# Random Osborne Algorithm for Matrix Balancing

Optimal transport report

Alexi Canesse[1,2]

Optimal transport course
Part of the MVA program at ENS Paris-Saclay.

MATHÉMATIQUES
VISION
APPRENTISSAGE

Computer Science and Mathematics
École Normale Supérieur Paris-Saclay
Orsay, France

7th January 2024

1 alexi.canesse@ens-lyon.fr, ENS de Lyon, France
2 alexi.canesse@ens-paris-saclay.fr, ENS Paris-Saclay, France

# Table of content

Abstract (½ page): What problems is studied? Why is it relevant? What solutions is proposed? Which contributions (theory, numerics, etc)?

# 1 Introduction (3 pages)

[AP23]

## 1.1 Presentation of the problem

Introduce the reasons to do this alg

**Notation 1.1** *Let c and r respectively be the column-wise sum and the row-wise sum of matrices ie.*

$$c : \begin{cases} \mathcal{M}_{n,m}(\mathbb{K}) & \to & \mathbb{K} \\ A & \mapsto & A^\top \mathbf{1} \end{cases} \quad \text{and} \quad r : \begin{cases} \mathcal{M}_{n,m}(\mathbb{K}) & \to & \mathbb{K} \\ A & \mapsto & A\mathbf{1}. \end{cases}$$

Use 2.3 to explain it using convex opti

**Definition 1.1** *Let $A \in \mathcal{M}_n(\mathbb{R}_+)$ be a non negative square matrix, $\varepsilon \geq 0$ and $k \in \mathbb{N}^\star$. The matrix $A$ is $(\varepsilon, k$-balanced if*

$$\frac{||c(A) - r(A)||_k}{\sum_{i,j} a_{i,j}} \leq 0.$$

*Furthermore, if $A$ is $(0, k)$-balanced, we say that $A$ is balanced.*

Use real formal definition from 2.1

**Definition 1.2** *The $\varepsilon, k$-**approximate matrix balancing problem** is: given a square non-negative matrix $K \in \mathcal{M}_n(\mathbb{R}_+)$, $\varepsilon \geq 0$ and $k \in \mathbb{N}^\star$, find a positive diagonal matrix $D \in \mathcal{D}_n(\mathbb{R}_+^\star)$ such that $DKD^{-1}$ is $(\varepsilon, k)$-balanced.*

## 1.2 Related work

Previous works (at least a few citations). If relevant, include things that you have seen during the MVA course (or possibly other courses).

Osborn algorithm introduced in[Osb60; PR71] and defaut in Scipy.

[CD00; Che01] improve accuracy of computation of eigen vectors, eigen values.

there are corner cases where balancing can actually worsen the conditioning. [Wat06] also [JLL14] in which they explain it and explain how to modify LAPACK to avoid it.

The standard matrix balancing algorithm is the Sinkhorn-Knopp algorithm [SK67], a special case of Bregman's balancing method [LS81] that iterates rescaling of each row and column until convergence. However, the algorithm converges linearly [Sou91], which is prohibitively slow for recently emerging large and sparse matrices.

[PR71] -> approximately balance matrices with a diagonal with only powers of 2. Advantage : no floating point error in computing the balanced matrix on base two computers. approximately "good enough"

[SNT17] -> matrix balancing on tensors

### 1.3 Contributions of the paper

Their main contribution is theorem 1.1. It exhibits a variant of OSBORN's algorithm with near-linear runtime in the input sparsity. It also shows that improving the runtime dependence in $\varepsilon$ can be improve from $\varepsilon^{-2}$ to $\varepsilon^{-1}$ without an additional factor $n$.

**Theorem 1.1** *Let $K \in \mathcal{M}_n(\mathbb{R}_+)$ be a balanceable non negative square matrix and $\varepsilon \geq 0$. Random OSBORN solves $(\varepsilon, 1)$-approximate matrix balancing problem in $T$ operations where there exists $c > 0, \delta > 0$ such that*

$$\mathbb{E}(T) = \mathcal{O}\left(\frac{m}{\varepsilon} \min\left\{\frac{1}{\varepsilon}; d\right\} \log \kappa\right) \quad \text{and} \quad \mathbb{P}\left(T \leq c\frac{m}{\varepsilon} \min\left\{\frac{1}{\varepsilon}; d\right\} \log \kappa \log \frac{1}{\delta}\right) \geq 1 - \delta$$

*where $m$ is the number of nonzero entries in $K$, $d$ is the diameter of the graph associated to $K$ and $\kappa = \sum_{i,j} K_{i,j} / \min_{i,j} K_{i,j}$.*

### 1.4 Our contributions

numerics? limits?

