Putting Randomness into LAPACK and Next Generation RandNLA Theory

Michael W. Mahoney[1] and Michał Dereziński[2]

[1]ICSI, LBNL, and Dept of Statistics, UC Berkeley
[2]Computer Science and Engineering, University of Michigan

Joint work with *Riley Murray* and many others!

March 2023

Outline

Standard libraries for numerical linear algebra

Basic Linear Algebra Subprograms
BLAS

Level 1. E.g.,

$$s = \boldsymbol{x}^\top \boldsymbol{y}$$

Level 2. E.g.,

$$\boldsymbol{y} = \alpha \boldsymbol{A}\boldsymbol{x} + \beta \boldsymbol{y}$$

Level 3. E.g.,

$$\boldsymbol{C} = \alpha \boldsymbol{A}\boldsymbol{B} + \beta \boldsymbol{C}$$

The Linear Algebra PACKage
LAPACK

Computational routines. E.g.,

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$$

$$\boldsymbol{A} = \boldsymbol{R}^\top \boldsymbol{R}$$

Drivers. E.g.,

$$\min \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$$

The situation

Communities that rely on NLA now vary widely.

They all want to solve larger and larger problems.

For decades, this hunger has been satiated by complementary innovations in hardware and software.

This progress should *not* be taken for granted.

Two factors increasingly present obstacles to scaling linear algebra to the next level.

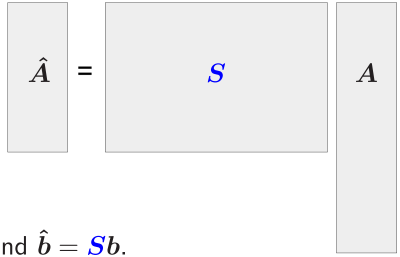
- Space and power constraints in hardware.
- NLA's maturity as a field.

Randomized numerical linear algebra (RandNLA)

Using randomized algorithms to solve deterministic problems.

Random sketching

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



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

High-level deterministic NLA

Next, solve the sketched problem

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

For example, by QR

$$\hat{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{R},$$

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

This is often called “sketch-and-solve”.

Randomized numerical linear algebra (RandNLA)

- Tutorials, light on prerequisites
  - "RandNLA: randomized numerical linear algebra," by Drineas and Mahoney [1]
  - "Lectures on randomized numerical linear algebra," by Drineas and Mahoney [2]
- Broad and proof-heavy resources
  - "Sketching as a tool for numerical linear algebra," by Woodruff [3]
  - "An introduction to matrix concentration inequalities," by Tropp [4]
  - "Lecture notes on randomized linear algebra," by Mahoney [5]
- Perspectives on theory, light on proofs
  - "Randomized algorithms for matrices and data," by Mahoney [6]
  - "Determinantal point processes in randomized numerical linear algebra," by Dereziński and Mahoney [7]
- Deep investigations of specific topics
  - "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," by Halko, Martinsson, and Tropp [8]
  - "Randomized algorithms in numerical linear algebra," by Kannan and Vempala [9]
  - "Randomized methods for matrix computations," by Martinsson [10]
  - "Randomized numerical linear algebra: Foundations and Algorithms," by Martinsson and Tropp [11]

What does randomization buy us?

- Efficient algorithms for computing approximate solutions

  Whole areas. E.g., low-rank approximation [8], convex optimization [12].

- Efficient algorithms for computing machine-precision solutions

  Specific problems. E.g., strongly overdetermined least squares [13, 14, 15], block column-pivoted QR [16, 17, 18].

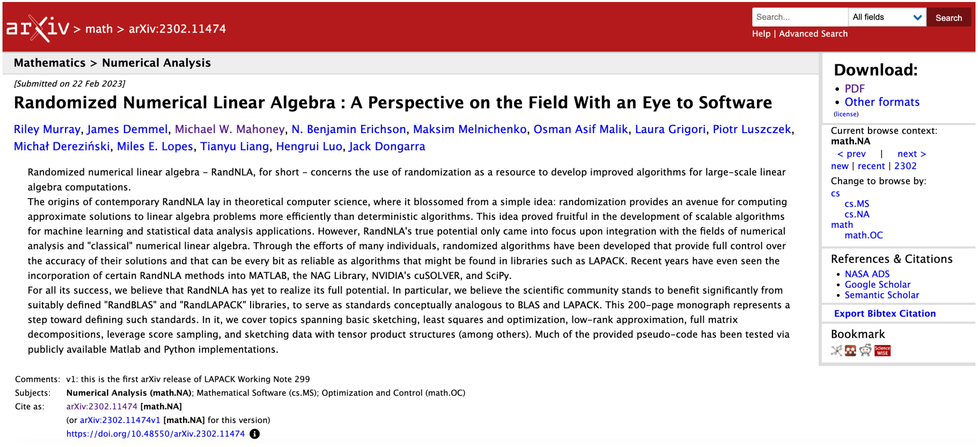
- Robust algorithms for intractable problems

  E.g., nonnegative matrix factorization [19], interpolative decomposition [20].

More generally

- Communication: lots of opportunities to reduce and redirect data movement.
- Finite-precision arithmetic: once a curse, now a blessing.

"The RandLAPACK book" [21]



Mathematics > Numerical Analysis

[Submitted on 22 Feb 2023]

Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software

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, Jack Dongarra

Randomized numerical linear algebra - RandNLA, for short - concerns the use of randomization as a resource to develop improved algorithms for large-scale linear algebra computations.
The origins of contemporary RandNLA lay in theoretical computer science, where it blossomed from a simple idea: randomization provides an avenue for computing approximate solutions to linear algebra problems more efficiently than deterministic algorithms. This idea proved fruitful in the development of scalable algorithms for machine learning and statistical data analysis applications. However, RandNLA's true potential only came into focus upon integration with the fields of numerical analysis and "classical" numerical linear algebra. Through the efforts of many individuals, randomized algorithms have been developed that provide full control over the accuracy of their solutions and that can be every bit as reliable as algorithms that might be found in libraries such as LAPACK. Recent years have even seen the incorporation of certain RandNLA methods into MATLAB, the NAG Library, NVIDIA's cuSOLVER, and SciPy.
For all its success, we believe that RandNLA has yet to realize its full potential. In particular, we believe the scientific community stands to benefit significantly from suitably defined "RandBLAS" and "RandLAPACK" libraries, to serve as standards conceptually analogous to BLAS and LAPACK. This 200-page monograph represents a step toward defining such standards. In it, we cover topics spanning basic sketching, least squares and optimization, low-rank approximation, full matrix decompositions, leverage score sampling, and sketching data with tensor product structures (among others). Much of the provided pseudo-code has been tested via publicly available Matlab and Python implementations.

Comments:	v1: this is the first arXiv release of LAPACK Working Note 299
Subjects:	**Numerical Analysis (math.NA)**; Mathematical Software (cs.MS); Optimization and Control (math.OC)
Cite as:	arXiv:2302.11474 **[math.NA]**
	(or arXiv:2302.11474v1 **[math.NA]** for this version)
	https://doi.org/10.48550/arXiv.2302.11474

"The RandLAPACK book" [21]

Table of Contents

"Randomized Numerical Linear Algebra: A Perspective on the Field with an Eye to Software," arXiv:2302.11474, R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M. E. Lopes, T. Liang, H. Luo, and J. Dongarra

Developing standard libraries for RandNLA

RandLAPACK

- Library that concerns algorithms for solving traditional linear algebra problems and advanced sketching functionality.
- To be written in C++, build on BLAS++/LAPACK++ portability layer [22].
- Main drivers:
  - Least squares and optimization.
  - Low-rank approximation
  - Full-rank decompositions.
- Prominent computational routines:
  - advanced sketching.
  - error estimation.
- The design spaces of algorithms for these tasks are large.

