

---

# The PageRank problem: extrapolation techniques

Michela REDIVO-ZAGLIA  
University of Padova - Italy

joint work with

Claude BREZINSKI  
University of Lille - France

- 
- The power method for the PageRank vector
  - Extrapolation techniques
  - Numerical experiments
  - New ideas for future works
  - Final remarks

---

## THE POWER METHOD FOR THE PAGERANK VECTOR

### Notations:

$\tilde{\mathbf{P}}$  is the stochastic reducible matrix, without dangling nodes, related to the Google matrix  $\mathbf{P} \in \mathbb{R}^{p \times p}$ .

$\mathbf{P}_c = c\tilde{\mathbf{P}} + (1 - c)\mathbf{E}$  is a stochastic and irreducible matrix where  $c \in [0, 1)$ ,  $\mathbf{E} = \mathbf{e}\mathbf{v}^T$ ,  $\mathbf{e} = (1, \dots, 1)^T$  and  $\mathbf{v} = \mathbf{e}/p$ .

As we saw, the power method

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

**converges** to the **unique** vector

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

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

---

The speed of convergence is  $\mathcal{O}(c^n)$ . For values  $c$  far away from 1, it is **fast** and **cheap**.

**Google** chooses  $c = 0.85$ , and we have

$n$	$c^n$
10	1.97e-01
20	3.88e-02
50	2.96e-04
80	2.26e-06
100	8.75e-08

For **acceleration** of the power method, see

- Kamvar, Haveliwala, Manning, Golub (2003)
- Haveliwala, Kamvar, Klein, Manning, Golub (2003)
- Brezinski, Redivo-Zaglia (2006)

---

## EXTRAPOLATION

We want to **compute**  $\mathbf{r}_c$  for a certain value of  $c$  (0.85 or closer to 1).

We choose **several** (**smaller**) values  $c_i$  of the parameter, and we compute the corresponding vectors  $\mathbf{r}_{c_i}$ .

We **interpolate** these vectors by some function of the parameter, and then we **extrapolate** the results at the desired  $c$ .

**This procedure only costs the number of iterations needed for  $\max_i c_i$ .**

**Why?**

---

We set  $\tilde{\mathbf{A}} = \tilde{\mathbf{P}}^T$  and  $\mathbf{A}_c = \mathbf{P}_c^T$ . Thus  $\mathbf{r}_c = \mathbf{A}_c \mathbf{r}_c$ .

Boldi, Santini, Vigna (2005) proved that

$$\mathbf{r}_c = \mathbf{v} + \mathbf{c}(\tilde{\mathbf{A}} - \mathbf{I}) \sum_{i=0}^{\infty} \mathbf{c}^i \tilde{\mathbf{A}}^i \mathbf{v},$$

and that the **power method produces the partial sums of this series**, that is

$$\begin{aligned} \mathbf{r}_c^{(n+1)} &= \mathbf{v} + \mathbf{c}(\tilde{\mathbf{A}} - \mathbf{I}) \sum_{i=0}^n \mathbf{c}^i \tilde{\mathbf{A}}^i \mathbf{v} \\ &= \mathbf{r}_c^{(n)} + \mathbf{c}^{n+1} (\tilde{\mathbf{A}} - \mathbf{I}) \tilde{\mathbf{A}}^n \mathbf{v}, \quad \mathbf{n} = 0, 1, \dots \end{aligned}$$

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

$$(\tilde{\mathbf{A}} - \mathbf{I}) \tilde{\mathbf{A}}^n \mathbf{v} = \frac{1}{\mathbf{c}^{n+1}} (\mathbf{r}_c^{(n+1)} - \mathbf{r}_c^{(n)}).$$

---

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

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

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

---

## WHAT IS EXTRAPOLATION?:

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

$$y_i = f(\mathbf{x}_i), \quad i = 1, \dots, k.$$

Choose a function  $\mathbf{F}_k$  belonging to some class of functions, and depending on  $k$  parameters:  $\mathbf{F}_k(\mathbf{a}_1, \dots, \mathbf{a}_k, \cdot)$

Compute  $\mathbf{a}_1^*, \dots, \mathbf{a}_k^*$  solution of the system of equations

$$\mathbf{F}_k(\mathbf{a}_1^*, \dots, \mathbf{a}_k^*, \mathbf{x}_i) = y_i, \quad i = 1, \dots, k.$$

$\mathbf{F}_k$  interpolates  $f$  at the points  $\mathbf{x}_i$ .

