Journal of
Applied & Computational Mathematics

Wen, J Appl Computat Math 2015, 4:5
DOI: 10.4172/2168-9679.1000240

Research Article | Open Access

An Accelerated Two-level Multigrid Method for Markov Chains

Chun Wen*

School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan, China

Abstract

Abstract Based on the quadratic extrapolation method and its generalization, this paper presents an accelerated two-level multigrid method for speeding up the numerical computation of the stationary probability vector of an irreducible Markov chain. It shows how to combine these vector extrapolation methods with the two-level multigrid method on the coarse level in detail. Numerical results on two Markov chain problems are provided to illustrate the effectiveness of our proposed method in terms of reducing the iteration counts and computing time.

Keywords: Markov chain; Two-level multigrid method; Vector extrapolation method; Stationary probability distribution; Accelerator

Mathematics Subject Classification (2000): 65F10: 60J22

Introduction

The use of Markov chains is of interest in a wide range of applications. For example, the web ranking and information retrieval [1-3], queuing systems [4-7], stochastic automata networks [8,9], manufacturing systems and inventory control [10] and communication systems [11,12] and so on. In order to analyze their performance measures, it is required to find their stationary probability distributions π by solving the linear system

$$\pi Q = 0,\ \pi > 0,\ \pi e = 1, \tag{1}$$

where $Q = (q_{ij}) \in R^{n\times n}$ is a generator and $e = (1,1,\ldots,1)^{\mathrm{T}} \in \mathbb{R}^n$ is a column vector.

For a finite irreducible and aperiodic Markov chain, there exists a unique stationary probability distribution π whose elements are strictly greater than zero; see, e.g., [13,14]. Hence, for simplicity, we rewrite (1) as the following homogeneous linear system

$$Ax = 0, with\ A = Q^{\mathrm{T}}, x = \pi^{\mathrm{T}}, \tag{2}$$

where A and x are the transposes of the generator matrix Q and the stationary probability distribution π, respectively. Here the coefficient matrix A has zero column sum, positive diagonal entries and non-positive off diagonal entries.

Recently, there are large amounts of works have devoted to solving the linear system (2). For instance, the matrix splitting iterative methods [13,15-17] Krylov, subspace methods [18-21] and some preconditioning techniques [6,9,17] and so on. What is more, based on the aggregation of Markov states, multigrid methods have been studied in the literature [22-27]. However, with the size of the Markov chains becomes large, the cost of multigrid methods is likely to have an increase. Therefore, it is natural to consider certain strategies to improve their applications.

In this paper, our concern is the two-level multigrid method. Starting from its basic framework [28,29], an accelerated two-level multigrid method is pro-posed for speeding up the numerical computation of the stationary probability vector of an irreducible Markov chain, by applying the quadratic extrapola-tion method discussed by Kamvar, Haveliwala, Manning and Golub [2] and its generalization presented by Sidi [30] to be the accelerators. It shows how to efficiently combine the two-level multigrid method with these vector extrapo-lation methods on the coarse level in detail. The new method is denoted as the two-level-extrapolation (TLE) method. Note that, the idea of improving some iterative methods by combining with vector extrapolation methods is not new [2,30-32]. As a matter of fact, the main algorithmic contribution of the TLE method is that the computation of the coarse-level equation $A_c x_c = 0$ is improved. Numerical experiments on two Markov chain problems are used to illustrate the efficiency and stability of the proposed method.

The rest of this paper is organized as follows. In Section 2, we briefly review and analyze the two-level multigrid method. In Section 3, the accelerated two-level multigrid method for Markov chains is proposed. In Section 4, numerical experiments are provided. Finally, conclusions are made in Section 5.

Two-level Multigrid Method for Markov Chains

In this section, the two-level multigrid method is briefly introduced to solve the stationary probability distribution of Markov chains.

