Journal of Computational and Applied Mathematics 313 (2017) 383–392

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A new extrapolation method for PageRank computations☆

Xueyuan Tan

Jiangsu Key Laboratory for NSLSCS, School of Mathematical Science, Nanjing Normal University, Nanjing 210046, PR China

ARTICLE INFO

Article history:
Received 5 February 2015
Received in revised form 4 March 2016

Keywords:
PageRank
Power method
Extrapolation method
Arnoldi-type algorithm
Trace

ABSTRACT

The PageRank algorithm is widely considered these years because of its great significance in search engine technology and other scientific domains. Though the power method is the initial measure to settle the PageRank problem, it gives poor convergence when the damping factor is sufficiently close to 1. This difficulty encourages researchers to present improved iterative methods for accelerating PageRank computations. In this paper, a cheap and practical extrapolation approach which is determined by the trace of the Google matrix is proposed, and it is more attractive when combined with the well-known Arnoldi-type algorithm. Convergence analysis of our algorithms is given. Numerical examples illustrate the efficiency of these accelerated schemes.

1. Introduction

PageRank is a system of scores based on the Web link structure to rank the importance of Web pages [1]. PageRank computations have played a vital role in modern Web search ranking systems and have received widespread attention by researchers since it was devised by Google. After years of development, the PageRank algorithm is now frequently applied to any graph or network in any domain [2], such as biology [3–5], neural network [6,7] and social network [8] analysis, bibliometric indexes [9] and so forth.

Computing PageRank is mathematically a process of determining the stationary probability vector called the PageRank vector of the Google matrix, where the Google matrix is a sparse Markov matrix with dimension in excess of some billions. The calculation of the PageRank vector can be viewed as either the solution of an eigenvector problem or an equivalent form, a homogeneous linear system. Due to the Web graph's high dimension and sparse structure, the power method and its variants based on matrix–vector products are preferred for the eigenvector problem. The power method was initially considered for the PageRank computation by Brin and Page et al. [1], since then a number of acceleration techniques have been presented such as adaptive method [10], lumpable aggregation method [11,12], parallel computation [13], quadratic extrapolation method [14], Arnoldi-type algorithm [15], vector extrapolation methods [16,17], inner–outer algorithms [18], and so on. For more details about the PageRank problem, see, e.g., [2,19,20].

Among the aforementioned strategies, Krylov subspace methods have been implemented constantly. In 2006, Golub and Greif proposed an outstanding Arnoldi-type algorithm [15] which is the restarted refined Arnoldi method [21] but entails no Ritz value computations. However, it does not perform efficiently when the Arnoldi step is small and the damping factor is high. Wu and Wei have done some improvements by developing the Power–Arnoldi algorithm [22] and the Arnoldi-Extrapolation algorithm [23], where the former method combines the power method with the thick restarted Arnoldi algorithm [24] and the latter knits an extrapolation method based on Ritz values with the Arnoldi-type algorithm. Motivated

☆ This work is supported by National Natural Science Foundation of China (2010101GZ30005 and 41571381), the Major Project Foundation for the Natural Science of Jiangsu Higher Education Institution (16KJA110001), and Jiangsu Innovation Foundation for Doctor of Science (CX07B-027z).

E-mail address: tanxueyuan@njnu.edu.cn.

http://dx.doi.org/10.1016/j.cam.2016.08.034
0377-0427/

by these works, we give a new extrapolation method to speed up the convergence of power iterations. Our new extrapolation strategy is easy to be executed as it is only determined by the trace of the Google matrix, and is similar to the quadratic extrapolation method in the numerical behavior but superior in the storage saving. Furthermore, by combining this new extrapolation process with the Arnoldi-type algorithm we establish a hybrid method which shows satisfactory performance in the PageRank computation especially when high damping factor is given.

The remainder of this paper is organized as follows. The background of the PageRank problem is introduced in Section 2. The power method with the new extrapolation measure is discussed and analyzed in Section 3. In Section 4 the new extrapolation strategy is combined with the Arnoldi-type algorithm and the convergence theorem is provided. Numerical experiments show the merits of our techniques in Section 5, and conclusions point out the future work in Section 6.

2. Background

The adjacency matrix $P \in \mathbb{R}^{n\times n}$ of the Web graph is described as

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

where $deg(i)$ denotes the outdegree of page i. If page i contains no outlinks, then the ith row of P will be zero and page i is called a dangling node. The existence of dangling nodes is a drawback for computing PageRank. To remedy this simply is to revise the form of P by setting

$$\bar{P} = P + dw^T,$$

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

$$d_i = \begin{cases} 1, & \text{if } deg(i) = 0, \\ 0, & \text{otherwise,} \end{cases}$$

and $0 \le w = (w_i) \in \mathbb{R}^n$ with $\|w\|_1 = 1$. Then $\bar{P}$ is a row-stochastic matrix, i.e., $\bar{P}e = e$ where $e = (1, 1, \ldots, 1)^T$. To guarantee the aperiodicity and irreducibility of $\bar{P}$, a damping factor α and a personalization vector v are introduced to establish the final Google matrix A as

$$\begin{aligned} A &= [\alpha\bar{P} + (1-\alpha)ev^T]^T, \\ &= \alpha\bar{P}^T + (1-\alpha)ve^T, \quad \alpha \in [0, 1), \end{aligned}$$

where v is a probability vector and is usually taken as $v = \frac{1}{n}e$ in the experiments. Therefore A is column-stochastic, irreducible and by the Perron–Frobenius theorem [25], A has a simple and maximum eigenvalue equal to 1, whose corresponding eigenvector is nonnegative and unique. When such eigenvector is normalized, it is then named the PageRank vector which is denoted as u^*. That is to say, the PageRank vector u^* satisfies

$$Au^* = u^*, \quad \|u^*\|_1 = 1.$$

The power method (Algorithm 1) is the traditional choice for PageRank computation. According to Algorithm 1, for a starting vector $u^{(0)}$, the iteration sequence $u^{(l)}$ converges to the unique dominant eigenvector of A as α^l [19]. Thus a small value of α ensures a fast convergence rate and α is originally chosen as 0.85 by Google. However, a higher value of α (close to 1) means the Google matrix A approaches the original Web link graph and gives a truer ranking, but it leads to apparent slow convergence. Hence it is important to design a simple and efficient accelerated method.

