Rank Revealing Algorithms and its Applications

Yujia Bao

Department of Mathematics
Shanghai Jiao Tong University

Thesis Defense, 2016

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Notations

Given a matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ with $m \geq n$. Organize the singular values $\sigma_i(\boldsymbol{M})$ as following

$$\sigma_1(\boldsymbol{M}) \geq \sigma_2(\boldsymbol{M}) \geq \cdots \geq \sigma_n(\boldsymbol{M})$$

Rank Revealing QR Factorization

Definition (RRQR factorization)

Given a matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ with $m \geq n$ and an integer k. Let $\boldsymbol{M}\boldsymbol{\Pi} = \boldsymbol{Q}\boldsymbol{R}$ be the QR factorization of $\boldsymbol{M}$ with its columns permuted according to the permutation matrix $\boldsymbol{\Pi}$. Partition $\boldsymbol{R}$ as

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{R}_{22} \end{bmatrix},$$

where $\boldsymbol{R}_{11} \in \mathbb{R}^{k \times k}$ is an upper triangular matrix. RRQR factorization aims to choose $\boldsymbol{\Pi}$ such that

$$\sigma_{\min}(\boldsymbol{R}_{11}) \approx \sigma_k(\boldsymbol{M}) \quad \text{or} \quad \sigma_{\max}(\boldsymbol{R}_{22}) \approx \sigma_{k+1}(\boldsymbol{M})$$

or both holds simultaneously.

Rank Revealing QR Factorization

The RRQR factorization problem can be formulated as

Problem-I: $\max_{\mathbf{\Pi}} \sigma_{\min}(\boldsymbol{R}_{11})$

Problem-II: $\min_{\mathbf{\Pi}} \sigma_{\max}(\boldsymbol{R}_{22})$

Problem-III: $\max_{\mathbf{\Pi}} \sigma_{\min}(\boldsymbol{R}_{11})$ and $\min_{\mathbf{\Pi}} \sigma_{\max}(\boldsymbol{R}_{22})$

Unification Principle

We can formulate Problem-II as

$$\min_{\mathbf{\Pi}} \sigma_{\max}(\boldsymbol{R}_{22}) = \min_{\mathbf{\Pi}} \frac{1}{\sigma_{\min}(\boldsymbol{R}_{22}^{-1})}$$

$$= \frac{1}{\max_{\mathbf{\Pi}} \sigma_{\min}(\boldsymbol{R}_{22}^{-1})}$$

$$= \frac{1}{\max_{\mathbf{\Pi}} \sigma_{\min}((\boldsymbol{R}_{22}^{-1})^T)}.$$

Unification Principle

We can formulate Problem-II as

$$\min_{\mathbf{\Pi}} \sigma_{\max}(\boldsymbol{R}_{22}) = \min_{\mathbf{\Pi}} \frac{1}{\sigma_{\min}(\boldsymbol{R}_{22}^{-1})}$$

$$= \frac{1}{\max_{\mathbf{\Pi}} \sigma_{\min}(\boldsymbol{R}_{22}^{-1})}$$

$$= \frac{1}{\max_{\mathbf{\Pi}} \sigma_{\min}((\boldsymbol{R}_{22}^{-1})^T)}.$$

Unification Principle: Problem-II can be solved by applying a Problem-I algorithm to $(\boldsymbol{R}^{-1})^T$.

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Notations

Let $\boldsymbol{R}^{(l)}$ be the matrix $\boldsymbol{R}$ at step l. Partition it as

$$\boldsymbol{R}^{(l)} = \begin{array}{c} \\ l \\ m-l \end{array} \begin{array}{c} \begin{array}{cc} l & n-l \end{array} \\ \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{C} \end{bmatrix} \end{array},$$

where $\boldsymbol{A}$ is an upper triangular matrix of size $l \times l$. Denote the columns of $\boldsymbol{B}$ and $\boldsymbol{C}$ by $\boldsymbol{b}_i, \boldsymbol{c}_i$ for $1 \leq i \leq n-l$. Let $\gamma_i = \|\boldsymbol{c}_i\|_2$.

