# Randomized Numerical Linear Algebra Approaches for Approximating Matrix Functions

Eugenia-Maria Kontopoulou

Department of Computer Science
Purdue University

Final Exam

Committee

Petros Drineas
David Gleich
Hemanta Maji
Kent Quanrud

## Matrix Functions

- Generalization of scalar function into multiple dimensions.

## Matrix Functions

- Generalization of scalar function into multiple dimensions.
- Multiple definitions (Hig08).

## Matrix Functions

- Generalization of scalar function into multiple dimensions.
- Multiple definitions (Hig08).
- We use the Jordan canonical form definition.

## Jordan Canonical Form

$$\mathbf{Z}\mathbf{A}\mathbf{Z}^{-1} = \mathbf{J} = \operatorname{diag}\left(\mathbf{J}_1,\ \mathbf{J}_2,\ \ldots,\ \mathbf{J}_p\right)$$

$$\mathbf{J}_k = \mathbf{J}_k(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix}$$

$\mathbf{A} \in \mathbb{C}^{n\times n}$: Input matrix.

$\mathbf{J} \in \mathbb{C}^{n\times n}$: Jordan matrix.

$\mathbf{Z} \in \mathbb{C}^{n\times n}$: non-singular matrix.

$\mathbf{J}_k \in \mathbb{C}^{m_k\times m_k}$: $k$-th Jordan block.

$\lambda_k$: $k$-th eigenvalue of $\mathbf{A}$.

## Matrix Functions

- Generalization of scalar function into multiple dimensions.
- Multiple definitions (Hig08).
- We use the Jordan canonical form definition.
- Consider positive semi-definite matrices (PSD).

**PSD Jordan Canonical Form**

$$\mathbf{Z}\mathbf{J}\mathbf{Z}^{-1} \equiv \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{*} \equiv \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{*}$$

Any PSD matrix, $\mathbf{A} \in \mathbb{C}^{n \times n}$:

1. has only real eigenvalues, $0 \leq \lambda_1, \lambda_2, \ldots, \lambda_n$;
2. has orthogonal eigenvectors, $\mathbf{U}$;
3. is always diagonalizable : $\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{U}^{*}$.

## Matrix Functions

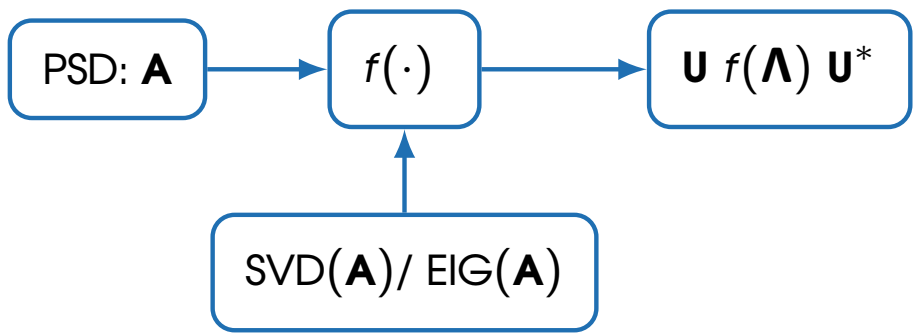
- Generalization of scalar function into multiple dimensions.
- Multiple definitions (Hig08).
- We use the Jordan canonical form definition.
- Consider positive semi-definite matrices (PSD).

**PSD Jordan Canonical Form**

$$\mathbf{Z}\mathbf{J}\mathbf{Z}^{-1} \equiv \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{*} \equiv \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{*}$$

Any PSD matrix, $\mathbf{A} \in \mathbb{C}^{n \times n}$:

1. has only real eigenvalues, $0 \leq \lambda_1, \lambda_2, \ldots, \lambda_n$;
2. has orthogonal eigenvectors, $\mathbf{U}$;
3. is always diagonalizable : $\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{U}^{*}$.



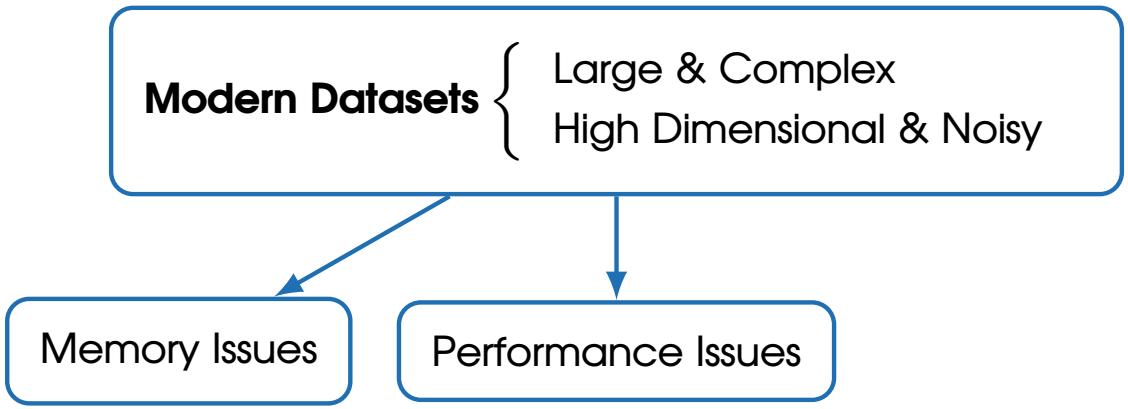
## The Big Data Problem in Numerical Linear Algebra

**Modern Datasets** { Large & Complex
High Dimensional & Noisy

## The Big Data Problem in Numerical Linear Algebra

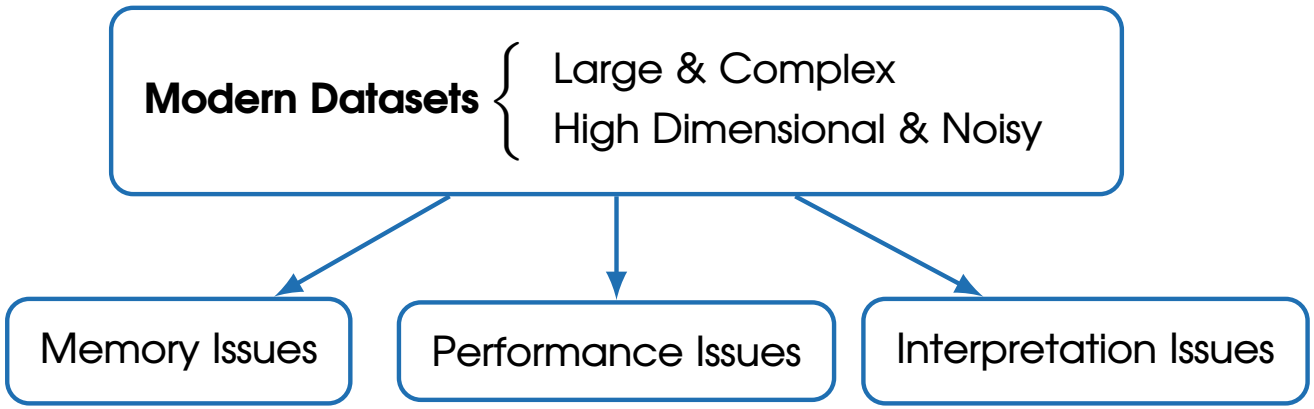

SVD/EIG require **A** in RAM!

## The Big Data Problem in Numerical Linear Algebra



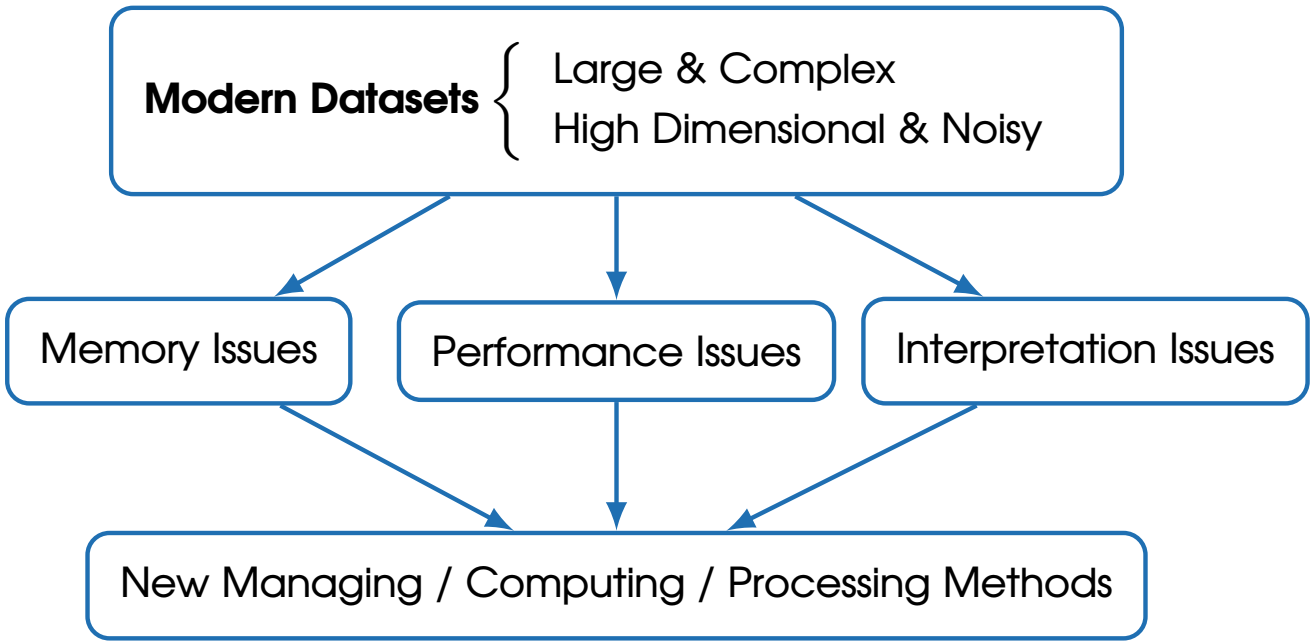

SVD/EIG require $\mathcal{O}\left(n^3\right)$ flops!

## The Big Data Problem in Numerical Linear Algebra

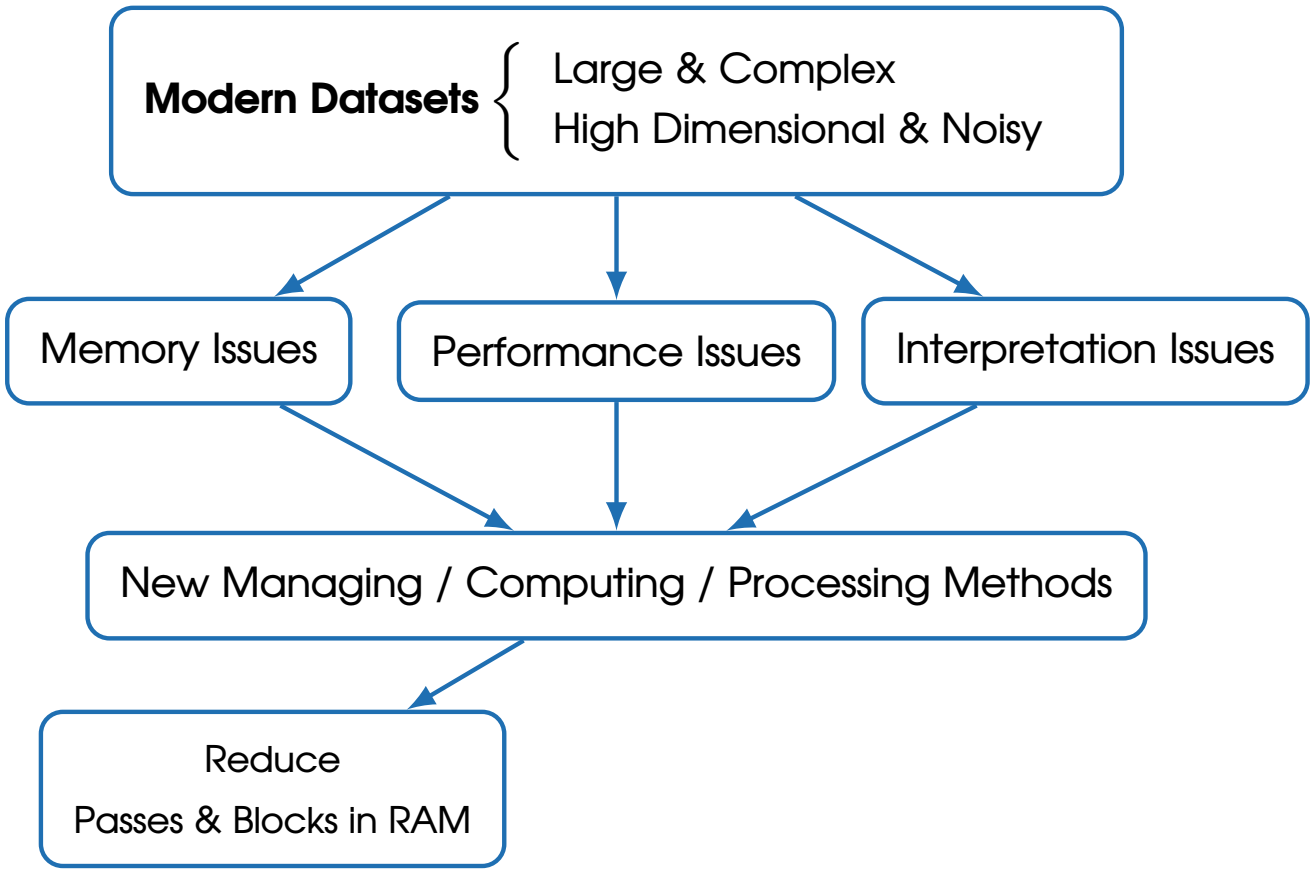



SVD represents observations as linear combination of features!

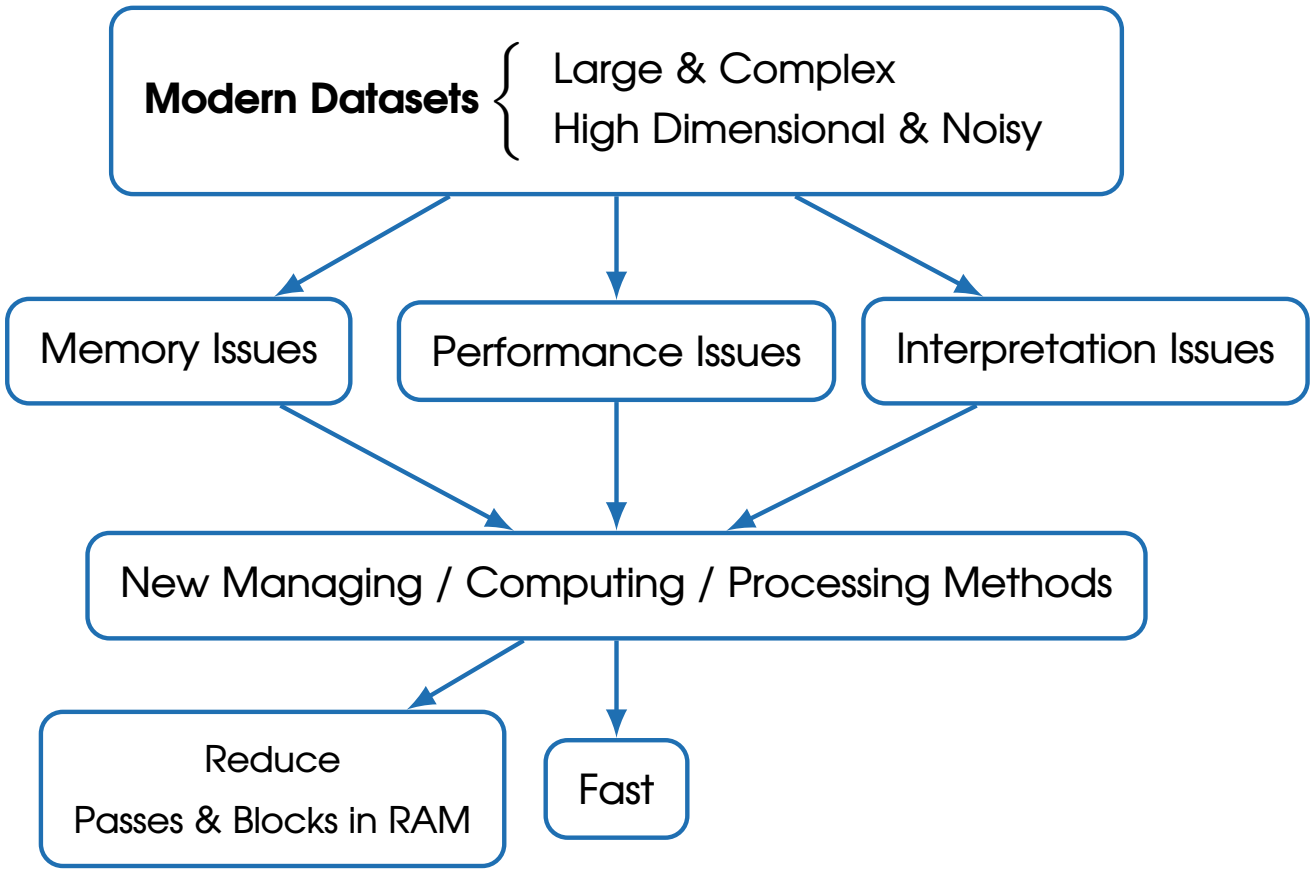
## The Big Data Problem in Numerical Linear Algebra