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

$$\mathbf{y}^* = \mathbf{F}_k(\mathbf{a}_1^*, \dots, \mathbf{a}_k^*, \mathbf{x}^*).$$

---

**EXAMPLE: ROMBERG'S METHOD:**

$y_i$  = result obtained by the trapezoidal rule with the step  $h_i$ .

Set  $x_i = h_i^2$ .

$F_k$  = polynomial of degree  $k - 1$ .

**Extrapolate** at  $x^* = 0$ .

**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  $h^2$ .

---

## EXTRAPOLATION OF THE PAGERANK VECTORS:

So, 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  $c$ .

Let

$\tilde{\lambda}_1 = 1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_p$  the **eigenvalues** of  $\tilde{\mathbf{P}}$  ( $1 \geq |\tilde{\lambda}_2| \geq \dots \geq |\tilde{\lambda}_p|$ )  
 $\mathbf{e}, \mathbf{x}_2, \dots, \mathbf{x}_p$  its corresponding **right eigenvectors** and  
 $\mathbf{y}, \mathbf{y}_2, \dots, \mathbf{y}_p$  its corresponding **left eigenvectors**  
( $\mathbf{y}$  is **one** of the PageRank vectors corresponding to  $c = 1$ )  
 $\tilde{\mathbf{r}} = \lim_{c \rightarrow \infty} \mathbf{r}_c$

---

As proved by Serra-Capizzano (2005) - Horn, Serra-Capizzano

→ if  $\tilde{\mathbf{P}}$  is **diagonalizable**

$$\mathbf{r}_c = \mathbf{y} + (1 - c) \sum_{i=2}^p \frac{\alpha_i}{1 - c\tilde{\lambda}_i} \mathbf{y}_i, \quad \alpha_i = \mathbf{x}_i^T \mathbf{v}$$

→ in the **general case**

$$\mathbf{r}_c = \mathbf{y} + \sum_{i=2}^p \mathbf{w}_i(c) \mathbf{y}_i \quad \text{with}$$

$$\mathbf{w}_2(c) = (1 - c)\alpha_2 / (1 - c\tilde{\lambda}_2)$$

$$\mathbf{w}_i(c) = [(1 - c)\alpha_i + c\beta_i \mathbf{w}_{i-1}(c)] / (1 - c\tilde{\lambda}_i), \quad i = 3, \dots, p$$

with  $\beta_i$  equal to 0 or 1. Furthermore if the eigenvalue  $\tilde{\lambda}_1 = 1$  has multiplicity  $m$  we have

$$\mathbf{r}_c = \tilde{\mathbf{r}} + \sum_{i=m+1}^p \mathbf{w}_i(c) \mathbf{y}_i \quad \text{where} \quad \tilde{\mathbf{r}} = \mathbf{y} + \sum_{i=2}^m \alpha_i \mathbf{y}_i$$

---

In both cases,

$r_c$  is a **rational function** with a **vector numerator** of degree  $p - 1$ , and a **scalar denominator** of degree  $p - 1$  in  $c$ .

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

$$k \ll p - 1.$$

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

---

## VECTOR RATIONAL EXTRAPOLATION (VREM):

We **interpolate** the vectors  $\mathbf{r}_{\mathbf{c}}$  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 coefficients of  $\mathbf{P}_k$  and  $Q_k$  are obtained by solving the interpolation problem

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

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

---

$\mathbf{P}_k$  and  $Q_k$  are given by **Lagrange's interpolation formula**

$$\mathbf{P}_k(\mathbf{c}) = \sum_{i=0}^k L_i(\mathbf{c}) \mathbf{P}_k(\mathbf{c}_i)$$

$$Q_k(\mathbf{c}) = \sum_{i=0}^k L_i(\mathbf{c}) Q_k(\mathbf{c}_i)$$

with

$$L_i(\mathbf{c}) = \prod_{\substack{j=0 \\ j \neq i}}^k \frac{\mathbf{c} - \mathbf{c}_j}{\mathbf{c}_i - \mathbf{c}_j}, \quad i = 0, \dots, k.$$

Thus

$$\mathbf{P}_k(\mathbf{c}) = \sum_{i=0}^k L_i(\mathbf{c}) Q_k(\mathbf{c}_i) \mathbf{p}_i.$$

---

Assume that, for  $\mathbf{c}^* \neq \mathbf{c}_i, i = 0, \dots, k$ , the vector  $\mathbf{r}_{\mathbf{c}^*}$  is known.