Algorithm 1 **(The Power Method)**
1. Choose an initial positive vector $u^{(0)}$ and a prescribed tolerance ε.
2. Set $l = 1$;
Repeat
$u^{(l)} = Au^{(l-1)}$;
$\tau = ||u^{(l)} - u^{(l-1)}||_1$;
$u^{(l)} = u^{(l)}/||u^{(l)}||_1$;
$l = l + 1$;
Until $\tau < \varepsilon$

In [14], Kamvar et al. give the quadratic extrapolation method to accelerate the convergence of the power method by periodically subtracting off estimates of the nonprincipal eigenvectors from the current iterate of the power method. Brezinski et al. generalize the quadratic extrapolation method by establishing its relationship with the Krylov subspace methods [16]. Sidi then applies the minimal polynomial extrapolation (MPE) and the reduced rank extrapolation (RRE) to the PageRank computations, and demonstrates that the quadratic extrapolation method is very closely related to MPE [17].

In the following section, we will depict a new extrapolation technique which depends on the trace of the Google matrix and is efficient in speeding up the convergence of the power method.

3. The power method with extrapolation process based on matrix trace

The Google matrix A will be more closer to the original Web link graph when the damping factor α is sufficiently close to 1, however, the increasing α will dramatically slow down the convergence of the power method. Accordingly, in this section we introduce our new extrapolation strategy based on the trace of A to accelerate the convergence of the power method. Our new algorithm will produce a sequence of subspaces that converges linearly to the invariant eigenspace $span\{u^*\}$, where u^* is the PageRank vector. We will also extend our discussion to show that the convergence rate of the new algorithm depends on a contraction number that is smaller than α^{m-1} (m is a described positive integer), which implies that our algorithm can improve the convergence of the power method.

3.1. Derivation

Since the Google matrix A is a primitive stochastic matrix, all its eigenvalues λ_i ($i = 1, 2, \ldots, n$) must satisfy $1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \geq 0$, which indicates that the characteristic polynomial of A can be written as $(\lambda - 1)Q_A(\lambda)$ where $Q_A(\lambda)$ is a polynomial of degree $n - 1$ without the factor $\lambda - 1$. Thus, by Cayley–Hamilton theorem [26], $(A - I)Q_A(A) = 0$ and $(A - I)Q_A(A)u = 0$ for any nonzero vector u. By considering the properties of minimal polynomial, we may conclude that there exists at least a nonzero vector $u^{(0)}$ such that $Q_A(A)u^{(0)} \neq 0$, which implies that $Q_A(A)u^{(0)}$ can be looked as an eigenvector corresponding to the eigenvalue 1, or in other words, $Q_A(A)u^{(0)}$ gives a representation of its parallel vector, the PageRank vector u^*. However, it is hopelessly complicated and impractical to compute $Q_A(A)u^{(0)}$ when A is a high order matrix. This leads to the motivation of a good approximation to the polynomial $Q_A(\lambda)$.

In general, the hyperlink matrix P of the Web graph may contain a considerable number of dangling nodes. Assume that P has l dangling nodes, then the Google matrix A has at least l identical columns, therefore zero is an eigenvalue of A with the algebraic multiplicity not less than $l - 1$. Furthermore, $Q_A(\lambda)$ has the form $Q_A(\lambda) = \lambda^{n-1} - (\mu - 1)\lambda^{n-2} + \alpha_3\lambda^{n-3} + \alpha_4\lambda^{n-4} + \cdots + \alpha_p\lambda^{n-p}$ ($p \leq n - l + 1$) where μ is the trace of A and α_i ($i = 3, 4, \ldots, p$) are scalars with $\alpha_p \neq 0$. As a result, the larger the l, the fewer the nonzero items $Q_A(\lambda)$ will have. Consequently, it is natural to use $\lambda^{n-1} - (\mu - 1)\lambda^{n-2}$ as a rational approximation to $Q_A(\lambda)$ when copious dangling nodes exist.[1] Then, we can select $[A^{n-1} - (\mu - 1)A^{n-2}]u^{(0)} \neq 0$ as an approximation for the PageRank vector u^*. Further, we observe that

$$\begin{aligned}
[A^{n-1} - (\mu - 1)A^{n-2}]u^{(0)} &= A^{n-2}[A - (\mu - 1)I]u^{(0)} \\
&= A^{n-m-1}[A - (\mu - 1)I]A^{m-1}u^{(0)} \\
&= A^{n-m-1}[A - (\mu - 1)I]u^{(m-1)} \\
&= A^{n-m-1}[u^{(m)} - (\mu - 1)u^{(m-1)}],
\end{aligned}$$

where $m < n - 1$ is a positive integer and $u^{(i)}$ ($i = m - 1, m$) is the ith iteration of the power method (Algorithm 1). Such observation inspires us to establish an extrapolation scheme based upon $u^{(m-1)}$, $u^{(m)}$ and μ to accelerate the power iteration. That is, for an initial vector $u^{(0)}$, we can execute the power iteration for m steps obtaining $u^{(m-1)}$ and $u^{(m)}$, then approximate the PageRank vector u^* by the linear combination of $u^{(m-1)}$ and $u^{(m)}$ which is given as $u^{(0)}_{new} = u^{(m)} - (\mu - 1)u^{(m-1)}$, where $u^{(0)}_{new}$ is viewed as the new initial vector to restart the power iteration. In a word, we can apply this new extrapolation process in the cycling mode to speed up the convergence of the power method.

Next we derive the formula of μ which stands for the trace of the Google matrix A. Remember that $A = \alpha\bar{P}^T + (1-\alpha)ve^T$, and by adding up all the main diagonal elements of A, we have $\mu = 1 + \alpha(\frac{l}{n} - 1)$. Moreover, it is obvious that $\mu < 1$ since $l \leq n$ and $0 \leq \alpha < 1$.

We now give the power method with the new extrapolation procedure based on μ as follows.