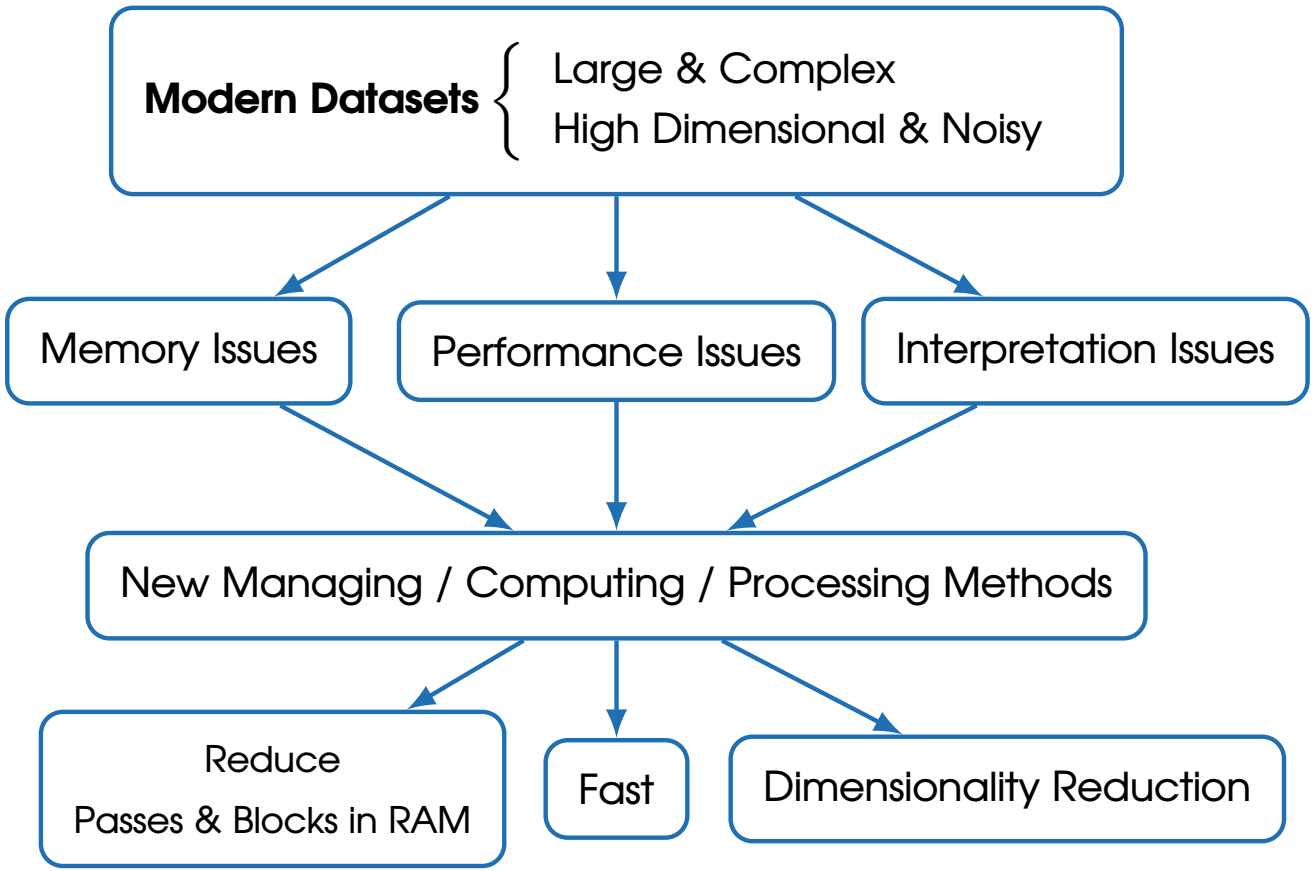
# The Big Data Problem in Numerical Linear Algebra



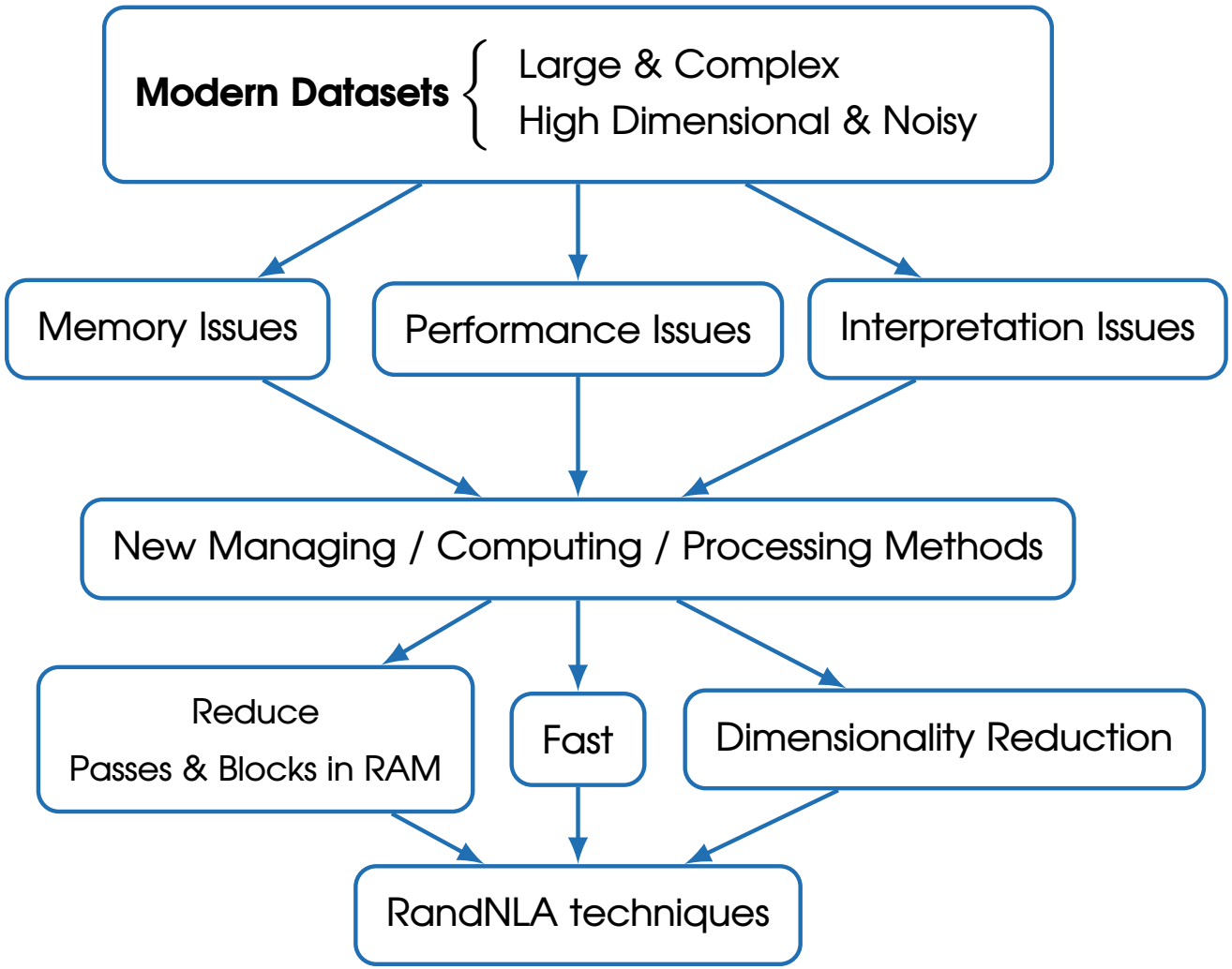
## The Big Data Problem in Numerical Linear Algebra



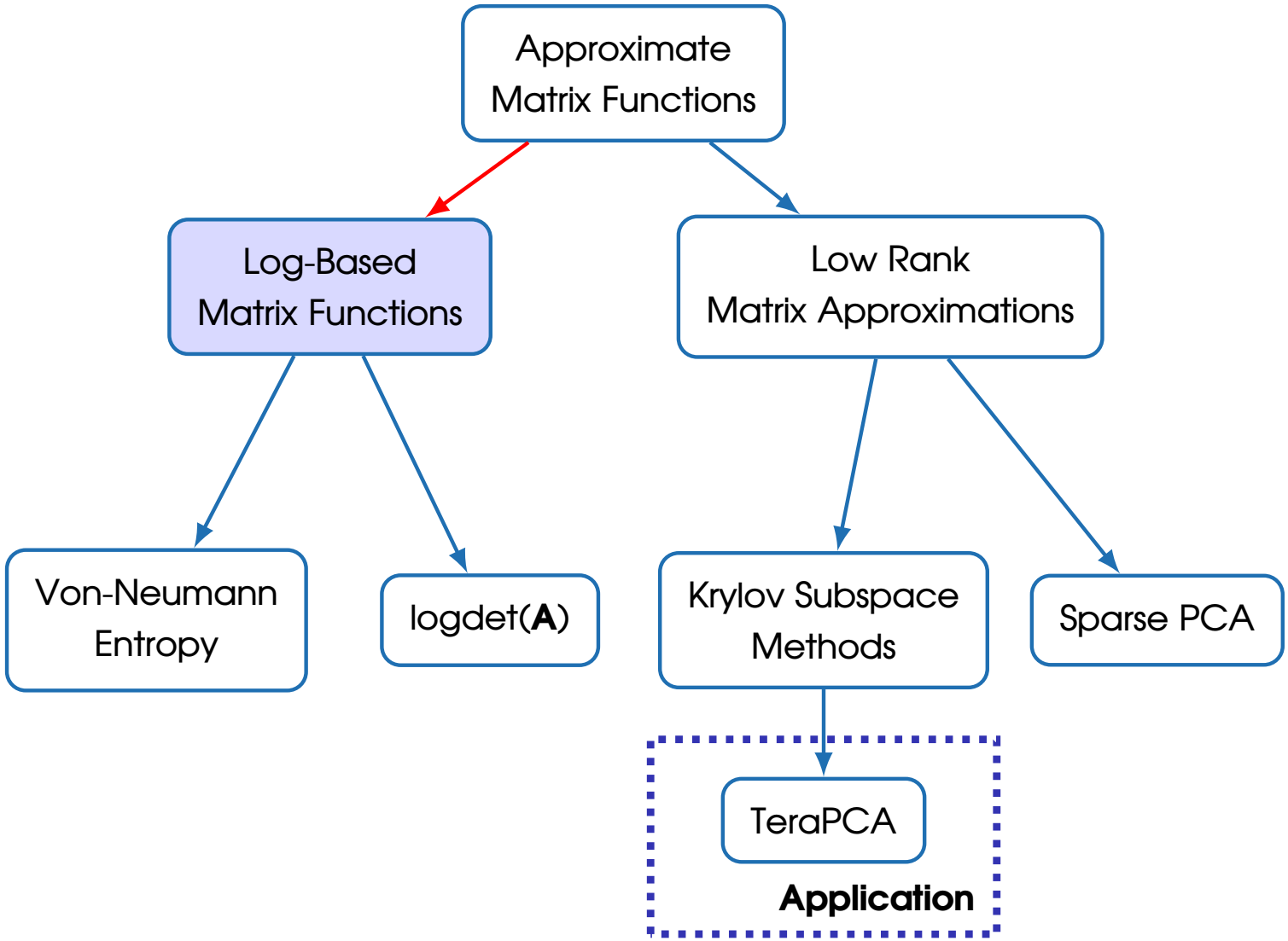
# The Big Data Problem in Numerical Linear Algebra



## The Big Data Problem in Numerical Linear Algebra



# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

## Functions of Form

$$f\left(\log\left(g\left(\mathbf{A}\right)\right)\right) = \gamma$$

where $f(\cdot)$ is a matrix or scalar function, $g(\cdot)$ is a matrix function, $\mathbf{A} \in \mathbb{C}^{n \times n}$ is a PSD and $\gamma \in \mathbb{R}$.

## Functions of Form

$$f\left(\log\left(g\left(\mathbf{A}\right)\right)\right) = \gamma$$

where $f(\cdot)$ is a matrix or scalar function, $g(\cdot)$ is a matrix function, $\mathbf{A} \in \mathbb{C}^{n \times n}$ is a PSD and $\gamma \in \mathbb{R}$.

## Von Neumann Entropy

$$\mathcal{H}\left[\mathbf{A}\right] = -Tr\left[\mathbf{A}\log\left[\mathbf{A}\right]\right]$$

$$f(\mathbf{X}) = -Tr\left[\mathbf{X} \cdot \exp\left[\mathbf{X}\right]\right] : \mathbb{C}^{n \times n} \rightarrow \mathbb{R}$$
$$g(\mathbf{X}) = \mathbf{X} : \mathbb{C}^{n \times n} \rightarrow \mathbb{C}^{n \times n}$$

## Functions of Form

$$f\left(\log\left(g\left(\mathbf{A}\right)\right)\right) = \gamma$$

where $f(\cdot)$ is a matrix or scalar function, $g(\cdot)$ is a matrix function, $\mathbf{A} \in \mathbb{C}^{n \times n}$ is a PSD and $\gamma \in \mathbb{R}$.

## Von Neumann Entropy

$$\mathcal{H}\left[\mathbf{A}\right] = -Tr\left[\mathbf{A}\log\left[\mathbf{A}\right]\right]$$

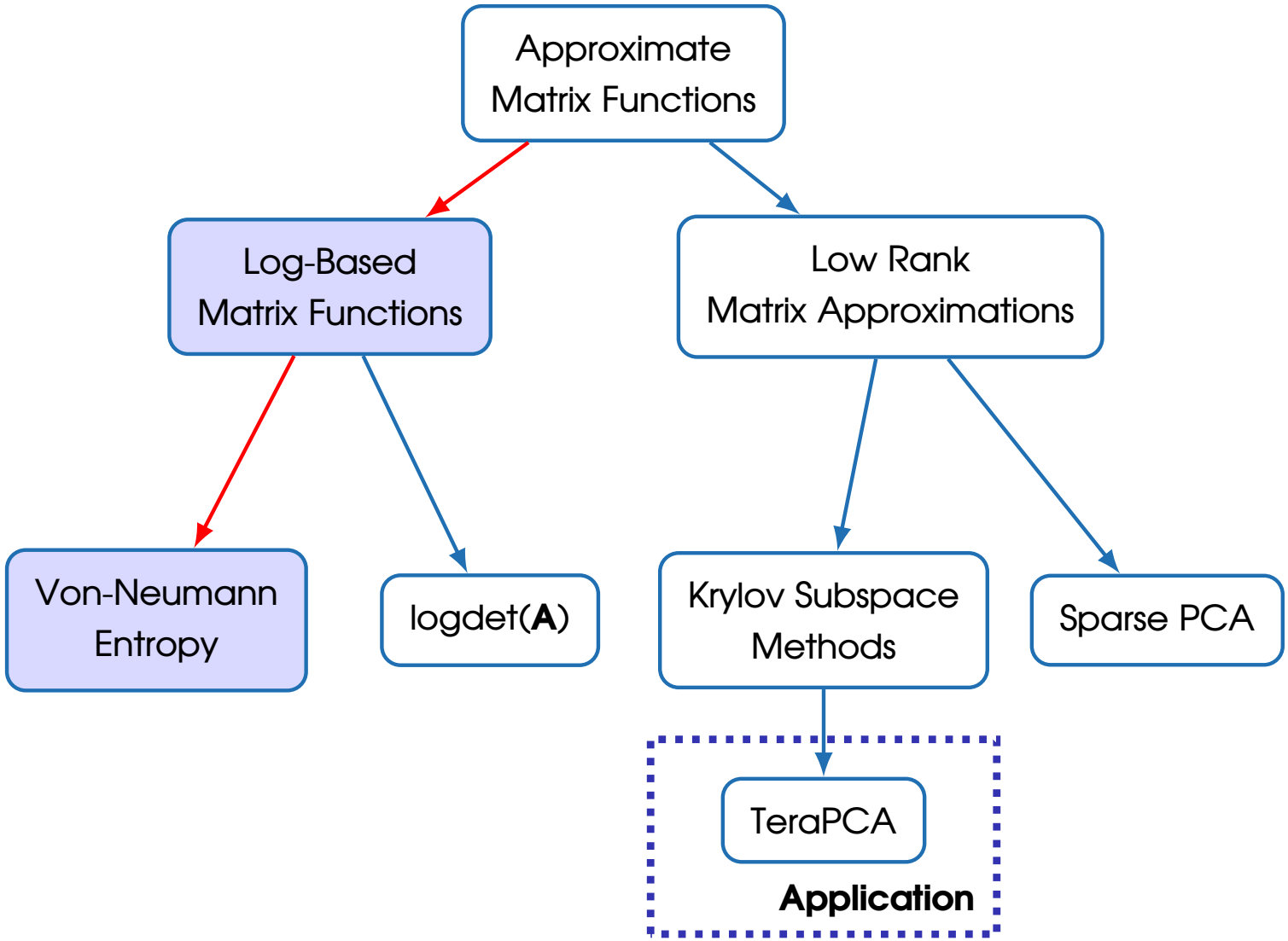
$$f(\mathbf{X}) = -Tr\left[\mathbf{X} \cdot \exp\left[\mathbf{X}\right]\right] : \mathbb{C}^{n \times n} \to \mathbb{R}$$
$$g(\mathbf{X}) = \mathbf{X} : \mathbb{C}^{n \times n} \to \mathbb{C}^{n \times n}$$

## Logarithm of Determinant

$$\mathrm{logdet}\left[\mathbf{A}\right] = \log\left(\det\left[\mathbf{A}\right]\right)$$

$$f(x) = x : \mathbb{R} \to \mathbb{R}$$
$$g(\mathbf{X}) = \det\left[\mathbf{X}\right] : \mathbb{C}^{n \times n} \to \mathbb{R}$$

# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

# Von-Neumann Entropy Problem

## Definition

Given a quantum system, compute (exactly or approximately) its Von-Neumann Entropy.

Application: Information theory, quantum mechanics, . . .

## What is the Von-Neumann Entropy?

- ✓ Extension of Gibbs/Shannon entropy concept in quantum mechanics.
- ✓ Described in 1932 by Von-Neumann in his book "Mathematische Grundlagen der Quantenmechanik".
- ✓ Fundamental notion: Density Matrix.

# Von-Neumann Entropy of Real Density Matrices I

## Definition

A Density Matrix is represented by the statistical mixture of pure states and has the form

$$\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Lambda}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n},$$

where the vectors $\mathbf{y}_i \in \mathbb{R}^n$ represent the pure states of a system and are pairwise orthogonal and normal, while $p_i$'s correspond to the probability of each state and satisfy $p_i > 0$ and $\sum_{i=1}^{n} p_i = 1$.

# Von-Neumann Entropy of Real Density Matrices II

## Straightforward Computation

1. Compute the eigenvalues of $\mathbf{R}$, $p_1, p_2, \ldots, p_n$ (e.g. using eigendecomposition).
2. Compute the Von-Neumann Entropy of $\mathbf{R}$ using $p_i$, $i = 1, \ldots, n$:

$$\mathcal{H}[\mathbf{R}] = -\sum_{i=1}^{n} p_i \log p_i$$

Time Complexity: $\mathcal{O}\left(n^3\right)$.

## Mathematical Manipulation of $\mathcal{H}[\mathbf{R}]$

Consider the function $h(x) = x\log(x) \in \mathbb{R}$.

$$
\begin{aligned}
h[\mathbf{R}] &= \mathbf{R}\log[\mathbf{R}] \\
&= \mathbf{Y}\Sigma_p\mathbf{Y}^\top \log\left[\mathbf{Y}\Sigma_p\mathbf{Y}^\top\right] \\
&= \mathbf{Y}\Sigma_p \log[\Sigma_p]\,\mathbf{Y}^\top \\
&= \mathbf{Y}h[\Sigma_p]\,\mathbf{Y}^\top
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{H}[\mathbf{R}] &= -\sum_{i=1}^{n} p_i \log p_i \\
&= -Tr[h[\Sigma_p]] \\
&= -Tr\left[\mathbf{Y}^\top\mathbf{Y}h[\Sigma_p]\right] \\
&= -Tr\left[\mathbf{Y}h[\Sigma_p]\,\mathbf{Y}^\top\right] \\
&= -Tr[h[\mathbf{R}]]
\end{aligned}
$$

### Two Approaches

1. Using a Taylor expansion for the logarithm we can further manipulate $\mathcal{H}[\mathbf{R}]$.
2. Approximate $h[\mathbf{R}]$ with Chebyshev Polynomials.

