RandBLAS

An aspiring standard library, and why it matters.

Kaiwen He

BLIS Retreat
University of California, Berkeley

August 28, 2023

Acknowledgments

This work is part of *BALLISTIC*, which is led by

Jim Demmel, Jack Dongarra, Mark Gates, Julie Langou, Julien Langou, Piotr Luszczek, and Michael Mahoney.

This talk's material features additional contributions from

Riley Murray (Sandia), Maksim Melnichenko (UTK), James Demmel (UCB), Michael W. Mahoney (ICSI, UCB, LBNL), N. Benjamin Erichson (ICSI, LBNL), Osman Asif Malik (LBNL), Laura Grigori (INRIA), Piotr Luszczek (UTK), Michał Dereziński (UM), Miles E. Lopes (UCD), Tianyu Liang (UCB), Hengrui Luo (LBNL), Jack Dongarra (UTK), Burlen Loring (LBNL), Oleg Balabanov (INRIA, LBNL), Parth Nobel (Stanford)

If this work interests you, check out our book [1]!

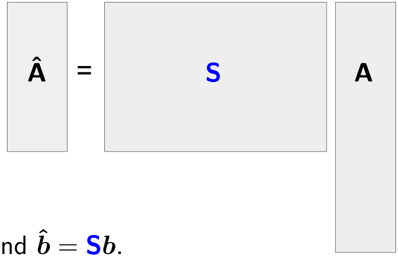
Randomized Numerical Linear Algebra: A Perspective on the Field With an Eye to Software (arXiv: 2302.11474)

Randomized numerical linear algebra (RandNLA)

Using randomized algorithms to solve deterministic problems.

Random sketching

E.g., for overdetermined least squares with data $(\mathbf{A}, \boldsymbol{b})$, obtain *sketched* data



and $\hat{\boldsymbol{b}} = \mathbf{S}\boldsymbol{b}$.

High-level deterministic NLA

Next, solve the sketched problem

$$\min_{\boldsymbol{x}} \|\mathbf{S}\left(\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\right)\|_2^2.$$

For example, by QR

$$\hat{\mathbf{A}} = \mathbf{QR},$$

$$\Rightarrow \quad \hat{\boldsymbol{x}} = \mathbf{R}^{-1}\mathbf{Q}^*\hat{\boldsymbol{b}}.$$

A Tale of Two Libraries

RandBLAS : an aspiring standard

- For sketching dense data matrices.
- Our reference implementation is in C++.
- Find code and docs here:

  `https://github.com/BallisticLA/RandBLAS`
  `https://randblas.readthedocs.io/en/latest/`.

RandLAPACK : a showcase for RandNLA

- Powered by RandBLAS.
- Has algorithms for ...
  - Least squares and optimization.
  - Low-rank approx.
  - **Full-rank decompositions**.
- Find code at `https://github.com/BallisticLA/RandLAPACK`.

Deciding the RandBLAS API: starting point

Premise: sketching with RandBLAS should look like GEMM.

Challenge: RandBLAS needs to work with linear operators.

Hypothetical GEMM with linear operators $(\mathbf{S}, \mathbf{A}, \mathbf{B})$:

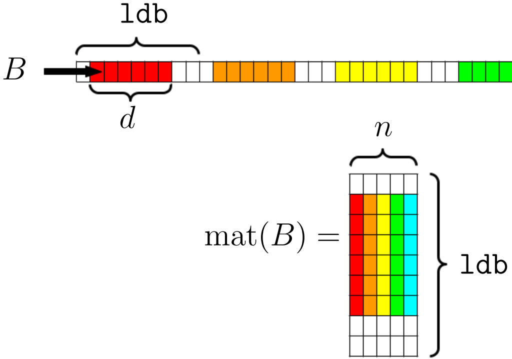
$$\mathbf{B} = \alpha \cdot \operatorname{op}(\mathbf{S}) \cdot \operatorname{op}(\mathbf{A}) + \beta \cdot \mathbf{B}.$$