Since  $\mathbf{p}(\mathbf{c}) = \mathbf{P}_k(\mathbf{c})/Q_k(\mathbf{c})$ , from the previous result, we will **approximate**  $\mathbf{r}_{\mathbf{c}^*}$  by

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

with  $a_i(\mathbf{c}^*) = Q_k(\mathbf{c}_i)/Q_k(\mathbf{c}^*)$ .

Let  $\mathbf{s}_0, \dots, \mathbf{s}_k$  be  $k + 1$  **linearly independent vectors**. After taking their **scalar products** with the vector  $\mathbf{p}(\mathbf{c}^*)$ , and with the vector  $\mathbf{r}_{\mathbf{c}^*}$ , we will look for  $a_0(\mathbf{c}^*), \dots, a_k(\mathbf{c}^*)$  solution of the system of  $k + 1$  linear equations

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

---

For an **arbitrary value** of  $\mathbf{c}$ , we obtain an **approximation** of  $\mathbf{r}_{\mathbf{c}}$  as

$$\begin{aligned}\mathbf{p}(\mathbf{c}) &= \frac{\mathbf{P}_k(\mathbf{c})/Q_k(\mathbf{c}^*)}{Q_k(\mathbf{c})/Q_k(\mathbf{c}^*)} = \frac{\sum_{i=0}^k L_i(\mathbf{c}) Q_k(\mathbf{c}_i)/Q_k(\mathbf{c}^*) \mathbf{p}_i}{\sum_{i=0}^k L_i(\mathbf{c}) Q_k(\mathbf{c}_i)/Q_k(\mathbf{c}^*)} \\ &= \frac{\sum_{i=0}^k L_i(\mathbf{c}) a_i(\mathbf{c}^*) \mathbf{p}_i}{\sum_{i=0}^k L_i(\mathbf{c}) a_i(\mathbf{c}^*)}.\end{aligned}$$

---

## Vector rational extrapolation procedure (VREM)

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

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

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

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

---

## A SIMPLER VECTOR RATIONAL EXTRAPOLATION (SVREM):

A vector rational extrapolation method by **truncating** the rational expression of  $\mathbf{r}_c$  given by Serra-Capizzano **after two terms** can be constructed.

We consider an extrapolation function of the form

$$\mathbf{p}(\mathbf{c}) = \tilde{\mathbf{y}} + (1 - \mathbf{c}) \frac{1}{1 - \mathbf{c}\lambda} \mathbf{z}.$$

Then, the two unknown vectors  $\tilde{\mathbf{y}}$  and  $\mathbf{z}$ , and the unknown scalar  $\lambda$  are computed by an **interpolation** procedure needing **only 3 values** of  $\mathbf{c}$ .

---

## A MINIMIZATION PROCEDURE (VMP):

We consider an approximation  $\mathbf{p}(\mathbf{c})$  of  $\mathbf{r}_c$  of the form

$$\mathbf{p}(\mathbf{c}) = (1 - \alpha)\mathbf{p}_0 + \alpha\mathbf{p}_1 = \mathbf{p}_0 + \alpha(\mathbf{p}_1 - \mathbf{p}_0),$$

where the parameter  $\alpha$  is chosen so that the **euclidean norm** of  $P_c^T \mathbf{p}(\mathbf{c}) - \mathbf{p}(\mathbf{c})$  (a vector which could be considered as a residual since  $P_c^T \mathbf{r}_c - \mathbf{r}_c$ ) **is minimum**.