For computing numerical solutions of the linear system $Ax = b$ with b the right-hand vector, certain multigrid methods have been presented in [24,28,29]. It is easy to find that the linear system (2) is in fact a special case of $Ax = b$ when $b = 0$. Without loss of generality, let P be the full rank prolongation matrix of size $n \times n_c$, and R be the restriction operator of size $n_c \times n$, where n_c is the size of the coarse-level matrix A_c. Here operators P and R are created by an automatic coarsening process described below. Then, starting from the description of multigrid methods for the linear system $Ax = b$ [24,28,29], the two-level multigrid method for Markov chains is proposed in Algorithm 1, where the matrix A is known in the linear system (2), G is an aggregation matrix generated by Algorithm 2, and *iter* denotes the number of cycles until the one-norm residual $\|Ax\|_1$ reaches the prescribed tolerance ϵ. Note that the approximate solution x only is normalized at the end of Algorithm 1 rather than in each iteration, since doing like this not only is advantageous for efficient computer implementation, but also is able to save the computing cost.

***Corresponding author:** Chun W, School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China, Tel: +86 28 8320 23; E-mail: wchun17@163.com

Received July 11, 2015; **Accepted** August 17, 2015; **Published** August 21, 2015

Citation: Wen C (2015) An Accelerated Two-level Multigrid Method for Markov Chains. J Appl Computat Math 4: 240. doi:10.4172/2168-9679.1000240

Algorithm 1: Two-level multigrid method for Markov chains

```
1. Give an initial guess $x$ and a prescribed accuracy $\epsilon$. Set $iter = 0$.
2. Do $v_1$ times $x \leftarrow \text{Relax}(A, x)$. % the pre-smoothing.
3. Rebuild aggregation matrix $G$ based on $x$ and $A$ in every cycle. Obtain $R \leftarrow G^T$ and $P \leftarrow \text{diag}(x)G$.
4. Form the coarse-level operator $A_c \leftarrow RAP\text{diag}(G^T x)^{-1}$
   and compute the corresponding coarse-level vector $\bar{x}_c \leftarrow G^T x$.
5. Solve the coarse-level equation $A_c x_c = 0$ by an iterative method with $\bar{x}_c$ being the coarse-level initial vector.
6. Coarse-level correction $x \leftarrow \text{diag}(G\text{diag}(G^T x)^{-1} x_c x)$.
7. Do $v_2$ times $x \leftarrow \text{Relax}(A, x)$. % the post-smoothing.
8. Let $iter \leftarrow iter + 1$ and check convergence. If $\|Ax\|_1 \le \epsilon$, then $x \leftarrow x/\|x\|_1$, otherwise go to step 2.
```

At steps 2 and 7 of Algorithm 1, the weighted Jacobi method with the weight ω, a variant of Jacobi method, is employed to the pre- and post-smoothing processes. Let the coefficient matrix A of (2) be split into

$$A = D - L - U,$$

where D is the diagonal part of the matrix A with $d_{ii} > 0 \forall i$, L and U are the negated strictly lower- and upper-triangular parts of A, respectively. Then the weighted Jacobi relaxation method can be written as

$$x \leftarrow (1-\omega)x + \omega D^{-1}(L+U)x \tag{3}$$

with weight $\omega \in (0,1)$.

At step 3 of Algorithm 1, it is of vital importance how the aggregation matrix G is built based on x and A, that is, we need to understand which nodes should be aggregated into a block and which nodes should be split between their neighbors. Here, in Algorithm 2, we adopt a strength-based aggregation procedure that proposed by De Sterck et al. as our aggregation method, since it is able to improve an algebraically smooth error that varies slowly in a local neighborhood by scaling the original problem matrix A [24,25].

Algorithm 2: Aggregation based on the strength matrix S

```
1. Set $J = 0$.
2. Choose state $j$, which is an unassigned state and has the largest value in the current iterate $x_k$, as the seed point of a new aggregate $G_{J+1}$.
3. Put all unassigned states $i$ that are strongly influenced by the seed point $j(S_{ij} = 1)$ into the new aggregate $G_{J+1}$.
4. Let $J \leftarrow J + 1$. If all the states are assigned, stop.
   Otherwise go to step 2.
5. Obtain the aggregation matrix $G$: if $i \in G_j, j = 1,2,\cdots,J$, then $G_{ij} = 1$, otherwise $G_{ij} = 0$.
```

Note that the computation of the strength matrix S is based on the problem matrix scaled by the current iterate, i.e., $\bar{A} = A\text{diag}(x_k) = (\bar{a}_{ij})$, rather than the original coefficient matrix A (for details see [24]), where diag(·) denotes a diagonal matrix formed with the current iterate x_k. Taking the similar way of defining the strength matrix S [24], then it follows that

$$S_{ik} = \begin{cases} 1, \text{if } i \neq k \text{ and } -\bar{a}_{ik} \ge \theta \max\limits_{i \neq j}(-\bar{a}ij), \\ 0, \text{otherwise}, \end{cases}$$

where θ is a strength of connection parameter. In Algorithm 2, the letter J denotes the number of aggregates, and the aggregation matrix G has the following form