Actual GEMM, with pointers, declared dimensions, and strides:

$$\operatorname{mat}(B) = \alpha \cdot \underbrace{\operatorname{op}(\operatorname{mat}(S))}_{d \times m} \cdot \underbrace{\operatorname{op}(\operatorname{mat}(A))}_{m \times n} + \beta \cdot \underbrace{\operatorname{mat}(B)}_{d \times n}$$

Submatrices and strides in GEMM

$\mathrm{mat}(B)$ can be any contiguous $d \times n$ submatrix of some larger matrix $\mathbf{B}_o$



More generally, we can specify $\mathrm{mat}(B)$ by row and column offsets in $\mathbf{B}_o$.

A GEMM-like interface for sketching

Left-sketching

$$\mathrm{mat}(B) = \alpha \cdot \underbrace{\mathrm{op}(\mathrm{submat}(\mathbf{S}))}_{d \times m} \cdot \underbrace{\mathrm{op}(\mathrm{mat}(A))}_{m \times n} + \beta \cdot \underbrace{\mathrm{mat}(B)}_{d \times n}$$

Right-sketching

$$\mathrm{mat}(B) = \alpha \cdot \underbrace{\mathrm{op}(\mathrm{mat}(A))}_{m \times n} \cdot \underbrace{\mathrm{op}(\mathrm{submat}(\mathbf{S}))}_{n \times d} + \beta \cdot \underbrace{\mathrm{mat}(B)}_{m \times d}$$

RandBLAS supports dense and sparse sketching operators.

Example RandBLAS function signature : left-sketching

We template **S** and dispatch type-specific implementations.

```
template <typename T, typename SKOP>
void sketch_general(
    blas::Layout layout,
    blas::Op opS,
    blas::Op opA,
    int64_t d, // B is d-by-n
    int64_t n, // \op(A) is m-by-n
    int64_t m, // \op(\submat(S)) is d-by-m
    T alpha,
    SKOP &S,
    int64_t i_off,
    int64_t j_off,
    const T *A,
    int64_t lda,
    T beta,
    T *B,
    int64_t ldb
);
```

Note: similar overloading is also possible in Fortran and C.

Two regimes for sketching

Sketching can look like *sampling* or like *embedding*.

$$\mathbf{A}^{\mathrm{sk}} = \mathbf{S}\,\mathbf{A}$$

Viable distributions:

- iid Gaussian
- iid uniform over ± 1
- structured sparse
- randomly subsampled FFT or DCT

Two regimes for sketching

Sketching can look like *sampling* or like *embedding*.

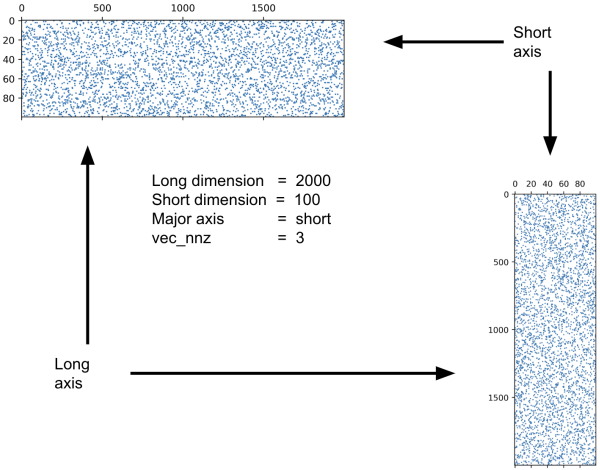
$$\mathbf{A}^{\mathrm{sk}} = \mathbf{S}\,\mathbf{A}$$

Viable distributions:

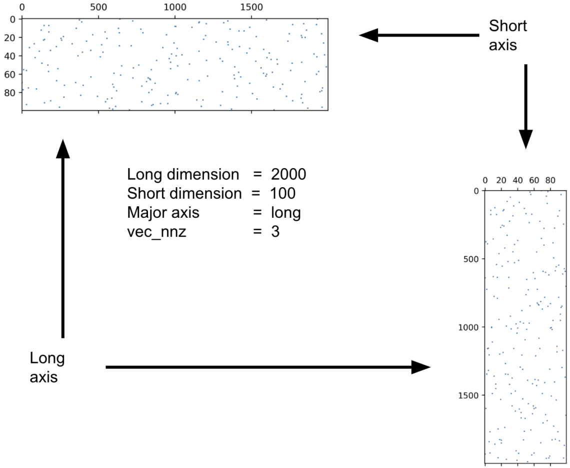
- structured sparse
- randomly subsampled block FFT or DCT

Distinguished by *relative sizes* of $(\mathbf{S}, \mathbf{A})$.

SASOs : short-axis-sparse operators



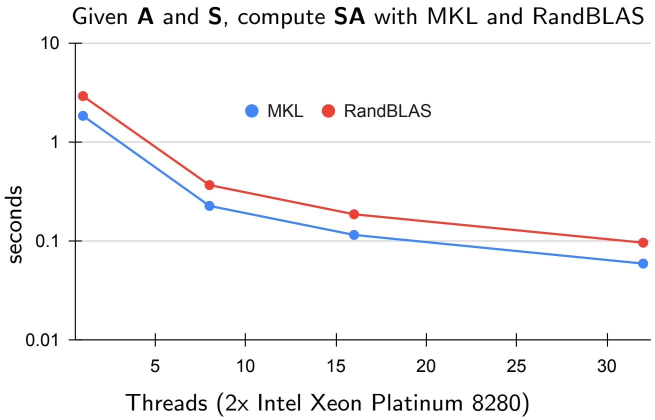
LASOs : long-axis-sparse operators



Efficiency of applying SASOs

Test data. **A** is $100{,}000 \times 2{,}000$, column-major storage.

S is $6{,}000 \times 100{,}000$ SASO with 8 nonzeros per column.



Usefulness of sparse sketching: existing algorithm

Task. Decompose $\mathbf{A} = \mathbf{QR}$ for tall column-orthogonal $\mathbf{Q}$ and upper-triangular $\mathbf{R}$.

Cholesky QR:

If $\mathbf{R} = \texttt{chol}(\mathbf{A}^*\mathbf{A}, \texttt{"upper"})$, then recover $\mathbf{Q} = \mathbf{A}\mathbf{R}^{-1}$ with TRSM.

The good: half the flops of Householder QR, and better cache efficiency!

The bad: in finite precision, we can only ensure

$$\|\mathbf{Q}^*\mathbf{Q} - \mathbf{I}\|_2 \lesssim \epsilon_{\mathrm{mach}} \cdot \mathrm{cond}(\mathbf{A})^2.$$

Preconditioned Cholesky QR [2]:

1. Sample $d \times m$ operator $\mathbf{S}$, with $d \gtrsim n$
2. sketch $\mathbf{A}_s = \mathbf{SA}$
3. $[\sim, \mathbf{R}_s] = \texttt{qr}(\mathbf{A}_s)$
4. $\mathbf{A}_p = \mathbf{A}\mathbf{R}_s^{-1}$ # preconditioning means condition number is $O(1)$
5. $[\mathbf{Q}, \mathbf{R}_p] = \texttt{chol\_qr}(\mathbf{A}_p)$ # stable!
6. $\mathbf{R} = \mathbf{R}_p\mathbf{R}_s$

A new algorithm in RandLAPACK

Task. Produce a permutation matrix $\mathbf{P}$ and a QR decomposition $\mathbf{AP} = \mathbf{QR}$.

Roughly: with good pivots, $\mathrm{diag}(\mathbf{R})$ approximates the spectrum of $\mathbf{A}$.