Developing standard libraries for RandNLA

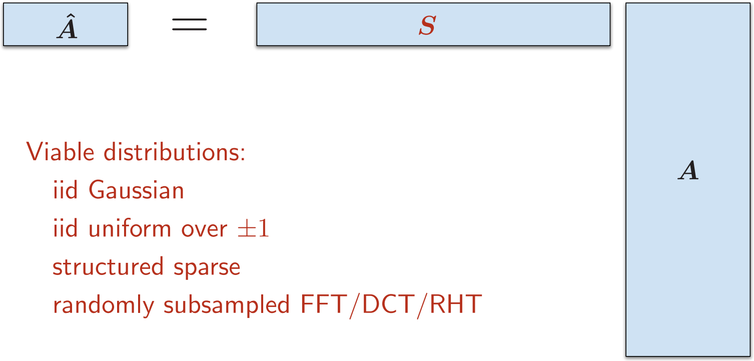
RandBLAS

- Library that concerns basic sketching.
- For sketching dense data matrices.
- Reference implementation in C++.
- Hope: it grows to become a community standard for RandNLA, in the sense that its API would see wider adoption than any single implementation.
- To achieve this goal, it is important to keep its scope narrowly focused.

Outline

Sketching in the RandBLAS

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



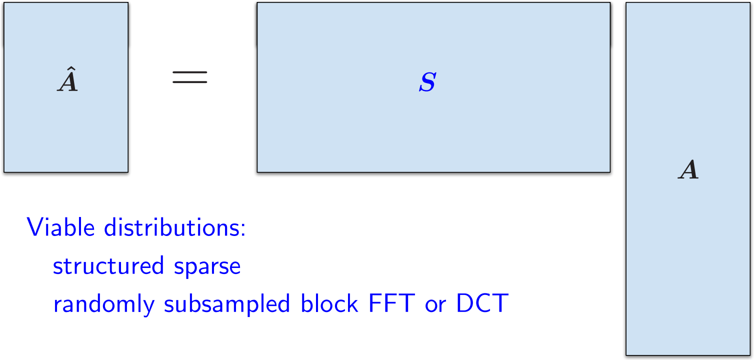
Viable distributions:

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

Distinguished by *relative sizes* of (S, A).

Sketching in the RandBLAS

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



Viable distributions:

- structured sparse
- randomly subsampled block FFT or DCT

Distinguished by *relative sizes* of (S, A).

SASOs: short-axis-sparse sketching operators

Consider $\boldsymbol{S} \in \mathbb{R}^{d \times m}$ that's very wide.

Independent columns; exactly ℓ ± 1's per column.

Examples from the literature: CountSketch ($\ell = 1$), SJLT, OSNAP [23, 24].

Sample ℓ indices from $\{1, \ldots, d\}$ without replacement, $m \gg d$ times.

Takes $O(d)$ workspace and $O(m\ell)$ time.

See GitHub: `https://tinyurl.com/sjlt-fy-recycle`

E.g., 10x speedup when $d = 6$K, $m = 100$K, and $\ell = 8$ on one laptop CPU core.

(Decrease from ≈ 1 second to 0.1 seconds.)

LASOs: long-axis-sparse sketching operators

Consider $\boldsymbol{S} \in \mathbb{R}^{d \times m}$ that's very wide.

Independent rows; at most ℓ non-zeros in each row:

- Sample $t_1, \ldots, t_\ell$ from $[m]$ according to a distribution $\mathbf{p}$ (e.g., uniform)
- Initialize a row of $\boldsymbol{S}$ with non-zeros in $\{t_1, \ldots, t_\ell\}$
- The non-zero entries are ± 1, scaled so that $\mathbb{E}[\boldsymbol{S}^* \boldsymbol{S}] = \boldsymbol{I}_m$.

Examples from the literature: LS sampling ($\ell = 1$), LESS, LessUniform [25, 26, 27].

LASO takes $O(d\ell)$ time

Faster than SASO with same ℓ, because $d \ll m$

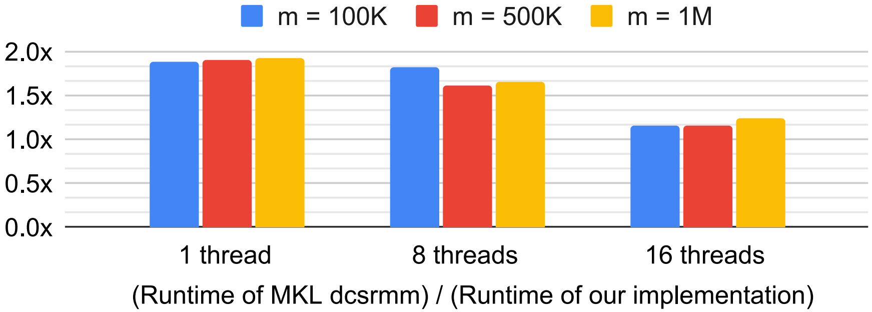
Quality depends on distribution $\boldsymbol{p}$ and the coherence of input matrix

Sparse matrix multiply faster than Intel MKL

This example fixes $(d, \ell) = (6\text{K}, 8)$ and varies m.

Compute $\boldsymbol{S}\boldsymbol{A}$ via **outer-product approach**:

- $\boldsymbol{S}$ stored in compressed sparse row (CSR)
- $\boldsymbol{A}$ has m rows, 2K columns, stored in row-major.



2.7 GHz Intel Xeon Platinum 8280, 192 GB DDR4.

Example: Preconditioned least squares

Least squares problem: $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ for $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

Preconditioner uses sketching operator $\boldsymbol{S} \in \mathbb{R}^{d \times m}$

```
1: $d = \min\{\lceil n \cdot \texttt{sampling\_factor} \rceil, m\}$
2: $\boldsymbol{S} = \texttt{SketchOpGen}(d, m)$
3: $\boldsymbol{Q}, \boldsymbol{R} = \texttt{qr\_econ}(\boldsymbol{S}\boldsymbol{A})$
4: $\boldsymbol{z}_o = \boldsymbol{Q}^\mathsf{T}\boldsymbol{S}\boldsymbol{b}$    # $\boldsymbol{R}^{-1}\boldsymbol{z}_o$ solves $\min_{\boldsymbol{x}}\{\|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2^2\}$
5: $\boldsymbol{A}_{\text{precond}} = \boldsymbol{A}\boldsymbol{R}^{-1}$ # as a linear operator
6: $\boldsymbol{z} = \texttt{iterative\_ls\_solver}(\boldsymbol{A}_{\text{precond}}, \boldsymbol{b}, \epsilon, L, \boldsymbol{z}_o)$
7: return $\boldsymbol{R}^{-1}\boldsymbol{z}$
```

Quality of the preconditioner (which affects the convergence of the solver) can, to an extent, be measured by $\text{cond}(\boldsymbol{A}\boldsymbol{R}^{-1}) = \text{cond}(\boldsymbol{S}\boldsymbol{U})$ for $\boldsymbol{U} = \texttt{orth}(\boldsymbol{A})$.

