Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 30(2): 732–754.
DOI: 10.3934/era.2022039
Received: 14 December 2021
Revised: 07 February 2022
Accepted: 13 February 2022
Published: 28 February 2022

Research article

Acceleration of the generalized FOM algorithm for computing PageRank

Yu Jin[1], **Chun Wen**[1,*] **and Zhao-Li Shen**[2,*]

[1] School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China

[2] College of Science, Sichuan Agricultural University, Ya'an 625000, China

* **Correspondence:** Email: wchun17@163.com, szlxiaoyao@163.com.

Abstract: In this paper, a generalized full orthogonalization method (GFOM) based on weighted inner products is discussed for computing PageRank. In order to improve convergence performance, the GFOM algorithm is accelerated by two cheap methods respectively, one is the power method and the other is the extrapolation method based on Ritz values. Such that two new algorithms called GFOM-Power and GFOM-Extrapolation are proposed for computing PageRank. Their implementations and convergence analyses are studied in detail. Numerical experiments are used to show the efficiency of our proposed algorithms.

Keywords: PageRank; FOM; generalized FOM; power method; ritz values; extrapolation procedure

1. Introduction

With the booming development of the internet, the way people obtain information is gradually changing to the web search. Google search engine is one of the most popular and successful search engines, and it uses PageRank to determine the importance of web pages [1].

Let's briefly introduce the mathematical model of PageRank problems. If the link structure of web pages is considered as a directed link graph, then the adjacency matrix $P \in \mathbb{R}^{n\times n}$ of the graph can be described as

$$P = (p_{ij}) = \begin{cases} \frac{1}{n_i}, & \text{if page } i \text{ links to page } j, \\ 0, & \text{otherwise,} \end{cases}$$

where n_i denotes the number of outlinks of page i. If page i contains no outlinks (page i is called as a dangling node [2]), then the ith row of P will be zero. The existence of dangling nodes is a drawback for computing PageRank. To correct this problem, a rank-1 modification is applied to obtain a row stochastic matrix, that is,

$$\tilde{P} = P + dw^{\mathrm{T}},$$

where $d = (d_i) \in \mathbb{R}^{n\times 1}$ with

$$d_i = \begin{cases} 1, & \text{if } n_i = 0, \\ 0, & \text{otherwise,} \end{cases}$$

and $w = (w_i) \in \mathbb{R}^{n\times 1}$ is a probability distribution vector.

To further guarantee the aperiodicity and irreducibility of $\tilde{P}$, a convex combination of $\tilde{P}$ is considered by introducing a damping factor α and a personalization vector v, which yields the Google matrix

$$A = [\alpha\tilde{P} + (1-\alpha)ev^{\mathrm{T}}]^{\mathrm{T}} = \alpha\tilde{P}^{\mathrm{T}} + (1-\alpha)ve^{\mathrm{T}},$$

where $\alpha \in (0,1)$ and $v = e/n$ with $e = [1,1,\ldots,1]^{\mathrm{T}} \in \mathbb{R}^{n\times 1}$. Usually, we set $w = v$. The Google matrix A is stochastic and irreducible after two corrections. Finally, the PageRank problem requires to solve the following linear system:

$$Ax = x, \quad \|x\|_1 = 1, \quad x > 0, \tag{1.1}$$

where x is called the unknown PageRank vector. More details about the formulation of PageRank problems, please refer to [3,4] and references therein.

The traditional method for computing PageRank is the power method [5] since it is based on matrix-vector products and requires small memory space. However, when the largest and the second largest eigenvalues of the matrix A can not be well separated, or in other words, when the damping factor α is sufficiently close to 1, the power method performs poorly. To speed up the PageRank computation, extensive studies have been carried out over the past few years, including the extrapolation methods [6–9], the inner-outer methods [10–14], the adaptive methods [15–17] and the multigrid methods [18,19].

In addition, Krylov subspace methods play a powerful role in solving PageRank problems. Golub and Greif proposed an Arnoldi-type algorithm [20] without Ritz value computations by using a relevant shift in the refined Arnoldi method [21] to be 1. Wu and Wei developed a Power-Arnoldi algorithm [22] by periodically knitting the power method with the thick restarted Arnoldi algorithm [23]. Hu et al. proposed a variant of the Power-Arnoldi algorithm [24] by using the power method with the extrapolation process based on trace (PET) [8]. Wu and Wei developed an Arnoldi-Extrapolation algorithm [25] by periodically combining the Arnoldi-type method [20] with the extrapolation method based on Ritz values. Zhang et al. proposed a FOM-Extrapolation algorithm [26] by using the same extrapolation method. Gu et al. developed a GMRES-Power algorithm [27] by periodically knitting the GMRES method [28] with the power method. Yin et al. proposed a generalized Arnoldi (GArnoldi) algorithm [15] by replacing the standard inner products with the weighted inner products during the Arnoldi process. Wen et al. developed a Power-GArnoldi algorithm [16] by taking the adaptive GArnoldi method as an accelerated technique for the power method. More numerical methods for computing PageRank problems can be found in [29–37].

Since the Google matrix A in system (1.1) is stochastic, its largest eigenvalue is 1, and thus the linear system (1.1) can be rewritten as the following singular linear system:

$$(I - A)x = 0, \quad \|x\|_1 = 1, \quad x > 0, \tag{1.2}$$

where $I \in \mathbb{R}^{n\times n}$ is an identity matrix.

In [26], the FOM algorithm is used to solve system (1.2). Hence, inspired by the GArnoldi algorithm [15–17], a generalized full orthogonalization method (GFOM) based on a weighted inner product is discussed for solving the singular linear system (1.2). To speed up the convergence performance, two accelerated techniques are applied to the GFOM algorithm respectively, one is the power method, and the other is the extrapolation method based on Ritz values, such that two new algorithms called GFOM-Power and GFOM-Extrapolation are proposed for computing PageRank. Their implementations and convergence analysis are studied in detail. Numerical results show that the performance of our proposed algorithms are superior to the other methods when the damping factor α is close to 1.

The remainder of the paper is organized as follows. In Section 2, we first briefly review the GArnoldi process, then discuss the GFOM algorithm for computing PageRank. In Section 3, we propose the GFOM-Power algorithm and the GFOM-Extrapolation algorithm for computing PageRank, and analyze their convergence properties, respectively. Numerical experiments and comparisons are reported in Section 4. Finally, conclusions are given in Section 5.

2. GFOM algorithm for computing PageRank

In this section, we first briefly review the generalized Arnoldi (GArnoldi) process [15] based on weighted inner products, and then discuss the generalized FOM (GFOM) algorithm for computing PageRank. Some comparison results for the FOM algorithm and GFOM algorithm are presented at last.

2.1. The GArnoldi process

Given a symmetric positive definite (SPD) matrix $G = (g_{ij}) \in \mathbb{R}^{n\times n}$, let $x = (x_i) \in \mathbb{R}^{n\times 1}$, $y = (y_i) \in \mathbb{R}^{n\times 1}$ be two vectors, then a G-inner product is defined as

$$(x, y)_G = x^{\mathrm{T}} G y = \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij} x_i y_j.$$

Since the matrix G is symmetric positive definite, it can be diagonalized. Suppose $G = Q^{\mathrm{T}} D Q$, where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, and $D = \mathrm{diag}\{d_1, d_2, \ldots, d_n\}$ is a $n \times n$ diagonal matrix with $d_i > 0$, $i = 1, 2, \ldots, n$. Then the corresponding G-norm is defined as

$$\|x\|_G = \sqrt{(x, x)_G} = \sqrt{x^{\mathrm{T}} G x} = \sqrt{x^{\mathrm{T}} Q^{\mathrm{T}} D Q x} = \sqrt{\sum_{i=1}^{n} d_i (Qx)_i^2}, \quad \forall x \in \mathbb{R}^{n\times 1}.$$

Given a general non-Hermitian matrix $\tilde{A} \in \mathbb{R}^{n\times n}$, an initial vector $v_0 \in \mathbb{R}^{n\times 1}$, a SPD matrix G, and the steps m of the GArnoldi process, then the GArnoldi process is described as follows. More details can be found in [15–17].

Algorithm 1. $[V_m, H_m, v_{m+1}, h_{m+1,m}, \beta]$ = GArnoldi $(\tilde{A}, v_0, G, m)$

1. Compute $\beta = \|v_0\|_G$, $v_1 = v_0/\beta$.
2. for $j = 1, 2, \ldots, m$

3. Compute $z = \tilde{A}v_j$
4. for $i = 1, 2, \ldots, j$
5. Compute $h_{i,j} = (v_i, z)_G$
6. Compute $z = z - h_{i,j}v_i$
7. end for
8. Compute $h_{j+1,j} = \|z\|_G$
9. if $h_{j+1,j} = 0$
10. break;
11. end if
12. $v_{j+1} = z/h_{j+1,j}$
13. end for

Note that when the matrix $G = I$, the GArnoldi process is reduced to the standard Arnoldi process [21, 38]. And according to Algorithm 1, the following relations hold:

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^{\mathrm{T}} = V_{m+1} H_{m+1,m}, \quad GAV_m = V_m H_m, \quad H_{m+1,m} = \begin{pmatrix} H_m \\ h_{m+1,m} e_m^{\mathrm{T}} \end{pmatrix}, \tag{2.1}$$

where $V_k = [v_1, v_2, \ldots, v_k] \in \mathbb{R}^{n\times k}$ ($k = m, m+1$) is a G-orthogonal matrix, $H_m = (h_{i,j}) \in \mathbb{R}^{m\times m}$ is an upper Hessenberg matrix and $e_m \in \mathbb{R}^{m\times 1}$ is the mth co-ordinate vector.