Algorithm 2 **(The Power Method with the Extrapolation Process Based on Trace)**

1. Choose an initial positive normalized vector $u^{(0)}$, a positive integer m and a prescribed tolerance ε.
2. Compute the number of dangling nodes l and set $\mu = 1 + \alpha(\dfrac{l}{n} - 1)$.
3. Run the power iteration m steps to obtain $u^{(m-1)}$ and $u^{(m)}$:

{ for $i = 1$ to m
$u^{(i)} = Au^{(i-1)}$;
$\tau = ||u^{(i)} - u^{(i-1)}||_1$;
$u^{(i)} = u^{(i)}/||u^{(i)}||_1$;
if $\tau <= \varepsilon$
$u^* = u^{(i)}$ and break;
else $i = i + 1$;
end for }

[1] In particular, if A is a 2×2 matrix, then its characteristic polynomial is $\lambda^2 - \mu\lambda + (\mu - 1)$ and the approximation $[A - (\mu - 1)I]u^{(0)}$ (for any vector $u^{(0)} \in R^2$ such that $[A - (\mu - 1)I]u^{(0)} \neq 0$) is an accurate eigenvector corresponding to eigenvalue 1.

Table 1
Comparison of matrix–vector multiplications and storage requirements.

	Power method	Quadratic extrapolation	Algorithm 2
Matrix–vector	1	3	1
Storage	2	4	2

4. Set $u^{(0)} = u^{(m)} - (\mu - 1)u^{(m-1)}$;
$u^{(0)} = u^{(0)}/||u^{(0)}||_1$;
$\tau = ||u^{(0)} - u^{(m)}||_1$;
if $\tau > \varepsilon$
go to step 3;
else
$u^* = u^{(0)}$ and break.

In Table 1, we compare Algorithm 2 with the power method and the quadratic extrapolation method [14] in terms of the storage requirements and computational cost. The advantage of Algorithm 2 is that it has approximately the same computational complexity as the power method and is more economical than the quadratic extrapolation method both in terms of storage and computational effort. As a result, Algorithm 2 is an effective technique for speeding up the convergence of the power method, especially in the high damping factor case (see examples in Section 5).

3.2. Convergence analysis

In this subsection, we pay attention to the convergence of Algorithm 2. Employing the view of subspace iteration, we can consider Algorithm 2[2] as the stationary subspace iteration

$$S^{(i)} = p(A)S^{(i-1)}, \qquad S^{(0)} = span\{u^{(0)}\}, \quad i = 1, 2, 3, \ldots,$$

where $p(A) = A^{m-1}[A - (\mu - 1)I]$ is a matrix polynomial of degree m. In order to reveal the convergence rate of Algorithm 2, we begin with the following theorems.

Theorem 1 ([27]). *If the row-stochastic matrix $\bar{P} \in \mathbb{R}^{n\times n}$ has at least two irreducible closed subsets (which is the case for the Web hyperlink matrix), then the second eigenvalue of the Google matrix A is given by $\lambda_2 = \alpha$ where α is the damping factor.*

Theorem 2 ([28]). *Let $B \in \mathbb{C}^{n\times n}$ (need not to be diagonalizable), and let p be a polynomial of degree m that is smaller than n. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ denote the eigenvalues of B, ordered so that $|p(\lambda_1)| \geq |p(\lambda_2)| \geq \cdots \geq |p(\lambda_n)|$. Suppose that k is an integer satisfying $1 \leq k < n$ for which $|p(\lambda_k)| > |p(\lambda_{k+1})|$, and let $\rho = |p(\lambda_{k+1})|/|p(\lambda_k)| < 1$. Let U and V be the invariant subspaces of B associated with $\lambda_1, \lambda_2, \ldots, \lambda_k$ and $\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_n$, respectively. Consider the stationary subspace iteration*

$$S^{(i)} = p(B)S^{(i-1)}, \qquad S^{(0)} = S, \quad i = 1, 2, 3, \ldots,$$

where S is a k-dimensional subspace satisfying $S \cap V = \{0\}$. Then for every $\hat{\rho}$ satisfying $\rho < \hat{\rho} < 1$ there is a constant C such that

$$d(S^{(i)}, U) \leq C\hat{\rho}^i, \quad i = 1, 2, 3, \ldots,$$

where $d(X, Y)$ denotes the distance between X and Y as

$$d(X, Y) = \max_{\substack{x\in X \\ \|x\|=1}} d(x, Y) = \max_{\substack{x\in X \\ \|x\|=1}} \min_{y\in Y} \|x - y\|.$$

Using Theorem 2, we can prove the following convergence theorem.

Theorem 3. *Let $A \in \mathbb{C}^{n\times n}$ be the Google matrix, let λ_i $(i = 1, 2, \ldots, n)$ be the eigenvalues of A ordered as $1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \geq 0$, and let U and V be the invariant subspaces of A associated with λ_1 and $\lambda_2, \lambda_3, \ldots, \lambda_n$, respectively. Define $p(\lambda) = \lambda^m - (\mu - 1)\lambda^{m-1}$ and $\rho = |p(\alpha)|/|p(1)|$, where μ is the trace of A, α is the damping factor and m is the positive integer given in Algorithm 2. Suppose that $S^{(0)} = span\{u^{(0)}\}$ where $u^{(0)} \in \mathbb{C}^n$ is an initial vector satisfying $S^{(0)} \cap V = \{0\}$. Consider Algorithm 2 as the stationary subspace iteration*

$$S^{(i)} = p(A)S^{(i-1)}, \qquad S^{(0)} = span\{u^{(0)}\}, \quad i = 1, 2, 3, \ldots.$$

[2] Here the last loop of Algorithm 2 is omitted for the sake of clarity. Generally, the last loop of Algorithm 2 will achieve the precision ε during the power iteration of step 3, and the subspace iteration of this loop is $S^{(i)} = A^r S^{(i-1)}$, $r < m$.

