“No-I-Meant-Another QR” (NIMA): Two-Stage Newton–Schulz–Refined Mixed-Precision QR

BLIS Retreat 2025

Nima Sahraneshin Samani
Supervisors: Sandra Catalan José R. Herrero José Ignacio Aliaga

September 25, 2025

Fast QR in FP16 — without losing orthogonality.

Why is Mixed-Precision QR Worth It?

- GPUs deliver $\sim 10\times$ more throughput in **FP16**, yet naive FP16-QR destroys orthogonality.
- We want **FP16** *speed* with near-**FP64** *accuracy*.
- Key: cheap Householder in FP16, enforce orthogonality with Newton–Schulz (NS) refinements in BLAS-3.

Can FP16 QR be “trustworthy by design”?

Road-map

1. Motivation and contributions
2. How GPUs reward BLAS-3, not BLAS-2
3. NIMA in a nutshell (family & algorithm)
4. Accuracy and stability results
5. Performance results
6. Take-aways & Q & A

Contributions

1. Two-stage NS guard: optional *local* (panel) + *global* sweeps.
2. Family of variants from “unsafe FP16” to “FP16 panels + FP64 updates”.
3. Open-source MATLAB reference implementation.

Classic QR vs GPUs: Why Push Work to BLAS-3?

- Householder panel factorisation is BLAS-2: memory-bound, poor GPU utilisation.
- BLAS-3 (SYRK/GEMM) is compute-dense: exploits caches and Tensor Cores.
- Strategy: 1) do the *cheap* part (panels) in low precision; 2) fix orthogonality with a few BLAS-3 NS sweeps.
- More FLOPs, yet lower wall-time on GPUs.

BLAS-3 in NIMA

Local guard (panel $Q_p \in \mathbb{R}^{m \times b}$)

SYRK: $G_p \leftarrow Q_p^\top Q_p$, GEMM: $Q_p \leftarrow \frac{1}{2} Q_p (3I - G_p)$

Global guard (thin $Q \in \mathbb{R}^{m \times n}$)

SYRK: $G \leftarrow Q^\top Q$, GEMM: $Q \leftarrow \frac{1}{2} Q (3I - G)$

Both operations map perfectly to Tensor Cores.

NIMA Family – At a Glance

Variant	Panel	Trailing	Guard strategy
H-QR	FP16	FP16	none (stress test)
H-QR-G	FP16	FP16	global NS ($g=3$–4)
HF-QR-G	FP16(+FP32 dots)	FP16	global NS
HF-QR-LG	FP16(+FP32 dots)	FP16	local $s=1$ + global $g=3$
HFD-QR-LG	FP16 → FP64 upd.	FP64	local $s=2$ + global $g=3$
D-QR (baseline)	FP64	FP64	—

Blocked Pipeline with Two-Stage NS

NIMA (*b* panel size, *s* local, *g* global NS iters)

Require: $A \in \mathbb{R}^{m \times n},\ m \geq n$
for $j = 1 : b : n$ **do**
  factor panel in low precision
  apply reflectors to $A(:, j{:}n)$ and to Q
  if $s > 0$ **then** ▷ local NS
    extract new b cols of Q; run s NS steps
  end if
end for
thin Q: keep first n columns
for $k = 1 : g$ **do** ▷ global NS
  $Q \leftarrow \frac{1}{2} Q\left(3I - Q^\top Q\right)$
end for
$R \leftarrow Q^\top A_{\text{orig}}$ ▷ FP64 product
enforce R triangular + $\text{diag}(R) \geq 0$
return (Q, R)

Expected Accuracy After Global NS

Variant	Orthogonality	Residual $\|\|A - QR\|\|$	Robustness
H-QR	poor–fair	poor–fair	weakest
H-QR-G	good–excellent	good–excellent	moderate
HF-QR-G	excellent	excellent	high
HF-QR-LG	excellent	excellent	high
HFD-QR-LG	excellent	excellent	highest
D-QR	FP64 baseline	FP64 baseline	reference