2.2. The GFOM algorithm

In this subsection, by using the G-inner product as given above, we consider a generalized FOM (GFOM) algorithm for computing PageRank. The specific implementation of the GFOM algorithm is given as follows.

Algorithm 2. The GFOM algorithm for computing PageRank

1. Given a nonzero initial vector x_0, the steps m of the GArnoldi process and a prescribed tolerance tol.
2. Compute the initial residual vector $r_0 = -(I - A)x_0$.
3. Set $G = I$.
4. Run Algorithm 1 as $[V_m, H_m, v_{m+1}, h_{m+1,m}, \beta]$ = GArnoldi$(I - A, r_0, G, m)$.
5. Compute the approximation vector $x = x_0 + V_m y$ with $y = \beta H_m^{-1} e_1$.
6. Compute $r = -h_{m+1,m}(e_m^{\mathrm{T}} y)v_{m+1}$.
7. if $\|r\|_2 / \|x\|_1 \le tol$
8. Output $x = x/\|x\|_1$ and stop
9. else
10. Set $r_0 = r$, $x_0 = x$, $G = \mathrm{diag}\{|r|/\|r\|_1\}$ and goto step 4
11. end if

Here, some remarks need to be given about Algorithm 2.

- In the line 5, the GFOM algorithm seeks a solution x satisfying the Galerkin condition: $-(I - A)x \perp \mathcal{K}_m(I - A, r_0)$, then we have the following formulation:

$$0 \;=\; V_m^{\mathrm{T}}(-(I - A)x) = V_m^{\mathrm{T}}(Ax_0 - x_0 + AV_m y - V_m y)$$

$$= V_m^{\mathrm{T}} r_0 - V_m^{\mathrm{T}}(I - A)V_m y = \beta e_1 - H_m y,$$

where $e_1 = [1, 0, \ldots, 0]^{\mathrm{T}}$. Thus if H_m is nonsingular, the vector y can be computed as $y = \beta H_m^{-1} e_1$.

- In the line 7, there are two differences between the GFOM algorithm and the FOM algorithm [26]. One is the 2-norm of the residual vector r, which can be computed as $\|r\|_2 = |h_{m+1,m}| \cdot |y(m :)| \cdot \|v_{m+1}\|_2$ rather than $|h_{m+1,m}| \cdot |y(m, :)|$. The other is the convergence condition, which is set as $\|r\|_2/\|x\|_1 \leq tol$ rather than $\|r\|_2/\|x\|_2 \leq tol$.
- In the first iteration, the GFOM algorithm reduces to the FOM algorithm since $G = I$. For other iterations, from the line 10 of Algorithm 2, we construct the matrix G as

$$G = \mathrm{diag}\{\delta_1, \delta_2, \ldots, \delta_n\}, \delta_i = |r_i|/\|r\|_1, i = 1, 2, \ldots, n, \tag{2.2}$$

where r_i is the ith component of the residual vector r.

2.3. Comparisons of the FOM algorithm and the GFOM algorithm

In this subsection, we aim at testing the effectiveness of the GFOM algorithm by comparing it with the FOM algorithm [26] for computing PageRank. The same initial guess $x_0 = e/n$ is used, where $e = [1, 1, \ldots, 1]^{\mathrm{T}}$. The damping factors are chosen as $\alpha = 0.99$, 0.993, 0.995 and 0.997, respectively. And the stopping criteria are set as $|h_{m+1,m}| \cdot |y(m, :)|/\|x\|_1 \leq tol$ for the FOM algorithm and $|h_{m+1,m}| \cdot |y(m :)| \cdot \|v_{m+1}\|_2/\|x\|_1 \leq tol$ for the GFOM algorithm, where $tol = 10^{-8}$ is the user described tolerance.

Here we show the choice of the restart number m by analyzing the numerical performance of the FOM algorithm and the GFOM algorithm for the wb-cs-stanford matrix, which is available from [39], and it obtains 9914 pages and 36,854 links. Table 1 reports the number of matrix-vector products for the FOM algorithm and the GFOM algorithm when $\alpha = 0.99$, 0.993, 0.995, 0.997 and $m = 6$, 7, 8, 9, 10, respectively. Figure 1 depicts the total CPU time of the FOM algorithm and the GFOM algorithm relative to different restart number m, respectively.

Table 1. The number of matrix-vector products of the FOM algorithm and the GFOM algorithm with different damping factor α and restart number m for the wb-cs-stanford matrix.

α	$m = 6$		$m = 7$		$m = 8$		$m = 9$		$m = 10$	
	FOM	GFOM	FOM	GFOM	FOM	GFOM	FOM	GFOM	FOM	GFOM
$\alpha = 0.99$	280	287	240	176	198	162	260	160	209	154
$\alpha = 0.993$	455	350	312	208	261	189	340	200	264	176
$\alpha = 0.995$	539	448	392	232	315	207	420	220	341	231
$\alpha = 0.997$	644	490	552	320	621	252	520	240	385	330

From Table 1, it observes that the number of matrix-vector products of the FOM algorithm and the GFOM algorithm is decreasing at first, and then is increasing as m increases in most cases. Specifically, it seems $m = 8$ is a clear turning point on the whole. Meanwhile, from Figure 1, we find that the optimal CPU time of the FOM algorithm and the GFOM algorithm is different for different damping factor α. However, considering the memory requirements of the FOM algorithm and the GFOM algorithm, it is obvious that $m = 8$ is a reasonable choice for most cases. Hence, we set $m = 8$ in the following numerical experiments.

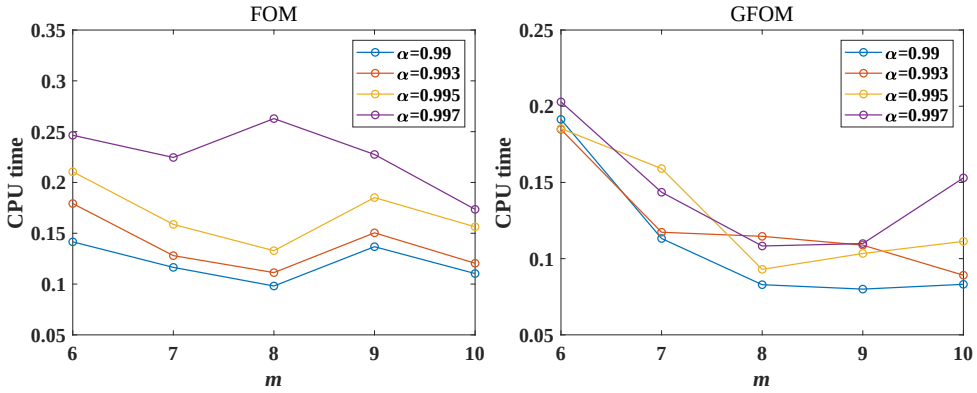

Figure 1. The total CPU time of the FOM algorithm and the GFOM algorithm versus different restart number m for the wb-cs-stanford matrix.

Figure 2 plots the convergence behavior of the FOM algorithm and the GFOM algorithm for the wb-cs-stanford matrix when α = 0.99, 0.993, 0.995 and 0.997, respectively. It shows that the GFOM algorithm has a faster convergence than the FOM algorithm. Thus it is meaningful to consider the weighted inner products in the standard Arnoldi process.

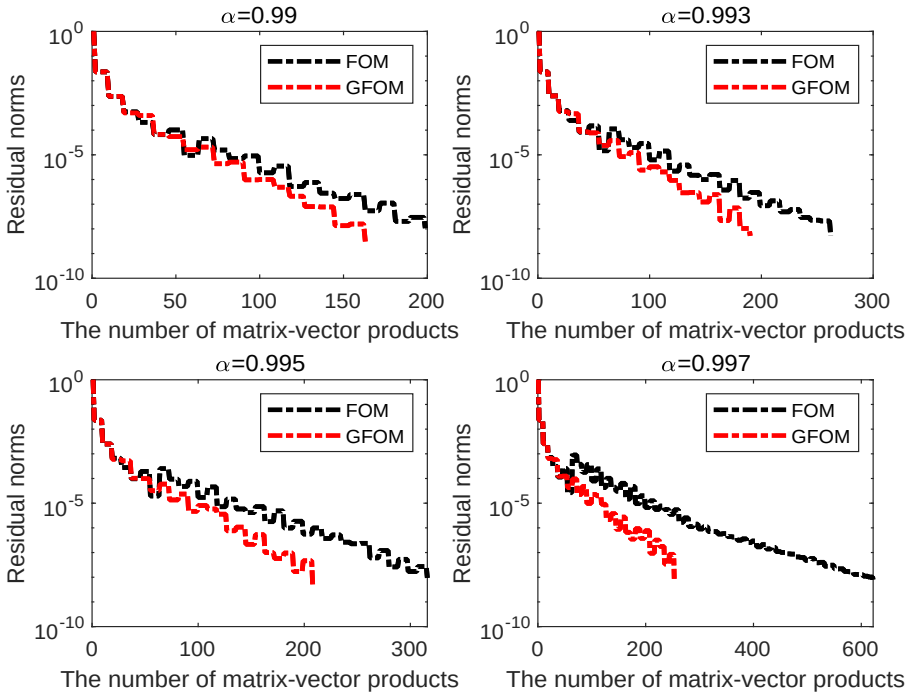

Figure 2. Convergence behavior of the FOM algorithm and the GFOM algorithm for the wb-cs-stanford matrix.

3. Acceleration of the GFOM algorithm for computing PageRank

Since the GFOM algorithm requires large memory as the restart number m increases, two accelerated techniques are used to improve the GFOM algorithm respectively, one is the power method and the other is the extrapolation method based on Ritz values. Thus two new algorithms named GFOM-Power and GFOM-Extrapolation are developed to compute PageRank problems, and their constructions and convergence analyses are discussed in detail.

