Numerical Linear Algebra Software in the AI Era
Present and Future

Maksim Melnichenko · Michael W. Mahoney · Riley Murray · Kimon Fountoulakis

RandBLAS & RandLAPACK

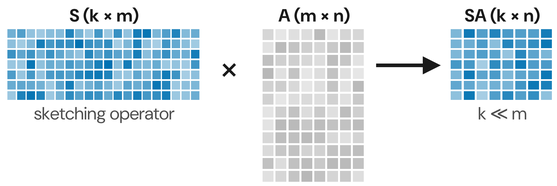
Building standard libraries for randomized NLA

RandNLA (*Randomized Numerical Linear Algebra*) [3] brings two decades of theoretical computer science to matrix computations, using randomness for speed and robustness. Our libraries turn it into the next generation of NLA software.

Responsible recklessness

Randomness as a tool buys speed and reliability, not uncertainty.

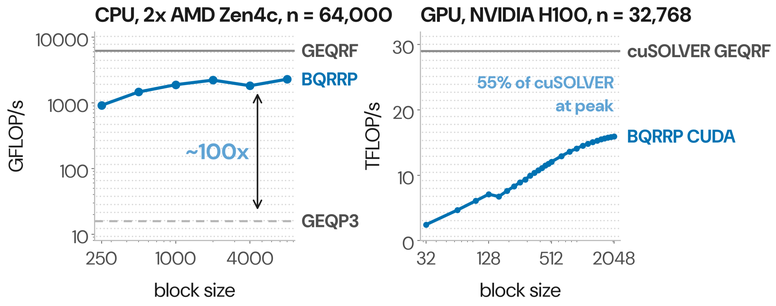
- **Sketching:** a random map S compresses A to a much smaller SA that keeps its geometry with high probability; the expensive step then runs on the small problem.
- **RandBLAS** implements these operators using a BLAS-style API, built to become a vendor-implemented standard like the BLAS; counter-based RNG (Random123 [6]) makes every draw reproducible.



S or A can be dense or sparse (COO, CSR, CSC)

Outperforming standard methods

RandLAPACK implements, tests and benchmarks RandNLA methods, novel and known, that outperform their counterparts in standard NLA libraries.



Task	RandLAPACK methods	AI tasks
Low-rank approximation	RSVD [18], RBKI [33]	LLM compression
Symmetric eigen, PSD	REVD2 [28], RPCholesky [19], Nyström EVD [3]	Kernel PCA, clustering
QR with pivoting	BQRRP [1], CQRRPT [2], HQRRP [20]	Structured pruning
Tall and Q-less QR	CQRRTO [7], sCholQR3 [21]	Distributed optimizers
Least squares, regression	KRILL [22], Blendenpik [23], LSQR [24]	Kernel ridge regression
Matrix functions, traces	funNyström++ [5], Lanczos [34]	GP likelihoods, Hessians

Delivering the libraries

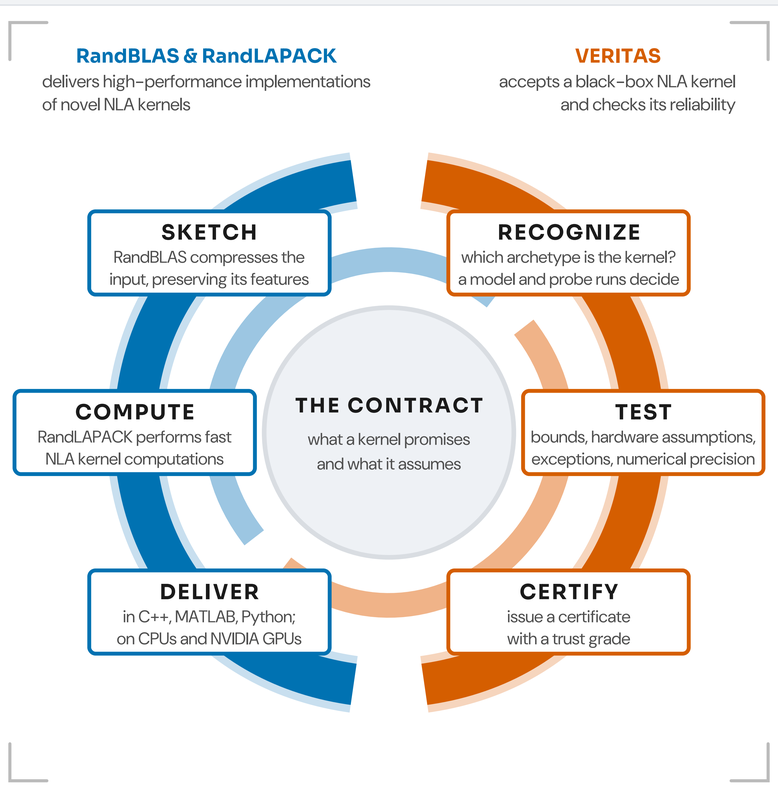
C++ at the core
Bindings for MATLAB and Python.

