Numerical Algorithms (2023) 92:1213–1231
https://doi.org/10.1007/s11075-022-01337-4

ORIGINAL PAPER

An adaptively preconditioned multi-step matrix splitting iteration for computing PageRank

Chun Wen[1] · **Qian-Ying Hu**[2] · **Zhao-Li Shen**[3]

Received: 24 October 2021 / Accepted: 18 May 2022 / Published online: 30 June 2022

Abstract
The multi-step matrix splitting iteration (MPIO) for computing PageRank is an efficient iterative method by combining the multi-step power method with the inner-outer iterative method. In this paper, with the aim of accelerating the computation of PageRank problems, a new method is proposed by preconditioning the MPIO method with an adaptive generalized Arnoldi (GArnoldi) method. The new method is called as an adaptive GArnoldi-MPIO method, whose construction and convergence analysis are discussed in detail. Numerical experiments on several PageRank problems are reported to illustrate the effectiveness of our proposed method.

Keywords PageRank · Multi-step matrix splitting iteration · Generalized Arnoldi method · Power method · The inner-outer iteration

Mathematics Subject Classification (2010) 65F15 · 65F10

1 Introduction

In our world, web search engines have become one of the most commonly used tools for information retrieval. When we use a web search engine to search something, we not only hope to obtain the search results as soon as possible, but also hope to get

✉ Chun Wen
wchun17@163.com

[1] School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan, 610054, People's Republic of China

[2] School of Mathematical Sciences, Guizhou Normal University, Guiyang, 550025, People's Republic of China

[3] College of Science, Sichuan Agricultural University, Ya'an, Sichuan, 625000, People's Republic of China

the most relevant web pages. Hence, it is necessary to measure the importance of web pages and order them. Based on the hyperlink structure of web pages, Google's PageRank is regarded as an efficient method to determine the importance of web pages [1]. From the view of numerical solutions, it requires to solve the following linear system

$$Ax = x, \quad A = \alpha P + (1-\alpha)ve^{\mathrm{T}}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is called a Google matrix, $x \in \mathbb{R}^n$ is a PageRank vector, $\alpha \in (0, 1)$ is a damping factor, $P \in \mathbb{R}^{n\times n}$ is a column-stochastic matrix, $e = [1, 1, \cdots, 1]^{\mathrm{T}} \in \mathbb{R}^n$ and $v = e/n$.

It is well-known that the power method is a classical method for computing PageRank. When the damping factor is small such as $\alpha = 0.85$, the power method has a fast convergence. On the contrary, if the damping factor is large such as $\alpha \geq 0.99$, then the power method suffers from slow convergence. In fact, the closer the damping factor α is to 1, the closer the Google matrix A is to the original web link graph. In other words, the PageRank vector derived from large α perhaps gives a "truer" PageRanking than small α [2–4]. Hence, it is meaningful to improve the power method for large values of α. Gu et al. [5] proposed a two-step matrix splitting iterative method (denoted as "PIO") for computing PageRank, where the power method is combined with the inner-outer iteration [6]. With this idea in mind, Wen et al. [7] presented a multi-step matrix splitting iterative method (called as "MPIO") by applying multi-step power method to combine with the inner-outer iteration. In addition, many strategies based on Arnoldi process are considered to speed up the power method. For instance, Wu and Wei [4] developed a Power-Arnoldi algorithm by periodically combining the power method with the thick restarted Arnoldi algorithm [8]. Hu et al. [9] proposed a variant of the Power-Arnoldi algorithm by using the power method with the extrapolation process based on trace (PET) [10]. Gu et al. [11] presented a GMRES-Power algorithm based on a periodic combination of the power method with the GMRES method [12, 13]. More numerical methods based on the Arnoldi process or the power method, please refer to [14–24].

Considering a weighted inner product into the Arnoldi process, Yin et al. [25] proposed an adaptive generalized Arnoldi (GArnoldi) method for computing PageRank. And then Wen et al. [26] developed an adaptive Power-GArnoldi algorithm by treating the adaptive GArnoldi method as an accelerated technique for the power method. Motivated by these works, we try to construct a new method by preconditioning the MPIO method with the adaptive GArnoldi method in this paper. One reason is that the MPIO method usually converges faster than the power method for computing PageRank [7]. Another reason is that the adaptive GArnoldi method with a weighted inner product can improve the robustness of the standard Arnoldi method with the Euclidean norm [25]. The new method is called as an adaptive GArnoldi-MPIO method, whose implementation and convergence would be analyzed in detail. It is worth noting that our new method is different from the Arnoldi-MSPI method in [19], since the latter used the thick restarted Arnoldi algorithm [8] to preprocess the multi-step splitting iteration.

The remainder of this paper is organized as follows. In Section 2, we briefly review the MPIO method and the adaptive GArnoldi method for computing PageRank. In Section 3, we give the construction of the adaptive GArnoldi-MPIO method and discuss its convergence. In Section 4, numerical experiments are used to illustrate the effectiveness of our proposed method. Finally, conclusions are presented in Section 5.

2 The MPIO iteration and the adaptive GArnoldi method for computing PageRank

In this section, we briefly review the MPIO iteration [7] and the adaptive GArnoldi method [25] for computing PageRank.

2.1 The MPIO iteration

According to the idea of the PIO iteration [5], Wen et al. [7] proposed a MPIO iteration by combining the multi-step power method with the inner-outer iteration. The MPIO iteration can be depicted as follows.

The MPIO iteration. Given an initial guess $x^{(0)}$. For $k = 0, 1, \cdots$, compute

$$\begin{cases} x^{\left(k+\frac{1}{m_1+1}\right)} = \alpha P x^{(k)} + (1-\alpha)v, \\ x^{\left(k+\frac{2}{m_1+1}\right)} = \alpha P x^{\left(k+\frac{1}{m_1+1}\right)} + (1-\alpha)v, \\ \cdots \\ x^{\left(k+\frac{m_1}{m_1+1}\right)} = \alpha P x^{\left(k+\frac{m_1-1}{m_1+1}\right)} + (1-\alpha)v, \\ (I-\beta P)x^{(k+1)} = (\alpha-\beta)P x^{\left(k+\frac{m_1}{m_1+1}\right)} + (1-\alpha)v, \end{cases} \tag{2}$$

until the sequence $\{x^{(k)}\}_{k=0}^{\infty}$ converges, where $\alpha \in (0, 1)$, $\beta \in (0, \alpha)$ and m_1 ($m_1 \geq 2$) is a multiple iteration parameter. Note that, if $m_1 = 1$, then the MPIO iteration is reduced to the PIO iteration [5].

From the construction of the MPIO iteration, we can see that the first m_1 steps of (2) are easy to implement since only matrix-vector products are used, while for the last step of (2), there is a computational problem when solving the linear system with $I - \beta P$. In order to overcome this problem, Gleich et al. [6] employed an inner Richardson iteration by setting

$$f = (\alpha-\beta)P x^{\left(k+\frac{m_1}{m_1+1}\right)} + (1-\alpha)v, \tag{3}$$

such that the inner linear system is defined as $(I - \beta P)y = f$. Then $x^{(k+1)}$ can be computed by the inner iteration

$$y^{(j+1)} = \beta P y^{(j)} + f, \quad j = 0, 1, 2, \cdots, l-1, \tag{4}$$

where $y^{(0)} = x^{\left(k+\frac{m_1}{m_1+1}\right)}$ and $y^{(l)} = x^{(k+1)}$.