3.1. The GFOM-Power algorithm for computing PageRank

In this subsection, we first give the description of the GFOM-Power algorithm, and then discuss its convergence.

3.1.1. The GFOM-Power algorithm

Similar to the construction of these algorithms in [16, 22, 24]. The idea of the GFOM-Power algorithm can be described as follows: given an initial vector x_0, we first run Algorithm 2 for a few times (e.g., 2–3 times) to get an approximation, if the approximation is unsatisfactory, then we use the resulting vector as the initial guess of the power method to obtain another approximation. If this approximation is still unsatisfactory, repeat the above procedure periodically until the required tolerance is achieved.

One of the problems is when and how to control the conversion between the power method and the GFOM algorithm. Here we use the parameters ϕ, $restart$, $maxit$ as the flip-flop [22]. Let τ^{curr} and τ^{prev} be the residual norm of the current power iteration and the previous power iteration, respectively. Denoting $ratio = \tau^{curr}/\tau^{prev}$, we estimate whether $ratio$ is greater than ϕ, if so, let $restart = restart + 1$. And then we check whether $restart$ is larger than $maxit$, if so, terminate the power iteration and cycle the GFOM algorithm. Otherwise, continue to run the power method. Considering the asymptotic convergence rate of the power method for PageRank problems [5], it is reasonable to choose $\phi = \alpha - 0.1$. In summary, the GFOM-Power algorithm that computes PageRank is given as follows.

Algorithm 3. The GFOM-Power algorithm for computing PageRank

1. Given a unit positive initial guess x_0, the steps m of the GArnoldi process, a prescribed tolerance tol, the parameters ϕ, $maxit$ and set $restart = 0$, $\tau = 1$, $\tau_0 = \tau$, $\tau_1 = \tau$.
2. Run Algorithm 2 for a few times (2–3 times): iterate steps 1–11 for the first run and steps 4–11 otherwise. If the residual norm satisfies the prescribed tolerance tol, then stop, else continue.
3. Run the power method with $\tilde{x}$ as the initial guess, where $\tilde{x}$ is the approximation vector obtained from the GFOM algorithm:
3.1. $restart = 0$
3.2. while $restart < maxit \quad \& \quad \tau > tol$
3.3. $\quad \tilde{x} = \tilde{x}/\|\tilde{x}\|_1$
3.4. $\quad ratio = 0$
3.5. $\quad$ while $ratio < \phi \quad \& \quad \tau > tol$
3.6. $\quad\quad x^{\mathrm{P}} = A\tilde{x}$

3.7. $r = x^{\mathrm{P}} - \tilde{x}$
3.8. $\tau = \|r\|_2$
3.9. $ratio = \tau/\tau_0$ % τ^{curr}/τ^{prev}
3.10. $\tilde{x} = x^{\mathrm{P}}$, $\tau_0 = \tau$
3.11. end while
3.12. if $\tau/\tau_1 > \phi$ % τ^{curr}/τ^{prev}
3.13. $restart = restart + 1$
3.14. end if
3.15. $\tau_0 = \tau$, $\tau_1 = \tau$
3.16. end while
3.17. Set $G = \mathrm{diag}\{|r|/\|r\|_1\}$
3.18. if $\tau \leq tol$, stop, else goto step 2

Note that the residual r in the line 3.7 of Algorithm 3 is used not only to construct G (see the line 3.17), but also for checking convergence.

3.1.2. Convergence analysis of the GFOM-Power algorithm

In this subsection, we pay attention to analyze the convergence of the GFOM-Power algorithm. In particular, our analysis focuses on the procedure when turning from the power method to the GFOM algorithm.

Assume that eigenvalues of the Google matrix A are ordered as $1 = |\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ and $\Lambda(A)$ denotes the set of eigenvalues of A. Let $\mathcal{P}_{m-1}$ be the set of polynomials of degree not greater than $m-1$ and (λ_i, μ_i), $i = 1, 2, \ldots, n$ denote the eigenpairs of A. For two given vectors v and w, $\cos\angle(v, w) = \frac{\langle v,w\rangle}{\|v\|_2\|w\|_2}$ indicates the cosin of the corresponding angle between them. We first introduce a useful theorem about the spectrum property of the Google matrix A.

Theorem 1 [40]. Let $\tilde{P}$ be an $n \times n$ column-stochastic matrix. Let α be a real number such that $0 < \alpha < 1$. Let E be an $n \times n$ rank-one column-stochastic matrix $E = ve^{\mathrm{T}}$, where e is the n-vector whose elements are all ones and v is an n-vector whose elements are all nonnegative and sum to 1. Let $A = \alpha\tilde{P} + (1-\alpha)E$ be an $n \times n$ column-stochastic matrix, then its dominant eigenvalue $\lambda_1 = 1$, $|\lambda_2| \leq \alpha$.

Now let's analyze the convergence of the GFOM-Power algorithm. Comparing the FOM algorithm and the GFOM algorithm, it is not difficult to find that the main difference between them lies in the Arnoldi process. The former uses the 2-norm, while the latter uses the G-norm. Suppose the weighted matrix G is given as in Eq (2.2), then the relation between the G-norm and the 2-norm is shown as follows.

Lemma 1 [16]. Let $G = \mathrm{diag}\{\delta_1, \delta_2, \cdots, \delta_n\}$, $\delta_i > 0, 1 \leq i \leq n$, be a diagonal matrix. For any vector $x \in \mathbb{R}^{n\times 1}$, according to the definitions of the G-norm and the 2-norm, it has

$$\min_{1\leq i\leq n} \delta_i \cdot \|x\|_2^2 \leq \|x\|_G^2 \leq \max_{1\leq i\leq n} \delta_i \cdot \|x\|_2^2 . \tag{3.1}$$

Since our analysis focuses on the procedure when turning from the power method to the GFOM algorithm, it is necessary to derive the iterative formula of the power method in Algorithm 3. Let $\tilde{x}$ be the initial vector for the power method, which is obtained from the GFOM algorithm. Then the power method of Algorithm 3 produces the vector $x^{\mathrm{P}} = \omega A^l \tilde{x}$, where $l \geq maxit$, and $\omega = 1/\|A^l\tilde{x}\|_1$ is the

normalizing factor. In the next cycle of the GFOM-Power algorithm, the resulting vector x^{P} will be used as the initial vector for an m-step GFOM algorithm, so that the new associated Krylov subspace is

$$\mathcal{K}_m(I-A, r^{\mathrm{P}}) = \mathrm{span}\{r^{\mathrm{P}}, (I-A)r^{\mathrm{P}}, \ldots, (I-A)^{m-1}r^{\mathrm{P}}\},$$

where $r^{\mathrm{P}} = -(I-A)x^{\mathrm{P}}$. For any vector $u \in \mathcal{K}_m(I-A, r^{\mathrm{P}})$, there exists a certain polynomial $p(x)$ achieving $\min\limits_{p\in\mathcal{P}_{m-1}} \max\limits_{\lambda\in\Lambda(A)/\lambda_1} |p(\lambda)|$, so that $u = x^{\mathrm{P}} + p(I-A)r^{\mathrm{P}}$. Then, we present the convergence result of the GFOM-Power algorithm for computing PageRank as follows.

Theorem 2. Let $\tilde{x}$ be the current approximate vector obtained from the GFOM algorithm, and assume that $\tilde{x} = \sum_{i=1}^{n} \gamma_i \mu_i$ with respect to the eigenbasis $\{\mu_i\}_{i=1,2,\ldots,n}$, in which $\|\mu_i\|_2 = 1$, $i = 1, 2, \ldots, n$ and $\gamma_i \neq 0$. Then, the relationship between the next approximate vector u and μ_1 holds asymptotically that

$$|\sin\angle(u, \mu_1)| \leq \frac{\sqrt{\max_{1\leq i\leq n}\delta_i}}{\sqrt{\min_{1\leq i\leq n}\delta_i}} \cdot \alpha^l \cdot \left(\sum_{i=2}^{n} \frac{|\gamma_i|}{|\gamma_1|}\right) \cdot \varepsilon_1,$$

where $l \geq maxit$ and $\varepsilon_1 = \min\limits_{q\in\mathcal{P}_{m-1}} \max\limits_{\lambda\in\Lambda(A)/\lambda_1} |q(\lambda)|$ with $q(\lambda) = 1-(1-\lambda)p(1-\lambda)$.

Proof. From the step 3 of Algorithm 3, we know that $\tilde{x}$ is used as an initial vector to run the power iterations, thus it has $x^{\mathrm{P}} = \omega A^l \tilde{x}$, which can be represented by

$$x^{\mathrm{P}} = \omega \sum_{i=1}^{n} \gamma_i A^l \mu_i.$$

The corresponding residual is computed as $r^{\mathrm{P}} = -(I-A)x^{\mathrm{P}}$, which is used as the initial residual vector for the next GFOM algorithm. Thus the next approximate vector $u \in \mathcal{K}_m(I-A, r^{\mathrm{P}})$ is obtained, and there exists a polynomial $p \in \mathcal{P}_{m-1}$ so that

$$\begin{aligned} u &= x^{\mathrm{P}} + p(I-A)r^{\mathrm{P}} \\ &= [I-(I-A)p(I-A)]x^{\mathrm{P}} \\ &= \omega \sum_{i=1}^{n} \gamma_i \lambda_i^l [1-(1-\lambda_i)p(1-\lambda_i)]\mu_i, \end{aligned}$$

where we used the facts that $A\mu_i = \lambda_i\mu_i$ $(i = 1, 2, \ldots, n)$.

