



# Rank Revealing QR Methods for Sparse Block Low Rank Solvers

---

**Eragul Korkmaz**, Mathieu Faverge, Grégoire Pichon, Pierre Ramet

11 July 2019 – Sparse Days

# Table of contents

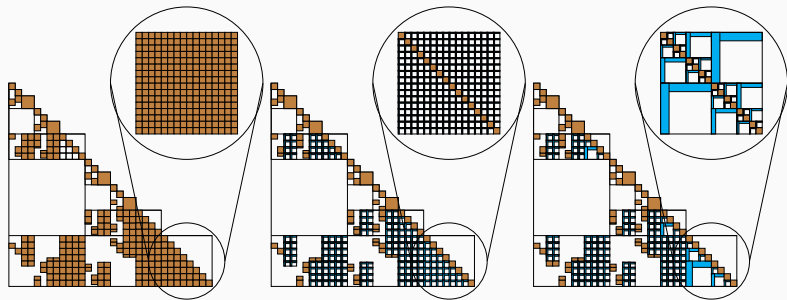
1. Background
2. Compression Methods
3. Numerical Results

# Background

---

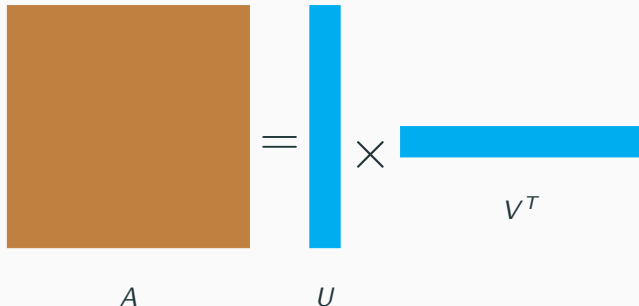
# Background - ANR SaSHiMi Project

## ANR SaSHiMi Project



- With: Mathieu Faverge, Pierre Ramet, Grégoire Pichon
- General Picture: Solve linear equations  $Ax=b$  for **large sparse systems**
- Full Rank Format: Too much memory usage
- Block Low Rank Format: Compression is possible, so less storage and faster
- Hierarchical Format: Even less computational complexity and memory consumption

## Background - Block Low Rank Structure



$$A \in \mathbb{R}^{m \times n}; U \in \mathbb{R}^{m \times r}; V \in \mathbb{R}^{n \times r}$$

- Compression reduces memory and cost of computations
- Fixed block size  $\leq 300$   $\xrightarrow{\text{Future}}$  variable and larger
- All the compression algorithms were existent methods in this presentation
- In PASTIX, the rank is numerically decided to be the smallest rank at an user defined precision:  $\|A - UV^T\|_F \leq \epsilon \|A\|_F$

# Background Information - Singular Value Decomposition(SVD)

## Main Features

- SVD has the form:  $A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}$
- $\Sigma$  is a diagonal singular values matrix, U and V are the left and right singular vectors of A
- Two options for the threshold:
  - $\sigma_{k+1} \leq \epsilon$  ✗
  - $\sqrt{\sum_{i=k+1}^n \sigma_i^2} \leq \epsilon$  ✓ (to be consistent with QR methods)

## Discussions

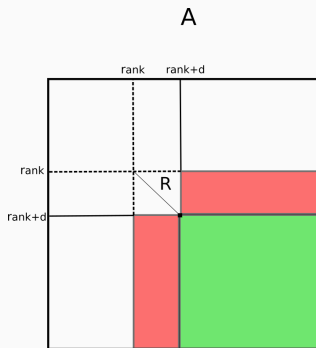
- 😊 Good accuracy
- 😞 Too costly

The aim of this study is to find closest ranks to SVD at the same precision with better performance

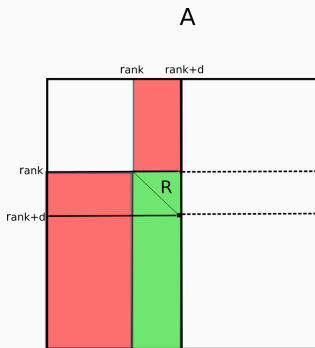
- Rank Revealing QR methods ( $A = QR$ )

# Left Looking vs Right Looking Algorithms

Right Looking



Left Looking



- Red parts are read / Green parts are updated
- Right Looking: Unnecessary updates but more suitable for parallelism
- Left Looking: Eliminated unnecessary updates but more storage needed