Then $\rho < \alpha^{m-1} < 1$, *and for every* $\hat{\rho}$ *satisfying* $\rho < \hat{\rho} < \alpha^{m-1} < 1$ *there exists a constant* C *such that*

$$d(S^{(i)}, U) \leq C\hat{\rho}^i, \quad i = 1, 2, 3, \ldots,$$

where $d(X, Y)$ *denotes the distance between* X *and* Y *as*

$$d(X, Y) = \max_{\substack{x \in X \\ \|x\|=1}} d(x, Y) = \max_{\substack{x \in X \\ \|x\|=1}} \min_{y \in Y} \|x - y\|.$$

Proof. By the conclusion of Theorem 1, we know that λ_2 equals to α. Thus,

$$p(\lambda_1) = p(1) = 2 - \mu,$$
$$p(\lambda_2) = p(\alpha) = \alpha^{m-1}[\alpha - (\mu - 1)].$$

It follows that $\rho = \frac{|p(\alpha)|}{|p(1)|} = |\frac{\alpha^{m-1}[\alpha-(\mu-1)]}{2-\mu}| < \frac{\alpha^{m-1}[1-(\mu-1)]}{2-\mu} = \alpha^{m-1} < 1$, which deduces that $|p(\lambda_1)| = |p(1)| > |p(\alpha)| = |p(\lambda_2)|$. Note that $S^{(0)} \cap V = \{0\}$ holds where V represents the invariant subspaces of A associated with $\lambda_2, \lambda_3, \ldots, \lambda_n$. Therefore, as an immediate consequence of Theorem 2, we can gain Theorem 3. □

Remark 1. Since $d(S^{(i)}, U)$ converges to zero, the subspace iteration related to Algorithm 2 generates a sequence of subspaces $\{S^{(i)}\}$ that converges to the dominant 1-dimensional invariant subspace U, where U is the invariant subspaces of A associated with λ_1. That is to say, Algorithm 2 generates a sequence of vectors which is convergent to the PageRank vector u^*.

Remark 2. Theorem 3 indicates that the convergence rate of Algorithm 2 depends on $\hat{\rho}$ where $\hat{\rho} \in (\rho, \alpha^{m-1})$, which means that Algorithm 2 can accelerate the convergence rate of the power method. In addition, the initial guess $u^{(0)}$ can be chosen at random provided that $span\{u^{(0)}\} \cap V = \{0\}$.

4. The Arnoldi-type method combined with Algorithm 2

In this section, we first present a hybrid approach which combines Algorithm 2 with the Arnoldi-type algorithm [15] in order to speed up the convergence performance for PageRank computing with a moderately small overhead, then we propose the convergence analysis of the approximate eigenvector obtained by this hybrid method to the exact one.

4.1. Blending Algorithm 2 into the Arnoldi-type method

The Arnoldi procedure (Algorithm 3) [29] is a famous orthogonal projection process for reducing a general matrix A into the upper Hessenberg form. In short, given a matrix A, an initial vector v and a number of steps k, we will have $AV_k = V_{k+1}\tilde{H}_k$ where the columns of V_k form an orthonormal basis of the Krylov subspace $\mathcal{K}_k(A, v) = span\{v, Av, A^2v, \ldots, A^{k-1}v\}$ and $\tilde{H}_k$ is a $(k + 1) \times k$ upper Hessenberg matrix.

Algorithm 3 **(Arnoldi Process)**
1. Given the initial vector v, Arnoldi steps number k.
2. $v_1 = v/||v||_2$;
for $j = 1$ to k
$z = Av_j$;
for $i = 1$ to j
$h_{i,j} = v_j^T z$;
$z = z - h_{i,j}z_i$;
end for
$h_{j+1,j} = ||z||_2$;
if $h_{j+1,j} = 0$, quit
$v_{j+1} = z/h_{j+1,j}$;
end for

By excluding the last row of $\tilde{H}_k$, we obtain the matrix H_k whose eigenvalues are known as the Ritz values. Ritz values and its corresponding eigenvectors are often utilized for approximating the eigenpairs of A. To find the dominant eigenpair of A, the restarted Arnoldi method can be applied which uses the Ritz vector computed from the previous k-step Arnoldi process as an initial vector for the next Arnoldi procedure. But in theory Ritz vectors may not converge to the true eigenvectors granted that Ritz values do. Thus the refined Arnoldi algorithm [21] is designed in which the refined Ritz vectors are calculated by the singular value decompositions.

In 2006, Golub and Greif proposed the remarkable Arnoldi-type algorithm (Algorithm 4) [15] which is a variant of the restarted refined Arnoldi algorithm by forcing the largest Ritz value to be 1. The algorithm tries to seek the vector $\tilde{u}$ satisfying

$$\|A\tilde{u} - \tilde{u}\|_2 = \min_{\substack{u \in \mathcal{K}_k(A,v) \\ \|u\|_2=1}} \|Au - u\|_2,$$

and

$$\|A\tilde{u} - \tilde{u}\|_2 = \sigma_{\min}(\tilde{H}_k - \tilde{I})$$

holds where $\sigma(\cdot)$ denotes the non-zero singular value of a matrix and $\tilde{I}$ stands for a $k \times k$ identity matrix augmented by a row of zeros.

Algorithm 4 **(The Arnoldi-type Method for Computing PageRank)**
1. Given an initial vector v, an Arnoldi steps number k, and a prescribed tolerance ε.
2. Repeat
Run the Arnoldi process for computing V_k and $\tilde{H}_k$;
Compute SVD: $\tilde{H}_k - \tilde{I} = U\Sigma S^T$;
Set $v = V_k S(:, m)$;
Until $\sigma_{min}(\tilde{H}_k - \tilde{I}) < \varepsilon$

Nevertheless, the Arnoldi-type method may not behave efficiently when a high damping factor α and a small number of Arnoldi steps k are picked [15,22]. Motivated by these tough problems, we address a hybrid method (Algorithm 5) that combines Algorithm 2 with the Arnoldi-type method.