Algorithm Greedy-I.1

1. Initialize $\boldsymbol{R}^{(0)} = \boldsymbol{M}, \boldsymbol{Q} = \boldsymbol{I}, \boldsymbol{\Pi} = \boldsymbol{I}$;
2. for $l = 0, 1, ..., k-1$
   a. Find j such that

$$\max_{1 \le i \le n-l} \sigma_{\min} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b}_i \\ \boldsymbol{0} & \gamma_i \end{bmatrix} = \sigma_{\min} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b}_j \\ \boldsymbol{0} & \gamma_j \end{bmatrix}$$

   b. Exchange columns $l+1$ and $l+j$ of $\boldsymbol{R}^{(l)}$ and update $\boldsymbol{\Pi}$;
   c. Retriangularize the first $l+1$ columns of $\boldsymbol{R}^{(l)}$ to get $\boldsymbol{R}^{(l+1)}$ and update $\boldsymbol{Q}$.

Approximation of Greedy-I.1

- Greedy-I.2 use the reciprocal of the largest row 2-norm of $\boldsymbol{A}^{-1}$;

Approximation of Greedy-I.1

- Greedy-I.2 use the reciprocal of the largest row 2-norm of $\boldsymbol{A}^{-1}$;
- Greedy-I.3 use the reciprocal of the magnitude of the largest element of the last column of $\boldsymbol{A}^{-1}$;

Approximation of Greedy-I.1

- Greedy-I.2 use the reciprocal of the largest row 2-norm of $\boldsymbol{A}^{-1}$;
- Greedy-I.3 use the reciprocal of the magnitude of the largest element of the last column of $\boldsymbol{A}^{-1}$;
- QR with column pivoting use the 2-norm of each $\boldsymbol{c}_i$;

Approximation of Greedy-I.1

- Greedy-I.2 use the reciprocal of the largest row 2-norm of $\boldsymbol{A}^{-1}$;
- Greedy-I.3 use the reciprocal of the magnitude of the largest element of the last column of $\boldsymbol{A}^{-1}$;
- QR with column pivoting use the 2-norm of each $\boldsymbol{c}_i$;
- Algorithm Chan, GKS, Foster.

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Notations

Partition the matrix $\boldsymbol{R}$ into two different ways,

$$\begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{C} \end{bmatrix} \end{array} = \boldsymbol{R}$$

and

$$\begin{array}{c} \\ k-1 \\ m-k+1 \end{array} \begin{array}{c} \begin{array}{cc} k-1 & n-k+1 \end{array} \\ \begin{bmatrix} \bar{\boldsymbol{A}} & \bar{\boldsymbol{B}} \\ \boldsymbol{0} & \bar{\boldsymbol{C}} \end{bmatrix} \end{array} = \boldsymbol{R}$$

Idea of Hybrid-I

- Use an algorithm for Problem-I to select the 'best' column among $\bar{C}$, the lower right $n-k+1$ portion, and permute it to the kth position.

Idea of Hybrid-I

- Use an algorithm for Problem-I to select the 'best' column among $\bar{\boldsymbol{C}}$, the lower right $n-k+1$ portion, and permute it to the kth position.
- Use an algorithm for Problem-II to select the 'worst' column among $\boldsymbol{A}$, the upper left k portion, and permute it to the kth position.

Idea of Hybrid-I

- Use an algorithm for Problem-I to select the 'best' column among $\bar{\boldsymbol{C}}$, the lower right $n-k+1$ portion, and permute it to the kth position.
- Use an algorithm for Problem-II to select the 'worst' column among $\boldsymbol{A}$, the upper left k portion, and permute it to the kth position.
- Repeat the above two procedure until no permutation is required.

Idea of Hybrid-I

Best column: The column with the largest 2-norm among all $n - k + 1$ columns of $\bar{\boldsymbol{C}}$.

Idea of Hybrid-I

Best column: The column with the largest 2-norm among all $n-k+1$ columns of $\bar{\boldsymbol{C}}$.

Worst column: The column with the largest 2-norm among all k columns of $(\boldsymbol{A}^{-1})^T$.