# Pivoting vs Rotating

- Rank revealing: gather important data and omit the remainings
- Two ways of data gathering methods:
  - Pivoting:  $AP = Q_{AP}R_{AP}$
  - Rotation:  $AQ_{Rot} = Q_{AQ}R_{AQ}$
- Pivoting: gather important data on the leftmost matrix
- Rotation: gather important data on the diagonal



# Compression Methods

---

# Rank Revealing QR Methods

- Partial QR with Column Pivoting (PQRCP)
  - LAPACK xGEPQ3 modified by Buttari A.(MUMPS)
- Randomized QR with Column Pivoting (RQRCP)
  - Duersch J. A. and Gu M. (2017)
  - Martinsson P. G. (2015)
  - Xiao J., Gu M. and Langou J. (2017)
- Truncated Randomized QR with Column Pivoting (TRQRCP)
  - Duersch J. A., Gu M. (2017)
- Randomized QR with Rotation (RQRRT)
  - Martinsson P. G. (2015)

# 1) Partial QR with Column Pivoting (PQRCP)

## Main Features

- Column pivoting: column with max 2-norm is the pivot
- $A = UV^T$  compression with column pivoting:
  - $AP = Q_{AP}R_{AP}$  is computed, where  $P$  is the permutation matrix
  - $U = Q_{AP}$  and  $V^T = R_{AP}P^T$
- Right Looking

## Discussions

- 😞 Need larger rank than SVD for the same accuracy
- 😞 Not fast enough
- To reduce the cost of pivot selection
  - [Randomized method with pivoting](#)

## 2) Randomized QR with Column Pivoting (RQRCP)

### Main Features

- Create independent and identically distributed Gaussian matrix  $\Omega$  of size  $b \times m$ , where  $b \ll m$
- Compute the sample matrix  $B = \Omega A$  of size  $b \times n$
- Find pivots on  $B$  where the row dimension is much smaller than  $A$ 
  - Less computations
  - Less communication
- Apply this pivoting to  $A$  like in PQRCP
- Right Looking
- Sample matrix updated

### Discussions

- 😞 Similar accuracy to PQRCP
- 😞 Not fast enough
- To eliminate the cost of trailing matrix update:
  - Truncated randomized method with pivoting

### 3) Truncated Randomized QR with Column Pivoting (TRQRCP)

#### Main Features

- Left Looking
  - Trailing matrix is not needed
- Extra storage: Reflector accumulations
- More efficient on large matrices with small ranks

#### Discussions

- 😊 Fastest in sequential
- 😐 Similar accuracy to previous algorithms
- 😞 Can be improved to give closer ranks to SVD
- Instead of pivoting, apply a reasonable rotation to gather important information to the diagonal blocks
  - [Randomized method with rotation](#)

## 4) Randomized QR with Rotation (RQRRT)

### Main Features

- Similar to RQRCP except:
  - Rotation applied to  $A$
  - Resampling
- In Randomized QR with Column Pivoting (RQRCP):
  - $BP_B = Q_B R_B$
  - $AP_B = Q_{AP} R_{AP}$
  - $U = Q_{AP}$  and  $V^T = R_{AP} P_B^T$
- In Randomized QR with Rotation (RQRRT):
  - $B^T = Q_B R_B$
  - $AQ_B = Q_{AQ} R_{AQ}$
  - $U = Q_{AQ}$  and  $V^T = R_{AQ} Q_B^T$
- Right Looking

### Discussions

- 😊 Ranks closest to SVD
- 😞 Slower and updates whole trailing matrix each iteration

# Complexities

- **Blue:** No change, **Green:** Reduced cost, **Red:** More costly
- Matrix size  $n \times n$ , block size  $b$ , rank  $k$

Methods	Features
SVD: $\mathcal{O}(n^3)$	
PQRCP: $\mathcal{O}(n^2k)$	pivot finding $\mathcal{O}(n^2)$ trailing matrix update $\mathcal{O}(n^2k)$
PQRCP: $\mathcal{O}(n^2k) \xrightarrow{\text{Randomization}} \text{RQRCP: } \mathcal{O}(n^2k)$	sample matrix generation (beginning) $\mathcal{O}(n^2b)$ pivot finding $\mathcal{O}(nb)$ update of sample matrix B $\mathcal{O}(nb^2)$ trailing matrix update $\mathcal{O}(n^2k)$
RQRCP: $\mathcal{O}(n^2k) \xrightarrow{\text{Truncation}} \text{TRQRCP: } \mathcal{O}(nk^2)$	sample matrix generation (beginning) $\mathcal{O}(n^2b)$ pivot finding $\mathcal{O}(nb)$ update of current panel $\mathcal{O}(nk^2)$ update of sample matrix B $\mathcal{O}(nb^2)$
RQRCP: $\mathcal{O}(n^2k) \xrightarrow{\text{Rotation}} \text{RQRRT: } \mathcal{O}(n^2k)$	resampling (each iteration) $\mathcal{O}(n^2b)$ rotation finding $\mathcal{O}(n^2k)$ rotation of A $\mathcal{O}(n^2k)$ trailing matrix update $\mathcal{O}(n^2k)$