Algorithm 5 **(The Arnoldi-type Method combined with Algorithm** 2 **)**
1. Given an initial positive vector $u^{(0)}$, a positive integer m, an Arnoldi steps number k, and prescribed tolerance values ε_1, ε ($\varepsilon_1 > \varepsilon$).
2. Run Algorithm 2 until $\tau < \varepsilon_1$;
3. Use the vector u resulted from step 2 as the initial vector for step 4;
4. Run Algorithm 4 until $\sigma_{min}(\tilde{H}_k - \tilde{I}) < \varepsilon$

Remark 3. Through step 2 and 3, we get the vector u fulfilling a sketchy convergence tolerance ε_1, and then use u as an initial vector of the Arnoldi-type method (step 4). Since u is a vector relatively approximating to the PageRank vector and is taken as the initial guess of step 4, the Arnoldi-type method will be accelerated in reaching the convergence precision ε. In the later section (Section 5), ε will be chosen to be 10^{-8} and ε_1 will be selected such that $\frac{\varepsilon_1}{\varepsilon} \in \{10, 10^2, 10^3, 10^4\}$, e.g., $\varepsilon_1 = 10^{-6}$ and $\varepsilon = 10^{-8}$, or $\varepsilon_1 = 10^{-5}$ and $\varepsilon = 10^{-8}$.

4.2. Convergence analysis

We begin with some definitions. Let $\mathcal{P}_k$ be the orthogonal projector onto the Krylov subspace $\mathcal{K}_k(A, v_1)$, and define

$$\epsilon_k = \min_{\substack{p \in P^*_{k-1} \\ p(\lambda_1)=1}} \max_{\lambda \in \wedge(A) - \lambda_1} |p(\lambda)|,$$

where P^*_{k-1} stands for the set of all polynomials of degree not exceeding $k-1$ and $\wedge(A)$ denotes the spectrum of A. To discuss the convergence behavior of Algorithm 5, we first introduce a fundamental theorem which studies the distance between the approximate eigenvector obtained from the Arnoldi's method and the exact one.

Theorem 4 (*[30]*)**.** *Assume that A is diagonalizable and that the initial vector v_1 in Arnoldi's method has the expansion $v_1 = \sum_{i=1}^n \alpha_i u_i$ with respect to A's eigenbasis $\{u_i\}_{i=1,2,\ldots,n}$ in which $\|u_i\|_2 = 1$, $i = 1, 2, \ldots, n$ and $\alpha_1 \neq 0$. Then the following inequality holds*

$$\|(I - \mathcal{P}_k)u_1\|_2 \leq \xi_1 \epsilon_k$$

where

$$\xi_1 = \sum_{i=2}^{n} \frac{|\alpha_i|}{|\alpha_1|}.$$

For the sake of ease of discussion, we assume that the Google matrix A is diagonalizable, then we state the convergence analysis of Algorithm 5 in the following theorem.

Theorem 5. *Assume that A is diagonalizable and the initial vector $u^{(0)}$ of Algorithm 5 can be written as $u^{(0)} = \sum_{i=1}^n \alpha_i u_i$ ($\alpha_1 \neq 0$) where u_i ($i = 1, 2, \ldots, n$) are A's normalized eigenvectors associated with eigenvalues λ_i ($i = 1, 2, \ldots, n$) and $1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \geq 0$, let $\xi_1 = \sum_{i=2}^n \frac{|\alpha_i|}{|\alpha_1|}$, then:*

(1) *The initial vector of step 4 in Algorithm 5 has the form*

$$u = \frac{1}{\omega}\hat{u} = \frac{1}{\omega}A^{s(m-1)+r}[A - (\mu - 1)I]^s \sum_{i=1}^{n} \alpha_i u_i, \quad \omega = \|\hat{u}\|_2,$$

where r is the number of power iterations in the last cycle in step 3 in Algorithm 2, and s is the total restarting number in step 4 in Algorithm 2.

(2) *The convergence about Algorithm* 5 *satisfies the following inequality:*

$$\|(I - \mathcal{Q}_k)u_1\|_2 \leq \alpha^{s(m-1)+r}\beta^s\xi_1\epsilon_k$$

where $\mathcal{Q}_k$ *represents the orthogonal projector onto the Krylov subspace* $\mathcal{K}_k(A, u)$, $\beta = \frac{\alpha+1-\mu}{2-\mu} < 1$ *and* α *is the damping factor.*

Proof. Since the initial vector of Algorithm 5 is supposed to be written as $u^{(0)} = \sum_{i=1}^{n}\alpha_i u_i$, it is easy to verify (1) from the construction of Algorithm 2.

Now, we turn to the verification of (2). Observe that for any $v \in \mathcal{K}_k(A, u)$ there exists a polynomial $q(x) \in P_{k-1}^*$ such that

$$\begin{aligned} v &= \frac{1}{\omega}A^{s(m-1)+r}[A - (\mu-1)I]^s q(A)\sum_{i=1}^{n}\alpha_i u_i \\ &= \frac{1}{\omega}A^{s(m-1)+r}[A - (\mu-1)I]^s\left[\alpha_1 q(1)u_1 + \sum_{i=2}^{n}\alpha_i q(\lambda_i)u_i\right] \\ &= \frac{1}{\omega}(2-\mu)^s\alpha_1 q(1)u_1 + \frac{1}{\omega}\sum_{i=2}^{n}\alpha_i q(\lambda_i)\lambda_i^{s(m-1)+r}[\lambda_i - (\mu-1)]^s u_i, \end{aligned}$$

where the last equality is the immediate consequence of $Au_1 = u_1$ and $Au_i = \lambda_i u_i$ ($i = 2, 3, \ldots, n$). Hence, we have

$$\begin{aligned} \left\|\frac{\omega v}{(2-\mu)^s\alpha_1 q(1)} - u_1\right\|_2 &= \left\|\sum_{i=2}^{n}\frac{\alpha_i q(\lambda_i)}{(2-\mu)^s\alpha_1 q(1)}\lambda_i^{s(m-1)+r}[\lambda_i - (\mu-1)]^s u_i\right\|_2 \\ &\leq \sum_{i=2}^{n}\frac{|\alpha_i|}{|\alpha_1|}\frac{|q(\lambda_i)|}{|q(1)|}\frac{1}{(2-\mu)^s}|\lambda_i|^{s(m-1)+r}(|\lambda_i| + |\mu-1|)^s \\ &\leq \frac{\alpha^{s(m-1)+r}(\alpha+1-\mu)^s}{(2-\mu)^s}\cdot\sum_{i=2}^{n}\frac{|\alpha_i|}{|\alpha_1|}\cdot\max_{i\neq 1}\frac{|q(\lambda_i)|}{|q(1)|}. \end{aligned}$$

