Extrapolation for PageRank and Multilinear PageRank

Michela REDIVO ZAGLIA

UNIVERSITY OF PADUA (ITALY)

PART I

in collaboration with **Claude Brezinski, University of Lille (France)**

- ➜ History: Web search and PageRank
- ➜ The Google matrices
- ➜ The PageRank vector
- ➜ The power method
- ➜ Approximations of the PageRank vector
- ➜ Acceleration of the power method
- ➜ Extrapolation for the PageRank vector

History: Web search and PageRank

- An important problem in **web search** is to **classify the pages according to their importance** beginning by the most important ones, making information on the Web more accessible.
- The start of Google's history date to **1995**, when **Larry Page (b. 1973)** and **Sergey Brin (b. 1973**, two PhD students, met at Stanford University and began working together on a project called **BackRub**.
- This project was based on a **new algorithm** called **PageRank**, which evaluated the relevance of web pages based on the links they received from other pages.
- The original idea for giving a **rank** to each page is that **a page is important if other important pages point to it**.

- The vector $\mathbf{r}$ containing these ranks is called the **PageRank vector** .

  It is only defined **implicitly** (that is **recursively**).

- In **1998**, Page and Brin founded Google Inc. and launched their own search engine under the name **Google**.

- The name **Google** derives from **googol** mathematical term indicating the number 1 followed by 100 zeros, representing the immense amount of information that the search engine would have to manage.

- Over the years, Google established itself as the **leading search engine in the world**.

- In **2004**, Google entered the stock market, with one of the largest and most successful IPOs (Initial Public Offerings) in history. This made Page and Brin billionaires, and allowed the company to finance further projects and acquisitions.
- In **December 2019**, Page and Brin announced their departure from Google, leaving **Sundar Pichai (b. 1972)** as CEO and they chose to focus on new projects and interests.
- It seems that nowadays **PageRank is still used in Google's ranking algorithms**. However, certainly the PageRank algorithm is very different than it was originally.

THE GOOGLE MATRICES

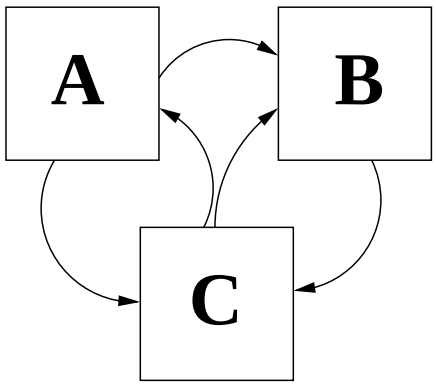
Let $\mathbf{deg(i)}$ be the **outdegree** of the page $\mathbf{i}$, that is the number of pages it points to.

The **Google matrix** $\mathbf{P} = (\mathbf{p_{ij}})$ is defined by

$$\mathbf{p_{ij}} = \begin{cases} \mathbf{1/deg(i)}, & \text{if page } \mathbf{i} \text{ links to } \mathbf{j}, \\ \mathbf{0}, & \mathbf{i = j}. \end{cases}$$

Remark: The dimension p is billions by billions !

Example:



$$\mathbf{P} = \begin{array}{c|ccc} & \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \hline \mathbf{A} & 0 & 1/2 & 1/2 \\ \mathbf{B} & 0 & 0 & 1 \\ \mathbf{C} & 1/2 & 1/2 & 0 \end{array}$$

The **PageRank vector** $\mathbf{r}$ is the **left** eigenvector of $\mathbf{P}$ with the eigenvalue $\mathbf{1}$, that is

$$\mathbf{r} = \mathbf{P^T r}.$$

We want to compute it by the **power method**

$$\mathbf{r^{(n+1)}} = \mathbf{P^T r^{(n)}}, \quad \mathbf{n = 0, 1, \ldots,} \quad \mathbf{r^{(0)} = v}.$$

Unfortunately, the power method has **convergence problems** since $\mathbf{P}$ is **not stochastic** (some of its rows are 0). This is due to **dangling nodes** (pages **without outlinks**).

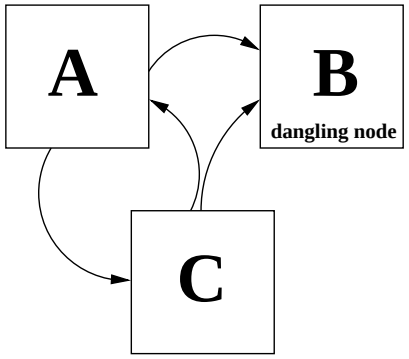
For avoiding these drawbacks, $\mathbf{P}$ is **replaced** by

$$\widetilde{\mathbf{P}} = \mathbf{P} + \mathbf{d}\mathbf{w}^{\mathbf{T}}$$

where $\mathbf{w}$ is a **probability vector** ($\mathbf{w} \geq \mathbf{0}$, $(\mathbf{w}, \mathbf{e}) = 1$, $\mathbf{e} = (1, \ldots, 1)^T$), for instance $\mathbf{w} = \mathbf{e}/p$,

and $\mathbf{d} = (d_i)$ another vector, with $\begin{cases} d_i = 1 \text{ if } \deg(i) = 0 \\ d_i = 0 \text{ otherwise} \end{cases}$

Example:



$\mathbf{P}=$

	A	B	C
A	0	1/2	1/2
B	0	0	0
C	1/2	1/2	0

$\widetilde{\mathbf{P}}=$

	A	B	C
A	0	1/2	1/2
B	1/3	1/3	1/3
C	1/2	1/2	0

Now, $\widetilde{\mathbf{P}}$ is **stochastic**, and has $\mathbf{1}$ as its dominant eigenvalue with $\mathbf{e} = (\mathbf{1}, \ldots, \mathbf{1})^{\mathbf{T}}$ as its corresponding right eigenvector.

But **another problem** arises since $\widetilde{\mathbf{P}}$ is **reducible**: it can have several eigenvalues on the unit circle, and it has **several** left eigenvectors corresponding to the dominant eigenvalue 1.

Thus, $\widetilde{\mathbf{P}}$ is **replaced** by the matrix

$$\mathbf{P_c} = \mathbf{c}\widetilde{\mathbf{P}} + (\mathbf{1} - \mathbf{c})\mathbf{E}, \quad \mathbf{E} = \mathbf{e}\mathbf{v}^{\mathbf{T}}$$

with $\mathbf{c} \in [\mathbf{0}, \mathbf{1})$, and $\mathbf{v}$ a **probability vector** (personalization vector), for instance $\mathbf{v} = \mathbf{e}/p$.

$\mathbf{P_c}$ is **stochastic** and **irreducible**

It has $\mathbf{1}$ as its dominant eigenvalue with
– $\mathbf{e}$ as its corresponding **right** eigenvector, and
– $\mathbf{r_c}$ as its corresponding **left** eigenvector.

The power iterations

$$\mathbf{r_c^{(n+1)} = P_c^T r_c^{(n)}}, \quad \mathbf{n = 0, 1, \ldots,} \qquad \mathbf{r_c^{(0)} = v},$$

now **converge**, with speed of convergence $\mathcal{O}(\mathbf{c^n})$, to the **unique** vector

$$\mathbf{r_c = P_c^T r_c}$$

which is chosen as the **PageRank vector**.

The **speed of convergence** depends on $\mathbf{c} \in [\mathbf{0}, \mathbf{1})$.
A **balance** has to be found between
– a small value of $\mathbf{c}$ (fast convergence, but $\mathbf{r_c}$ is not a good approximation of the **true PageRank vector** $\widetilde{\mathbf{r}} = \lim_{\mathbf{c} \to \mathbf{1}^-} \mathbf{r_c}$)
– a value of $\mathbf{c}$ close to 1 (better $\mathbf{r_c}$, but slow convergence)

Google chooses $\mathbf{c = 0.85}$

THE PAGERANK VECTOR

We will denote

- $\widetilde{\lambda}_1 = 1, \widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_p$ the **eigenvalues** of $\widetilde{\mathbf{P}}$ ($1 \geq |\widetilde{\lambda}_2| \geq \cdots \geq |\widetilde{\lambda}_p|$)
- $\mathbf{e}, \mathbf{x_2}, \ldots, \mathbf{x_p}$ its corresponding **right eigenvectors** and
- $\mathbf{y}, \mathbf{y_2}, \ldots, \mathbf{y_p}$ its corresponding **left eigenvectors**

(remark that $\mathbf{y}$ is **one** of the PageRank vectors corresponding to $\mathbf{c} = \mathbf{1}$, since $\widetilde{\mathbf{P}}$ can have several left eigenvectors w.r.t. $\widetilde{\lambda}_1 = 1$)

- $\widetilde{\mathbf{r}} = \lim_{\mathbf{c} \to 1^-} \mathbf{r_c}$

We set $\widetilde{\mathbf{A}} = \widetilde{\mathbf{P}}^{\mathbf{T}}$ and $\mathbf{A_c} = \mathbf{P_c^T}$. Thus

$$\mathbf{r_c} = \mathbf{A_c r_c}$$

$\mathbf{r_c} \geq \mathbf{0}$ since it is a **probability** vector, and it is **normalized** so that $(\mathbf{r_c}, \mathbf{e}) = \mathbf{1}$.

We will now give **implicit** and **explicit** expressions for $\mathbf{r_c}$.

These expressions will be used later.

IMPLICIT EXPRESSIONS

From the expression for $\widetilde{\mathbf{P}}$, we have $\mathbf{r_c} = \mathbf{c}\widetilde{\mathbf{A}}\mathbf{r_c} + \mathbf{v}$.
Thus $\mathbf{r_c}$ is the solution of the system $(\mathbf{I} - \mathbf{c}\widetilde{\mathbf{A}})\mathbf{r_c} = (\mathbf{1} - \mathbf{c})\mathbf{v}$ and we have

$$\begin{aligned} \mathbf{r_c} &= (\mathbf{1} - \mathbf{c})(\mathbf{I} - \mathbf{c}\widetilde{\mathbf{A}})^{-\mathbf{1}}\mathbf{v} \\ &= \mathbf{v} + \mathbf{c}(\widetilde{\mathbf{A}} - \mathbf{I})(\mathbf{I} - \mathbf{c}\widetilde{\mathbf{A}})^{-\mathbf{1}}\mathbf{v} \end{aligned}$$

and we immediately obtain (Boldi, Santini, Vigna, 2005)

$$\mathbf{r_c} = (\mathbf{1} - \mathbf{c}) \sum_{\mathbf{i}=\mathbf{0}}^{\infty} \mathbf{c}^{\mathbf{i}} \widetilde{\mathbf{A}}^{\mathbf{i}} \mathbf{v}$$

$$\mathbf{r_c} = \mathbf{v} + \mathbf{c}(\widetilde{\mathbf{A}} - \mathbf{I}) \sum_{\mathbf{i}=\mathbf{0}}^{\infty} \mathbf{c}^{\mathbf{i}} \widetilde{\mathbf{A}}^{\mathbf{i}} \mathbf{v}$$

These power series converge since $\rho(\widetilde{\mathbf{A}}) = \mathbf{1}$ and $\mathbf{0} \leq \mathbf{c} < \mathbf{1}$.