### Two Randomized Numerical Linear Algebra tools

1. Power method with provable bounds (Bou+17; Tre11).
2. Randomized trace estimators (AT11).

## Lemma

*Let* $\mathbf{R} \in \mathbb{R}^{n \times n}$ *be a density matrix whose probabilities lie in the interval* $[\ell, u]$*, for some* $0 < \ell \leq u \leq 1$*. Then,*

$$\mathcal{H}[\mathbf{R}] = \log(u^{-1}) + \underbrace{\sum_{k=1}^{\infty} \frac{Tr\left[\mathbf{R}(\mathbf{I} - u^{-1}\mathbf{R})^k\right]}{k}}_{\Delta}$$

We estimate the trace of $\mathbf{R}(\mathbf{I} - u^{-1}\mathbf{R})^k$ using Gaussian trace estimator and $\Delta$ by truncation. The largest eigenvalue, $u$, is estimated using the power method with provable bounds.

---

Gaussian Trace Estimator Lemma
Power Method Lemma
Proof

**Input:** $R \in \mathbb{R}^{n \times n}$, accuracy parameter $\varepsilon > 0$, integer $m > 0$.
**Output:** $\widehat{\mathcal{H}}[\mathbf{R}]$, the approximation to the $\mathcal{H}[R]$.

1: Compute $\hat{p}_1$, the estimation of the largest singular value of $R$, using power method.
2: Set $u = \min\{1, 6\hat{p}_1\}$
3: $C = I_n - u^{-1}R$
4: Generate $s = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.
5: Compute $\widehat{\mathcal{H}}[\mathbf{R}]$ as:

$$\widehat{\mathcal{H}}[\mathbf{R}] = \log u^{-1} + \frac{1}{s} \sum_{i=1}^{s} \sum_{k=1}^{m} \frac{g_i^{\top} R C^k g_i}{k}$$

## Theorem

*Let* $\mathbf{R}$ *be a density matrix such that all probabilities* $p_i$, $i = 1 \ldots n$ *satisfy* $0 < \ell \leq p_i$. *Let* $u$ *be computed using the power method and let* $\widehat{\mathcal{H}(\mathbf{R})}$ *be the output of the Taylor-based Algorithm on inputs* $\mathbf{R}$, $m$, *and* $\epsilon < 1$; *Then, with probability at least* $1 - 2\delta$,

$$\left| \widehat{\mathcal{H}(\mathbf{R})} - \mathcal{H}\left[\mathbf{R}\right] \right| \leq 2\epsilon \mathcal{H}\left[\mathbf{R}\right]$$

*by setting* $m = \left\lceil \frac{u}{\ell} \log\left(1/\epsilon\right) \right\rceil$.

## Running Time

$$\mathcal{O}\left( \frac{u}{\ell} \cdot \frac{\log\left(1/\varepsilon\right) \log\left(1/\delta\right)}{\varepsilon^2} \cdot \mathrm{nnz}\left(\mathbf{R}\right) + \log\left(n\right) \cdot \log\left(1/\delta\right) \cdot \mathrm{nnz}\left(\mathbf{R}\right) \right)$$

---

Sketch of the Proof

## Lemma

*We can approximate* $h(x) = x\log(x)$ *in the interval* $(0, u]$ *by*

$$f_m(x) = \sum_{w=0}^{m} \alpha_w \mathcal{T}_w(x)$$

*where* $\mathcal{T}_w(x) = \cos(w \cdot \arccos((2/u)x - 1))$*, the Chebyshev polynomials of the first kind for* $w > 0$ *and,*

$$\alpha_0 = \frac{u}{2}\left(\log\frac{u}{4} + 1\right), \quad \alpha_1 = \frac{u}{4}\left(2\log\frac{u}{4} + 3\right), \quad \text{and} \quad \alpha_w = \frac{(-1)^w u}{w^3 - w} \text{ for } w \geq 2$$

*For any* $m \geq 1$*,*

$$|h(x) - f_m(x)| \leq \frac{u}{2m(m+1)} \leq \frac{u}{2m^2}$$

*for* $x \in [0, u]$*.*

Using the Lemma we approximate $\mathcal{H}(\mathbf{R})$ by $\widehat{\mathcal{H}(\mathbf{R})}$ as follows:

$$\begin{aligned}\mathcal{H}(\mathbf{R}) &= -Tr\left[h\left[\mathbf{R}\right]\right] \\ &\approx -Tr\left[f_m\left[\mathbf{R}\right]\right] \\ &\approx -\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^{\top} f_m\left[\mathbf{R}\right]\mathbf{g}_i \\ &= \widehat{\mathcal{H}(\mathbf{R})}\end{aligned}$$

We estimate $u$ using the power method and $Tr\left[f_m\left[\mathbf{R}\right]\right]$ using a Gaussian trace estimator. We compute the scalars $\mathbf{g}_i^{\top} f_m\left[\mathbf{R}\right]\mathbf{g}_i$ using the Clenshaw algorithm.

**Input:** $R \in \mathbb{R}^{n \times n}$, accuracy parameter $\varepsilon > 0$, integer $m > 0$.
**Output:** $\widehat{\mathcal{H}}[\mathbf{R}]$, the approximation to the $\mathcal{H}[R]$.

1: Compute $\hat{p}_1$, the estimation of the largest singular value of $R$, using power method.
2: Set $u = \min\{1, 6\hat{p}_1\}$
3: Generate $s = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.
4: Compute $\widehat{\mathcal{H}}[\mathbf{R}]$ as:

$$\widehat{\mathcal{H}}[\mathbf{R}] = -\frac{1}{s} \sum_{i=1}^{s} g_i^\top f_m(R) g_i$$

## Theorem

*Let* $\mathbf{R}$ *be a density matrix such that all probabilities* $p_i$, $i = 1 \ldots n$ *satisfy* $0 < \ell \leq p_i$. *Let* $u$ *be computed using the power method and let* $\widehat{\mathcal{H}(\mathbf{R})}$ *be the output of the Chebyshev-based Algorithm on inputs* $\mathbf{R}$, $m$, *and* $\epsilon < 1$; *Then, with probability at least* $1 - 2\delta$,

$$\left| \widehat{\mathcal{H}(\mathbf{R})} - \mathcal{H}\left[\mathbf{R}\right] \right| \leq 3\epsilon \mathcal{H}\left[\mathbf{R}\right]$$

*by setting* $m = \sqrt{\frac{u}{2\epsilon\ell \ln(1/(1-\ell))}}$

## Running Time

$$\mathcal{O}\left(\sqrt{\frac{u}{\ell}} \cdot \sqrt{\frac{1}{\log(1/(1-\ell))}} \cdot \frac{\log(1/\delta)}{\varepsilon^{2.5}} \cdot \mathrm{nnz}\left(\mathbf{R}\right) + \log(n) \cdot \log(1/\delta) \cdot \mathrm{nnz}\left(\mathbf{R}\right)\right)$$

## Theorem

*Every Hermitian matrix* $\mathbf{A} \in \mathbb{C}^{n \times n}$ *can be expressed as*

$$\mathbf{A} = \mathbf{B} + i\mathbf{C} \tag{1}$$

*where* $\mathbf{B} \in \mathbb{R}^{n \times n}$ *is symmetric and* $\mathbf{C} \in \mathbb{R}^{n \times n}$ *is anti-symmetric (or skew-symmetric). If* $\mathbf{A} \in \mathbb{C}^{n \times n}$ *is positive semi-definite, then* $\mathbf{B}$ *is also positive semi-definite.*

## Theorem

*The trace of a Hermitian matrix* $\mathbf{A} \in \mathbb{C}^{n \times n}$ *expressed as in eqn. (1) is equal to the trace of its real part:*

$$Tr\left[\mathbf{A}\right] = Tr\left[\mathbf{B}\right]$$

## Algorithmic design

- The trace estimator works for Hermitian PSD matrices.
- Taylor and Chebyshev polynomials work in complex space.
- No guarantees are known for power method $\rightarrow$ set $u = 1$.

## Low Rank Density Matrices

Assume that the density matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$, has at most $k$ non-zero probabilities, $p_i$. This means that at most $k$ of its states are pure.

### Issue & Solution

- ✗ $n - k$ probabilities are zero $\rightarrow$ Chebyshev/Taylor approaches are not working.
- ✓ Project to a smaller full-dimension space $\rightarrow$ Random Projections.
- ✓ Fast construction of the random projector.

### Construction of the Random Projector

- Gaussian Random Projector
- Sub-sampled Randomized Hadamard Transform
- Input Sparsity Transform
- Hartley Transform

**Input:** $R \in \mathbb{R}^{n \times n}$, integer $k \ll n$.

**Output:** $\widehat{\mathcal{H}(R)}$, the approximation to the $\mathcal{H}[R]$.

1: Construct the random projection matrix $\Pi \in \mathbb{R}^{n \times s}$.
2: Compute $\tilde{R} = R\Pi \in \mathbb{R}^{n \times s}$.
3: Compute and return the (at most) $k$ non-zero singular values of $\tilde{R}$, $\tilde{p}_i$, $i = 1 \ldots k$.
4: Compute $\widehat{\mathcal{H}(R)}$ as:

$$\widehat{\mathcal{H}(R)} = \sum_{i=1}^{k} \tilde{p}_i \log \frac{1}{\tilde{p}_i}$$

## Theorem

*Let* $\mathbf{R}$ *be a density matrix with at most* $k \ll n$ *non-zero probabilities and let* $\varepsilon < 1/2$ *be an accuracy parameter. Then, with probability at least 0.9, the output of the Random Projection based Algorithm satisfies*

$$\left|p_i^2 - \tilde{p}_i^2\right| \leq \varepsilon \cdot p_i^2$$

*for all* $i = 1 \ldots k$. *Additionally,*

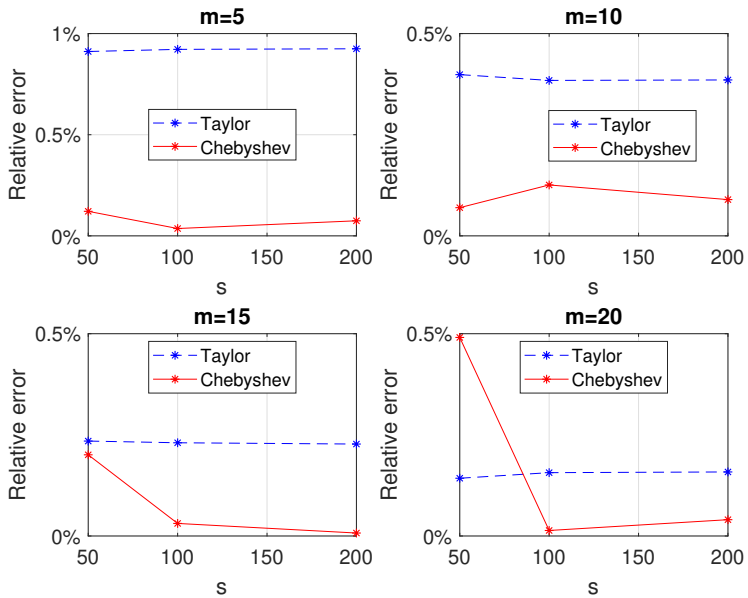
$$\left|\mathcal{H}(\mathbf{R}) - \widehat{\mathcal{H}(\mathbf{R})}\right| \leq \sqrt{\varepsilon}\mathcal{H}(\mathbf{R}) + \sqrt{\frac{3}{2}\varepsilon}$$

## Running Time

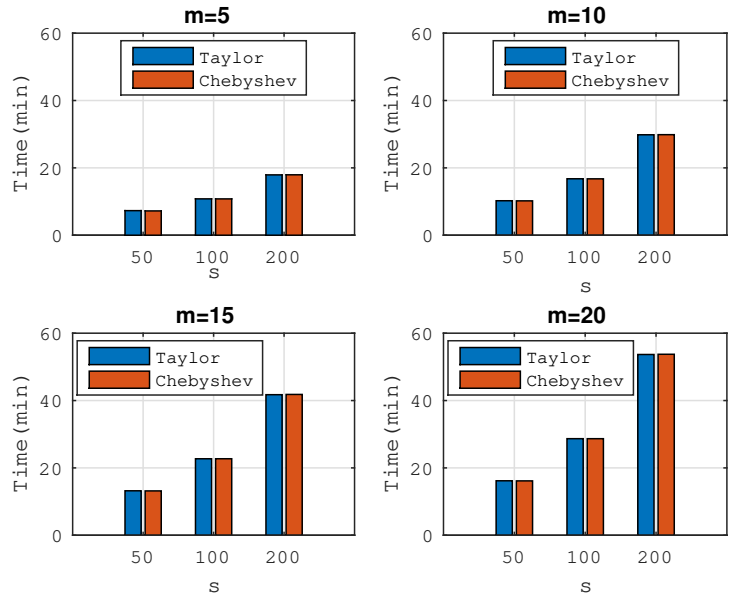
Algorithm 4 (combined with the Input Sparsity Transform) runs in time

$$\mathcal{O}\left(\mathrm{nnz}\left(\mathbf{R}\right) + \mathrm{nk}^4/\varepsilon^4\right)$$

Matrix of size $30,000 \times 30,000$, $m = [5 : 5 : 20]$ and $u \approx \lambda_{max}$.




$s = [50 : 50 : 200]$

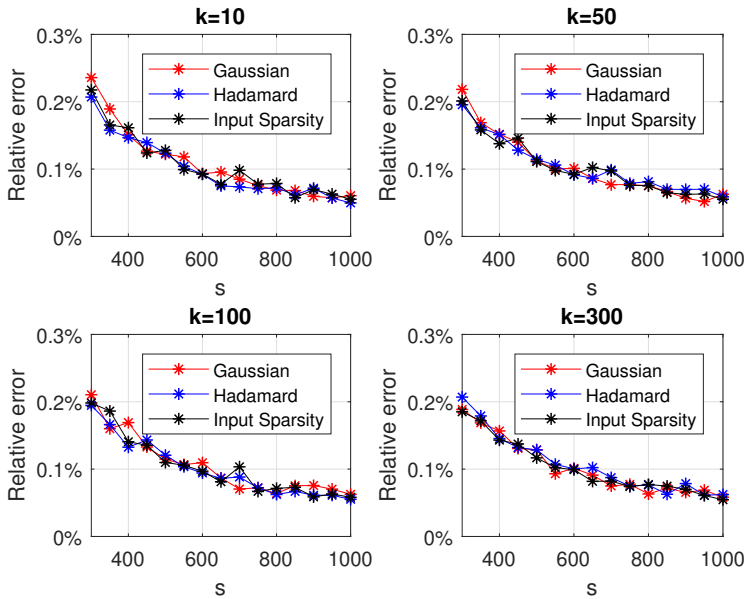



$s = \{50, 100, 200\}$

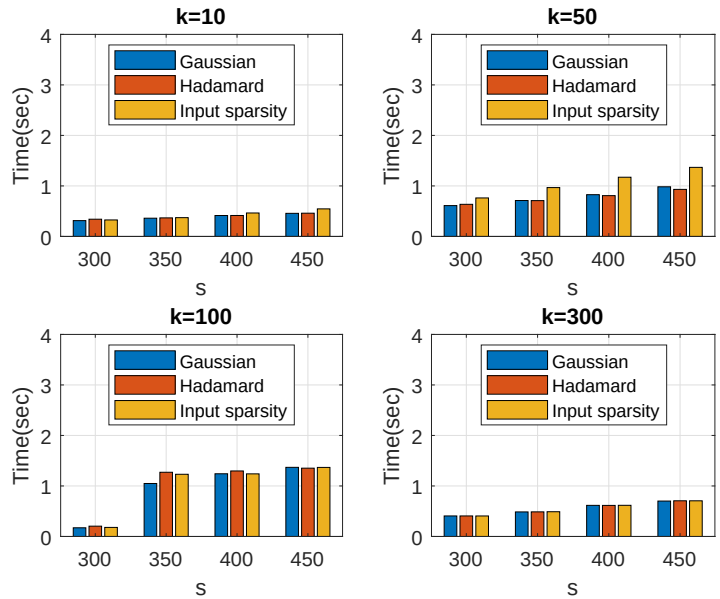
## Notes

- Exact computation: 5.6 hours.
- Approximation of $\lambda_{max}$: 3.6 minutes.

Matrix of size $4,096 \times 4,096$ and $k = \{10, 50, 100, 300\}$.




$s = \{400, 600, 800, 1000\}$




$s = [300 : 50 : 450]$

Exact computation for various $k$.

| **k** | **10** | **50** | **100** | **300** |
|---|---|---|---|---|
| **Time** | 1.5 sec | 8 sec | 15 sec | 1 min |

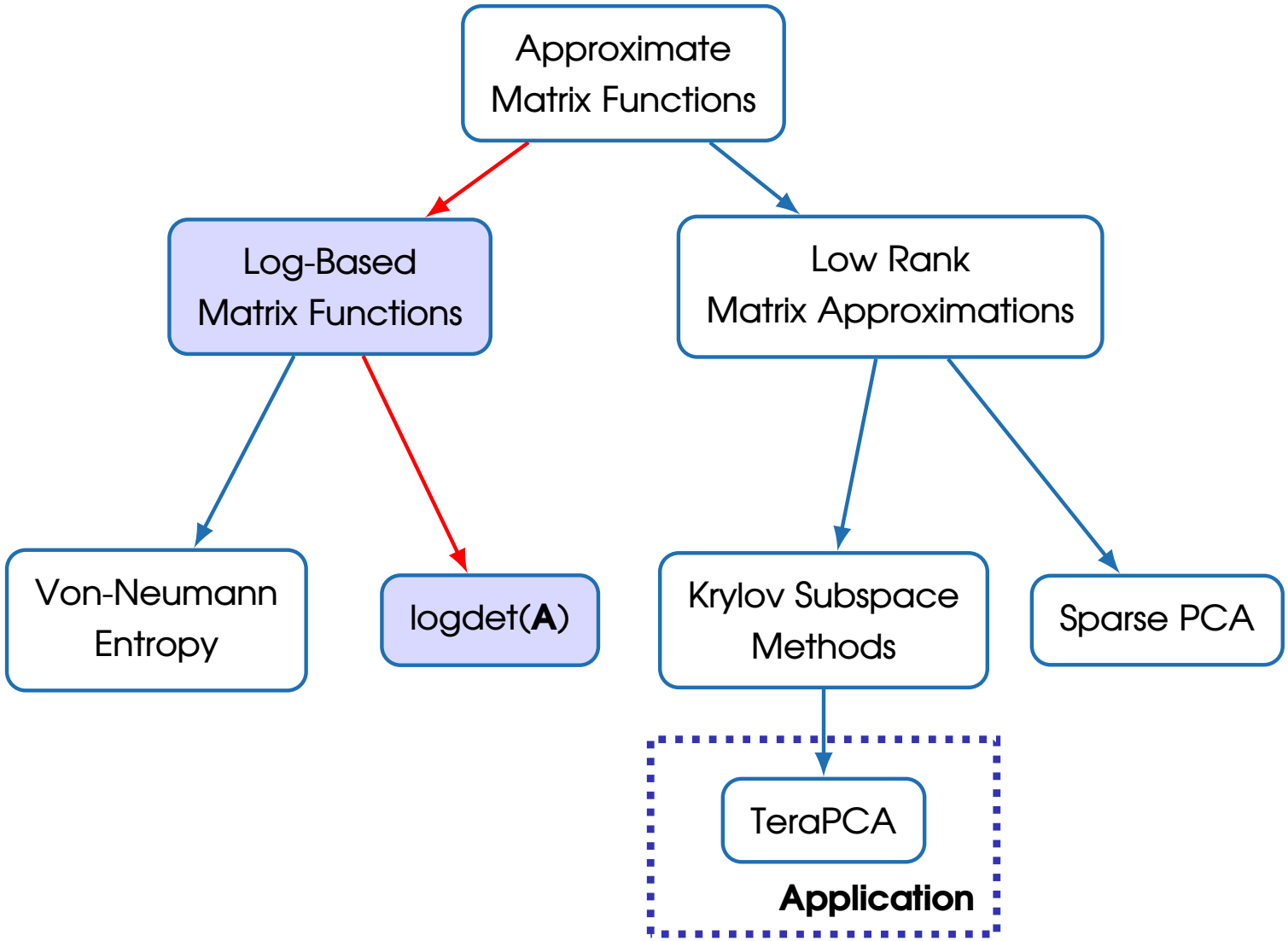
More Experiments

# Publications


(Kon+18) E. Kontopoulou, A. Grama, W. Szpankowski, P. Drineas, ``**Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices**'', in Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT), pp. 2486-2490

(Kon+20) E. Kontopoulou, G. Dexter, A. Grama, W. Szpankowski & P. Drineas, ``**Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices**'', in IEEE Transactions on Information Theory, to appear

# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

## Definition

Given a Symmetric Positive Definite (SPD) matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, compute (exactly or approximately) $\mathrm{logdet}\,[\mathbf{A}]$.