By computing, we have

$$|\sin\angle(u, \mu_1)| \leq \frac{\left\|\sum_{i=2}^{n} \gamma_i \lambda_i^l [1-(1-\lambda_i)p(1-\lambda_i)]\mu_i\right\|_G}{\|\gamma_1\mu_1\|_G}. \tag{3.2}$$

Using Eq (3.1), for the numerator of Eq (3.2), it has

$$\begin{aligned} \left\|\sum_{i=2}^{n} \gamma_i \lambda_i^l [1-(1-\lambda_i)p(1-\lambda_i)]\mu_i\right\|_G &\leq \sqrt{\max_{1\leq i\leq n}\delta_i} \cdot \left\|\sum_{i=2}^{n} \gamma_i \lambda_i^l [1-(1-\lambda_i)p(1-\lambda_i)]\mu_i\right\|_2 \\ &\leq \sqrt{\max_{1\leq i\leq n}\delta_i} \cdot \left(\sum_{i=2}^{n} |\gamma_i|\,|\lambda_i|^l\,|1-(1-\lambda_i)p(1-\lambda_i)|\right). \end{aligned} \tag{3.3}$$

On the other hand, for the denominator of Eq (3.2), it has

$$\|\gamma_1\mu_1\|_G \geq \sqrt{\min_{1\leq i\leq n}\delta_i} \cdot \|\gamma_1\mu_1\|_2 \geq \sqrt{\min_{1\leq i\leq n}\delta_i} \cdot |\gamma_1|\,. \tag{3.4}$$

Substituting Eqs (3.3) and (3.4) into (3.2), and let $q(\lambda) = 1 - (1-\lambda)p(1-\lambda)$, we have

$$\begin{aligned}
|\sin\angle(u,\mu_1)| &\le \frac{\sqrt{\max_{1\le i\le n}\delta_i}\cdot\left(\sum_{i=2}^{n}|\gamma_i|\,|\lambda_i|^l\,|1-(1-\lambda_i)p(1-\lambda_i)|\right)}{\sqrt{\min_{1\le i\le n}\delta_i}\cdot|\gamma_1|}\\
&\le \frac{\sqrt{\max_{1\le i\le n}\delta_i}}{\sqrt{\min_{1\le i\le n}\delta_i}}\cdot\alpha^l\cdot\max_{i\ne 1}|1-(1-\lambda_i)p(1-\lambda_i)|\cdot\left(\sum_{i=2}^{n}\frac{|\gamma_i|}{|\gamma_1|}\right)\\
&\le \frac{\sqrt{\max_{1\le i\le n}\delta_i}}{\sqrt{\min_{1\le i\le n}\delta_i}}\cdot\alpha^l\cdot\left(\sum_{i=2}^{n}\frac{|\gamma_i|}{|\gamma_1|}\right)\cdot\min_{q\in\mathcal{P}_{m-1}}\max_{\lambda\in\Lambda(A)/\lambda_1}|q(\lambda)|\\
&= \frac{\sqrt{\max_{1\le i\le n}\delta_i}}{\sqrt{\min_{1\le i\le n}\delta_i}}\cdot\alpha^l\cdot\left(\sum_{i=2}^{n}\frac{|\gamma_i|}{|\gamma_1|}\right)\cdot\varepsilon_1,
\end{aligned}$$

where we used the result in Theorem 1, and $|\lambda_n| \le \cdots \le |\lambda_2| \le \alpha$. □

3.2. The GFOM-Extrapolation algorithm for computing PageRank

In this subsection, we describe the GFOM-Extrapolation algorithm and discuss its convergence.

3.2.1. The GFOM-Extrapolation algorithm

The extrapolation procedure is based on the power iterations and Ritz values. Assume that the initial approximation x_0 can be written as the following linear combination:

$$x_0 = \mu_1 + a_2\mu_2 + a_3\mu_3,$$

where μ_1, μ_2, μ_3 are the first three largest eigenvectors of the Google matrix A corresponding to λ_1, λ_2 and λ_3, respectively. After two power iterations, it has

$$x_1 = Ax_0 = \mu_1 + a_2\lambda_2\mu_2 + a_3\lambda_3\mu_3,$$

$$x_2 = Ax_1 = \mu_1 + a_2\lambda_2^2\mu_2 + a_3\lambda_3^2\mu_3.$$

Then it has

$$\mu_1 = (x_2 - (\lambda_2+\lambda_3)x_1 + \lambda_2\lambda_3x_0)/(1-\lambda_2)(1-\lambda_3). \tag{3.5}$$

By normalizing μ_1, an improved approximation to the PageRank vector can be computed as

$$x^{\mathrm{Ex}} = \mu_1/\|\mu_1\|_1. \tag{3.6}$$

One advantage of the extrapolation procedure is that the unknown eigenvalues λ_2 and λ_3 of the Google matrix A can be estimated along with the step 4 in Algorithm 2. It is well-known that the Ritz values are defined as the eigenvalues of the upper Hessenberg matrix H_m and they can be used to approximate the true eigenvalues of the cofficient matrix $I - A$. Let $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ be the second and third largest eigenvalues of H_m, then we make a simple transformation,

$$\lambda_2 \doteq 1 - \tilde{\lambda}_2, \quad \lambda_3 \doteq 1 - \tilde{\lambda}_3.$$

Since $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ may be complex valued, the following classifications are made, please refer to [25,26] for details.
Case 1. If both $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ are real, or $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ are conjugate, then x^{Ex} is computed as Eq (3.6).
Case 2. If $\tilde{\lambda}_2$ is real but $\tilde{\lambda}_3$ is complex, we assume that $x_0 = \mu_1 + a_2\mu_2$ and $x_1 = Ax_0 = \mu_1 + a_2\lambda_2\mu_2$, then x^{Ex} is computed as

$$x^{\mathrm{Ex}} = (x_1 - \lambda_2 x_0)/\|x_1 - \lambda_2 x_0\|_1. \tag{3.7}$$

Case 3. If $\tilde{\lambda}_2$ is complex but $\tilde{\lambda}_3$ is real, or both $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ are complex but not conjugate, then x^{Ex} is computed as

$$x^{\mathrm{Ex}} = (x_1 - \alpha x_0)/\|x_1 - \alpha x_0\|_1. \tag{3.8}$$

Now we accelerate the GFOM algorithm with the extrapolation procedure based on Ritz values for computing PageRank. The specific implementation is given as follows.

Algorithm 4. The GFOM-Extrapolation algorithm for computing PageRank

1. Given a unit positive initial guess x_0, the steps m of the GArnoldi process, a prescribed tolerance *tol*, and the number for applying extrapolation procedure *maxnum*.
2. Run Algorithm 2 for a few times (2–3 times): iterate steps 1-11 for the first run and steps 4–11 otherwise. If the residual norm satisfies the prescribed tolerance *tol*, then stop, else continue.
3. Compute the eigenvalues of H_m obtained from Algorithm 2, and select the second and third largest Ritz values: $\tilde{\lambda}_2$, $\tilde{\lambda}_3$.
4. Set $\lambda_2 = 1 - \tilde{\lambda}_2$, $\lambda_3 = 1 - \tilde{\lambda}_3$.
5. Run the extrapolation procedure with $x_0 = \tilde{x}/\|\tilde{x}\|_1$ as the initial guess, where $\tilde{x}$ is the approximation vector obtained from the Algorithm 2:
5.1. for $i = 1 : maxnum$
5.2. $x_1 = Ax_0$
5.3. $x_2 = Ax_1$
5.4. if case 1 is satisfied
5.5. $x^{\mathrm{Ex}} = (x_2 - (\lambda_2 + \lambda_3)x_1 + \lambda_2\lambda_3 x_0)/\|x_2 - (\lambda_2 + \lambda_3)x_1 + \lambda_2\lambda_3 x_0\|_1$
5.6. else the case 2
5.7. $x^{\mathrm{Ex}} = (x_1 - \lambda_2 x_0)/\|x_1 - \lambda_2 x_0\|_1$
5.8. else the case 3
5.9. $x^{\mathrm{Ex}} = (x_1 - \alpha x_0)/\|x_1 - \alpha x_0\|_1$
5.10. end if
5.11. Compute $r = x^{\mathrm{Ex}} - x_0$
5.12. Set $x_0 = x^{\mathrm{Ex}}$
5.13. end for
5.14. if $\|r\|_2 \le tol$, stop, else set $G = \mathrm{diag}\{|r|/\|r\|_1\}$ and goto step 2

Note that the choice of the number *maxnum* in the line 5.1 of Algorithm 4 would be discussed in Section 4.

3.2.2. Convergence analysis of the GFOM-Extrapolation algorithm

In this subsection, we analyze the convergence of the GFOM-Extrapolation algorithm. In particular, our analysis focuses on the procedure when turning from the extrapolation procedure to the GFOM algorithm.

As shown in [41], the second largest eigenvalue λ_2 of the Google matrix A is semi-simple. Here we assume that the third largest eigenvalue λ_3 is also semi-simple and A is diagonalizable. The following theorem shows the form of the new PageRank vector obtained after the extrapolation procedure in Algorithm 4.

Theorem 3. Let $\tilde{x}$ be the initial vector for the extrapolation procedure, which is obtained from the previous GFOM algorithm. Assume that $\tilde{x}$ can be expressed as $\tilde{x} = \mu_1 + \sum_{i=2}^{n} a_i\mu_i$ with respect to the eigenbasis $\{\mu_i\}_{i=1,2,\ldots,n}$, in which $\|\mu_i\|_2 = 1$, $i = 1, 2, \ldots, n$. Then the new initial vector which is obtained from the extrapolation procedure for the next GFOM algorithm is

$$x^{\mathrm{Ex}} = c^{-1}\left\{(1-\lambda_2)(1-\lambda_3)\mu_1 + \sum_{i=4}^{n} a_i\lambda_i^f[\lambda_i^2 - (\lambda_2+\lambda_3)\lambda_i + \lambda_2\lambda_3\lambda_i]\mu_i\right\}, \tag{3.9}$$

where $c = \|x_2 - (\lambda_2 + \lambda_3)x_1 + \lambda_2\lambda_3 x_0\|_2$ is the scaling factor and f is the number for applying the extrapolation procedure *maxnum*.