EXPLICIT EXPRESSIONS

There are two types of explicit expressions for $\mathbf{r_c}$:

polynomial and **rational**.

POLYNOMIAL EXPLICIT EXPRESSION:

Let Π_m be the minimal polynomial of $\mathbf{A_c}$ for the vector $\mathbf{v}$ ($m \leq p$), that is the polynomial of least degree such that $\Pi_m(\mathbf{A_c})\mathbf{v} = \mathbf{0}$.

Since this matrix has an eigenvalue equal to $\mathbf{1}$, then

$$\Pi_m(\lambda) = (\lambda - 1)Q_{m-1}(\lambda).$$

So,

$$\Pi_m(\mathbf{A_c})\mathbf{v} = \mathbf{A_c}Q_{m-1}(\mathbf{A_c})\mathbf{v} - Q_{m-1}(\mathbf{A_c})\mathbf{v} = \mathbf{0}.$$

Thus

$$\mathbf{r_c} = Q_{m-1}(\mathbf{A_c})\mathbf{v}$$

RATIONAL EXPLICIT EXPRESSION: As proved by Serra-Capizzano (2005), for a **general** $\widetilde{\mathbf{P}}$ we have

$$\mathbf{r_c} = \widetilde{\mathbf{r}} + \sum_{\mathbf{i=s+1}}^{\mathbf{p}} \mathbf{w_i(c)y_i}$$

where

- $\widetilde{\mathbf{r}} = \mathbf{y} + \sum_{\mathbf{i=2}}^{\mathbf{s}} \alpha_\mathbf{i}\, \mathbf{y_i}$ ($\mathbf{s}$ is the multiplicity of $\widetilde{\lambda}_\mathbf{1} = \mathbf{1}$)
- $\mathbf{w_i(c)} = \dfrac{[(\mathbf{1}-\mathbf{c})\alpha_i + \mathbf{c}\,\beta_i\,\mathbf{w_{i-1}(c)}]}{(\mathbf{1}-\mathbf{c}\widetilde{\lambda}_\mathbf{i})}, \quad i = s+1,\ldots,p$
  with β_i equal to 0 or 1
- $\alpha_\mathbf{i} = \mathbf{x_i^T v}, \quad i = 2,\ldots,p$

So, $\mathbf{r_c}$ is a **rational function** of type $(p-1, p-1)$ in the variable $\mathbf{c}$

Remark: If $\widetilde{\mathbf{P}}$ is diagonalizable the expression simplifies.

The power method

We consider the **power method**

$$\begin{aligned} \mathbf{r_c^{(0)}} &= \mathbf{v} \\ \mathbf{r_c^{(n+1)}} &= \mathbf{A_c r_c^{(n)}}, \quad \mathbf{n} = 0, 1, \ldots \end{aligned}$$

It holds $\forall \mathbf{n}, \mathbf{r_c^{(n)}} = \mathbf{A_c^n v} \geq \mathbf{0}$ and $\mathbf{e^T r_c^{(n)}} = \mathbf{1}$.

Remark: The iterates of the power method are in fact the **partial sums** of the series expansion of $\mathbf{r_c}$ (implicit expressions) given by (Boldi, Santini, Vigna, 2005).

IMPLEMENTATION:

The **power method** can be **easily implemented** since we have

$$\mathbf{r_c^{(n+1)}} = \mathbf{c}\,\mathbf{P^T r_c^{(n)}} + (\mathbf{c} - \|\mathbf{c}\,\mathbf{P^T r_c^{(n)}}\|_\mathbf{1})\mathbf{w} + (\mathbf{1} - \mathbf{c})\mathbf{v}$$

- $\mathbf{P}$ (the exact Google matrix) is extremely sparse.
- The average number of nonzeros per row is less than 10
- each vector-matrix multiplication requires $O(nnz(P)) \simeq O(p)$ flops
- at each iteration, the method only requires the storage of one vector.

PROBLEMS:

- When $\mathbf{c}$ is close to $\mathbf{1}$ the matrix $\mathbf{A_c}$ becomes more and more **ill conditioned** since its conditioning behaves like $(\mathbf{1}-\mathbf{c})^{-1}$, (Kamvar-Haveliwala, 2003).
- When $\mathbf{c}$ is close to $\mathbf{1}$, the convergence becomes **slow**.
- $\mathbf{r_c}$ **highly depends** on $\mathbf{c}$ and on $\mathbf{v}$, (Serra-Capizzano, 2004).
- We need **continuous updates** to ranking, and computing $\mathbf{r_c}$ can take **several days**.
- We need to compute $\mathbf{r_c}$ for many personalized vector $\mathbf{v}$.

POSSIBLE ANSWERS: **Approximations** of $\mathbf{r_c}$.

Acceleration of the power method.

Extrapolation near $c = 1$.

APPROXIMATIONS OF THE PAGERANK VECTOR

As seen above, $\mathbf{r_c}$ is a rational function of type $(p-1, p-1)$ (or $(m-1, m-1)$), in the variable $\mathbf{c}$ and its vector Taylor series expansion is known.

The **partial sums** of this series could be used for constructing a rational approximation of $\mathbf{r_c}$ called **vector Padé-type approximants** (Van Iseghem, 1986). So, chosen $k < m \leq p$, we are looking for a rational function

$$(k-1/k-1)_{\mathbf{r_c}}(\mathbf{c}) = \frac{\mathbf{P}_{k-1}(\mathbf{c})}{Q_{k-1}(\mathbf{c})}$$

($\mathbf{P_{k-1}(c)}$ has vectors coefficients $\mathbf{a_i} \in \mathbb{R}^p$ and $Q_{k-1}(\mathbf{c})$ scalars coefficients $b_i \in \mathbb{R}$) so that

$$Q_{k-1}(\mathbf{c})\mathbf{r_c}(\mathbf{c}) - \mathbf{P}_{k-1}(\mathbf{c}) = \mathcal{O}(\mathbf{c}^k)$$

Remark: It is not possible to construct $[k-1/k-1]_{\mathbf{r_c}}(\mathbf{c})$. that is the vector Padé approximant of $\mathbf{r_c}$ of order $\mathcal{O}(\mathbf{c}^{2k-1})$.

ACCELERATION OF THE POWER METHOD

The idea behind a **convergence acceleration method** is **extrapolation**.

Let $(\mathbf{x}^{(n)})$ be **a vector sequence converging to** $\mathbf{x}$.
An extrapolation procedure can be viewed as a **sequence transformation**

$$T : (\mathbf{x}^{(n)}) \longrightarrow (\mathbf{y}^{(n)})$$

such that, **under some assumptions**,

$$\lim_{n\to\infty} \frac{\|\mathbf{y}^{(n)} - \mathbf{x}\|}{\|\mathbf{x}^{(n)} - \mathbf{x}\|} = 0$$

that is **the sequence** $(\mathbf{y}^{(n)})$ **converges to** $\mathbf{x}$ **faster than** $(\mathbf{x}^{(n)})$.

In **our case**, the sequence to be accelerated is generated by the power method, that is

$$\mathbf{x^{(n)}} = \mathbf{r_c^{(n)}} = \mathbf{A_c^n v}, \quad \mathbf{n = 0, 1, \ldots}$$

Let Q_{k-1} be a polynomial of degree $k-1 < m-1$ approximating the polynomial Q_{m-1} defined in the polynomial explicit expression of $\mathbf{r_c}$, and constructed from the vectors $\mathbf{r_c^{(n)}}, \mathbf{r_c^{(n+1)}}, \ldots$.
Since

$$\mathbf{r_c} = Q_{m-1}(\mathbf{A_c})\mathbf{v}$$

we will consider the new sequences

$$\mathbf{r_c^{(k,n)}} = Q_{k-1}(\mathbf{A_c})\mathbf{v}$$

with either k fixed and n tending to infinity, or n fixed and k tending to infinity.

In **2003** two papers

- *S.D. Kamvar, T.H. Haveliwala, C.D. Manning, G.H. Golub*, Extrapolations methods for accelerating PageRank computations.
- *T. Haveliwala, S. Kamvar, D. Klein, C. Manning, G.H. Golub* Computing PageRank using power extrapolation.

proposed three methods for **accelerating the sequence of iterates** obtained by the **PageRank power method**: **Aitken Extrapolation, Epsilon Extrapolation**, and **Quadratic Extrapolation**.

The authors did not realize that two of their methods were exactly two mathematically equivalent formulas of Aitken's Δ^2 process, and that the third one could easily be generalized as we will show now.

VECTOR LEAST SQUARES EXTRAPOLATION

Let us construct an approximation

$$P_k(\lambda) = a_0 + \cdots + a_{k-1}\lambda^{k-1} + a_k\lambda^k$$

of the minimal polynomial Π_m.

We have, with $a_k = 1$,

$$\mathbf{A_c^n} P_k(\mathbf{A_c})\mathbf{v} = a_0\mathbf{r_c^{(n)}} + \cdots + a_{k-1}\mathbf{r_c^{(n+k-1)}} + \mathbf{r_c^{(n+k)}} \simeq 0.$$

That is

$$\mathbf{R_n a} \simeq -\mathbf{r_c^{(n+k)}}$$

with $\mathbf{R_n} = [\mathbf{r_c^{(n)}}, \ldots, \mathbf{r_c^{(n+k-1)}}]$ and $\mathbf{a} = (a_0, \ldots, a_{k-1})^T$.

Solving this system in the **least squares sense** gives the coefficients of $P_k(\lambda)$

$$\mathbf{a} = -(\mathbf{R_n^T R_n})^{-1}\mathbf{R_n^T r_c^{(n+k)}}.$$

Now, since

$$P_k(\lambda) = (\lambda - 1)Q_{k-1}(\lambda)$$
$$a_0 + \cdots + a_{k-1}\lambda^{k-1} + \lambda^k = (\lambda - 1)(b_0 + \cdots + b_{k-1}\lambda^{k-1}).$$

Thus

$$b_i = a_{i+1} + \cdots + a_k, \quad i = 0, \ldots, k-1$$

and it follows

$$\mathbf{r_c^{(k,n)}} = Q_{k-1}(\mathbf{A_c})\mathbf{r_c^{(n)}} = b_0\mathbf{r_c^{(n)}} + \cdots + b_{k-1}\mathbf{r_c^{(n+k-1)}}.$$

Let $\mathbf{e^{(k,n)}} = \mathbf{A_c r_c^{(k,n)}} - \mathbf{r_c^{(k,n)}}$. It is easy to prove that

$$\|\mathbf{e^{(k+1,n)}}\| \leq \|\mathbf{e^{(k,n)}}\|.$$

Moreover $\mathbf{e^{(k,n)}}$ and $\mathbf{r_c^{(k,n)}}$ can be expressed as **ratios of determinants** and **Schur complements**.