Analysis for Algorithm Hybrid-I

- When Hybrid-I terminates, it guarantees

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_k(\boldsymbol{M})}{\sqrt{k(n-k+1)}},$$

$$\sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{\min}(\boldsymbol{R}_{11})\sqrt{k(n-k+1)}.$$

Analysis for Algorithm Hybrid-I

- When Hybrid-I terminates, it guarantees

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_k(\boldsymbol{M})}{\sqrt{k(n-k+1)}},$$

$$\sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{\min}(\boldsymbol{R}_{11})\sqrt{k(n-k+1)}.$$

- Since $|\det(\boldsymbol{A})|$ strictly increase at each step, Hybrid-I does terminate at some time.

Idea of Hybrid-II

- One way to get Hybrid-II is through unification principle;

Idea of Hybrid-II

- One way to get Hybrid-II is through unification principle;
- Applying Hybrid-I with $k+1$ solves Problem-II;

Idea of Hybrid-II

- One way to get Hybrid-II is through unification principle;
- Applying Hybrid-I with $k+1$ solves Problem-II;
- When Hybrid-II terminates, it guarantees

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_{\max}(\boldsymbol{R}_{22})}{\sqrt{(k+1)(n-k)}},$$

$$\sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{k+1}(\boldsymbol{M})\sqrt{(k+1)(n-k)}.$$

Idea of Hybrid-II

- One way to get Hybrid-II is through unification principle;
- Applying Hybrid-I with $k+1$ solves Problem-II;
- When Hybrid-II terminates, it guarantees

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_{\max}(\boldsymbol{R}_{22})}{\sqrt{(k+1)(n-k)}},$$

$$\sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{k+1}(\boldsymbol{M})\sqrt{(k+1)(n-k)}.$$

- Since $|\det(\boldsymbol{A})|$ strictly increase at each step, Hybrid-II does terminate at some time.

Idea of Hybrid-III

- Alternate between Hybrid-I and Hybrid-II until both of them agree with the current permutation.

Idea of Hybrid-III

- Alternate between Hybrid-I and Hybrid-II until both of them agree with the current permutation.
- When the algorithm halt, the bounds for Hybrid-I and Hybrid-II should hold simultaneously,

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_k(\boldsymbol{M})}{\sqrt{k(n-k+1)}},$$

$$\sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{k+1}(\boldsymbol{M})\sqrt{(k+1)(n-k)}.$$

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Deficiency of RRQR Factorization

- We can measure the performance of the factorization by monitoring $\sigma_k(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{R}_{11})$ and $\sigma_{\max}(\boldsymbol{R}_{22})/\sigma_{k+1}(\boldsymbol{M})$. But these two measurements may not be available in practice.

Deficiency of RRQR Factorization

- We can measure the performance of the factorization by monitoring $\sigma_k(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{R}_{11})$ and $\sigma_{\max}(\boldsymbol{R}_{22})/\sigma_{k+1}(\boldsymbol{M})$. But these two measurements may not be available in practice.
- RRQR factorization cannot guarantee

$$\boldsymbol{\Pi} \begin{bmatrix} -\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12} \\ \boldsymbol{I}_{n-k} \end{bmatrix}$$

to be a stable approximation of the right null space of $\boldsymbol{M}$.

Strong Rank Revealing QR Factorization

Definition (Strong RRQR factorization)

Let $\boldsymbol{M\Pi} = \boldsymbol{QR}$ be the QR factorization of $\boldsymbol{M\Pi}$. Partition $\boldsymbol{R}$ as

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{R}_{22} \end{bmatrix},$$

where $\boldsymbol{R}_{11}$ is an upper triangular matrix with order k. Strong RRQR factorization aims to choose $\boldsymbol{\Pi}$ such that

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_k(\boldsymbol{M})}{q_1(k,n)}, \qquad \sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{k+1}(\boldsymbol{M}) q_2(k,n),$$

$$\left|(\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12})_{i,j}\right| \leq q_3(k,n), \quad \text{for } 1 \leq i \leq k, 1 \leq j \leq n-k$$

where $q_1(k,n)$, $q_2(k,n)$ and $q_3(k,n)$ are functions bounded by some low-degree polynomials in k and n.

Notations

- Let $\mathbf{\Pi}_{i,j}$ denotes a permutation matrix that permutes the ith and jth columns of a given matrix.

