Randomized Householder-Cholesky QR Factorization with Multisketching

Andrew J. Higgins [2] Daniel B. Szyld [1]
Erik G. Boman [2] Ichitaro Yamazaki [2]

[1]Department of Mathematics, Temple University, Philadelphia

[2]Center for Computing Research, Sandia National Laboratories, Albuquerque

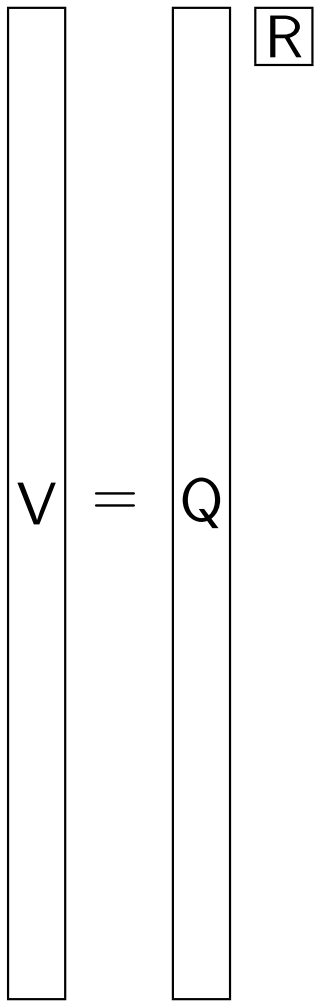
Workshop on Randomized Numerical Linear Algebra
ICERM
3 February 2026

Tall-and-skinny QR Factorizations

- Given $V \in \mathbb{R}^{n \times m}$, with $n \gg m$, we want to compute $V = QR$, with $Q^T Q = I$ and R $n \times n$ upper triangular.

Tall-and-skinny QR Factorizations

- Given $V \in \mathbb{R}^{n \times m}$, with $n \gg m$, we want to compute $V = QR$, with $Q^T Q = I$ and R $n \times n$ upper triangular.



Tall-and-skinny QR Factorizations

- Necessary for many scientific & engineering applications, including:
  - large least squares problems
  - dimensionality reduction methods for data analysis (e.g., PCA)
  - **block orthogonalization kernels for solving linear systems and eigenvalue problems within block or *s*-step Krylov methods**
    - require relatively robust and high performance TSQR
    - requires 1 TSQR per iteration
    - *s*-step methods produce matrix V formed using matrix powers kernel $\rightarrow$ ill-conditioned

Matrix power kernel for s-step GMRES

Matrix Powers Kernel: $V = \texttt{MPK}(A, b, s)$

1: $v_1 = Ab$
2: **for** $k = 2, \ldots, s$ **do**
3: $\quad v_k = Av_{(k-1)}$
4: **end for**

Standard approaches

- Gram-Schmidt
- Modified Gram-Schmidt
- Use Givens rotations
- Use Householder reflections

Gram-Schmidt

Jorgen P. Gram (1850-1916) a Danish actuary presented the orthogonalization method implicitly in 1883 (unaware that Pierre-Simon Laplace (1749-1827) had earlier presented the method).

Gram-Schmidt

Jorgen P. Gram (1850-1916) a Danish actuary presented the orthogonalization method implicitly in 1883 (unaware that Pierre-Simon Laplace (1749-1827) had earlier presented the method).
Erhard Schmidt (1876-1959), a student of Herman Schwarz and David Hilbert explicitly used the method in 1907.

Gram-Schmidt

Jorgen P. Gram (1850-1916) a Danish actuary presented the orthogonalization method implicitly in 1883 (unaware that Pierre-Simon Laplace (1749-1827) had earlier presented the method).
Erhard Schmidt (1876-1959), a student of Herman Schwarz and David Hilbert explicitly used the method in 1907.

Gram-Schmidt orthogonalization

1: $q_1 = v_1 / \|v_1\|$
2: **for** $i = 2, \ldots, m$ **do**
3: $\quad w_i = v_i$
4: $\quad$ **for** j=1, . . . , i-1 **do**
5: $\quad\quad r_{ji} = v_i^T q_j$
6: $\quad\quad w_i = w_i - r_{ji} q_j$
7: $\quad$ **end for**
8: $\quad r_{ii} = \|w_i\|$
9: $\quad q_i = w_i / r_{ii}$
10: **end for**

Gram-Schmidt is unstable. Use Modified Gram-Schmidt

- Gram-Schmidt is known to be unstable for numerical computations at lease since 1965 [James Wilkinson].

Gram-Schmidt is unstable. Use Modified Gram-Schmidt

- Gram-Schmidt is known to be unstable for numerical computations at lease since 1965 [James Wilkinson].
- Many people looked for alternatives and Modified Gram-Schmidt is a very good alternative. Essentially the same operations in a different order. Extensively studied by Åke Björk [LAA, 1994]

Gram-Schmidt is unstable. Use Modified Gram-Schmidt

- Gram-Schmidt is known to be unstable for numerical computations at lease since 1965 [James Wilkinson].
- Many people looked for alternatives and Modified Gram-Schmidt is a very good alternative. Essentially the same operations in a different order. Extensively studied by Åke Björk [LAA, 1994]

Modified Gram-Schmidt orthogonalization

1: **for** $i = 1, \ldots, m$ **do**
2: **end for**
3: $w_i = v_i$
4: **for** $i = 1, \ldots, m$ **do**
5: $\quad r_{ii} = \|w_i\|$
6: $\quad q_i = w_i / r_{ii}$
7: $\quad$ **for** j=i+1, . . . , m **do**
8: $\quad\quad r_{ij} = w_j^T q_i$
9: $\quad\quad w_j = w_j - r_{ij} q_i$
10: $\quad$ **end for**
11: **end for**

Householder Triangulation

Alton Householder [1958]: Apply orthogonal reflections to triangularize V, and obtain R.
The product of the reflections is Q^T. That is, $Q^T V = R$

Householder Triangulation

Alton Householder [1958]: Apply orthogonal reflections to triangularize V, and obtain R.
The product of the reflections is Q^T. That is, $Q^T V = R$
It is as stable as Modified Gram-Schmidt.
(Some more details below).

Our Goal for Tall-and-skinny QR Factorizations

- Numerically, we want an efficient algorithm with $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$ with few restrictions on V. $\mathbf{u}$ is the unit of round-off.