So, when n is **fixed** and k **increases**, the $\mathbf{r_c^{(k,n)}}$'s become, in general, **more accurate approximations** of $\mathbf{r_c}$ and, for $k = m$, **the exact result** $\mathbf{r_c}$ is obtained.

The highest k, the greatest the **number of vectors to store**. Thus, in practice, the values of k are to be kept **small**.

Remark: the vector $\mathbf{a}$ can be computed by using **any left inverse** $\mathbf{Z_n}$ of $\mathbf{R_n}$, and we get

$$\mathbf{a} = -(\mathbf{Z_n^T R_n})^{-1} \mathbf{Z_n^T r_c^{(n+k)}}.$$

This procedure is a generalization to an arbitrary value of k of the **Quadratic Extrapolation** of Kamvar et al. which corresponds to $\mathbf{k = 3}$, and give it a theoretical justification.

For $k = 2$, we obtain the new vector sequence transformation

$$\mathbf{r_c^{(2,n)}} = (\mathbf{A_c} - \alpha_\mathbf{n}\mathbf{I})\mathbf{r_c^{(n)}}$$

with

$$\alpha_\mathbf{n} = \frac{(\mathbf{\Delta r_c^{(n)}}, \mathbf{\Delta r_c^{(n+1)}})}{(\mathbf{\Delta r_c^{(n)}}, \mathbf{\Delta r_c^{(n)}})}$$

where $\mathbf{\Delta r_c^{(n)}} = \mathbf{r_c^{(n+1)}} - \mathbf{r_c^{(n)}}$.

THE ε–ALGORITHMS

Let $(\mathbf{x}^{(\mathbf{n})})$ be a sequence of vectors converging to $\mathbf{x}$.

The **vector ε–algorithm (P. Wynn, 1962)** consists in the recursive rules

$$\begin{aligned}
\boldsymbol{\varepsilon}_{\mathbf{-1}}^{(\mathbf{n})} &= \mathbf{0} \\
\boldsymbol{\varepsilon}_{\mathbf{0}}^{(\mathbf{n})} &= \mathbf{x}^{(\mathbf{n})} \\
\boldsymbol{\varepsilon}_{\mathbf{k+1}}^{(\mathbf{n})} &= \boldsymbol{\varepsilon}_{\mathbf{k-1}}^{(\mathbf{n+1})} + \left[\boldsymbol{\varepsilon}_{\mathbf{k}}^{(\mathbf{n+1})} - \boldsymbol{\varepsilon}_{\mathbf{k}}^{(\mathbf{n})}\right]^{-\mathbf{1}}, \quad k = 0, 1, \ldots, n = 0, 1, \ldots
\end{aligned}$$

where the **inverse of a vector** $\mathbf{y}$ is defined by $\mathbf{y}^{-\mathbf{1}} = \mathbf{y}/(\mathbf{y}, \mathbf{y})$.
The vectors with an odd lower index are intermediate computations, while those with an even lower index approximate $\mathbf{x}$.
These rules are also valid for the **scalar ε–algorithm (P. Wynn, 1956)** with $\varepsilon_0^{(\mathbf{n})} = (\mathbf{x}^{(\mathbf{n})})_i$, the i th component of $\mathbf{x}^{(\mathbf{n})}$.

For **any** of these algorithms, if the sequence $(\mathbf{x}^{(\mathbf{n})})$ satisfies

$$b_0(\mathbf{x}^{(\mathbf{n})} - \mathbf{x}) + \cdots + b_{m-1}(\mathbf{x}^{(\mathbf{n+m-1})} - \mathbf{x}) = 0, \quad \mathbf{n} = 0, 1, \ldots,$$

then

$$\boldsymbol{\varepsilon}_{\mathbf{2m-2}}^{(\mathbf{n})} = \mathbf{x}, \qquad n = 0, 1, \ldots$$

This is **exactly our case** since

$$\begin{aligned} \mathbf{r_c} &= Q_{m-1}(\mathbf{A_c})\mathbf{r}_{\mathbf{c}}^{(\mathbf{n})} \\ \mathbf{r_c} &= Q_{m-1}(\mathbf{A_c})\mathbf{r_c}. \end{aligned}$$

Thus, applying one of the ε**–algorithms** to the vector sequence $(\mathbf{r}_{\mathbf{c}}^{(\mathbf{n})})$ yields, for $k \geq 2$ and $n = 0, 1, \ldots$,

$$\boldsymbol{\varepsilon}_{\mathbf{2k-2}}^{(\mathbf{n})} = \mathbf{r}_{\mathbf{c}}^{(\mathbf{k,n})} = Q_{k-1}(\mathbf{A_c})\mathbf{r}_{\mathbf{c}}^{(\mathbf{n})}$$

AITKEN'S Δ^2 PROCESS:

When we take $\mathbf{k = 2}$ in the **scalar ε–algorithm**, **Aitken's Δ^2 process** is recovered.

It can be written in different ways. For example, when applied to a **scalar sequence** (S_n) (in our case **each component of the vectors $\mathbf{r_c^{(n)}}$ plays successively the role of** S_n), we have the **three following equivalent formulae:**

$$\begin{aligned}
\varepsilon_2^{(n)} &= S_n - \frac{(S_{n+1} - S_n)^2}{S_{n+2} - 2S_{n+1} + S_n} && \Longleftarrow \text{ Aitken Extrapolation} \\
&= S_{n+1} - \frac{(S_{n+2} - S_{n+1})(S_{n+1} - S_n)}{S_{n+2} - 2S_{n+1} + S_n} && \Longleftarrow \text{ Epsilon Extrapolation} \\
&= S_{n+2} - \frac{(S_{n+2} - S_{n+1})^2}{S_{n+2} - 2S_{n+1} + S_n}
\end{aligned}$$

which are the sequence transformations proposed by Kamvar et al.

EXTRAPOLATION FOR THE PAGERANK VECTOR

➜ We want to **compute** $\mathbf{r_c}$ for a certain value of $\mathbf{c}$ (0.85 or closer to 1), and we know that when $\mathbf{c}$ is close to 1, the power method becomes slower, and the problem is ill-conditioned.

➜ Thus our idea is to consider **several** (smaller) values $\mathbf{c_i}$ of the parameter, and to compute, by the power method, the corresponding vectors $\mathbf{r_{c_i}}$.

➜ After that, we **interpolate** these vectors by *some function* of the parameter, and then we **extrapolate** the results at the desired $\mathbf{c}$.

Important: It is possible to apply the power method for different values of $\mathbf{c}$ at a low additional cost (only costs the number of iterations needed for $\max_\mathbf{i} \mathbf{c_i}$).

Why?

As said before, the iterates of the power method are the partial sums of the series expansion of $\mathbf{r_c}$ (implicit expressions) given by Boldi, Santini, Vigna.

That is

$$\begin{aligned} \mathbf{r_c^{(n+1)}} &= \mathbf{v} + \mathbf{c}(\widetilde{\mathbf{A}} - \mathbf{I})\sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{c^i}\widetilde{\mathbf{A}}^{\mathbf{i}}\mathbf{v} \\ &= \mathbf{r_c^{(n)}} + \mathbf{c^{n+1}}(\widetilde{\mathbf{A}} - \mathbf{I})\widetilde{\mathbf{A}}^{\mathbf{n}}\mathbf{v}, \quad \mathbf{n = 0, 1, \ldots} \end{aligned}$$

with $\mathbf{r_c^{(0)}} = \mathbf{v}$. Thus, **for any c**

$$(\widetilde{\mathbf{A}} - \mathbf{I})\widetilde{\mathbf{A}}^{\mathbf{n}}\mathbf{v} = \frac{1}{\mathbf{c^{n+1}}}(\mathbf{r_c^{(n+1)}} - \mathbf{r_c^{(n)}}).$$

This relation shows that it is possible to **apply the power method for different values of $\mathbf{c}$ at a low additional cost**.

Indeed, since the vectors $(\widetilde{\mathbf{A}} - \mathbf{I})\widetilde{\mathbf{A}}^{\mathbf{n}}\mathbf{v}$ are **independent of** $\mathbf{c}$, the vectors $\mathbf{r}_{\widetilde{\mathbf{c}}}^{(\mathbf{n})}$ corresponding to a **different value** $\widetilde{\mathbf{c}}$ of the parameter can be directly computed by

$$\begin{aligned} \mathbf{r}_{\widetilde{\mathbf{c}}}^{(\mathbf{0})} &= \mathbf{v} \\ \mathbf{r}_{\widetilde{\mathbf{c}}}^{(\mathbf{n+1})} &= \mathbf{r}_{\widetilde{\mathbf{c}}}^{(\mathbf{n})} + \widetilde{\mathbf{c}}^{\mathbf{n+1}} \frac{1}{\mathbf{c}^{\mathbf{n+1}}} (\mathbf{r}_{\mathbf{c}}^{(\mathbf{n+1})} - \mathbf{r}_{\mathbf{c}}^{(\mathbf{n})}), \quad \mathbf{n} = \mathbf{0}, \mathbf{1}, \ldots \end{aligned}$$

WHAT IS EXTRAPOLATION?:

Assume that the values of a function $\mathbf{f}$ are known at $\mathbf{k}$ points $\mathbf{x_i}$, that is

$$\mathbf{y_i = f(x_i), \quad i = 1, \ldots, k.}$$

Choose a function $\mathbf{F_k}$ belonging to some class of functions, and depending on $\mathbf{k}$ parameters: $\mathbf{F(a_1, \ldots, a_k, \cdot)}$

Compute $\mathbf{a_1^*, \ldots, a_k^*}$ solution of the system of equations

$$\mathbf{F_k(a_1^*, \ldots, a_k*, x_i) = y_i, \quad i = 1, \ldots, k.}$$

$\mathbf{F_k}$ **interpolates** $\mathbf{f}$ at the points $\mathbf{x_i}$.