It holds

$$\alpha = - \frac{(P_c^T (\mathbf{p}_1 - \mathbf{p}_0) - (\mathbf{p}_1 - \mathbf{p}_0), P_c^T \mathbf{p}_0 - \mathbf{p}_0)}{\|P_c^T (\mathbf{p}_1 - \mathbf{p}_0) - (\mathbf{p}_1 - \mathbf{p}_0)\|^2}.$$

This procedure needs of **only 2 values** of  $\mathbf{c}$ . Obviously it could be extended to a more general form of minimization where

$$\mathbf{p}(\mathbf{c}) = \alpha_0 \mathbf{p}_0 + \cdots + \alpha_k \mathbf{p}_k$$

with  $\alpha_0 + \cdots + \alpha_k = 1$ .

---

## NUMERICAL EXPERIMENTS

A matrix  $\mathbf{P} = (p_{ij}) \in \mathbb{R}^{p \times p}$  having the **same properties** as those coming out from the web is randomly constructed and then the corresponding matrices  $\tilde{\mathbf{P}}$  and  $\mathbf{P}_c$ :

- We select a **random integer**  $q$  between  $1$  and  $p/10$ .
- We generate a **random integer vector**  $\mathbf{m}$  of dimension  $p$  with components between  $1$  and  $q$ . and **each row  $i$  of our matrix  $\mathbf{P}$  will contain, at most,  $\mathbf{m}(i)$  nonzero elements.**
- So, 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  $\text{deg}(i) \leq \mathbf{m}(i)$ , and its components give the indexes  $j$  of the columns such that  $p_{ij} = 1/\text{deg}(i)$ , all others elements being set to zero.
- Among all rows, we **randomly set to zero**  $p/5$  of them, corresponding to the **dangling nodes**.

---

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) :**

$$\begin{aligned} \mathbf{r}_c &= ( 0.23 \quad 0.24 \quad 0.26 \quad 0.27 )^T \\ \text{rank}(\mathbf{r}_c) &= \quad 4 \quad 3 \quad 2 \quad 1 \end{aligned}$$

$$\begin{aligned} \mathbf{r}_1(\mathbf{c}) &= ( 0.27 \quad 0.26 \quad 0.24 \quad 0.25 )^T \\ \text{rank}(\mathbf{r}_1(\mathbf{c})) &= \quad 1 \quad 2 \quad 4 \quad 3 \end{aligned}$$

$$\|\mathbf{r}_c - \mathbf{r}_1(\mathbf{c})\|_\infty = 0.04 \quad (\text{small error, but incorrect ranking})$$

$$\begin{aligned} \mathbf{r}_2(\mathbf{c}) &= ( 0 \quad 0.001 \quad 0.002 \quad 0.997 )^T \\ \text{rank}(\mathbf{r}_2(\mathbf{c})) &= \quad 4 \quad 3 \quad 2 \quad 1 \end{aligned}$$

$$\|\mathbf{r}_c - \mathbf{r}_2(\mathbf{c})\|_\infty = 0.727 \quad (\text{bigger error, but correct ranking})$$

---

## Notations for the methods

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

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

**SVREM 3**  $\longrightarrow$  Simpler vector rational extrapolation with  
 $\mathbf{c}_5, \mathbf{c}_6, \mathbf{c}_7$  (that is  $\mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7$ ).

**VMP 2**  $\longrightarrow$  Minimization procedure with  
 $\mathbf{c}_5, \mathbf{c}_7$  (that is  $\mathbf{p}_5$  and  $\mathbf{p}_7$ ).

$\mathbf{r}_{\mathbf{c}}$  is computed by the power method with a precision of  $10^{-8}$ .

$\mathbf{p}(\mathbf{c})$  is obtained by the preceding procedures.

---

## Notations for the results

**nch** → total number of changes in the ranking between the Pagerank vector  $\mathbf{r}_c$  and the extrapolated vector  $\mathbf{p}(c)$ .

**ich** → rank of the first change occurred after sorting by descending values  $\mathbf{r}_c$  and  $\mathbf{p}(c)$ .

$d_{\max}$  → 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.

$ix_{\max}, iy_{\max}$  → The ranks of the page corresponding to  $d_{\max}$   
 $ix_{\max}$  in the sorted  $\mathbf{r}_c$   
 $iy_{\max}$  in the sorted  $\mathbf{p}(c)$ .