Notations

- Let $\mathbf{\Pi}_{i,j}$ denotes a permutation matrix that permutes the ith and jth columns of a given matrix.
- Define matrix operators $\mathcal{A}_k, \mathcal{C}_k, \mathcal{R}_k$ by

$$\mathcal{A}_k(\boldsymbol{M}) = \boldsymbol{A}_k, \quad \mathcal{C}_k(\boldsymbol{M}) = \boldsymbol{C}_k, \quad \mathcal{R}_k(\boldsymbol{M}) = \boldsymbol{R}_k,$$

where $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{R}_k$ is the QR factorization of $\boldsymbol{M}$ up to its kth column and

$$\boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{A}_k & \boldsymbol{B}_k \\ \mathbf{0} & \boldsymbol{C}_k \end{bmatrix},$$

where $\boldsymbol{A}_k$ is an upper triangular matrix of order k.

Algorithm SRRQR-1

1. QR factor $\boldsymbol{M}$ up to the kth column, $\boldsymbol{QR} = \boldsymbol{M}$;
2. Initialize $\boldsymbol{\Pi} = \boldsymbol{I}$, set $f \geq 1$;
3. while exist i, j such that $\det(\mathcal{A}(\boldsymbol{R\Pi}_{i,j+k}))/\det(\mathcal{A}(\boldsymbol{R})) > f$
   a. Find such i, j;
   b. Permute column i and $j+k$ of $\boldsymbol{R}$. Update $\boldsymbol{\Pi}, \boldsymbol{R}, \boldsymbol{Q}$.

Analysis for Algorithm SRRQR-1

Lemma

Assume $\boldsymbol{A}_k = \mathcal{A}_k(\boldsymbol{R})$ *has positive diagonal elements (Householder reflections and Givens rotations give the flexibility to choose the sign). Let* $\bar{\boldsymbol{A}}_k = \mathcal{A}_k(\boldsymbol{R}\boldsymbol{\Pi}_{i,k+j})$, *where* $i \leq k$ *and* $j > 1$. *Then*

$$\frac{\det(\bar{\boldsymbol{A}}_k)}{\det(\boldsymbol{A}_k)} = \sqrt{(\boldsymbol{A}_k^{-1}\boldsymbol{B}_k)_{i,j}^2 + \left(\frac{\gamma_j(\boldsymbol{C}_k)}{\omega_i(\boldsymbol{A}_k)}\right)^2}, \tag{1}$$

where $\gamma_j(\boldsymbol{C}_k)$ *denote the 2-norm of the* j*th column of* $\boldsymbol{C}$ *and* $\omega_i(\boldsymbol{A}_k)$ *denote the reciprocal of the 2-norm of the* i*th row of* $\boldsymbol{A}^{-1}$.

Analysis for Algorithm SRRQR-1

When SRRQR-1 terminates, it guarantees

$$\sigma_{\min}(\boldsymbol{R}_{11}) \geq \frac{\sigma_i(\boldsymbol{M})}{\sqrt{1+f^2k(n-k)}},$$

$$\sigma_{\max}(\boldsymbol{R}_{22}) \leq \sigma_{j+k}(\boldsymbol{M})\sqrt{1+f^2k(n-k)},$$

$$\left|(\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12})_{i,j}\right| \leq f, \quad \text{for } 1 \leq i \leq k, 1 \leq j \leq n-k.$$

Implementation Notes of SRRQR-1

- Instead of computing the ratio of two determinant, we terminate the loop if

$$\max_{1\leq i\leq k, 1\leq j\leq n-k} \sqrt{(\boldsymbol{A}_k^{-1}\boldsymbol{B}_k)_{i,j}^2 + \left(\frac{\gamma_j(\boldsymbol{C}_k)}{\omega_i(\boldsymbol{A}_k)}\right)^2} \leq f.$$

Implementation Notes of SRRQR-1

- Instead of computing the ratio of two determinant, we terminate the loop if

$$\max_{1\le i\le k, 1\le j\le n-k} \sqrt{(\boldsymbol{A}_k^{-1}\boldsymbol{B}_k)_{i,j}^2 + \left(\frac{\gamma_j(\boldsymbol{C}_k)}{\omega_i(\boldsymbol{A}_k)}\right)^2} \le f.$$

- Relaxing the loop condition to

$$\max_{1\le i\le k, 1\le j\le n-k} \max\left\{ \left|(\boldsymbol{A}_k^{-1}\boldsymbol{B}_k)_{i,j}\right|, \frac{\gamma_j(\boldsymbol{C}_k)}{\omega_i(\boldsymbol{A}_k)} \right\} \le f.$$

  gives algorithm SRRQR-2.