For $\mathbf{x^* \notin [\min_i x_i, \max_i x_i]}$, compute the **extrapolated** value

$$\mathbf{y^* = F_k(a_1^*, \ldots, a_k^*, x^*).}$$

EXAMPLE: ROMBERG'S METHOD:

$\mathbf{y_i} =$ result obtained by the trapezoidal rule with the step $\mathbf{h_i}$.

Set $\mathbf{x_i} = \mathbf{h_i^2}$.

$\mathbf{F_k} =$ polynomial of degree $\mathbf{k-1}$.

Extrapolate at $\mathbf{x^* = 0}$.

Why is Romberg's method working so well?

Because, by the Euler-Maclaurin formula, the results of the trapezoidal rule behave **like** a polynomial in $\mathbf{h^2}$.

EXTRAPOLATION OF THE PAGERANK VECTORS:

For extrapolation to work well (that is for choosing the class of functions), we have to

analyze the behavior of $\mathbf{r_c}$ with respect to $\mathbf{c}$.

But, from the **rational explicit expressions** we know (Serra-Capizzano (2005)) that $\mathbf{r_c}$ is a **rational function** with a **vector numerator** of degree $p-1$, and a **scalar denominator** of degree $p-1$ in $\mathbf{c}$.

So, the class of functions used for **extrapolation** will be the class of **rational functions** of the same type, but of degree

$$k << p-1.$$

A first account of such extrapolation procedures was given in Brezinski, Redivo-Zaglia, Serra-Capizzano, (2005).

VECTOR RATIONAL EXTRAPOLATION METHOD (VREM)

We **interpolate** the vectors $\mathbf{r_c}$ corresponding to several values of the parameter $\mathbf{c}$ by the **vector rational function**

$$\mathbf{p}(\mathbf{c}) = \frac{\mathbf{P}_k(\mathbf{c})}{Q_k(\mathbf{c})}.$$

The vector coefficients of $\mathbf{P}_k$ and the scalar coefficients of Q_k are obtained by solving the interpolation problem

$$Q_k(\mathbf{c}_i)\mathbf{p}_i = \mathbf{P}_k(\mathbf{c}_i), \qquad i = 0, \ldots, k,$$

with $\mathbf{p}_i = \mathbf{r}_{\mathbf{c}_i}$, and the $\mathbf{c}_i$'s **distinct points** in $]0, 1[$.

For solving this problem we used the **Lagrange's interpolation formula** and the characteristics polynomials $L_i(\mathbf{c})$ (all the details can be found in Brezinski, Redivo-Zaglia (2008)).

Vector rational extrapolation method (VREM)

1. Choose $k+2$ distinct values of $\mathbf{c}$: $\mathbf{c}_0, \ldots, \mathbf{c}_k$ and $\mathbf{c}^*$.
2. Compute $\mathbf{p}_i = \mathbf{r}_{\mathbf{c}_i}$ for $i = 0, \ldots, k$, and $\mathbf{r}_{\mathbf{c}^*}$ (low cost formula).
3. Choose $k+1$ linearly independent vectors $\mathbf{s}_0, \ldots, \mathbf{s}_k$, or take $\mathbf{s}_i = \mathbf{p}_i$ for $i = 0, \ldots, k$.
4. Compute $a_0(\mathbf{c}^*), \ldots, a_k(\mathbf{c}^*)$ by solving the system

$$\sum_{i=0}^{k} (\mathbf{p}_i, \mathbf{s}_j) L_i(\mathbf{c}^*) a_i(\mathbf{c}^*) = (\mathbf{r}_{\mathbf{c}^*}, \mathbf{s}_j), \qquad j = 0, \ldots, k,$$

5. Compute an approximation of $\mathbf{r}_{\mathbf{c}}$ by

$$\mathbf{p}(\mathbf{c}) = \frac{\sum_{i=0}^{k} L_i(\mathbf{c}) a_i(\mathbf{c}^*) \mathbf{p}_i}{\sum_{i=0}^{k} L_i(\mathbf{c}) a_i(\mathbf{c}^*)}.$$

NUMERICAL EXPERIMENTS

$P = (p_{ij})$ is randomly constructed. Dimension p.

First we select a random integer q between 1 and $p/10$.

Then, we generate a random integer vector $\mathbf{m}$ of dimension p with components between 1 and q.

Each **row** i of our matrix P will contain, at most, $\mathbf{m}(i)$ nonzero elements.

Then, we randomly choose, for each i, an integer vector of dimension $\mathbf{m}(i)$, with components between 1 and p, and we eliminate its identical components and those equal to i.

The length of the reduced vector is $\deg(i) \leq \mathbf{m}(i)$, and its components give the **indexes** j of the **columns** such that $p_{ij} = 1/\deg(i)$, all others elements being set to zero.

Finally, among all rows, we randomly set to zero $p/5$ of them, corresponding to the dangling nodes.

Such matrices P (and the corresponding matrices $\widetilde{P}$ and $P_{\mathbf{c}}$) have the **same properties** as those coming out from the web.

A very important point to mention, is that we are not interested in the exact values of the components of the real and extrapolated PageRank vectors, but in their relative values, that is the rank of each of them compared with the other components.

The values and the ranks can be quite sensitive:

- stability of PageRank algorithm (Lempel, Moran, 2005)
- rank-stability (Borodin et al., 2005)
- detailed explanations (Langville, Meyer, 2006)

Example (Ipsen, ANAW 2006, Pisa) :

$$\mathbf{r_c} = (\quad 0.23 \quad 0.24 \quad 0.26 \quad 0.27 \quad)^T$$

$$\text{rank}(\mathbf{r_c}) = \quad 4 \quad 3 \quad 2 \quad 1$$

$$\mathbf{r_1(c)} = (\quad 0.27 \quad 0.26 \quad 0.24 \quad 0.25 \quad)^T$$

$$\text{rank}(\mathbf{r_1(c)}) = \quad 1 \quad 2 \quad 4 \quad 3$$

$\|\mathbf{r_c} - \mathbf{r_1(c)}\|_\infty = 0.04$ (**small error**, but **incorrect ranking**)

$$\mathbf{r_2(c)} = (\quad 0 \quad 0.001 \quad 0.002 \quad 0.997 \quad)^T$$

$$\text{rank}(\mathbf{r_2(c)}) = \quad 4 \quad 3 \quad 2 \quad 1$$

$\|\mathbf{r_c} - \mathbf{r_2(c)}\|_\infty = 0.727$ (**bigger error**, but **correct ranking**)

NOTATIONS

In all the examples, we choose 9 different values for $\mathbf{c}$:
$\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_7$ and $\mathbf{c}^*$ and $\mathbf{w} = \mathbf{v}$.

VREM n $\longrightarrow$ Vector rational extrapolation with
$\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{\mathbf{n}-2}$ and $\mathbf{c}^*$.

nch $\longrightarrow$ total number of changes in the ranking
between the Pagerank vector $\mathbf{r_c}$ and
the extrapolated vector $\mathbf{p(c)}$.

ich $\longrightarrow$ rank of the first change occurred
after sorting by descending values $\mathbf{r_c}$ and $\mathbf{p(c)}$.

$\mathbf{d}_{\max}$ $\longrightarrow$ maximum displacement of a page.
A positive value of $d_{\max}$ means that the
corresponding page went up in the list,
and that it went down if it is negative.

$ix_{\max}$, $iy_{\max}$ $\longrightarrow$ The ranks of the page corresponding to $\mathbf{d}_{\max}$ ($ix_{\max}$ in the sorted $\mathbf{r_c}$, $iy_{\max}$ in the sorted $\mathbf{p(c)}$).

It seems that the most two important parameters to consider are $\mathbf{d}_{\max}$ and **ich**.

$\mathbf{d}_{\max}$ indicates the **size of the largest change in the ranking**.
The **smallest** $\mathbf{d}_{\max}$, the **better the ranking**.
So, a criterion of good quality is to have a small value of $\mathbf{d}_{\max}$.
But $\mathbf{d}_{\max}$ can be **large** if **ich** is also **large**.

In fact, **ich** indicates the **location of the first change** in the ranking. So, a **correct ranking** has been obtained for the **ich-1 first components of the extrapolated vector**.

It is **not so important** to have many changes **(ich large)** in the ranking if they are small, that is if $\mathbf{d}_{\max}$ **is small**.

First example

$p = 5000, nnz = 942806.$

Google parameter $\mathbf{c} = 0.85$.

8 iterations with power method for a precision of 10^{-8}.

The highest and the smallest components of the PageRank vector were $3.84636884 \cdot 10^{-4}$ and $1.48826460 \cdot 10^{-4}$, respectively, thus meaning that, when p is large, many components can differ only in the last digits.

#	Method	$\|\mathbf{r}_c - \mathbf{p}\|_\infty$	$\|\mathbf{r}_c - \mathbf{p}\|_1/p$	nch	ich	$\mathbf{d}_{\max}$	$ix_{\max}$	$iy_{\max}$
1	VREM 4	2.43e-6	2.57e-8	4417	18	-47	1553	1600
2	VREM 5	5.13e-8	3.32e-9	1667	29	6	2651	2645
3	VREM 6	6.03e-8	2.34e-9	1254	190	-4	2358	2362
4	**VREM 7**	2.77e-8	1.24e-9	**689**	**190**	**2**	890	888
5	VREM 8	3.04e-8	1.89e-9	1029	190	-4	2358	2362
6	VREM 9	2.87e-8	1.74e-9	939	190	4	2765	2761

$p = 5000$, $\mathbf{c} = 0.85$ (8 iterations)

$\mathbf{c}_i = 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45,$ $\mathbf{c}^* = 0.5$ (5 iterations)

Interchanging $\mathbf{c}_3 = 0.25$ and $\mathbf{c}^* = 0.5$ does not change much the results. Best method for both tests seems to be **VREM 7**. Here **VREM 5** give comparable results.