Let $p(\lambda_i) = \frac{q(\lambda_i)}{q(1)}$ ($i = 2, 3, \ldots, n$), the above inequality then becomes

$$\begin{aligned} \left\|\frac{\omega v}{(2-\mu)^s\alpha_1 q(1)} - u_1\right\|_2 &\leq \frac{\alpha^{s(m-1)+r}(\alpha+1-\mu)^s}{(2-\mu)^s}\cdot\xi_1\cdot\max_{i\neq 1}|p(\lambda_i)| \\ &= \alpha^{s(m-1)+r}\beta^s\xi_1\cdot\max_{i\neq 1}|p(\lambda_i)|, \end{aligned}$$

where $\beta = \frac{\alpha+1-\mu}{2-\mu} < 1$ due to $\alpha < 1$. Since

$$\|(I - \mathcal{Q}_k)u_1\|_2 = \min_{v\in\mathcal{K}_k(A,u)}\|v - u_1\|_2,$$

then the result in (2) can be obtained directly. □

Remark 4. Theorem 5 proclaims that when the power method with the extrapolation process (Algorithm 2) is combined with the Arnoldi-type method (Algorithm 4), the convergence speed can be successfully accelerated by the factor of at least $\alpha^{s(m-1)+r}\beta^s$, where both α and β are constants smaller than 1.

5. Numerical results

In this section we provide numerical experiments to illustrate the speedup achievements of Algorithm 2 and Algorithm 5 by MATLAB 2014b programming package on 2.6 GHz CPU with 8 GB RAM. The test matrices are downloaded from http://www.cise.ufl.edu/research/sparse/matrices. As a rule, $\frac{1}{n}e$ is selected both as the starting vector $u^{(0)}$ and personalization vector v for all the numerical examples. α is chosen in the interval [0.85, 0.999]. $\|Au^{(k)} - u^{(k)}\|_1 \leq 10^{-8}$ is employed as the stopping criterion where $u^{(k)}$ is the approximate eigenvector computed by the current iteration of the involved method. Power, QuaExt, Arnoldi-type are respectively the brief denotations of the power method, the quadratic extrapolation method, and the Arnoldi-type method. To exhibit the comparisons between the previous three methods and our algorithms (Algorithm 2 and Algorithm 5), we enumerate total number of matrix–vector products and CPU time (in seconds) required for convergence in tables.

Example 1. The test matrices we use here are 281 903 × 281 903 *web-Stanford* and 916 428 × 916 428 *web-Google*, containing respectively 172 and 176 974 dangling nodes. We compare the numerical behaviors of Power, QuaExt and Algorithm 2. We

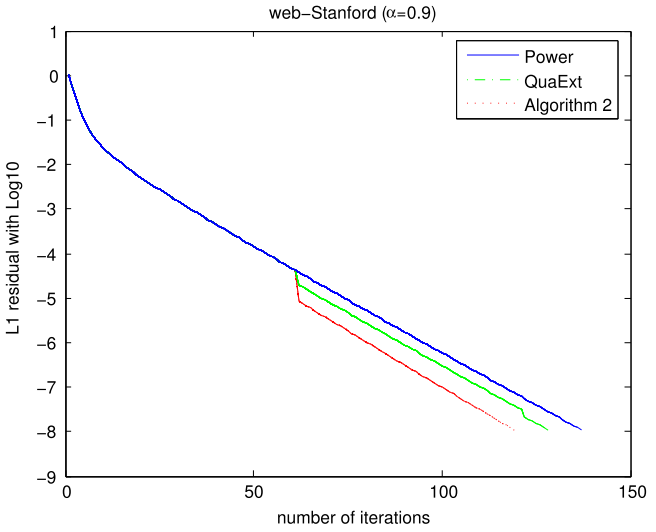

Fig. 1. $\alpha = 0.9$.

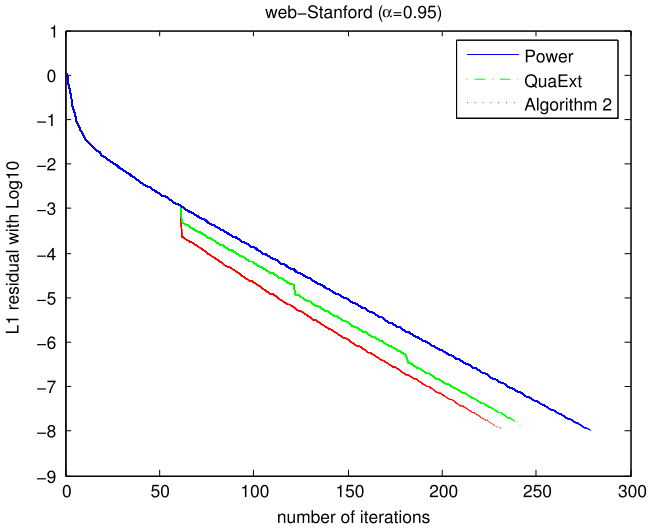

Fig. 2. $\alpha = 0.95$.

set $\alpha = 0.9, 0.95$ for *web-Stanford* and $\alpha = 0.9, 0.99$ for *web-Google*. The quadratic extrapolation is applied every 60 power iterations, and to be fair, the extrapolation strategy based on trace is also performed at every 60th power iteration (i.e., $m = 60$). In the following figures, the x-axis exhibits the iteration numbers (the times that multiplication Au occurs) and y-axis shows the 1-norms of the residual vectors with *Log* function.

From Figs. 1 to 4, we notice that when the extrapolation strategy based on trace is applied periodically, Algorithm 2 can apparently accelerate the convergence of Power and lead to rapid decrease in the residual norms even α is close to 1. The numerical performance of Algorithm 2 is slightly better than QuaExt in the *web-Stanford* case, but somewhat similar to QuaExt in the *web-Google* case. Nevertheless, Algorithm 2 is easier to be used in practice and more economical both in terms of storage and computational effort compared to QuaExt.

