

Finding Structure with Randomness

Probabilistic Algorithms for Approximate Matrix Decompositions
“Sparsity and Compressed Sensing” course

Lilian Besson

École Normale Supérieure de Cachan (Master MVA)

EPFL, Lausanne, February 08, 2016

Everything (slides, report, programs) is on <http://lbo.k.vu/pcs2016>.
If needed: [lilian.besson\[at\]ens-cachan.fr](mailto:lilian.besson@ens-cachan.fr).

Problematic

“Low rank” approximation: find \mathbf{A} 's rank k , and \mathbf{B} , \mathbf{C} such that:

$$\underbrace{\mathbf{A}}_{m \times n} \approx \underbrace{\mathbf{B}}_{m \times k} \times \underbrace{\mathbf{C}}_{k \times n} \quad (1)$$

Problematic

“Low rank” approximation: find \mathbf{A} 's rank k , and \mathbf{B} , \mathbf{C} such that:

$$\underbrace{\mathbf{A}}_{m \times n} \approx \underbrace{\mathbf{B}}_{m \times k} \times \underbrace{\mathbf{C}}_{k \times n} \quad (1)$$

Issue to solve

Traditional matrix decomposition algorithms, (**QR**, **SVD**), **can be not adapted** to inaccurate matrices, or large matrices with low rank.

Complexity is in $O(m \times n \times \min(m, n))$ usually.

Problematic

“Low rank” approximation: find \mathbf{A} 's rank k , and \mathbf{B} , \mathbf{C} such that:

$$\underbrace{\mathbf{A}}_{m \times n} \approx \underbrace{\mathbf{B}}_{m \times k} \times \underbrace{\mathbf{C}}_{k \times n} \quad (1)$$

Issue to solve

Traditional matrix decomposition algorithms, (**QR**, **SVD**), **can be not adapted** to inaccurate matrices, or large matrices with low rank.

Complexity is in $O(m \times n \times \min(m, n))$ usually.

\implies need for a framework to solve these kinds of problems.

Hope: complexity in $O(m \times n \times k)$ or $O(m \times n \times \log(k))$.

Overview of the presentation

- 1 Presentation
- 2 Algorithms
- 3 Theoretical analysis for error bounds
- 4 Numerical experiments
- 5 Conclusion

Two stages approximation framework

Input: Matrix $A \in \mathbb{M}_{m,n}(\mathbb{K})$, of **unknown** rank k .

“Two stages” approximation framework

Stage 1: Compute an orthonormal low-rank basis Q for A :

$$A \approx QQ^*A,$$

Stage 2: Compute the matrix factorization^a on the reduced matrix $B \stackrel{\text{def}}{=} Q^*A$.

Output: Rank k and basis Q , and decomposition, e.g. $A \approx U\Sigma V^*$.

^aExample: SVD, QR, etc.

Two stages approximation framework

“Two stages” approximation framework

Stage 1: Compute an orthonormal low-rank basis \mathbf{Q} for \mathbf{A} :

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^* \mathbf{A},$$

Stage 2: Compute the matrix factorization^a on the reduced matrix $\mathbf{B} \stackrel{\text{def}}{=} \mathbf{Q}^* \mathbf{A}$.

Output: Rank k and basis \mathbf{Q} , and decomposition, e.g. $\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.

^aExample: SVD, QR, etc.

Adding **randomization in Stage 1** will permit to span the range of \mathbf{A} more efficiently.

Goal: from $\min(m, n)$ to k or $\log(k)$ in the algorithm complexity.

Two similar problems

The fixed precision problem

Given \mathbf{A} and a **tolerance** $\varepsilon > 0$, find \mathbf{Q} such that:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \varepsilon.$$

The fixed rank problem

Given \mathbf{A} and **rank** $k \in \mathbb{N}$, find \mathbf{B} such that:

$$\|\mathbf{A} - \mathbf{B}\| \approx \min_{\text{rank}(\mathbf{X}) \leq k} \|\mathbf{A} - \mathbf{X}\|.$$

Two similar problems

The fixed precision problem

Given \mathbf{A} and a **tolerance** $\varepsilon > 0$, find \mathbf{Q} such that:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \varepsilon.$$

The fixed rank problem

Given \mathbf{A} and **rank** $k \in \mathbb{N}$, find \mathbf{B} such that:

$$\|\mathbf{A} - \mathbf{B}\| \approx \min_{\text{rank}(\mathbf{X}) \leq k} \|\mathbf{A} - \mathbf{X}\|.$$

Link between ε and k ? \implies (tail) singular values of \mathbf{A} !