#	Method	$\|\mathbf{r}_c - \mathbf{p}\|_\infty$	$\|\mathbf{r}_c - \mathbf{p}\|_{1/p}$	**nch**	**ich**	$\mathbf{d}_{\max}$	$ix_{\max}$	$iy_{\max}$
1	VREM 4	2.43e-6	2.57e-8	4420	18	-47	1553	1600
2	**VREM 5**	4.31e-8	2.01e-9	**1102**	**190**	**-4**	2358	2362
3	VREM 6	3.07e-8	1.86e-9	1010	190	-4	2358	2362
4	**VREM 7**	2.50e-8	1.54e-9	**827**	**190**	**4**	2765	2761
5	VREM 8	3.10e-8	1.90e-9	1033	190	-4	2358	2362
6	VREM 9	9.59e-7	2.71e-8	4520	10	-57	2710	2767

Same matrix with $\mathbf{c}_i$'s closer to 0.85:

$\mathbf{c}_i = 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,$ 0.65 (6 iterations), $\mathbf{c}^* = 0.25$

$\mathbf{c} = 0.85$ (8 iterations)

#	Method	$\|\mathbf{r}_c - \mathbf{p}\|_\infty$	$\|\mathbf{r}_c - \mathbf{p}\|_1/p$	**nch**	**ich**	$\mathbf{d}_{\max}$	$ix_{\max}$	$iy_{\max}$
1	VREM 4	9.30e-7	1.31e-8	3788	18	-16	2461	2477
2	VREM 5	2.01e-8	1.17e-9	635	207	3	2765	2762
3	VREM 6	1.14e-8	6.92e-10	385	251	2	936	934
4	**VREM 7**	2.65e-9	1.29e-10	**66**	**272**	**-1**	272	273
5	VREM 8	3.16e-9	2.02e-10	114	272	-1	272	273
6	VREM 9	2.07e-9	1.25e-10	66	272	-1	272	273

Example 2: Stanford web matrix

$p = 281903, nnz = 2312497,$ $\mathbf{c} = 0.85$ (91 iterations)

$\mathbf{c}_i = 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45,$ $\mathbf{c}^* = 0.5$ (22 iterations)

#	Method	$\|\mathbf{r}_c - \mathbf{p}\|_\infty$	$\|\mathbf{r}_c - \mathbf{p}\|_1/p$	nch	ich	$\mathbf{d}_{\max}$	$ix_{\max}$	$iy_{\max}$
1	VREM 4	1.22e-3	6.26e-7	261573	4	-162408	26841	189249
2	VREM 5	1.78e-3	1.40e-7	261445	4	-105526	19635	125161
3	VREM 6	7.67e-4	1.02e-7	261208	4	-89744	52409	142153
4	VREM 7	4.52e-4	7.50e-8	260291	4	-44139	32553	76692
5	VREM 8	3.00e-4	5.25e-8	260629	4	-52413	116455	168868
6	**VREM 9**	2.57e-4	6.93e-8	**281652**	**11**	**-219944**	61958	281902

Then, we will consider extrapolation with larger values.

$p = 281903, nnz = 2312497,\ \mathbf{c} = 0.85$ (91 iterations)

$\mathbf{c}_i = 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65$ (39 iterations), $\mathbf{c}^* = 0.25$

#	Method	$\|\mathbf{r}_c - \mathbf{p}\|_\infty$	$\|\mathbf{r}_c - \mathbf{p}\|_1/p$	nch	ich	$\mathbf{d}_{\max}$	$ix_{\max}$	$iy_{\max}$
1	VREM 4	1.05e-3	4.15e-7	261425	4	-104574	26841	131415
2	VREM 5	5.26e-4	9.61e-8	261240	4	-54793	32553	87346
3	VREM 6	4.02e-4	5.95e-8	260085	7	-37547	32553	70100
4	VREM 7	9.55e-5	1.45e-8	258487	14	-23600	32553	56153
5	**VREM 8**	3.32e-5	7.33e-9	**257896**	**29**	**-20639**	32553	53192
6	VREM 9	1.03e-5	2.98e-9	254360	14	-13364	38289	51653

REFERENCES FOR THE PART I

- P. Boldi, M. Santini, S. Vigna, *PageRank as a function of the damping factor*, Poster Proceedings of the 14th International World Wide Web Conference, May 10-14, 2005, Chiba, Japan.
- C. Brezinski, M. Redivo-Zaglia, S. Serra-Capizzano, *Extrapolation methods for PageRank computations*, C.R. Acad. Sci. Paris, Sér. I, 340 (2005) 393–397.
- C. Brezinski, M. Redivo-Zaglia, *The PageRank vector: properties, computation, approximation, and acceleration*, SIAM J. Matrix Anal. Appl., 28 (2006) 551–575.
- C. Brezinski, M. Redivo-Zaglia, *Rational extrapolation for the PageRank vector*, Math. Comput., 77 (2008) 1585–1598.

- S. Brin, L. Page, *The anatomy of a large–scale hypertextual web search engine*, Comput. Networks ISDN Syst., 30 (1998) 107-117.
- T. Haveliwala, S. Kamvar, D. Klein, C. Manning, G.H. Golub *Computing PageRank using power extrapolation*, Stanford University Technical Report, July 2003.
- S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, *Extrapolations methods for accelerating PageRank computations*, in Proceedings of the Twelfth International World Wide Web Conference, ACM Press, New York, 2003, 261-270.
- S. Serra-Capizzano, *Jordan canonical form of the Google matrix: a potential contribution to the PageRank computation*, SIAM J. Matrix Anal. Appl., 27 (2005) 305-312.

- P. Wynn, *On a device for computing the $e_m(S_n)$ transformation*, , MTAC, 10 (1956) 91-96.
- P. Wynn, *Acceleration techniques for iterated vector and matrix problems*, Math. Comput., 16 (1962) 301-322.

Part II

in collaboration with **Stefano Cipolla, University of Southampton (UK), Francesco Tudisco, University of Edinburgh (UK)**

- **Higher-order Markov Chains**
- **Multilinear PageRank (MPR)**
- **Algorithms for MPR: SS-HOPM, Inner-Outer**
- **The simplified Topological ε-algorithm.**
- **Numerical Results for extrapolated algorithms**

Higher-order Markov Chains

Definition (m-th order, n-state Markov Chain)

Stochastic process $(X_t)_{t=1,2,\ldots}$ such that

$$\mathbb{P}(X_t = i_1 | X_{t-1} = i_2, \cdots, X_1 = i_t) =$$
$$\mathbb{P}(X_t = i_1 | X_{t-1} = i_2, \cdots, X_{t-m} = i_{m+1}),$$

with $i_k \in \{1, \ldots, n\}$.

More precise predictive value in many applications: e-mail communication, web browsing behavior, network clustering etc...

A Markov Chain with memory m can be conveniently represented by a **order-$(m+1)$ tensor** $\mathcal{P} = [p_{i_1, i_2 \ldots, i_{m+1}}]$

$$p_{i_1, i_2 \ldots, i_{m+1}} := \mathbb{P}(X_t = i_1 | X_{t-1} = i_2, \cdots, X_{t-m} = i_{m+1}) \; \forall \; t;$$

$p_{i_1, i_2 \ldots, i_{m+1}} \geq 0$ and $\sum_{i_1=1}^{n} p_{i_1, i_2 \ldots, i_{m+1}} = 1$ for all $(i_2, \ldots, i_{m+1})$.

"As it can be easily understood, a critical ingredient in a Markov Chain with memory m is the joint probability of the variables $(X_{t-1}, \cdots, X_{t-m})$ and its evolution (Wu and Chu, 2017)"

The **Joint probability distribution**: is a **order-m tensor** $S^{(t-1)} = [s^{(t-1)}_{i_2 i_3 \ldots i_{m+1}}]$

$$s^{(t-1)}_{i_2 i_3 \ldots i_{m+1}} := \underbrace{\mathbb{P}(X_{t-1} = i_2, X_{t-2} = i_3, \ldots X_{t-m} = i_{m+1})}_{\text{Joint Probability Distribution}};$$

$s^{(t-1)}_{i_2 i_3 \ldots i_{m+1}} \geq 0$ and $\sum_{i_2 i_3 \ldots i_{m+1}} s^{(t-1)}_{i_2 i_3 \ldots i_{m+1}} = 1.$

Higher Stationary Distribution Problem ($m = 2$)

Find $S \in \mathbb{R}^{n \times n}$ s.t. $S = \mathcal{P}S$, where $(\mathcal{P}S)_{i_1 i_2} = \sum_{i_3} p_{i_1 i_2 i_3} S_{i_2 i_3}$.

For avoiding the $O(n^2)$ space complexity in solving this problem, Li and Ng (2014) using a Rank-1 Approximation of S

Assuming that the stationary distribution is symmetric and of rank one, that is

$$S = \mathbf{s}\mathbf{s}^T (\mathbf{s} \text{ stochastic}),$$

the problem reduces to solve the following Z-eigenvalue problem, having $O(n)$ space complexity

$$\mathcal{P}\mathbf{s}^2 = \mathbf{s}$$

Higher-Order PageRank vs Multilinear Pagerank, $m = 2$

$\mathcal{P} \in \mathbb{R}^{n\times n\times n}$ transition tensor of a 2-nd order Markov Chain,
$\mathcal{V}$ rank-one tensor: $\mathcal{V}_{i_1 i_2 i_3} = v_{i_1}$ $\mathbf{v}$ being a stochastic vector,
$\alpha \in (0, 1)$.

$$\mathcal{A} := \alpha\mathcal{P} + (1 - \alpha)\mathcal{V}.$$

Definition (Higher-Order PageRank)

Find $S \in \mathbb{R}^{n\times n}$, $\sum_{i_1 i_2}^{n} s_{i_1 i_2} = 1$, $s_{i_1 i_2} \geq 0$ for all i_1, i_2, such that

$$\mathcal{A}S = S$$

(rewrite the problem as a $n^2 \times n^2$ linear system and use Perron-Frobenius theory).

Definition (Multilinear PageRank, Gleich et al., 2015)

Find $\mathbf{s} \in \Omega := \{\mathbf{s} \in \mathbb{R}^n | \|\mathbf{s}\|_1 = 1, s_i \geq 0 \text{ for } i = 1, \ldots, n\}$ s.t.

Stationary Distribution Form: $\mathcal{A}\mathbf{s}^2 = \mathbf{s}$, or equivalently,

$(\alpha, \mathcal{P}, \mathbf{s})$ MPR problem: $\begin{cases} \alpha\mathcal{P}\mathbf{s}^2 + (1 - \alpha)\mathbf{v} = \mathbf{s}, \\ \alpha P(\mathbf{s} \otimes \mathbf{s}) + (1 - \alpha)\mathbf{v} = \mathbf{s} \end{cases}$

where $P \in \mathbb{R}^{n\times n^2}$ is a stochastic unfolding.