For the whole iterations, the stopping criteria of the outer iteration (the last step of (2)) and the inner iteration (4) are set as

$$\|(1-\alpha)v - (I - \alpha P)x^{(k+1)}\|_2 < tol, \tag{5}$$

and

$$\|f - (I - \beta P)y^{(j+1)}\|_2 < \eta, \tag{6}$$

respectively, where tol and η are the prescribed tolerances. The corresponding algorithm of the MPIO iteration for computing PageRank is presented as follows [7, 19].

Algorithm 1 The MPIO iteration for computing PageRank.

Input: $P, \alpha, \beta, tol, \eta, v, m_1$
Output: PageRank vector x
1. $x = v$;
2. $z = Px$;
3. while $\|(1-\alpha)v + \alpha z - x\|_2 \geq tol$
4. for $i = 1 : m_1$, do
5. $x = \alpha z + (1-\alpha)v$;
6. $z = Px$;
7. end for
8. $f = (\alpha - \beta)z + (1-\alpha)v$;
9. repeat
10. $x = f + \beta z$;
11. $z = Px$;
12. until $\|f + \beta z - x\|_2 < \eta$
13. end while
14. $x = \alpha z + (1-\alpha)v$;

Note that, in Algorithm 1, we use the 2-norm of the residual as the stopping criterion for being consistent with the choices of the following methods.

2.2 The adaptive GArnoldi method

The Arnoldi process with weighted inner products, instead of the Euclidean norm, can be viewed as a generalization of the standard Arnoldi process. By changing the weights with the current residual vector corresponding to the approximate PageRank vector, Yin et al. [25] proposed an adaptive GArnoldi method for computing PageRank, which can be described as follows.

Algorithm 2 The adaptive GArnoldi method for computing PageRank.

Input: A, v, m, tol.
Output: PageRank vector x.
1. Set $G = I, x = v$.
2. For $l = 1, 2, \cdots$, until convergence,
3. Compute V_{m+1} and $H_{m+1,m}$ by using the GArnoldi process:
3.1. Compute $v_1 = x/\|x\|_G$.
3.2. for $j = 1, 2, \cdots, m$
3.3. $q = Av_j$;
3.4. for $i = 1, 2, \cdots, j$
3.5. $h_{i,j} = (q, v_i)_G, q = q - h_{i,j}v_i$;
3.6. end for
3.7. $h_{j+1,j} = \|q\|_G$;
3.8. if $h_{j+1,j} = 0$, break; end if
3.9. $v_{j+1} = q/h_{j+1,j}$;
3.10. end for
4. Compute a singular value decomposition $U\Sigma S^{\mathrm{T}} = H_{m+1,m} - [I; 0]^{\mathrm{T}}$.
5. Compute $x = V_m s_m, r = \sigma_m V_{m+1} u_m$.
6. If $\|r\|_2 < tol$, break; End If
7. Set $G = diag\{|r|/\|r\|_1\}$.
8. End For

Some remarks about Algorithm 2 are given as follows.

- In the first line, the input parameter A is the Google matrix as shown in (1), $v = e/n$ is used as an initial vector, m is the steps of the GArnoldi process and tol is a prescribed tolerance.
- In the step 3, there is a GArnoldi process, in which the matrix $G \in \mathbb{R}^{n\times n}$ is a symmetric positive define (SPD) matrix. In the line 3.5, there is a G-inner product defined as $(x, y)_G = x^{\mathrm{T}}Gy, \forall x \in \mathbb{R}^n, y \in \mathbb{R}^n$. Correspondingly, in the line 3.1 and 3.7, there is a G-norm defined as $\|x\|_G = \sqrt{(x, x)_G}, \forall x \in \mathbb{R}^n$. Noth that, when $G = I$, then the GArnoldi process reduces to the standard Arnoldi process with the Euclidean norm. The aim of step 3 is to obtain the matrix V_{m+1} and $H_{m+1,m}$, where $V_{m+1} = [v_1, v_2, \cdots, v_{m+1}] \in \mathbb{R}^{n\times(m+1)}$ is a G-orthogonal matrix and $H_{m+1,m} = (h_{ij}) \in \mathbb{R}^{(m+1)\times m}$ is an upper Hessenberg matrix. More details about the GArnoldi process can be found in [25].
- In the step 5, σ_m denotes the minimal singular value of the matrix $H_{m+1,m} - [I; 0]^{\mathrm{T}}$, s_m and u_m denotes the right and left singular vector associated with σ_m respectively. The matrix V_m consists of the first m columns of the matrix V_{m+1}.
- Since all SPD matrices are diagonalized, for simplicity, it is reasonable to set G as a diagonal matrix. As shown in the step 7, the matrix G is chosen as $G = diag\{|r|/\|r\|_1\}$, where r is the residual vector obtained from the step 5. It is worth mentioning that the residual vector r changes after every cycle of Algorithm 2, such that the matrix G, or the weights, is adaptively changed with the changing of the current residual vector.

3 The adaptive GArnoldi-MPIO method for computing PageRank

In this section, for accelerating the computations of PageRank problems, a new method is proposed by using the adaptive GArnoldi method as a preconditioner of the MPIO method. The new method is called as an adaptive GArnoldi-MPIO method. We first give its construction, and then discuss its convergence.

3.1 The adaptive GArnoldi-MPIO method

The construction of the adaptive GArnoldi-MPIO method is partially similar to the construction of these methods in [4, 11, 19, 26]. However, there are several obvious differences between our new method and the other methods. For example, comparing the adaptive GArnoldi-MPIO method with the Power-Arnoldi method [4], there are three main differences between them. The first one is that the aim of our new method is to accelerate the MPIO method, not the power method. The second one is that the former employs the adaptive GArnoldi method as a preconditioner, while the latter uses the thick restarted Arnoldi method. The last one is that our proposed method first runs the adaptive GArnoldi method for a few times such that an approximate vector is obtained, while the Power-Arnoldi method first runs the power method. Now we outline the steps of the adaptive GArnoldi-MPIO method for computing PageRank as follows.

Algorithm 3 The adaptive GArnoldi-MPIO method for computing PageRank.

1. Given an unit initial vector v, an inner tolerance η, a prescribed tolerance tol, the steps of the GArnoldi process m, a multiple iteration parameter m_1, and three parameters α_1, α_2 and $maxit$ to control the MPIO method. Set $restart = 0$, $\tau = 1$, $\tau_0 = \tau$, $\tau_1 = \tau$, $d = 1$, $d_0 = d$.
2. Run Algorithm 2 for a few times (2–3 times): iterate steps 1–8 for the first run and steps 2–8 otherwise. If the residual norm satisfies the prescribed tolerance, then stop, else continue.
3. Run the MPIO method with $\tilde{x}_1$ as the initial vector, where $\tilde{x}_1$ is the approximate vector obtained from the adaptive GArnoldi method:
3.1. $restart = 0$;
3.2. while $restart < maxit$ & $\tau \geq tol$
3.3. $x = \tilde{x}_1/\|\tilde{x}_1\|_1$; $z = Px$; $ratio = 0$;
3.4. while $ratio < \alpha_1$ & $\tau \geq tol$
3.5. for $i = 1 : m_1$
3.6. $x = \alpha z + (1-\alpha)v$;
3.7. $z = Px$;
3.8. end
3.9. $f = (\alpha - \beta)z + (1-\alpha)v$;
3.10. $ratio_1 = 0$;
3.11. while $ratio_1 < \alpha_2$ & $d \geq \eta$
3.12. $x = f + \beta z$
3.13. $z = Px$;

3.14. $d = \|f + \beta z - x\|_2$;
3.15. $ratio_1 = d/d_0$;
3.16. $d_0 = d$;
3.17. end
3.18. $r = \alpha z + (1-\alpha)v - x$;
3.19. $\tau = \|r\|_2$;
3.20. $ratio = \tau/\tau_0$;
3.21. $\tau_0 = \tau$;
3.22. end
3.23. $x = \alpha z + (1-\alpha)v$; $x = x/\|x\|_1$;
3.24. if $\tau/\tau_1 > \alpha_1$
3.25. $restart = restart + 1$
3.26. end
3.27. $\tau_0 = \tau$; $\tau_1 = \tau$;
3.28. end
3.29. Set $G = diag\{|r|/\|r\|_1\}$.
3.30. if $\tau < tol$, stop, else goto step 2.