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & . & . & . \\ 1 & 0 & 0 & 0 & . & . & . \\ 0 & 1 & 0 & 0 & . & . & . \\ 0 & 0 & 1 & 0 & . & . & . \\ 0 & 0 & 1 & 0 & . & . & . \\ 0 & 0 & 0 & 1 & . & . & . \\ 0 & 0 & 0 & 1 & . & . & . \\ . & . & . & . & . & & \\ . & . & . & . & & . & \\ . & . & . & . & & & . \end{bmatrix} \in \mathbb{R}^{n \times J}. \tag{4}$$

From (4), the matrix G has the properties that there exists only one element $G_{ij} = 1$ in each row, but each column may have several elements $G_{ij} = 1$, and the sum of the elements in the jth column denotes the number of the nodes which are combined into the jth aggregate.

At step 5 of Algorithm 1, it is necessary to discuss the computation of the coarse-level linear system $A_c x_c = 0$. Clearly solving the coarse-level equation $A_c x_c = 0$ is easier than computing the original system $Ax = 0$, since the size of A_c is smaller than that of A. When the size of A_c is small, the direct methods such as Gaussian elimination are effective. While when the size of A_c is large, iterative methods may be a better choice. Here we employ the Gauss-Seidel method to solve the coarse-level equation $A_c x_c = 0$, since this method makes use of these most recently available component approximations. Let the matrix A_c be split into

$$A_c = D_c - L_c - U_c,$$

where D_c is the diagonal part of the matrix A_c, L_c and U_c are the negated strictly lower- and upper-triangular parts of A_c, respectively. Then the Gauss-Seidel method for the homogeneous systems $A_c x_c = 0$ can be written as

$$x_c^{k+1} = (D_c - L_c)^{-1} U_c x_c^{k}. \tag{5}$$

Let $H_{GS} = (D_c - L_c)^{-1} U_c$, then (5) is equivalent to $x_c^{k+1} = H_{GS} x_c^k$. Hence, the Gauss-Seidel method for $A_c x_c = 0$ is found to be identical to the power method applied to H_{GS} [13]. With the initial approximation $\bar{x}_c$ obtained at step 4 of Algorithm 1, Gauss-Seidel method given in (5) modifies this approximation such that it becomes closer and closer to the true solution at each iteration. However, the procedure has a major disadvantage, that is, it often requires a very long time to converge to the desired solution. In order to overcome this problem, it is natural to consider improving the coarse-level computation by some strategies.

Accelerated Two-level Multigrid Method

In this section, we first give a short introduction to the generalization of the quadratic extrapolation method proposed by Sidi [30], and then show how to combine this vector extrapolation method with the two-level multigrid method on the coarse level for accelerating the numerical computation of the stationary probability distribution for Markov chains.

As far as we know, various kinds of vector extrapolation methods have been discussed in SIAM Review [33]. For example, the polynomial-type vector extrapolation methods which include the minimal polynomial extrapolation (MPE) of Cabay and Jackson [34], and the epsilon vector extrapolation methods which utilize the scalar and vector epsilon methods of Wynn [35,36], and the topological epsilon method of Brezinski [37].

It should be noted that the starting point of the vector extrapolation algo-rithms is to accelerate the convergence of the sequences $\{x_j\}$ generated from such a fixed-point iterative method of the form

$$x_{j+1} = F(x_j), j = 0,1, \cdots; F: \mathrm{R}^n \dashrightarrow \mathrm{R}^n, \tag{6}$$

where x_0 is an initial vector. In recent years, applications of the vector extrap-olation methods to compute the stationary probability distribution of Markov chains have been reported in [2,30-32]. Numerical simulations have also illus-trated that the polynomial-type methods are in general more economic than the epsilon vector extrapolation methods with respect to the computing time and storage requirements. Therefore, in this paper, we apply the polynomial-type vector extrapolation methods, i.e., the quadratic extrapolation method and its generalization, to be our accelerators.