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Settings

- Implemented in MATLAB 2016a where the machine precision is ϵ =2.2204e-16.
- For Greedy-I.1 and Chan, power method is terminated until the error of the singular value is less than 0.01.
- The parameter δ in Foster is set to 1e-04.

Settings

- Implemented in MATLAB 2016a where the machine precision is ϵ =2.2204e-16.
- For Greedy-I.1 and Chan, power method is terminated until the error of the singular value is less than 0.01.
- The parameter δ in Foster is set to 1e-04.
- For SRRQR-1 and SRRQR-2, I set

$$f = \sqrt{k(n-k) + \min(k, n-k)} / \sqrt{k(n-k)}$$

so that they have similar bound on $\sigma_k(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{R}_{11})$ and $\sigma_{\max}(\boldsymbol{R}_{22})/\sigma_{k+1}(\boldsymbol{M})$ as hybrid algorithms.

Kahan Matrix

Definition

Kahan matrix of order n is defined as

$$\boldsymbol{K}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & s & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & s^{n-1} \end{bmatrix} \begin{bmatrix} 1 & -c & \cdots & -c \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -c \\ 0 & \cdots & 0 & 1 \end{bmatrix},$$

where $c^2 + s^2 = 1$.

Algorithm Greedy-I.1, Greedy-I.2, Greedy-I.3, QR with column pivoting won't perform any permutations for Kahan matrix.

Kahan Matrix

- Matrix $\boldsymbol{M}$, size 50×50, $c = 0.2$, elements defined as the previous page.
- $\text{cond}(\boldsymbol{M})$ =4.9910e+04, $\text{rank}(\boldsymbol{M}) = 49$;
- Set $k = 48$. We want to test whether the algorithm can split the 'bad' column away from the first 48 columns.

Kahan Matrix

SRRQR-1 guarantees

$$\max\left\{\frac{\sigma_k(\boldsymbol{M})}{\sigma_k(\boldsymbol{R}_{11})}, \frac{\sigma_1(\boldsymbol{R}_{22})}{\sigma_{k+1}(\boldsymbol{M})}\right\} < \sqrt{1 + k(n-k) + \min(k, n-k)} = 9.9499$$

and

$$\max|\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12}| < \frac{\sqrt{k(n-k) + \min(k, n-k)}}{\sqrt{k(n-k)}} = 1.0104$$

Kahan Matrix

	$n = 50$, rank$(\boldsymbol{M}) = 49$, rank$(\boldsymbol{M})$ =4.9910e+04, $k = 48$			
	Time	$\sigma_k(\boldsymbol{M})/\sigma_k(\boldsymbol{R}_{11})$	$\sigma_1(\boldsymbol{R}_{22})/\sigma_{k+1}(\boldsymbol{M})$	$\max \lvert \boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12} \rvert$
QR	0.0003	606.0540	0.2000	1.0533e+03
Greedy-I.1	0.0319	0.2012	0.2191	0.8333
Greedy-I.2	0.0026	249.4408	0.2399	654.2344
Greedy-I.3	0.0015	249.4408	0.2399	654.2344
ColumnPivot	0.0004	249.4408	0.2399	654.2344
Chan	0.0009	606.0540	0.2040	1.0533e+03
GKS	0.9957	0.2012	0.2191	0.8333
Foster	0.0006	606.0540	0.2040	1.0533e+03
Hybrid-I	0.0004	0.2012	0.2191	0.8333
Hybrid-II	0.0005	0.2012	0.2191	0.8333
Hybrid-III	0.0014	0.2012	0.2191	0.8333
SRRQR-1	0.0005	0.2012	0.2191	0.8333
SRRQR-2	0.0005	0.2012	0.2191	0.8333

Scaled Random Matrix

- Matrix $\boldsymbol{M}$, size 50×50, elements generated from $[0, 1]$ under uniform distribution;
- Scale the ith row of $\boldsymbol{M}$ by $20\epsilon^{i/50}$;
- cond($\boldsymbol{M}$) =4.9411e+15, rank($\boldsymbol{M}$) = 46;