---

Computations performed with a 1.73 GHz Pentium M processor, using Matlab 7.x.

### First example

$p = 5000, nnz = 942806.$

Google parameter  $c = 0.85.$

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	<b>VREM 7</b>	2.77e-8	1.24e-9	<b>689</b>	<b>190</b>	<b>2</b>	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
7	VREM 4	6.86e-7	1.04e-8	3479	18	12	2691	2679
8	SVREM 3	1.17e-5	9.68e-8	4863	1	-234	3529	3763
9	VMP 2	1.20e-5	7.88e-8	4865	1	-236	3529	3765

$p = 5000$ ,  $\mathbf{c} = 0.85$  (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.

---

## Quality of the results

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

$d_{\max}$  indicates the **size of the largest change in the ranking**.

The **smallest**  $d_{\max}$ , the **better the ranking**.

So, a criterion of **good quality** is to have a **small value of**  $d_{\max}$ .

But  $d_{\max}$  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 (**nch large**) in the ranking if they are small, that is if  $d_{\max}$  **is small**.

---

## Another check

We also used the **Kendall  $\tau$  test** which allows to quantify if several criteria give the same ranking for a list of objects. A coefficient of concordance **equal to 1** means that all criteria agree. This coefficient is assumed to follow a  $\chi^2$  law.

This test applied to our numerical results confirms the improvement brought by our extrapolation procedures, and also our criteria for checking the quality of the results.

The best value given by the Kendall test corresponds to the best value according to our criteria.

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	<b>VREM 7</b>	2.65e-9	1.29e-10	<b>66</b>	<b>272</b>	<b>-1</b>	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
7	VREM 4	3.52e-5	2.25e-6	4810	10	128	2362	2234
8	SVREM 3	4.02e-6	4.48e-8	4673	14	-101	3529	3630
9	VMP 2	4.15e-6	4.07e-8	4683	14	-102	3529	3631

---

Extrapolation for  $\mathbf{r}_c$  with  $\mathbf{c} = 0.99$  (13 iterations) and  $\mathbf{c}_i = 0.55, 0.6, 0.65$  (6 iterations),  $\mathbf{c}^* = 0.25$  gives

#	Method	$\ \mathbf{r}_c - \mathbf{p}\ _\infty$	$\ \mathbf{r}_c - \mathbf{p}\ _{1/p}$	nch	ich	$d_{\max}$	$ix_{\max}$	$iy_{\max}$
7	VREM 4	3.52e-5	2.25e-6	4810	10	128	2362	2234
8	<b>SVREM 3</b>	3.08e-5	2.30e-6	<b>4349</b>	<b>29</b>	<b>-40</b>	3529	3569
9	<b>VMP 2</b>	3.03e-5	2.27e-6	<b>4359</b>	<b>29</b>	<b>-40</b>	3529	3569

---

## 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	<b>VREM 9</b>	2.57e-4	6.93e-8	<b>281652</b>	<b>11</b>	<b>-219944</b>	61958	281902
7	VREM 4	2.00e-3	6.74e-7	261353	4	-80743	51569	132312
8	SVREM 3	1.59e-3	8.74e-7	261573	1	-160026	19635	179661
9	VMP 2	1.51e-3	9.27e-7	261574	1	-160620	19635	180255

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	<b>VREM 8</b>	3.32e-5	7.33e-9	<b>257896</b>	<b>29</b>	<b>-20639</b>	32553	53192
6	VREM 9	1.03e-5	2.98e-9	254360	14	-13364	38289	51653
7	VREM 4	7.72e-4	2.76e-7	260586	4	-37167	32553	69720
8	SVREM 3	1.59e-3	2.95e-7	261569	1	-79828	51569	131397
9	VMP 2	1.01e-3	4.61e-7	261576	1	-81710	51569	133279

---

### Example 3: A small matrix

$p = 1000, nnz = 18729, \mathbf{c} = 0.85$  (12 iterations)

$\mathbf{c}_i = 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, \mathbf{c}^* = 0.5$  (9 iterations)