In fact, using vector extrapolation methods as the accelerators is com-mon. For instance, Kamvar et al. have considered the quadratic extrapola-tion method to speed up the computation of the dominant eigenvector of the PageRank problem [2]. Based on Ritz values, Wu and Wei discussed its close connection with the Arnoldis method [32]. Moreover, Sidi reported that it was closely related to the MPE of Cabay and Jackson [34] and thus proposed a generalization of the quadratic extrapolation (GQE) method along with the implementation of MPE [30]. According to [30], the algorithm of the GQE is provided in Algorithm 3.

Algorithm 3: The generalization of quadratic extrapolation method

1. Input the vectors $x_0, x_1, \cdots, x_{k+1}$.

2. Compute $u_i = x_{i+1} - x_0,\ i = 0,1,\cdots, k$, set $U_k = [u_0, u_1, \cdots, u_k]$. Compute the QR-factorization of U_k, namely, $U_k = Q_k R_k$.

Obtain $R_{k-1} := R_k(1:k, 1:k), Q_{k-1} := Q_k(:, 1:k)$.

3. Solve the linear system $R_{k-1}d = -Q_{k-1}^{\mathrm{T}} u_k, d = [d_0, d_1, \cdots, d_{k-1}]^{\mathrm{T}}$.

4. Set $d_k = 1$ and compute $c = [c_0, c_1, \cdots, c_k]^{\mathrm{T}}$ by $c_i = \sum_{j=i}^{k} d_j, i = 0,1,...,k$.

5. Compute $\hat{x}_{k+1} = (\sum_{i=0}^{k} c_i)x_0 + Q_k(R_k c)$.

It is clear that the case $k = 2$ corresponds to the quadratic extrapolation method proposed in [2]. One feature of Algorithm 3 is that there exists a QR-factorization at step 2 for $U_k = Q_k R_k$, where $Q_k \in \mathrm{R}^{n\times(k+1)}$ is unitary, and $Q_k \in \mathbb{R}^{n\times(k+1)}$ is an upper triangular matrix with positive diagonal elements. Precisely, they have

$$Q_k = (q_0, q_1, \cdots, q_k) \in \mathbb{R}^{n\times(k+1)}, Q_k^{\mathrm{T}} Q_k = I_{(k+1)\times(k+1)}.$$

$$R_k = \begin{pmatrix} r_{00} & r_{01} & r_{02} & \cdot & \cdot & \cdot & r_{0k} \\ & r_{11} & r_{12} & \cdot & \cdot & \cdot & r_{1k} \\ & & r_{22} & \cdot & \cdot & \cdot & r_{2k} \\ & & & \cdot & & & \cdot \\ & & & & \cdot & & \cdot \\ & & & & & \cdot & \cdot \\ & & & & & & r_{kk} \end{pmatrix}, r_{ii} > 0\,\forall i.$$

To develop an efficient implementation for the QR-factorization, we apply the modified Gram-Schmidt process (MGS) to vectors $u_0, u_1, \cdots, u_k$; [30,38]. The MGS algorithm is given as follows.

MGS algorithm

Step 1. Compute $r_{00} = (u_0, u_0)^{1/2}$, and set $q_0 = u_0 / r_{00}$;

Step 2. For $j = 1 : k$, set $u_j^{(0)} = u_j$;

Step 3. For $i = 1 : j$, compute $r_{ij} = (q_i, u_j^{(i)})$ and $u_j^{(i+1)} = u_j^{(i)} - r_{ij} q_i$;

Step 4. Compute $r_{jj} = (u_j^{(j)}, u_j^{(j)})^{1/2}$ and $q_j = u_j^{(j)} / r_{jj}$.

From Algorithm 3, we observe that the updated iterate x_{k+1} is given only in terms of the previous iterates $x_j, j = 0,1,\cdots, k+1$, and no other input is required. For example, when $k = 2$, there are only four vectors x_0, x_1, x_2, x_3 are needed to storage and as the input vectors. Furthermore, according to their analytic properties discussed in [2,30], these vector extrapolation methods are naturally considered as the effective accelerators in order to improve the convergence of the vector sequence $\{x_j\}$. In particular, given the vector $x_0, x_1, \cdots, x_{k+1}$ and the QR-factorization in the MGS algorithm, the opera-tion counts of Algorithms 3 consist of $1/2(k^2+5k+2)$ vector additions, $1/2(k^2+5k)$ scalar-vector multiplications and $1/2(k^2+3k+2)$ inner products [30].