Our Goal for Tall-and-skinny QR Factorizations

- Numerically, we want an efficient algorithm with $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$ with few restrictions on V. $\mathbf{u}$ is the unit of round-off.

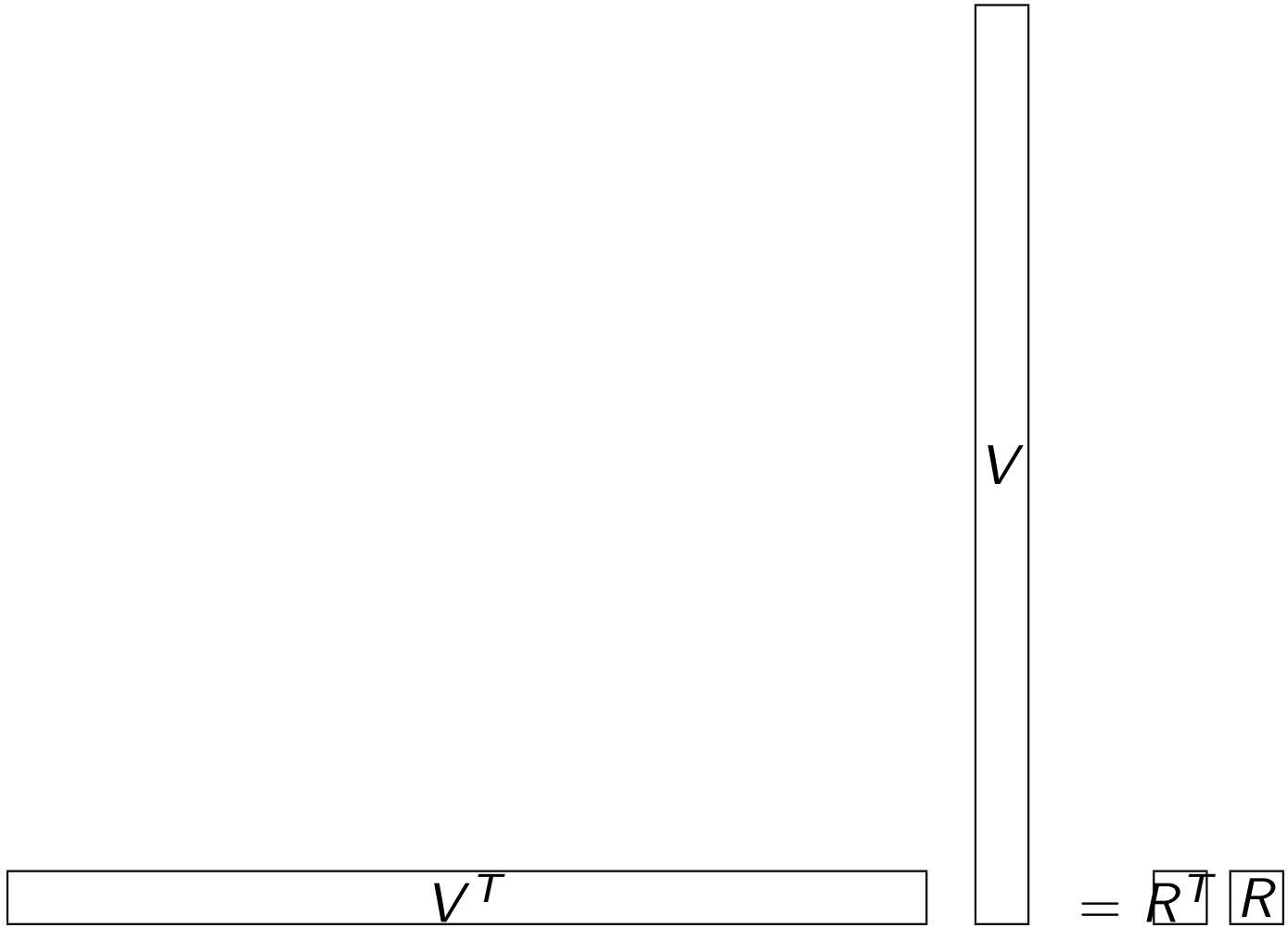
- For performance, we also want to limit communication

Our Goal for Tall-and-skinny QR Factorizations

- Numerically, we want an efficient algorithm with $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$ with few restrictions on V. $\mathbf{u}$ is the unit of round-off.
- For performance, we also want to limit communication
- First idea: Householder QR
  - \+ Does the job for any V
  - – Communication intensive $\rightarrow$ severe performance bottleneck