SS-HOPM

In **2011, Kolda and Mayo** proposed the Shifted Symmetric Higher-Order Power Method (SS-HOPM), a generalization of the symmetric higher-order power method. They reformulated the fixed point equation $\mathbf{s} = \alpha\mathcal{P}\mathbf{s}^2 + (1-\alpha)\mathbf{v}$ like

$$\mathbf{s}_{\ell+1} = \frac{\alpha}{1+\gamma}\mathcal{P}\mathbf{s}_\ell^2 + \frac{1-\alpha}{1+\gamma}\mathbf{v} + \frac{\gamma}{1+\gamma}\mathbf{s}_\ell$$

Inner-Outer iteration

In **2015, Gleich et al** reformulated the fixed point equation $\mathbf{s} = \mathcal{A}\mathbf{s}^2$ like

$$\mathbf{s}_{\ell+1} = \frac{\alpha}{2}\mathcal{A}\mathbf{s}_{\ell+1}^2 + (1-\frac{\alpha}{2})\mathbf{s}_\ell$$

Each step of the Inner-Outer method requires the solution of $(\frac{\alpha}{2}, \mathcal{A}, \mathbf{s}_\ell)$ MPR problem.

Theorem: If $\mathbf{s}_0$ is a stochastic vector, bothe methods generate stochastic iterations and, if $\alpha < 1/2$, then they converges to the unique solution $\mathbf{s}$ of the Multilinear PageRank.

In their paper, Kolda and Mayo wrote **"Can the convergence rate of the SS-HOPM be accelerated?"**

The answer is . . . **YES, also that of Inner-Outer method!**

The simplified topological ϵ-algorithms (STEA)

In **1975, Brezinski** proposed two new transformations, that can be used for any sequence of elements of a general vector space E (scalars, vectors, matrices or also tensors), and the related algorithms. They were greatly simplified in **Brezinski, Redivo-Zaglia (2014)** The new implementations have the great advantage to reduce the memory requirements, to simplify the formlulæ and, moreover, to use the linear functional $\mathbf{y}$ only in the initializations. A public domain Matlab package EPSfun have been published in **2017**.

For instance, in the **STEA2** the terms $\tilde{\varepsilon}_{2k}^{(n)} = \tilde{\mathbf{e}}_k(\mathbf{s}_n)$ are obtained by the rule

$$\tilde{\varepsilon}_{2k+2}^{(n)} = \tilde{\varepsilon}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}} (\tilde{\varepsilon}_{2k}^{(n+2)} - \tilde{\varepsilon}_{2k}^{(n+1)})$$

where $\tilde{\varepsilon}_0^{(n)} = \mathbf{s}_n \in E$ and the scalar quantities are computed by the scalar ε-algorithm **(Wynn, 1956)** applied to $s_n = \langle \mathbf{y}, \mathbf{s}_n \rangle$.

SS-HOPM or Inner-Outer + Extrapolation Restarted Form

Algorithm 1: Restarted Method

Data: Choose $2k$ and $\mathbf{x}_0$

for $i = 0, 1, \ldots,$ *cycles* **do**
  Set $\mathbf{s}_0 = \mathbf{x}_i$
  for $\ell = 1, \ldots, 2k$ **do**
    Compute $\mathbf{s}_\ell = F(\mathbf{s}_{\ell-1})$
  end
  Apply STEA to $\mathbf{s}_0, \ldots \mathbf{s}_{2k}$
  Set $\mathbf{x}_{i+1} = \tilde{\varepsilon}_{2k}^{(0)}$
end

Extrapolated SS-HOPM:

$$F(\mathbf{s}_{\ell-1}) := \frac{\alpha}{1+\gamma}\mathcal{P}\mathbf{s}_{\ell-1}^2 + \frac{1-\alpha}{1+\gamma}\mathbf{v} + \frac{\gamma}{1+\gamma}\mathbf{s}_{\ell-1}.$$

Extrapolated Inner-Outer:

$$F(\mathbf{s}_{\ell-1}) := \text{ unique solution to the } (\frac{\alpha}{2}, \mathcal{A}, \mathbf{s}_{\ell-1}) \text{ MPR problem.}$$

Numerical Results

Problem Set n.1: 29 Problems $3 \times 3 \times 3$, $4 \times 4 \times 4$, $6 \times 6 \times 6$

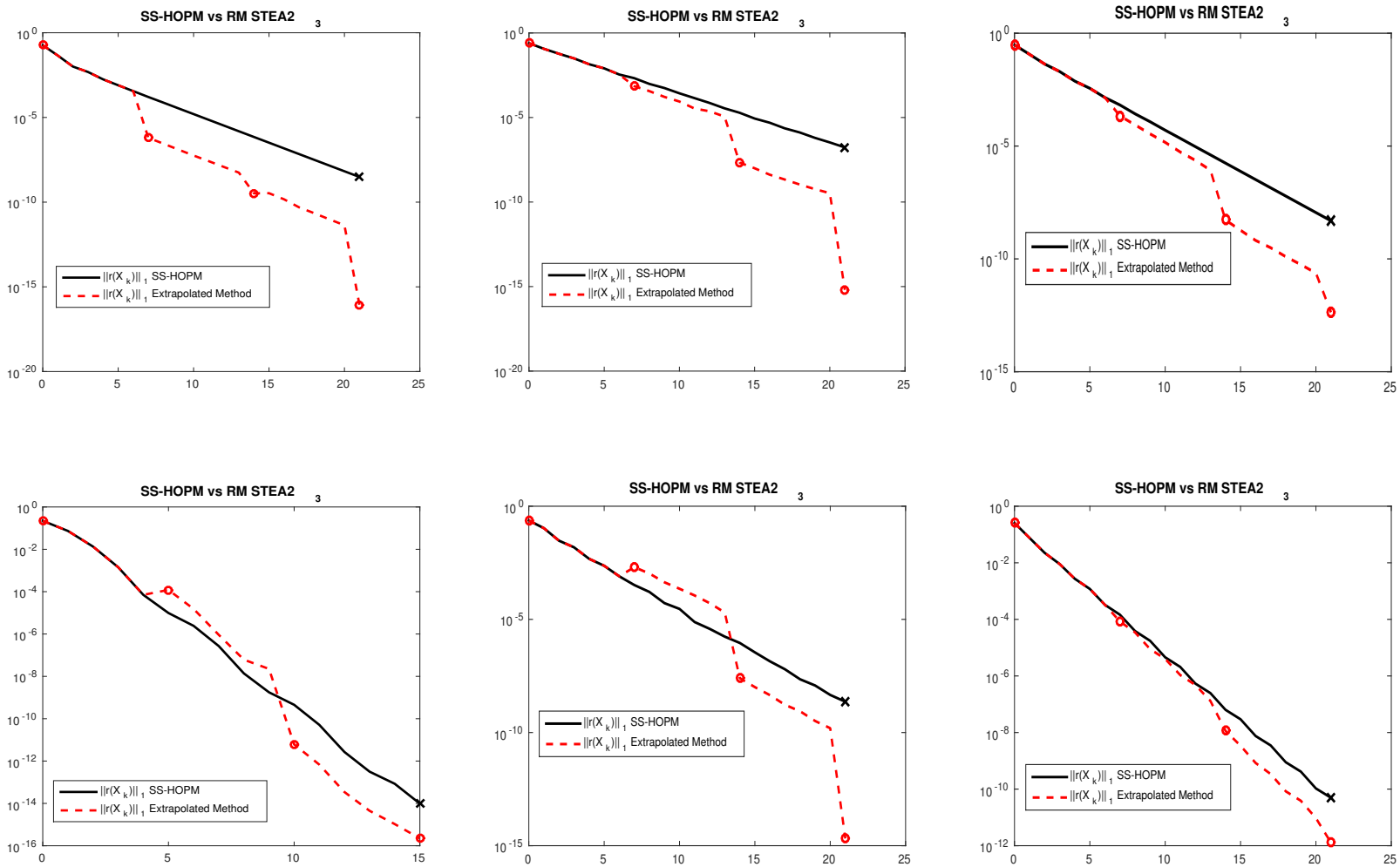
(Gleich et al, 2015)

$$\boldsymbol{R}_{6,3} = \left[\begin{array}{cccccc|cccccc|cccccc|cccccc|cccccc|cccccc} 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&1&0&0&1&1&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&1&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&0 \\ 1&1&1&1&1&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \end{array}\right]$$

$$\boldsymbol{R}_{6,4} = \left[\begin{array}{cccccc|cccccc|cccccc|cccccc|cccccc|cccccc} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&1&1&1&0&0&0&0&1&1&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&1&0&1&0&1&0 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&0&0&0&1&0&1&0&1&1&1&1&0&0&0&1&1&1&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0 \\ 1&1&1&1&1&1&1&1&0&1&0&1&0&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1 \end{array}\right]$$

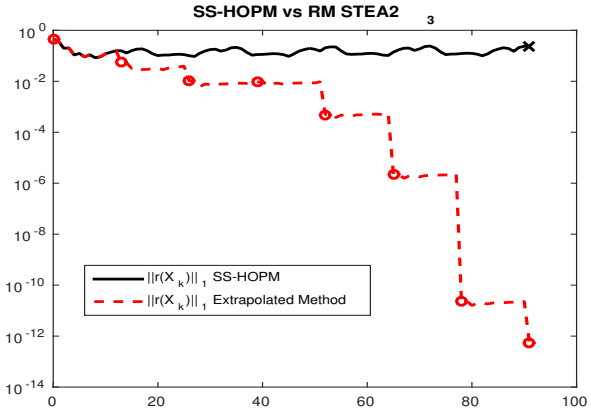
$$\boldsymbol{R}_{6,5} = \left[\begin{array}{cccccc|cccccc|cccccc|cccccc|cccccc|cccccc} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&1&0&0&1&1&0&1&0&1&0&0&1&0&0&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&1&1&1&0&0&0&0&0 \\ 1&1&1&1&1&1&1&0&0&0&1&1&0&1&0&1&0&0&1&1&1&1&0&1&0&0&1&0&1&0&0&0&0&0&0&1 \end{array}\right]$$

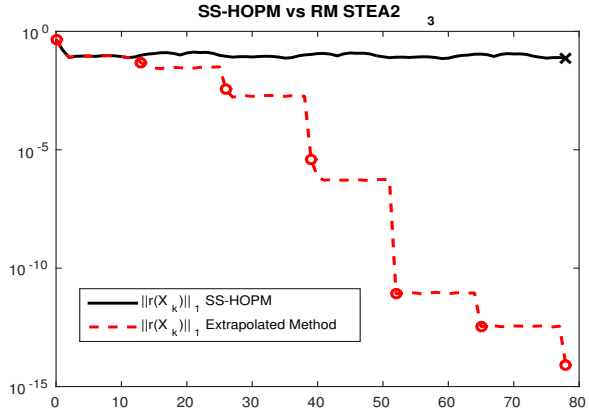
Problem Set n.1: SS-HOPM $\alpha < 1/2$

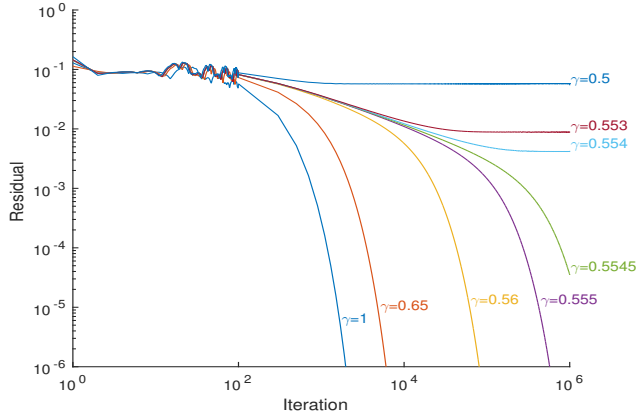


$\alpha = 0.499, \gamma = 0$. **Upper row: best performances obtained on problems** $R3_1,\ R4_4,\ R6_2$. **Lower row: worst performances obtained on problems** $R3_3,\ R4_{19},\ R6_3$.