Corresponds to the distortion of $\boldsymbol{S}$ as a *subspace embedding* for $\boldsymbol{A}$.

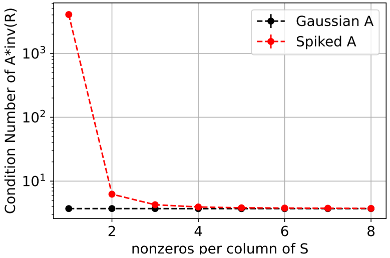
SASO quality: can we get a good preconditioner?

This example fixes $(d, m) = (6\text{K}, 100\text{K})$ and varies ℓ.

Consider two types of $100\text{K} \times 2\text{K}$ matrices $\boldsymbol{A}$:

- Gaussian: entries are iid standard normal.
- Spiked: stack identities and randomly scale 2K rows by 10K ("high coherence").

Let $\boldsymbol{U} = \texttt{orth}(\boldsymbol{A})$ and consider the condition number $\text{cond}(\boldsymbol{SU})$.



SASOs in action with PARLA

Least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

$\boldsymbol{A}$ is $100\text{K} \times 2\text{K}$

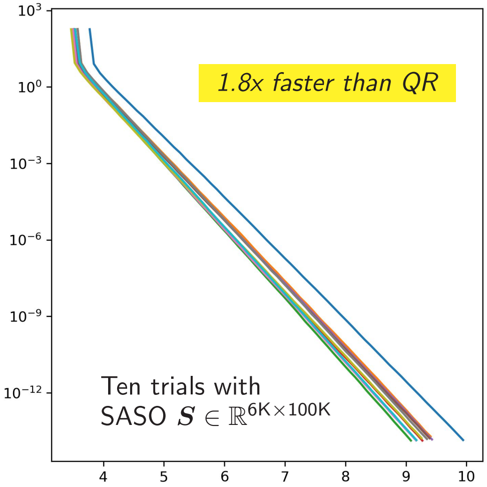
$$\mathrm{cond}(\boldsymbol{A}) = 100\text{K}$$

MATLAB times (seconds)

Using `qr(A,0)`:	17.3
Using `svd(A,"econ")`:	25.4

Core i7-1065G7

$\|(\boldsymbol{A}\boldsymbol{R}^{-1})^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2$ vs time in seconds



SASOs in action with PARLA

Least squares problem

$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$

$\boldsymbol{A}$ is $100\text{K} \times 2\text{K}$

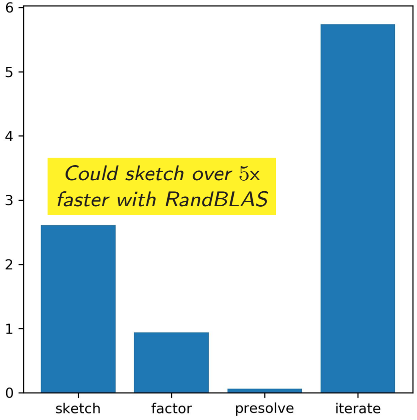
$\text{cond}(\boldsymbol{A}) = 100\text{K}$

MATLAB times (seconds)

Using `qr(A,0)`:	17.3
Using `svd(A,"econ")`:	25.4

Core i7-1065G7

Mean time in each algorithm phase



Ten trials with SASO $\boldsymbol{S} \in \mathbb{R}^{6\text{K} \times 100\text{K}}$

Performance landscape of randomized least squares

Least squares problem: $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ for $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

Preconditioner uses sketching operator $\boldsymbol{S} \in \mathbb{R}^{d \times m}$

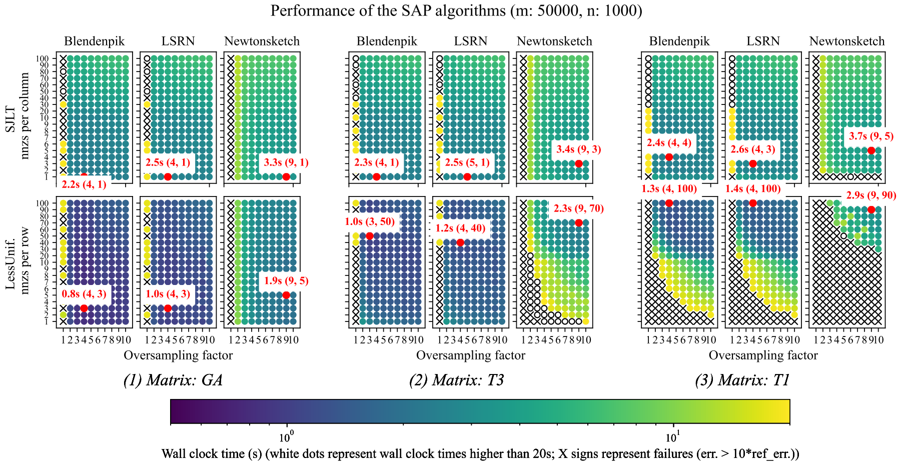
The key parameters that affect performance:

- Choice of solver
  - Type, e.g., Blendenpik [14], LSRN [15], or a more general-purpose optimization method like Newton Sketch [28] (included for comparison)
  - Stopping criterion and tolerance
- Choice of sketch
  - Type, e.g., SASO or LASO (or Gaussian or subsampled FFT, etc.)
  - Oversampling factor d/n
  - Sparsity parameter ℓ, i.e., nnzs per row/column of $\boldsymbol{S}$ (only for SASO or LASO)

Performance landscape of randomized least squares with PARLA

Input matrices: GA (low coherence), T3 (medium coherence), T1 (high coherence)

Sketching operators: SJLT (SASO), LessUniform (LASO with uniform distribution $\boldsymbol{p}$)



Performance landscape of randomized least squares with PARLA

Some take-aways from the performance landscape:

- Custom least squares solvers (Blendenpik, LSRN) are (predictably) better than a general purpose optimization method (Newton Sketch).
- LessUniform (LASO) with best sketch parameters is faster than SJLT (SASO) with best sketch parameters, regardless of solver and input matrix.
- SJLT (SASO) is more robust to the choice of sparsity and oversampling factor than LessUniform (LASO), for hard input matrices (high coherence).

The above takeaways should translate to the eventual optimized RandBLAS implementation.

Least squares and saddle point problems

Consider data $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$, and $\mu \geq 0$.

Primal problem

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \mu \|\boldsymbol{x}\|_2^2 + 2\boldsymbol{c}^{\mathsf{T}}\boldsymbol{x}.$$

Dual problem

$$\underset{\boldsymbol{y} \in \mathbb{R}^m}{\operatorname{argmin}} \|\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} - \boldsymbol{c}\|_2^2 + \mu \|\boldsymbol{y} - \boldsymbol{b}\|_2^2.$$

Application of sketching in the embedding regime:

1. $\sim, \boldsymbol{\Sigma}, \boldsymbol{V} = \mathtt{svd}(\boldsymbol{S}\boldsymbol{A})$
2. define preconditioner $\boldsymbol{M} = \boldsymbol{V}(\boldsymbol{\Sigma}^2 + \mu \boldsymbol{I})^{-1/2}$.

Least squares and saddle point problems

Primal-dual optimal solutions completely characterized by ...