Application: Maximum likelihood estimations, Gaussian processes prediction, logdet-divergence metric, barrier functions in interior point methods . . .

## Straightforward Computation

1. Compute the Cholesky Factorization of $\mathbf{A}$, and let $\mathbf{L}$ be the Cholesky factor.
2. Compute the log-determinant of $\mathbf{A}$ using $\mathbf{L}$:

$$\mathrm{logdet}\,[\mathbf{A}] = \mathrm{logdet}\,[\mathbf{L}]^2 = 2\log\prod_{i=1}^{n} l_{ii} = 2\sum_{i=1}^{n}\log(l_{ii}) = 2Tr\,[\log\,[\mathbf{L}]]$$

Time Complexity: $\mathcal{O}\left(n^3\right)$

## Additive Error Approximation

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an SPD matrix whose dominant eigenvalue is bounded by $\alpha$. Then,

$$\operatorname{logdet}[\mathbf{A}] \approx n \log(u) - \sum_{k=1}^{m} \frac{1}{k} \left( \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^{\top} \left( \mathbf{I}_n - \alpha^{-1} \mathbf{A} \right)^k \mathbf{g}_i \right)$$

## Lemma

*Let* $\widehat{\text{logdet}}\,[\mathbf{A}]$ *be the approximation of* $\text{logdet}\,[\mathbf{A}]$ *using the LogDetAdditive Algorithm on inputs* $\mathbf{A}$, $m$ *and* $\varepsilon$. *Then, we* **prove** *that with probability at least* $1 - 2\delta$,

$$\left|\widehat{\text{logdet}}\,[\mathbf{A}] - \text{logdet}\,[\mathbf{A}]\right| \leq 2\varepsilon \sum_{i=1}^{n} \log\left(7 \cdot \kappa\left(\mathbf{A}\right)\right)$$

*setting* $m \geq \lceil 7\kappa(\mathbf{A}) \log(\frac{1}{\varepsilon}) \rceil$

## Running Time

$$\mathcal{O}\left(7 \cdot \kappa(\mathbf{A}) \cdot \frac{1}{\varepsilon^2} \cdot \log\left(\frac{1}{\varepsilon}\right) \cdot \log\left(\frac{1}{\delta}\right) \cdot \text{nnz}\left(\mathbf{A}\right) + \log n \cdot \log\left(\frac{1}{\delta}\right) \cdot \text{nnz}\left(\mathbf{A}\right)\right)$$

## Additive Error Approximation

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an SPD matrix whose dominant eigenvalue is bounded by $\alpha$. Then,

$$\operatorname{logdet}[\mathbf{A}] \approx n \log(u) - \sum_{k=1}^{m} \frac{1}{k} \left( \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^{\top} \left( \mathbf{I}_n - \alpha^{-1} \mathbf{A} \right)^k \mathbf{g}_i \right)$$

## Relative Error Approximation

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an SPD matrix whose eigenvalues lie in the interval $(\theta_1, 1)$, for some $0 < \theta_1 < 1$. Then,

$$\operatorname{logdet}[\mathbf{A}] \approx - \sum_{k=1}^{m} \frac{1}{k} \left( \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^{\top} \left( \mathbf{I}_n - \mathbf{A} \right)^k \mathbf{g}_i \right)$$

## Lemma

*Let* $\widehat{\text{logdet}}\,[\mathbf{A}]$ *be the approximation of* $\text{logdet}\,[\mathbf{A}]$ *using the LogDetRelative Algorithm on inputs* $\mathbf{A}$, $m$ *and* $\varepsilon$. *Then, we* **prove** *that with probability at least* $1-\delta$,

$$|\widehat{\text{logdet}}\,[\mathbf{A}] - \text{logdet}\,[\mathbf{A}]\,| \leq 2\varepsilon \cdot |\text{logdet}\,[\mathbf{A}]\,|$$

*and* $m \geq \lceil \frac{1}{\theta_1} \cdot \log(\frac{1}{\varepsilon}) \rceil$

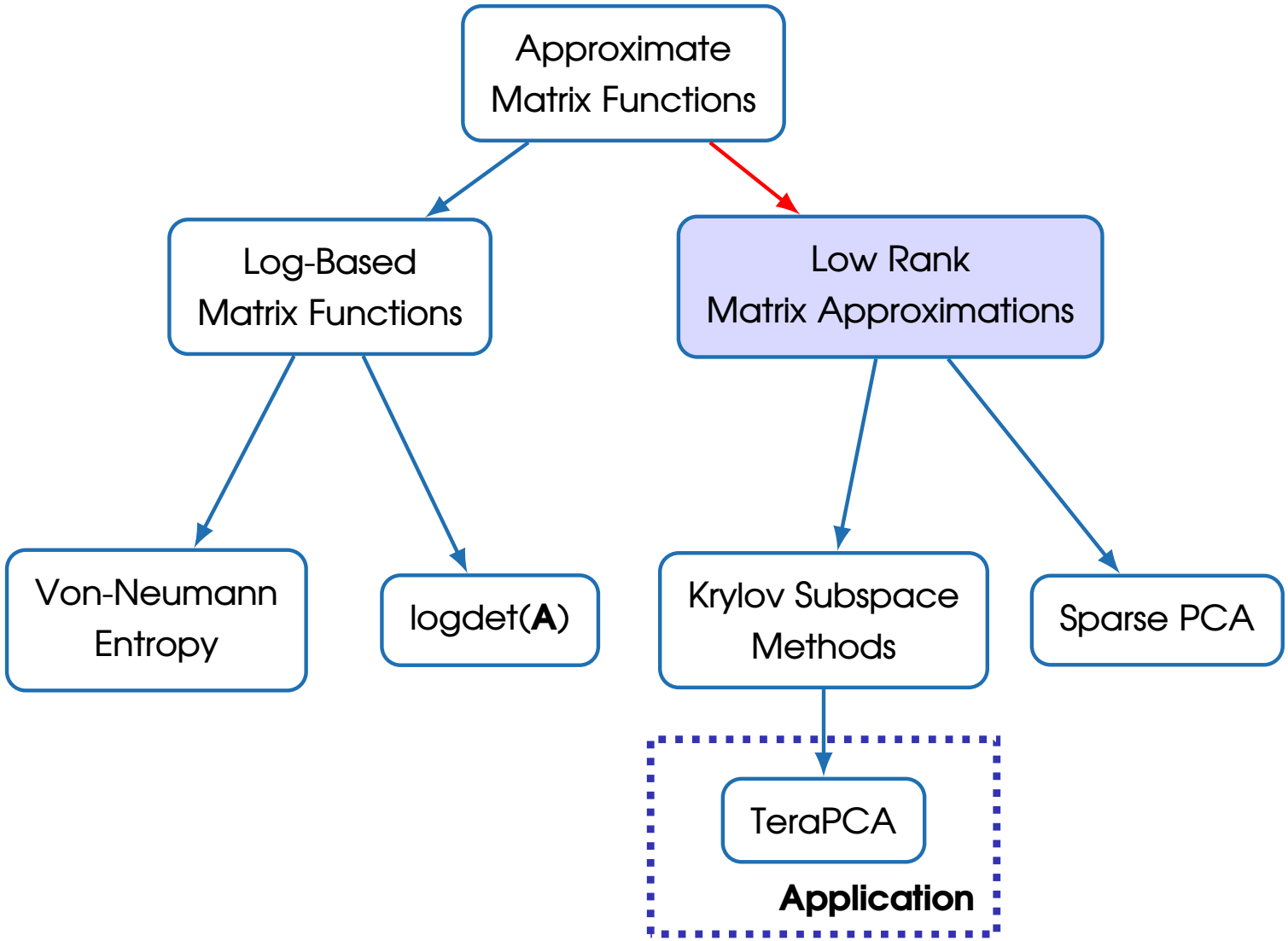
## Running Time

$$\mathcal{O}\left(\frac{1}{\theta_1} \cdot \frac{1}{\varepsilon^2} \cdot \log\left(\frac{1}{\varepsilon}\right) \cdot \log\left(\frac{1}{\delta}\right) \cdot \text{nnz}\,(\mathbf{A})\right)$$

# Publications

(Bou+17) C. Boutsidis, P. Drineas, P. Kambadur, E. Kontopoulou, A. Zouzias, ``**A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix**'', in Linear Algebra and its Applications, 533, pp.95-117.

# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

## Low Rank Approximation

Given an $m \times n$ matrix $\mathbf{A}$ and a rank parameter $k \ll \min\{m, n\}$, the Low-Rank Approximation problem is to find a matrix $\mathbf{Z}$ of rank $k$ such that $\|\mathbf{A} - \mathbf{Z}\|_{2,F}$ is sufficient small.

## Eckart-Young Theorem

The minimization problem:

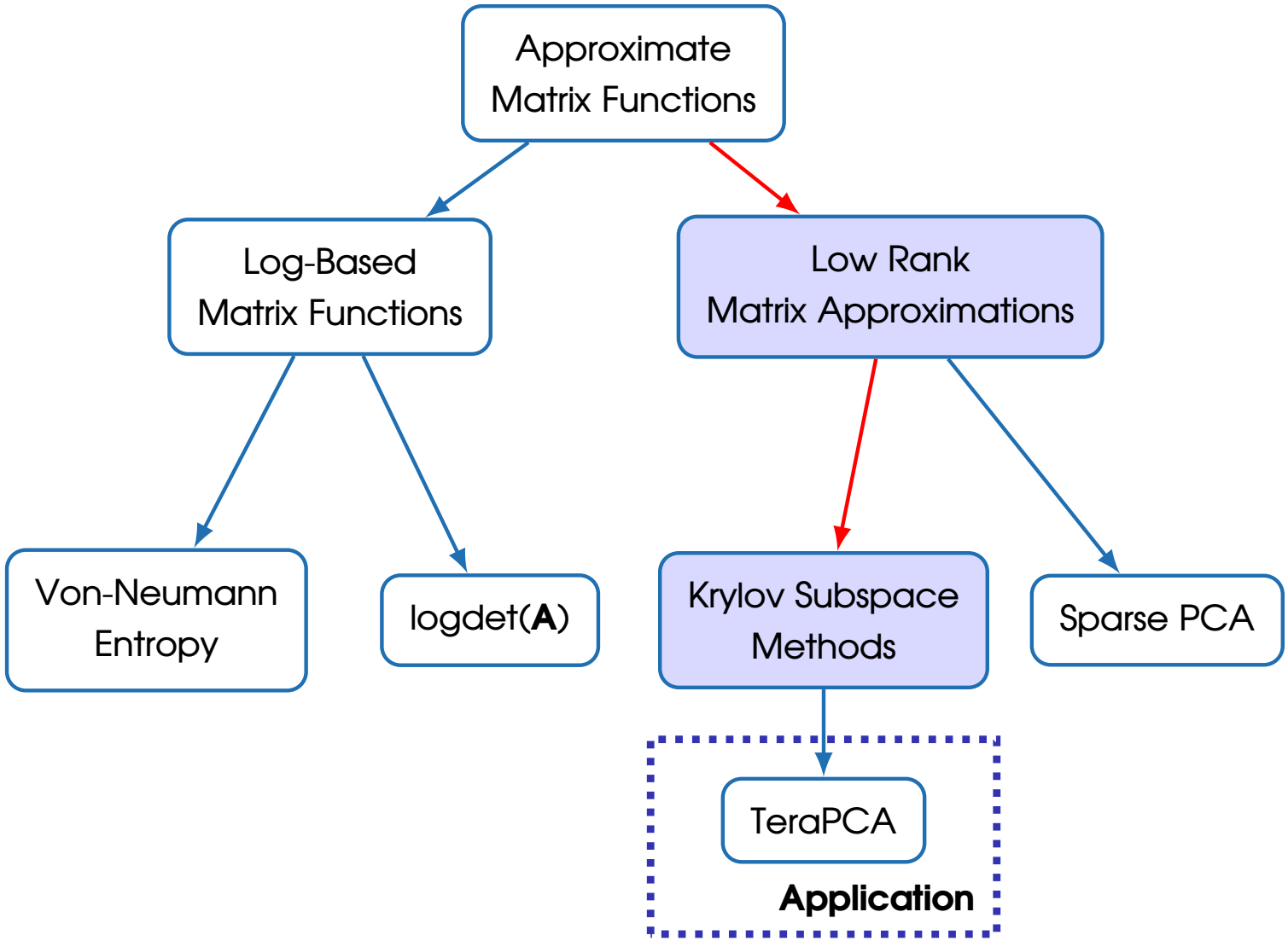
$$\min_{\mathrm{rank}(\mathbf{Z})=\mathrm{k}} \|\mathbf{A} - \mathbf{Z}\|_{2,F}$$

has a solution given by the truncated SVD:

$$\mathbf{Z} = \mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^\top$$

We are interested in measuring the quality of the approximation to top singular vectors and the extraction of meaningful sparse principal components.

# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

Given an arbitrary matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a starting random guess $\mathbf{X} \in \mathbb{R}^{n \times s}$, we build the Krylov space in $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}\mathbf{X}$:

$$\mathcal{K}_q \equiv \mathcal{K}_q(\mathbf{A}\mathbf{A}^\top, \mathbf{A}\mathbf{X}) = \text{range}\left(\mathbf{A}\mathbf{X} \quad (\mathbf{A}\mathbf{A}^\top)\mathbf{A}\mathbf{X} \quad \dots \quad (\mathbf{A}\mathbf{A}^\top)^{\mathrm{q}}\mathbf{A}\mathbf{X}\right)$$

## Assumptions

1. We assume exact arithmetic (there are no issues of numerical stability).
2. The dimension of the Krylov Space is maximal: $\dim\left(\mathcal{K}_{\mathrm{q}}\right) = (\mathrm{q}+1)\mathrm{s}$.
3. $\sigma_k > \sigma_{k+1} > 0$, where $k$ is the number of singular vectors we seek to approximate and $\sigma_k$ ($\sigma_{k+1}$) is the $k$-th ($k+1$-st) singular value of $\mathbf{A}$.

Assume $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a positive integer $k < \operatorname{rank}(\mathbf{A})$. Let $\mathbf{U}_k$ be the top-$k$ left singular vectors of $\mathbf{A}$. The objective is to construct approximations $\widehat{\mathbf{U}_k} \in \mathbb{R}^{m \times k}$ for $\mathbf{U}_k$.

## Dominant Subspace Reconstruction

We are interested in the angles between $\operatorname{range}(\mathbf{U}_k)$ and $\operatorname{range}(\widehat{\mathbf{U}_k})$. This metric is well defined only if $\mathbf{U}_k$ is unique.

## Low Rank Approximation

We are interested in the approximation error between $\mathbf{A}$ and its projection into $\operatorname{range}(\widehat{\mathbf{U}_k})$:

$$\left\|\mathbf{A} - \widehat{\mathbf{U}_k}\widehat{\mathbf{U}_k}^{\top}\mathbf{A}\right\|_{2,F}$$

This metric is well-defined even if $\mathbf{U}_k$ is not unique.

## Dominant and Subdominant Spaces

Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the full SVD of $\mathbf{A}$ with $\mathbf{U} \in \mathbb{R}^{m\times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$, then for an integer $0 < k < \operatorname{rank}(\mathbf{A})$ we can perform the following partitioning:

$$\mathbf{A} = \underbrace{\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top}_{\text{dominant spaces}} + \underbrace{\mathbf{U}_{k,\perp}\mathbf{\Sigma}_{k,\perp}\mathbf{V}_{k,\perp}^\top}_{\text{sub-dominant spaces}}$$

## Principal Angles Matrix

Assume $\mathbf{U}_k \in \mathbb{R}^{m\times k}$ and $\mathbf{X} \in \mathbb{R}^{m\times s}$, with orthonormal columns:

**Principal Angles:**

$$\theta_i = \cos^{-1}(\sigma_i(\mathbf{U}_k^\top \mathbf{X}))$$

**Principal Angles Matrix**:

$$\mathbf{\Theta}(\mathbf{U}_k, \mathbf{X}) = \operatorname{diag}(\theta_1, \theta_2, \ldots, \theta_k) \in \mathbb{R}^{k\times k}$$

## Theorem 1

Let $\phi(x)$ be a polynomial of degree $2q+1$ with odd powers only such that $\phi(\mathbf{\Sigma}_k)$ is nonsingular. If $\mathrm{rank}\left(\mathbf{V}_k^T\mathbf{X}\right) = k$, then,

$$\| \sin \mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\|_{2,F} \leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \, \|\phi(\mathbf{\Sigma}_k)^{-1}\|_2 \, \|\mathbf{V}_{k,\perp}^T\mathbf{X}(\mathbf{V}_k^T\mathbf{X})^\dagger\|_{2,F}$$

If, in addition, $\mathbf{X}$ has orthornomal or linearly independent columns, then,

$$\|\mathbf{V}_{k,\perp}^T\mathbf{X}(\mathbf{V}_k^T\mathbf{X})^\dagger\|_{2,F} = \| \tan \mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\|_{2,F}$$

and

$$\| \sin \mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\|_{2,F} \leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \, \|\phi(\mathbf{\Sigma}_k)^{-1}\|_2 \, \| \tan \mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\|_{2,F}$$

## The starting guess $\mathbf{X}$

The starting guess $\mathbf{X}$ can be any random matrix, e.g. random Gaussian, random sign, sub-sampled randomized Hadamard transform.