According to Algorithm 3, the mechanism of the adaptive GArnoldi-MPIO method can be simply summarized as follows: given an unit initial vector v, we first run the adaptive GArnoldi method (Algorithm 2) for a few times (e.g., 2–3 times) to get an approximate PageRank vector. If the approximate PageRank vector is unsatisfactory, we use the resulting vector as the initial vector of the MPIO method to obtain another approximate PageRank vector. If this approximate PageRank vector is still below the prescribed tolerance, rerun the adaptive GArnoldi method. Repeating the above procedure analogously until the described accuracy is achieved.

In Algorithm 3, one problem is that when and how to control the conversion between the MPIO method and the adaptive GArnoldi method. To solve this problem, a simple and easily realized strategy as given in [19] is chosen. That is, the parameters $\alpha_1, \alpha_2, restart, maxit$ are used to control the flip-flop between the MPIO method and the adaptive GArnoldi method. Specifically, let τ^{curr} and τ^{pre} be the residual norm of the current and the previous MPIO method, respectively. Denote $ratio = \tau^{curr}/\tau^{pre}$. If $ratio > \alpha_1$, then let $restart = restart + 1$, terminate the MPIO method and run the adaptive GArnoldi method. Let d^{curr} and d^{pre} be the residual norm of the current and the previous inner iteration of the MPIO method, respectively. Denote $ratio_1 = d^{curr}/d^{pre}$. If $ratio_1 > \alpha_2$, then keep on running the inner iteration. In order to make sure the stability of our new method, it is important to set the values of α_1 and α_2. Since the largest eigenvalue of the Google matrix A is $\lambda_1 = 1$, and its second largest eigenvalue satisfies $|\lambda_2| \leq \alpha$ [27], it is reasonable to choose $\alpha_1 = \alpha - 0.1$ or $\alpha_1 = \alpha - 0.2$, and $\alpha_2 = \alpha - 0.1$ or $\alpha_2 = \alpha - 0.2$.

Now we consider the memory and the computational costs of the adaptive GArnoldi-MPIO method. According to the steps 2 and 3 of Algorithm 3, we find the main storage requirements are the G-orthogonal matrix V_{m+1}, the upper Hessenberg matrix $H_{m+1,m}$, the approximate PageRank vector x, the residual vector r, as well as the intermediate vectors z (line 3.7) and f (line 3.9). Thus the total memory cost of

Algorithm 3 is approximately $(m+5)n+\frac{m^2}{2}+2m$ in each cycle. Since the matrix G is chosen as a diagonal matrix, i.e., $G = diag\{|r|/\|r\|_1\}$, the G-inner product and G-norm in the GArnoldi process can be implemented by elementwise multiplication. So the main computational cost of our proposed method consists of the matrix-vector multiplications. For each cycle, it needs m matrix-vector multiplications in the GArnoldi process phase, and in the MPIO iteration phase, it requires m_1 matrix-vector multiplications for the power iteration (lines 3.5–3.8), while we do not know how many matrix-vector multiplications will be implemented for the inner-outer iteration (lines 3.11–3.17) because of the existence of the parameters η, $ratio_1$ and α_2. Thus the main computational cost of Algorithm 3 is m matrix-vector multiplications or at lease $m+m_1$ matrix-vector multiplications in each cycle.

3.2 Convergence analysis of the adaptive GArnoldi-MPIO method

In this subsection, we discuss the convergence analysis of the adaptive GArnoldi-MPIO method. Particularly, our analysis focuses on the procedure when turning from the MPIO iteration to the adaptive GArnodli method.

Assume that eigenvalues of the Google matrix A are arranged in decreasing order $1 = |\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Let $\mathcal{L}_{m-1}$ represent the set of polynomials of degree not exceeding $m-1$, $\sigma(A)$ denote the set of eigenvalues of the matrix A, $(\lambda_i, \varphi_i), i = 1, 2, \cdots, n$ and $(\widetilde{\lambda}_j, \widetilde{y}_j), j = 1, 2, \cdots, m$ denote the eigenpairs of A and H_m, respectively. The Arnoldi method usually uses $\widetilde{\lambda}_j$ to approximate λ_j, $\widetilde{\varphi}_j = V_m \widetilde{y}_j$ to approximate φ_j. However, for each $\widetilde{\lambda}_j$, instead of using $\widetilde{\varphi}_j$ to approximate φ_j, Jia [3] tried to seek a unit norm vector $\widetilde{u}_j \in \mathcal{K}_m(A, v_1)$ satisfying the condition

$$\|(A-\widetilde{\lambda}_j I)\widetilde{u}_j\|_2 = \min_{u\in\mathcal{K}_m(A,v_1)} \|(A-\widetilde{\lambda}_j I)u\|_2 \tag{7}$$

and use it to approximate φ_j, where $\mathcal{K}_m(A, v_1) = span(v_1, Av_1, \cdots, A^{m-1}v_1)$ is a Krylov subspace, and $\widetilde{u}_j$ is called a refined approximate eigenvector corresponding to λ_j. Convergence of the refined Arnoldi method is given below.

Theorem 1 [3]. *Under the above notations, assume that $v_1 = \sum_{i=1}^{n} \gamma_i \varphi_i$ with respect to the eigenbasis $\{\varphi_i\}_{i=1,2,\cdots,n}$ in which $\|\varphi_i\|_2 = 1, i = 1, 2, \cdots, n$ and $\gamma_i \neq 0$, let $S = [\varphi_1, \varphi_2, \cdots, \varphi_n]$, and*

$$\xi_j = \sum_{i\neq j} |\lambda_i - \widetilde{\lambda}_j| \cdot \frac{|\gamma_i|}{|\gamma_j|}.$$

Then

$$\|(A-\widetilde{\lambda}_j I)\widetilde{u}_j\|_2 \leq \frac{\sigma_{max}(S)}{\sigma_{min}(S)} \left(|\lambda_j - \widetilde{\lambda}_j| + \xi_j \min_{p\in\mathcal{L}_{m-1}, p(\lambda_j)=1} \max_{i\neq j} |p(\lambda_i)| \right),$$

where $\sigma_{max}(S)$ and $\sigma_{min}(S)$ are the largest and smallest singular value of the matrix S, respectively.

Before we give the convergence of the adaptive GArnoldi-MPIO method, a few useful conclusions are shown as follows.