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{A}^* & -\mu\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{c} \end{bmatrix}.$$

Using $\boldsymbol{y} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$, arrive at the *normal equations*

$$(\boldsymbol{A}^*\boldsymbol{A} + \mu\boldsymbol{I})\,\boldsymbol{x} = \boldsymbol{A}^*\boldsymbol{b} - \boldsymbol{c}.$$

Use reformulations. E.g., solve $\boldsymbol{c} = \boldsymbol{A}^*\boldsymbol{b}_{\text{shift}}$ for $\boldsymbol{b}_{\text{shift}}$, then set

$$\boldsymbol{A}_\mu = \begin{bmatrix} \boldsymbol{A} \\ \sqrt{\mu}\boldsymbol{I} \end{bmatrix} \quad \text{and} \quad \boldsymbol{b}_\mu = \begin{bmatrix} \boldsymbol{b} - \boldsymbol{b}_{\text{shift}} \\ \boldsymbol{0} \end{bmatrix},$$

so $\boldsymbol{x}$ solves normal equations iff ...

$$\boldsymbol{x} = \underset{\tilde{\boldsymbol{x}} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \|\boldsymbol{A}_\mu \tilde{\boldsymbol{x}} - \boldsymbol{b}_\mu\|_2^2 \right\}.$$

Reformulations have a major impact on the available iterative solvers!

Kernel ridge regression

Given $\lambda > 0$, pos def $m \times m$ "kernel matrix" $\boldsymbol{K}$, and observations $\boldsymbol{h} \in \mathbb{R}^m$, solve

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\operatorname{argmin}} \left\{ \frac{1}{m} \|\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{h}\|_2^2 + \lambda\, \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{K} \boldsymbol{\alpha} \right\}.$$

$\boldsymbol{K}$ is defined by a *kernel function* and data $\{\boldsymbol{x}_i\}_{i=1}^m \subset \mathcal{X}$. E.g.,

$$K_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2} \right).$$

Sketch-and-solve [29]:

- Note, optimal $\boldsymbol{\alpha}$ solves *KRR normal equations* $(\boldsymbol{K} + m\lambda)\boldsymbol{\alpha} = \boldsymbol{h}$.
- Approximate $\boldsymbol{K} \approx \boldsymbol{A}\boldsymbol{A}^{\mathsf{T}}$ by a RandNLA method.
- Solve $(\boldsymbol{A}\boldsymbol{A}^{\mathsf{T}} + m\lambda)\boldsymbol{\alpha} = \boldsymbol{h}$.

New: the sketched linear system is equivalent to a "dual" LS problem!

Low-rank approximation

Produce a suitably factored representation of a low-rank matrix $\hat{\boldsymbol{A}}$, which is an approximation of a target matrix $\boldsymbol{A}$.

Representations include ...

- SVD
- Hermitian eigenvalue decomposition
- CX and interpolative decompositions. E.g.,

  $\hat{\boldsymbol{A}} = \boldsymbol{C}\boldsymbol{X}$, for $\boldsymbol{C} = k$ columns of $\boldsymbol{A}$, suitable $\boldsymbol{X}$
- CUR decompositions

Algorithms in RandLAPACK

- can accept parameter k, produce $\hat{\boldsymbol{A}}$ where $\operatorname{rank}\hat{\boldsymbol{A}} = \min\{k, \operatorname{rank}\boldsymbol{A}\}$.
- some algs can accept ϵ and ensure $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\| \leq \epsilon$ (automatically determine k).

Example: Randomized SVD

A two-phase approach from Halko, Martinsson, and Tropp [8]:

```
1 $\boldsymbol{Y} = \boldsymbol{AS}$ # Sample from the range of $\boldsymbol{A}$
2 $\boldsymbol{Q} = \texttt{orth}(\boldsymbol{Y})$
3 $\boldsymbol{B} = \boldsymbol{Q}^\top \boldsymbol{A}$ # Implicitly, $\hat{\boldsymbol{A}} = \boldsymbol{QB} = \boldsymbol{QQ}^\top \boldsymbol{A}$.
4 $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^\top = \texttt{svd}(\boldsymbol{B})$
5 $\boldsymbol{U} = \boldsymbol{QU}$ # Implicitly, $\hat{\boldsymbol{A}} = \boldsymbol{U\Sigma V}^\top$.
6 return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^\top)$
```

Many variations! Two general strategies:

1. The sketching operator $\boldsymbol{S}$ can be "data-aware." Leverage *power iteration*.
2. Alternative constructions of $(\boldsymbol{Q}, \boldsymbol{B})$. Can proceed *iteratively*.

Full-rank decompositions: QR with column pivoting (QRCP)

Given $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, produce $(\boldsymbol{p}, \boldsymbol{Q}, \boldsymbol{R})$ where

$$\boldsymbol{A}[:, \boldsymbol{p}] = \boldsymbol{Q}\boldsymbol{R}.$$

- Would like to have $|\boldsymbol{R}[i,i]| \approx \sigma_i(\boldsymbol{A})$.
- Useful for ill-conditioned least-squares and low-rank approximation.

Standard methods: LAPACK's **QP3**, rank-revealing QR, window-pivoting.

All are much slower than unpivoted QR (LAPACK's **QRF**).

Householder QR with randomization for pivoting

Independently developed by Martinsson [16] and Duersch and Gu [17].

- Compute pivots in blocks of size b (e.g., $b = 64$) at a time.
- Base a block's pivots on a **sketch $\boldsymbol{Y} = \boldsymbol{S}\boldsymbol{A}$** (where $\boldsymbol{Y}$ has $k \gtrsim b$ rows).
- Update $(\boldsymbol{A}, \boldsymbol{Y})$ after each block.

We modified C code by Martinsson et al. [18] to use LAPACK++:

`https://github.com/rileyjmurray/hqrrp`.

Can easily link against Intel MKL, Apple Accelerate, AMD AOCL, etc...

We'll refer to the algorithm as "**QPR**."

Comparing QPR to QP3 and QRF

Core i7-1065G, 83.2 – 249.6 GFLOPS peak, 16GB DDR4 at 1866 MHz.

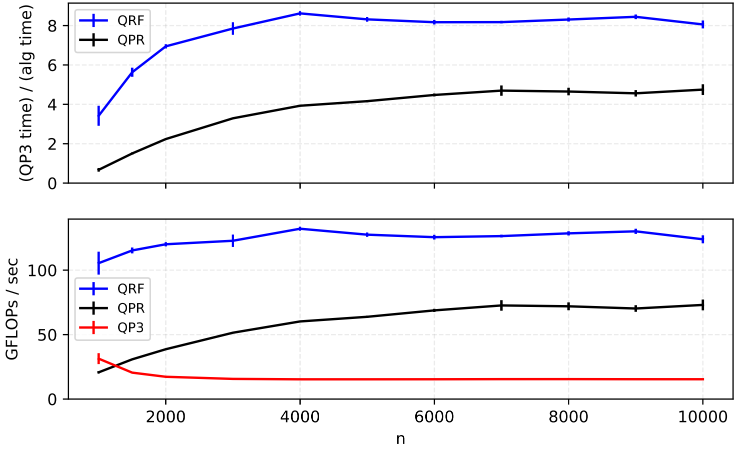


Figure: *"QPR" is* ***4x faster than MKL's QP3*** *once* $n \geq 5000$.

Comparing QPR to QP3 and QRF

Running on a 2020 Mac Mini with M1 CPU and 16GB RAM.