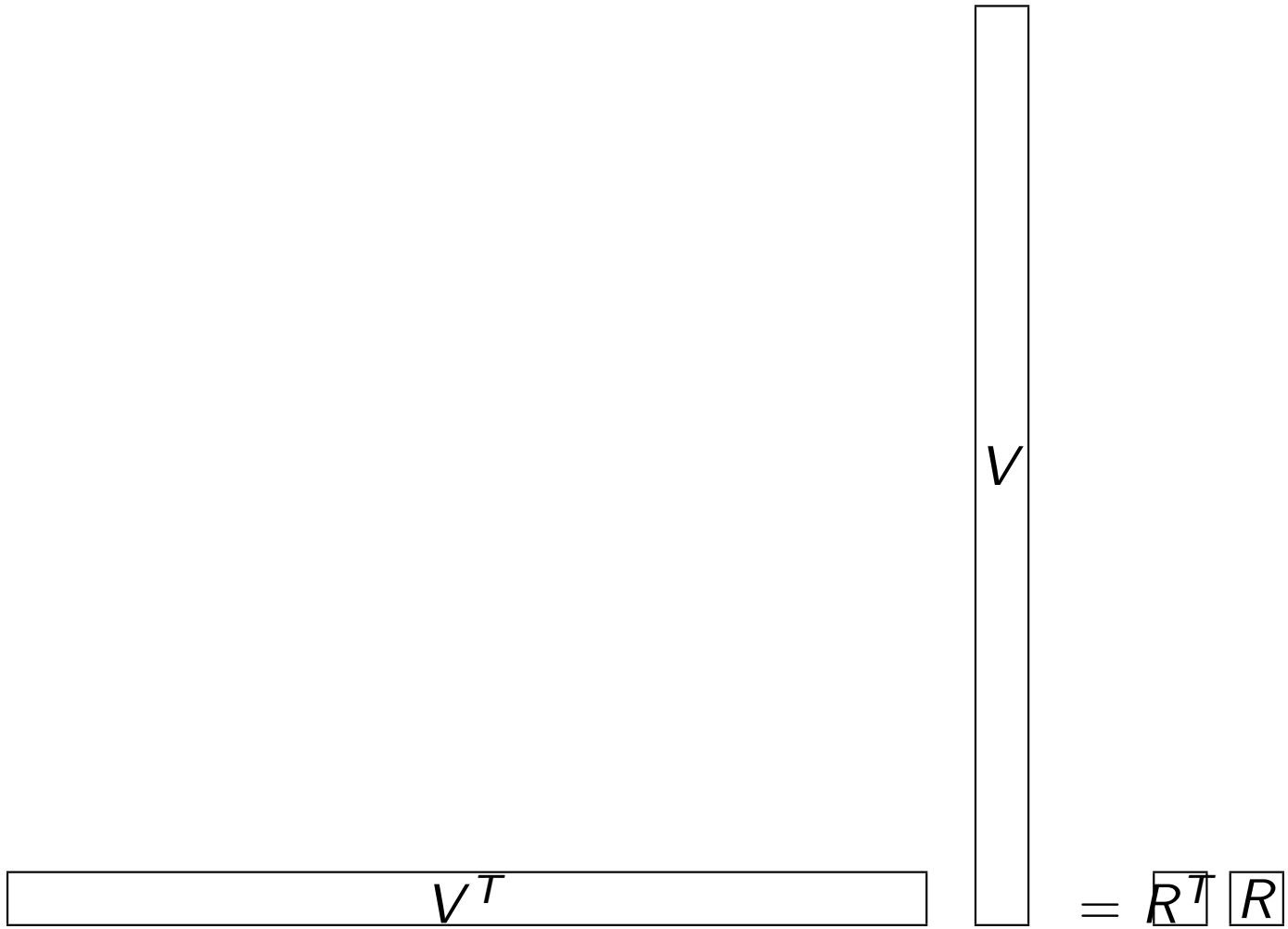
What to do?

First recall that if $V = QR$, then $V^T V = R^T Q^T QR = R^T R$.



What to do?

First recall that if $V = QR$, then $V^T V = R^T Q^T QR = R^T R$.



In other words R is the Cholesky factor of the $m \times m$ symmetric positive definite matrix $V^T V$.

What to do? (cont.)

Idea: Compute the Cholesky factorization of $V^T V$, obtaining R, then

What to do? (cont.)

Idea: Compute the Cholesky factorization of $V^T V$, obtaining R, then
Find $Q = VR^{-1}$ by solving m systems with triangular R.
Namely $R^T q_j^T = v_j^T$, (v_j, q_j rows of V, Q).

What to do? (cont.)

Idea: Compute the Cholesky factorization of $V^T V$, obtaining R, then
Find $Q = VR^{-1}$ by solving m systems with triangular R.
Namely $R^T q_j^T = v_j^T$, (v_j, q_j rows of V, Q).
This is called Cholesky QR (CholQR).

What to do? (cont.)

Idea: Compute the Cholesky factorization of $V^T V$, obtaining R, then
Find $Q = VR^{-1}$ by solving m systems with triangular R.
Namely $R^T q_j^T = v_j^T$, (v_j, q_j rows of V, Q).
This is called Cholesky QR (CholQR).
Some history: This idea was around in the 1970's and 80's.
Implicit in the classic book by Gene Golub and Charles Van Loan [First ed. 1983].

What to do? (cont.)

Idea: Compute the Cholesky factorization of $V^T V$, obtaining R, then
Find $Q = VR^{-1}$ by solving m systems with triangular R.
Namely $R^T q_j^T = v_j^T$, (v_j, q_j rows of V, Q).
This is called Cholesky QR (CholQR).
Some history: This idea was around in the 1970's and 80's.
Implicit in the classic book by Gene Golub and Charles Van Loan [First ed. 1983].
A paper by Walter Gander [ETH report 1980] indicates that CholQR more accurate (stable) than Classical Gram-Schmidt, but not as much as Modified GS, in other words:
This first idea is not very accurate!

What to do? (cont.)

Idea: Compute the Cholesky factorization of $V^T V$, obtaining R, then
Find $Q = VR^{-1}$ by solving m systems with triangular R.
Namely $R^T q_j^T = v_j^T$, (v_j, q_j rows of V, Q).
This is called Cholesky QR (CholQR).
Some history: This idea was around in the 1970's and 80's.
Implicit in the classic book by Gene Golub and Charles Van Loan [First ed. 1983].
A paper by Walter Gander [ETH report 1980] indicates that CholQR more accurate (stable) than Classical Gram-Schmidt, but not as much as Modified GS, in other words:
This first idea is not very accurate!
A personal note: I recall in 1986, Pete Stewart suggesting CholQR after a talk by Dianne O'Leary at a conference in Loen, Norway.

What to do? (cont.)

CholQR brought back in the context of modern architectures and high level BLAS by Stathopoulos and Wu [SISC, 2002]

What to do? (cont.)

CholQR brought back in the context of modern architectures and high level BLAS by Stathopoulos and Wu [SISC, 2002]
But they explicitly said: Do not use it!

What to do? (cont.)

Next idea:

Do it again: call what you did $V = Q_0 R_0$, do "Cholesky QR" on $Q_0 = Q_1 R_1$, obtaining $V = Q_1(R_1 R_0)$.

This is Cholesky QR2

[Fukaya, Nakatsukasa, Yanagisawa, Yamamoto, 2014].

What to do? (cont.)

Next idea:

Do it again: call what you did $V = Q_0 R_0$, do "Cholesky QR" on $Q_0 = Q_1 R_1$, obtaining $V = Q_1(R_1 R_0)$.

This is Cholesky QR2

[Fukaya, Nakatsukasa, Yanagisawa, Yamamoto, 2014].

You can think of this as "iterative improvement" for QR.

