NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2012; **19**:73–85
Published online 20 November 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.789

On adaptively accelerated Arnoldi method for computing PageRank

Jun-Feng Yin [1], Guo-Jian Yin [1] and Michael Ng [2,*,†]

[1] *Department of Mathematics, Tongji University, Shanghai, China*
[2] *Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong*

SUMMARY

A generalized refined Arnoldi method based on the weighted inner product is presented for computing PageRank. The properties of the generalized refined Arnoldi method were studied. To speed up the convergence performance for computing PageRank, we propose to change the weights adaptively where the weights are calculated based on the current residual corresponding to the approximate PageRank vector. Numerical results show that the proposed Arnoldi method converges faster than existing methods, in particular when the damping factor is large. Copyright © 2011 John Wiley & Sons, Ltd.

Received 8 May 2010; Revised 16 March 2011; Accepted 9 April 2011

KEY WORDS: PageRank; eigenvalue and eigenvector; power method; Arnoldi process; weighted least squares problem

1. INTRODUCTION

The computation of eigenvalues and eigenvectors of large nonsymmetric matrices is one of the most important topics in the numerical linear algebra community. These kinds of problems often arise from many applications in science and engineering, for instance, quantum chemistry, dynamic structural analysis and Maxwell's equations; for details, see [1] and references therein.

Recently, with the booming development of the internet, web search engines became one of the most important internet tools for information retrieval. PageRank is related to a link analysis algorithm used by the Google internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents in the World Wide Web. Here the weighting obtained by PageRank provides the relative importance of each document. The PageRank weighting are the entries of the dominant eigenvector of the modified adjacency matrix:

$$A = \alpha P + (1-\alpha)E,$$

where P is a column-stochastic matrix (i.e., the dangling nodes are already replaced by columns with $1/n$), α is a damping factor, and E is a rank-one matrix, see [2] for the details.

We note that when α is large, A is dominated by the original hyperlink structure of webpages. It was observed in [3] that α close to 1 does not produce useful ranking results. Usually, α is taken as 0.85. On the other hand, the dominant eigenvector corresponding to the PageRank weighting can be interpreted as the stationary probability distribution vector of the associated Markov chain [4]. For surveys of the PageRank problem, we refer the readers to [5, 6].

*Correspondence to: Michael Ng, Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong.
†E-mail: mng@math.hkbu.edu.hk

Usually, the matrix A involved is extremely large, so that direct decomposition techniques (such as LU and QR decomposition) cannot be considered for computing PageRank. Iterative methods based on matrix–vector products have been widely studied for this computation.

The power method was firstly considered for computing PageRank. However, the eigenvalues of A is the scaling of those of P, except for the dominant eigenvalue, for details, see the proofs in [7,8]. Thus, when the largest eigenvalue of A is not well separated from other eigenvalues and the damping factor α is close to 1, then the power method converges very slowly. Many researchers proposed several methods to accelerate the convergence of the power method. For instance, the quadratic extrapolation method [9], the adaptive method [10], and the block method [2] are employed. The extrapolation method can improve the convergence performance of the power method, however, it still strongly depends on the convergence speed of the power method and it may not be effective at each step of the extrapolation method.

The adaptive method [10] can save the computation of matrix–vector products, while the drawback of the adaptive method is that it cannot improve the convergence performance of the power method. To make the power method more effective, the block structure of the web is taken into use for computing the PageRank. The web structure should be exploited and the matrix should be preprocessed and managed in the right manner so that the effective block method can be employed.

On the other hand, other iterative methods are studied for computing PageRank. Iterative methods based on the Arnoldi process are good alternatives for computing the dominant eigenvector. We note that the largest eigenvalue of PageRank matrix A is known to be 1. Golub and Greif [11] proposed an Arnoldi-type method combined with SVD by considering the known largest eigenvalue as a shift so that the computation of the largest Ritz value is avoided. This approach is very efficient as it can avoid the complex arithmetic even if the largest Ritz value is complex. Wu and Wei [12] also proposed an Arnoldi-type method combined with the power method. In their approach, the thick restarted Arnoldi method was used. However, the largest Ritz value is required to be computed in their method. The complex arithmetic may be needed. In addition, the corresponding Ritz vector may be complex-valued, both the real and imaginary parts of the Ritz vector are taken as the restart vectors in the calculation.

Recently, the idea of introducing weighted inner products into the Arnoldi process has been successfully applied into the solution of linear equation [13] and the computation of eigenvalues [14]. By taking the advantage of the Arnoldi-type algorithm in [11] and the idea of weighted Arnoldi process, we propose a new algorithm for computing the PageRank vector in this paper.

We first present a generalized and refined Arnoldi method based on the general inner product for computing the PageRank and study its properties. In order to speed up the convergence performance for computing the PageRank, we propose to change the weights in the weighted matrix inner product adaptively where the weights are calculated based on the current residual of the approximate PageRank vector. Numerical results show that the performance of the proposed Arnoldi method is better than the other methods, in particular, when the damping factor is close to 1.

The paper is organized as follows. In Section 2, we review the Arnoldi process and the Arnoldi-type method for computing the PageRank. In Section 3, we study the generalized Arnoldi method and its theoretical properties. In Section 4, we propose an adaptively accelerated Arnoldi method for computing the PageRank. Numerical results and comparisons are reported in Section 5. Finally, some concluding remarks are given in Section 6.