Lemma 1 [26]. *Let $G = diag\{w_1, w_2, \cdots, w_n\}$, $w_i > 0$ $(1 \le i \le n)$, then for any vector $x \in \mathbb{R}^n$, we have*

$$\min_{1\le i\le n} w_i \cdot \|x\|_2^2 \le \|x\|_G^2 \le \max_{1\le i\le n} w_i \cdot \|x\|_2^2, \tag{8}$$

where $\|\cdot\|_2$ denotes the 2-norm and $\|\cdot\|_G$ denotes the G-norm.

Theorem 2 [27]. *Let 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 $A = \alpha P + (1-\alpha)E$ be an $n \times n$ column-stochastic matrix, then its dominant eigenvalue $\lambda_1 = 1$, $|\lambda_2| \le \alpha$.*

Theorem 3 [28]. *Assume that the spectrum of the column-stochastic matrix P is $\{1, \pi_2, \cdots, \pi_n\}$, then the spectrum of the matrix $A = \alpha P + (1-\alpha)ve^{\mathrm{T}}$ is $\{1, \alpha\pi_2, \cdots, \alpha\pi_n\}$, where $\alpha \in (0,1)$ and v is a vector with nonnegative elements such that $e^{\mathrm{T}}v = 1$.*

Since our analysis focuses on the procedure when turning from the MPIO iteration to the adaptive GArnodli method, it is necessary to derive the iterative formula of the MPIO method in Algorithm 3.

Lemma 2 *Let v_1 be the initial vector for the MPIO method, which is obtained from the previous adaptive GArnoldi method. Then, the MPIO method in Algorithm 3 produces the vector*

$$v_1^{new} = \omega T^k v_1, \tag{9}$$

where $k \ge maxit$, ω is a normalizing factor, and the iterative matrix T is expressed as

$$T = (I - \beta P)^{-1}\left[\alpha^{m_1-1}(\alpha-\beta)P^{m_1}A + (\alpha-\beta)PM_{m_1-2}(\alpha, P)(A-\alpha P) + (A-\alpha P)\right],$$

where

$$M_{m_1-2}(\alpha, P) = \alpha^{m_1-2}P^{m_1-2} + \alpha^{m_1-3}P^{m_1-3} + \cdots + \alpha P + I$$

sand m_1 is a multiple iteration parameter for the power iteration in the MPIO method.

Proof Let v_1 be the initial vector for the MPIO method, then it has $x^{(k)} = v_1$. According to (2), we have

$$\begin{aligned} x^{(k+1)} &= \alpha^{m_1}(\alpha-\beta)(I-\beta P)^{-1}P^{m_1+1}x^{(k)} + (I-\beta P)^{-1}(1-\alpha)v \\ &\quad +(\alpha-\beta)(I-\beta P)^{-1}P[(\alpha^{m_1-1}P^{m_1-1}+\alpha^{m_1-2}P^{m_1-2}+\cdots+\alpha P+I)(1-\alpha)v] \\ &= \alpha^{m_1-1}(\alpha-\beta)(I-\beta P)^{-1}P^{m_1}[\alpha P x^{(k)}+(1-\alpha)v]+(I-\beta P)^{-1}(1-\alpha)v \\ &\quad +(\alpha-\beta)(I-\beta P)^{-1}P[(\alpha^{m_1-2}P^{m_1-2}+\cdots+\alpha P+I)(1-\alpha)v] \\ &= \alpha^{m_1-1}(\alpha-\beta)(I-\beta P)^{-1}P^{m_1}[\alpha P+(1-\alpha)E]x^{(k)}+(I-\beta P)^{-1}(1-\alpha)Ex^{(k)} \\ &\quad +(\alpha-\beta)(I-\beta P)^{-1}PM_{m_1-2}(\alpha,P)(1-\alpha)Ex^{(k)} \\ &= (I-\beta P)^{-1}\left[\alpha^{m_1-1}(\alpha-\beta)P^{m_1}A+(\alpha-\beta)PM_{m_1-2}(\alpha,P)(A-\alpha P)+(A-\alpha P)\right]x^{(k)}, \end{aligned}$$

where we used the relationships separately $A = \alpha P + (1-\alpha)ve^{\mathrm{T}}$, $E = ve^{\mathrm{T}}$ and $e^{\mathrm{T}}x^{(k)} = 1$. Thus, the conclusion in Lemma 2 is proved. □

Remark 1 We need to indicate that our iterative matrix T in Lemma 2 is different from that in Lemma 3 of [19], more details please refer to it.

In the next cycle of the adaptive GArnoldi-MPIO method, v_1^{new} is used as an initial vector for an m-step GArnoldi process (step 2 in Algorithm 3), so that the new Krylov subspace

$$\mathcal{K}_m(A, v_1^{new}) = span(v_1^{new}, Av_1^{new}, \cdots, A^{m-1}v_1^{new})$$

will be constructed. The following theorem shows the convergence of the adaptive GArnoldi-MPIO method.

Theorem 4 *Under the above notations, assume that* $v_1 = \sum_{i=1}^{n}\gamma_i\varphi_i$ *with respect to the eigenbasis* $\{\varphi_i\}_{i=1,2,\cdots,n}$ *in which* $\|\varphi_i\|_2 = 1, i = 1, 2, \cdots, n$ *and* $\gamma_1 \neq 0$, *let* $S = [\varphi_1, \varphi_2, \cdots, \varphi_n]$, $G = diag\{w_1, w_2, \cdots, w_n\}$, $w_i > 0$ $(1 \le i \le n)$, *and*

$$\xi = \sum_{i=2}^{n}|\lambda_i - 1|\cdot\frac{|\gamma_i|}{|\gamma_1|}, \quad \zeta = \sqrt{\frac{\max_{1\le i\le n} w_i}{\min_{1\le i\le n} w_i}}.$$

Then

$$\|(A-I)u\|_G \le \left(\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}\right)^k \frac{\xi\cdot\zeta}{\sigma_{min}(S)} \min_{p\in\mathcal{L}_{m-1},p(\lambda_1)=1} \max_{\lambda\in\sigma(A)/\{\lambda_1\}} |p(\lambda)|,$$

where $u \in \mathcal{K}_m(A, v_1^{new})$, *and* $\sigma_{min}(S)$ *is the smallest singular value of the matrix* S.

Proof According to Theorem 3, let $\pi_1 = 1, \pi_2, \cdots, \pi_n$ be eigenvalues of the matrix P, then $\lambda_1 = 1, \lambda_2 = \alpha\pi_2, \cdots, \lambda_n = \alpha\pi_n$ are eigenvalues of the matrix $A = \alpha P + (1-\alpha)ve^{\mathrm{T}}$, and $\mu_1 = \frac{1}{1-\beta}, \mu_2 = \frac{1}{1-\beta\pi_2}, \cdots, \mu_n = \frac{1}{1-\beta\pi_n}$ are eigenvalues of

the matrix $(I-\beta P)^{-1}$. Such that we have

$$\begin{aligned} T\varphi_i &= (I-\beta P)^{-1}\left[\alpha^{m_1-1}(\alpha-\beta)P^{m_1}A+(\alpha-\beta)PM_{m_1-2}(\alpha,P)(A-\alpha P)+(A-\alpha P)\right]\varphi_i \\ &= \frac{\alpha^{m_1-1}(\alpha-\beta)\pi_i^{m_1}\lambda_i+(\alpha-\beta)\pi_i M_{m_1-2}(\alpha,\pi_i)(\lambda_i-\alpha\pi_i)+(\lambda_i-\alpha\pi_i)}{1-\beta\pi_i}\varphi_i, \end{aligned}$$

where $M_{m_1-2}(\alpha,\pi_i)=\alpha^{m_1-2}\pi_i^{m_1-2}+\cdots+\alpha\pi_i+1, i=1,2,\cdots,n.$