## RandNLA: bounds for $\| \tan \mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k) \|_{2,F}$

Much work on RandNLA has been focused on bounding $\| \tan \mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k) \|_{2,F}$ using matrix concentration inequalities (e.g. matrix Chernoff, matrix Bernstein, matrix Hoeffding inequalities).

## Full rank of $\mathbf{V}_k^\top \mathbf{X}$

It guarantees that $\mathrm{range}\,(\mathbf{V}_\mathrm{k})$ and $\mathrm{range}\,(\mathbf{X})$ are sufficiently close and all principal angles between them are less than $\pi/2$.

# Exact Arithmetic Algorithm

to construct approximations for $U_k$ from $\mathcal{K}_q$

---

**Input:** $A \in \mathbb{R}^{m\times n}$, starting guess $X \in \mathbb{R}^{n\times s}$

Target rank $k < \mathrm{rank}\,(\mathrm{A})$, and assume $\sigma_k > \sigma_{k+1}$

Block dimension $q \geq 1$ with $k \leq (q+1)s \leq m$

**Output:** $\hat{U}_k \in \mathbb{R}^{m\times k}$ with orthonormal columns

---

1: Set $K_q = \begin{pmatrix} AX & (AA^\top)AX & \cdots & (AA^\top)^q AX \end{pmatrix} \in \mathbb{R}^{m\times(q+1)s}$, and assume that $\mathrm{rank}\,(\mathrm{K_q}) = (\mathrm{q}+1)\mathrm{s}$.

2: Run an exact arithmetic Rayleigh-Ritz procedure to find the approximation $U_{W,k}$ of the top $k$ left singular vectors of $W \in \mathbb{R}^{(q+1)s\times k}$ i.e. the projection of $A$ into the orthonormal basis, $U_{K_q}$, of $\mathrm{range}\,(\mathrm{K_q})$.

3: Return $\hat{U}_k = U_{K_q} U_{W,k} \in \mathbb{R}^{m\times k}$.

## Theorem 2

Let $\phi(x)$ be a polynomial of degree $2q+1$ with odd powers only such that $\phi(\mathbf{\Sigma}_k)$ is nonsingular, and $\phi(\sigma_i) \geq \sigma_i$ for $1 \leq i \leq k$. If $\mathrm{rank}\left(\mathbf{V}_k^{\mathrm{T}}\mathbf{X}\right) = \mathrm{k}$,

$$\|\mathbf{A} - \hat{\mathbf{U}}_k\hat{\mathbf{U}}_k^T\mathbf{A}\|_{2,F} \leq \|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^\top\mathbf{A}\|_{2,F} + \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \,\|\tan\mathbf{\Theta}(\mathbf{X},\mathbf{V}_k)\|_F$$

## Gap-amplifying polynomials

A gap-amplifying polynomial satisfies the following three properties:

- ✓ the small values remain small,
- ✓ the large values are amplified, and
- ✓ the large values are growing super-linearly.

We use rescaled Chebyshev-based gap-amplifying polynomials of the form:

$$\phi(x) = \frac{(1+\gamma)\alpha}{\psi_{q'}(1+\gamma)}\psi_{q'}(x/\alpha)$$

where

$$\gamma = \frac{\sigma_k - \sigma_{k+1}}{\sigma_{k+1}}$$

$q' = 2q + 1$, $x \in [0, \alpha]$ and $\psi_{q'}(x)$ is the Chebyshev polynomial of first kind.

Let $\varepsilon > 0$ be an accuracy parameter. If we select

$$q \geq \frac{1}{2\sqrt{\gamma}} \log_2 \frac{4\| \tan \mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\|_2}{\varepsilon}$$

where $\gamma = \frac{\sigma_k - \sigma_{k+1}}{\sigma_{k+1}}$, then the bounds of Theorem 2 become relative:

$$\|\mathbf{A} - \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^T \mathbf{A}\|_{2,F} \leq (1 + \varepsilon)\sigma_{k+1}$$

Remember that:

$$\sigma_{k+1} = \|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k{}^T \mathbf{A}\|_2 \leq \|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k{}^T \mathbf{A}\|_F$$

## Novelty

We combined:

- ✓ traditional Lanczos convergence analysis (Saa11), with
- ✓ optimal low-rank approximations via least squares problems (BDM11; BDM14).

**Theorem 1** We connect principal angles with least squares residuals.

**Theorem 2** We use least squares residuals to interpret orthogonal projections.

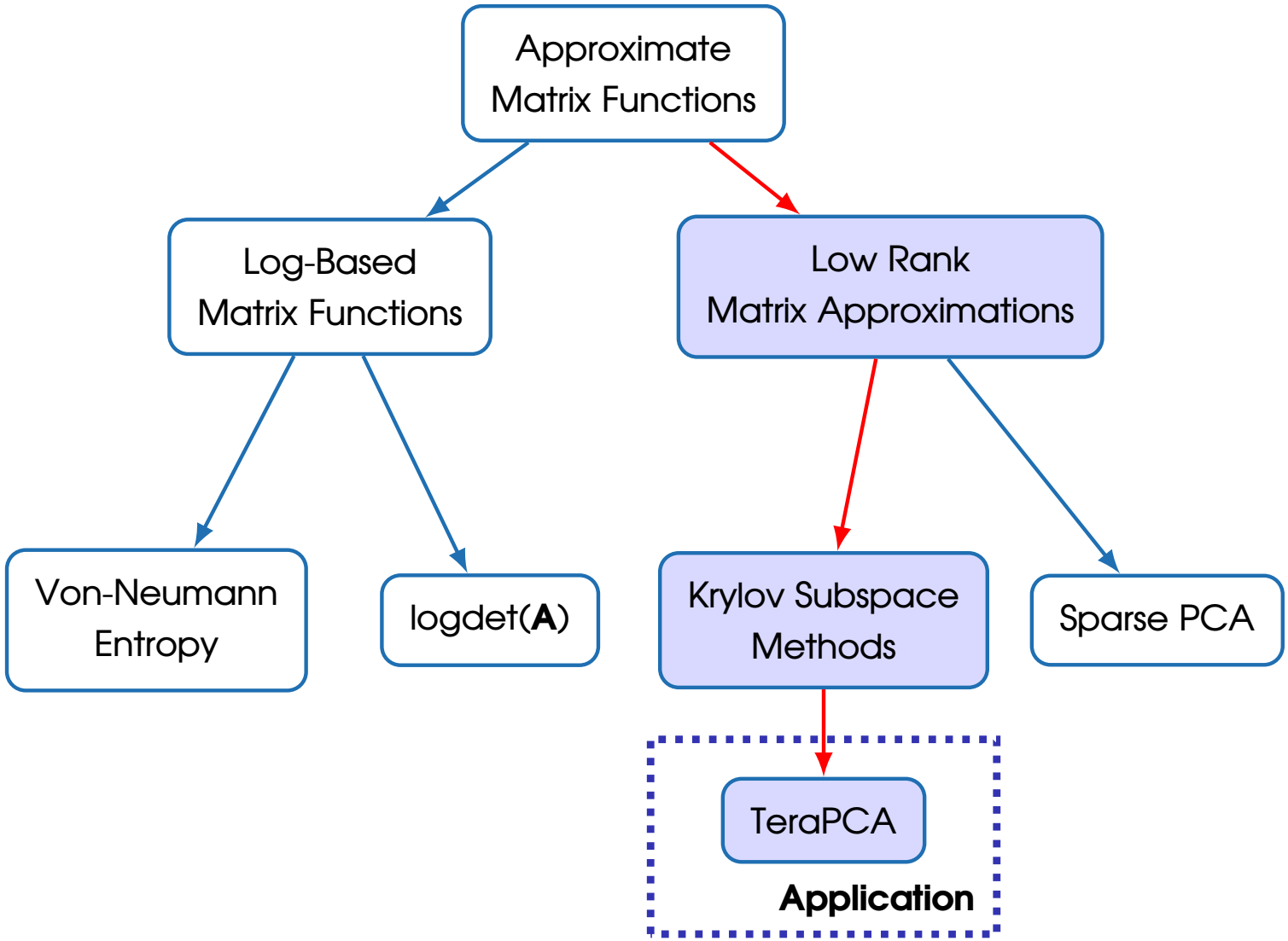
# Open Problems

- Is it possible to drop the assumption that $\mathbf{V}_k^\top \mathbf{X}$ is full-rank?
- Are our bounds tight enough to be informative?
- Can our bounds be useful in implementing block-Lanczos type methods?

# Publications

(Dri+18) P. Drineas, I. Ipsen, E. Kontopoulou, M. Magdon-Ismail, "**Structural Convergence Results for Approximation of Dominant Subspaces from Block Krylov Spaces**, in SIAM Journal on Matrix Analysis and Applications, 39(2):567-586

# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

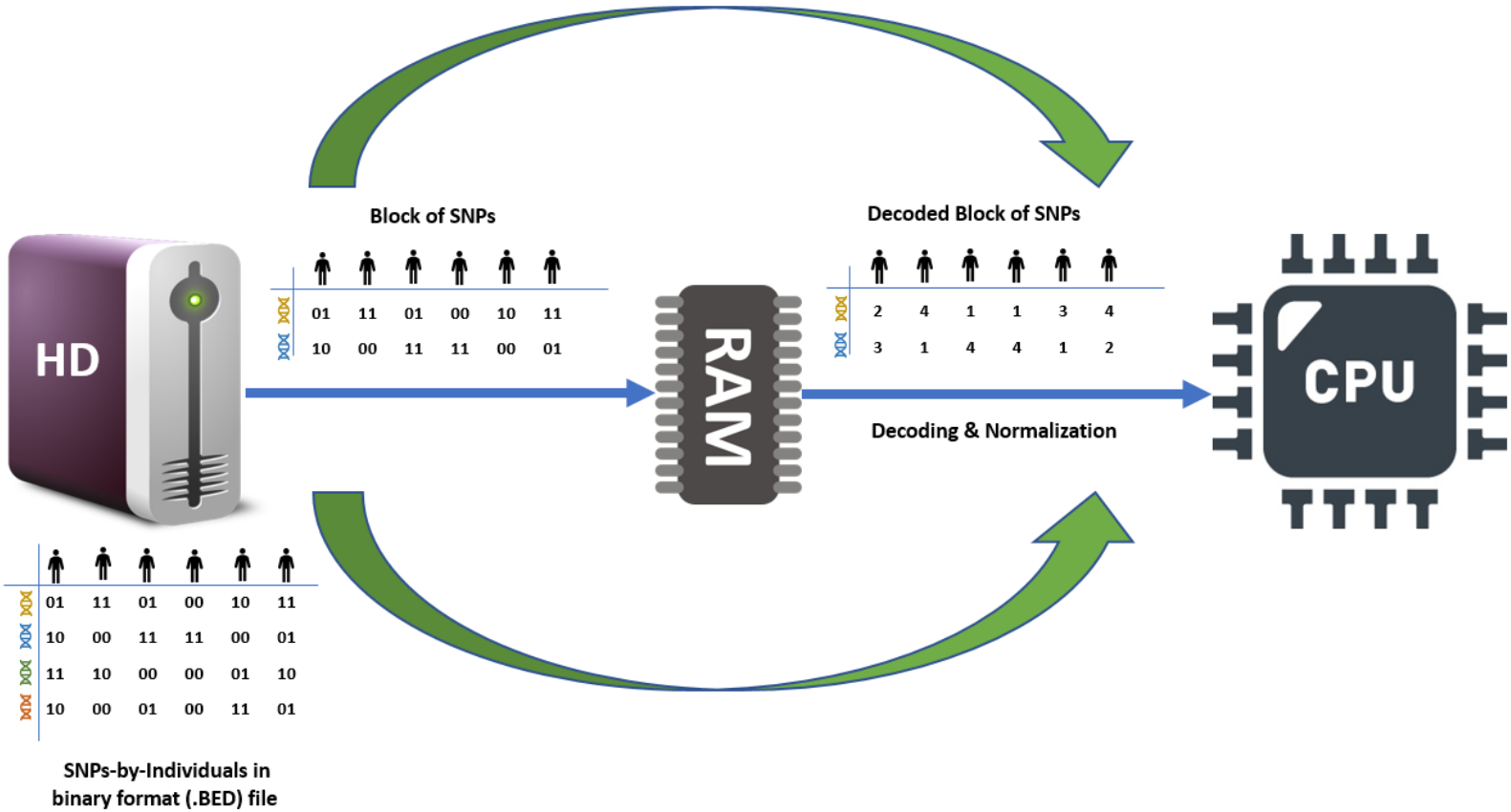
## Motivation

- PCA is a key tool in studying population structure in human genetics.
- Genetic datasets are continuously growing larger in size.
- Need of out-of-core implementations.
- Typical applications in genetics only require a very small number of PCs (e.g., 10) within a small accuracy (e.g., two-three digits).

## What is TeraPCA

- C++ library for out-of-core PCA of large genetic datasets.
- TeraPCA computes the sought PCs by partially solving a symmetric eigenvalue problem.
- This eigenvalue problem is solved using Randomized Subspace Iteration.
- Randomized Subspace Iteration features block iteration thus allowing higher granularity in out-of-core settings.

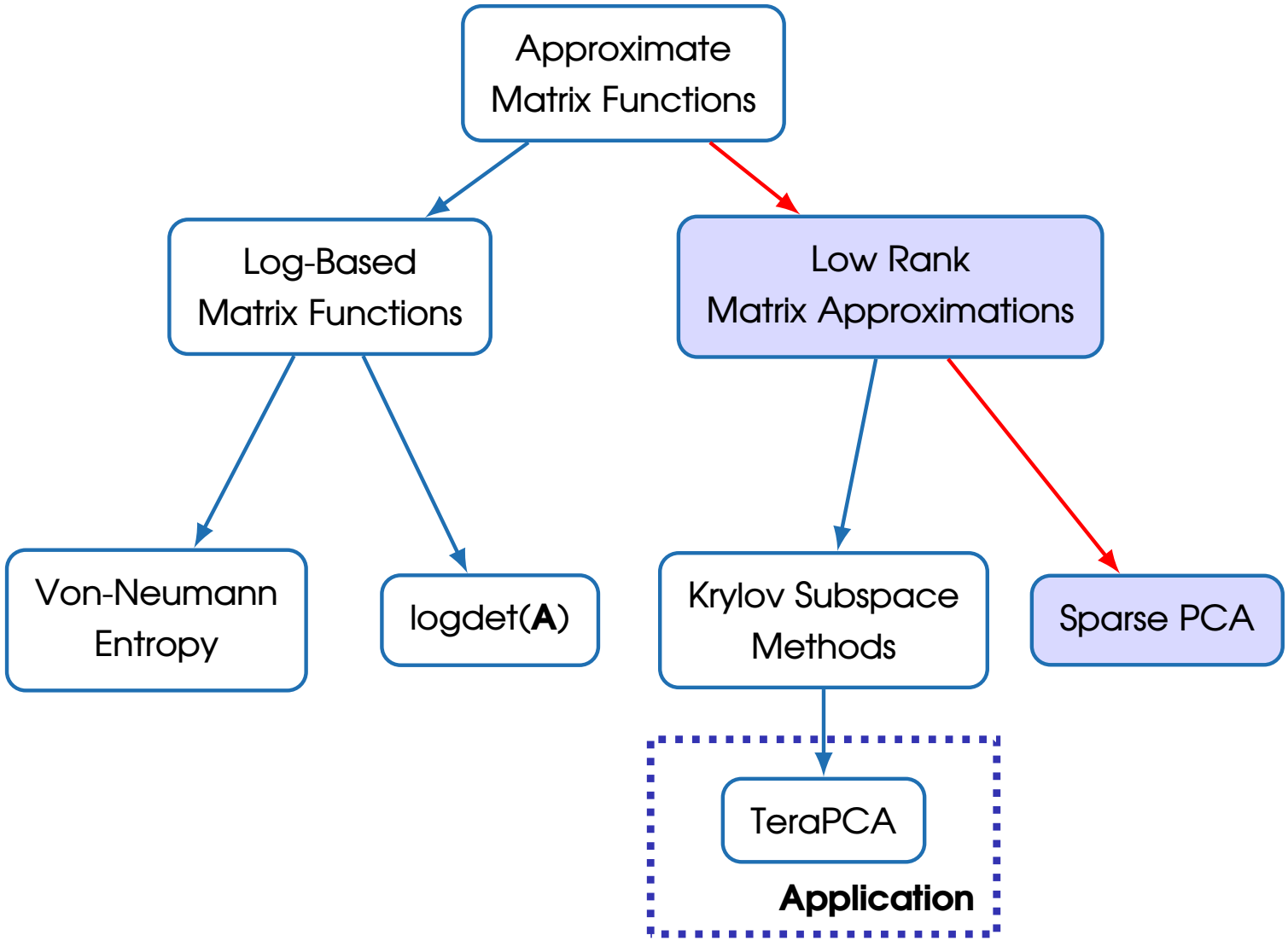
# TeraPCA in a Nutshell



# Publications

(Bos+19) A. Bose, V. Kalantzis, E-M. Kontopoulou, M. Elkady, P. Paschou and P. Drineas, **"ATeraPCA: a Fast and Scalable Software Package to Study Genetic Variation in Tera-scale Genotypes"**, in Oxford Bioinformatics, Vol. 35(19), pp. 3679-3683

# Roadmap


Approximate
Matrix Functions
Log-Based
Matrix Functions
Low Rank
Matrix Approximations
Von-Neumann
Entropy
logdet(**A**)
Krylov Subspace
Methods
Sparse PCA
TeraPCA
**Application**

## Definition

Given a centered matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ and the matrix $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$, we seek to find the vector $\mathbf{w}_{opt}$ that solves:

$$\begin{aligned} \underset{\mathbf{w} \in \mathbb{R}^n}{\text{maximize}} \quad & \mathbf{w}^\top \mathbf{A}\, \mathbf{w} \\ \text{subject to} \quad & \|\mathbf{w}\|_2 = 1 \end{aligned} \tag{2}$$

The objective function of Problem (2) is the Rayleigh Quotient, $\mathbf{R}$, and for PSD matrix like $\mathbf{A}$ the maximum value of $\mathbf{R}$ is the dominant eigenvalue while $\mathbf{w}_{opt}$ is the corresponding eigenvector.