2. THE ARNOLDI-TYPE METHOD FOR COMPUTING PageRank

In this section, we briefly review the Arnoldi process and the Arnoldi type method for computing PageRank.

The Arnoldi process was developed by Arnoldi in 1951 [15]. Given a general non-Hermitian matrices $A \in \mathbb{R}^{n\times n}$ and an initial vector $q_0 \in \mathbb{R}^n$, the Arnoldi process gives an orthonormal basis $q_1, q_2, \dots, q_m$ of the Krylov subspace

$$\mathcal{K}_m(A, v) = \{q_0, Aq_0, \dots, A^{m-1}q_0\}.$$

Based on the Arnoldi process, several iterative methods were proposed and studied for the solution of linear equations, for example, GMRES [16], as well as the computation of the eigenvalues, for instance, the implicit restarted Arnoldi method [17, 18]. The Arnoldi process can be implemented with the modified Gram–Schmidt algorithm (MGS) as follows:

Method 2.1. $[Q_m, H_m] = \text{Arnoldi}(A, q_0, m)$

```
1.  Compute $q_1 = q_0/\|q_0\|_2$,
2.  For $j = 1, 2, \ldots, m$ Do:
3.      Compute $w = Aq_j$,
4.      For $k = 1, 2, \ldots, j$ Do:
5.          Compute $h_{kj} = q_k^T w$,
6.          Compute $w = w - h_{kj} q_k$,
7.      EndDo
8.      Compute $h_{j+1,j} = \|w\|_2$
9.      If $h_{j+1,j} = 0$,
10.         Stop and exit.
11.     Else
12.         Set $q_{j+1} = w/h_{j+1,j}$,
13.     EndIf
14. EndDo
```

According to Method 2.1, we have

$$AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T \tag{2.1}$$

and

$$Q_m^T A Q_m = H_m,$$

where $Q_m = [q_1, q_2, \ldots, q_m] \in \mathbb{R}^{n \times m}$ is column-orthogonal and $H_m = \{h_{i,j}\} \in \mathbb{R}^{m \times m}$ is an upper Hessenberg matrix. Because H_m is an orthogonal projection for the matrix A onto the Krylov subspace $\mathcal{K}_m(A, q_0)$, the eigenvalues of H_m can be used as the approximation for that of the original matrix A. When y is an eigenvector of H_m, $Q_m y$ is the approximation eigenvector of A, which is called Ritz eigenvectors, see [18, 19] for the detail. Furthermore, formula (2.1) can be rewritten as

$$AQ_m = Q_m H_{m+1,m},$$

where $H_{m+1,m} = \{h_{i,j}\} \in \mathbb{R}^{(m+1) \times m}$ is also an upper Hessenberg matrix. For detailed properties of the Arnoldi method, we refer the reader to [1].

When m is increasing, the computation and memory requirement of the Arnoldi process will be more expensive. Thus, the restart technique is usually taken so that m is small compared with n, and the computation cost of this kind of projection method is very cheap.

By taking the fact that the largest eigenvalue of PageRank matrix A is known to be 1, Golub and Greif [11] proposed an Arnoldi-type method for computing PageRank as follows:

Method 2.2. Arnoldi-type method

```
1. Choose $q_0$ with $\|q_0\|_2 = 1$.
2. For $l = 1, 2, \ldots$, until convergence, Do:
3.   Compute $[Q_m, H_{m+1,m}] = \text{Arnoldi}(A, q_0, m)$
4.   Compute singular value decomposition $H_{m+1,m} - [I; 0] = U \Sigma V^T$
5.   Compute $q_0 = Q_m v_m$
6.   Compute $r = \sigma_m Q_{m+1} u_m$
7.   If $\|r\|_1 <$ TOL then stop; EndIf
8. EndDo
```

Here, m is the number of the restart, v_m is the right singular vector corresponding to the minimal singular value. Instead of computing the eigenvalues of H_m, a singular value decomposition of

$H_m - [I\,;0]$ is calculated in Method 2.2, where all the singular values and singular vectors are real. Note that the smallest singular value of the shifted Hessenberg matrix will converge to zero in the Arnoldi-type method.

3. THE GENERALIZED ARNOLDI METHOD

In this section, we study the generalized Arnoldi method and its theoretical properties.

Let $G \in \mathbb{R}^{n\times n}$ be a symmetric positive definite (SPD) matrix and $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ be two vectors, then a G-inner product is defined as

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

where g_{ij} is the ith row and jth column element of G. This inner product is well defined if and only if the matrix G is SPD. Assume that $G = Q^T D Q$, where Q is an orthogonal matrix, and $D = \operatorname{diag}\{d_1, d_2, \ldots, d_n\}$ is a diagonal matrix with $d_i > 0$, $i = 1, 2, \ldots, n$. The norm associated with this inner product is defined by

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

which is the called G-norm $\|\cdot\|_G$.

In the following, the generalized Arnoldi (GArnoldi) method can be defined as follows.