(Mirsky's theorem from 1969.)

An “old” theorem to the rescue!

To solve the problem, we will use **A's SVD**: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.
Let σ_j the j -th *largest* singular value of \mathbf{A} .

Leon Mirsky's singular theorem (1969)

$$\min_{\text{range}(\mathbf{X}) \leq k} \|\mathbf{A} - \mathbf{X}\| = \sigma_{k+1}(\mathbf{A}). \quad (2)$$

(See [Woolfe et al., 2008, lemma 3.1] for a more recent proof)

An “old” theorem to the rescue!

To solve the problem, we will use \mathbf{A} 's **SVD**: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.
Let σ_j the j -th *largest* singular value of \mathbf{A} .

Leon Mirsky's singular theorem (1969)

$$\min_{\text{range}(\mathbf{X}) \leq k} \|\mathbf{A} - \mathbf{X}\| = \sigma_{k+1}(\mathbf{A}). \quad (2)$$

(See [Woolfe et al., 2008, lemma 3.1] for a more recent proof)

One way to find the minimizer is to we choose $\mathbf{X} = \mathbf{Q}\mathbf{Q}^* \mathbf{A}$, where \mathbf{Q} are the k dominant (left) singular vectors of \mathbf{A} , and k is numerically determined as the smallest s.t. $\sigma_{k+1} \leq \varepsilon$.

(Not new, see e.g. [Grivet and Ribot, 2013, section 10.6.2])

Prototype stage 1 algorithm

$k \stackrel{\text{def}}{=} \text{rank}(\mathbf{A})$ is given. Chose $p > 0$ **oversampling** parameter,

Stage 1 “Proto-Algorithm” (common stage 1 skeleton)

1. Draw an $n \times (k + p)$ “random” matrix $\mathbf{\Omega}$, (Column selection)
2. Form the matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\mathbf{\Omega}$,
3. Construct a matrix \mathbf{Q} , whose columns form an orthonormal basis for the range of \mathbf{Y} .

Questions:

Prototype stage 1 algorithm

$k \stackrel{\text{def}}{=} \text{rank}(\mathbf{A})$ is given. Chose $p > 0$ **oversampling** parameter,

Stage 1 “Proto-Algorithm” (common stage 1 skeleton)

1. **Draw an $n \times (k + p)$ “random” matrix Ω** , (Column selection)
2. Form the matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct a matrix \mathbf{Q} , whose columns form an orthonormal basis for the range of \mathbf{Y} .

Questions:

– how to **draw** Ω ?

e.g. Gaussian, SRFT

Prototype stage 1 algorithm

$k \stackrel{\text{def}}{=} \text{rank}(\mathbf{A})$ is given. Chose $p > 0$ **oversampling** parameter,

Stage 1 “Proto-Algorithm” (common stage 1 skeleton)

1. Draw an $n \times (k + p)$ “random” matrix Ω , (Column selection)
2. Form the matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct a matrix \mathbf{Q} , whose columns form an orthonormal basis for the range of \mathbf{Y} .

Questions:

- how to **find** the rank k ? → **adaptive** algorithm (cf. report)

Prototype stage 1 algorithm

$k \stackrel{\text{def}}{=} \text{rank}(\mathbf{A})$ is given. Chose $p > 0$ **oversampling** parameter,

Stage 1 “Proto-Algorithm” (common stage 1 skeleton)

1. Draw an $n \times (k + p)$ “random” matrix $\mathbf{\Omega}$, (Column selection)
2. Form the matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\mathbf{\Omega}$,
3. Construct a matrix \mathbf{Q} , whose columns form an orthonormal basis for the range of \mathbf{Y} .

Questions:

- how to **chose** the oversampling parameter p ? → guess!