- Flops cost (< is less flops):  
 $\text{TQRCP} \ll \text{PQRCP} < \text{RQRCP} < \text{RQRRT} \ll \text{SVD}$

## Conclusion

- SVD: Smallest rank but too costly
- PQRCP: Right looking. Randomization is suggested for pivoting cost
- RQRCP: Unnecessary trailing matrix update. Truncation is introduced
- TRQRCP: Lowest cost, similar accuracy.
- RQRRT: Closest ranks to SVD. Most costly QR variant. Promising for parallelism
- In PASTIX, the smallest rank is decided numerically at an user defined precision



## Numerical Results

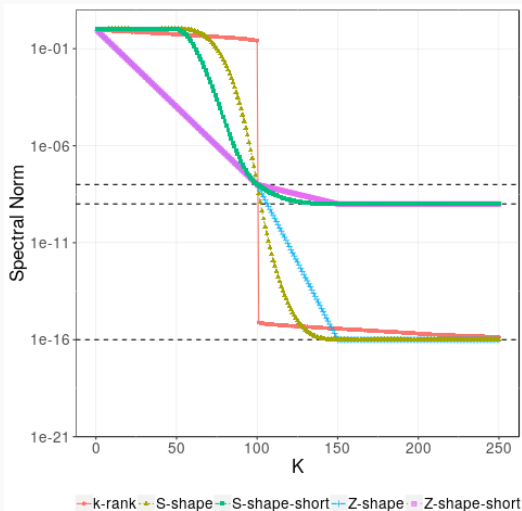
---

# Test Cases - Singular Values

For all Test Cases(Modes):

- Modes are different matrices
- Matrix Size 500
- Rank 100
- Generation Precision  $\epsilon = 10^{-8}$
- $A = UDV^T$ 
  - D is a diagonal matrix with singular values
  - U and V are orthonormal random matrices

Spectral Norms

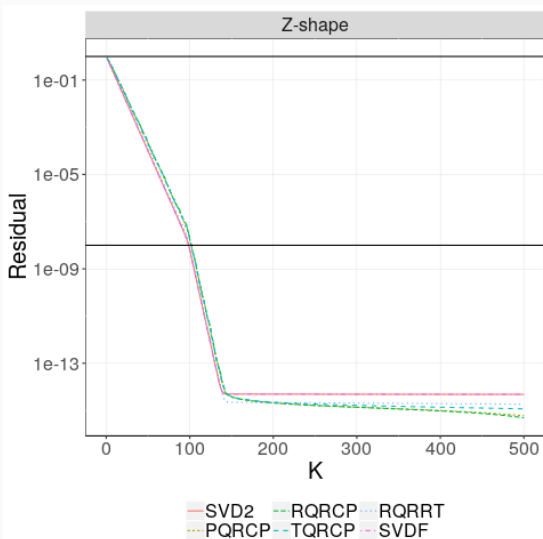


# Stability - First Test Case Residual Norms

For all Methods:

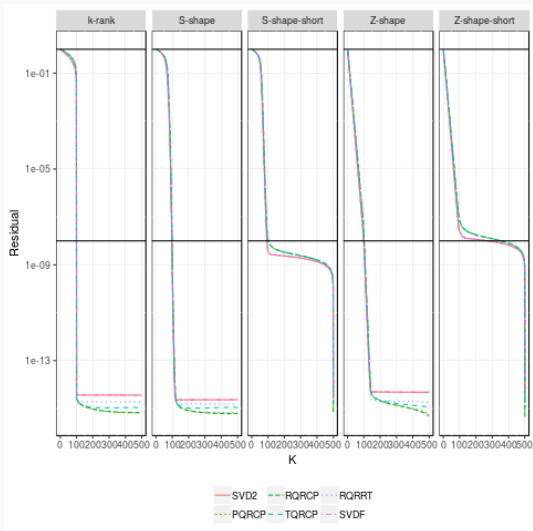
- Mode 1
- Matrix is fully factorized without any stopping criterion
- Residual:  $\frac{\|A - U_K V_K^T\|_F}{\|A\|_F}$
- K stands for index values of the matrix