Method 3.1. $[\widetilde{Q}_m, \widetilde{H}_m] = \text{GArnoldi}(A, G, q_0, m)$

```
1. Compute $\widetilde{q}_1 = q_0/\|q_0\|_G$,
2. For $j = 1, 2, \ldots, m$ Do:
3.     Compute $w = A\widetilde{q}_j$
4.     For $k = 1, 2, \ldots, j$ Do:
5.         Compute $\widetilde{h}_{kj} = (w, \widetilde{q}_k)_G$
6.         Compute $w = w - \widetilde{h}_{kj}\widetilde{q}_k$
7.     EndDo
8.     Compute $\widetilde{h}_{j+1,j} = \|w\|_G$
9.     If $\widetilde{h}_{j+1,j} = 0$
10.        Stop and exit
11.    Else
12.        Set $\widetilde{q}_{j+1} = w/\widetilde{h}_{j+1,j}$
13.    EndIf
14. EndDo
```

It is clear that when G is the identity matrix, Method 3.1 is reduced to the standard Arnoldi Method 2.1. It is noted that in line 5 of the generalized Arnoldi method 3.1, the inner product

$$(w, \widetilde{q}_k)_G = w^T G \widetilde{q}_k$$

is required to compute. Actually, $G\widetilde{q}_k$ will be used $m-k$ times when $k = 1, 2, \ldots, m-1$. To reduce such additional computation, we suggest to save these vectors when they are firstly computed and employ them when they are required.

Method 3.1 constructs a G-orthogonal basis of $\mathcal{K}_m(A, q_0)$ starting with a vector q_0. Next, we give the properties of Method 3.1 as follows.

Theorem 3.1
The vectors $\widetilde{q}_1, \widetilde{q}_2, \ldots, \widetilde{q}_m$ form a G-orthonormal basis of the subspace $\widetilde{\mathcal{K}}_m(A, q_0) = \operatorname{span}\{\widetilde{q}_1, A\widetilde{q}_1, \ldots, A^{m-1}\widetilde{q}_1\}$. Denote $\widetilde{Q}_m = [\widetilde{q}_1, \widetilde{q}_2, \ldots, \widetilde{q}_m] \in \mathbb{R}^{n\times m}$, then

$$\widetilde{Q}_m^T G \widetilde{Q}_m = I.$$

The proof is easily obtained from the G-orthonormal columns of $\widetilde{Q}_m$.

Theorem 3.2
Denote $\widetilde{Q}_m = [\widetilde{q}_1, \widetilde{q}_2, \ldots, \widetilde{q}_m] \in \mathbb{R}^{n\times m}$ and $\widetilde{H}_m \in \mathbb{R}^{m\times m}$ be the Hessenberg matrix whose nonzero entries are defined by Method 3.1. Then the following relations hold:

$$\begin{aligned} A\widetilde{Q}_m &= \widetilde{Q}_m \widetilde{H}_m + \widetilde{h}_{m+1,m}\widetilde{q}_{m+1} e_m^T & (3.2)\\ \widetilde{Q}_m^T G A \widetilde{Q}_m &= \widetilde{H}_m. & (3.3) \end{aligned}$$

Proof
The relation (3.2) follows from the following equality, which is readily derived from lines 6 and 12 in Method 3.1:

$$A\widetilde{q}_j = \sum_{i=1}^{j+1} \widetilde{h}_{ij}\widetilde{q}_i, \quad j = 1, 2, \ldots, m.$$

Relation (3.3) follows by multiplying both sides of (3.2) by $\widetilde{Q}_m^T G$ and making use of $\widetilde{Q}_m^T G \widetilde{Q}_m = I_m$ from Theorem 3.1. □

Theorem 3.3
Denote $\widetilde{Q}_m = [\widetilde{q}_1, \widetilde{q}_2, \ldots, \widetilde{q}_m] \in \mathbb{R}^{n\times m}$, and $\widetilde{H}_m \in \mathbb{R}^{m\times m}$ be the G-orthogonal matrix and the Hessenberg matrix defined by Method 3.1, $(\lambda_i, v_i), i = 1, 2, \ldots, m$ are the eigenpairs of $\widetilde{H}_m$. If $m = n$, then $\lambda_i (i = 1, 2, \ldots, n)$ are the eigenvalues of A, and the corresponding eigenvectors g_i are defined by

$$g_i = \widetilde{Q}_n v_i, \quad i = 1, 2, \ldots, n. \tag{3.4}$$

Proof
If $m = n$, from $\widetilde{Q}_n^T G \widetilde{Q}_n = I$, we have $\widetilde{Q}_n^{-1} = \widetilde{Q}_n^T G$. Thus,

$$\begin{aligned} \widetilde{H}_n &= \widetilde{Q}_n^T G A \widetilde{Q}_n,\\ &= \widetilde{Q}_n^{-1} A \widetilde{Q}_n. \end{aligned}$$

Then, $\lambda_i (i = 1, 2, \ldots, n)$ are the eigenvalues of A, and the corresponding eigenvectors are given by $\widetilde{Q}_n v_i, i = 1, 2, \ldots, n$. □

We should remark that when $m < n$, the eigenvalues of the Hessenberg matrix $\widetilde{H}_m$ are the approximation to the eigenvalues of A provided by the projection $\widetilde{Q}_m$, and called the Ritz eigenvalues of A. One of the important advantages is that the Ritz approximate eigenvector can be represented by $\widetilde{Q}_m v_i$ where v_i is the eigenvector associated with $\lambda_i, i = 1, 2, \ldots, n$.