Prototype stage 1 algorithm

$k \stackrel{\text{def}}{=} \text{rank}(\mathbf{A})$ is given. Chose $p > 0$ **oversampling** parameter,

Stage 1 “Proto-Algorithm” (common stage 1 skeleton)

1. Draw an $n \times (k + p)$ “random” matrix $\mathbf{\Omega}$, (Column selection)
2. Form the matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\mathbf{\Omega}$,
3. **Construct a matrix \mathbf{Q}** , whose columns form an orthonormal basis for the range of \mathbf{Y} .

Questions:

– how to **construct** \mathbf{Q} ?

→ classical QR algorithm

Prototype stage 1 algorithm

$k \stackrel{\text{def}}{=} \text{rank}(\mathbf{A})$ is given. Chose $p > 0$ **oversampling** parameter,

Stage 1 “Proto-Algorithm” (common stage 1 skeleton)

1. Draw an $n \times (k + p)$ “random” matrix Ω , (Column selection)
2. Form the matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct a matrix \mathbf{Q} , whose columns form an orthonormal basis for the range of \mathbf{Y} .

Questions:

- how to draw Ω ? e.g. Gaussian, SRFT
- how to find the rank k ? → adaptive algorithm (cf. report)
- how to chose the oversampling parameter p ? → guess!
- how to construct \mathbf{Q} ? → classical QR algorithm

2nd algorithm: Randomized Range Finder

First explicit algorithm:

Stage 1

Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ **standard Gaussian** random matrix Ω ,
2. Form the $m \times (k + p)$ matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct \mathbf{Q} from \mathbf{Y} 's **QR** factorization.

About p ?

[Tropp, 2014]

Unknown oversampling parameter $p > 0$ should depend on: the matrix dimensions m, n , and the decrease of the ordered singular spectrum, and the random test matrices.

E.g. for Gaussian matrices \mathbf{A} , p between 5 or 10 yields good results.

2nd algorithm: Randomized Range Finder

Stage 1

Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ standard Gaussian random matrix Ω ,
2. Form the $m \times (k + p)$ matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct \mathbf{Q} from \mathbf{Y} 's QR factorization.

Complexity:

(i.e. number of "flops")

About

2nd algorithm: Randomized Range Finder

Stage 1

Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ standard Gaussian random matrix Ω ,
2. Form the $m \times (k + p)$ matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct \mathbf{Q} from \mathbf{Y} 's QR factorization.

Complexity:

(i.e. number of "flops")

About $n \times (k + p) \times T_{\text{rand}}$

– $T_{\text{rand}} = O(1)$ for drawing *one* Gaussian value. (Ziggurat method)

2nd algorithm: Randomized Range Finder

Stage 1

Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ standard Gaussian random matrix Ω ,
2. Form the $m \times (k + p)$ matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct \mathbf{Q} from \mathbf{Y} 's QR factorization.

Complexity:

(i.e. number of "flops")

About

$$+ (k + p) \times T_{\text{mult}}$$

– $T_{\text{mult}} = O(mn)$ for a matrix-vector product between \mathbf{A} (size $m \times n$) and one column of Ω (size n).

2nd algorithm: Randomized Range Finder

Stage 1

Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ standard Gaussian random matrix Ω ,
2. Form the $m \times (k + p)$ matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct \mathbf{Q} from \mathbf{Y} 's QR factorization.

Complexity:

(i.e. number of "flops")

About

$$+ m \times (k + p)^2.$$

2nd algorithm: Randomized Range Finder

Stage 1

Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ standard Gaussian random matrix Ω ,
2. Form the $m \times (k + p)$ matrix $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$,
3. Construct \mathbf{Q} from \mathbf{Y} 's QR factorization.

Complexity:

(i.e. number of “flops”)

$$\begin{aligned} \text{About } n \times (k + p) \times T_{\text{rand}} + (k + p) \times T_{\text{mult}} + m \times (k + p)^2. \\ = O(mn(k + p)). \end{aligned}$$

Works well for \mathbf{A} with *fast-decaying* singular spectrum.