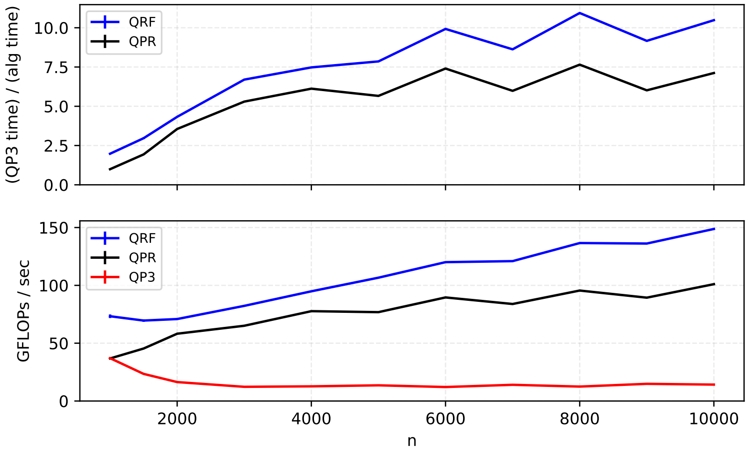


Figure: *"QPR" is 5x faster than Accelerate's QP3 once* $n \geq 3000$.

Outline

RandNLA theory: Aims motivated by RandLAPACK

1. *Sharp error estimates.*

   E.g., $\mathrm{Error} = (1 \pm o(1)) \cdot \mathrm{Estimate}$ instead of $\mathrm{Error} = \tilde{O}(\mathrm{Bound})$; all else being equal, "with high probability" is preferred over "in expectation".

2. *Practical parameter regimes.*

   E.g., sketch size $2n$ for least squares or $k + 5$ for low-rank, instead of $\tilde{O}(n)$ or $\tilde{O}(k)$.

3. *Non-asymptotic input dimensions.*

   E.g., for all $m \times n$ matrices $\boldsymbol{A}$, rather than asymptotically as m and n go to infinity.

4. *Fast sketching operators.*

   E.g., extremely sparse sketching matrices, with few nnzs per row/column, instead of dense Gaussian or ± 1 matrices. Although, what is fast depends a lot on the problem.

RandNLA theory: Aims motivated by RandLAPACK

1. *Sharp error estimates.*
2. *Practical parameter regimes.*
3. *Non-asymptotic input dimensions.*
4. *Fast sketching operators.*

- TCS analysis usually gets Aims 3 and 4, and works with fast sketches like Subsampled FFT/RHT, SASOs (CountSketch, SJLT), but fast is in big-O sense.
- Specialized analysis for Gaussian sketches can often get Aims 1 - 3.
- General RMT analysis works with dense ± 1 sketches and gets Aims 1 and 2.
- Some recent works using free probability techniques can extend the RMT analysis to Subsampled FFT/RHT, which (at least in theory) qualify for Aim 4.
- Most recently, we have been able to get Aims 1 - 4 with certain fast LASOs (LESS embeddings).

RMT analysis in RandNLA: Toy example

Consider input matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and iid Gaussian sketching matrix $\boldsymbol{S} \in \mathbb{R}^{d\times m}$

Quality of the sketch $\hat{\boldsymbol{A}} = \boldsymbol{S}\boldsymbol{A}$ is often measured by cond$(\boldsymbol{S}\boldsymbol{U})$ for $\boldsymbol{U} = \texttt{orth}(\boldsymbol{A})$ (e.g., subspace embedding, quality of a preconditioner, etc.)

Thanks to the rotation invariance of Gaussian distribution, $\boldsymbol{S}\boldsymbol{U}$ is also Gaussian, so we can use the Marchenko-Pastur law:

$$\sigma_{\min}(\boldsymbol{S}\boldsymbol{U}) \sim 1 - \sqrt{\frac{n}{d}}, \qquad \sigma_{\max}(\boldsymbol{S}\boldsymbol{U}) \sim 1 + \sqrt{\frac{n}{d}}$$

Sharp non-asymptotic high-probability statements can be obtained as well. [30]

This Random Matrix Theory (RMT) approach recovers bounds on cond$(\boldsymbol{S}\boldsymbol{U})$ for almost any sketch size $d > n$, whereas the Johnson-Lindenstrauss (JL) approach is vacuous for $d < Cn$ for some constant $C = O(1)$.

RMT analysis in RandNLA: Least squares

Consider sketching matrix $\boldsymbol{S} \in \mathbb{R}^{d\times m}$ with iid Gaussian entries.

Least squares: $\boldsymbol{x}^* = \mathrm{argmin}_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$

- Sketch-and-precondition: Construct preconditioner $\boldsymbol{R}^{-1}$ from the QR of $\boldsymbol{S}\boldsymbol{A}$

$$\mathrm{cond}(\boldsymbol{A}\boldsymbol{R}^{-1}) \le 6 \quad \text{with high probability for} \quad d \ge 2n.$$

- Sketch-and-solve: Compute $\hat{\boldsymbol{x}} = \mathrm{argmin}_{\boldsymbol{x}} \|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2^2$ directly

$$\mathbb{E}\|\boldsymbol{A}(\hat{\boldsymbol{x}} - \boldsymbol{x}^*)\|_2^2 = \frac{n}{d-n-1}\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2^2 \quad \text{for} \quad d \ge n+2.$$

Those statements are not recovered by JL-style analysis.

Many related results [31, 32, 33, 34], including ridge regression [35, 36, 37, 38].

RMT analysis in RandNLA: Low-rank approximation

Randomized SVD: Compute $\boldsymbol{Q} = \texttt{orth}(\boldsymbol{A}\boldsymbol{S})$, and implicitly, $\hat{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$.

Once again, there is a simple bound for Gaussian $\boldsymbol{S}$, relative to best rank-k approximation $\boldsymbol{A}_k$ [39]:

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_F^2 \leq \left(1 + \frac{k}{d-k-1}\right) \cdot \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 \quad \text{for} \quad d \geq k+2.$$

Moreover, using RMT, we can show that for Gaussian and ± 1 matrices $\boldsymbol{S}$:

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_F^2 = (1 \pm o(1)) \cdot \alpha \quad \text{for} \quad \alpha \quad \text{such that} \quad \sum_i \frac{\sigma_i^2(\boldsymbol{A})}{d\sigma_i^2(\boldsymbol{A}) + \alpha} = 1.$$

This is sharper when $\boldsymbol{A}$ exhibits realistic spectral decays, e.g., allowing for small approximation factor even with sketch size $d = k$.

Many related results available, e.g., [40, 41, 42].

RMT analysis in RandNLA: Iterative sketching

Consider input matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and iid Gaussian or ± 1 matrices $\boldsymbol{S}_t \in \mathbb{R}^{d\times m}$.
Note: In iterative sketching, input matrix may change in each iteration.