Assume that

$$\phi_i=\frac{\alpha^{m_1-1}(\alpha-\beta)\pi_i^{m_1}\lambda_i+(\alpha-\beta)\pi_i M_{m_1-2}(\alpha,\pi_i)(\lambda_i-\alpha\pi_i)+(\lambda_i-\alpha\pi_i)}{1-\beta\pi_i}, i=1,2,\cdots,n, \tag{10}$$

then, it has

$$\phi_1=1, T\varphi_1=\varphi_1, T^k\varphi_1=\varphi_1, T\varphi_i=\varphi_i, T^k\varphi_i=\phi_i^k\varphi_i, i=2,\cdots,n. \tag{11}$$

From the result in Theorem 2, we have $|\lambda_i|\le\alpha, i=2,\cdots,n$. For $i=2,\cdots,n$, substituting the relationship $\pi_i=\frac{\lambda_i}{\alpha}$ into (10), we get

$$|\phi_i|=\left|\frac{\alpha^{m_1-1}(\alpha-\beta)\pi_i^{m_1}\lambda_i}{1-\beta\pi_i}\right|=\left|\frac{(\alpha-\beta)\frac{1}{\alpha}\lambda_i^{m_1+1}}{1-\frac{\beta}{\alpha}\lambda_i}\right|\le\frac{(\alpha-\beta)\frac{1}{\alpha}|\lambda_i|^{m_1+1}}{1-\frac{\beta}{\alpha}|\lambda_i|}\le\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}. \tag{12}$$

Since for any $u\in\mathcal{K}_m(A,v_1^{new})$, there exists $q(x)\in\mathcal{L}_{m-1}$ such that

$$\begin{aligned} \|(A-I)u\|_G &= \min_{q\in\mathcal{L}_{m-1}}\frac{\|(A-I)q(A)v_1^{new}\|_G}{\|q(A)v_1^{new}\|_G}=\min_{q\in\mathcal{L}_{m-1}}\frac{\|(A-I)q(A)\omega T^kv_1\|_G}{\|q(A)\omega T^kv_1\|_G} \\ &= \min_{q\in\mathcal{L}_{m-1}}\frac{\|(A-I)q(A)T^k\gamma_1\varphi_1+\sum_{i=2}^n(A-I)q(A)T^k\gamma_i\varphi_i\|_G}{\|\sum_{i=1}^n q(A)T^k\gamma_i\varphi_i\|_G} \\ &= \min_{q\in\mathcal{L}_{m-1}}\frac{\|\sum_{i=2}^n(\lambda_i-1)q(\lambda_i)\phi_i^k\gamma_i\varphi_i\|_G}{\|\sum_{i=1}^n q(\lambda_i)\phi_i^k\gamma_i\varphi_i\|_G}, \end{aligned} \tag{13}$$

where we used the conditions of the theorem, the relationships in (9) and (11). Using (8) and (12), for the numerator of (13), it has

$$\begin{aligned} \left\|\sum_{i=2}^n(\lambda_i-1)q(\lambda_i)\phi_i^k\gamma_i\varphi_i\right\|_G &\le \sqrt{\max_{1\le i\le n}w_i}\cdot\left\|\sum_{i=2}^n(\lambda_i-1)q(\lambda_i)\phi_i^k\gamma_i\varphi_i\right\|_2 \\ &\le \sqrt{\max_{1\le i\le n}w_i}\cdot\sum_{i=2}^n|\lambda_i-1|\cdot|\phi_i|^k\cdot|\gamma_i|\cdot|q(\lambda_i)| \\ &\le \sqrt{\max_{1\le i\le n}w_i}\cdot\sum_{i=2}^n\left(\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}\right)^k\cdot|\lambda_i-1|\cdot|\gamma_i|\cdot|q(\lambda_i)|. \end{aligned} \tag{14}$$

For the denominator of (13), it obtains

$$\left\|\sum_{i=1}^{n} q(\lambda_i)\phi_i^k\gamma_i\varphi_i\right\|_G^2 \geq \min_{1\leq i\leq n} w_i \cdot \left\|\sum_{i=1}^{n} q(\lambda_i)\phi_i^k\gamma_i\varphi_i\right\|_2^2$$
$$\geq \min_{1\leq i\leq n} w_i \cdot \sigma_{min}^2(S) \cdot \sum_{i=1}^{n} |\phi_i^k|^2 \cdot |\gamma_i|^2 \cdot |q(\lambda_i)|^2. \quad (15)$$

Combining (14) and (15) into (13), we get

$$\begin{aligned}\|(A-I)u\|_G &\leq \min_{q\in\mathcal{L}_{m-1}} \frac{\sqrt{\max_{1\leq i\leq n} w_i}\cdot\sum_{i=2}^{n}\left(\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}\right)^k\cdot|\lambda_i-1|\cdot|w_i|\cdot|q(\lambda_i)|}{\sqrt{\min_{1\leq i\leq n} w_i\cdot\sigma_{min}^2(S)\cdot\sum_{i=1}^{n}|\phi_i^k|^2\cdot|\gamma_i|^2\cdot|q(\lambda_i)|^2}}\\ &\leq \frac{1}{\sigma_{min}(S)}\cdot\sqrt{\frac{\max_{1\leq i\leq n} w_i}{\min_{1\leq i\leq n} w_i}}\cdot\min_{q\in\mathcal{L}_{m-1}}\frac{\sum_{i=2}^{n}\left(\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}\right)^k\cdot|\lambda_i-1|\cdot|\gamma_i|\cdot|q(\lambda_i)|}{|\gamma_1|\cdot|q(\lambda_1)|}\\ &= \frac{1}{\sigma_{min}(S)}\cdot\sqrt{\frac{\max_{1\leq i\leq n} w_i}{\min_{1\leq i\leq n} w_i}}\cdot\left(\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}\right)^k\cdot\min_{q\in\mathcal{L}_{m-1}}\sum_{i=2}^{n}|\lambda_i-1|\cdot\frac{|\gamma_i|}{|\gamma_1|}\cdot\frac{|q(\lambda_i)|}{|q(\lambda_1)|}.\end{aligned}$$

Let $p(\lambda) = q(\lambda)/q(1)$, where $p(1) = 1$, then we have

$$\|(A-I)u\|_G \leq \left(\frac{\alpha^{m_1}(\alpha-\beta)}{1-\beta}\right)^k \frac{\xi\cdot\zeta}{\sigma_{min}(S)} \min_{p\in\mathcal{L}_{m-1},\,p(\lambda_1)=1}\ \max_{\lambda\in\sigma(A)/\{\lambda_1\}} |p(\lambda)|.$$

Therefore, we complete the proof of Theorem 4. □

Remark 2 Comparing our result in Theorem 4 with the result in Theorem 3 of [26], it is easy to find that the adaptive GArnoldi-MPIO method can increase the convergence speed of the adaptive Power-GArnoldi method by a factor of $(\alpha^{m_1-1})^k \cdot \left(\frac{\alpha-\beta}{1-\beta}\right)^k$ when turning from the MPIO method to the adaptive GArnoldi method. Therefore, from the view of theory, our proposed method will have a faster convergence than the adaptive Power-GArnoldi method.