(Proved numerically and theoretically)

3rd algorithm: Randomized Power Iteration

- **Issue:** what if \mathbf{A} 's singular spectrum is not fast-decaying?
[Tropp, 2014, p.41]
- ↪ **Idea:** reduce weights on the small singular values $\sigma_j(\mathbf{A})$.
- **Trick:** instead of $\mathbf{A} = \mathbf{A}_0$, work on $\mathbf{A}_q \stackrel{\text{def}}{=} (\mathbf{A}\mathbf{A}^*)^q \mathbf{A}$.

3rd algorithm: Randomized Power Iteration

- **Issue:** what if \mathbf{A} 's singular spectrum is not fast-decaying?
[Tropp, 2014, p.41]
- ↪ **Idea:** reduce weights on the small singular values $\sigma_j(\mathbf{A})$.
- **Trick:** instead of $\mathbf{A} = \mathbf{A}_0$, work on $\mathbf{A}_q \stackrel{\text{def}}{=} (\mathbf{A}\mathbf{A}^*)^q \mathbf{A}$.

Stage 1

Randomized "Power Iteration" algorithm

1. Draw an $n \times (k + p)$ standard Gaussian random matrix Ω ,
2. Form $\mathbf{Y}_q \stackrel{\text{def}}{=} (\mathbf{A}\mathbf{A}^*)^q \mathbf{A}\Omega$, via alternative application of \mathbf{A} and \mathbf{A}^*
3. Construct $\mathbf{Q} = \mathbf{Q}_q$ from \mathbf{Y}_q 's QR factorization.

In practice: $q = 3, 4$ works well.

4th algorithm: Fast Randomized Range Finder

- **Issue:** Gaussian matrices are not adapted for dense or structured matrices. [Tropp, 2014, p.63]
- ↪ **Idea:** use Fast Fourier Transform (**FFT**) to bring structure. [Cooley and Tukey, 1965, Duhamel and Vetterli, 1990]
- **Trick:** choose a *structured* random matrix Ω (**SRFT**).

4th algorithm: Fast Randomized Range Finder

- **Issue:** Gaussian matrices are not adapted for dense or structured matrices. [Tropp, 2014, p.63]
- ↪ **Idea:** use Fast Fourier Transform (**FFT**) to bring structure. [Cooley and Tukey, 1965, Duhamel and Vetterli, 1990]
- **Trick:** choose a *structured* random matrix Ω (**SRFT**).

Stage 1

Fast Randomized Range Finder algorithm

1. Draw an $n \times (k + p)$ **SRFT test matrix** Ω ,
2. Form $Y \stackrel{\text{def}}{=} A\Omega$,
3. Construct Q from Y 's QR factorization.

SRFT

What is a SRFT ?

A **sub-sampled random Fourier transform** (SRFT) is:

$$\mathbf{\Omega} = \sqrt{\frac{n}{k+p}} \times \mathbf{D} \times \mathcal{F} \times \mathbf{R}.$$

[Tropp, 2012]

SRFT

What is a SRFT ?

A **sub-sampled random Fourier transform** (SRFT) is:

$$\mathbf{\Omega} = \sqrt{\frac{n}{k+p}} \times \mathbf{D} \times \mathcal{F} \times \mathbf{R}.$$

[Tropp, 2012]

Where:

- \mathbf{D} is a $n \times n$ random diagonal Rademacher matrix,
- \mathcal{F} is **the** $n \times n$ unitary discrete Fourier transform, and
- \mathbf{R} is **an** $n \times (k+p)$ matrix whose columns are drawn from \mathbb{I}_n .

One algorithm for stage 2: Direct SVD

Stage 2

Direct SVD algorithm

Input: A , and Q from stage 1.

1. Form the matrix $B \stackrel{\text{def}}{=} Q^* A$,
2. Compute^a the SVD of the matrix $B = \tilde{U} \Sigma V^*$,
3. Form the orthonormal matrix $U \stackrel{\text{def}}{=} Q \tilde{U}$.