- Iterative Hessian Sketch: $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta\big(\boldsymbol{A}^*\boldsymbol{S}_t^*\boldsymbol{S}_t\boldsymbol{A}\big)^\dagger \boldsymbol{g}_t$.

$$\mathbb{E}\,\frac{\|\boldsymbol{x}_{t+1}-\boldsymbol{x}^*\|^2}{\|\boldsymbol{x}_t-\boldsymbol{x}^*\|^2} = (1\pm o(1))\cdot\bigg((1-\eta)^2 + \frac{n}{d-n}\eta^2\bigg).$$

  This lets us derive the optimal step size η. [43, 44, 45]

- Sketch-and-Project (Generalized Kaczmarz): $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - (\boldsymbol{S}_t\boldsymbol{A})^\dagger\boldsymbol{S}_t(\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{b})$

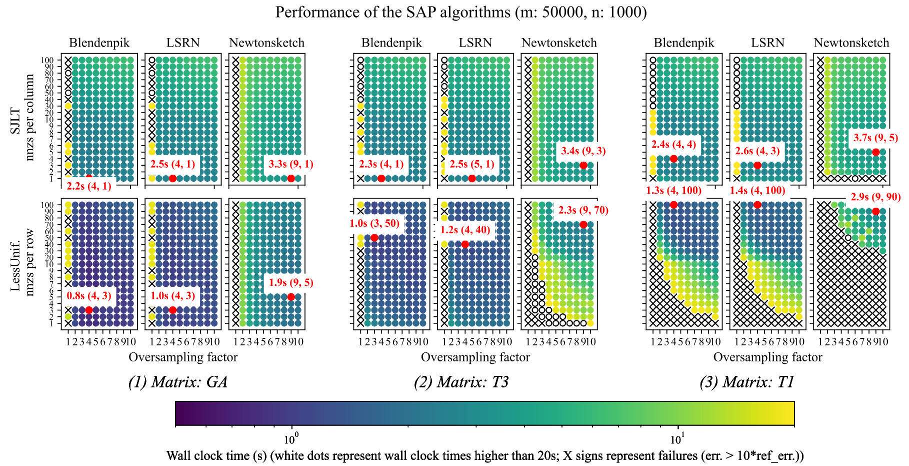
$$\mathbb{E}\,\frac{\|\boldsymbol{x}_{t+1}-\boldsymbol{x}^*\|^2}{\|\boldsymbol{x}_t-\boldsymbol{x}^*\|^2} \le 1-(1-o(1))\cdot\frac{d\sigma^2_{\min}(\boldsymbol{A})}{\mathbb{E}\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|_F^2},$$

  This relates the convergence rate of Generalized Kaczmarz to the approximation error $\mathbb{E}\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|_F^2$ of Randomized SVD. [46, 47, 48]

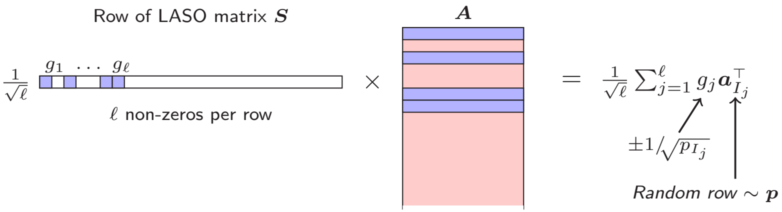
Can we extend this to sparse sketching operators?

Recall: Performance landscape of randomized least squares with sparse sketches

Sketching operators: SJLT (SASO), LessUniform (LASO with uniform distribution $\boldsymbol{p}$)



Central Limit Theorem for sparse sketching operators


Row of LASO matrix $\boldsymbol{S}$
$\frac{1}{\sqrt{\ell}}$
$g_1 \; \dots \; g_\ell$
ℓ non-zeros per row
$\times$
$\boldsymbol{A}$
$= \frac{1}{\sqrt{\ell}} \sum_{j=1}^{\ell} g_j \boldsymbol{a}_{I_j}^\top$
$\pm 1/\sqrt{p_{I_j}}$
Random row $\sim \boldsymbol{p}$

Central Limit Theorem leads to implicit "Algorithmic Gaussianization" of the sketch:

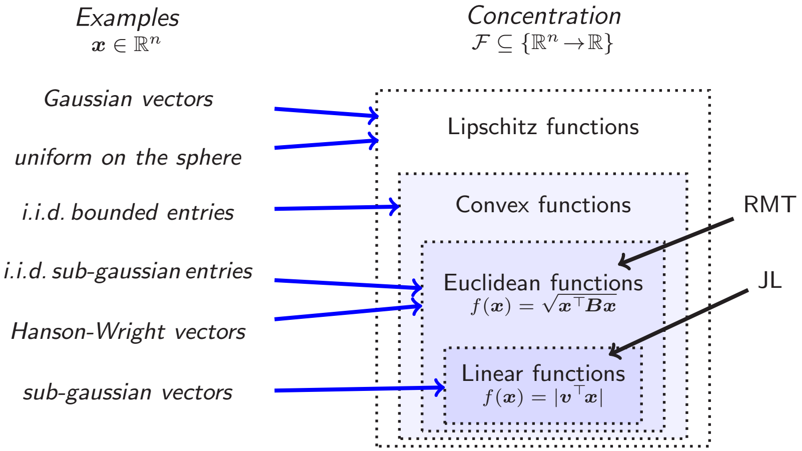
$$\frac{1}{\sqrt{\ell}} \sum_{j=1}^{\ell} g_j \boldsymbol{a}_{I_j}^\top \quad \xrightarrow{\ell \to \infty} \quad \mathcal{N}(\boldsymbol{0}, \boldsymbol{A}^\top \boldsymbol{A})$$

- How many samples/non-zeros do we need?
- When is the sketch sufficiently "Gaussianized"?
- How do we quantify the convergence?
  e.g. Wasserstein distance, total variation (TV) distance, etc.

The hierarchy of Gaussianized vectors

Sub-gaussian concentration of $\boldsymbol{x} \in \mathbb{R}^n$ w.r.t. a set of functions $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$

$$\forall f \in \mathcal{F} : \qquad X = f(\boldsymbol{x}) - \mathbb{E}\, f(\boldsymbol{x}) \quad \text{is} \quad O(\|f\|_{\mathrm{Lip}})\text{-sub-gaussian}$$


Examples
$\boldsymbol{x} \in \mathbb{R}^n$
Concentration
$\mathcal{F} \subseteq \{\mathbb{R}^n \to \mathbb{R}\}$
Gaussian vectors
uniform on the sphere
i.i.d. bounded entries
i.i.d. sub-gaussian entries
Hanson-Wright vectors
sub-gaussian vectors
Lipschitz functions
Convex functions
Euclidean functions
$f(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{x}}$
Linear functions
$f(\boldsymbol{x}) = |\boldsymbol{v}^\top \boldsymbol{x}|$
RMT
JL

CLT characterization for LASO sketches

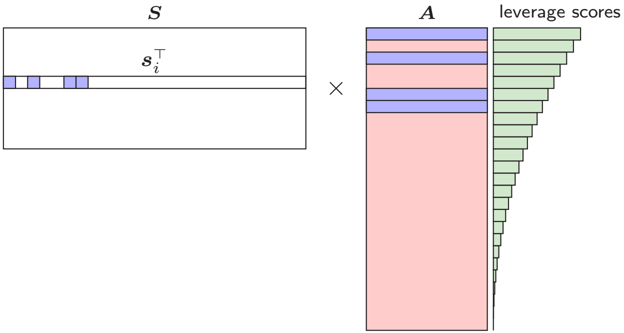
Consider tall input matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, i.e., $m \gg n$, and a wide LASO matrix $\boldsymbol{S} \in \mathbb{R}^{d\times m}$ with distribution $\boldsymbol{p}$ and $\ell \in [m]$ non-zeros per row.

Informal statement. [49]
If $\boldsymbol{p}$ is a τ-approximation of the leverage score distribution of $\boldsymbol{A}$, and we use $\ell \geq \tau n \log(nd/\delta)$ non-zeros, then $\boldsymbol{SA}$ is total variation distance δ away from a sketch $\tilde{\boldsymbol{S}}\boldsymbol{A}$ that satisfies Euclidean function concentration.