4 Numerical experiments

In this section, we give some numerical examples to test the effectiveness of the adaptive GArnoldi-MPIO method (denoted as "GA-MPIO"), and compare it with the MPIO method [7], the Power-Arnoldi method (denoted as "PA") [4], the adaptive Power-GArnoldi method (denoted as "PGA") [26] and the Arnoldi-MSPI method (denoted as "AMS") [19] in terms of the iteration counts (IT), the number of matrix-vector products (Mv) and the computing time (CPU) in seconds. All the numerical results are obtained by using MATLAB R2016a on the Windows 10 (64 bit) operating system with 2.40 GHz Intel(R) Core(TM) i7-5500U CPU and RAM 8.00 GB.

In Table 1, we list the characteristics of our test matrices, where n denotes the matrix size, nnz is the number of nonzero elements, and den is the density which is

Table 1 The characteristic of test matrices

Name	n	nnz	den
wb-cs-stanford	9,914	36,854	0.375×10^{-1}
web-Stanford	281,903	2,312,497	0.291×10^{-2}
Stanford-Berkeley	683,446	7,583,376	0.162×10^{-2}

defined by $den = \frac{nnz}{n \times n} \times 100$. All the test matrices are available from https://sparse.tamu.edu/.

For the sake of justification, in all the methods we use the same initial vector $x^{(0)} = e/n$ with $e = [1, 1, \cdots, 1]^{\mathrm{T}}$. Meanwhile, similar to [9, 11, 18, 19], we set the damping factor as $\alpha = 0.99, 0.993, 0.995$ and 0.997, respectively. The 2-norm of residual vector is chosen as our stopping criterion in all experiments, and the prescribed tolerance is set as $tol = 10^{-8}$.

Additionally, in the MPIO, AMS and GA-MPIO methods, we set the default multiple iteration parameter as $m_1 = 3$ and the inner tolerance $\eta = 10^{-2}$, $\beta = 0.5$ because they can yield nearly optimal results for the inner-outer method [6]. However, it is hard to determine the optimal choices of the parameters m and $maxit$, because their optimal choices are different for different damping factors α and different PageRank problems. Hence, based on the discussions and advisable values from the related papers [9, 11, 15, 18, 19, 25, 26], we uniformly set $m = 8$ and $maxit = 8$ in our experiments for a fair comparison. The parameters chosen to flip-flop are set as $\alpha_1 = \alpha - 0.1$ and $\alpha_2 = \alpha - 0.1$ in the AMS and GA-MPIO methods. And the same choice is used for the PA and PGA methods. In the PA and AMS methods, we run the thick restarted Arnoldi procedure two times per cycle with the number of approximate eigenpairs $g = 6$. Similarly, in the PGA and GA-MPIO methods, we run the adaptive GArnoldi procedure two times per cycle. Moreover, in order to describe the efficiency of our proposed method, we define

$$\mathrm{Spt} = \frac{\mathrm{CPU}_{\mathrm{MPIO}} - \mathrm{CPU}_{\mathrm{GA\text{-}MPIO}}}{\mathrm{CPU}_{\mathrm{MPIO}}} \times 100\%.$$

to show the speedup of the GA-MPIO method with respect to the MPIO method.

Numerical results of the five methods for all the test matrices are provided in Tables 2, 3, and 4. From Tables 2, 3, and 4, we can see that

- For all the test matrices, the GA-MPIO method outperforms the MPIO method in terms of the iteration counts, the number of matrix-vector products and the computing time. Especially, the speedup of the GA-MPIO method with respect to the MPIO method is up to 86.35%. Hence, it shows that considering the adaptive GArnoldi method as a preconditioned technique for the MPIO method is meaningful.
- The GA-MPIO method works better than the PA and PGA methods in terms of the iteration counts and the computing time for all the test matrices, even though it needs a little more matrix-vector products in some cases. One possible reason is that the MPIO method needs more matrix-vector products than the power

Table 2 Numerical results of the five methods for the wb-cs-stanford matrix

α	MPIO	PA	PGA	AMS	GA-MPIO
$\alpha = 0.99$					
IT	250	100	71	29	23
Mv	1001	161	151	197	177
CPU (Spt)	0.1571	0.0846	0.0695	0.0747	0.0465 (70.40%)
$\alpha = 0.993$					
IT	357	133	81	37	25
Mv	1429	215	177	239	197
CPU (Spt)	0.1925	0.0978	0.0741	0.0868	0.0486 (74.75%)
$\alpha = 0.995$					
IT	500	140	102	46	34
Mv	2001	222	214	289	249
CPU (Spt)	0.2762	0.1098	0.0763	0.1065	0.0589 (78.67%)
$\alpha = 0.997$					
IT	835	174	118	51	37
Mv	3341	275	262	324	270
CPU (Spt)	0.4461	0.1346	0.0972	0.1167	0.0609 (86.35%)

Table 3 Numerical results of the five methods for the web-Stanford matrix

α	MPIO	PA	PGA	AMS	GA-MPIO
$\alpha = 0.99$					
IT	285	134	105	57	39
Mv	1141	247	281	319	258
CPU (Spt)	11.3068	5.1062	5.3975	4.6193	3.7956 (66.43%)
$\alpha = 0.993$					
IT	408	200	139	69	49
Mv	1633	338	379	387	317
CPU (Spt)	16.5180	6.4072	7.1201	5.6974	4.0768 (75.32%)
$\alpha = 0.995$					
IT	572	239	151	81	56
Mv	2289	432	407	445	364
CPU (Spt)	23.1163	8.4735	7.8941	6.4341	4.6790 (79.76%)
$\alpha = 0.997$					
IT	954	317	174	105	81
Mv	3817	584	462	593	512
CPU (Spt)	38.6842	11.4853	8.8188	8.4995	6.7237 (82.62%)

Table 4 Numerical results of the five methods for the Stanford-Berkeley matrix

α	MPIO	PA	PGA	AMS	GA-MPIO
$\alpha = 0.99$					
IT	304	259	206	74	67
Mv	1217	516	542	412	435
CPU (Spt)	21.8189	21.5090	21.4558	11.6537	10.7943 (50.53%)
$\alpha = 0.993$					
IT	437	352	283	116	85
Mv	1749	702	747	639	549
CPU (Spt)	29.8509	29.8306	29.6589	17.7532	14.1027 (52.76%)
$\alpha = 0.995$					
IT	612	475	355	164	124
Mv	2449	950	963	897	766
CPU (Spt)	40.3098	39.1994	38.2646	26.1607	19.8909 (50.65%)
$\alpha = 0.997$					
IT	1022	729	452	244	193
Mv	4089	1470	1252	1323	1192
CPU (Spt)	65.7275	61.5272	49.0371	36.5741	31.2351 (52.48%)