Problem Set n.1: SS-HOPM $\alpha = 0.99$, Problem $R4_{19}$







Problem $R4_{19}$**,** $\alpha = 0.99$**.** $\gamma = 0.1$ **(left),** $\gamma = 0.5$ **(right).**
Bottom results from Gleich et al

Problem Set n.1: SS-HOPM $\alpha = 0.99$, $\gamma = 1$. Solved Problems

"Solved" if:

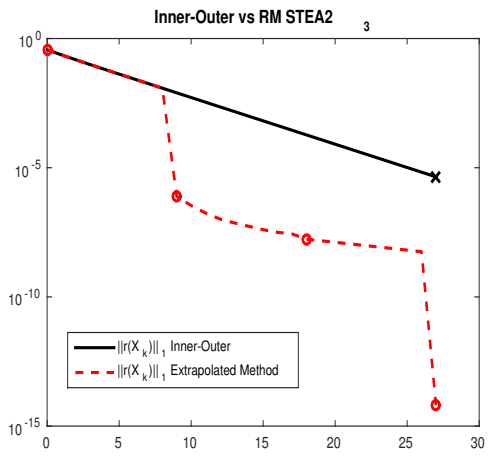
$$\|\alpha \mathcal{P} \mathbf{s}_{final}^{m-1} + (1-\alpha)\mathbf{v} - \mathbf{s}_{final}\|_1 \leq 10^{-8}$$

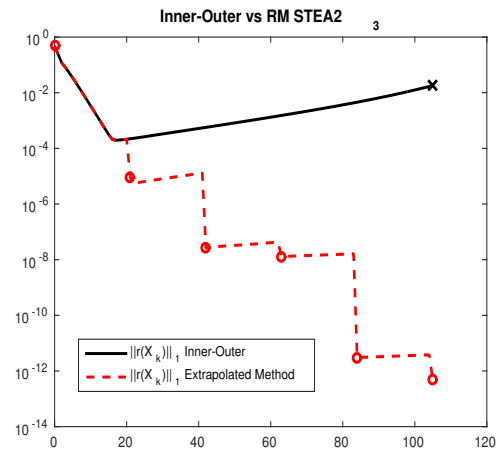
Table: Our performances

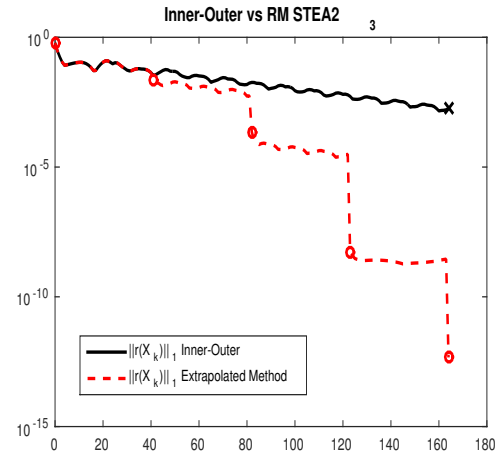
	Solved Problems
$n = 3$	$5/5$
$n = 4$	$11/19$
$n = 6$	$2/5$
Total	**18/29**

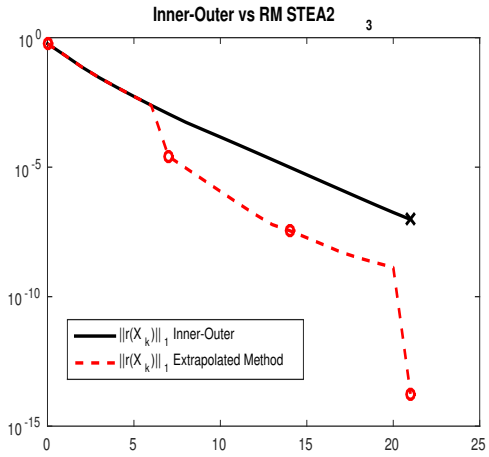
Gleich et al Solved problems: 10/29.

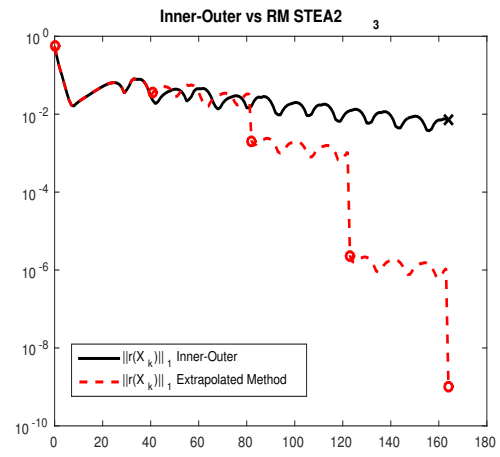
Problem Set n.1: Inner-Outer $\alpha = 0.99$. Solved 28/29 vs 26/29

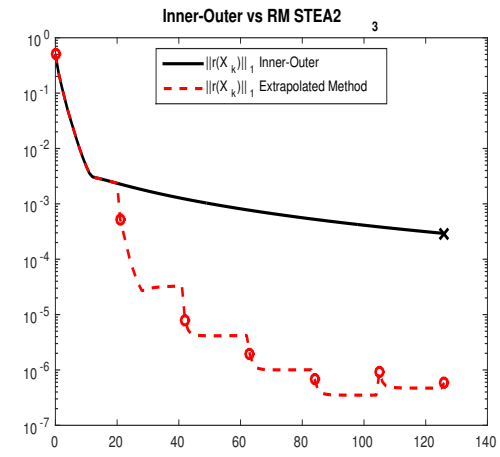












$\alpha = 0.99$. **Upper row: best performances obtained on problems** $R3_1$, $R4_{12}$, $R6_2$. **Lower row: worst performances obtained on problems** $R3_4$, $R4_{12}$, $R6_3$ **(not solved)**

Problem Set n.2: how stochastic tensor are generated

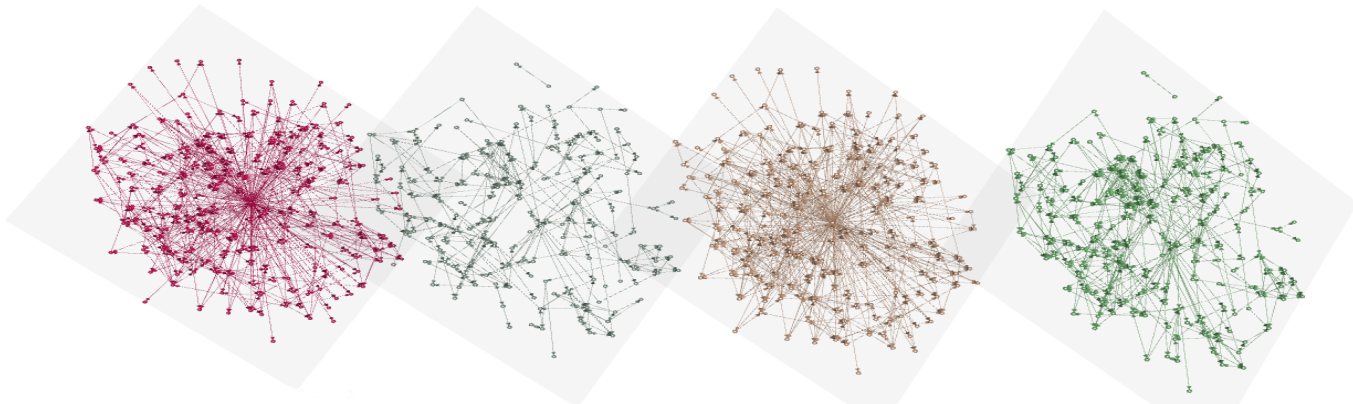
We use CONTEST **(Taylor and D. Higham, 2009)** to generate random graphs!

Algorithm 2: Stochastic Tensor generator

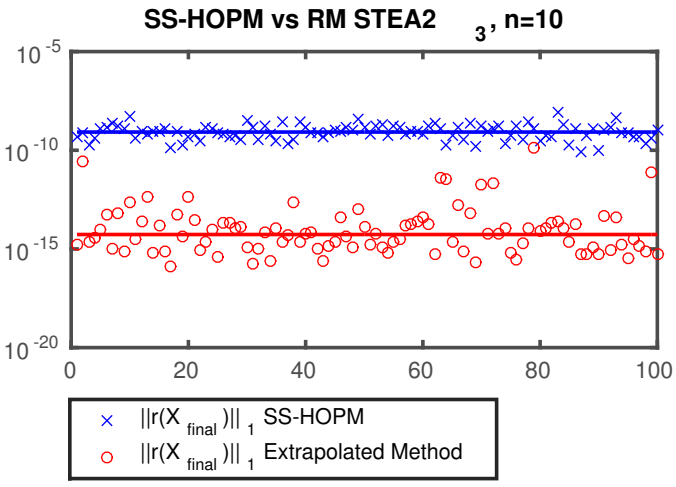
Data: n (size of the tensor),
$\mathbf{m} = \{smallw(n), gilbert(n), erdrey(n), pref(n), geo(n), lockandkey(n), rank1(n)\}$