Motivated by the study of [2,24,30,32]. Since using iterative methods like Gauss-Seidel method given in (5) to calculate the coarse-level linear system $A_c x_c = 0$ may require a very long time to converge to the desired solution. Our main contribution is to apply the vector extrapolation method (Algorithm 3) to modify the two-level multigrid method (Algorithm 1) on the coarse level, such that the convergence of calculating the stationary probability distribution of Markov chains becomes faster. The proposed method is denoted as the two-level-extrapolation (TLE) method and given in Algorithm 4.

Algorithm 4: Accelerated two-level multigrid method by GQE

1. Obtain the coarse-level matrix A_c and vector $\bar{x}_c$ by the steps 1-4 of Algorithm 1.

2. Compute the coarse-level equation $A_c x_c = 0$ by Algorithm 3.

(a). Set $\bar{x}_0 = \bar{x}_c$ as the coarse-level initial vector,

(b). Obtain the input vectors $\bar{x}_0, \bar{x}_1, \cdots, \bar{x}_{k+1}$ by (5),

(c). Compute the coarse-level approximate solution x_c by Algorithm 3.

3. Obtain the approximate solution x of (2) by the steps 6-8 of Algorithm 1 and check convergence.

Comparing Algorithm 1 with Algorithm 4, the main difference between our proposed method and the standard two-level multigrid method is that, in the process of computing the coarse-level equation $A_c x_c = 0$, the former takes advantage of Algorithm 3 to obtain the approximate coarse-level solution x_c, while the latter only exploits the Gauss-Seidel method to get the approximate coarse-level solution x_c. Let n_c be the size of the coarse-level operator A_c. Assume m to be the number of using Gauss-Seidel method to solve $A_c x_c = 0$ in Algorithm 1. Then in each cycle of Algorithm 1, the dominant cost spent in calculating the coarse-level equation $A_c x_c = 0$ is $O(mn_c^2)$. However, as analyzed above, given the input vectors and QR-factorization, the total cost of Algorithm 3 is almost $O(k^2 n_c)$ plus $O((k+1)n_c^2)$

[30]. Hence, in each cycle of Algorithm 4, the dominant cost spent in solving the coarse-level equation $A_c x_c = 0$ is $O((k+1)n^2_c)$ because of $n_c \gg k.z$ Generally speaking, since the size of *k* is often taken to be smaller than that of *m* in numerical experiments, and thus it follows $O((k+1)n^2_c) \le O(mn^2_c)$, which indicates that the total cost of the TLE method should be less than that of the TL method. For illustrating this point, numerical experiments are given in the next section.

Numerical Experiments

In this section, we report numerical results obtained by using Matlab 7.0.1 implementation on a Windows XP with 2.93GHz 64-bit processor and 2GB memory. The main goal is to examine the accelerated two-level multigrid method and show its efficiency in improving the numerical solution of the stationary probability distribution for Markov chains. In Algorithm 1, let the number of using Gauss-Seidel method to solve the coarse-level equation $A_c x_c = 0$ be $m = 10$ and $m = 20$, then for simplicity, we denote the standard two-level multigrid method as TL(10) and TL(20), respectively. In Algorithm 3, we set the letter *k* as $k = 2,3,4,5,6,7$, and then the corresponding methods in Algorithm 4 are denoted as TLE(2), TLE(3), TLE(4), TLE(5), TLE(6) and TLE(7), respectively. Here two Markov chain problems studied in [9,25] are considered in our experiments.

Now, some special sets of parameters are supplied in this paragraph from [24,25]. As mentioned in the previous sections, the weighted Jacobi method has been used as the pre- and post-smoothing approaches in Algorithm 1. Let $\nu_1 = \nu_2 = 1$ and set the relaxation parameter $\omega = 0.7$ in our experiments since this value works well for all tests that we have considered, even though it is likely to be problem-dependent. The strength of connection parameter is chosen as $\theta = 0.8$ and the initial guess is generated by randomly sampling the uniform (0, 1) distribution and normalized to one in the one norm. All the iterations are terminated when $\| Ax \|_1 \le \epsilon = 10^{-6}$ with *x* the current approximate solution, or when the computing time (referred to as CPU) exceeds 600 seconds. Finally, numerical results in terms of iteration counts (referred as to IT) and CPU are reported by means of tables, while convergence histories are shown in figures with the number of iterations on the horizontal axis versus Relres (defined as $\log_{10}$ of the updated relative 1-norms, i.e., $\log_{10} \| Ax \|_1$) on the vertical axis.