Proof. The proof of Theorem 3 is totally similar to the convergence of the Arnoldi-Extrapolation algorithm and the FOM-Extrapolation algorithm, please refer to [25, 26] for details. □

Theorem 4. Under the above assumptions and set $S = [\lambda_1, \lambda_2, \lambda_3]$. Then the relationship between the next approximate vector u and μ_1 holds asymptotically that

$$|\sin\angle(u,\mu_1)| \le \frac{\sqrt{\max_{1\le i\le n}\delta_i}}{\sqrt{\min_{1\le i\le n}\delta_i}} \cdot \xi \cdot \alpha^f \cdot \varepsilon_2,$$

where $\xi = \frac{\sum_{i=4}^{n}|a_i||\lambda_i^2-(\lambda_2+\lambda_3)\lambda_i+\lambda_2\lambda_3|}{|(1-\lambda_2)(1-\lambda_3)|}$ and $\varepsilon_2 = \min\limits_{q\in\mathcal{P}_{m-1}} \max\limits_{\lambda\in\Lambda(A)/S} |q(\lambda)|$ with $q(\lambda) = 1-(1-\lambda)p(1-\lambda)$.

Proof. According to Algorithm 4 and Theorem 3, after the extrapolation procedure, the residual vector of the approximate vector x^{Ex} is computed as $r^{\mathrm{Ex}} = -(I-A)x^{\mathrm{Ex}}$, which is used as the initial residual vector to run the next GFOM algorithm. Then we get the next approximation $u \in \mathcal{K}_m(I-A, r^{\mathrm{Ex}})$, and there exists a polynomial $p \in \mathcal{P}_{m-1}$ so that

$$\begin{aligned} u &= x^{\mathrm{Ex}} + p(I-A)r^{\mathrm{Ex}} \\ &= [I-(I-A)p(I-A)]x^{\mathrm{Ex}} \\ &= c^{-1}\left\{(1-\lambda_2)(1-\lambda_3)\mu_1 + \sum_{i=4}^{n} a_i\lambda_i^f[\lambda_i^2-(\lambda_2+\lambda_3)\lambda_i+\lambda_2\lambda_3\lambda_i][1-(1-\lambda_i)p(1-\lambda_i)]\mu_i\right\}, \end{aligned}$$

where we used the facts that $\lambda_1 = 1$, $A\mu_1 = \mu_1$ and $A\mu_i = \lambda_i\mu_i$ $(i = 4, \ldots, n)$.

Then we have

$$|\sin\angle(u,\mu_1)| \le \frac{\left\|\sum_{i=4}^{n} a_i\lambda_i^f[\lambda_i^2-(\lambda_2+\lambda_3)\lambda_i+\lambda_2\lambda_3][1-(1-\lambda_i)p(1-\lambda_i)]\mu_i\right\|_G}{\|(1-\lambda_2)(1-\lambda_3)\mu_1\|_G}. \tag{3.10}$$

Now, using Eq (3.1), for the numerator of Eq (3.10), it has

$$\begin{aligned}
&\left\| \sum_{i=4}^{n} a_i \lambda_i^f [\lambda_i^2 - (\lambda_2 + \lambda_3)\lambda_i + \lambda_2\lambda_3][1 - (1-\lambda_i)p(1-\lambda_i)]\mu_i \right\|_G \\
&\leq \sqrt{\max_{1\leq i\leq n} \delta_i} \cdot \left\| \sum_{i=4}^{n} a_i \lambda_i^f [\lambda_i^2 - (\lambda_2 + \lambda_3)\lambda_i + \lambda_2\lambda_3][1 - (1-\lambda_i)p(1-\lambda_i)]\mu_i \right\|_2 \\
&\leq \sqrt{\max_{1\leq i\leq n} \delta_i} \cdot \left(\sum_{i=4}^{n} |a_i|\,|\lambda_i|^f\,|1 - (1-\lambda_i)p(1-\lambda_i)|\,|\lambda_i^2 - (\lambda_2 + \lambda_3)\lambda_i + \lambda_2\lambda_3| \right). \qquad (3.11)
\end{aligned}$$

On the other hand, for the denominator of Eq (3.10), it has

$$\begin{aligned}
\|(1-\lambda_2)(1-\lambda_3)\mu_1\|_G &\geq \sqrt{\min_{1\leq i\leq n} \delta_i} \cdot \|(1-\lambda_2)(1-\lambda_3)\mu_1\|_2 \\
&\geq \sqrt{\min_{1\leq i\leq n} \delta_i} \cdot |(1-\lambda_2)(1-\lambda_3)|\,. \qquad (3.12)
\end{aligned}$$

Substituting Eqs (3.11) and (3.12) into (3.10), and let $q(\lambda) = 1 - (1-\lambda)p(1-\lambda)$, we have

$$\begin{aligned}
|\sin \angle(u, \mu_1)| &\leq \frac{\sqrt{\max_{1\leq i\leq n} \delta_i} \cdot \left(\sum_{i=4}^{n} |a_i|\,|\lambda_i|^f\,|1 - (1-\lambda_i)p(1-\lambda_i)|\,|\lambda_i^2 - (\lambda_2 + \lambda_3)\lambda_i + \lambda_2\lambda_3| \right)}{\sqrt{\min_{1\leq i\leq n} \delta_i} \cdot |(1-\lambda_2)(1-\lambda_3)|} \\
&\leq \frac{\sqrt{\max_{1\leq i\leq n} \delta_i}}{\sqrt{\min_{1\leq i\leq n} \delta_i}} \cdot \alpha^f \cdot \max_{4\leq i\leq n} |1 - (1-\lambda_i)p(1-\lambda_i)| \cdot \frac{\sum_{i=4}^{n} |a_i|\,|\lambda_i^2 - (\lambda_2 + \lambda_3)\lambda_i + \lambda_2\lambda_3|}{|(1-\lambda_2)(1-\lambda_3)|} \\
&\leq \frac{\sqrt{\max_{1\leq i\leq n} \delta_i}}{\sqrt{\min_{1\leq i\leq n} \delta_i}} \cdot \xi \cdot \alpha^f \cdot \min_{q\in\mathcal{P}_{m-1}} \max_{\lambda\in\Lambda(A)/S} |q(\lambda)| \\
&= \frac{\sqrt{\max_{1\leq i\leq n} \delta_i}}{\sqrt{\min_{1\leq i\leq n} \delta_i}} \cdot \xi \cdot \alpha^f \cdot \varepsilon_2,
\end{aligned}$$

where we also used the result in Theorem 1, and $|\lambda_n| \leq \cdots \leq |\lambda_2| \leq \alpha$. □

4. Numerical experiments

In this section, we present numerical experiments to verify the effectiveness of the proposed algorithms: the GFOM-Power algorithm and the GFOM-Extrapolation algorithm in terms of the computing time (CPU) in seconds and the number of matrix-vector products (Mv). All the numerical results are obtained by using MATLAB 2019a on the Windows 10 64 bit operating system with 2.40 GHz Intel(R) Core(TM) i7-5500U CPU and RAM 8.00 GB.

The characteristic of test matrices is listed in Table 2, where n stands for the matrix size, nnz denotes the number of nonzero elements, $numd$ is the number of dangling nodes and den is the density which is defined by $den = \frac{nnz}{n\times n} \times 100$. All test matrices are available from [39]. For the sake of justice, all algorithms use the same initial guess $x_0 = e/n$ with $e = [1, 1, \ldots, 1]^{\mathrm{T}}$ and the same tolerance $tol = 10^{-8}$. And the damping factors are set as $\alpha = 0.99, 0.993, 0.995$ and 0.997, respectively.

Table 2. The characteristic of test matrices.

Name	n	nnz	$numd$	den
California	9664	5027	4637	0.538×10^{-1}
wb-cs-stanford	9914	36,854	2861	0.375×10^{-1}
soc-Epinions1	75,888	60,341	15,547	0.105×10^{-2}
flickr	820,878	9,837,214	265,189	0.146×10^{-2}
eu-2005	862,664	790,989	71,675	0.106×10^{-3}
in-2004	1,382,908	1,100,602	282,306	0.575×10^{-4}
wikipedia-20051105	1,634,989	19,753,078	72,556	0.739×10^{-3}
wiki-Talk	2,394,385	5,021,410	2,246,783	0.876×10^{-4}

4.1. Numerical comparisons for the GFOM-Power algorithm

In this subsection, we test the effectiveness of the GFOM-Power algorithm (denoted as "GFOM-P"), and compare it with the the Power-Arnoldi algorithm (denoted as "PA") [22], the Arnoldi-Chebyshev algorithm (denoted as "AC") [32] as well as the GFOM algorithm (denoted as "GFOM") as given in Algorithm 2. The test matrices are soc-Epinions1, eu-2005, in-2004 and wikipedia-20051105. We define

$$\mathrm{Spe}_1 = \frac{\mathrm{CPU}_{\mathrm{GFOM}} - \mathrm{CPU}_{\mathrm{GFOM\text{-}P}}}{\mathrm{CPU}_{\mathrm{GFOM}}} \times 100\%.$$

to record the speedup of the GFOM-Power algorithm with respect to the GFOM algorithm in terms of CPU time. Note that, in the Power-Arnoldi algorithm, we run the thick restarted Arnoldi procedure two times per cycle with the number of approximate eigenpairs $k = 6$. In the GFOM-Power algorithm, we run the GFOM procedure also two time per cycle. And in the Arnoldi-Chebyshev algorithm, we set the Chebyshev steps $l = 10$.