Scaled Random Matrix

Since $\sigma_1/\sigma_{25} > 1000$, set $k = 25$, then SRRQR-1 guarantees

$$\max\left\{\frac{\sigma_k(\boldsymbol{M})}{\sigma_k(\boldsymbol{R}_{11})}, \frac{\sigma_1(\boldsymbol{R}_{22})}{\sigma_{k+1}(\boldsymbol{M})}\right\} < \sqrt{1 + k(n-k) + \min(k, n-k)}$$
$$= 25.5147$$

and

$$\max|\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12}| < \frac{\sqrt{k(n-k) + \min(k, n-k)}}{\sqrt{k(n-k)}} = 1.0124$$

Scaled Random Matrix

	$n = 50$, rank$(\boldsymbol{M}) = 46$, cond$(\boldsymbol{M})$ =4.9411e+15, set $k = 25$			
	Time	$\sigma_k(\boldsymbol{M})/\sigma_k(\boldsymbol{R}_{11})$	$\sigma_1(\boldsymbol{R}_{22})/\sigma_{k+1}(\boldsymbol{M})$	$\max \lvert \boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12} \rvert$
QR	0.0002	3.6819	1.0827	10.9676
Greedy-I.1	0.0087	0.6233	1.0045	3.0780
Greedy-I.2	0.0021	3.5619	1.0856	12.8870
Greedy-I.3	0.0010	3.5619	1.0856	12.8870
ColumnPivot	0.0003	3.5619	1.0856	12.8870
Chan	0.0004	0.7278	0.3980	1.2874
GKS	0.0974	4.3802	1.4820	8.3536
Foster	0.0004	1.3973	1.1835	5.1992
Hybrid-I	0.0006	0.5178	1.4561	1.0888
Hybrid-II	0.0004	0.5789	0.3813	1.6461
Hybrid-III	0.0012	0.5178	0.4561	1.0886
SRRQR-1	0.0017	0.3335	0.4003	1.0097
SRRQR-2	0.0019	0.3863	0.3514	0.9317

Outline

1 Introduction

2 Greedy RRQR Algorithms

3 Hybrid RRQR Algorithms

4 Strong RRQR Algorithms

5 Applications and Numerical Results
- Revealing Matrix Rank Deficiency
- Matrix Approximation

Problem Setup

Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, want to find a rank-$k$ matrix $\boldsymbol{A}_k$ such that

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$$

is minimized.

Truncated SVD-based Solution

The problem is solved by truncated SVD,

$$\boldsymbol{A}_k = \sum_{i=1}^{k} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^T .$$

Truncated SVD-based Solution

The problem is solved by truncated SVD,

$$\boldsymbol{A}_k = \sum_{i=1}^{k} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^T .$$

This solution gives

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}.$$

RRQR-based Solution

Suppose the RRQR factorization of $\boldsymbol{A}$ is given by

$$\boldsymbol{A\Pi} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{R}_{22} \end{bmatrix}.$$

RRQR-based Solution

Suppose the RRQR factorization of $\boldsymbol{A}$ is given by

$$\boldsymbol{A\Pi} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{R}_{22} \end{bmatrix}.$$

The truncated RRQR approximation is

$$\boldsymbol{B}_k = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Pi}^T.$$

RRQR-based Solution

The RRQR based approximation gives

$$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2 = \left\| \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{R}_{22} \end{bmatrix} \boldsymbol{\Pi}^T - \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Pi}^T \right\|_2$$

$$= \|\boldsymbol{R}_{22}\|_2$$

$$\leq \sigma_{k+1}(\boldsymbol{A}) q_2(k, n).$$

RRQR-based Solution

The RRQR based approximation gives

$$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2 = \left\| \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{R}_{22} \end{bmatrix} \boldsymbol{\Pi}^T - \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Pi}^T \right\|_2$$

$$= \|\boldsymbol{R}_{22}\|_2$$

$$\leq \sigma_{k+1}(\boldsymbol{A}) q_2(k, n).$$

The approximation is good as long as $\sigma_{k+1}(\boldsymbol{A})$ is sufficiently small.