Leverage score of the ith row $\boldsymbol{a}_i$ of $\boldsymbol{A}$ is: $\ell_i(\boldsymbol{A}) = \boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{a}_i$.

- When $\boldsymbol{p}$ is the leverage score distribution, we call this LEverage Score Sparsification (LESS).
- When $\boldsymbol{p}$ is the uniform distribution (LessUniform), then τ is simply the *coherence* of matrix $\boldsymbol{A}$.
- We show this by establishing a version of the Hanson-Wright inequality that is restricted to the subspace defined by the columns of $\boldsymbol{A}$.

Leverage Score Sparsified (LESS) embeddings [34]



leverage score of ith row $\boldsymbol{a}_i$: $\ell_i(\boldsymbol{A}) = \boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{a}_i$.

Leverage Score Sparsified (LESS) embeddings [34]

- Sparse sketching operators which are essentially indistinguishable from Gaussian.

- The central limit theorem characterization provides:
  - a way to convert RMT-style results from Gaussian sketches to sparse sketching operators, including for least squares, low-rank and iterative sketching.
  - an explanation for the empirical behavior of LASO sketches, in terms of the coherence of the input matrix $\boldsymbol{A}$.

- Example of a non-asymptotic RMT-style result for LESS embeddings: [49]

$$\mathbb{E}\|\boldsymbol{A}(\hat{\boldsymbol{x}} - \boldsymbol{x}^*)\|_2^2 = \big(1 \pm O(\tfrac{1}{\sqrt{d}})\big) \cdot \frac{n}{d-n-1}\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2^2$$

RandNLA directions looking beyond RandLAPACK

What will we need from RandNLA in 10-20 years,
when RandLAPACK is ubiquitous and ChatGPT runs the world?

- RandNLA in large-scale continuous optimization.
- RandNLA in massively distributed computing environments.
- Statistical/ML properties of RandNLA algorithms.

Outline

Conclusions

Things we covered:

- The nature of RandBLAS and RandLAPACK.
- Efficient sparse sketching.
- The importance of sparse sketching in a least squares context.
- Randomized algorithms for low-rank and full-rank decompositions.
- Theoretical directions in RandNLA motivated by RandLAPACK.

Extra slides

A word on “drivers” and “computational routines”

Most algorithms are either drivers or computational routines (terms borrowed from LAPACK’s API).
Drivers:

- solve higher-level problems than computational routines,
- their implementations tend to use a small number of computational routines,
- are used only for traditional linear algebra problems.

Computational routines:

- address a mix of traditional linear algebra problems and specialized RandNLA problems.

We use this taxonomy to push much of the RandNLA design space into computational routines.

- essential: to keeping drivers simple and few in number.
- side effect: since choices made in the computational routines affect drivers, it is hard to state theoretical guarantees for the drivers without being prescriptive on the choice of computational routine (which we don’t want to do).

Least Squares and Optimization (Section 3)

3.1 Problem classes

- 3.1.1 Minimizing regularized quadratics
- 3.1.2 Solving least squares and basic saddle point problems

3.2 Drivers

- 3.2.1 Sketch-and-solve for overdetermined least squares
- 3.2.2 Sketch-and-precondition for least squares and saddle point problems
- 3.2.3 Nystrom PCG for minimizing regularized quadratics
- 3.2.4 Sketch-and-solve for minimizing regularized quadratics

3.3 Computational routines

- 3.3.1 Technical background: optimality conditions for saddle point problems
- 3.3.2 Preconditioning least squares and saddle point problems: tall data matrices
- 3.3.3 Preconditioning least squares and saddle point problems: data matrices with fast spectral decay
- 3.3.4 Deterministic preconditioned iterative solvers

Low-rank Approximation (Section 4)

4.1 Problem classes

- 4.1.1 Spectral decompositions
- 4.1.2 Submatrix-oriented decompositions

4.2 Drivers

- 4.2.1 Methods for SVD
- 4.2.2 Methods for Hermitian eigendecomposition
- 4.2.3 Methods for CUR and two-sided ID

4.3 Computational routines

- 4.3.1 Power iteration
- 4.3.2 Orthogonal projections: QB and rangefinders
- 4.3.3 Column-pivoted matrix decompositions
- 4.3.4 One-sided ID and CSS
- 4.3.5 Estimating matrix norms
- 4.3.6 Oblique projections

Further Possibilities for Drivers (Section 5)

5.1 Multi-purpose matrix decompositions

- 5.1.1 QR decomposition of tall matrices
- 5.1.2 QR decomposition with column pivoting
- 5.1.3 UTV, URV, and QLP decompositions

5.2 Solving unstructured linear systems

- 5.2.1 Direct methods
- 5.2.2 Iterative methods

5.3 Trace estimation

- 5.3.1 Sampling-based methods
- 5.3.2 Quadrature-based methods via Krylov subspaces
- 5.3.3 There's much more to say

References I

[1] P. Drineas and M. W. Mahoney.
RandNLA: Randomized numerical linear algebra.
Communications of the ACM, 59:80–90, 2016.

[2] P. Drineas and M. W. Mahoney.
Lectures on randomized numerical linear algebra.
In M. W. Mahoney, J. C. Duchi, and A. C. Gilbert, editors, *The Mathematics of Data*, IAS/Park City Mathematics Series, pages 1–48. AMS/IAS/SIAM, 2018.
Available at https://arxiv.org/abs/1712.08880.

[3] David P. Woodruff.
Sketching as a tool for numerical linear algebra.
Found. Trends Theor. Comput. Sci., 10(1–2):1–157, October 2014.

[4] Joel A. Tropp.
An introduction to matrix concentration inequalities.
Foundations and Trends® in Machine Learning, 8(1-2):1–230, 2015.

References II

[5] M. W. Mahoney.
Lecture notes on randomized linear algebra.
Technical Report Preprint: arXiv:1608.04481, 2016.

[6] M. W. Mahoney.
Randomized algorithms for matrices and data.
Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011.

[7] M. Derezinski and M. W. Mahoney.
Determinantal point processes in randomized numerical linear algebra.
Notices of the AMS, 68(1):34–45, 2021.

[8] N. Halko, P. G. Martinsson, and J. A. Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Review, 53(2):217–288, January 2011.

References III

[9] Ravindran Kannan and Santosh Vempala.
Randomized algorithms in numerical linear algebra.
Acta Numerica, 26:95–135, 2017.

[10] Per-Gunnar Martinsson.
Randomized methods for matrix computations.
The Mathematics of Data, 25(4):187–239, 2018.
Note: preprint arXiv:1607.01649 published in 2016, updated in 2019.

[11] Per-Gunnar Martinsson and Joel A. Tropp.
Randomized numerical linear algebra: Foundations and Algorithms.
Acta Numerica, 29:403–572, 2020.

[12] Mert Pilanci and Martin J Wainwright.
Iterative Hessian Sketch: Fast and accurate solution approximation for constrained least-squares.
J. Mach. Learn. Res., 17(1):1842–1879, January 2016.

References IV

[13] V Rokhlin and M Tygert.
A fast randomized algorithm for overdetermined linear least-squares regression.
Proceedings of the National Academy of Sciences, 105(36):13212–13217, September 2008.

[14] Haim Avron, Petar Maymounkov, and Sivan Toledo.
Blendenpik: Supercharging LAPACK's Least-Squares Solver.
SIAM Journal on Scientific Computing, 32(3):1217–1236, January 2010.