Example 2. The test matrix *wb-cs-stanford* is a small one which contains 9914 nodes and 2861 dangling nodes. In Table 2, we display experimental comparisons of Power, QuaExt, Arnoldi-type, Algorithm 2 and Algorithm 5 in terms of the total number of matrix–vector multiplications and CPU time. For every 40 power iterations, we perform the new extrapolation strategy based on trace to Algorithm 2 and Algorithm 5, and run the quadratic extrapolation procedure to QuaExt. Set the number of the Arnoldi steps $k = 4$ or $k = 6$ for the Arnoldi process, and the tolerances $\varepsilon_1 = 10^{-4}$ or $\varepsilon_1 = 10^{-5}$ for Algorithm 5.

Observe that Algorithm 2 behaves similar to QuaExt when $\alpha = 0.85$ or $\alpha = 0.9$, but a little better than QuaExt both in the number of matrix–vector products and CPU time when α is sufficiently near 1. Moreover, Algorithm 5 records superior

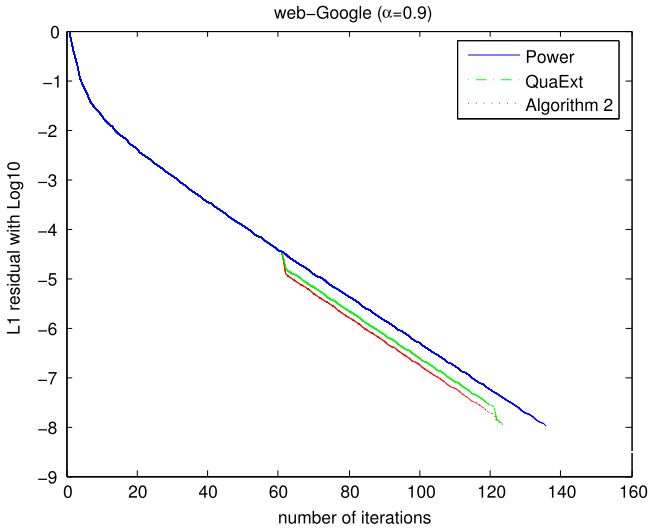

Fig. 3. $\alpha = 0.9$.

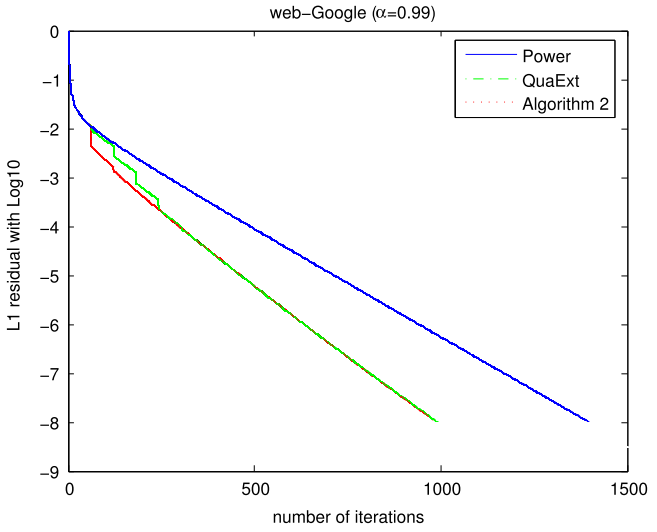

Fig. 4. $\alpha = 0.99$.

Table 2
Total number of matrix–vector products and CPU time(in brackets), *wb-cs-stanford*.

α	Power	QuaExt	Arnoldi-type		Algorithm 2	Algorithm 5			
			$k = 4$	$k = 6$		$\varepsilon_1 = 10^{-4}$		$\varepsilon_1 = 10^{-5}$	
						$k = 4$	$k = 6$	$k = 4$	$k = 6$
0.85	80(0.016)	77(0.013)	76(0.081)	60(0.065)	78(0.012)	56(0.029)	50(0.027)	63(0.031)	61(0.029)
0.9	118(0.027)	107(0.018)	92(0.11)	78(0.089)	113(0.016)	75(0.034)	67(0.026)	83(0.037)	77(0.031)
0.99	1143(0.23)	934(0.14)	596(0.61)	390(0.41)	930(0.12)	435(0.28)	305(0.11)	395(0.08)	431(0.06)
0.999	11 396(2.31)	4230(0.63)	3320(3.39)	798(0.85)	4185(0.56)	1483(0.92)	965(0.42)	1615(0.34)	1477(0.21)

results when α is of high value. For instance, $\alpha = 0.999$, $k = 4$, and $\varepsilon_1 = 10^{-5}$, Algorithm 5 reduces the CPU time needed to reach a residual of 10^{-8} by about 85.3%, 46%, 90% relative to Power, QuaExt and Arnoldi-type, respectively.

Example 3. *Stanford-Berkeley* is a 683 446 × 683 446 Web matrix with 68 062 dangling nodes. We run the corresponding extrapolation process on QuaExt, Algorithm 2 and Algorithm 5 every 50 power iterations. Let the Arnoldi steps number $k = 6$ or 8, select $\varepsilon_1 = 10^{-6}$ or $\varepsilon_1 = 10^{-7}$ in Algorithm 5. Then Table 3 provides the related numerical results.

It is stated that when α is approaching 1, Algorithm 5 performs perfectly with a smaller number of matrix–vector multiplications and less CPU time compared with other methods. In the case where $\alpha = 0.999$ and $k = 8$, Algorithm 5

Table 3
Total number of matrix–vector products and CPU time(in brackets), *Stanford-Berkeley*.