Current Practical Communication-Avoiding Tall-and-skinny QR Factorizations

In other words,

- Alternative 1 : CholeskyQR

CholeskyQR:
$[Q, R] = \texttt{cholQR}(V)$

1: $G = V^T V$
2: $R = \texttt{chol}(G)$
3: $Q = VR^{-1}$

Current Practical Communication-Avoiding Tall-and-skinny QR Factorizations

In other words,

- Alternative 1 : CholeskyQR

CholeskyQR:
$[Q, R] = \texttt{cholQR}(V)$

1: $G = V^T V$
2: $R = \texttt{chol}(G)$
3: $Q = VR^{-1}$

+ Only 1 processor communication
– Inaccurate: high orthog. error $||I - Q^T Q|| = O(\kappa^2(V)\ \mathbf{u})$
[Yamamoto, Nakatsukasa, Yanagisawa, Fukaya, 2015]

Another popular idea: CholeskyQR2

CholeskyQR2:
$[Q, R] = \texttt{cholQR2}(V)$

1: $[Q_0, R_0] = \texttt{cholQR}(V)$
2: $[Q, R_1] = \texttt{cholQR}(Q_0)$
3: $R = R_1 R_0$

Another popular idea: CholeskyQR2

CholeskyQR2:
$[Q, R] = \texttt{cholQR2}(V)$

1: $[Q_0, R_0] = \texttt{cholQR}(V)$
2: $[Q, R_1] = \texttt{cholQR}(Q_0)$
3: $R = R_1 R_0$

+ Uses highly parallelizable matrix operations only, 2 communications
– Unstable: requires $\kappa^2(V) \lessapprox \mathbf{u}^{-1}$ to achieve $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$
[Yamamoto, Nakatsukasa, Yanagisawa, Fukaya, 2015]
cf. E. Carson, K. Lund, M. Rozložník, and S. Thomas [LAA, 2022] for the block case

Newer idea: shifted CholeskyQR3

Shifted CholeskyQR3:

$[Q, R] = \texttt{sCholQR3}(V, \omega)$

1: $G = V^T V + \omega I$
2: $R = \texttt{chol}(G)$
3: $Q_0 = V R_0^{-1}$
4: $[Q, R_1] = \texttt{cholQR2}(Q_0)$
5: $R = R_1 R_0$

Newer idea: shifted CholeskyQR3

Shifted CholeskyQR3:
$[Q, R] = \texttt{sCholQR3}(V, \omega)$

1: $G = V^T V + \omega I$
2: $R = \texttt{chol}(G)$
3: $Q_0 = V R_0^{-1}$
4: $[Q, R_1] = \texttt{cholQR2}(Q_0)$
5: $R = R_1 R_0$

\+ More stable: for appropriately chosen shift ω, requires $\kappa(V) \lesssim \mathbf{u}^{-1}$ to achieve $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$
[Fukaya, Kannan, Nakatsukasa, Yamamoto, Yanagisawa, 2020]

– 50% higher communication and computational cost of CholeskyQR2

An Ideal Tall-and-Skinny QR Algorithm

Ideally, we would like a QR algorithm that:

- Has CholeskyQR2's low computational cost with similar operations
  - This would give same number of communications as CholeskyQR2

An Ideal Tall-and-Skinny QR Algorithm

Ideally, we would like a QR algorithm that:

- Has CholeskyQR2's low computational cost with similar operations
  - This would give same number of communications as CholeskyQR2
- Has better stability properties than CholeskyQR2, maybe similar to shifted CholeskyQR3
  - e.g., for any numerically full rank V (i.e., $\kappa(V) \lessapprox \mathbf{u}^{-1}$), $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$

An Ideal Tall-and-Skinny QR Algorithm

Ideally, we would like a QR algorithm that:

- Has CholeskyQR2's low computational cost with similar operations
  - This would give same number of communications as CholeskyQR2
- Has better stability properties than CholeskyQR2, maybe similar to shifted CholeskyQR3
  - e.g., for any numerically full rank V (i.e., $\kappa(V) \lessapprox \mathbf{u}^{-1}$), $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$

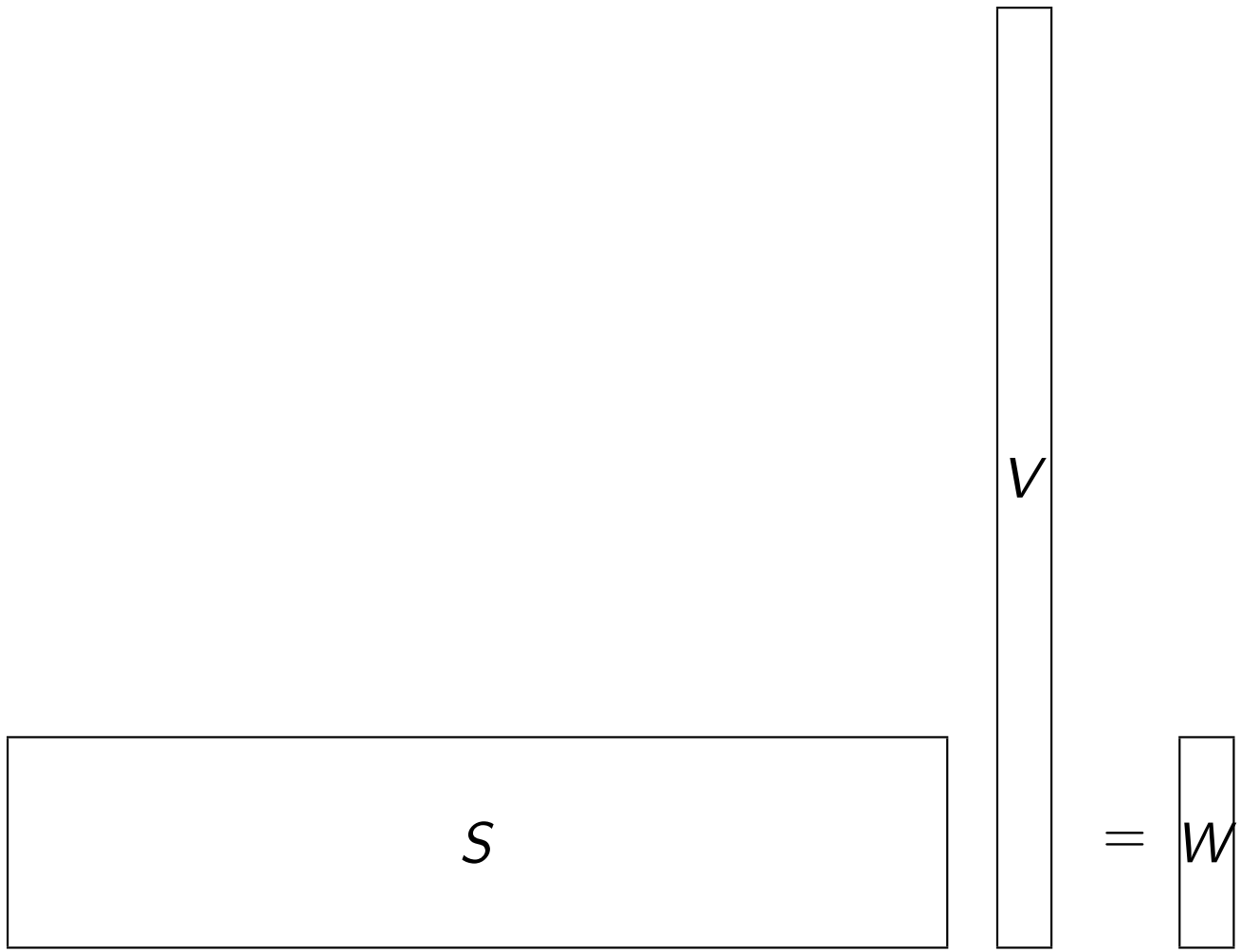
Tool to accomplish this: *random sketching*