It was pointed out in [13] that the generalized Arnoldi method 3.1 is different from the preconditioned Arnoldi method, though the latter was widely used in many applications, for example, eigenvalue problems [14] and Markov chains [20].

Assume that the preconditioner is G, it is easily deduced that $H_m = Q_m^T G A Q_m$ is still valid with $Q_m^T Q_m = I$. However, the eigenvalues of the Hessenberg matrix H_m are the approximation to the eigenvalues of GA, not A, and correspondingly the Ritz value of A can not be obtained. Moreover, the relation (3.4) between the eigenvectors of H_m and those of A do not hold any more.

Because the relation (3.4) is satisfied, similar to the Arnoldi-type method in [11], a generalized Arnoldi method for computing PageRank can be defined as follows:

Method 3.2. Generalized Arnoldi method for computing PageRank

```
1. Choose SPD matrix $G$.
2. Choose $q_0$ with $\|q_0\|_G = 1$.
3. For $l = 1, 2, \ldots$, until convergence, Do:
4.     Compute $[\widetilde{Q}_m, \widetilde{H}_{m+1,m}] = \text{GArnoldi}(A, G, q_0, m)$
5.     Compute singular value decomposition $\widetilde{H}_{m+1,m} - [I; 0]^T = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$
6.     Compute $q_0 = \widetilde{Q}_m \widetilde{v}_m$
7. EndDo
```

Here, we give some remarks about the proposed algorithm.

- Similar with Method 2.2, $\widetilde{v}_m$ in the fifth line is the right singular vector corresponding to the smallest singular value.
- The restart number m of the generalized Arnoldi method can be chosen artificially. Usually, the restart number m should be relatively small for saving the memory and corresponding computation.
- If $\widetilde{h}_{j+1,j}$ in Method 3.2 is small enough, the Arnoldi process breaks down, and by computing SVD as in line 5 of Method 3.2, the PageRank vector can still be obtained.

It is obvious that the different positive-definite G matrices will lead to different accelerated-restarted Arnoldi methods. If the G-norm $\|\cdot\|_G$ is defined as the Euclidean norm, then Method 3.2 is the same with the standard Arnoldi-type method in [11]. Furthermore, if G is a diagonal matrix with positive diagonal elements, it can cover the weighted Arnoldi algorithm developed in [14] for computing the eigenvalues of a nonsymmetric matrix.

Next, we give the property of the residual of the approximate PageRank vector given by Method 3.2 as follows.

Theorem 3.4
Denote $\widetilde{H}_{m+1,m} \in \mathbb{R}^{(m+1)\times m}$ be the Hessenberg matrices defined by Method 3.1. Let u_m and v_m be the left and right singular vector of $\widetilde{H}_{m+1,m} - [I; 0]^T$ associated with the minimal singular value $\widetilde{\sigma}_m$. Denote q be the refined Ritz approximated eigenvector $q = \widetilde{Q}_m v_m$, then,

$$Aq - q = \widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m,$$

and therefore,

$$\|Aq - q\|_G = \widetilde{\sigma}_m.$$

Proof
This follows from multiplying both sides of Method (3.2) by y_i:

$$\begin{aligned} Aq - q &= A\widetilde{Q}_m v_m - \widetilde{Q}_m v_m \\ &= \widetilde{Q}_{m+1}\widetilde{H}_{m+1,m} v_m - \widetilde{Q}_m v_m \\ &= \widetilde{Q}_{m+1}\left[\widetilde{H}_{m+1,m} - \begin{pmatrix} I_m \\ 0 \end{pmatrix}\right] v_m \\ &= \widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m. \end{aligned}$$

It follows that

$$\|Aq - q\|_G = \sqrt{(\widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m)^T G \widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m} = \widetilde{\sigma}_m.$$

□

For an SPD matrix G, we can design a new Arnoldi method similar with [1, 18] to compute the eigenvalues of a nonsymmetric matrix. Here, we give the accelerated Arnoldi method for computing PageRank based on the merit of the Arnoldi-type algorithm in [11].

4. ADAPTIVELY ACCELERATED ARNOLDI METHOD FOR COMPUTING PageRank

In this section, a practical choice of G is proposed from the viewpoint of solving weighted least squares problems. We can adaptively apply in every outer iteration of Arnoldi Method 3.2 by assigning the suitable weights in the residual of the approximate PageRank vector so that the convergence can be improved. It is seen that in every outer iteration of the Arnoldi Method 3.2, we are looking for a vector q satisfying

$$\min \|Aq - q\|_G, \quad q \in \widetilde{\mathcal{K}}_m(A, q_0).$$

Every symmetric positive definite matrix can be diagonalized. For simplicity, we consider G here to be a diagonal matrix, for example, $G = \mathrm{diag}\{\omega_1, \ldots, \omega_n\}$ where $\omega_i > 0$, $i = 1, \ldots, n$. Denote $r = Aq - q \triangleq [r_1, \ldots, r_n]^T$, it is clear that

$$\begin{aligned} \min \|Aq - q\|_G &= \min \|r\|_G \\ &= \min \sqrt{\omega_1 r_1^2 + \omega_2 r_2^2 + \ldots + \omega_n r_n^2} \\ &= \min \sqrt{\sum_{i=1}^{n} \omega_i r_i^2}. \end{aligned}$$

It leads to a weighted least squares problem where ω_i is actually the weight for r_i, the ith component of residual, $i = 1, \ldots, n$.