## Definition

Given a centered data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the matrix $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$ and a parameter $k$, we seek to find the vector $\mathbf{w}_{opt}$ that solves:

$$\begin{aligned} \underset{\mathbf{w} \in \mathbb{R}^n}{\text{maximize}} \quad & \mathbf{w}^\top \mathbf{A}\, \mathbf{w} \\ \text{subject to} \quad & \|\mathbf{w}\|_0 \leq k \\ & \|\mathbf{w}\|_2 = 1 \end{aligned} \tag{3}$$

- ✓ $k$ enforces the sparsity of $\mathbf{w}_{opt}$, (at most $k$ non-zero entries).
- ✓ NP-hard if $k$ grows with $n$.
- ✓ Non-convex constraints.
- ✓ **Common approaches:** thresholding the top singular vector, convex relaxations of the constraints, semi-definite programming, . . .

## Definition

Given a centered data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the matrix $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$ and a parameter $k$, we seek to find the vector $\mathbf{w}_{opt}$ that solves:

$$\begin{aligned} \underset{\mathbf{w} \in \mathbb{R}^n}{\text{maximize}} \quad & \mathbf{w}^\top \mathbf{A}\, \mathbf{w} \\ \text{subject to} \quad & \|\mathbf{w}\|_1 \leq \sqrt{k}, \\ & \|\mathbf{w}\|_2 \leq 1 \end{aligned} \tag{4}$$

- ✓ (convex) $l_1$ relaxation of the sparsity constraint.
- ✓ convex relaxation of the 2-norm constraint.

# Algorithm

Two-step algorithm:

1. Compute a stationary point $\tilde{\mathbf{w}}_{opt}$.
2. Invoke a randomized rounding strategy to compute $\hat{\mathbf{w}}_{opt}$.

# Algorithm

Two-step algorithm:

1. Compute a stationary point $\tilde{\mathbf{w}}_{opt}$.
2. Invoke a randomized rounding strategy to compute $\hat{\mathbf{w}}_{opt}$.

## How we find the stationary point - Projected Gradient Ascent

1. Compute the gradient and make a gradient step.
2. Project onto the $l_1$ ball with radius $\sqrt{k}$.
3. Repeat until a relative error threshold is reached.

# Algorithm

Two-step algorithm:

1. Compute a stationary point $\tilde{\mathbf{w}}_{opt}$.
2. Invoke a randomized rounding strategy to compute $\hat{\mathbf{w}}_{opt}$.

---

**Input:** $\mathbf{x} \in \mathbb{R}^n$, integer $s > 0$.
**Output:** $\hat{\mathbf{x}} \in \mathbb{R}^n$ with $\mathbf{E}\left[\|\hat{\mathbf{x}}\|_0\right] \leq s$.

---

1: **for** $i = 1, \ldots, n$ **do**
2: $\quad p_i = \min\left\{\frac{s|\mathbf{x}_i|}{\|\mathbf{x}\|_1}, 1\right\}$
3: $\quad \hat{\mathbf{x}}_i = \begin{cases} \frac{1}{p_i}\mathbf{x}_i, & \text{with probability } p_i. \\ 0, & \text{otherwise.} \end{cases}$
4: **end for**

## Theorem

*Let $\mathbf{w}_{opt}$ be the optimal solution of the Sparse PCA problem (2) satisfying $\|\mathbf{w}_{opt}\|_2 = 1$ and $\|\mathbf{w}_{opt}\|_0 \leq k$. Let $\hat{\mathbf{w}}_{opt}$ be the vector returned when the rounding sparsification strategy is applied on the optimal solution $\tilde{\mathbf{w}}_{opt}$ of the optimization problem (3), with $s = 200k/\varepsilon^2$, where $\varepsilon \in (0, 1]$ is an accuracy parameter. Then, $\hat{\mathbf{w}}_{opt}$ has the following properties:*

1. $\mathbf{E}\left[\|\hat{\mathbf{w}}_{opt}\|_0\right] \leq s$
2. *With probability at least 3/4,*

$$\|\hat{\mathbf{w}}_{opt}\|_2 \leq 1 + 0.15\varepsilon$$

3. *With probability at least 3/4,*

$$\hat{\mathbf{w}}_{opt}^\top \mathbf{A}\hat{\mathbf{w}}_{opt} \geq \mathbf{w}_{opt}^\top \mathbf{A}\mathbf{w}_{opt} - \varepsilon$$

# Publications

(Fou+17) K. Fountoulakis, A. Kundu, E. Kontopoulou, P. Drineas, ``**A Randomized Rounding Algorithm for Sparse PCA**'', in the ACM Transactions on Knowledge Discovery from Data (TKDD), 11(3):38

Thank you!

# Questions?

# References I


(AT11) Haim Avron and Sivan Toledo. ``Randomized Algorithms for Estimating the Trace of an Implicit Symmetric Positive Semi-definite Matrix''. In: *Journal of the ACM* 58.2 (2011), p. 8.

(BDM11) C. Boutsidis, P. Drineas, and M. Magdon-Ismail. ``Near-optimal Column-based Matrix Reconstruction''. In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science--FOCS 2011*. IEEE Computer Soc., Los Alamitos, CA, 2011, pp. 305-314.

(BDM14) C. Boutsidis, P. Drineas, and M. Magdon-Ismail. ``Near-optimal Column-based Matrix Reconstruction''. In: *SIAM J. Comput.* 43.2 (2014), pp. 687-717.

(Bos+19) A. Bose et al. ``TeraPCA: a Fast and Scalable Software Package to Study Genetic Variation in Tera-scale Genotypes''. In: *Bioinformatics* 35.19 (2019), pp. 3679-3683.

(Bou+17) Christos Boutsidis et al. ``A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix''. In: *Linear Algebra and its Applications* 533 (2017), pp. 95-117.

# References II


(dGJ07) A. d'Aspremont, L. El Ghaoui, and M. I. Jordan. ``A Direct Formulation for Sparse PCA using Semidefinite Programming''. In: *SIAM Review* (2007), pp. 434-448.

(Dri+18) P. Drineas et al. ``Structural Convergence Results for Approximation of Dominant Subspaces from Block Krylov Spaces''. In: *SIAM Journal on Matrix Analysis and Applications* 39 (2018), pp. 567-586.

(Fou+17) K. Fountoulakis et al. ``A Randomized Rounding Algorithm for Sparce PCA''. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11.3 (2017), pp. 1-26.

(Hig08) N. Higham. *Functions of Matrices: Theory and Computation*. Vol. 104. SIAM, 2008.

(Kon+18) E. Kontopoulou et al. ``Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices''. In: *2018 IEEE International Symposium on Information Theory*. 2018.

(Kon+20) E. Kontopoulou et al. ``Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices''. In: *IEEE Transactions on Information Theory* to appear (2020).

# References III


(Saa11) Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Revised. Classics in Applied Mathematics. Philadelphia: SIAM, 2011.

(Sjö+12) K. Sjöstrand et al. ``Spasm: A matlab toolbox for sparse statistical modeling''. In: *Journal of Statistical Software (Accepted for publication)*. 2012.

(Tre11) L. Trevisan. *Graph Partitioning and Expanders*. Handout 7. 2011.

# Appendix

# Log-Based Matrix Functions

## Lemma

*Let* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be a symmetric matrix whose eigenvalues all lie in the interval* $(-1, 1)$. *Then,*

$$\log\left[\mathbf{I}_n - \mathbf{A}\right] = -\sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k}$$

Given Lemma we can further manipulate $\mathcal{H}\left[\mathbf{R}\right]$ as:

$$\begin{aligned}
\mathcal{H}\left[\mathbf{R}\right] &= -Tr\left[\mathbf{R}\log\left[\mathbf{R}\right]\right] \\
&= -Tr\left[\mathbf{R}\log\left[uu^{-1}\mathbf{R}\right]\right] \\
&= -Tr\left[\log(u)\mathbf{R}\right] - Tr\left[\mathbf{R}\log\left[\mathbf{I}_n - (\mathbf{I}_n - u^{-1}\mathbf{R})\right]\right] \\
&= \log(u^{-1}) - Tr\left[-\mathbf{R}\sum_{k=1}^{\infty}\frac{(\mathbf{I}_n - u^{-1}\mathbf{R})^k}{k}\right] \\
&= \log(u^{-1}) + \sum_{k=1}^{\infty}\frac{Tr\left[\mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k\right]}{k}
\end{aligned}$$

Back to Main Slides

# Analysis of the Power Method

Boutsidis et al., LAA 2017 (Bou+17)

In (Bou+17) appears the following lemma that builds on (Tre11) and guarantees a relative error approximation to the dominant eigenvalue:

## Lemma

*Let $\tilde{p}_1$ be the output of the Power Method algorithm with $q = \lceil 4.82 \log(1/\delta) \rceil$ and $t = \left\lceil \log \sqrt{4n} \right\rceil$. Then, with probability at least $1 - \delta$,*

$$\frac{1}{6} p_1 \leq \tilde{p}_1 \leq p_1$$

**Input:** SPD $A \in \mathbb{R}^{n \times n}$, failure probability $\delta < 1$ and integers $q = \lceil 4.82 \log(1/\delta) \rceil$ and $t = \lceil \log \sqrt{4n} \rceil$.

**Output:** $\widehat{\lambda_{\max}(A)}$, the estimate of $\lambda_{\max}(A)$.

1: **for** $i = 1, \ldots, q$ **do**
2: Create uniformly at random a Rademacher vector $x_0^{(i)} \in \mathbb{R}^n$.
3: **for** $k = 1, \ldots, t$ **do**
4: $x_k^{(i)} = A \cdot x_{k-1}^{(i)}$
5: **end for**
6: Compute $\widehat{\lambda_{\max}(A)}^{(i)}$ as:

$$\widehat{\lambda_{\max}(A)}^{(i)} = \frac{{x_t^{(i)}}^\top A \, x_t^{(i)}}{{x_t^{(i)}}^\top x_t^{(i)}}$$

7: **end for**
8: **return** $\widehat{\lambda_{\max}(A)}$ as:

$$\widehat{\lambda_{\max}(A)} = \max_{i=1,\ldots,q} (\widehat{\lambda_{\max}(A)}^{(i)}).$$

Back to Main Slides

# Trace Estimators

Avron & Toledo 2011 (AT11)

## Definition

A Gaussian trace estimator for a symmetric positive-definite matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is

$$\mathbf{G} = \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^\top \mathbf{A}\, \mathbf{g}_i,$$

where the $\mathbf{g}_i$'s are $p$ independent random vectors whose entries are i.i.d. standard normal variables.

## Lemma

*Let* $\mathbf{A}$ *be an SPD matrix in* $\mathbb{R}^{n \times n}$*, let* $0 < \varepsilon < 1$ *be an accuracy parameter, and let* $0 < \delta < 1$ *be a failure probability. Then for* $s = \lceil 20 \log(2/\delta)\varepsilon^{-2} \rceil$*, with probability at least* $1 - \delta$*,*

$$|Tr\,[\mathbf{A}] - \mathbf{G}| \leq \varepsilon \cdot Tr\,[\mathbf{A}]$$

**Input:** SPD $A \in \mathbb{R}^{n \times n}$, accuracy parameter $\varepsilon < 1$ and failure probability $\delta < 1$.

**Output:** $\widehat{Tr\left[A\right]}$, the estimate of $Tr\left[A\right]$.

1: Generate $s = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.

2: Compute $\widehat{Tr\left[A\right]}$ as:

$$\widehat{Tr\left[A\right]} = \frac{1}{s} \sum_{i=1}^{s} g_i^\top \; A \; g_i$$

Back to Main Slides

# Bounding the Absolute Error I

## Taylor-based Algorithm

We manipulate $\Delta = \left| \widehat{\mathcal{H}\left[\mathbf{R}\right]} - \mathcal{H}\left[\mathbf{R}\right] \right|$ as follows:

$$
\begin{aligned}
\Delta &= \left| \sum_{k=1}^{m} \frac{1}{k} \cdot \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^{\top} \mathbf{R}\mathbf{C}^k \mathbf{g}_i - \sum_{k=1}^{\infty} \frac{1}{k} Tr\left[\mathbf{R}\mathbf{C}^k\right] \right| \\
&\leq \left| \sum_{k=1}^{m} \frac{1}{k} \cdot \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^{\top} \mathbf{R}\mathbf{C}^k \mathbf{g}_i - \sum_{k=1}^{m} \frac{1}{k} Tr\left[\mathbf{R}\mathbf{C}^k\right] \right| + \left| \sum_{k=m+1}^{\infty} \frac{1}{k} Tr\left[\mathbf{R}\mathbf{C}^k\right] \right| \\
&= \underbrace{\left| \frac{1}{s} \sum_{i=1}^{s} \mathbf{g}_i^{\top} \left( \sum_{k=1}^{m} \mathbf{R}\mathbf{C}^k / k \right) \mathbf{g}_i - Tr\left[ \sum_{k=1}^{m} \frac{1}{k} \mathbf{R}\mathbf{C}^k \right] \right|}_{\Delta_1} + \underbrace{\left| \sum_{k=m+1}^{\infty} Tr\left[\mathbf{R}\mathbf{C}^k\right] / k \right|}_{\Delta_2}
\end{aligned}
$$

After algebra we conclude:

$$\Delta_1 \leq \epsilon \cdot Tr\left[\sum_{k=1}^{\infty} \mathbf{R}\mathbf{C}^k/k\right],$$

and

$$\Delta_2 \leq \left(1-\frac{\ell}{u}\right)^m \sum_{k=1}^{\infty} Tr\left[\mathbf{R}\mathbf{C}^k\right]/k$$

Combining the two bounds we get:

$$\begin{aligned}
\left|\widehat{\mathcal{H}[\mathbf{R}]} - \mathcal{H}[\mathbf{R}]\right| &\leq \left(\epsilon + \left(1-\frac{\ell}{u}\right)^m\right) \sum_{k=1}^{\infty} \frac{Tr\left[\mathbf{R}\mathbf{C}^k\right]}{k} \\
&\leq \left(\epsilon + \left(1-\frac{\ell}{u}\right)^m\right) \left(\mathcal{H}[\mathbf{R}] - \log u^{-1}\right) \\
&\leq \left(\epsilon + \left(1-\frac{\ell}{u}\right)^m\right) \mathcal{H}[\mathbf{R}] \\
&\leq 2\epsilon\mathcal{H}[\mathbf{R}]
\end{aligned}$$

Back to Main Slides

## The Clenshaw Algorithm

The Clenshaw algorithm is a recursive procedure that evaluates fast Chebyshev polynomials:

---

**Input:** Coefficients $\alpha_i$, $i = 0, \ldots, m$, matrix $R \in \mathbb{R}^{n \times n}$ and vectors $g \in \mathbb{R}^n$

---

1: Set $y_{m+2} = y_{m+1} = 0$
2: **for** $k = m, m-1, \ldots, 0$ **do**
3: $\quad y_k = \alpha_k g + \frac{4}{u} R y_{k+1} - 2y_{k+1} - y_{k+2}$
4: **end for**

**Output:** $g^\top f_m(R) g = \frac{1}{2}\left(\alpha_0 (g^\top g) + g^\top (y_0 - y_2)\right)$

---

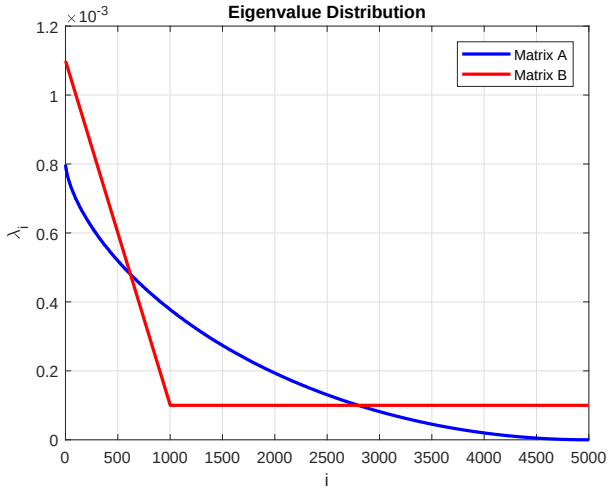
Back to Main Slides

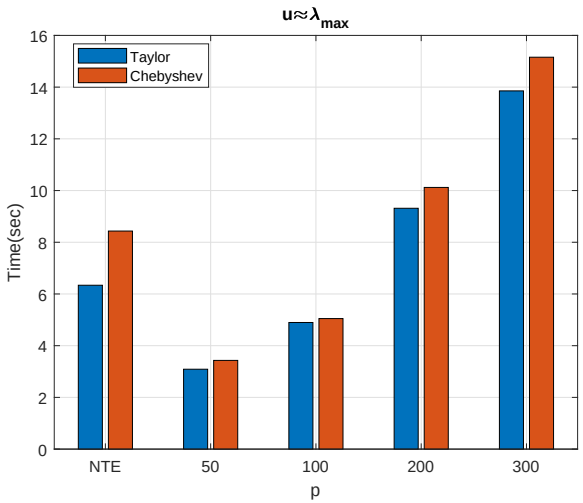
# Experiment 1

## Running Time

Random density matrices of size $5,000 \times 5,000$

- ✓ **Matrix A:** exponentially decaying probabilities.
- ✓ **Matrix B:** 1,000 linearly decaying probabilities.




## Parameters

- ✓ Polynomial terms: $m = [5 : 5 : 30]$
- ✓ Gaussian vectors: $s = \{50, 100, 200, 300\}$
- ✓ Largest probability: $u \approx \lambda_{max}$

## Notes

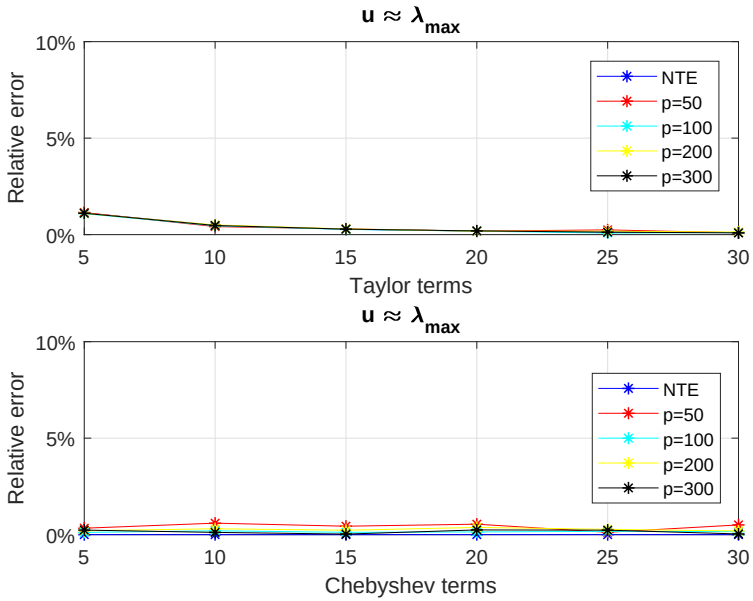
- Exact computation: 1.5 minutes.
- Approximation of $\lambda_{max}$: $<$ 1 second.