Random Sketching

We can use a random *sketch matrix* $S \in \mathbb{R}^{s \times n}$ for $s \ll n$ to compress $V \in \mathbb{R}^{n \times m}$ while approximately preserving its condition number. $W = SV$, $\kappa(W) \approx \kappa(V)$ with high probability.

Random Sketching

We can use a random *sketch matrix* $S \in \mathbb{R}^{s \times n}$ for $s \ll n$ to compress $V \in \mathbb{R}^{n \times m}$ while approximately preserving its condition number. $W = SV$, $\kappa(W) \approx \kappa(V)$ with high probability.



Random Sketching (cont.)

Definition ((ε, d, m) oblivious ℓ_2-subspace embedding)

The sketch matrix $S \in \mathbb{R}^{s \times n}$ is an (ε, d, m) *oblivious ℓ_2-subspace embedding* if for any fixed m-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$,

$$\sqrt{1-\varepsilon}\,\|x\|_2 \leq \|Sx\|_2 \leq \sqrt{1+\varepsilon}\,\|x\|_2, \quad \forall x \in \mathcal{V}$$

with probability at least $1 - d$.

Random Sketching (cont.)

Definition ((ε, d, m) oblivious ℓ_2-subspace embedding)

The sketch matrix $S \in \mathbb{R}^{s \times n}$ is an (ε, d, m) *oblivious ℓ_2-subspace embedding* if for any fixed m-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$,

$$\sqrt{1-\varepsilon}\,\|x\|_2 \leq \|Sx\|_2 \leq \sqrt{1+\varepsilon}\,\|x\|_2, \quad \forall x \in \mathcal{V}$$

with probability at least $1 - d$.

Corollary

If $S \in \mathbb{R}^{s \times n}$ is an (ε, d, m) oblivious ℓ_2-subspace embedding, and $V \in \mathbb{R}^{n \times m}$ has rank m, then with probability at least $1 - d$,

$$\kappa(V) \leq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\,\kappa(SV).$$

[Sarlos,2006]

Idea of a Randomized QR Algorithm

Given a sketch $S \in \mathbb{R}^{s \times n}$ and $V \in \mathbb{R}^{n \times m}$, with $s \ll n$, $m \ll n$, consider:

Randomized QR Framework

1: Apply random sketch to compress V: $W = SV \in \mathbb{R}^{s \times m}$
2: Apply Householder QR to W: $[Q_{tmp}, R_0] = \text{HouseholderQR}(W)$
3: Approximately orthogonalize V: $Q_0 = VR_0^{-1}$
4: Reorthogonalize: $[Q, R_1] = \texttt{cholQR}(Q_0)$
5: Return R: $R = R_1 R_0$

Idea of a Randomized QR Algorithm

Given a sketch $S \in \mathbb{R}^{s \times n}$ and $V \in \mathbb{R}^{n \times m}$, with $s \ll n$, $m \ll n$, consider:

Randomized QR Framework

1: Apply random sketch to compress V: $W = SV \in \mathbb{R}^{s \times m}$
2: Apply Householder QR to W: $[Q_{tmp}, R_0] = \text{HouseholderQR}(W)$
3: Approximately orthogonalize V: $Q_0 = VR_0^{-1}$
4: Reorthogonalize: $[Q, R_1] = \texttt{cholQR}(Q_0)$
5: Return R: $R = R_1 R_0$

- Does it work?
  - $\kappa(Q_0) = \kappa(VR_0^{-1}) \leq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\, \kappa(SVR_0^{-1}) = \sqrt{\frac{1-\varepsilon}{1+\varepsilon}} = O(1)$
  - Since $\kappa(Q_0) = O(1)$, step 4 gives fully orthogonal Q (up to $\mathbf{u}$).

Idea of a Randomized QR Algorithm

Given a sketch $S \in \mathbb{R}^{s \times n}$ and $V \in \mathbb{R}^{n \times m}$, with $s \ll n$, $m \ll n$, consider:

Randomized QR Framework

1: Apply random sketch to compress V: $W = SV \in \mathbb{R}^{s \times m}$
2: Apply Householder QR to W: $[Q_{tmp}, R_0] = \text{HouseholderQR}(W)$
3: Approximately orthogonalize V: $Q_0 = VR_0^{-1}$
4: Reorthogonalize: $[Q, R_1] = \texttt{cholQR}(Q_0)$
5: Return R: $R = R_1 R_0$

- Does it work?
  - $\kappa(Q_0) = \kappa(VR_0^{-1}) \leq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\, \kappa(SVR_0^{-1}) = \sqrt{\frac{1-\varepsilon}{1+\varepsilon}} = O(1)$
  - Since $\kappa(Q_0) = O(1)$, step 4 gives fully orthogonal Q (up to $\mathbf{u}$).
- Is it a good idea?
  - **If** S reduces the size of V sufficiently in step 1, step 2 is cheap
  - **If** sketching is cheap, similar arithmetic cost to CholeskyQR2, and also only requires 2 communications $\Rightarrow$ similar performance to CholeskyQR2
  - Good idea **if** sketching is cheap and $W = SV$ is sufficiently small