Stability of Newton–Schulz Orthogonalisation

- **NS iteration:** $\boldsymbol{Q_{k+1} = \frac{1}{2} Q_k (3I - Q_k^\top Q_k)}$ **– quadratic if** $\boldsymbol{Q_0^\top Q_0}$ **eigenvalues** $\in (0, 2)$**.**
- **Local NS keeps each panel inside that basin; global sweeps drive** $\boldsymbol{\|I - Q^\top Q\|_2}$ **to** $\mathcal{O}(\varepsilon)$**.**
- **Exact power-of-two column scaling prevents FP16 overflow/underflow.**

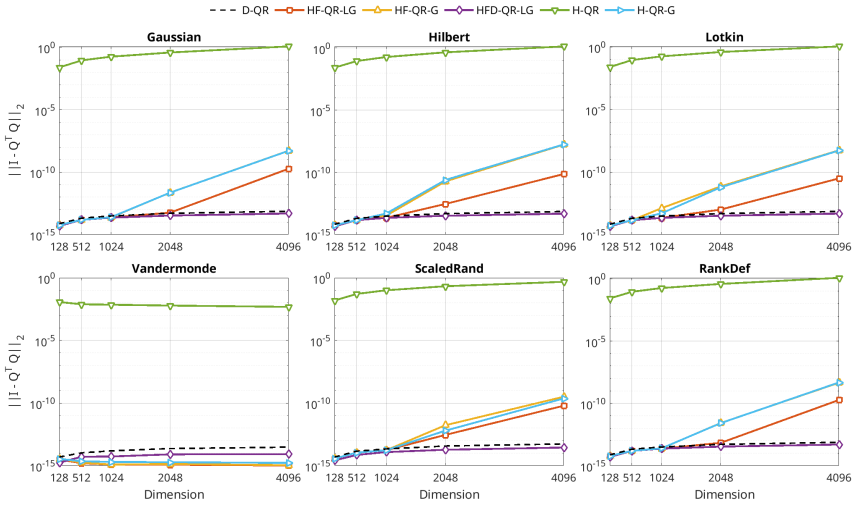
Representation: WY vs Explicit Q

Key point

Pure WY is awkward for NS: the GEMM needs Q explicitly.

- Store WY in **FP16** during panels (fast, compact).
- Form Q explicitly once a panel is done.
- R refinement is optional; many workflows only need high-quality Q.

Orthogonality Tracks MATLAB QR



Accuracy Summary (n =1k–4k)

n	$\kappa_2^{\max}$	max $\|A - QR\|_2 / \|A\|_2$	max **post-**$\|I - Q^\top Q\|_2$	max **pre-**$\|I - Q^\top Q\|_2$
1024	3.1×10^{28}	1.3×10^{-15}	4.9×10^{-15}	2.7×10^{-2}
2048	6.0×10^{28}	1.3×10^{-15}	4.8×10^{-15}	6.0×10^{-2}
3072	4.2×10^{32}	1.6×10^{-15}	5.2×10^{-15}	8.6×10^{-2}
4096	2.2×10^{32}	1.7×10^{-15}	5.3×10^{-15}	1.2×10^{-1}

Hilbert

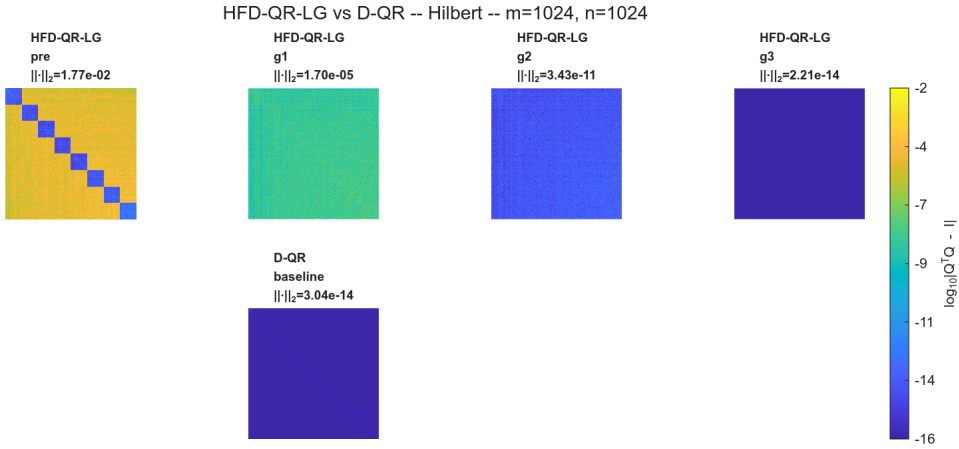


Figure: NS refinement (local/global) for Hilbert $n = 1024$

Note: Updated from recorded version—new random instance and improved visualization format.