#	Method	$\ \mathbf{r}_c - \mathbf{p}\ _\infty$	$\ \mathbf{r}_c - \mathbf{p}\ _{1/p}$	nch	ich	$d_{\max}$	$ix_{\max}$	$iy_{\max}$
1	VREM 4	5.14e-5	1.91e-6	913	9	-51	348	399
2	VREM 5	8.29e-6	5.44e-7	603	12	9	398	389
3	VREM 6	1.39e-6	1.15e-7	175	13	4	486	482
4	VREM 7	1.23e-6	1.03e-7	152	62	4	486	482
5	<b>VREM 8</b>	5.38e-7	4.60e-8	<b>81</b>	<b>127</b>	<b>-2</b>	395	397
6	<b>VREM 9</b>	2.66e-7	2.35e-8	<b>46</b>	<b>127</b>	<b>2</b>	145	143
7	VREM 4	5.42e-6	4.46e-7	615	15	-7	467	474
8	SVREM 3	1.17e-4	3.63e-6	969	8	101	425	324
9	VMP 2	1.35e-4	3.77e-6	966	8	101	425	324

---

---

## REFERENCES:

- C. Brezinski, M. Redivo-Zaglia, S. Serra-Capizzano, *Extrapolation methods for PageRank computations*, **C.R. Math. Acad. Sci. Paris**, **340** (2005) 393-397.
- C. Brezinski, M. Redivo-Zaglia, *Rational extrapolation for the PageRank vector*, **Math. Comp.**, to appear.

---

## NEW IDEAS FOR FUTURE WORK

- $q$ -algorithm: componentwise vector rational extrapolation at infinity.
- Scalar rational extrapolation: Thiele interpolation process constructed by means of a continued fraction.
- restart of the power iterations after extrapolation.

---

## THE $\varrho$ -ALGORITHM:

We consider a sequence  $(c_n)$  of values of the parameter and the corresponding **vectors**  $\mathbf{p}_n = \mathbf{r}_{c_n}$ .

We want to compute an approximation of  $\mathbf{r}_c$  for a given  $c$

We set  $d_n = \frac{1}{c - c_n}$ ,  $\varrho_0^{(n)} = \mathbf{p}_n$  and  $\varrho_{-1}^{(n)} = 0, \forall n$ .

We apply the **scalar  $\varrho$ -algorithm** and we compute **componentwise** the **vectors**  $\varrho_k^{(n)}$  by the relation

$$(\varrho_{k+1}^{(n)})_j = (\varrho_{k-1}^{(n+1)})_j + \frac{d_{n+k+1} - d_n}{(\varrho_k^{(n+1)})_j - (\varrho_k^{(n)})_j}, \quad k = 0, 1, \dots, \quad n = 0, 1, \dots$$

for  $j = 1, \dots, p$ .

---

$(\varrho_{2k}^{(n)})_j$  is the **value at infinity** of the scalar rational function with a numerator and a denominator of degree  $k$  which interpolates  $(\mathbf{p}_n)_j, \dots, (\mathbf{p}_{n+2k})_j$  at the points  $d_n, \dots, d_{n+2k}$ .

Thus  $\varrho_{2k}^{(n)}$  **is an approximation of  $\mathbf{r}_c$** .

**Remark:** In the  $\varrho$ -algorithm, the quantities with an odd lower index are intermediate computations without an interesting meaning.

**Remark:**  $\varrho_{2k}^{(n)}$  depends on  $k$  and on  $n$ . By changing these values we obtain different approximations of  $\mathbf{r}_c$ .

---

## SCALAR RATIONAL EXTRAPOLATION:

We consider  $2k + 1$  distinct points  $c_0, \dots, c_{2k}$  and the corresponding vectors  $\mathbf{p}_0, \dots, \mathbf{p}_{2k}$ . The idea is to interpolate **component by component** the vectors  $\mathbf{p}_i$  by a rational function, and then to compute the value at the point  $c$ .

