAM J. Matrix Anal. & Appl.*,Vol.13,No.4(1992),pp.1204–1245.

[3] Andreas Stathopoulos and Kesheng Wu: “A block orthogonalization procedure with constant synchronization requirements”, *SIAM J. Sci. Comput.*,Vol.23,No.6(2002),pp.2165–2182.

[4] 村上弘:「非常に細長い大規模行列に対する記憶参照局所性が高い正規直交化法の実験」, **日本応用数理学会論文誌** ,Vol.17,No.4(2007),pp.399–454.

[5] James Demmel,Laura Grigori,Mark Hoemmen and Julien Langou: “Communication-optimal parallel and sequential QR and LU factorizations”, *SIAM J. Sci. Comput.*,Vol.34,No.1(2012),pp.A206–A239.

[6] 村上弘:「T-S 疎行列に対する特異値分解の直交性を適応的に反復補正する計算法」, **第** *44* **回数値解析シンポジウム** *NAS2015* **講演予稿集** (2015 年 6 月),pp.41–44.