Numerical Results

Matrix $n = 100, k = 50$			Error		Time	
σ_1	σ_{50}	$\sigma_{51}, \ldots, \sigma_{100}$	$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2$	$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$	RRQR	SVD
1000	1	10^{-1}	0.6456	0.1000	0.0432	0.2947
1000	1	10^{-4}	6.5303e-04	1.0000e-04	0.0383	0.2981
1000	1	10^{-7}	6.5195e-07	1.0000e-07	0.0383	0.2843

Matrix $n = 100, k = 90$			Error		Time	
σ_1	σ_{90}	$\sigma_{91}, \ldots, \sigma_{100}$	$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2$	$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$	RRQR	SVD
1000	1	10^{-1}	0.6275	0.1000	0.0551	0.9645
1000	1	10^{-4}	4.8696e-04	1.0000e-04	0.0542	0.8683
1000	1	10^{-7}	6.0481e-07	1.0000e-07	0.0521	0.8742

Numerical Results

Matrix $n = 100, k = 50$			Error		Time	
σ_1	σ_{50}	$\sigma_{51}, \ldots, \sigma_{100}$	$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2$	$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$	RRQR	SVD
1000	1	10^{-1}	0.6456	0.1000	0.0432	0.2947
1000	1	10^{-4}	6.5303e-04	1.0000e-04	0.0383	0.2981
1000	1	10^{-7}	6.5195e-07	1.0000e-07	0.0383	0.2843
Matrix $n = 100, k = 90$			Error		Time	
σ_1	σ_{90}	$\sigma_{91}, \ldots, \sigma_{100}$	$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2$	$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$	RRQR	SVD
1000	1	10^{-1}	0.6275	0.1000	0.0551	0.9645
1000	1	10^{-4}	4.8696e-04	1.0000e-04	0.0542	0.8683
1000	1	10^{-7}	6.0481e-07	1.0000e-07	0.0521	0.8742

- The approximation error of RRQR based rank-k approximation is of the same order as SVD based rank-k approximation;

Numerical Results

Matrix $n = 100, k = 50$			Error		Time	
σ_1	σ_{50}	$\sigma_{51}, \ldots, \sigma_{100}$	$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2$	$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$	RRQR	SVD
1000	1	10^{-1}	0.6456	0.1000	0.0432	0.2947
1000	1	10^{-4}	6.5303e-04	1.0000e-04	0.0383	0.2981
1000	1	10^{-7}	6.5195e-07	1.0000e-07	0.0383	0.2843
Matrix $n = 100, k = 90$			Error		Time	
σ_1	σ_{90}	$\sigma_{91}, \ldots, \sigma_{100}$	$\|\boldsymbol{A} - \boldsymbol{B}_k\|_2$	$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2$	RRQR	SVD
1000	1	10^{-1}	0.6275	0.1000	0.0551	0.9645
1000	1	10^{-4}	4.8696e-04	1.0000e-04	0.0542	0.8683
1000	1	10^{-7}	6.0481e-07	1.0000e-07	0.0521	0.8742

- The approximation error of RRQR based rank-k approximation is of the same order as SVD based rank-k approximation;
- RRQR (Hybrid-III) runs much faster than SVD.

Conclusions

- This thesis focuses on algorithms for computing an RRQR factorization, including greedy RRQR algorithms, hybrid RRQR algorithms and strong RRQR algorithms.

Conclusions

- This thesis focuses on algorithms for computing an RRQR factorization, including greedy RRQR algorithms, hybrid RRQR algorithms and strong RRQR algorithms.
- Analytical and numerical analysis both show that strong RRQR algorithms give the best RRQR factorization.

Conclusions

- This thesis focuses on algorithms for computing an RRQR factorization, including greedy RRQR algorithms, hybrid RRQR algorithms and strong RRQR algorithms.
- Analytical and numerical analysis both show that strong RRQR algorithms give the best RRQR factorization.
- RRQR factorization can be applied into rank deficient problems and it gives approximately the same result as SVD while it is much more efficient in computation.

Acknowledgement

I would like to express my sincere gratitude to

- Prof. Zengqi Wang (Advisor for this thesis)
- Prof. Jinyan Fan, Prof. Xiaomin Wang (Committee Members)
- Prof. Shi Jin (Advisor in UW-Madison)