Improvement on the Precision of QR Factorization

An Analysis on the Fractional Implementation

James Guo, Tina Shen, and Anna Dai
Johns Hopkins University

Contact Information:
Department of Mathematics,
Department of Applied Mathematics and Statistics
Email: `{sguo45,xshen43,ndai3}@jhu.edu`

Abstract

QR factorization is a fundamental algorithm in the field of computational mathematics, whose applications are among solving linear systems, eigenvalue problems, and linear regression. The current computations using *classical and modified Gram-Schmidt* algorithms are susceptible to floating-point errors, and could lead to *catastrophic cancellation*. Our project investigates a $\mathbb{Q}$-based fractional computation model to reduce roundoff errors, explores alternatives around certain issues, and analyzes the trade-off between computational complexity and numerical precision. Results show Fractional QR in applications achieves better and arbitrary numerical precision compared to traditional QR factorization methods.

Introduction

The main concern is to address the QR factorization in the background of machine representation.

QR Factorization

Let $A \in \mathbb{R}^{n\times n}$ be a matrix, its QR factorization factors A into $A = QR$, where $Q \in \mathbb{R}^{n\times n}$ is orthogonal (*i.e.*, $Q^\intercal Q = I$) and $R \in \mathbb{R}^{n\times n}$ is upper triangular. Meanwhile, A does not need to be square matrix, and can be consisted of $\mathbb{C}$ entries, and it is a trivial extension.

Machine Representation

Since $\mathbb{R}$ is uncountable, and the machine can at most represent a countable set of numbers, so the real numbers cannot be fully represented. The current machine standard uses IEEE standards:

- For single precision, there are 32 bits (or 8 bytes).

1 sign bit	`#e` 8 exponent bits	`#f` 23 mantissa bits

- For double precision, there are 64 bits (or 8 bytes).

1 sign bit	`#e` 11 exponent bits	`#f` 52 mantissa bits

With double precision, the smallest positive machine representable number (MRN) is 2^{-127}, and the $\epsilon_{\text{machine}}$ is 2^{-52}.

Catastrophic Cancellation

When subtraction were between two close, large numbers result in precision loss for the differences.

Background

The typical algorithm for the QR factorization is the Gram Schmidt algorithm, decomposing V into $Q \cdot R$, and the classical and modified versions differ in error bound.

Classical:
```
• for $j \leftarrow 1, \cdots, n$:
    – Let $v_j \leftarrow a_j$.
    – for $i \leftarrow 1, \cdots, j-1$:
        * $r_{i,j} \leftarrow q_i^\intercal a_j$.
        * $v_j \leftarrow v_j - r_{i,j} q_i$.
    – $r_{j,j} \leftarrow \|v_j\|_2$.
    – $q_j \leftarrow v_j / r_{j,j}$.
```

Modified:
```
• for $i \leftarrow 1, \cdots, n$:
    – Let $v_i \leftarrow a_i$.
• for $i \leftarrow 1, \cdots, n$:
    – $r_{i,i} \leftarrow \|v_i\|_2$.
    – $q_i \leftarrow v_i / r_{i,i}$
    – for $j \leftarrow i+1, \cdots, n$:
        * $r_{i,j} \leftarrow q_i^\intercal v_j$.
        * $v_j \leftarrow v_j - r_{i,j} q_i$
```

For the classical version, the lower bound if $\sqrt{\epsilon_{\text{machine}}}$, and for the modified version, the lower bound is $\epsilon_{\text{machine}}$.

Methods

The motivation of our project is to use a fractional representation for matrix operations. Since $\mathbb{Q}$ is *countable* and dense in $\mathbb{R}$ ($\mathbb{Q}[i]$ lattice is **dense** in $\mathbb{C}$), so it is (theoretically) representable, so we can ensure entry-wise completeness for matrix multiplications.

Newton's Method

Note that $\mathbb{Q}$ is not closed under square root operator, so we decided to go around using Newton's method, which converges to the square root result.