^aClassical algorithm, for full or truncated SVD.

$$\begin{aligned}
 A &\approx Q Q^* A = Q Q^* A \\
 &= Q U \Sigma V^* = Q U \Sigma V^*
 \end{aligned}$$

Bounds using the singular spectrum tail

Decompose \mathbf{A} 's SVD with Σ_1 the k largest singular values $\sigma_j(\mathbf{A})$:

$$\begin{cases} \mathbf{A} = \mathbf{U} \begin{pmatrix} \Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1^* \\ V_2^* \end{pmatrix}, \\ \Omega_1 = V_1^* \Omega \quad \text{and} \quad \Omega_2 = V_2^* \Omega. \end{cases}$$

Bounds using the singular spectrum tail

Decompose \mathbf{A} 's SVD with Σ_1 the k largest singular values $\sigma_j(\mathbf{A})$:

$$\begin{cases} \mathbf{A} = \mathbf{U} \begin{pmatrix} \Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2 \end{pmatrix} \begin{pmatrix} \mathbf{V}_1^* \\ \mathbf{V}_2^* \end{pmatrix}, \\ \Omega_1 = \mathbf{V}_1^* \Omega \quad \text{and} \quad \Omega_2 = \mathbf{V}_2^* \Omega. \end{cases}$$

Theorem 1

Error bound for the Proto-Algorithm

Assume that Ω_1 has full row rank. The *spectral norm error* satisfies:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\|^2 = \|(I - P_{\mathbf{Y}}) \mathbf{A}\|^2 \leq \|\Sigma_2\|^2 + \|\Sigma_2 \Omega_2 \Omega_1^\dagger\|^2.$$

With $\mathbf{Y} \stackrel{\text{def}}{=} \mathbf{A}\Omega$, if $P_{\mathbf{Y}}$ is the orthonormal projector of same range that \mathbf{Y} 's, then $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\| = \|(I - P_{\mathbf{Y}}) \mathbf{A}\|$.

[Tropp, 2014, slide 53] and [Halko et al., 2011, theorem 9.1].

Randomized Range Finder

Let $k, p \geq 2$ and $k + p \leq \min(m, n)$.

Theorem 2 Error bound for the Randomized Range Finder

For the **expected** Fröbenius norm error:

$$\mathbb{E} \left[\|(I - P_{\mathbf{Y}})\mathbf{A}\|_F \right] \leq \left(1 + \frac{k}{p-1} \right)^{1/2} \left(\sum_{j>k} \sigma_j^2(\mathbf{A}) \right)^{1/2}.$$

Randomized Range Finder

Let $k, p \geq 2$ and $k + p \leq \min(m, n)$.

Theorem 2 Error bound for the Randomized Range Finder

For the **expected** *Fröbenius* norm error:

$$\mathbb{E} \left[\|(I - P_{\mathbf{Y}})\mathbf{A}\|_F \right] \leq \left(1 + \frac{k}{p-1} \right)^{1/2} \left(\sum_{j>k} \sigma_j^2(\mathbf{A}) \right)^{1/2}.$$

And for the **expected** *spectral* norm error:

$$\mathbb{E} \left[\|(I - P_{\mathbf{Y}})\mathbf{A}\| \right] \leq \left(1 + \frac{k}{p-1} \right) \sigma_{k+1}(\mathbf{A}) + \frac{e\sqrt{k+p}}{p} \left(\sum_{j>k} \sigma_j^2(\mathbf{A}) \right)^{1/2}.$$

Both depend on \mathbf{A} 's **singular spectrum tail** $\|\Sigma_2\|$ (ℓ^2 -norm).

[Tropp, 2014, slide 59] and [Halko et al., 2011, theorem 9.2].

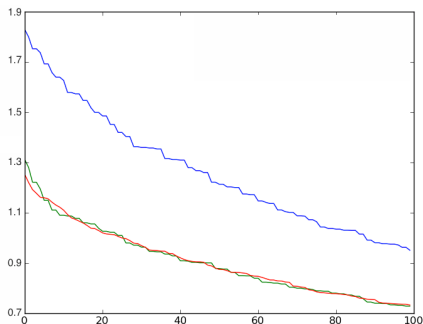
A first simple experiment

Quick overview of experiment 1

- Generate a dense random Gaussian matrix \mathbf{A} of size 500×500 ,
- Make it s -sparse, with a small $s = 30$,
- Compute its singular spectrum directly, with exact SVD,
- Then compare with each stage 1 algorithm (and DirectSVD for the stage 2), on their norm errors $\|A - U_i \Sigma_i V_i^*\|$, and on their singular spectra $\tilde{\sigma}_j^i$.