Simple Efficient Examples of (ε, d, m) oblivious ℓ_2-subspace embeddings

- Gaussian Sketch: $S = \frac{1}{\sqrt{s}} G \in \mathbb{R}^{s \times n}$ where $g_{i,j}$ are i.i.d. Gaussian random variables
  - \+ Requires sketch size $s = O(m)$
    - random QR framework does HouseholderQR on small $O(m) \times m$ matrix $\rightarrow$ fast
  - – Applying to $V \in \mathbb{R}^{n \times m}$ has $O(nm^2)$ complexity
    - Sketching cost dominates

Simple Efficient Examples of (ε, d, m) oblivious ℓ_2-subspace embeddings

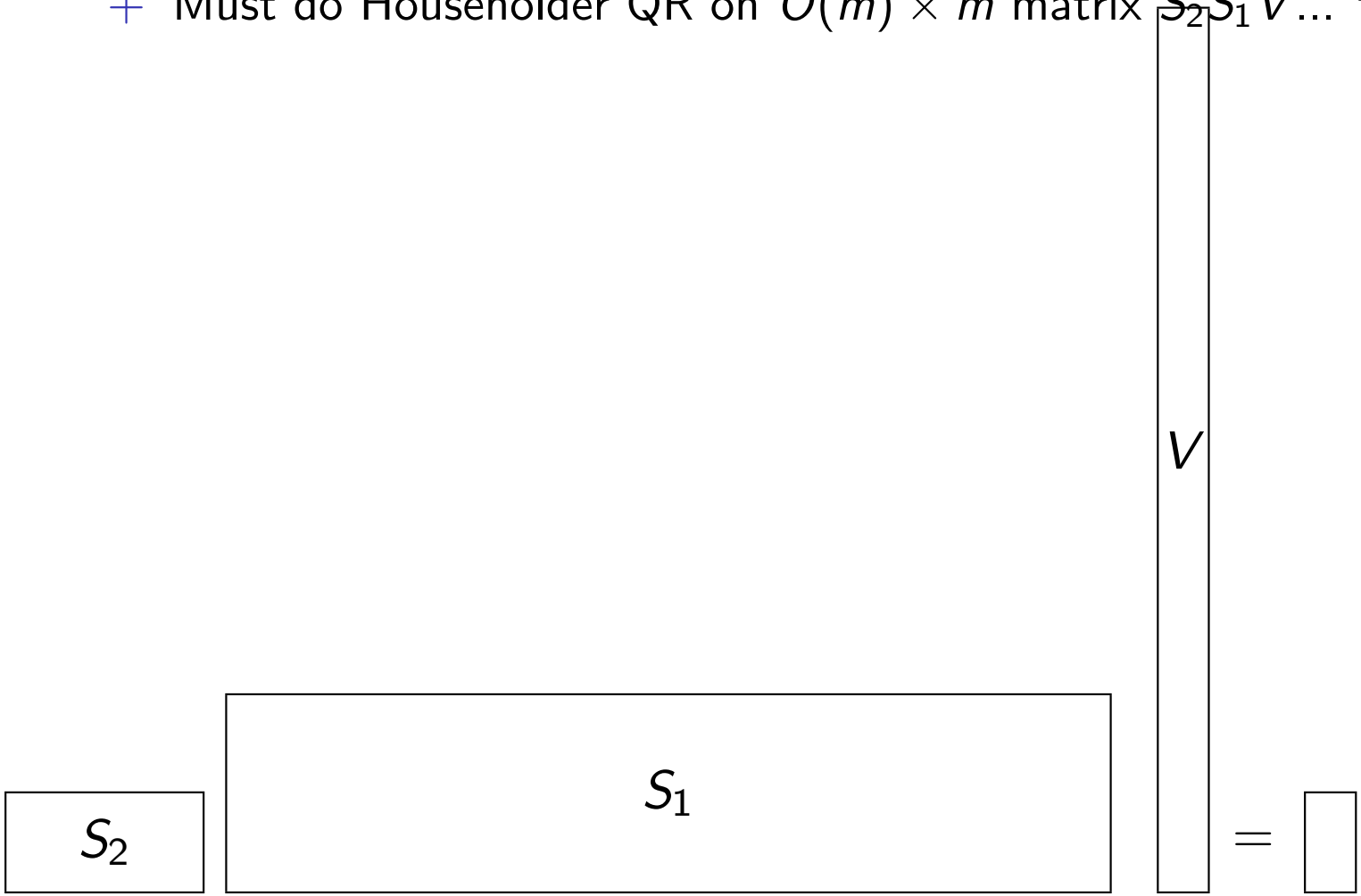
- Gaussian Sketch: $S = \frac{1}{\sqrt{s}} G \in \mathbb{R}^{s \times n}$ where $g_{i,j}$ are i.i.d. Gaussian random variables
  - \+ Requires sketch size $s = O(m)$
    - random QR framework does HouseholderQR on small $O(m) \times m$ matrix $\rightarrow$ fast
  - – Applying to $V \in \mathbb{R}^{n \times m}$ has $O(nm^2)$ complexity
    - Sketching cost dominates
- CountSketch: $S = \frac{1}{\sqrt{s}} C \in \mathbb{R}^{s \times n}$ where C is sparse with one ± 1 per column placed in a random row
  - \+ Applying to $V \in \mathbb{R}^{n \times m}$ has $O(nm)$ complexity
    - Sketching is cheap
  - – Requires sketch size $s = O(m^2)$
    - random QR framework must do HouseholderQR on $O(m^2) \times m$ matrix $\rightarrow$ potential bottleneck

Randomized Householder-Cholesky QR

- What if we use two sketches S_1, S_2 where we apply a $S_1 \in \mathbb{R}^{O(m^2) \times n}$ CountSketch matrix to V and then reduce the size again with a smaller $S_2 \in \mathbb{R}^{O(m) \times O(m^2)}$ Gaussian
  - \+ Sketching is cheap (same asymptotic cost as 1 CountSketch)
  - \+ Must do Householder QR on $O(m) \times m$ matrix $S_2 S_1 V$... fast!

Randomized Householder-Cholesky QR

- What if we use two sketches S_1, S_2 where we apply a $S_1 \in \mathbb{R}^{O(m^2) \times n}$ CountSketch matrix to V and then reduce the size again with a smaller $S_2 \in \mathbb{R}^{O(m) \times O(m^2)}$ Gaussian
  - \+ Sketching is cheap (same asymptotic cost as 1 CountSketch)
  - \+ Must do Householder QR on $O(m) \times m$ matrix $S_2 S_1 V$... fast!