Precision and Complexity Trade-off

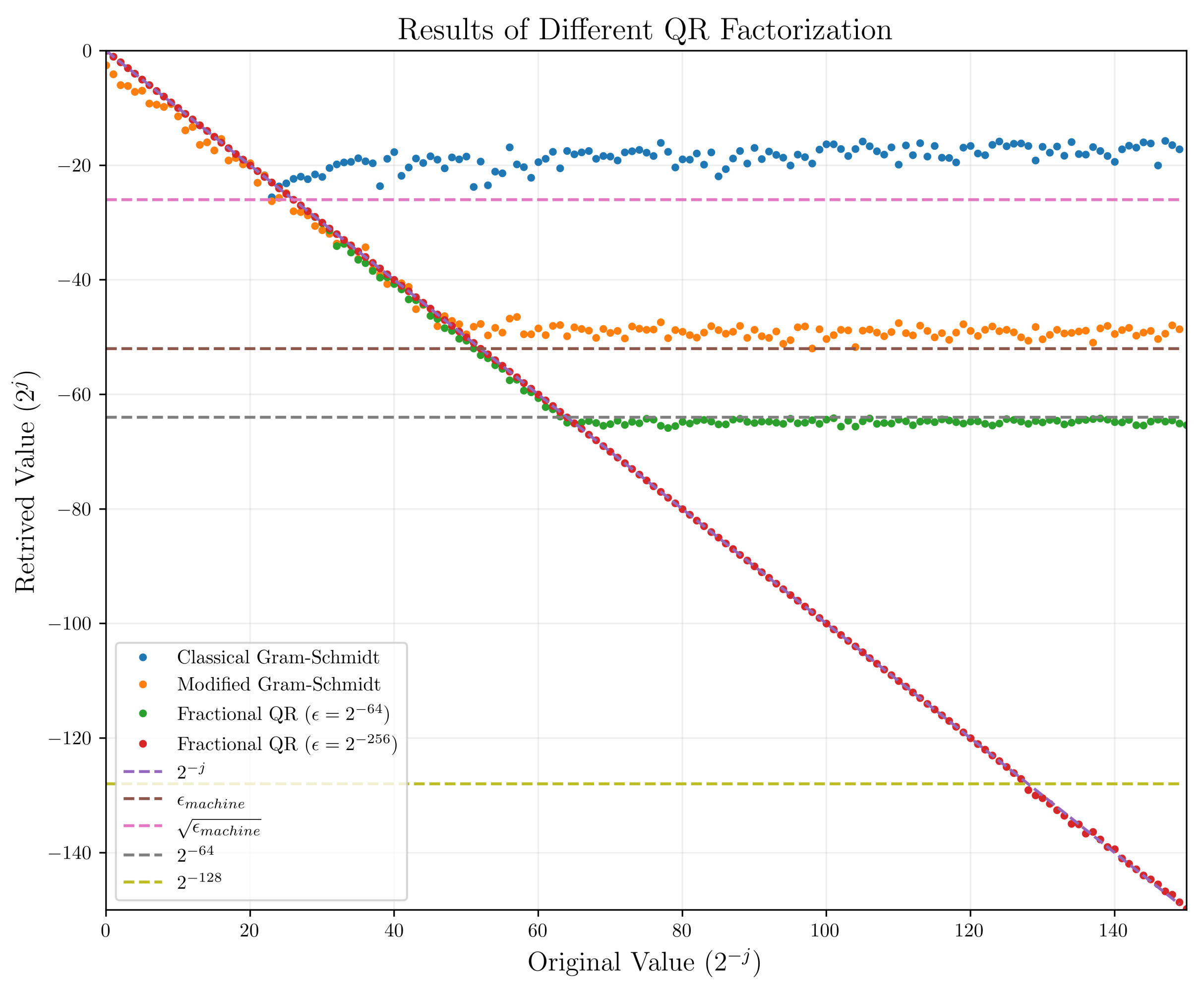
When integers are large, we account for the large cost of additions and multiplications outside asymptotic behavior.

Results

For the experiment, we compare the following modes:

- Classical Gram-Schmidt algorithm.
- Improved Gram-Schmidt algorithm.
- Fractional QR with $\epsilon = 2^{-64}$.
- Fractional QR with $\epsilon = 2^{-256}$.

During the testing, we initialized a diagonal matrices with dyadic entries, compose it with a orthogonal matrix, and attempt to use QR factorization to retrieve the original dyadic numbers.



- We achieved a very good result when the result is larger than $\sqrt{\epsilon}$ and a reasonable result with some deviation when the result is larger than ϵ.

Discussion

Our implementation, although having higher precision, costs much more time than the traditional methods. This is expected, as the additions and multiplications are more complicated.
When thinking about our Fraction QR implementation with $\epsilon = 2^{-256}$, each addition is about 256 times of single bit addition, and each multiplication is about 6561 times of single bit addition. When doing the fractional field operation, each addition is composed of three integer multiplications and one integer addition, and each multiplication is composed of two multiplications. Hence, the operation is about 25 to 383 times slower than the floating-point operation. Hence, each addition and multiplication is about 25 to 383 times slower than the floating point operation.
The Fractional adaption of the algorithms is not only limited to QR factorization, and we can apply it to a wider class of algorithms: when we need precision, for some computations, there is no room for deviation, we shall consider it.

References

[1] David Bau III Lloyd N. Trefethen. *Numerical Linear Algebra*.