\implies Each algorithm seemed to work as expected/predicted.

An image processing application



Comparison of 3 stage-A algorithms, decay of the first 100 singular values.

- The **Random Range Finder** (blue) runs for 7 sec.
- The **Random Power Iteration** (darkgreen) runs for 12 sec ($q = 4$).
- The **Fast Random Range Finder** (red) runs for 10 sec.

Quick sum-up

I studied. . .

- Classical matrix factorization algorithm, (QR, SVD)
- Limitations of the classical framework, (e.g. are linear in k)
- The “two stages” framework for matrix factorization.

Mainly from [Halko et al., 2011]

Quick sum-up

We saw how to . . .

- Use *randomization* in stage 1, to efficiently capture A 's range,
- Different algorithm for different structure of A ,
- And then use the *classical* QR / SVD for stage 2.

Quick sum-up

Experimentally, I . . .

- Implemented all these algorithms in Octave/MATLAB,
- Designed a first simple experiment on a random matrix A (Gaussian, uniform),
- Reproduced a less trivial one on a (relatively) “large” sparse matrix (from image processing),
- And both experiments confirmed the theoretical analysis for complexity and for error bounds.

Thank you!

Thank you for your attention.

Questions ?

Want to know more?

- ↪ Explore the references, or read the project report,
- ↪ And e-mail me if needed [lilian.besson\[at\]ens-cachan.fr](mailto:lilian.besson@ens-cachan.fr).

Main references

- J. Tropp (2014), *“Finding Structure with Randomness”*, tutorial slides.
- N. Halko, P.-G. Martinsson and J. Tropp (2011), *“Finding Structure with Randomness: Probabilistic Algorithms for Constructing approximate Matrix Decompositions”*, longer article.
- Z. Zhang (2015), *“Randomized Numerical Linear Algebra (RNLA): review and progresses”*, tutorial slides.

Appendix

Outline of the appendix

- More references given below,
- Code and raw results from some experiments:
→ <http://lbo.k.vu/pcs2016>.
- MIT Licensed.

More references I



Cooley, J. W. and Tukey, J. W. (1965).

An algorithm for the machine calculation of complex Fourier series.

Mathematics of computation, 19(90):297–301.



Duhamel, P. and Vetterli, M. (1990).

Fast Fourier transforms: a tutorial review and a state of the art.

Signal processing, 19(4):259–299.






Grivet, J.-P. and Ribot, M. (DL 2013, cop. 2013).

Méthodes numériques appliquées pour le scientifique et l'ingénieur (in French).

Grenoble Sciences. EDP sciences, Les Ulis, 2nd edition.

More references II

-  Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding Structure with Randomness: Probabilistic algorithms for constructing Approximate Matrix Decompositions. *SIAM review*, 53(2):217–288.
-  Marsaglia, G., Tsang, W. W., et al. (2000). The Ziggurat Method for Generating Random Variables. *Journal of Statistical Software*, 5(8):1–7.
-  Tropp, J. A. (2012). User-friendly tools for Random Matrices. *Neural Information Processing Systems (NIPS), Stateline*.

More references III



Tropp, J. A. (2014).

Finding Structure with Randomness.

Tutorial slides, <http://users.cms.caltech.edu/~jtropp/slides/Tro14-Finding-Structure-ICML.pdf>.



Woolfe, F., Liberty, E., Rokhlin, V., and Tygert, M. (2008).

A Fast Randomized Algorithm for the Approximation of Matrices.
Applied and Computational Harmonic Analysis, 25:335–366.



Zhang, Z. (2014).

Randomized Numerical Linear Algebra (RNLA): review and progresses.

Tutorial slides,

<http://bcmi.sjtu.edu.cn/~zhzhang/papers/rnla.pdf>.

Open-Source Licensed

License?

These slides and the report^a are open-sourced under the terms of the **MIT License** (see `lbesson.mit-license.org`).

2015–2016, © Lilian Besson.

^aAnd the additional resources – including code, images, etc.