Example one: Uniform 2D lattice

This test problem is a 2D lattice with uniform weights [25]. It is similar to an isotropic elliptic PDE problem. Here we let the 2D lattice be square and use *h* to denote the number of nodes in every row or column, e.g., $h = 20,40,60$, then the size of rows of the coefficient matrix *A* in the linear system (2) is $n = h^2$.

Table 1 has provided the IT and CPU of the TL and TLE methods for Example one. By making comparisons, we observe that, the IT and CPU of our accelerated two-level multigrid method are less than those of the stan-dard two-level multigrid method. Particularly, the TLE(7) has given the best results. Taking *n*=400 as an example, the iteration counts of the TL(10) and TL(20) are reduced about 91% and 87% by comparing with that of the TLE(7) respectively (Figure 1).

For obtaining an intuitive comparison, Figure 1 has plotted the convergence histories of the TL(10), TL(20), TLE(3), TLE(5) and TLE(7) methods for Example one with *n*=400. It is not difficult to find that the accelerated two-level multigrid method has faster convergence.

Example two: Two-queue over flow networks

This test problem is the two-queue overflow networks with the customer arrival rate and service rate of the servers being λ_i and $\mu_i (i = 1,2)$, respectively. Suppose the number of serves is s_i and the waiting space is $l_i - s_i - 1$ $(i = 1,2)$. Then the size of rows of the matrix *A* in the linear system (2) is given by $n = l_1 l_2$. Here we let (l_1, l_2) = (16, 16), (32, 32) and (64, 32) in the test. The queueing discipline is First-come-first-served. Specifically, we allow the overflow of customers to occur from queue 2 to queue 1 when queue 2 is full and there is still waiting space in queue 1. The description of the two-queue overflow networks and the form of its generator matrix have been presented in a few papers; e.g., [9]. For simplicity, in this test, we set $s_1 = s_2 = 1, \lambda_1 = \lambda_2 = 1$ and $\mu_1 = \mu_2 = 1$.

Numerical results of the TL and TLE methods for this test problem have been given in Table 2. Again, we see that the IT and CPU of our accelerated two-level multigrid method are less than those of the standard two-level multigrid method. Moreover, the higher order

n	400		1600		3600	
	IT	CPU	IT	CPU	IT	CPU
TL(10)	35	6.6144	97	115.8304	115	545.7664
TL(20)	23	3.6280	54	67.9160	66	332.7826
TLE(2)	7	0.9480	7	8.6851	7	35.2121
TLE(3)	5	0.6528	5	6.1523	5	26.2183
TLE(4)	4	0.5580	4	4.9210	4	21.9744
TLE(5)	4	0.5315	4	4.8115	4	20.9744
TLE(6)	3	0.3956	3	3.6160	3	16.8174
TLE(7)	3	0.3718	3	3.5030	3	15.9021

Table 1: IT and CPU of the TL and TLE methods for Example one.

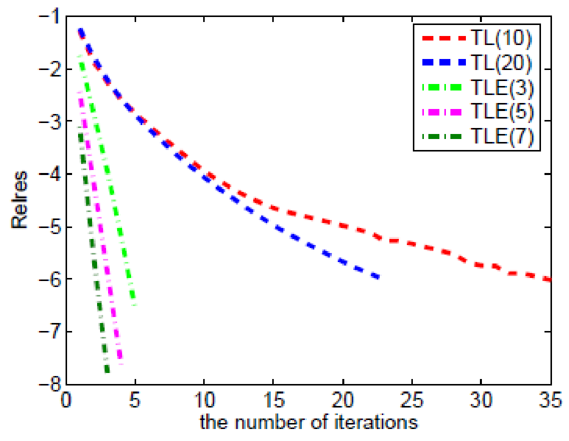

Figure 1: The convergence histories of the TL(10), TL(20), TLE(3), TLE(5) and TLE(7) methods for Example one with *n*=400.

n	256		1024		2048	
	IT	CPU	IT	CPU	IT	CPU
TL(10)	36	3.7404	103	60.7502	218	245.0183
TL(20)	27	2.1347	62	38.4012	188	201.2844
TLE(2)	6	0.4518	6	3.6774	7	7.4840
TLE(3)	5	0.3851	5	3.0647	5	5.3628
TLE(4)	4	0.3045	4	2.4517	4	4.2915
TLE(5)	4	0.3002	4	2.4083	4	4.2887
TLE(6)	3	0.1854	3	1.8926	3	3.2419
TLE(7)	3	0.1786	3	1.8712	3	3.2005

Table 2: IT and CPU of the TL and TLE methods for Example two.