Randomized Householder-Cholesky QR

- We call this algorithm `rand_cholQR`
  - Is this actually more numerically stable than CholeskyQR2?
  - Is its performance similar to CholeskyQR2 in practice?

Randomized Householder-Cholesky QR

- We call this algorithm `rand_cholQR`
  - Is this actually more numerically stable than CholeskyQR2?
  - Is its performance similar to CholeskyQR2 in practice?

Randomized Householder-Cholesky QR: [Q,R] = `rand_cholQR`(V)

1: Apply random sketches to compress V: $W = S_2 S_1 V \in \mathbb{R}^{s_2 \times m}$
2: Apply Householder QR to W: $[Q_{tmp}, R_0] = \text{HouseholderQR}(W)$
3: Approximately orthogonalize V: $Q_0 = V R_0^{-1}$
4: Reorthogonalize: $[Q, R_1] = \texttt{cholQR}(Q_0)$
5: Return R: $R = R_1 R_0$

Key Numerical Properties of `rand_cholQR`

First, we define a set of assumptions stating V is numerically full rank (i.e., $\kappa(V) \leq O(\mathbf{u}^{-1})$), V is tall-and-skinny, and that the sketch matrices S_1, S_2 simultaneously satisfy the subspace embedding properties with probability at least $1 - d$.

Key Numerical Properties of `rand_cholQR`

First, we define a set of assumptions stating V is numerically full rank (i.e., $\kappa(V) \leq O(\mathbf{u}^{-1})$), V is tall-and-skinny, and that the sketch matrices S_1, S_2 simultaneously satisfy the subspace embedding properties with probability at least $1 - d$.

Assumptions (Matrix Conditioning and Embedding)

Suppose $S_1 \in \mathbb{R}^{p_1 \times m}$ *and* $S_2 \in \mathbb{R}^{p_2 \times p_1}$ *are* (ε_1, d_1, m) *and* (ε_2, d_2, m) *oblivious* ℓ_2*-subspace embeddings respectively, generated independently. Define* $d = d_1 + d_2 - d_1 d_2$, $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$, *where* $\varepsilon_L \in \left[0, \frac{616}{625} - \frac{9}{625}\varepsilon_H\right)$. *Further, suppose* $V \in \mathbb{R}^{n \times m}$ *has full rank and* $1 < m \leq p_2 \leq p_1 \leq n$ *where* $nm\mathbf{u} \leq \frac{1}{12}$, $p_1\sqrt{p_2}\mathbf{u} \leq \frac{1}{12}$, *and*

$$\delta := \frac{383\left(\sqrt{1+\varepsilon_H}\; p_2 m^{3/2} + \sqrt{m}\|S_2\|_2(p_1\sqrt{p_2}\sqrt{1+\varepsilon_1} + n\|S_1\|_F)\right)}{\sqrt{1-\varepsilon_L}}\mathbf{u}\,\kappa(V) \leq 1.$$

Key Numerical Properties of `rand_cholQR`

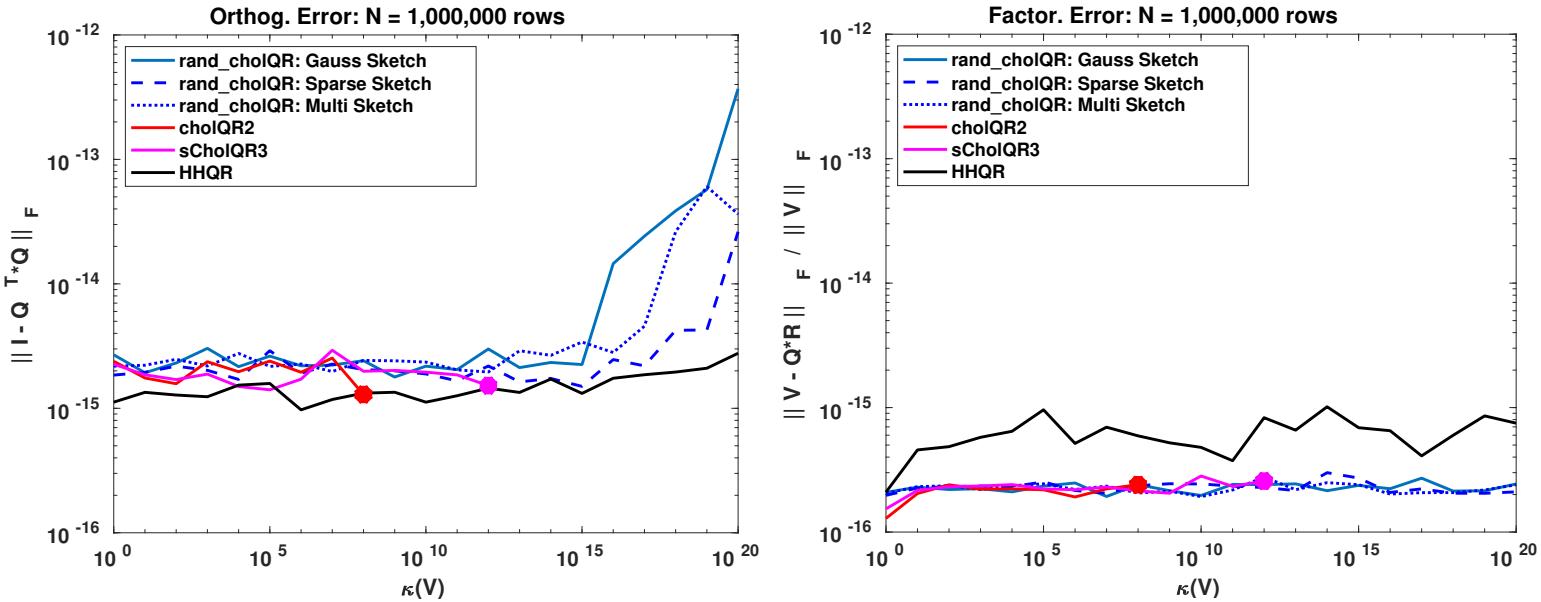
Theorem (Higgins, S., Boman, Yamazaki, 2023)

Suppose the matrix conditioning and embedding assumptions are satisfied. With probability at least $1 - d$*, the computed* $\hat{Q}, \hat{R}$ *factors obtained with* `rand_cholQR` *has* $O(\mathbf{u})$ *orthogonality error and relative factorization error. That is,*

$$\|\hat{Q}^T \hat{Q} - I\|_2 \leq c_1(n, m, \varepsilon_L, \varepsilon_H)\ \mathbf{u},$$
$$\|V - \hat{Q}\hat{R}\|_2 / \|V\|_2 \leq c_2(n, m, \varepsilon_L, \varepsilon_H)\ \mathbf{u}$$