# Experiment 1

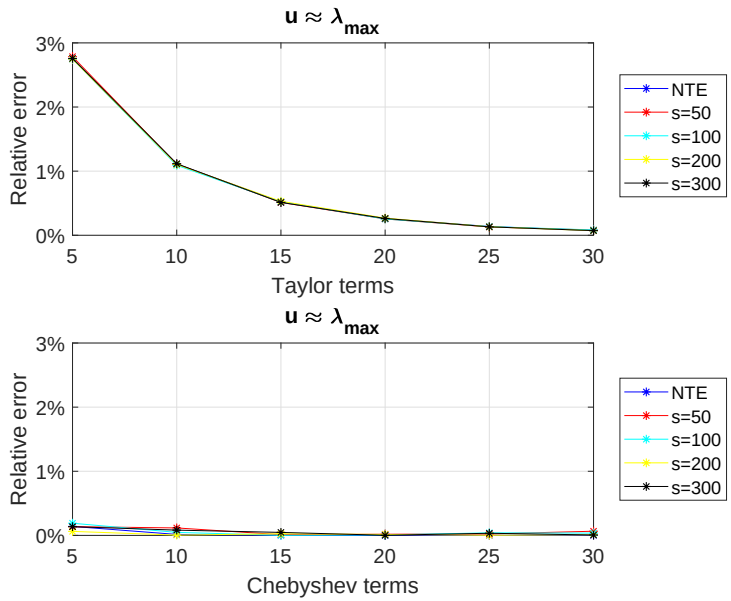
## Relative Error

## Parameters

- ✓ Polynomial terms: $m = [5 : 5 : 30]$
- ✓ Gaussian vectors: $s = \{50, 100, 200, 300\}$
- ✓ Largest probability: $u \approx \lambda_{max}$




Matrix A

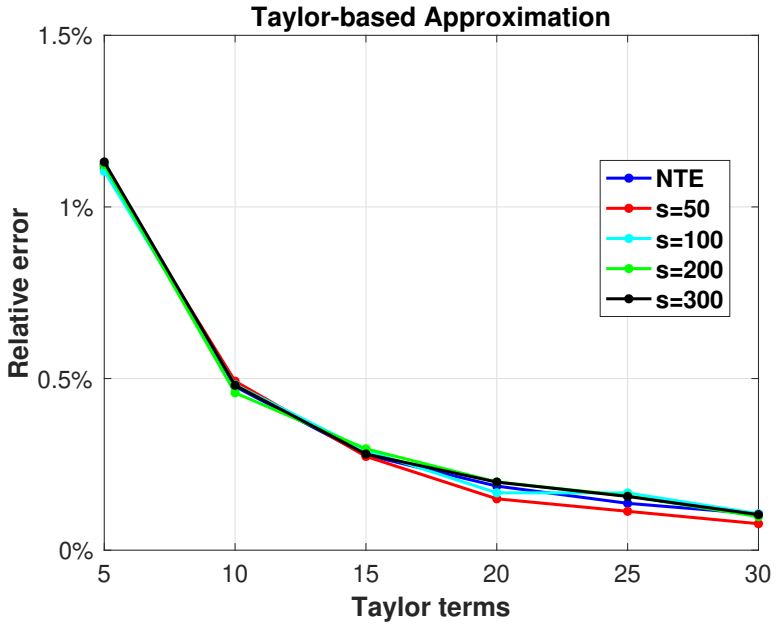


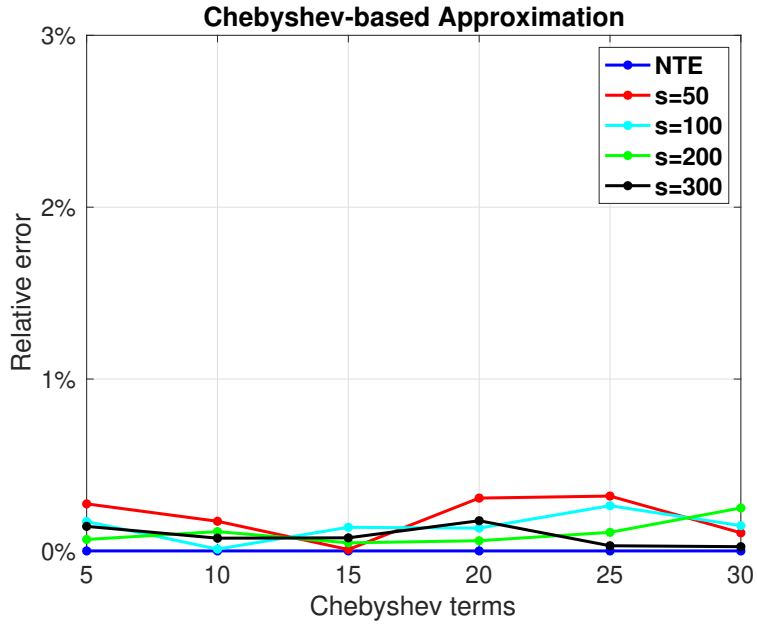

Matrix B

# Experiment 2

Random **complex** density matrix of size $5,000 \times 5,000$

- ✓ Polynomial terms: $m = [5 : 5 : 20]$
- ✓ Gaussian vectors: $s = \{50, 100, 200, 300\}$

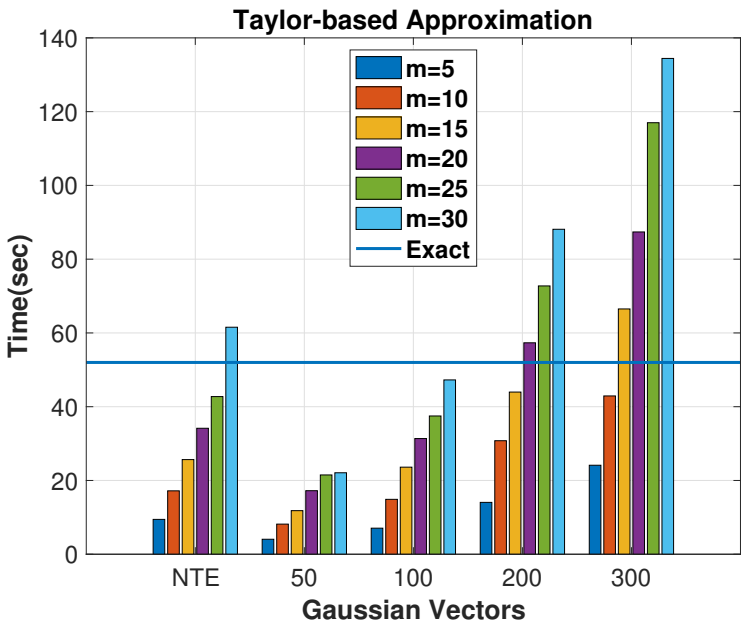


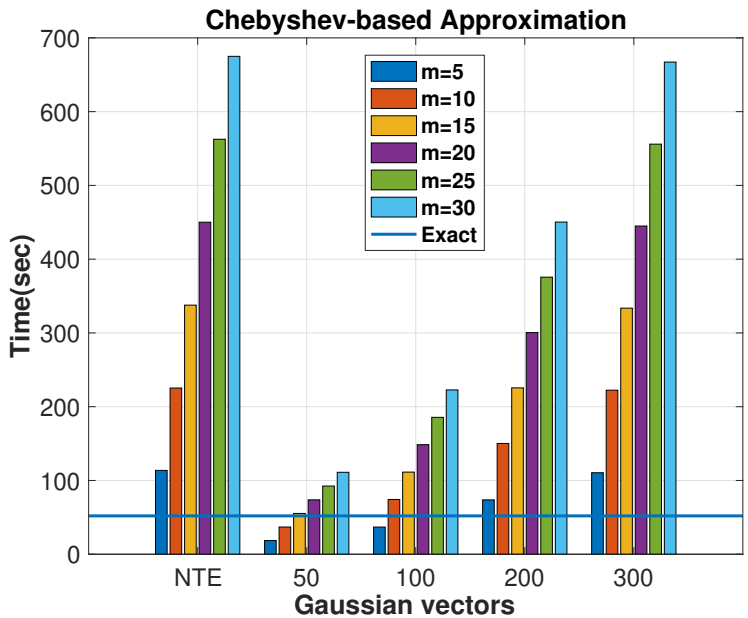



## Experiment 2 cntn'd

Random **complex** density matrix of size $5,000 \times 5,000$

- ✓ Polynomial terms: $m = [5 : 5 : 20]$
- ✓ Gaussian vectors: $s = \{50, 100, 200, 300\}$




## Notes

- Exact computation: 52 seconds.

---

Back to Main Slides

# Mathematical Manipulation of logdet $[\mathbf{A}]$

$$\begin{aligned}
\text{logdet}\left[\mathbf{A}\right] &= \text{logdet}\left[\mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^{\top}\right] \\
&= \log\left(\det\left[\mathbf{\Lambda}\right]\right) \\
&= \log\left(\prod_{i=1}^{n}\lambda_i\right) \\
&= \sum_{i=1}^{n}\log(\lambda_i) \\
&= Tr\left[\log\left[\mathbf{A}\right]\right]
\end{aligned}$$

$$\begin{aligned}
Tr\left[\log\left[\mathbf{A}\right]\right] &= Tr\left[\log\left[\mathbf{I}_n - \mathbf{I}_n + \mathbf{A}\right]\right] \\
&= Tr\left[\log\left[\mathbf{I}_n - \underbrace{\left(\mathbf{I}_n - \mathbf{A}\right)}_{\mathbf{C}}\right]\right] \\
&= Tr\left[\log\left[\mathbf{I}_n - \mathbf{C}\right]\right] \\
&= Tr\left[-\sum_{k=1}^{\infty}\frac{\mathbf{C}^k}{k}\right] \\
&= -\sum_{k=1}^{\infty}\frac{Tr\left[\mathbf{C}^k\right]}{k}
\end{aligned}$$

Back to Main Slides

**Input:** $A \in \mathbb{R}^{n \times n}$, accuracy parameter $\varepsilon > 0$, integer $m > 0$.
**Output:** $\widehat{logdet}\,[A]$, the approximation to the $\log\det[A]$.

1: Compute $\tilde{\lambda}_1(A)$, the estimation of the largest eigenvalue of $A$, using the power method.
2: Set $u = 7\tilde{\lambda}_1(A)$
3: $C = I_n - u^{-1}A$
4: Generate $s = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.
5: Compute $\widehat{logdet}\,[A]$ as:

$$\widehat{logdet}\,[A] = n\log(u) - \sum_{k=1}^{m} \frac{1}{k} \left( \frac{1}{s} \sum_{i=1}^{s} g_i^\top C^k g_i \right)$$

Back to Main Slides

**Input:** $A \in \mathbb{R}^{n \times n}$ with eigenvalues lie in $(\theta_1, 1)$ where $\theta_1 > 0$, accuracy parameter $\varepsilon > 0$, integer $m > 0$.

**Output:** $\widehat{logdet}\,[A]$, the approximation to $\log\det[A]$.

1: $C = I_n - A$

2: Create $s = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.

3: Generate $\widehat{logdet}\,[A]$ as:

$$\widehat{logdet}\,[A] = \sum_{k=1}^{m} \frac{1}{k} \left( \frac{1}{s} \sum_{i=1}^{s} g_i^\top \, C^k \, g_i \right)$$

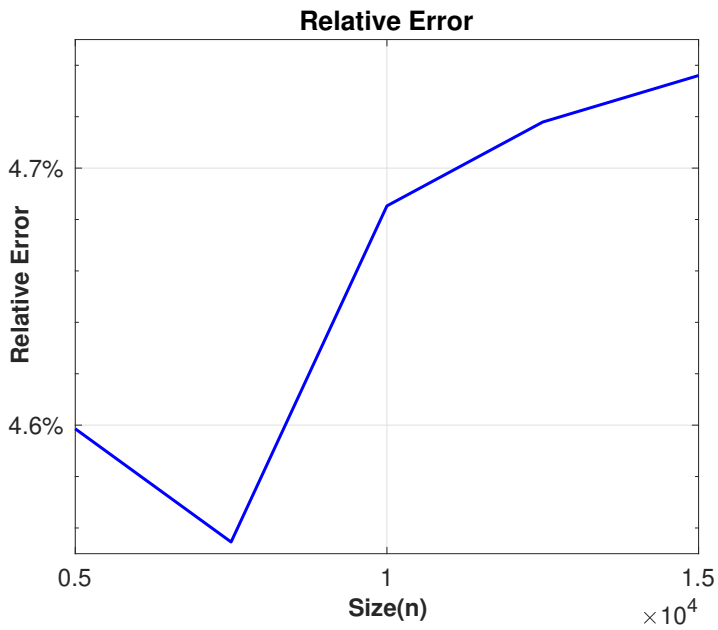
Back to Main Slides

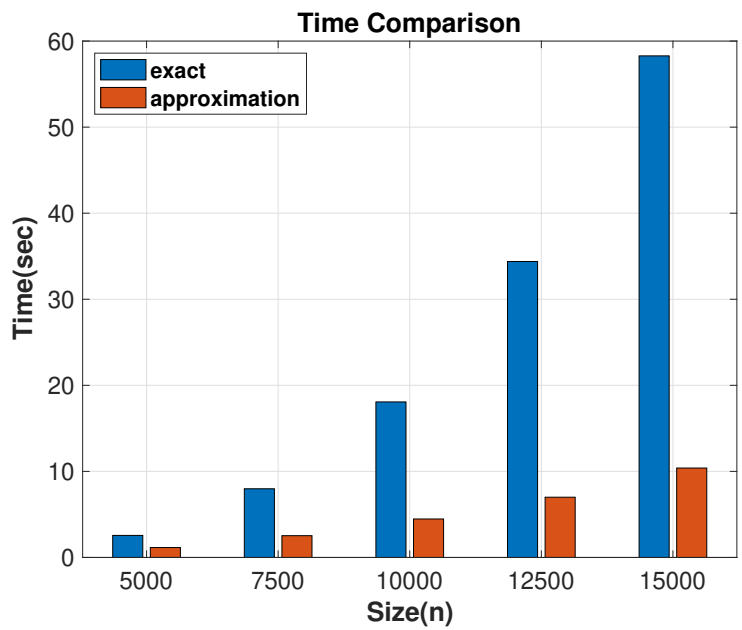
# Experiments

## Dense Random Matrices

### Parameters

- ✓ Polynomial terms: $m = 4$.
- ✓ Gaussian vectors: $s = 60$.



# Experiments

## Real Sparse Matrices
## University of Florida Sparse Matrix Collection

## Parameters

- ✓ Polynomial terms: $m = 1 : 5 : 150$.
- ✓ Gaussian vectors: $s = 5$.

| matrix name | n | nnz | logdet [A] | | | time (sec) | | m |
|---|---|---|---|---|---|---|---|---|
| | | | exact | approx | | exact | approx | |
| | | | | mean | std | | mean | |
| thermal2 | 1228045 | 8580313 | 1.3869e6 | 1.3928e6 | 964.79 | 31.28 | 31.24 | 149 |
| ecology2 | 999999 | 4995991 | 3.3943e6 | 3.403e6 | 1212.8 | 18.5 | 10.47 | 125 |
| ldoor | 952203 | 42493817 | 1.4429e7 | 1.4445e7 | 1683.5 | 117.91 | 17.60 | 33 |
| thermomech_TC | 102158 | 711558 | -546787 | -546829.4 | 553.12 | 57.84 | 2.58 | 77 |
| boneS01 | 127224 | 5516602 | 1.1093e6 | 1.106e6 | 247.14 | 130.4 | 8.48 | 125 |

Back to Main Slides

# TeraPCA

# Randomized Subspace Iteration

**Input:** $A^\top \in \mathbb{R}^{n \times m}$, initial guess matrix $X_0 \in \mathbb{R}^{m \times s}$ with elements drawn i.i.d. from the normal distribution $\mathcal{N}(0, 1)$, $k \geq 1$, and $s \geq k$.