α	Power	QuaExt	Arnoldi-type		Algorithm 2	Algorithm 5			
			$k=6$	$k=8$		$\varepsilon_1=10^{-6}$		$\varepsilon_1=10^{-7}$	
						$k=6$	$k=8$	$k=6$	$k=8$
0.9	138(2.19)	123(1.82)	132(20.54)	120(18.10)	120(1.68)	93(3.38)	97(4.18)	107(2.56)	109(2.96)
0.99	1341(21.27)	1031(14.64)	936(163.74)	888(172.21)	1041(14.77)	639(11.21)	635(10.51)	837(12.37)	839(13.02)
0.999	12 569(201.52)	8144(117.62)	8382(1497.44)	7080(1429.68)	8089(114.92)	4834(154.53)	4532(109.37)	6116(85.06)	6118(85.66)

with $\varepsilon_1 = 10^{-7}$ speeds up the CPU time required for computing the PageRank vector by approximately 57.5%, 27.2%, 94% relative to Power, QuaExt and Arnoldi-type, respectively. We also observe that both Algorithm 2 and QuaExt behave well on the CPU time due to their cheap memory requirements and ease of implementation. In addition, when $\alpha = 0.999$ and $k = 6$, Algorithm 5 with $\varepsilon_1 = 10^{-6}$ takes a little more time than QuaExt since its Arnoldi process is time-consuming for lower k and higher α. To resolve this, we set $\varepsilon_1 = 10^{-7}$ or give higher value of k at the price of more matrix–vector products or higher storage requirements.

6. Conclusions

We have presented a new extrapolation strategy for accelerating PageRank computation by taking advantage of the trace of the Google matrix. We also combine the extrapolation scheme with the Arnoldi-type algorithm and demonstrate by theorems that these approaches can accelerate the convergence of the iterative PageRank computation. Experimental tests show that our methods are more attractive and efficient than some existed methods when the damping factor is close to 1. To further investigations, we would like to discuss the accelerated technique on the dynamic networks and explore the large scale applications based on random walks such as social networks, neural networks etc.

Acknowledgments

The author wishes to express thanks to Professor Li Wang for her helpful suggestions on a first draft of this paper, and is deeply indebted to the editor and referees for their valuable comments.

References

[1] L. Page, S. Brin, R. Motwami, T. Winograd, The PageRank citation ranking: Bringing order to the web, Technical report, Computer Science Department, Stanford University, Stanford, CA, 1999.
[2] D. Gleich, PageRank beyond the web, SIAM Rev. 57 (2015) 321–363.
[3] L. Morrison, R. Breitling, D. Higham, D. Gilbert, GeneRank: Using search engine technology for the analysis of microarray experiments, BMC Bioinform. 6 (2005) 233–246.
[4] V. Freschi, Protein function prediction from interaction networks using a random walk ranking algorithm, in: Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, IEEE, 2007, pp. 42–48.
[5] R. Singh, J. Xu, B. Berger, Pairwise global alignment of protein interaction networks by matching neighborhood topology, in: Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology, Oakland, CA, in: Lecture notes in Comput. Sci., vol. 4453, Springer, Berlin, Heidelberg, 2007, pp. 16–31.
[6] D. Shepelyansky, D. Zhirov, Towards Google matrix of brain, Phys. Lett. A 374 (2010) 3206–3209.
[7] X. Zuo, R. Ehmke, M. Mennes, D. Imperati, F. Castellanos, O. Sporns, M. Milham, Network centrality in the human functional connectome, Cereb. Cortex 22 (2012) 1862–1875.
[8] F. Pedroche, Competitivity groups on social network sites, Math. Comput. Model. 52 (2010) 1052–1057.
[9] P. Amodio, L. Brugnano, Recent advances in bibliometirc indexes and the PageRank problem, J. Comput. Appl. Math. 267 (2014) 182–194.
[10] S. Kamvar, T. Haveliwala, G. Golub, Adaptive methods for the computation of PageRank, Linear Algebra Appl. 386 (2004) 51–65.
[11] A. Langville, C. Meyer, Updating PageRank with iterative aggregation, in: Proceedings of the Thirteenth World Wide Web Conference, ACM Press, New York, 2004, pp. 392–393.
[12] Y. Lin, X. Shi, Y. Wei, On computing PageRank via lumping the Google matrix, J. Comput. Appl. Math. 224 (2009) 702–708.
[13] D. Gleich, L. Zhukov, P. Berkhin, Fast parallel PageRank: a linear system approach, Technical report, Yahoo! Inc., 2004.
[14] 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, pp. 261–270.
[15] G. Golub, C. Greif, An Arnoldi-type algorithm for computing PageRank, BIT 46 (2006) 759–771.
[16] C. Brezinski, M. Redivo-Zaglia, The PageRank vector: properties, computation, approximation, and acceleration, SIAM J. Matrix Anal. Appl. 28 (2006) 551–575.
[17] A. Sidi, Vector extrapolation methods with applications to solution of large systems of equations and to PageRank computations, Comput. Math. Appl. 56 (2008) 1–24.
[18] D. Gleich, A. Gray, C. Greif, T. Lau, An inner-outer iteration for computing PageRank, SIAM J. Sci. Comput. 32 (2010) 349–371.
[19] A. Langville, C. Meyer, Deeper inside PageRank, Internet Math. 1 (3) (2005) 335–380.
[20] A. Langville, C. Meyer, A survey of eigenvector methods for Web information retrieval, SIAM Rev. 47 (2005) 131–161.
[21] Z. Jia, Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems, Linear Algebra Appl. 259 (1997) 1–23.
[22] G. Wu, Y. Wei, A Power-Arnoldi algorithm for computing PageRank, Numer. Linear Algebra Appl. 14 (2007) 521–546.
[23] G. Wu, Y. Wei, An Arnoldi-Extrapolation algorithm for computing PageRank, J. Comput. Appl. Math. 234 (2010) 3196–3212.
[24] R. Morgan, M. Zheng, A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity, Linear Algebra Appl. 415 (2006) 96–113.
[25] A. Berman, R. Plemmons, Nonnegaitve Matrices in the Mathematical Sciences, SIAM, Philadelphia, PA, 1994.
[26] R. Horn, C. Johnson, Matrix Analysis, Cambridge Universtiy Press, 1985.
[27] T. Haveliwala, S. Kamvar, The second eigenvalue of the Google matrix, Technical report, Stanford University, Stanford, CA, 2003.
[28] D. Watkins, The matrix Eigenvalue Problem: GR and Krylov Subspace Methods, SIAM, Philadelphia, 2007.
[29] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, PA, 2003.
[30] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Rev. ed., SIAM, 2011.