## 2 Main body (10 pages)

### 2.1 Notations

### 2.2 Presentation of the method

```
osborn(K, ε):
    x = 0 ∈ ℝ^n
    while not(is_balanced(𝔻(e^x)K𝔻(e^{-x}), ε)):
        Choose k ∈ [n] # This is where variants differ
        x += (log(c_k(𝔻(e^x)K𝔻(e^{-x}))) - log(r_k(𝔻(e^x)K𝔻(e^{-x}))))/2
    return x
```

There are *many* way to choose the next coordinate to update and hence many variants of the algorithm. The article focuses on four of them:

- **Cyclic Osborn** Cycle through the coordinates. (*eg.* 1, 2, 3, 1, 2, 3, 1, ...).
- **Random-Reshuffle Cyclic Osborn** Cycle through the coordinates using a new random permutation for each cycle. (*eg.* 2, 1, 3, 1, 2, 3, 1, 3, 2, ...).
- **Greedy Osborn** Choose $k$ where the imbalance is maximal *eg.*

$$k = \text{argmax}_k \left| \sqrt{r_k(\mathbb{D}(e^x)K\mathbb{D}(e^{-x}))} - \sqrt{c_k(\mathbb{D}(e^x)K\mathbb{D}(e^{-x}))} \right|.$$

- **Random Osborn** Uniformly sample $k$ independently between each call.

Talk about the implementation

### 2.3 Theoretical guarantees

### 2.4 Numerics

it indeed converges even with high sparsity (cf theorem)

#### 2.4.1 Sparsity

We conducted numerical experiments to investigate the behavior of Osborn's algorithm in the presence of varying numbers of zero entries within randomly generated matrices. Each matrix has a size of (10, 10) with uniformly distributed values in the range [0, 1]. The number of zero entries in the matrices was systematically varied. Our objective was to assess the algorithm's ability to find solutions even with a small number of non-zero inputs, as proven in [ref].
The experiment involved measuring the execution time of Osborn's algorithm for each matrix, with the time recorded as a function of the number of zero entries. This exploration aimed to provide insights into the near-linear convergence time of the algorithm, shedding light on its performance characteristics under different sparsity levels. These results contribute valuable empirical evidence to support the theoretical findings presented in [ref]."

Use obeservation 2.5 to compare with other convex optim algorithms.

Look at per iteration runtime (5.2)

## 3 Conclusion and perspective

Summary of the result obtained: pros and cons (limitation, problems, error in the articles, etc) Possible improvement/extension

## 4 Connexion with the course

MANDATORY SECTION:. What are the notions/results/algorithms presented in the course that are used or related to the one presented in this paper?

## Todo list



## References


[AP23] Jason M Altschuler and Pablo A Parrilo. "Near-linear convergence of the Random Osborne algorithm for Matrix Balancing". In: *Mathematical Programming* 198.1 (2023), pp. 363–397.

[CD00] Tzu-Yi Chen and James W Demmel. “Balancing sparse matrices for computing eigenvalues”. In: *Linear algebra and its applications* 309.1-3 (2000), pp. 261–287.
[Che01] Tzu-Yi Chen. *Preconditioning sparse matrices for computing eigenvalues and solving linear systems of equations.* University of California, Berkeley, 2001.
[JLL14] Rodney James, Julien Langou and Bradley R Lowery. “On matrix balancing and eigenvector computation”. In: *arXiv preprint arXiv:1401.5766* (2014).
[LS81] B Lamond and Neil F Stewart. “Bregman’s balancing method”. In: *Transportation Research Part B: Methodological* 15.4 (1981), pp. 239–248.
[Osb60] EE Osborne. “On pre-conditioning of matrices”. In: *Journal of the ACM (JACM)* 7.4 (1960), pp. 338–345.
[PR71] Beresford N Parlett and Christian Reinsch. “Balancing a matrix for calculation of eigenvalues and eigenvectors”. In: *Handbook for Automatic Computation* 2 (1971), pp. 315–326.
[SK67] Richard Sinkhorn and Paul Knopp. “Concerning nonnegative matrices and doubly stochastic matrices”. In: *Pacific Journal of Mathematics* 21.2 (1967), pp. 343–348.
[SNT17] Mahito Sugiyama, Hiroyuki Nakahara and Koji Tsuda. “Tensor balancing on statistical manifold”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3270–3279.
[Sou91] George W Soules. “The rate of convergence of Sinkhorn balancing”. In: *Linear algebra and its applications* 150 (1991), pp. 3–40.
[Wat06] David S Watkins. “A case where balancing is harmful”. In: *Electron. Trans. Numer. Anal* 23 (2006), pp. 1–4.