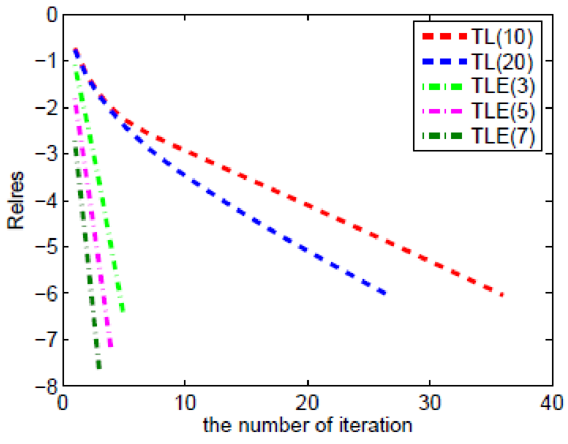

Figure 2: The convergence histories of the TL(10), TL(20), TLE(3), TLE(5) and TLE(7) methods for Example two with n=256.

vector extrapolation methods are used, the less iteration counts and computing time are needed. In particular, the TLE(7) has supplied the best results. For instance, when n=256, the TLE(7) only needs about $8\%-11\%$ of the iteration steps and computing time of the TL method. Therefore, we can say that our proposed methods can efficiently speed up the convergence of the standard two-level multigrid method.

In order to further compare their numerical behavior from an intuitive point, Figure 2 has described the convergence histories of the TL(10), TL(20),TLE(3), TLE(5) and TLE(7) methods for Example two with n=256. These curves illustrate that the accelerated two-level multigrid method outperforms the standard two-level multigrid method once again (Figure 2).

Conclusions

In this paper, an accelerated two-level multigrid method by the use of the quadratic extrapolation method and its generalization has been proposed for improving the numerical calculation of the stationary probability distribution of an irreducible Markov chain. The main algorithm has been given in Algorithm 4. It has shown how to combine Algorithm 1 with Algorithm 3 on the coarse level in detail. Numerical results in Tables 1 and 2 have indicated that the TLE method is superior to the TL method in terms of decreasing the IT and CPU. On the other hand, Figures 1 and 2 have illustrated the fast convergence of the accelerated two-level multigrid method.

Acknowledgement

This research is supported by the Fundamental Research Funds for the Central Universities (ZYGX2014J084)