Index vs Error



# Stability - All Test Cases Residual Norms

## Index vs Error

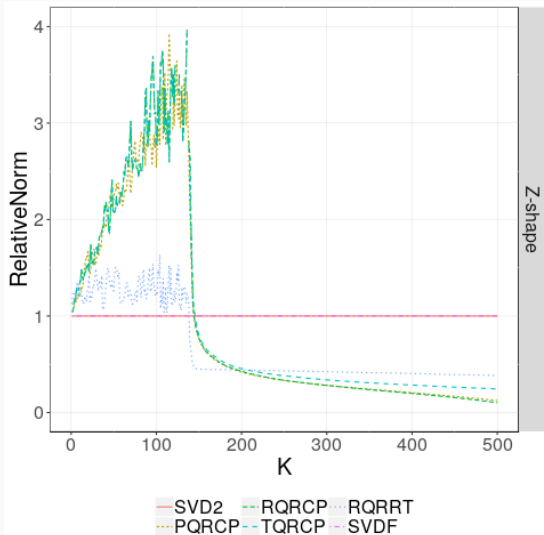


# Stability - First Test Case Relative Residual Norms

Index vs Relative Error

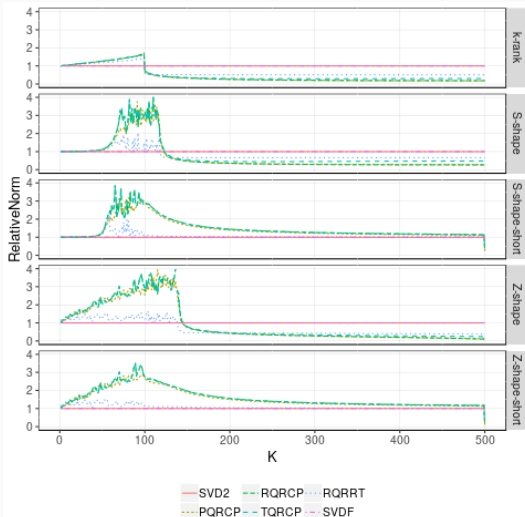
For all Methods:

- Matrix is fully factorized without any stopping criterion
- Relative Norm:  
$$\frac{\text{Residual}}{\text{Residual}^{\text{SVDf}}}$$
- K stands for index values of the matrix



# Stability - All Test Cases Relative Residual Norms

## Index vs Relative Error

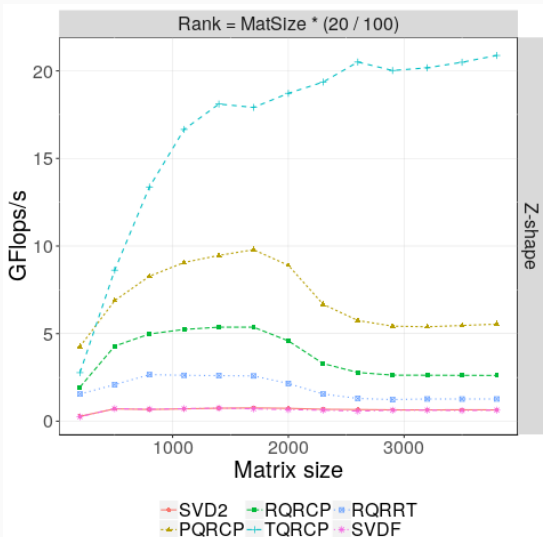


# Performance - First Test Case Gflops

For all Methods:

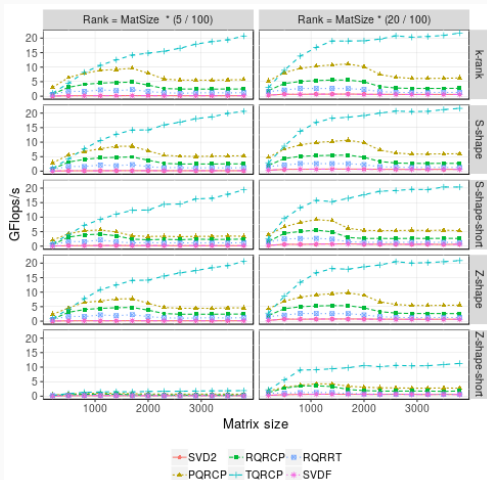
- Mode 1
- Rank =  
 $Matrix\_size \times \frac{20}{100}$
- Different matrix sizes are checked
- Compression Precision  
 $\epsilon = 10^{-8}$
- $Gflops = \frac{\min(Gflops)}{t}$
- Threshold is applied