[15] Xiangrui Meng, Michael A. Saunders, and Michael W. Mahoney.
LSRN: A parallel iterative solver for strongly over- or underdetermined systems.
SIAM Journal on Scientific Computing, 36(2):C95–C118, January 2014.
Software at `https://web.stanford.edu/group/SOL/software/lsrn/`.

References V

[16] P. G. Martinsson.
Blocked rank-revealing QR factorizations: How randomized sampling can be used to avoid single-vector pivoting.
arXiv preprint arXiv:1505.08115, 2015.

[17] Jed A. Duersch and Ming Gu.
Randomized QR with column pivoting.
SIAM Journal on Scientific Computing, 39(4):C263–C291, January 2017.

[18] Per-Gunnar Martinsson, Gregorio Quintana Ortí, Nathan Heavner, and Robert van de Geijn.
Householder QR factorization with randomization for column pivoting (HQRRP).
SIAM Journal on Scientific Computing, 39(2):C96–C115, January 2017.

[19] N. Benjamin Erichson, Ariana Mendible, Sophie Wihlborn, and Nathan J. Kutz.
Randomized nonnegative matrix factorization.
Pattern Recognition Letters, 104:1–7, 2018.

References VI

[20] Sergey Voronin and Per-Gunnar Martinsson.
Efficient algorithms for CUR and interpolative matrix decompositions.
Advances in Computational Mathematics, 43(3):495–516, November 2016.

[21] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M. E. Lopes, T. Liang, H. Luo, and J. Dongarra.
Randomized numerical linear algebra: A perspective on the field with an eye to software.
arXiv preprint arXiv:2302.11474, 2023.

[22] Mark Gates, Piotr Luszczek, Ahmad Abdelfattah, Jakub Kurzak, Jack Dongarra, Konstantin Arturov, Cris Cecka, and Chip Freitag.
C++ API for BLAS and LAPACK.
Technical Report 02, ICL-UT-17-03, June 2017.
Revision 02-21-2018.

References VII

[23] Jelani Nelson and Huy L. Nguyen.
OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings.
In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, October 2013.

[24] Kenneth L. Clarkson and David P. Woodruff.
Low-rank approximation and regression in input sparsity time.
J. ACM, 63(6), January 2017.
This is the journal version of a 2013 STOC article by the same name.

[25] P. Drineas, M. W. Mahoney, and S. Muthukrishnan.
Sampling algorithms for ℓ_2 regression and applications.
In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1127–1136, 2006.

References VIII

[26] Michał Dereziński, Zhenyu Liao, Edgar Dobriban, and Michael Mahoney.
Sparse sketches with small inversion bias.
In *Conference on Learning Theory (COLT)*, pages 1467–1510. PMLR, 2021.

[27] Michał Dereziński, Jonathan Lacotte, Mert Pilanci, and Michael W Mahoney.
Newton-LESS: Sparsification without trade-offs for the sketched Newton update.
Advances in Neural Information Processing Systems, 34, 2021.

[28] Mert Pilanci and Martin J Wainwright.
Newton Sketch: A near linear-time optimization algorithm with linear-quadratic convergence.
SIAM Journal on Optimization, 27(1):205–245, January 2017.

[29] A. El Alaoui and M. W. Mahoney.
Fast randomized kernel methods with statistical guarantees.
In *Annual Advances in Neural Information Processing Systems*, pages 775–783, 2015.

References IX

[30] Mark Rudelson and Roman Vershynin.
Non-asymptotic theory of random matrices: extreme singular values.
In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, pages 1576–1602. World Scientific, 2010.

[31] E. Dobriban and S. Liu.
Asymptotics for sketching in least squares regression.
Technical Report Preprint: arXiv:1810.06089, 2018.

[32] Miles Lopes, Shusen Wang, and Michael Mahoney.
Error estimation for randomized least-squares algorithms via the bootstrap.
In *International Conference on Machine Learning*, pages 3217–3226. PMLR, 2018.

[33] Miles E Lopes, Shusen Wang, and Michael W Mahoney.
A bootstrap method for error estimation in randomized matrix multiplication.
The Journal of Machine Learning Research, 20(1):1434–1473, 2019.

References X

[34] Michał Dereziński, Zhenyu Liao, Edgar Dobriban, and Michael Mahoney.
Sparse sketches with small inversion bias.
In *Conference on Learning Theory*, pages 1467–1510. PMLR, 2021.

[35] Michał Dereziński, Burak Bartan, Mert Pilanci, and Michael W Mahoney.
Debiasing distributed second order optimization with surrogate sketching and scaled regularization.
Advances in Neural Information Processing Systems, 33:6684–6695, 2020.

[36] Edgar Dobriban and Yue Sheng.
Wonder: Weighted one-shot distributed ridge regression in high dimensions.
J. Mach. Learn. Res., 21(66):1–52, 2020.

[37] Daniel LeJeune, Hamid Javadi, and Richard Baraniuk.
The implicit regularization of ordinary least squares ensembles.
In *International Conference on Artificial Intelligence and Statistics*, pages 3525–3535. PMLR, 2020.

References XI

[38] Edgar Dobriban and Yue Sheng.
Distributed linear regression by averaging.
The Annals of Statistics, 49(2):918–943, 2021.

[39] Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert.
An algorithm for the principal component analysis of large data sets.
SIAM Journal on Scientific computing, 33(5):2580–2594, 2011.

[40] Fan Yang, Sifan Liu, Edgar Dobriban, and David P Woodruff.
How to reduce dimension with PCA and random projections?
arXiv preprint arXiv:2005.00511, 2020.

[41] Michał Dereziński, Feynman T Liang, Zhenyu Liao, and Michael W Mahoney.
Precise expressions for random projections: Low-rank approximation and randomized newton.
Advances in Neural Information Processing Systems, 33, 2020.

References XII

[42] Rui Yuan, Alessandro Lazaric, and Robert M Gower.
Sketched newton–raphson.
SIAM Journal on Optimization, 32(3):1555–1583, 2022.

[43] Jonathan Lacotte and Mert Pilanci.
Faster least squares optimization.
arXiv preprint arXiv:1911.02675, 2019.

[44] Jonathan Lacotte, Sifan Liu, Edgar Dobriban, and Mert Pilanci.
Optimal iterative sketching methods with the subsampled randomized hadamard transform.
Advances in Neural Information Processing Systems, 33:9725–9735, 2020.

[45] Michał Dereziński, Jonathan Lacotte, Mert Pilanci, and Michael W Mahoney.
Newton-less: Sparsification without trade-offs for the sketched newton update.
Advances in Neural Information Processing Systems, 34:2835–2847, 2021.

References XIII

[46] Robert M Gower and Peter Richtárik.
Randomized iterative methods for linear systems.
SIAM Journal on Matrix Analysis and Applications, 36(4):1660–1690, 2015.

[47] Elizaveta Rebrova and Deanna Needell.
On block gaussian sketching for the kaczmarz method.
Numerical Algorithms, 86:443–473, 2021.

[48] Michał Dereziński and Elizaveta Rebrova.
Sharp analysis of sketch-and-project methods via a connection to randomized singular value decomposition.
arXiv preprint arXiv:2208.09585, 2022.

[49] Michał Dereziński.
Algorithmic gaussianization through sketching: Converting data into sub-gaussian random designs.
arXiv preprint arXiv:2206.10291, 2022.