According to Theorem 3.4, the residual r is computed by $\widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m$. If r_i is small, it means that the convergence speed of the residual in the direction corresponding to r_i is fast; otherwise, it means that the residual of this direction converges very slowly and requires to be accelerated.

One natural idea from solving weighted least squares problems is to strengthen the weight of those components in which converges very slowly. Based on this idea, we can define ω_i as follows:

$$\omega_i = |\tilde{r}_i| / \|\tilde{r}\|_1,$$

where $\tilde{r}$ is the residual computed by the last accelerated Arnoldi process, and $\sum_{i=1}^{n} \omega_i = 1$. Thus, the choice for G is given by

$$G = \mathrm{diag}\{|\tilde{r}| / \|\tilde{r}\|_1\}.$$

Because the residual finally converges to zero, minimizing both $\|r\|_2$ and $\|r\|_G$ will lead to the same solution.

Moreover, it is observed that the residual varies after every outer iteration of the Arnoldi Method 3.2. Thus, the weight of the components of the residual can also adaptively change with the changing of the residual at every outer iteration step. This leads to the adaptively accelerated Arnoldi method for computing the PageRank as follows.

Method 4.1. Adaptively Accelerated Arnoldi method for computing Pagerank

```
1.  Choose $q_0$
2.  Set $G = I$ and $\|q_0\|_G = 1$
3.  For $l = 1, 2, \ldots,$ until convergence, Do:
4.      Compute $[\widetilde{Q}_m, \widetilde{H}_{m+1,m}] = \mathrm{GArnoldi}(A, G, q_0, m)$
5.      Compute singular value decomposition $\widetilde{H}_{m+1,m} - [I; 0]^T = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$
6.      Compute $q_0 = \widetilde{Q}_m \widetilde{v}_m$
7.      Compute $r = \widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m$
8.      If $\|r\|_1 <$ TOL then stop; EndIf
9.      Set $G = \mathrm{diag}\left\{\frac{|r|}{\|r\|_1}\right\}$
10. EndDo
```

The implementation of Method 4.1 is similar to that of Method 2.2 except for the computation of the diagonal matrix G. We also remark that the residual r is computed not only for constructing G but also for checking the convergence using some stopping criteria.

One drawback of the Arnold process based methods is that they require increasingly more memory with the number of iterations or the restarting cycle m, whereas the power-based methods require constant memory. This is because the Arnold process require storing the orthonormal vectors $q_1, q_2, \dots, q_m$ in the modified Gram–Schmidt process, as well as the Hessenberg matrix.

Table I shows the memory required other than A for the k-th iteration for the power method, the power method with quadratic extrapolation (called as 'QE-power'), the Arnoldi-type method without restart (called as 'Arnoldi') and the adaptively accelerated Arnoldi method without restart (called as 'A-Arnoldi'). Here, w, x and r are the intermediate vectors, and Q_k denotes the k orthonormal vectors in the modified Gram–Schmidt process.

Because the computational cost of the adaptively accelerated Arnoldi method is nearly the same as that of the Arnoldi-type method, except that the general Arnoldi method is replaced by the standard Arnoldi method. Table II shows the work for each cycle of the Arnoldi method 2.1 and the general Arnoldi method 3.1. Because the matrix G we chose is a diagonal matrix, the matrix–vector products in terms of G can be implemented by elementwise multiplication. Here, nnz represents the number of nonzero elements of matrix A.

From Table II, it is seen that the computation cost for each cycle of the general Arnoldi method is slightly expensive than that of the Arnoldi method. However, the convergence performance of the Arnoldi method is greatly improved by the adaptively accelerating technique, which will be seen in the next section. In addition, the computational cost of the adaptively accelerated Arnoldi method can be reduced if we added the additional memory for storing $G\widetilde{q}_k$.

It is also seen from Table II that except for the operation of the matrix–vector multiplication, the computation of the norms (inner products) and SAXPY also influence the total computational cost very much. When m increases, the computational cost in every cycle is increasing while the number of total iteration is decreasing. Hence, it is difficult to choose the optimal value of restart to minimize total computational cost (CPU time), which will be further discussed in next section.

5. NUMERICAL RESULTS

In this section, we present the numerical experiments to illustrate the efficiency of the adaptively accelerated Arnoldi method.

In our experiment, we compare the adaptively accelerated Arnoldi method (called as 'A-Arnoldi') with the power method, the power method with the quadratic extrapolation (called as 'QE-power') in [9], and the Arnoldi-type (called as 'Arnoldi') method proposed in [11]. For the power method with

Table I. Intermediate memory required for the k-th iteration for each method.

	dim(n)	dim(k)	total
power	x_k, x_{k-1}		$2n$
QE-power	$x_k, x_{k-1}, x_{k-2}, x_{k-3}$		$4n$
Arnoldi	Q_k, w	H_k	$(k+1)n + k^2/2 + 2k$
A-Arnoldi	Q_k, w, r	H_k	$(k+2)n + k^2/2 + 2k$

QE-power, power method with quadratic extrapolation.