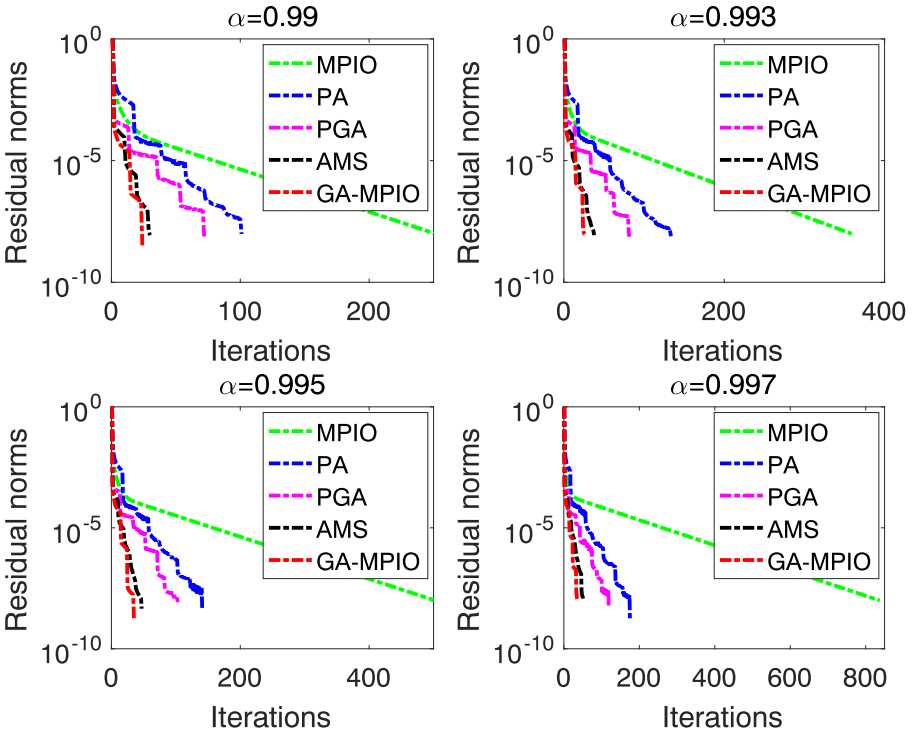

Fig. 1 Convergence behavior of the five methods for the wb-cs-stanford matrix

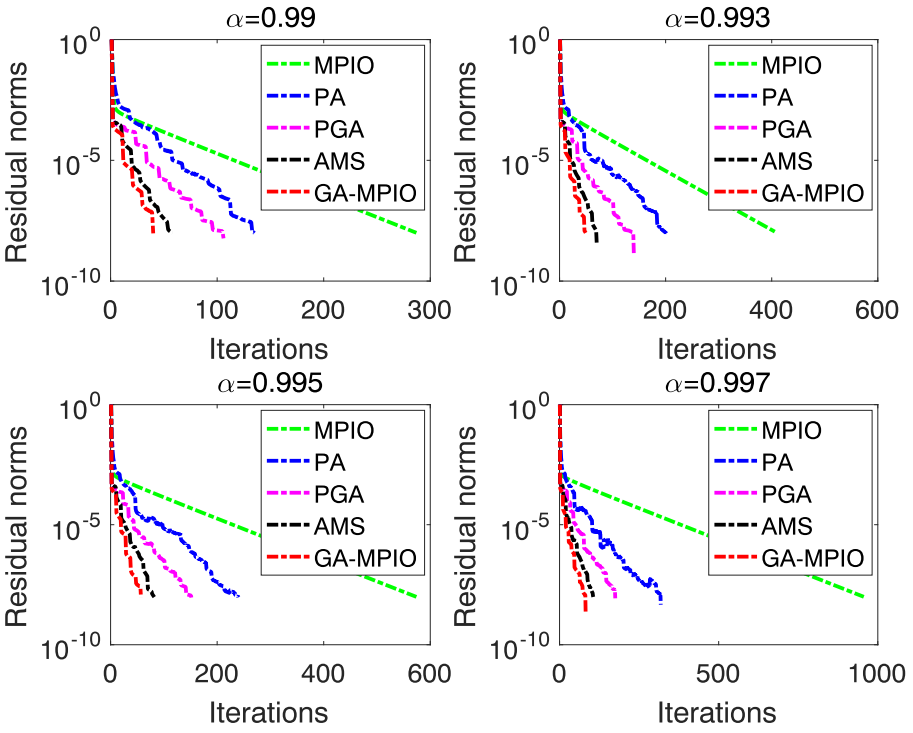

Fig. 2 Convergence behavior of the five methods for the web-Stanford matrix

method in each iteration. Another possible reason is that, in the Arnoldi process, 1 matrix-vector product is accomplished with much more vector-operations than in the power method. As we know, for the power iteration, 1 matrix-vector product is mainly accomplished with 1 vector-scaling operation and 1 vector-addition operation, while in the Arnoldi process, 1 matrix-vector product is accomplished mainly with $\frac{m+1}{2}$ vector inner-product operations, $\frac{m+1}{2} + \frac{m+1}{m}$ vector-scaling operations, $\frac{m+1}{2}$ vector-addition operations and $\frac{m+1}{m}$ norm computations. From the results in Tables 2, 3, and 4, it observes that the GA-MPIO method gets the smallest iteration counts, thus a smallest number of the Arnoldi process is implemented compared with the PA and PGA methods. That is why the GA-MPIO method costs remarkably smaller computing time, although it gets similar number of matrix-vector products with the PA and GPA methods in some cases. For example, when $\alpha = 0.997$, for the PA and PGA methods, we find that their computing time are reduced by 54.75% and 37.34% in Table 1, 41.46% and 23.76% in Table 3, 49.23% and 36.30% in Table 4, respectively. Therefore, these numerical performances suggest that our proposed method has a faster convergence than the PA and PGA methods, which verifies our theoretical analysis in Remark 2.

- The last two columns list the numerical results of the AMS and GA-MPIO methods. As mentioned in Section 1, the main difference between the AMS and