GPUs
RandLAPACK has CUDA support; RandBLAS CUDA support is in development.

Vendor BLAS and LAPACK
Through BLAS++ and LAPACK++ [27]: tested with OpenBLAS [29], Intel MKL [30], Apple Accelerate [31] and AMD AOCL [32].

Release
RandBLAS 1.1.0 is available now; RandLAPACK's 1.0 release candidate follows in November 2026.

For over fifty years, experts have built the standardized libraries of numerical linear algebra (NLA) that science runs on; we continue that tradition with ***next-generation NLA software***. Meanwhile, by 2026 ***generative AI*** has entered development pipelines, bringing ambitious new kernels to NLA, and ***new reliability risks***. Trust can no longer rest on authorship, so verification matters more than ever: we intend to build the ***means to trust kernels***.



100x
or more over LAPACK's pivoted QR (BQRRP on CPU) [1]

55%
of cuSOLVER's unpivoted QR speed, with pivoting (BQRRP) on an H100 [1]

5/5
top kernels of a 2026 GPU QR competition wrong on legal inputs

1,050
legal inputs tried per kernel, against the 19 the competition scored

10x
fewer matrix-vector products than prior trace estimators (funNyström++) [5]

1700x
faster than scikit-learn [25] kernel PCA at 100,000 points (RPCholesky [19])

27k
theorem lines on NLA in the VSCL Lean library

104
theorem lines on NLA in all of Mathlib [17] (182,000 total)

Project links

- One-command install of RandBLAS & RandLAPACK
- MATLAB and Python bindings
- The VSCL Lean library
- **Sources [1] to [34]**

VERITAS

Bringing trust to NLA kernels

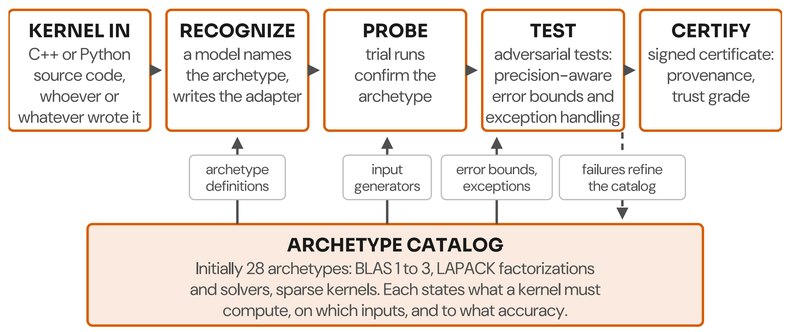
VERITAS (*Verification Engine for Reliable and Intelligent Testing of Algebra Systems*) is a **proposed** framework for testing and certifying numerical kernels, whoever or whatever wrote them.

The motivation

Round 2 of a 2026 GPU fast-QR competition [4]: top-5 kernels' speedup over cuSOLVER and wrong factorizations on 1,050 legal unscored inputs.

Leaderboard	Scored	Speedup	# Failed tests
rank 1	19/19	107x	157
rank 2	19/19	106x	46
rank 3	19/19	102x	330
rank 4	19/19	91x	3
rank 5	19/19	87x	360

VERITAS: the proposed design



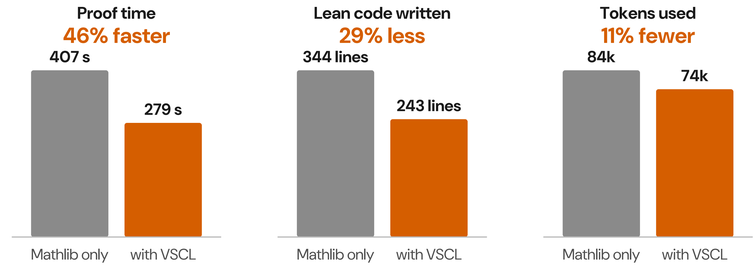
VSCL: the theorems behind VERITAS tests

The Verified Scientific Computing Library (VSCL) puts numerical analysis in Lean 4 [26]; its formalized results are the basis of VERITAS tests.

1.4M
lines of Lean code on top of Mathlib

43,858
theorems and lemmas in total; 0 sorry

- **60 proof tasks** from 54 numerical-analysis papers, 3 runs each: mean per task



- **Formalized:** Higham [8]. **In progress:** LeVeque [9], Vershynin [10], Drineas and Mahoney [11], Nocedal and Wright [12], Bhatia [13].
  Next: Greenbaum [14], Saad [15], Boyd [16].