Midwest Randomized Linear Algebra Workshop

Breakout 1: RandNLA Software Ecosystem

What Do We Build and How Do We Sustain It?

SESSION PARTICIPANTS

Joshua Cape · Chao Chen · Ilse Ipsen · Piotr Luszczek
Max Melnichenko · Raphael Meyer · Riley Murray · Nathaniel Pritchard

Facilitator: Jiaming Yang, University of Michigan

Where RandNLA Software Stands Today

- **Architectural mismatch —** LAPACK's data model is the BLAS data model: dense only, no native structure for sparse, sparse + low-rank, or kernel matrices. Modern RandNLA needs these.
- **Distribution is the operational pain point —** Apple silicon can't even use Intel MKL; hardware/system layer blocks math-fluent users. RandBLAS / RandLAPACK inherit this problem.
- **Awareness gap —** most users in adjacent communities do not know these tools exist.
- **Skill mismatch —** fluency in math does not imply fluency in C or hardware stacks.

Stable Core, Faster Satellites

Problem: Bringing RandNLA into LAPACK helps ***distribution***, but LAPACK's data model can't carry ***sparse / structured matrices***.

- **Stable upstream API as distribution surface —** what downstream stacks (SciPy, Julia, MATLAB) trust and build against. API stability is important.
- **Faster-moving satellites —** PLASMA / MAGMA precedent (mimic LAPACK API, 5× faster, GPU support, faster development) — where experimental sparse-aware RandNLA can live.
- **Raphael's bridge —** implement generically in RandLAPACK; LAPACK inherits the dense projection. A concrete design making the two-tier model work.

LAPACK Lineage as Anchor

INTEGRATION

- LAPACK is upstream of SciPy, Julia, MATLAB, landing RandNLA there reaches all of them through existing wrappers.
- Magnitude difference between BLAS and LAPACK; column- vs row-major switch is impractical at the source level.
- Feasibility of LLM-assisted line-by-line Fortran to C translation.

SUSTAINABILITY

- LAPACK / LINPACK as 10+ year success cases.
- Sustain the same maintaining group over decades.
- Ship parameters as expertise: ATLAS-style offline tuning encodes numerical-analysis expertise in shipped defaults.
- Students hard to graduate on software-only contributions: a structural disincentive.

Four Recommendations for NSF OAC

1. Tiered architecture (LAPACK + satellites): land dense RandNLA in LAPACK for distribution, build PLASMA/MAGMA-style satellite libraries for sparse/structured experimental work.

2. Distribution & packaging infrastructure: hardware-portable builds across ARM (Apple silicon), GPU, and Intel MKL. Solves the install bottleneck blocking adoption.

3. Ship parameters as expertise: ATLAS-style offline tuning encodes numerical-analysis expertise in shipped defaults, also lowers the barrier for non-NLA users.

4. Sustained teams + software career path: fund the same maintaining group over decades; support PhD that work on software works.

Open Disagreements

Worth surfacing to OAC rather than smoothing into consensus.

- LAPACK upstream vs. satellite-library model: where does the boundary fall between stable distribution surface and faster-moving experimental code?
- Fortran → C maintenance ownership — LLM-assisted translation is feasible; who owns the translated codebase is not.
- Require LAPACK contributions to publish in RandNLA: incentive or overreach?
- Decade-scale funding for the same group: can OAC operationalize a commitment of that length?