Table II. Computation cost for each cycle of the Arnoldi method and the general Arnoldi method.

line	operation	Arnoldi	A-Arnoldi
3	matrix–vector products	$2mnnz$	$2mnnz$
5	inner products	$m(m+1)n$	$3m(m+1)n/2$
6	saxpy:$x + \alpha y$	$m(m+1)n$	$m(m+1)n$
8	norm computation	$2mn$	$3mn$
12	vector scaling	mn	mn

quadratic extrapolation, it was observed in [9] that the quadratic extrapolation does not necessarily need to be applied too often to achieve maximum benefit. Hence, the quadratic extrapolation was applied every fifth iteration in our experiment. We recorded the number of matrix–vector products in our experiments, which was actually equivalent to the number of iteration steps in these methods.

The test matrices are obtained from: http://snap.stanford.edu/data/index.html and http://www.cs.toronto.edu/∼ tsap/experiments/download/download.html. In Table III, we describe the characteristics of our test matrices, including number of rows (n), number of nonzeros (nnz), number of zero columns ($zcol$), average nonzeros of every row ($annz$) and density (den) which is defined by

$$den = \frac{nnz}{n \times n} \times 100.$$

Here, the number of zero columns is actually corresponding to the number of dangling nodes. In our experiments, the largest test matrix is of size 683,446 and has 7,583,376 nonzeros.

The initial vector is taken as $q_0 = (1, 1, \ldots, 1)^T$. Because the 1-norm is a natural choice for the power method, in our comparisons we chose the 1-norm of the residual as stopping criterion for a fair comparison. For the power method and the power method with quadratic extrapolation, the stopping criterion is

$$\|Aq - q\|_1 \leq 1.0 \times 10^{-6},$$

whereas for the Arnoldi-type method and the adaptively accelerated Arnoldi method, the stopping criterion is

$$\|\widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m\|_1 \leq 1.0 \times 10^{-6}.$$

From Theorem 3.3, it is seen that $Aq - q = \widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m$ so that these stopping criterions are equivalent to each other. We should remark that the computation of $\widetilde{\sigma}_m \widetilde{Q}_{m+1} u_m$ is actually cheaper than that of $Aq - q$ when m is small. Numerical experiments were done on a machine with a 2.10 GHz CPU and a three-gigabyte memory.

5.1. *Choice of the number of restart m*

Table IV shows the number of iterations of the adaptively accelerated Arnoldi method for matrix ‘Search_engines’ when α varies from 0.85 to 0.99 and m grows from 3 to 9, respectively.

Table III. The characteristic of test matrices.

No.	Matrix	n	nnz	$zcol$	$annz$	den
1	Death_Penalty	4,298	21,956	1,639	5.11	1.19×10^{-1}
2	Search_engines	11,659	292,236	4,082	25.07	2.15×10^{-1}
3	Email-Enron	36,692	367,662	0	10.02	2.73×10^{-1}
4	Stanford-web	281,903	2,312,497	172	8.20	2.90×10^{-3}
5	Amazon0505	410,236	3,356,824	13,433	8.18	2.00×10^{-3}
6	Stanford berkeley web	683,446	7,583,376	4,735	11.10	1.60×10^{-3}

Table IV. The number of iteration of A-Arnoldi versus restart number m for ‘Search_engines’ matrix.

α	3	4	5	6	7	8	9
0.85	47	38	32	27	27	26	26
0.90	53	46	42	34	34	31	33
0.95	89	65	54	53	41	38	41
0.99	117	119	116	90	57	49	59

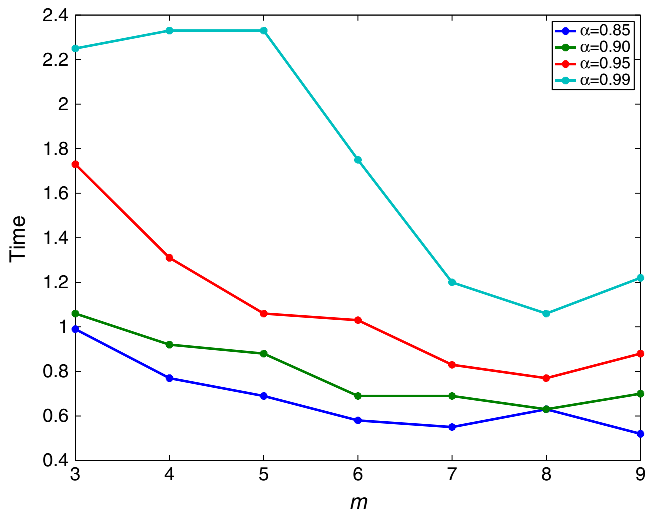

Figure 1. The total CPU time versus restart number m for 'Search_engines' Data.

Table V. Matrix-vector products and CPU time versus damping factor α.