4.1.1. The choice of the number *maxit*

In this subsection, we show the choice of the number *maxit* according to the numerical performance of the Power-Arnoldi algorithm and the GFOM-Power algorithm for the wb-cs-stanford matrix. Table 3 lists the number of matrix-vector products of the Power-Arnoldi algorithm and the GFOM-Power algorithm when the damping factor $\alpha = 0.99, 0.993, 0.995, 0.997$ and the number *maxit* varies from 6 to 9, respectively. Figure 3 depicts the total CPU time of the Power-Arnoldi algorithm and the GFOM-Power algorithm versus different number *maxit*.

From Table 3, we find that the number of matrix-vector products of the Power-Arnoldi algorithm and the GFOM-Power algorithm is first decreasing and then increasing as the number *maxit* increases when the damping factor $\alpha = 0.99$ and 0.993. As for $\alpha = 0.995$ and 0.997, it shows that the number of matrix-vector products of the Power-Arnoldi algorithm and the GFOM-Power algorithm is relatively small when the number $maxit = 8$. At the same time, from Figure 3, we observe that the optimal number *maxit* to the minimum CPU time of the Power-Arnoldi algorithm and the GFOM-Power algorithm is different for different damping factor α. In the left picture of Figure 3, we find that the total CPU time of the Power-Arnoldi algorithm is less when $maxit = 8$. In the right picture of Figure 3, it seems that both the number $maxit = 8$ and 9 are feasible. Therefore, as a compromise choice, $maxit = 8$ is selected in the following numerical experiments.

Table 3. The number of matrix-vector products of the Power-Arnoldi algorithm and the GFOM-Power algorithm with different damping factor and number *maxit* for the wb-cs-stanford matrix.

α	*maxit* = 6		*maxit* = 7		*maxit* = 8		*maxit* = 9	
	PA	GFOM-P	PA	GFOM-P	PA	GFOM-P	PA	GFOM-P
$\alpha = 0.99$	148	153	139	130	145	174	149	192
$\alpha = 0.993$	183	350	159	167	176	161	181	190
$\alpha = 0.995$	212	205	256	190	196	189	210	177
$\alpha = 0.997$	235	238	267	234	224	231	243	209

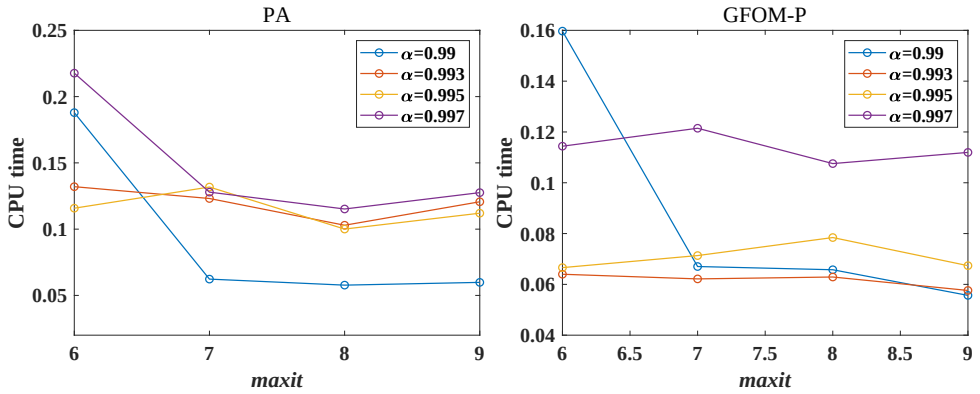

Figure 3. The total CPU time of the Power-Arnoldi algorithm and the GFOM-Power algorithm versus different number *maxit* for the wb-cs-stanford matrix.

4.1.2. Numerical comparisons

In this subsection, Table 4 lists numerical results of the GFOM algorithm, the Power-Arnoldi algorithm, the Arnoldi-Chebyshev algorithm and the GFOM-Power algorithm for these four test matrices when $\alpha = 0.99, 0.993, 0.995$ and 0.997, respectively.

From Table 4, we can observe that,

- The GFOM-Power algorithm makes great improvements on the GFOM algorithm in terms of the number of matrix-vector products and the CPU time for these four test matrices with different damping factor α. Particularly, the reduction proportion of the CPU time is up to 53.76% for the soc-Epinions1 matrix with $\alpha = 0.993$, which shows that using power method to accelerate the GFOM algorithm is meaningful.
- The GFOM-Power algorithm also outperforms the Power-Arnoldi algorithm and the Arnoldi-Chebyshev algorithm in terms of the number of matrix-vector products and the CPU time for these four test matrices in most cases. Hence, we can say that the GFOM-Power algorithm is more efficient than the Power-Arnoldi algorithm and the Arnoldi-Chebyshev algorithm, especially when the damping factor α is high.

Figure 4 plots the convergence behavior of the GFOM algorithm, the Power-Arnoldi algorithm, the Arnoldi-Chebyshev algorithm and the GFOM-Power algorithm for the soc-Epinions1 matrix when $\alpha =$

Table 4. Numerical comparisons of the GFOM algorithm, the Power-Arnoldi algorithm, the Arnoldi-Chebyshev algorithm and the GFOM-Power algorithm for these four test matrices, where "∼" denotes stagnation.

		GFOM		PA		AC		GFOM-P		Spe_1
	α	Mv	CPU	Mv	CPU	Mv	CPU	Mv	CPU	
soc-Epinions1	0.99	126	0.5695	118	0.3991	151	0.3545	100	0.2755	51.62%
	0.993	153	0.5443	118	0.3103	151	0.3070	104	0.2516	53.76%
	0.995	162	0.5183	161	0.4407	171	0.3379	138	0.3837	26.20%
	0.997	234	0.7594	198	0.6292	291	0.5857	151	0.3995	47.39%
eu-2005	0.99	171	12.9438	215	11.9705	271	12.5739	168	9.4228	27.20%
	0.993	225	15.0967	218	12.7750	320	14.8302	198	12.0650	20.08%
	0.995	315	23.1987	288	17.5105	391	17.5914	211	12.6781	45.35%
	0.997	450	32.5724	∼	∼	551	24.9951	295	18.9743	41.75%
in-2004	0.99	342	31.6287	535	46.8129	851	48.0653	284	23.8216	24.68%
	0.993	378	46.1470	731	78.5646	1191	66.2892	345	34.7248	24.75%
	0.995	423	48.4014	1051	113.7902	1591	88.8401	377	35.9242	25.78%
	0.997	585	70.8687	1531	164.7577	2010	112.7999	486	47.3361	33.21%
wikipedia-20051105	0.99	126	23.0604	135	19.5614	151	32.1638	76	12.6008	45.36%
	0.993	162	32.3998	136	21.1544	191	40.5334	119	20.5868	36.46%
	0.995	207	42.5474	165	27.1784	231	50.9236	119	21.3030	49.93%
	0.997	243	48.4092	206	33.5083	351	84.0431	144	22.9271	52.64%

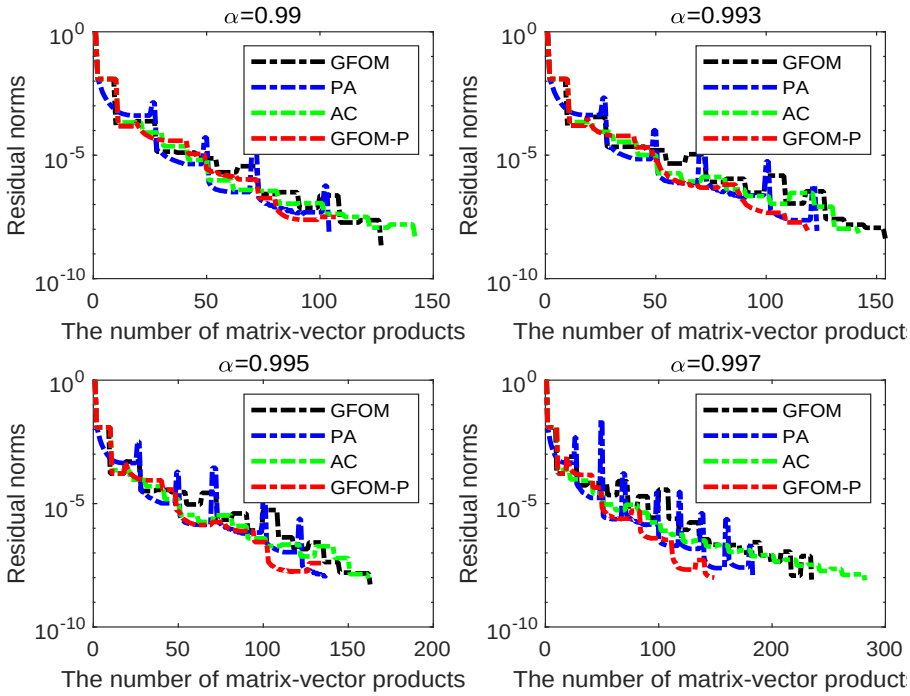

Figure 4. Convergence behavior of the GFOM algorithm, the Power-Arnoldi algorithm, the Arnoldi-Chebyshev algorithm and the GFOM-Power algorithm for the soc-Epinions1 matrix.

0.99, 0.993, 0.995 and 0.997, respectively. It shows that the GFOM-Power algorithm converges faster

than the other two algorithms. Similar convergence behaviors can be found for the other three test matrices, so we do not plot them for simplicity.