Empirical Verification of Numerical Properties

Tested stability of Multi-Sketch `rand_cholQR`, `cholQR2`, `sCholQR3`, and Householder QR



(a) Orthogonality Error (b) Relative Factorization Error

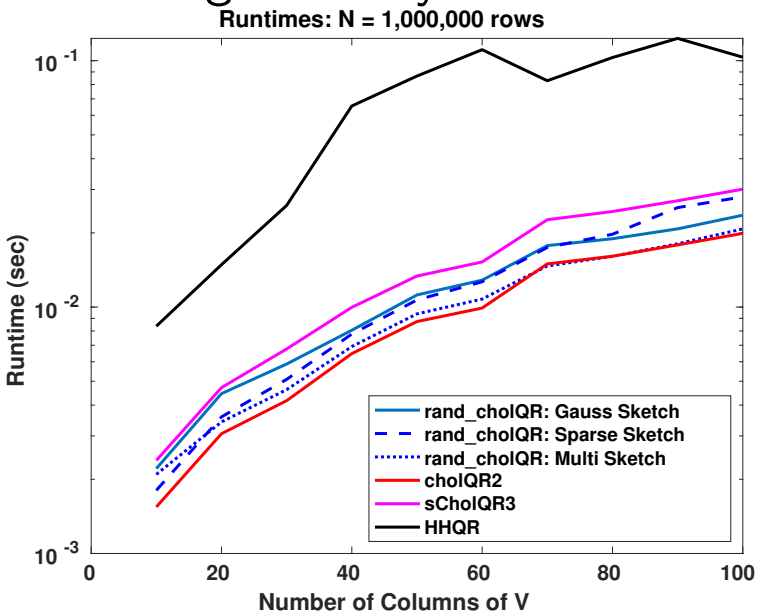
Figure: Orthogonality (left) and relative factorization error (right), $m = 70$. Indicated by a large dot, lines for `cholQR2` end at $\kappa(V) = 10^8$, as the method fails beyond this point.

Parameters for our Numerical Experiments for Multisketching

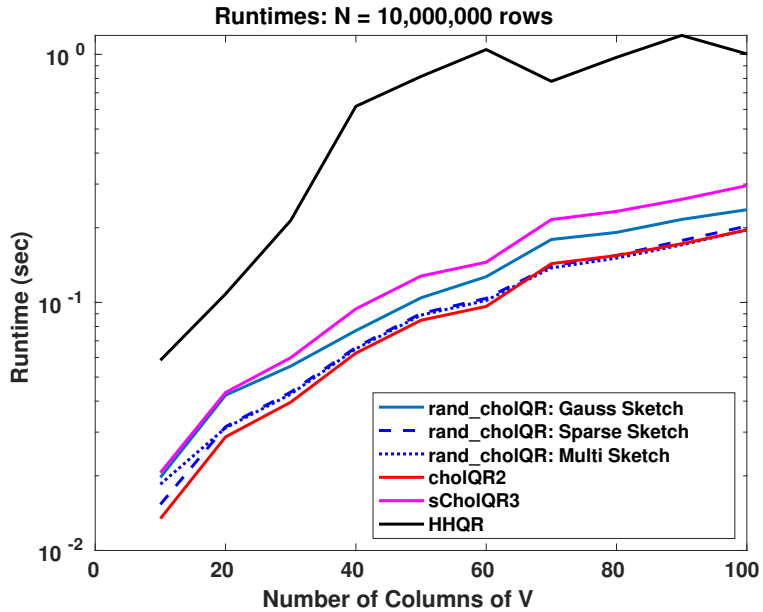
- $S_1 \in \mathbb{R}^{p_1 \times n}$ a CountSketch with $\varepsilon_1 = 0.9$ requiring sketch size $p_1 = \lceil 8.24(m^2 + m) \rceil$ to produce a $(0.9, 0.15, m)$ oblivious ℓ_2-subspace embedding
- $S_2 \in \mathbb{R}^{p_2 \times p_1}$ a Gaussian sketch with $p_2 = \lceil 74.3 \log(p_1) \rceil$ giving $\varepsilon_2 = 0.49$ to give a $(0.49, 1/m, m)$ oblivious ℓ_2-subspace embedding.
- $S_2 S_1$ produced an embedding with $\varepsilon_L \approx 0.9490$, $\varepsilon_H \approx 1.8310$, and $d \approx 0.15$
- Thus, our assumptions are satisfied

Performance Results

- We tested performance of multi-sketch `rand_cholQR` vs `rand_cholQR` with a single Gaussian sketch and single CountSketch, `cholQR2`, `sCholQR3`, and Householder QR on an NVIDIA A100 GPU using the Kokkos C++ library, and a few direct calls to cuSOLVER.
- Multi-sketched `rand_cholQR` is:
  - significantly more stable than `cholQR2` at no cost (sometimes faster!)
  - significantly faster than Householder QR and `sCholQR3`



(a) $n = 1,000,000$ rows



(b) $n = 10,000,000$ rows

Conclusions

Using multi-sketching, `rand_cholQR` has:

- CholeskyQR2's low computational cost with similar operations
  - Gives same number of communications, gives similar performance in practice on a high performance GPU
- significantly better stability properties than CholeskyQR2 in theory and practice
  - i.e., for any numerically full rank V, $||I - Q^T Q|| = O(\mathbf{u})$ and $\|V - QR\|/\|V\| = O(\mathbf{u})$
- better stability properties than `sCholQR3` in practice with 50% less communication and $\sim 50\%$ faster runtime on a GPU

Best of all: very easy to get high performance with standard libraries

Reference

With Andrew Higgins, Erik Boman, and Ichitaro Yamazaki, Analysis of Randomized Householder-Cholesky QR Factorization with Multisketching [*Numerische Mathematik* **157** (2025) 1695–1737]

On ArXiv, or on https://faculty.cst.temple.edu/~szyld

Ad 1: Poster TODAY by Andrew Higgins: "A High Performance GPU CountSketch Implementation and Application to Least Squares Problems"

Ad 2: Temple Math is looking for a chair: https://www.mathjobs.org/jobs/list/28041