		Power		QE-power		Arnoldi		A-Arnoldi	
No.	α	IT	Time	IT	Time	IT	Time	IT	Time
	0.850	79	0.13	51	0.09	48	0.08	38	0.06
	0.900	118	0.19	68	0.14	67	0.16	47	0.08
1	0.950	226	0.36	108	0.25	112	0.25	57	0.13
	0.990	960	1.61	364	0.77	233	0.55	112	0.22
	0.998	4460	7.44	869	1.81	418	0.97	107	0.22
	0.850	82	1.30	38	0.66	42	0.72	32	0.56
	0.900	126	2.05	51	0.84	58	0.95	42	0.72
2	0.950	252	3.81	72	1.19	93	1.53	54	0.89
	0.990	1102	17.17	151	2.52	260	4.30	116	1.92
	0.998	5092	78.52	271	4.52	324	5.36	201	3.30
	0.850	85	2.06	39	1.01	39	1.09	35	0.91
	0.900	127	2.98	46	1.23	53	1.50	44	1.17
3	0.950	257	6.23	64	1.75	83	2.33	56	1.59
	0.990	1300	31.08	125	3.38	183	5.19	113	3.14
	0.998	6539	156.94	207	5.67	333	9.45	164	4.38
	0.850	86	18.91	71	18.25	83	21.91	57	15.13
	0.900	131	28.97	107	27.66	114	30.27	84	22.81
4	0.950	265	58.64	203	52.52	210	55.89	133	34.22
	0.990	1321	292.50	897	232.48	753	200.41	422	107.77
	0.998	6400	1416.55	3301	855.76	2754	732.81	1036	263.83
	0.850	94	30.20	43	15.58	47	17.67	36	14.28
	0.900	138	42.91	56	20.50	59	22.27	48	17.92
5	0.950	260	81.05	82	30.13	79	29.88	66	25.13
	0.990	977	304.97	208	76.63	229	86.95	127	50.33
	0.998	3843	1200.31	820	302.91	918	348.61	291	186.59
	0.850	88	45.98	75	45.88	88	55.47	58	36.03
	0.900	134	70.52	112	69.55	130	82.52	89	54.41
6	0.950	273	144.19	219	136.70	263	168.00	172	106.28
	0.990	1377	728.16	980	613.09	1163	743.03	632	385.66
	0.998	6820	3596.28	3532	2206.25	4163	2646.47	2167	1319.80

From Table IV, it is seen that the number of iteration is first decreasing then increasing except $\alpha = 0.85$, as the restart number m increases.

Because the storage requirements and computational cost for each cycle of the Arnoldi process increase as m increases, the total CPU time of the adaptively accelerated Arnoldi method will be expensive for large m.

Figure 1 depicts the curves of total CPU time of the adaptively accelerated Arnoldi method versus the number of restart for the 'Search_engines' matrix when $\alpha = 0.85, 0.90, 0.95, 0.99$, respectively.

It is observed from Figure 1 that the optimal m to minimize the total CPU time of the adaptively accelerated Arnoldi method is different when α is changing. So does for different problem. Hence, in our experiment, to keep the memory requirement of Arnoldi process relatively small, we choose $m = 5$ for the Arnoldi-type method in [11] and the adaptively accelerated Arnoldi method and compare them with the power method and the power method with quadratic extrapolation method.

5.2. Comparison of CPU time and the number of iteration

In Table V, we list the number of matrix-vector products of the power method, the power method with quadratic extrapolation, the Arnoldi-type method and the adaptively accelerated Arnoldi method respectively for all the test matrices when α varies from 0.85 to 0.998.

From Table V, it is observed that the adaptively accelerated Arnoldi method performs the best among the four iteration methods. This phenomenon is more obvious when α is relatively large, for instance, when $\alpha = 0.95$ and 0.998.

In Figure 2, we plot the curves of the norm of residual versus iteration number of the power method, the power method with quadratic extrapolation, the Arnoldi-type method and the adaptively accelerated Arnoldi method for 'Death_Penalty' when $\alpha = 0.85, 0.95, 0.99$, and 0.998, respectively.