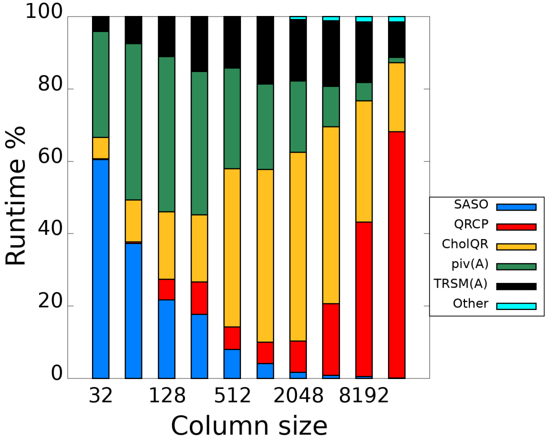
CQRRPT: Cholesky QR with randomized pivoting for tall matrices [1]

- Basic idea is simple adaptation of [2]:
  - run **pivoted** QR on **SA** and be careful about numerical rank.
- Easy to show in exact arithmetic:
  - If $\mathrm{rank}(\mathbf{A}_s) = \mathrm{rank}(\mathbf{A})$, then $\mathbf{AP} = \mathbf{QR}$.
  - $\kappa(\mathbf{A}_p) = \kappa(\mathbf{SU})$, where $\mathbf{U}$ is o.n.b. for $\mathrm{range}(\mathbf{A})$.
- Soon on arXiv: proof of numerical stability!

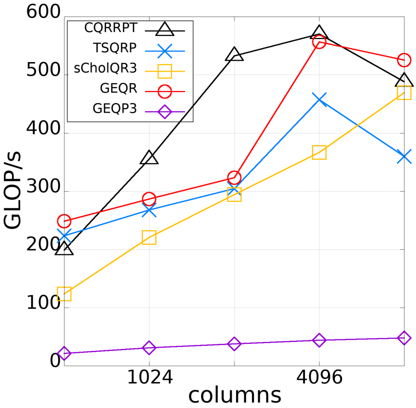
Next: benchmarks with Intel Xeon Gold 6140.

Runtime breakdown for CQRRPT

A has $m = 2^{17} \approx 130{,}000$ rows, **S** is a SASO



RandLAPACK's CQRRPT implementation



(all matrices have $m = 2^{17}$ rows)

CQRRPT is

- As fast as **unpivoted TSQR**.
- Faster than **tall-and-skinny QRCP.**
  GEQRF on **A**, GEQP3 on **R**.
- *Way* faster than **standard QRCP**.

(Note: **sCholQR3** is just another unpivoted QR method.)

Dense sketching operators

Things to address

- Reproducibility of RandBLAS (using CBRNG's)
- Ability to generate submatrices of Sketching Operators.
- Callback to earlier concept: Major axis translates to row-major vs column-major fill order

Take-aways

1. RandLAPACK demonstrates the practical importance of RandNLA algorithms.
2. Efficient sketching can make-or-break RandNLA algorithms.
3. RandBLAS offers high-performance sketching with a carefully designed API.

References I

[1] Riley Murray, James Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E. Lopes, Tianyu Liang, Hengrui Luo, and Jack Dongarra. Randomized numerical linear algebra : A perspective on the field with an eye to software, 2023.

[2] Yuwei Fan, Yixiao Guo, and Ting Lin. A novel randomized XR-based preconditioned CholeskyQR algorithm, 2021.

Example: pivot quality in CQRRPT

Test matrix $\mathbf{A} \in \mathbb{R}^{(2^{17}) \times 2{,}000}$

- First 200 singular values

$$s_1 = \cdots = s_{200} = 1$$

- Remaining singular values s_k decay polynomially to 10^{-10}.

Pivot quality for CQRRPT is almost identical to LAPACK's standard method (GEQP3).