**Output:** The $k$ leading approximate left singular vectors of $A$.

1: $C = A(A^\top X_0)$
2: **repeat**
3: $\quad Q = \texttt{orth}(C)$
4: $\quad C = AA^\top Q$
5: $\quad M = Q^\top C$
6: $\quad$ Compute the eigenvalue decomposition $M = XDX^\top$
7: $\quad C = QX$
8: **until** convergence
9: **return** first $k$ columns of $Q$

# Out-of-core MMV $\mathbf{C} = \mathbf{A}(\mathbf{A}^\top \mathbf{X})$

**Input:** $\zeta > 0, X \in \mathbb{R}^{m \times s}$.
**Output:** $C \in \mathbb{R}^{m \times s}$.

1: $C = \mathbf{0}$
2: **for** $i = 1 : \zeta$ **do**
3: Fetch the $i$-th row-block of $A^\top$
4: $C = C + A_i(A_i^\top X)$
5: **end for**

Back to Main Slides

## Datasets & Experimental Setup

- Approximate the top 10 PCs.
- Initial subspace size $s = 20$.
- All our experiments ran at Purdue's Brown cluster on a dedicated node which features an Intel Xeon Gold 6126 @ 2.6 GHz processor, 96 GB RAM and 64-bit CentOS Linux 7 operating system.

| Dataset | Size (.PED file) | Size (.BED file) | # Samples | # SNPs |
|---|---|---|---|---|
| $S_1$ (simulated) | 19 GB | 120 MB | 5,000 | 1,000,000 |
| $S_2$ (simulated) | 38 GB | 239 MB | 10,000 | 1,000,000 |
| $S_3$ (simulated) | 373 GB | 24 GB | 100,000 | 1,000,000 |
| $S_4$ (simulated) | 1.9 TB | 117 GB | 500,000 | 1,000,000 |
| $S_5$ (simulated) | 3.7 TB | 233 GB | 1,000,000 | 1,000,000 |
| $S_6$ (simulated) | 38 GB | 2.4 GB | 100,000 | 100,000 |
| $S_7$ (simulated) | 150 GB | 9.4 GB | 2,000 | 20,000,000 |
| HGDP | 615 MB | 39 MB | 1,043 | 154,417 |
| 1000 Genomes | 8.4 GB | 483 MB | 2,504 | 808,704 |
| PRK | 2 GB | 126 MB | 4,706 | 111,831 |
| T2D | 1.8 GB | 111 MB | 6,370 | 72,457 |

# Time Comparisons

## Comparison with FlashPCA2

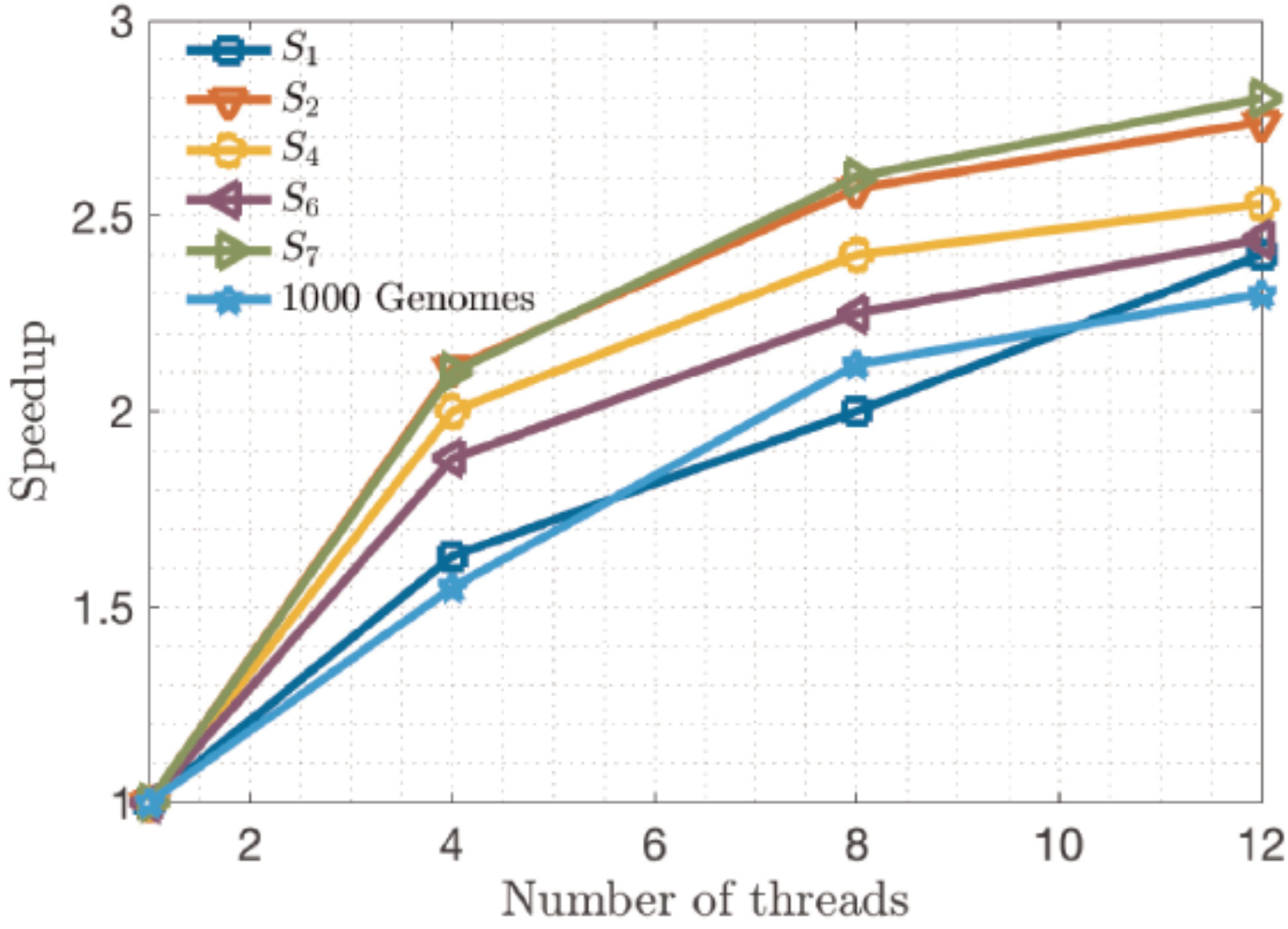
* indicates no convergence after 50 hrs.
Max RAM size allowed: 2GB

| Dataset | TeraPCA | FlashPCA2 | Speed-up |
|---|---|---|---|
| $S_1$ | 26.2 mins | 33.3 mins | 1.27 |
| $S_2$ | 39.3 mins | 87.5 mins | 2.22 |
| $S_3$ | 7.9 hrs | 35.6 hrs | 4.50 |
| $S_4$ | 7.3 hrs | n/a* | $\infty$ |
| $S_5$ | 13.2 hrs | n/a* | $\infty$ |
| $S_6$ | 39.5 mins | 141.1 mins | 3.57 |
| $S_7$ | 37.3 mins | 106.5 mins | 2.86 |
| HGDP | 6.5 secs | 7.7 secs | 1.22 |
| 1000 Genomes | 4.3 mins | 3.5 mins | 0.81 |
| T2D | 96 secs | 119 secs | 1.24 |
| PRK | 76 secs | 73 secs | 0.96 |

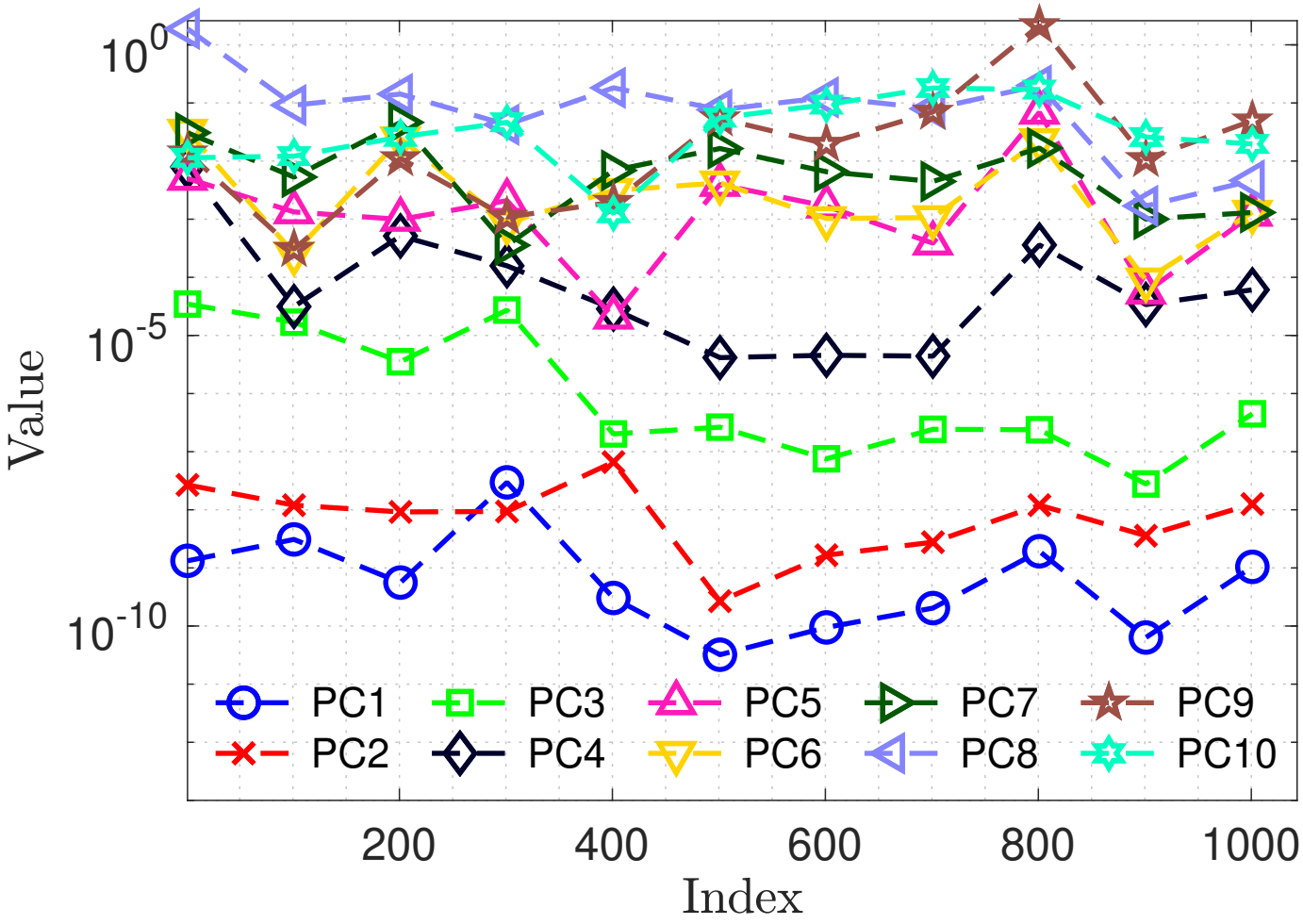
TeraPCA has an advantage over FlashPCA2 (which is based on Implicit Restarted Arnoldi) due to its block nature which allows to:

- search for multiple PCs simultaneously
- perform more computations per epoch
- take advantage of state-of-the-art dense linear algebra kernels (e.g., BLAS, LAPACK)

# Speedup using Multi-threading



# Accuracy of Leading PCs




Element-wise relative error of the 10 leading PCs computed by TeraPCA versus those computed by LAPACK for the HGDP dataset.

# Accuracy of Leading Eigenvalues

Accuracy of the 10 leading eigenvalues computed for TeraPCA and FlashPCA2.

| eigenvalue index | relative error | | eigenvalue index | relative error | |
|---|---|---|---|---|---|
| | TeraPCA | FlashPCA2 | | TeraPCA | FlashPCA2 |
| 1 | 9.91E-15 | 1.74E-03 | 6 | 3.01E-06 | 7.63E-04 |
| 2 | 1.02E-13 | 1.30E-03 | 7 | 3.36E-06 | 1.47E-03 |
| 3 | 5.65E-11 | 1.49E-03 | 8 | 1.04E-05 | 6.81E-04 |
| 4 | 2.18E-08 | 1.31E-03 | 9 | 7.11E-05 | 1.28E-03 |
| 5 | 2.65E-06 | 1.10E-03 | 10 | 1.74E-04 | 7.44E-04 |

Back to Main Slides

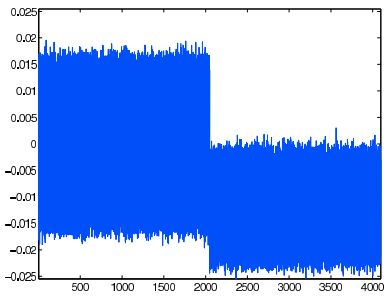
# Sparse PCA

# Experiments

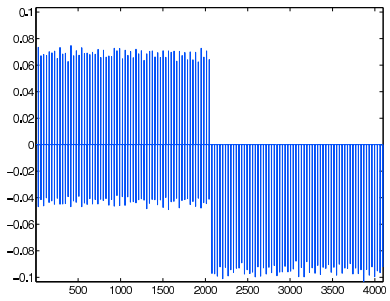
## Synthetic dataset

We test our algorithm (Naive & SVD-based) with other SPCA software like `MaxComp` (Naive & SVD-based) and `Spasm` (Sjö+12).

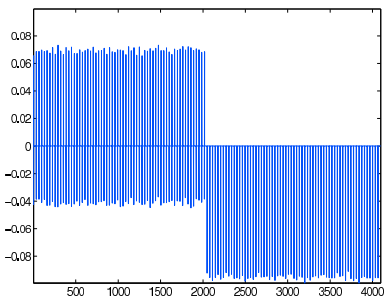
## Pattern capture



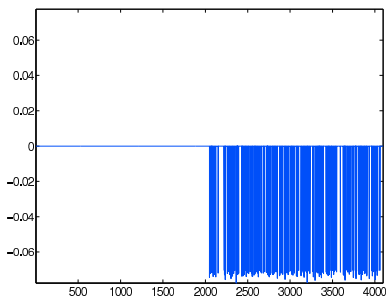
(a) Actual eigenvector



(b) `rspca`

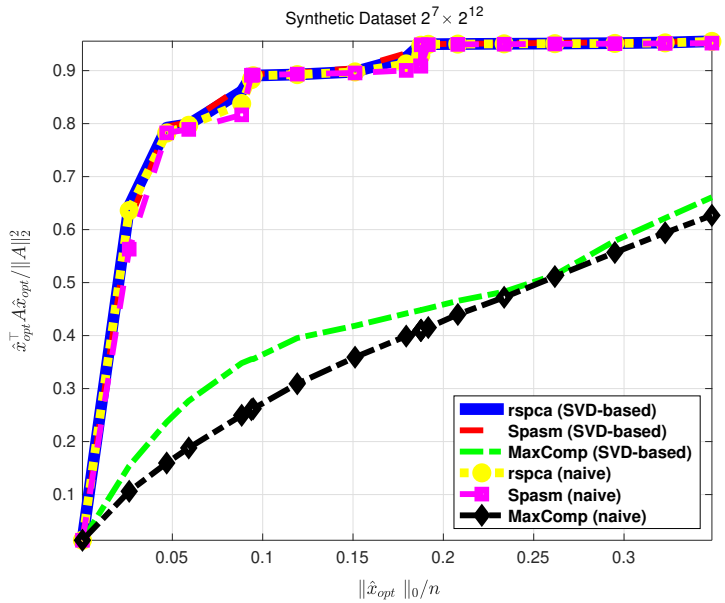


(c) `Spasm`



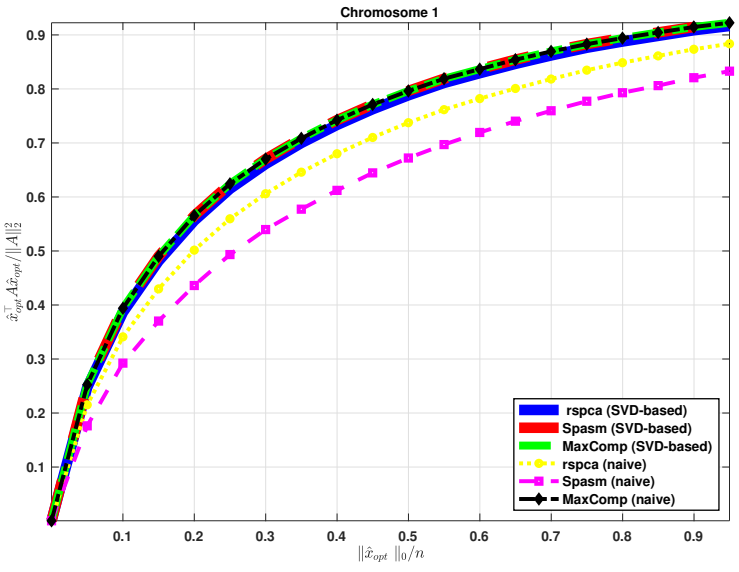
(d) `MaxComp`

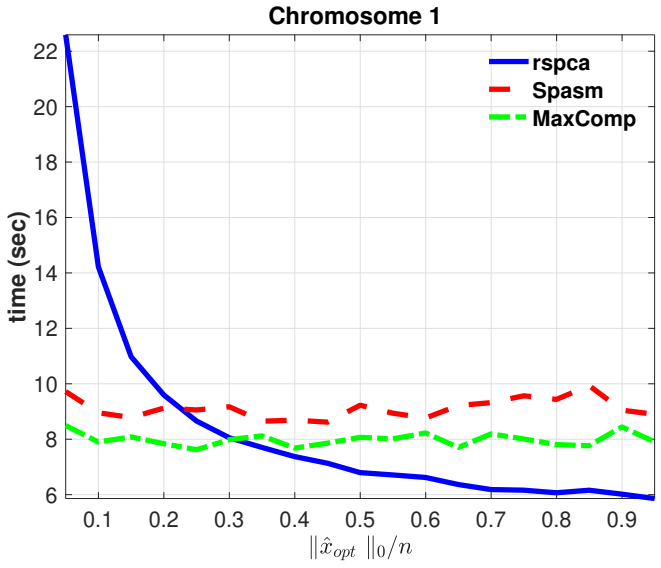
## Sparsity ratio vs Variance capture



# Experiments

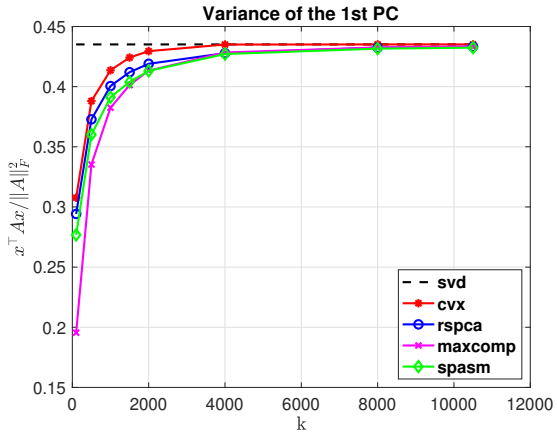
HGDP Chromosome 1: $m = 2,500$ samples, $n = 37,493$ SNPs

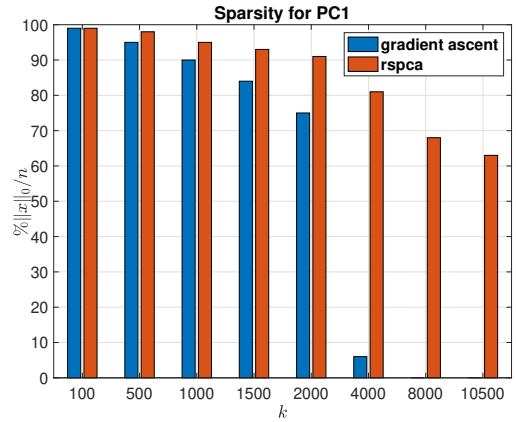


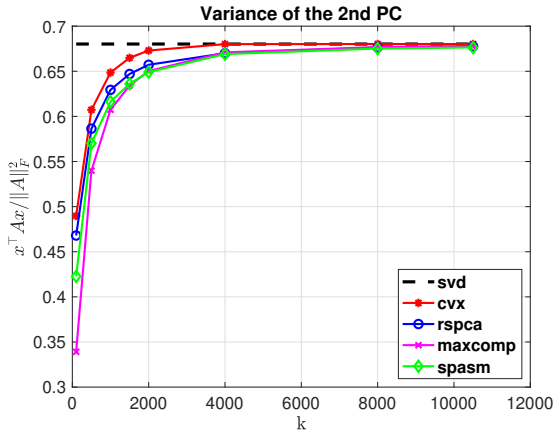



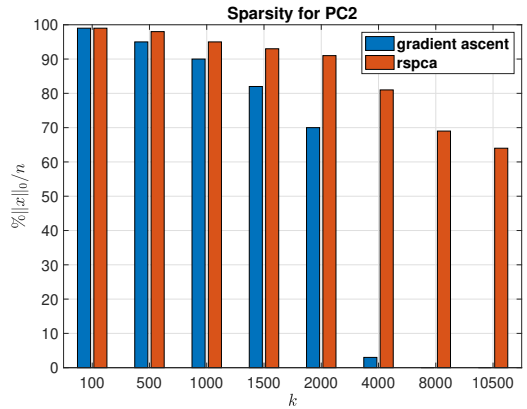
# Experiments

Classic-2: $m = 2,858$ documents , $n = 12,427$ terms




Use of deflation for PC2. Complicated to guarantee orthogonality.

---

Back to Main Slides