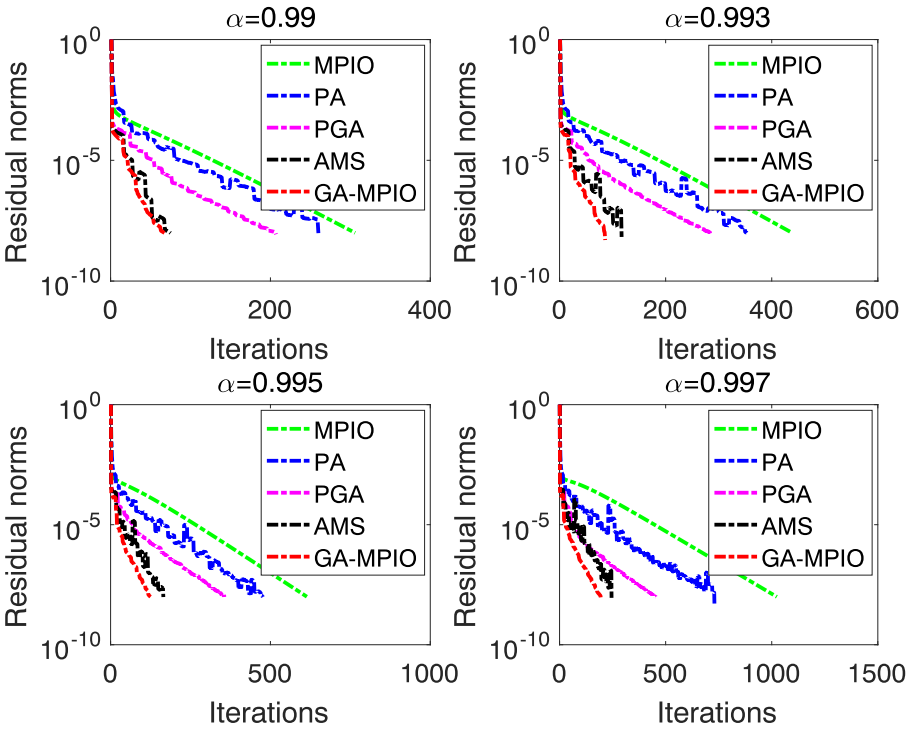

Fig. 3 Convergence behavior of the five methods for the Stanford-Berkeley matrix

GA-MPIO methods lies in that the former used the thick restarted Arnoldi algorithm to preprocess the multi-step splitting iteration, while the latter used the adaptive GArnoldi method. According to the results in Tables 2, 3, and 4, we can see that all the numerical performance of the GA-MPIO method is superior to the AMS method for all the test matrices, except the Stanford-Berkeley matrix with $\alpha = 0.99$ where the number of matrix-vector products of the GA-MPIO method is inferior to the AMS method. The main reason is that the convergence performance of the Arnoldi method is improved by the adaptively accelerating technique. For instance, when $\alpha = 0.997$, the computing time of the AMS method is reduced by 47.81% in Table 2. Hence, we can say that our proposed method outperforms the AMS method when α is high, which also indicates that introducing a weighted inner product into the Arnoldi process for computing PageRank problems is significant.

Figures 1, 2 and 3 plot the convergence behavior of the MPIO method, the PA method, the PGA method, the AMS method and the GA-MPIO method for $\alpha = 0.99, 0.993, 0.995$ and 0.997, respectively. They show that our proposed method converges faster than its counterparts again.

5 Conclusions

In this paper, we present a new method by considering the adaptive GArnoldi method as a preconditioned technique to accelerate the MPIO method. The new method is called as the adaptive GArnoldi-MPIO method, whose construction and convergence analysis can be found in Section 3. Numerical experiments in Section 4 show that our proposed method is efficient and has a faster convergence than its counterparts.

In the future, we would like to discuss the choice of experimental parameters, e.g., how to determine the optimal m and $maxit$ so that the new method can work more efficiently. In addition, the optimal choice of the weighted matrix G is also required to be further analyzed.

Acknowledgements The authors would like to thank the anonymous referees for their valuable comments and suggestions on the original manuscript, which greatly improved the quality of this article.

Funding This research is supported by the National Natural Science Foundation of China (12101433), and the Two-Way Support Programs of Sichuan Agricultural University (1921993077).

Data availability The datasets generated or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Page, L., Brin, S., Motwami, R., Winograd, T.: The PageRank citation ranking: Bringing order to the web, Technical report, Computer Science Department, Stanford University, Stanford CA (1999)
2. Langville, A., Meyer, C.: Deeper inside PageRank. Internet Math. **1**, 335–380 (2005)
3. Jia, Z.X.: Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. Linear Algebra Appl. **259**, 1–23 (1997)
4. Wu, G., Wei, Y.: A Power-Arnoldi algorithm for computing pagerank. Numer. Linear Algebra Appl. **14**, 521–546 (2007)
5. Gu, C.Q., Xie, F., Zhang, K.: A two-step matrix splitting iteration for computing pagerank. J. Comput. Appl. Math. **278**, 19–28 (2015)
6. Gleich, D., Gray, A., Greif, C., Lau, T.: An inner-outer iteration for computing PageRank. SIAM J. Sci. Comput. **32**, 349–371 (2010)
7. Wen, C., Huang, T.Z., Shen, Z.L.: A note on the two-step matrix splitting iteration for computing pagerank. J. Comput. Appl. Math. **315**, 87–97 (2017)
8. Morgan, R., Zeng, M.: A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity. Linear Algebra Appl. **415**, 96–113 (2006)
9. Hu, Q.Y., Wen, C., Huang, T.Z., Shen, Z.L., Gu, X.M.: A variant of the Power-Arnoldi algorithm for computing PageRank. J. Comput. Appl. Math. **381**, 113034 (2021)
10. Tan, X.Y.: A new extrapolation method for pagerank computations. J. Comput. Appl. Math. **313**, 383–392 (2017)
11. Gu, C.Q., Jiang, X.L., Shao, C.C., Chen, Z.B.: A GMRES-power algorithm for computing PageRank problems. J. Comput. Appl. Math. **343**, 113–123 (2018)
12. Saad, Y., Schultz, M.H.: GMRES: A Generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 857–869 (1986)

13. Pu, B.Y., Huang, T.Z., Wen, C.: A preconditioned and extrapolation-accelerated GMRES method for PageRank. Appl. Math. Lett. **37**, 95–100 (2014)
14. Golub, G.H., Greif, C.: An Arnoldi-type algorithm for computing PageRank. BIT. **46**, 759–771 (2006)
15. Wu, G., Wei, Y.: An Arnoldi-Extrapolation algorithm for computing PageRank. J. Comput. Appl. Math. **234**, 3196–3212 (2010)
16. Miao, Q.C., Tan, X.Y.: Accelerating the Arnoldi method via Chebyshev polynomials for computing PageRank. J. Comput. Appl. Math. **377**, 112891 (2020)
17. Gu, C.Q., Wang, L.: On the multi-splitting iteration method for computing pagerank. J. Appl. Math. Comput. **42**, 479–490 (2013)
18. Gu, C.Q., Wang, W.W.: An Arnoldi-Inout algorithm for computing PageRank problems. J. Comput. Appl. Math. **309**, 219–229 (2017)
19. Gu, C.Q., Jiang, X.L., Nie, Y., Chen, Z.B.: A preprocessed multi-step splitting iteration for computing PageRank. Appl. Math. Comput **338**, 87–100 (2018)
20. Tian, Z.L., Liu, Y., Zhang, Y., Liu, Z.Y., Tian, M.Y.: The general inner-outer iteration method based on regular splittings for the PageRank problem. Appl. Math. Comput. **356**, 479–501 (2019)
21. Shen, Z.L., Huang, T.Z., Carpentieri, B., Wen, C., Gu, X.M., Tan, X.Y.: Off-diagonal low-rank preconditioner for difficult PageRank problems. J. Comput. Appl. Math. **346**, 456–470 (2019)
22. Shen, Z.L., Huang, T.Z., Carpentieri, B., Gu, X.M., Wen, C.: An efficient elimination strategy for solving Pagerank problems. Appl. Math. Comput. **298**, 111–122 (2017)
23. Zhang, H.F., Huang, T.Z., Wen, C., Shen, Z.L.: FOM accelerated by an extrapolation method for solving Pagerank problems. J. Comput. Appl. Math. **296**, 397–409 (2016)
24. Gu, X.M., Lei, S.L., Zhang, K., Shen, Z.L., Wen, C., Carpentieri, B.: A Hessenberg-type algorithm for computing PageRank Problems. Numer. Algorithms **89**, 1845–1863 (2021)
25. Yin, J.F., Yin, G.J., Ng, M.: On adaptively accelerated Arnoldi method for computing PageRank. Numer. Linear Algebra Appl. **19**, 73–85 (2012)
26. Wen, C., Hu, Q.Y., Yin, G.J., Gu, X.M., Shen, Z.L.: An adaptive Power-GArnoldi algorithm for computing PageRank. J. Comput. Appl. Math. **386**, 113209 (2021)
27. Haveliwala, T., Kamvar, S.: The second eigenvalue of the google matrix. In: Proceedings of the Twelfth International World Wide Web of Conference (2003)
28. Langville, A., Meyer, C.: Google's PageRank and beyond: The Science of the Search Engine Rankings. Princeton University Press (2006)