Vandermonde

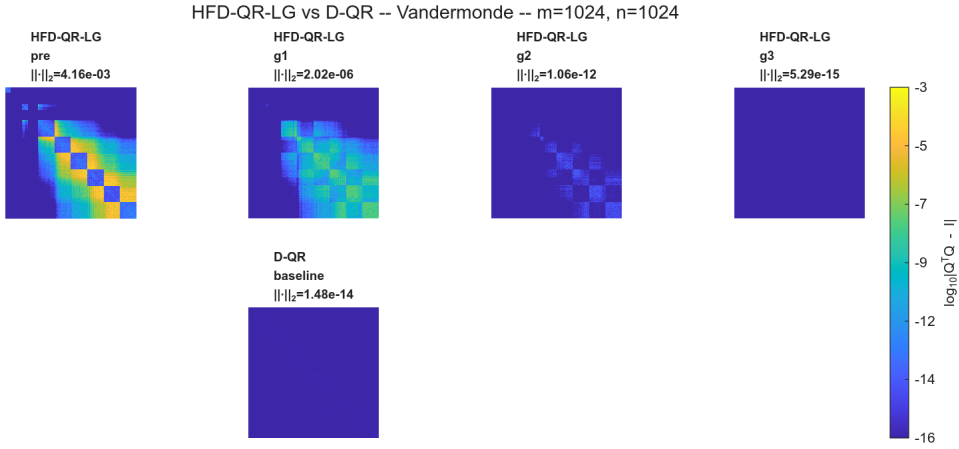


Figure: NS refinement (local/global) for Vandermonde $n = 1024$

Note: Updated from recorded version—new random instance and improved visualization format.

Projected Performance Model (Guarded FP16)

Model

$$\rho = \frac{\text{FP16 peak}}{\text{FP64 peak}}, \quad T(n;\rho,g,s,b) = \frac{T_{\text{QR64}}(n)}{\rho} + g\,T_{\text{NS}}(n) + s\,c_{\text{loc}}\left\lceil \frac{n}{b} \right\rceil, \quad c_{\text{loc}} = 0.2\ \text{ms}$$

$$\text{Speed}-\text{up}(n;\rho,g,s) = \frac{1}{\frac{1}{\rho} + \Delta_{s,g}(n)}, \quad \Delta_{s,g}(n) = \frac{g\,T_{\text{NS}}(n) + s\,c_{\text{loc}}\lceil n/b \rceil}{T_{\text{QR64}}(n)}.$$

- V100-like ratio $\rho \approx 3$: up to $\sim 3\times$ speed-up for $n \lesssim 4096$.
- Higher ratios ($\rho \in [6, 8]$): up to $\sim 5\times$ around $n \approx 8192$.
- For very large n, global NS sweeps dominate unless g is reduced.

Measured Runtime on V100 (H-QR-L)