Matrix Size vs GFlops



# Performance - All Test Cases Gflops

## Matrix Size vs GFlops



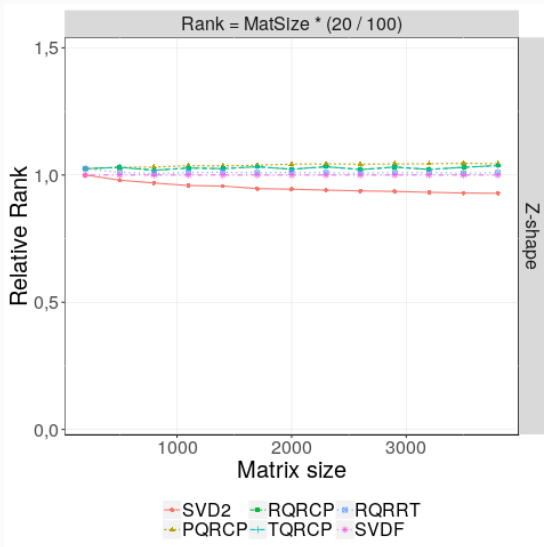


# Compression Ranks - First Test Case Relative Rank

Matrix Size vs Relative Rank

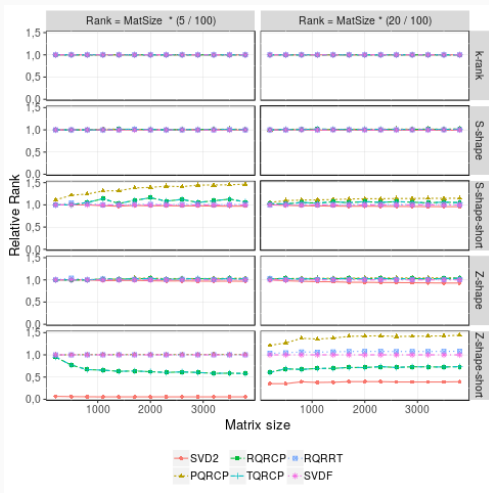
For all Methods:

- Mode 1
- Rank =  
 $Matrix\_size \times \frac{20}{100}$
- Different matrix sizes are checked
- Compression Precision  $\epsilon = 10^{-8}$
- RelativeRank =  
 $\frac{comp\_rank}{comp\_rank^{SVDF}}$
- Threshold is applied






# Compression Ranks - All Test Cases Relative Rank

## Matrix Size vs Relative Rank



- Application in the parallel framework of PASTIX with real, larger and tricky matrices
  - Why PQRCP has more performance than RQRCP and TRQRCP?
  - RQRRT has the worst QR performance but is promising for parallel environment
  - Why RQRCP and TRQRCP finds smaller ranks than SVDF for the mode 3
  - Tuning the best method for the given parameters

-  Duersch, J. A.; Gu, M. (2017). "Randomized QR with Column Pivoting".
-  Xiao, J.; Gu, M.; Langou, J. (2017). "Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-rank Matrix Approximations".
-  Martinsson, P. G. (2015). "Blocked Rank-revealing QR Factorizations: how randomized sampling can be used to avoid single-vector pivoting".

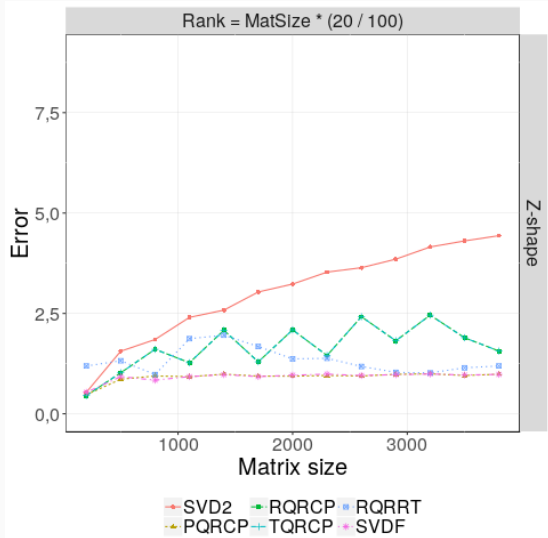
*THANK YOU!*

# Accuracy - First Test Case

## Matrix Size vs Relative Rank

For all Methods:

- Mode 1
- Rank =  
 $Matrix\_size \times \frac{20}{100}$
- Different matrix sizes are checked
- Compression Precision  
 $\epsilon = 10^{-8}$
- Error =  $\frac{\|A - U_k V_k^T\|_F}{\epsilon \|A^{[0]}\|_F}$
- Threshold is applied



# Accuracy - All Test Cases

## Matrix Size vs Relative Rank