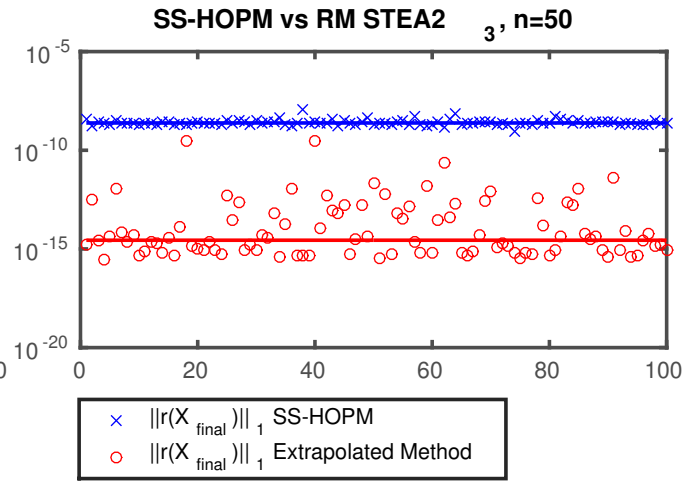
1 **for** *i=1:n* **do**
2 | Choose randomly an element $\mathbf{m}_r$ of $\mathbf{m}$;
3 | Set $\mathcal{P}_{:,:,i} = \mathbf{m}_r$;
4 | Transform $\mathcal{P}_{:,:,i}$ into a stochastic matrix ;
5 **end**

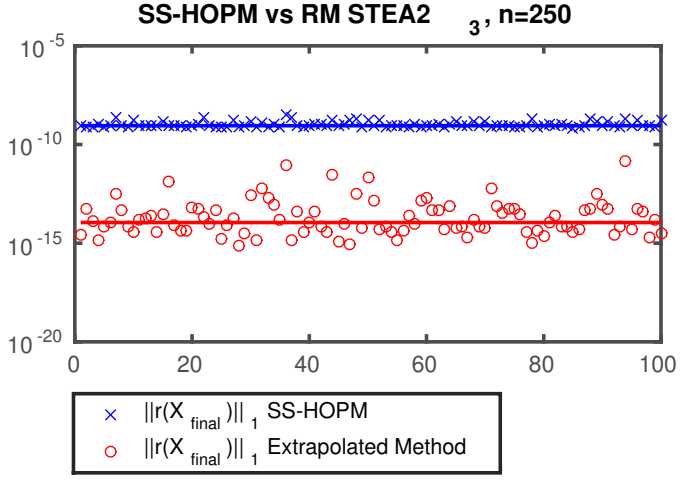
We considered 100 **tensors obtained by this Algorithm.**



Problem Set n.2: SS-HOPM $\alpha < 1/2$

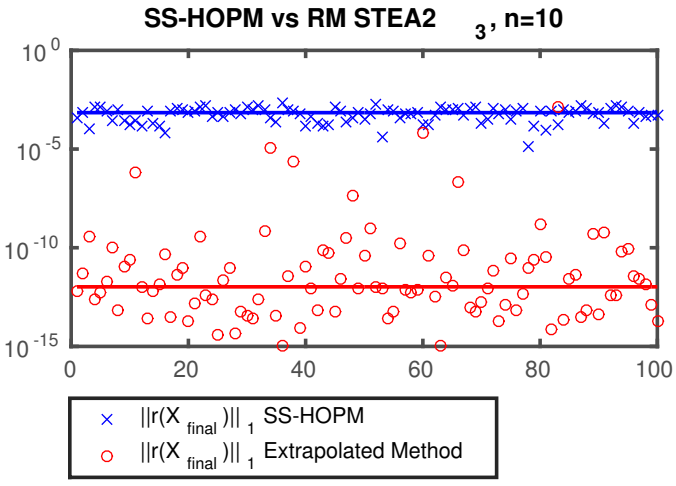


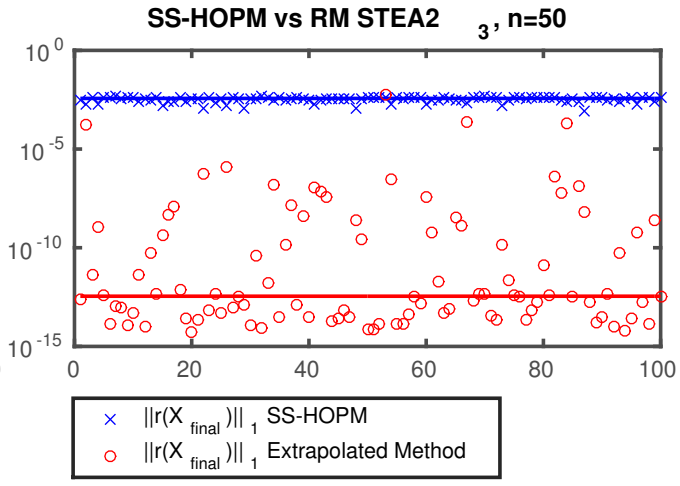


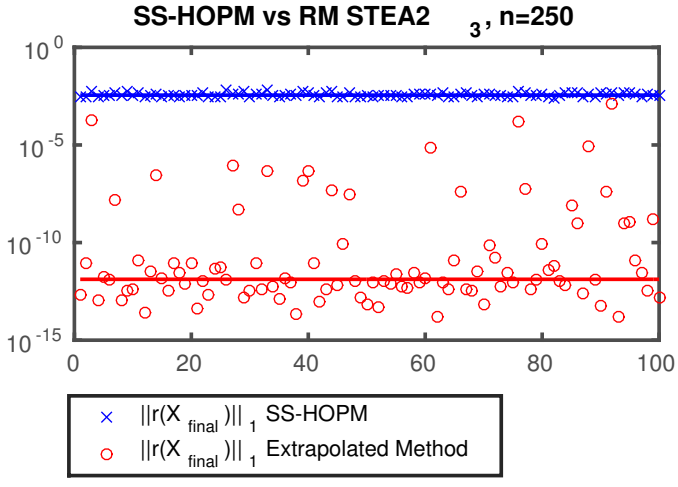


$\alpha = 0.499,\ \gamma = 0,\ 2k = 10,\ \mathit{cycles} = 2.$

Problem Set n.2: SS-HOPM, $\alpha = 0.99$

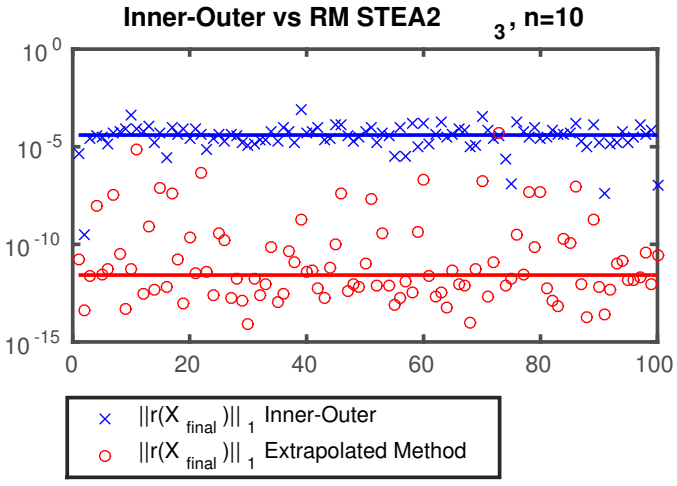


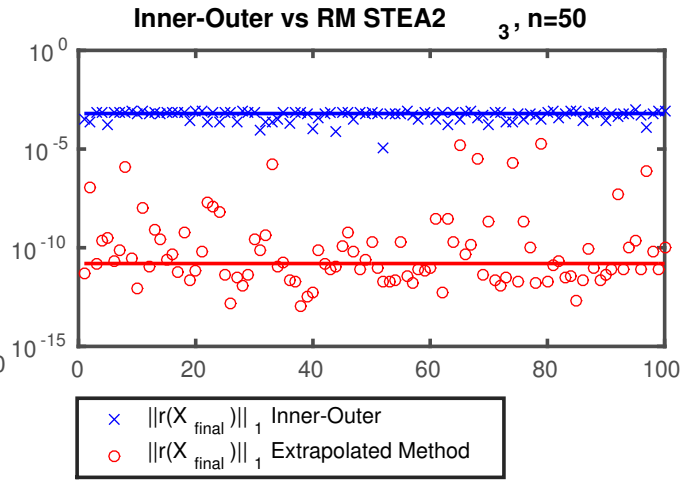


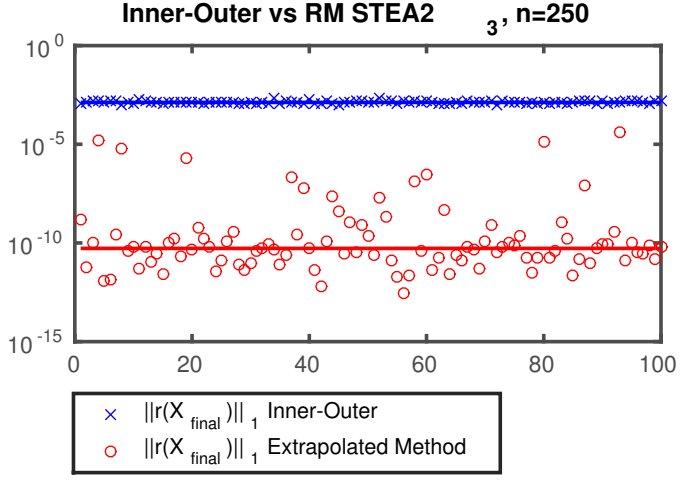


$\alpha = 0.99,\ \gamma = 1,\ 2k = 32,\ \mathit{cycles} = 4.$

Problem Set n.2: Inner-Outer, $\alpha = 0.99$







$\alpha = 0.99,\ 2k = 32,\ \mathit{cycles} = 4.$

References for the Part II

- C. Brezinski, *Généralisation de la transformation de Shanks, de la table de Padé et de l'ε-algorithms*, Calcolo, 12 (1975), pp. 317-360.
- C. Brezinski, M. Redivo-Zaglia, *The simplified topological ε-algorithms for accelerating sequences in a vector space*, SIAM J. Sci. Comput 36(5) (2014) 2227-2247.
- C. Brezinski, M. Redivo-Zaglia, *The simplified topological ε-algorithms: software and applications*, Numer. Algorithms 74(4) (2017) 1237-1260.
- S. Cipolla, M. Redivo-Zaglia, F. Tudisco, *Extrapolation Methods for fixed point multilinear PageRank computations*, Linear Algebra Appl. 27 (2020) e2280, 22pp.

- D.F. Gleich, , L.H. Lim and Y. Yu, *Multilinear pagerank*, SIAM J. Matrix Anal. Appl. 36(4) (2015) 1507-1541.
- T.G. Kolda, J.R. Mayo, *Shifted power method for computing tensor eigenpairs*, SIAM J. Matrix Anal. Appl. 32(4) (2011) 1095-1124.
- W. Li and M.K. Ng, *On the limiting probability distribution of a transition probability tensor*, 62(3) (2014) 362-385.
- A. Taylor, D.J. Higham, *CONTEST: A controllable test matrix toolbox for MATLAB*, ACM Transactions on Mathematical Software (TOMS) 35(4) (2009) 26.