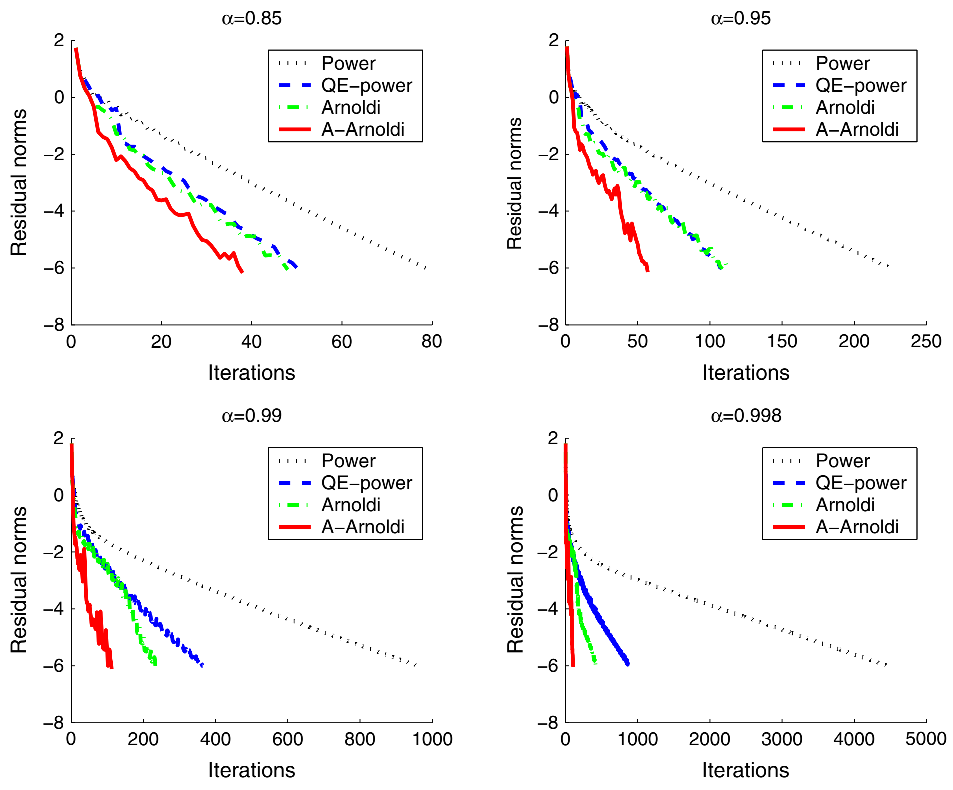

Figure 2. The norm of residual versus iteration number for 'Death_Penalty' data.

From Figure 2, it can be seen that the speedup of our new method is very significantly, especially when the damping factor is close to 1, for example, $\alpha = 0.99$ and $\alpha = 0.998$. This further confirms the advantage of our proposed approach.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new approach for computing the PageRank. The method adaptively changes the weighted least squares problem according to the component of the residual and then using the generalized Arnoldi method to find the approximate PageRank vector. Numerical results showed that the proposed method is quite efficient and better than the existing methods, especially when the damping factor is close to 1.

In the future, the theory of the generalized Arnoldi process and the optimal choice of the matrix G is still required to be further analyzed. In addition, it is also interesting to study the performance of our methods for the more general Markov chains in [21, 22], for instance, 'slowly mixing' Markov chains.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their valuable comments and suggestion on the original manuscript. This work was supported by the Program for Young Excellent Talents in Tongji University, Pujiang Program of Shanghai (No. 09PJ1409800), SRF for ROCS, SEM, NSFC (No. 10801106), and the self-determining research project 'High-Performance Methods, Theories and Implementations for Large-Scale Eigenvalue Problems' of State Key Laboratory of Scientific/Engineering Computing, Chinese Academy of Sciences, China. Research supported in part by HKRGC grants and HKBU FRG grants.

REFERENCES

1. Saad Y. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press: Manchester, UK, 1992.
2. Kamvar SD, Haveliwala TH, Manning CD, Golub GH. Exploiting the block structure of the web for computing PageRank. *Technical Report, SCCM-03-02*, Stanford University, 2003.
3. De Sterck H, Manteuffel TA, Mccormick SF, Nguyen Q, Ruge J. Multilevel adaptive aggregation for markov chains, with application to web ranking. *SIAM Journal on Scientific Computing* 2008; **30**:2235–2262.
4. Langville AN, Meyer CD. A reordering for the PageRank problem. *SIAM Journal on Scientific Computing* 2006; **27**:2112–2120.
5. Berkhin P. A survey on PageRank computing. *Internet Mathematics* 2005; **2**:73–120.
6. Langville AN, Meyer CD. Deeper inside PageRank. *Internet Mathematics* 2005; **1**:335–380.
7. Elden L. A note on the eigenvalues of the Google matrix. *Report LiTH-MAT-R-04-01*, Linkoping University, 2003.
8. Langville AN, Meyer CD. Fiddling with PageRank. *Technical Report*, Department of Mathematics, North Carolina State University, 2003.
9. Kamvar SD, Haveliwala TH, Manning CD, Golub GH. Extrapolation methods for accelerating PageRank computations. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.
10. Kamvar SD, Haveliwala TH, Golub GH. Adaptive methods for the computation of PageRank. *Linear Algebra and its Applications* 2004; **386**:51–65.
11. Golub GH, Greif C. An Arnoldi-type algorithm for computing PageRank. *BIT Numerical Mathematics* 2006; **46**:759–771.
12. Gang Wu, Yimin Wei. A Power–Arnoldi algorithm for computing PageRank. *Numerical Linear Algebra with Applications* 2007; **14**:521–546.
13. Essai A. Weighted FOM and GMRES for solving nonsymmetric linear systems. *Numerical Algorithms* 1998; **18**:277–292.
14. Saberi Najafi H, Ghazvini H. Weighted restarting method in the weighted Arnoldi algorithm for computing the eigenvalues of a nonsymmetric matrix. *Applied Mathematics and Computation* 2006; **175**:1276–1287.
15. Arnoldi WE. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics* 1951; **9**:17–29.
16. Saad Y, Schultz M. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1986; **7**:856–869.
17. Lehoucq RB, Sorensen DC. Deflation Techniques for an Implicitly Restarted Arnoldi Iteration. *SIAM Journal on Scientific Computing* 1996; **17**:789–821.
18. Sorensen DC. Implicitly restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations 1995.

19. Jia Z. Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. *Linear Algebra and its Applications* 1997; **259**:1–23.
20. Philippe B, Saad Y, Stewart WJ. Numerical Methods in Markov Chain Modeling. *Operations Research* 1992; **40**:1156–1179.
21. De Sterck H, Miller K, Sanders G, Winlaw M. Recursively accelerated multilevel aggregation for Markov Chains. *SIAM Journal on Scientific Computing* 2010; **32**:1652–1671.
22. Treister E, Yavneh I. Square and stretch multigrid for stochastic matrix eigenproblems. *Numerical Linear Algebra Appl* 2010; **17**:229–251.