4.2. Numerical comparisons for the GFOM-Extrapolation algorithm

In this subsection, we test the effectiveness of the GFOM-Extrapolation algorithm (denoted as "GFOM-EXT"), and compare it with the FOM-Extrapolation algorithm (denoted as "FOM-EXT") [26], the Arnoldi-Chebyshev algorithm [32] and the GFOM algorithm (denoted as "GFOM") as given in Algorithm 2. The test matrices are California, soc-Epinions1, flickr and wiki-Talk. We define

$$\mathrm{Spe}_2 = \frac{\mathrm{CPU}_{\mathrm{GFOM}} - \mathrm{CPU}_{\mathrm{GFOM\text{-}EXT}}}{\mathrm{CPU}_{\mathrm{GFOM}}} \times 100\%.$$

to record the speedup of the GFOM-Extrapolation algorithm with respect to the GFOM algorithm in terms of CPU time. Note that we run the GFOM procedure two times per cycle in the GFOM-Extrapolation algorithm.

4.2.1. The choice of the number *maxnum*

In this subsection, we show the choice of the number *maxnum* by analyzing the numerical behavior of the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm for the wb-cs-stanford matrix. Table 5 lists the number of matrix-vector products for the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm when the damping factor $\alpha = 0.99, 0.993, 0.995, 0.997$ and the number $maxnum = 2, 3, 4, 5$, respectively. Figure 5 depicts the total CPU time of the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm versus different number *maxnum*.

Table 5. The number of matrix-vector products of the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm with different damping factor and number *maxnum* for the wb-cs-stanford matrix.

	FOM-EXT				GFOM-EXT			
α / *maxnum*	0.99	0.993	0.995	0.997	0.99	0.993	0.995	0.997
maxnum = 2	217	279	341	461	182	213	217	304
maxnum = 3	159	198	231	297	159	192	192	288
maxnum = 4	167	202	210	272	167	167	202	245
maxnum = 5	175	212	249	360	148	185	296	259

From Table 5, it observes that the number of matrix-vector products of the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm is first decreasing and then increasing as the number *maxnum* increases in most cases. Meanwhile, from Figure 5, we find that the CPU time of the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm has a similar trend for different damping factor α in most cases, i.e., the CPU time is relatively less when $maxnum = 4$. Hence, we choose $maxnum = 4$ in the following numerical experiments.

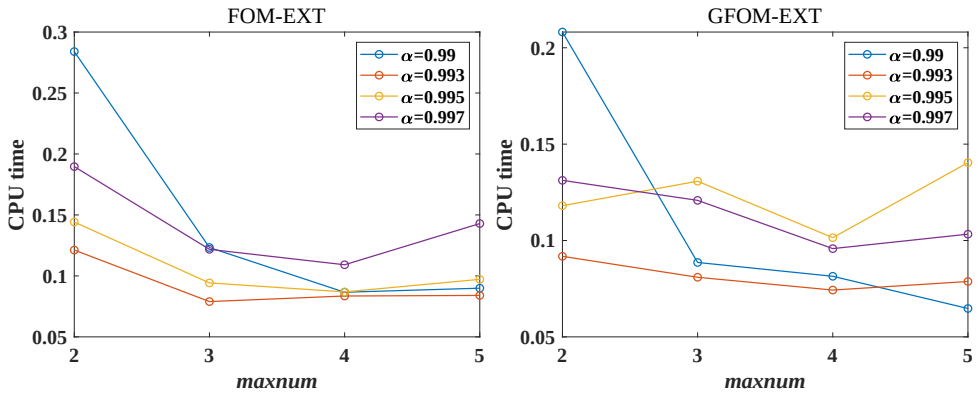

Figure 5. The total CPU time of the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm versus different number *maxnum* for the wb-cs-stanford matrix.

4.2.2. Numerical comparisons

In this subsection, Table 6 lists numerical results of the GFOM algorithm, the Arnoldi-Chebyshev algorithm, the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm for these four test matrices when α = 0.99, 0.993, 0.995 and 0.997, respectively.

Table 6. Numerical comparisons of the GFOM algorithm, the Arnoldi-Chebyshev algorithm, the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm for these four test matrices.

		GFOM		AC		FOM-EXT		GFOM-EXT		Spe_2
	α	Mv	CPU	Mv	CPU	Mv	CPU	Mv	CPU	
California	0.99	99	0.0872	131	0.0760	104	0.1081	70	0.0723	17.09%
	0.993	117	0.0547	140	0.0371	122	0.0682	70	0.0479	12.43%
	0.995	117	0.0598	160	0.0415	122	0.0599	70	0.0430	28.09%
	0.997	117	0.0482	171	0.0405	122	0.0480	96	0.0385	20.12%
soc-Epinions1	0.99	126	0.5695	151	0.3499	148	0.4959	122	0.4215	25.98%
	0.993	153	0.5443	151	0.3119	156	0.4356	130	0.4169	23.41%
	0.995	162	0.5183	171	0.3330	200	0.6044	156	0.4591	11.42%
	0.997	234	0.7594	291	0.5868	252	0.7025	174	0.5378	29.06%
flickr	0.99	126	9.0932	160	6.8268	174	11.0006	100	5.9112	34.99%
	0.993	144	9.6250	200	20.2166	226	13.0329	118	7.1573	25.64%
	0.995	189	12.1234	231	23.4411	252	14.8641	144	8.4009	30.71%
	0.997	234	16.6600	280	29.0787	304	18.5578	166	10.1311	39.20%
wiki-Talk	0.99	90	12.5757	151	33.3121	102	10.2323	52	5.9181	52.94%
	0.993	108	16.1044	191	41.3051	119	12.7449	70	9.1337	43.28%
	0.995	117	17.4529	231	55.9126	153	16.4464	78	9.7445	44.17%
	0.997	171	28.5547	351	81.1511	204	22.0147	96	12.3118	56.90%

From Table 6, we find that,

- The GFOM-Extrapolation algorithm is superior to the GFOM algorithm in terms of the number

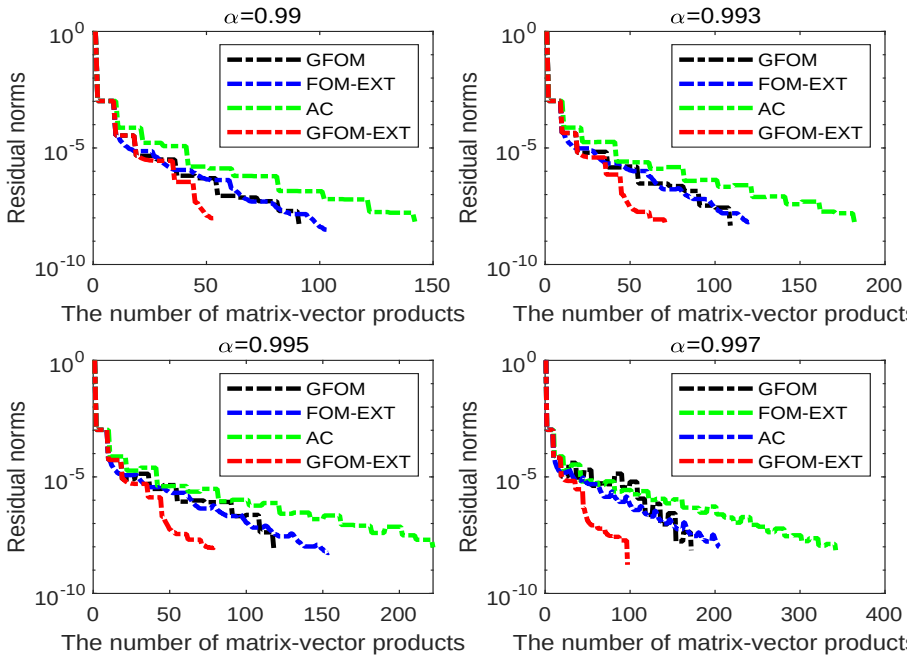

Figure 6. Convergence behavior of the GFOM algorithm, the Arnoldi-Chebyshev algorithm, the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm for the wiki-Talk matrix.

of matrix-vector products and the CPU time for these four test matrices with different damping factor α. Particularly, the speedup is up to 56.90% for the wiki-Talk matrix with $\alpha = 0.997$. Hence, we can say that it is meaningful to use the extrapolation procedure based on Ritz values to improve the GFOM algorithm.

- The GFOM-Extrapolation algorithm needs more CPU time than the Arnoldi-Chebyshev algorithm in some cases. However, in terms of the number of matrix-vector products, the GFOM-Extrapolation algorithm is superior to the Arnoldi-Chebyshev algorithm for all the test matrices.
- The GFOM-Extrapolation algorithm works better than the FOM-Extrapolation algorithm in terms of the number of matrix-vector products and the CPU time for these test matrices. Therefore, this suggests that the GFOM-Extrapolation algorithm has a faster convergence than the FOM-Extrapolation algorithm, which again illustrate that using weighted inner products in the standard Arnoldi process is significant.

Figure 6 plots the convergence behavior of the GFOM algorithm, the Arnoldi-Chebyshev algorithm, the FOM-Extrapolation algorithm and the GFOM-Extrapolation algorithm for the wiki-Talk matrix when $\alpha = 0.99$, 0.993, 0.995 and 0.997, respectively. It shows that the GFOM-Extrapolation algorithm converges faster than its counterparts. Note that the other three test matrices have the similar convergence behaviors.

In a word, all the above numerical experiments have shown that the GFOM-Power algorithm and the GFOM-Extrapolation algorithm are superior to their counterparts for computing PageRank when the damping factor α is close to 1.

5. Conclusions

In this paper, a generalized FOM (GFOM) algorithm based on weighted inner products is discussed for computing PageRank as given in Algorithm 2. With the aim at improving the convergence performance, we develop the GFOM-Power algorithm and the GFOM-Extrapolation algorithm by using the power method and the extrapolation method based on Ritz values as the accelerated techniques respectively. Their specific implementations and convergence analyses can be found in Section 3. Numerical experiments in Section 4 are presented to illustrate the efficiency of our proposed algorithms. Furthermore, the proposed algorithms are worth trying to solve other Markov problems, such as ProteinRank and CiteRank.