References

1. Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: Bringing order to the web. Computer Science Department, Stanford.
2. Kamvar SD, Haveliwala TH, Manning CD, Golub GH (2003) Extrapolation methods for accelerating PageRank computations. In: Proceedings of the Twelfth International World Wide Web Conference.
3. Langville AN, Meyer CD (2005) A survey of eigenvector methods for web information retrieval. SIAM Review 47: 135-161.
4. Kim B, Kim J (2015) Stability of a two-class two-server retrial queuing system. Performance aEvaluation.
5. Buchholz P (1994) A class of hierarchical queuing networks and their analysis. Queuing Systems 15: 59-80.
6. Ching WK (2003) Iterative methods for queuing systems with batch arrivals and negative customers. BIT 43: 285-296.
7. Kim B, Kim J (2015) A single server queue with Markov modulated service rates and impatient customers. Performance Evaluation 83: 1-15.
8. Stewart WJ, Atif K, Plateau B (1995) The numerical solution of stochastic automata networks. Euro J Operat Res 86: 503-525.
9. Chan R, Ching WK (2000) Circulant preconditioners for stochastic automata networks. Numer Math 87: 35-57.
10. Buzacott J, Shanthikumar J (1993) Stochastic Models of Manufacturing Systems. Prentice-Hall International Editions, New Jersey.
11. Heffes H, Lucantoni D (1986) A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. IEEE J Select Areas Commun 4: 856-868.
12. Young H, Byung B, Chong K (1992) Performance analysis of leaky-bucket bandwidth enforcement strategy for bursty traffics in an ATM network. Comput Net ISDN Syst 25: 295-303.
13. Stewart WJ (2009) Probability, Markov chains, Queues, and Simulation. Princeton Univ Pr.
14. Latouche G, Ramaswami V (1999) Introduction to Matrix Analytic Methods in Stochastic Modeling. ASA-SIAM Series on Statistics and Applied Probability.
15. Wen C, Huang TZ, Wang C (2011) Triangular and skew-symmetric splitting method for numerical solutions of Markov chains. Comput Math Appl 62: 4039-4048.
16. Saad Y (2003) Iterative Methods for Sparse Linear Systems. 2nd ed, SIAM, Philadelphia, PA.
17. Philippe B, Saad Y, Stewart WJ (1992) Numerical methods in Markov chain modeling. Operations Research 40: 1156-1179.
18. Freund RW, Hochbruck M (1994) On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling. Numer Linear Algebra Appl 1: 403-420.
19. Schneider O (2005) Krylov subspace methods and their generalizations for solving singular operator equations with applications to continuous time Markov chains. der Technischen Universit¨at Bergakademie Freiberg.
20. Benzi M, Kuhlemann V (2011) Restricted Additive Schwarz Methods for Markov Chains. Numer Linear Algebra Appl 18: 1011-1029.
21. Saad Y (1995) Preconditioned Krylov subspace methods for the numerical solution of Markov chains. Computations with Markov chains, Kluwer academic publishers pp: 49-64.
22. Wen C, Huang TZ, Wu DA, Li L (2011) The finest level acceleration of multilevel aggregation for Markov chains. International Journal of Numerical Analysis and Modeling, Series B 2: 27-41.
23. Isensee C, Horton G (2004) A multi-level method for the steady state solution of Markov chains. Simulation and Visualization, Magdeburg, Germany.
24. Sterck HD, Manteuffel TA, McCormick SF, Nguyen Q, Ruge J (2008) Multilevel adaptive aggregation for Markov chains, with applications to web ranking. SIAM J Sci Comput 30: 2235-2262.
25. Sterck HD, Manteuffel TA, McCormick SF, Miller K, Pearson J et al (2010) Smoothed aggregation multigrid for Markov chains. SIAM J Sci Comput 32: 40-61.
26. Buchholz P (2000) Multilevel solutions for structured Markov chains. SIAM J Matrix Anal Appl 22: 342-357.
27. Buchholz P, Dayar T (2007) On the convergence of a class of multilevel methods for large, sparse Markov chains. SIAM J Matrix Anal Appl 29: 1025-1049.
28. Brandt A, McCormick S, Ruge J (1984) Algebraic multigrid (AMG) for sparse matrix equations, Evans DJ(Ed.) Sparsity and its Applications.
29. Vanek P, Mandel J, Brezina M (1996) Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. Computing 56: 179-196.
30. Sidi A (2008) Vector extrapolation methods with applications to solution of large systems of equations and to Page Rank computations. Computers and Math Appl 56: 1-24.

31. Pu BY, Huang TZ, Wen C (2014) A preconditioned and extrapolation-accelerated GM-RES method for PageRank. Appl Math Letters 37: 95-100.

32. Wu G, Wei Y (2010) An Arnoldi-extrapolation algorithm for computing PageRank. J Comput Appl Math 234: 3196-3212.

33. Smith DA, Ford WF, Sidi A (1987) Extrapolation methods for vector sequence. SIAM Review 29: 199-233.

34. Cabay S, Jackson LW (1976) A polynomial extrapolation method for finding limits and antilimits of vector sequences. SIAM J Numer Anal 13: 734-752.

35. Wynn P (1962) Accelerated techniques for iterated vector and matrix problems. Math Comp 16: 301-322.

36. Wynn P (1956) On a device for computing $e_m(S_n)$ transformation. Math Tab Other Aids Comput 10: 91-96.

37. Brezinski C (1975) Generalizations de la transformation de Shanks, de la table de Pad´e, et de 1'ϵ-algorithme. Calcolo 11: 317-360.

38. Sidi A (1991) Efficient implementation of minimal polynomial and reduced rank extrapolation methods. J Comput Appl Math 36: 305-337.