n	MAGMA QR	orgqr	NS	D2H	H2D	R-hQR	Speed-up
1024	461.7	18.0	1.2	0.1	0.1	175.6	2.7
2048	1448.3	43.6	5.4	0.1	0.1	542.6	2.7
3072	1873.0	12.9	17.2	0.2	0.2	689.0	2.7
4096	1128.5	24.3	35.7	0.3	0.3	508.1	2.3
5120	1648.4	45.1	67.4	0.4	0.4	797.4	2.1
6144	2007.3	64.5	112.3	0.5	0.5	1071.4	1.9
7168	2827.6	97.9	187.9	0.7	0.7	1605.6	1.8
8192	2752.3	133.9	280.9	0.9	0.9	1895.8	1.5
9216	2977.9	189.6	407.2	1.2	1.2	2406.2	1.3
10240	4544.3	264.7	558.8	1.5	1.5	3458.9	1.4
11264	5366.9	380.1	757.1	1.8	1.8	4444.0	1.3
12288	5878.6	444.1	989.2	2.2	2.2	5375.4	1.2
13312	6253.7	547.2	1267.8	2.5	2.5	6440.2	1.1
14336	7162.3	695.7	1580.2	2.8	2.8	7829.3	1.0
15360	8970.6	824.8	1971.0	3.2	3.2	9734.3	1.0
16384	10033.7	1032.0	2359.6	3.7	3.7	11462.6	1.0
17408	11206.1	1229.5	2886.4	4.4	4.4	13632.7	0.9
18432	12962.7	1396.7	3434.8	5.0	5.0	16031.9	0.9
19456	14272.5	1655.6	4059.4	5.4	5.4	18602.1	0.9
20480	19241.3	1930.0	4789.1	5.7	5.7	22722.4	0.9

- Peak speed-up ≈2.7× for $n \leq 4{,}096$ matches the analytic model.
- As n grows beyond 12k, global NS cost dominates and speed-up falls to parity.

Take-aways

- FP16 panels *can* be made as trustworthy as FP64 with two-stage NS.
- BLAS-3 NS sweeps convert “extra FLOPs” into GPU utilisation – wall-time wins.

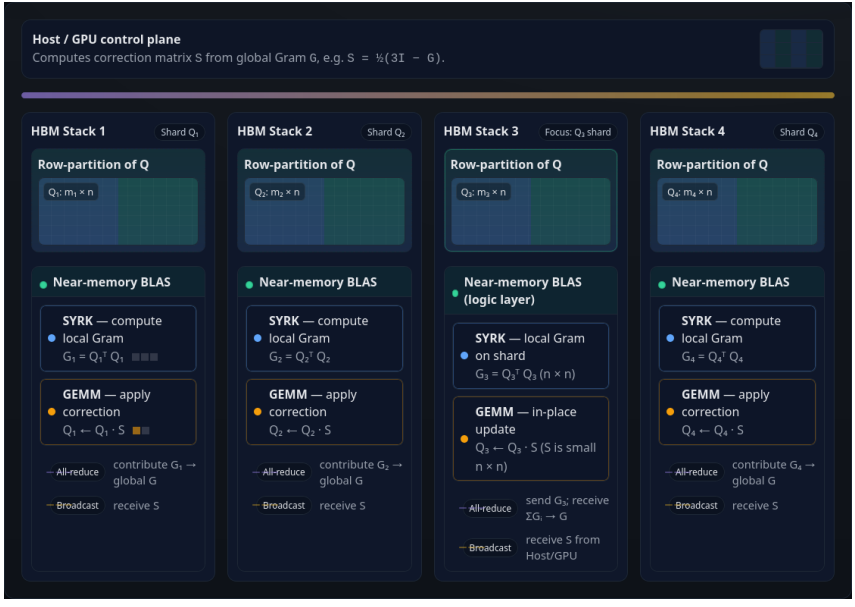
FP16 speed, FP64 trust

Newton–Schulz makes mixed-precision QR practical

Limitations & Future Work

- Extremely ill-conditioned may need extra guard.

Near-Memory BLAS: The Big Idea



Near-Memory BLAS: The Big Idea

- **Observation.** Modern HBM/3D-DRAM stacks include a *logic layer* that can host a few thousand low-power MAC units.
- **Near-Memory BLAS** = run Level-3 BLAS kernels *inside that logic layer*, so data stay on-stack.
- **Why it matters**
  1. Frequent patterns “stream matrix, reuse it once” become *compute-bound* instead of *bandwidth-bound*.
- **Programming model** • Same BLAS/LAPACK calls • Library decides “run near-memory” when operands already live in HBM

(Think of it as “cuBLAS, but the SMs sit *under* the DRAM dies.”)

Questions?

snima@duck.com