That is, for  $j = 1, \dots, p$ , we have to determine the scalar polynomials  $P_k$  and  $Q_k$  of **degree**  $k$

$$(\mathbf{p}(t))_j = \frac{P_k(t)}{Q_k(t)},$$

such that

$$(\mathbf{p}(c_i))_j = (\mathbf{p}_i)_j = \frac{P_k(c_i)}{Q_k(c_i)}, \quad i = 0, \dots, 2k,$$

---

## THE THIELE INTERPOLATION PROCESS:

A function  $f$  known at the points  $t_0, t_1, \dots$  can be interpolated by a rational function using **Thiele continued fraction**

$$C(t) = \alpha_0 + \cfrac{t - t_0}{\alpha_1} + \cfrac{t - t_1}{\alpha_2} + \dots$$

The  **$2k$ -th convergent** of this continued fraction can be written as

$$C_{2k}(t) = \frac{A_{2k}(t)}{B_{2k}(t)},$$

where  $A_{2k}$  and  $B_{2k}$  are polynomials of **degree  $k$** , and it satisfies

$$C_{2k}(t_i) = f(t_i) \quad \text{for } i = 0, \dots, 2k.$$

---

These polynomials satisfy the recurrence relations

$$A_n(t) = \alpha_n A_{n-1}(t) + (t - t_{n-1}) A_{n-2}(t)$$

$$B_n(t) = \alpha_n B_{n-1}(t) + (t - t_{n-1}) B_{n-2}(t)$$

for  $n = 1, \dots, 2k$ , with

$$A_{-1}(t) = 1, \quad A_0(t) = \alpha_0$$

$$B_{-1}(t) = 0, \quad B_0(t) = 1.$$

---

The coefficients  $\alpha_i$  are given by

$$\alpha_0 = \varrho_0^{(0)} \quad \text{and} \quad \alpha_n = \varrho_n^{(0)} - \varrho_{n-2}^{(0)}, \quad n = 1, \dots, 2k,$$

where the quantities  $\varrho_n^{(i)}$  are computed by the  **$\varrho$ -algorithm**

$$\begin{aligned} \varrho_{-1}^{(i)} &= 0, & \varrho_0^{(i)} &= f(t_i), & i &= 0, \dots, \\ \varrho_{n+1}^{(i)} &= \varrho_{n-1}^{(i+1)} + \frac{t_{i+n+1} - t_i}{\varrho_n^{(i+1)} - \varrho_n^{(i)}}, & n, i &= 0, 1, \dots \end{aligned}$$

**In our case,**

$$t_i = c_i$$

$$f(t_i) = (\mathbf{p}_i)_j$$

$$t = c$$

and we will have

$$C_{2k}(c) \simeq (\mathbf{r}_c)_j.$$

---

## OTHER RATIONAL PROCEDURES:

- Vector  $\varrho$ -algorithm for extrapolation at infinity.
- Topological vector  $\varrho$ -algorithm for extrapolation at infinity.
- Vector rational interpolation by means of a vector Thiele continued fraction (M. De Bruin, P. Graves-Morris, J. Van Iseghem, P. Wynn, ...).
- ...

---

## FINAL REMARKS

→ As pointed out by [Sebastiano Vigna \(2007\)](#)

*Stanford matrix considered harmful*

*“... there are criteria that should delimit what we use in our experiments if we want to derive sensible conclusions, and the Stanford matrix largely falls short of such criteria.” ...*

**Answer:**

*David Gleich Matlab package that works with Sebastiano Vigna super-compressed graphs! (private communication, September 5, 2007)*

→ Maybe Google and other search engines no longer use PageRank.

*It is now **only** a game for mathematicians!*

**BUT...**

---

## ...THERE ARE OTHER APPLICATIONS

### → **Networks**

- M.E.J. Newman, The structure and function of complex networks, *SIAM Rev.*, 45 (2003) 167–256.
- E. Ravasz, A.-L. Barabási, Hierarchical organization in complex networks, *Phys. Rev., E* 67, 026112 (2003).
- R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, U. Alon, Superfamilies of designed and evolved networks, *Science*, 303 (2004) 1538–1542.

### → **Small** (but **reducible**) matrices.

- **Kerner, 2006**: agglomeration of viruses (matrices of dimensions 10 or 20!)

### → **Reliability of industrial systems**

- T. Aven, U. Jensen, *Stochastic Models in Reliability*, Springer, 1999