Acknowledgments

The authors are grateful to the anonymous referees for their much constructive comments and valuable suggestions, which greatly improved the original manuscript. This research is supported by the National Natural Science Foundation of China under grant 12101433, and the Two-Way Support Programs of Sichuan Agricultural University (Grant No.1921993077).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. L. Page, S. Brin, R. Motwami, T. Winograd, The PageRank citation ranking: Bringing order to the web, *Stanford Digital Library Technol. Proj.*, 1998. https://doi.org/10.1007/978-3-319-08789-4-10
2. I. C. Ipsen, T. M. Selee, PageRank computation, with special attention to dangling nodes, *SIAM J. Matrix Anal. Appl.*, **29** (2008), 1281–1296. https://doi.org/10.1137/060664331
3. A. Langville, C. Meyer, A survey of eigenvector methods for web information retrieval, *SIAM Rev.*, **47** (2005), 135–161. https://doi.org/10.1137/S0036144503424786
4. A. Langville, C. Meyer, Deeper inside PageRank, *Internet Math.*, **1** (2004), 335–380. https://doi.org/10.1080/15427951.2004.10129091
5. G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3^{rd} edition, The Johns Hopkins University Press, Baltimore, London, 1996. https://doi.org/10.1007/978-1-4612-5118-7-5
6. S. Kamvar, T. Haveliwala, C. Manning, G. Golub, Extrapolation methods for accelerating PageRank computations, in *Proceedings of the Twelfth Internatinal World Wide Web Conference*, ACM Press, New York, (2003), 261–270. https://doi.org/10.1145/775152.775190
7. A. Sidi, Vector extrapolation methods with applications to solution of large systems of equations and to PageRank computations, *Comput. Appl. Math.*, **56** (2008), 1–24. https://doi.org/10.1016/j.camwa.2007.11.027

8. X. Y. Tan, A new extrapolation method for PageRank computations, *J. Comput. Appl. Math.*, **313** (2017), 383–392. https://doi.org/10.1016/j.cam.2016.08.034
9. S. Cipolla, M. Redivo-Zaglia, F. Tudisco, Extrapolation methods for fixed-point multilinear PageRank computations, *Numer. Linear Algebra Appl.*, **27** (2020), e2280. https://doi.org/10.1002/nla.2280
10. D. Gleich, A. Gray, C. Greif, T. Lau, An inner-outer iteration for computing PageRank, *SIAM J. Sci. Comput.*, **32** (2010), 349–371. https://doi.org/10.1137/080727397
11. Z. Z. Bai, On convergence of the inner-outer iteration method for computing PageRank, *Numer. Algebra Control Optim.*, **2** (2012), 855–862. https://doi.org/10.3934/naco.2012.2.855
12. C. Q. Gu, F. Xie, K. Zhang, A two-step matrix splitting iteration for computing PageRank, *J. Comput. Appl. Math.*, **278** (2015), 19–28. https://doi.org/10.1016/j.cam.2014.09.022
13. C. Wen, T. Z. Huang, Z. L. Shen, A note on the two-step matrix splitting iteration for computing PageRank, *J. Comput. Appl. Math.*, **315** (2017), 87–97. https://doi.org/10.1016/j.cam.2016.10.020
14. Z. L. Tian, Y. Liu, Y. Zhang, Z. Y. Liu, M. Y. Tian, The general inner-outer iteration method based on regular splittings for the PageRank problem, *Appl. Math. Comput.*, **356** (2019), 479–501. https://doi.org/10.1016/j.amc.2019.02.066
15. J. F. Yin, G. J. Yin, M. Ng, On adaptively accelerated Arnoldi method for computing PageRank, *Numer. Linear Algebra Appl.*, **19** (2012), 73–85. https://doi.org/10.1002/nla.789
16. C. Wen, Q. Y. Hu, G. J. Yin, X. M. Gu, Z. L. Shen, An adaptive Power-GArnoldi algorithm for computing PageRank, *J. Comput. Appl. Math.*, **386** (2021), 113209. https://doi.org/10.1016/j.cam.2020.113209
17. C. Wen, Q. Y. Hu, B. Y. Pu, Y. Y. Huang, Acceleration of an adaptive generalized Arnoldi method for computing PageRank, *AIMS Math.*, **6** (2021), 893–907. https://doi.org/10.3934/math.2021053
18. H. D. Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, J. Ruge, Multilevel adaptive aggregation for Markov chains, with application to web ranking, *SIAM J. Sci. Comput.*, **30** (2008), 2235–2262. https://doi.org/10.1137/070685142
19. Z. L. Shen, T. Z. Huang, B. Carpentieri, C. Wen, X. M. Gu, Block-accelerated aggregation multigrid for Markov chains with application to PageRank problems, *Commun. Nonlinear. Sci. Numer. Simulat.*, **59** (2018), 472–487. https://doi.org/10.1016/j.cnsns.2017.11.031
20. G. H. Golub, C. Greif, An Arnoldi-type algorithm for computing PageRank, *BIT Numer. Math.*, **46** (2006), 759–771. https://doi.org/10.1007/s10543-006-0091-y
21. Z. X. Jia, Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems, *Linear Algebra Appl.*, **259** (1997), 1–23. https://doi.org/10.1016/S0024-3795(96)00238-8
22. G. Wu, Y. M. Wei, A Power-Arnoldi algorithm for computing PageRank, *Numer. Linear Algebra Appl.*, **14** (2007), 521–546. https://doi.org/10.1002/nla.531
23. R. B. Morgan, M. Zeng, A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity, *Linear Algebra Appl.*, **415** (2006), 96–113. https://doi.org/10.1016/j.laa.2005.07.024

24. Q. Y. Hu, C. Wen, T. Z. Huang, Z. L. Shen, X. M. Gu, A variant of the Power-Arnoldi algorithm for computing PageRank, *J. Comput. Appl. Math.*, **381** (2021), 113034. https://doi.org/10.1016/j.cam.2020.113034

25. G. Wu, Y. M. Wei, An Arnoldi-Extrapolation algorithm for computing PageRank, *J. Comput. Appl. Math.*, **234** (2010), 3196–3212. https://doi.org/10.1016/j.cam.2010.02.009

26. H. F. Zhang, T. Z. Huang, C. Wen, Z. L. Shen, FOM accelerated by an extrapolation method for solving PageRank problems, *J. Comput. Appl. Math.*, **296** (2016), 397–409. https://doi.org/10.1016/j.cam.2015.09.027

27. C. Q. Gu, X. L. Jiang, C. C. Shao, Z. B. Chen, A GMRES-Power algorithm for computing PageRank problems, *J. Comput. Appl. Math.*, **343** (2018), 113–123. https://doi.org/10.1016/j.cam.2018.03.017

28. Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, **7** (1986), 857–869. https://doi.org/10.1137/0907058

29. Z. L. Shen, T. Z. Huang, B. Carpentieri, X. M. Gu, C. Wen, An efficient elimination strategy for solving PageRank problems, *Appl. Math. Comput.*, **298** (2017), 111–122. https://doi.org/10.1016/j.amc.2016.10.031

30. Z. L. Shen, T. Z. Huang, B. Carpentieri, X. M. Gu, C. Wen, X. Y. Tan, Off-diagonal low-rank preconditioner for difficult PageRank problems, *J. Comput. Appl. Math.*, **346** (2019), 456–470. https://doi.org/10.1016/j.cam.2018.07.015

31. B. Y. Pu, T. Z. Huang, C. Wen, A preconditioned and extrapolation-accelerated GMRES method for PageRank, *Appl. Math. Lett.*, **37** (2014), 95–100. https://doi.org/10.1016/j.aml.2014.05.017

32. C. Q. Miao, X. Y. Tan, Accelerating the Arnoldi method via Chebyshev polynomials for computing PageRank, *J. Comput. Appl. Math.*, **377** (2020), 112891. https://doi.org/10.1016/j.cam.2020.112891

33. X. M. Gu, S. L. Lei, K. Zhang, Z. L. Shen, C. Wen, B. Carpentieri, A Hessenberg-type algorithm for computing PageRank problems, *Numer. Algorithms*, 2021. https://doi.org/10.1007/s11075-021-01175-w

34. Z. L. Shen, H. Yang, B. Carpentieri, X. M. Gu, C. Wen, A preconditioned variant of the refined Arnoldi method for computing PageRank eigenvectors, *Symmetry*, **13** (2021), 1327. https://doi.org/10.3390/sym13081327

35. Z. L. Tian, Y. Zhang, J. X. Wang, C. Q. Gu, Several relaxed iteration methods for computing PageRank, *J. Comput. Appl. Math.*, **388** (2021), 113295. https://doi.org/10.1016/j.cam.2020.113295

36. Z. L. Tian, Z. Y. Liu, Y. H. Dong, The coupled iteration algorithms for computing PageRank, *Numer. Algor.*, (2021), 1–15. https://doi.org/10.1007/s11075-021-01166-x

37. Y. H. Feng, J. X. You, Y. X. Dong, An extrapolation iteration and its lumped type iteration for computing PageRank, *Bull. Iran. Math. Soc.*, (2021), 1–8. https://doi.org/10.1007/s41980-021-00656-x

38. Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

39. SuiteSparse Matrix Collection, Available from: https://sparse.tamu.edu/.

40. T. Haveliwala, S. Kamvar, The second eigenvalue of the Google matrix, in *Proceedings of the Twelfth International World Wide Web of Conference*, 2003.

41. R. Horn, S. Serra-Capizzano, A general setting for the parametric Google matrix, *Internet Math.*, **3** (2008), 385–411. https://doi.org/10.1080/15427951.2006.10129131

AIMS Press