THE FINEST LEVEL ACCELERATION OF MULTILEVEL AGGREGATION FOR MARKOV CHAINS

CHUN WEN, TING-ZHU HUANG, DE-AN WU, AND LIANG LI

Abstract. In this paper, we consider a class of new accelerated multilevel aggregation methods using two polynomial-type vector extrapolation methods, namely the reduced rank extrapolation (RRE) and the generalization of quadratic extrapolation (GQE) methods. We show how to combine the multilevel aggregation methods with the RRE and GQE algorithms on the finest level in order to speed up the numerical computation of the stationary probability vector for an irreducible Markov chain. Numerical experiments on typical Markov chain problems are reported to illustrate the efficiency of the accelerated multilevel aggregation methods.

Key words. Markov chains, multilevel aggregation, acceleration, vector extrapolation methods.

1. Introduction

The use of Markov chains is of interest in a wide range of applications, including information retrieval and web ranking [12, 28, 29, 32], queueing systems [16] and stochastic automata networks [14], as well as performance modeling of computer and communication systems, dependability and security analysis, and analysis of biological systems [40].

In this paper, we study a class of new accelerated multilevel aggregation methods which are efficient for computation of the stationary probability vector of an irreducible Markov chain. Mathematically, the problem to be solved is given by

\[ Bx = x, \quad x_i \geq 0 \quad \forall i, \quad ||x||_1 = 1, \]

where \( B = (b_{ij}) \in \mathbb{R}^{n \times n} \) is a column stochastic matrix, i.e., \( 0 \leq b_{ij} \leq 1 \) \( \forall i,j \) and \( e^T B = e^T \), with \( e \) the column vector with all elements as one, and \( x \in \mathbb{R}^n \) is the stationary probability vector of Markov chains. In fact, if \( B \) is irreducible [5, 40], which is equivalent to the existence of a directed path from any vertex \( i \) to any other vertex \( j \) in the directed graph of \( B \), then \( x \) is the unique solution to the linear system (1). Moreover, the stationary probability vector \( x \) of the Markov chains satisfies the inequality \( x_i > 0 \) \( \forall i \).

For convenience, we rewrite the equation (1) as

\[ Ax = 0, \]

with

\[ A = I - B, \]

where \( I \) is an identity matrix, and \( A \) is a singular \( M \)-matrix with diagonal elements being the negative column sums of its off-diagonal elements. Hence, our interest is...
to solve the $n \times n$ homogeneous linear system (2) corresponding to an irreducible Markov chain.

Iterative procedures are commonly used numerical methods to compute the stationary probability vector for an irreducible Markov chain. Examples of iterative techniques include the power methods for calculating the dominant eigenvector of the matrix $B$ [26–28, 33, 38], the Gauss-Seidel, SOR and SSOR iteration methods based on splitting of the matrix $A$ [33, 34], the iterative aggregation/disaggregation algorithms for Markov chains [30, 31, 36, 41], the hybrid algorithm for queueing systems [48], and the well-known Krylov subspace methods such as the Arnoldi’s algorithms [23, 45], BiCGSTAB and GMRES methods [33–35].

However, the iteration methods are likely to suffer from a slow convergence for some linear systems, for example, computing the principal eigenvector for Google matrix. Thus it is necessary to employ the idea of preconditioning. Philippe, Saad and Stewart considered three different incomplete factorizations: ILU0, ILUTH and ILUK as preconditioners for numerical solutions of Markov chain modelling in [33]. Virnik presented an algebraic multigrid preconditioner for $M$-matrices in [44]. Benzi and Uçar developed the block triangular and product preconditioners based on the alternating iteration [1] for $M$-matrices and the Markov chain problems in [2] and [3], respectively. In addition, the applications of circulant preconditioners for Markov chains had been report in [14, 15].

Recently, multilevel methods based on aggregation of the Markov states have been studied in the literature [17–20, 24, 25]. Isensee and Horton considered multi-level methods for the steady state solution of a continuous-time (CTMC) and discrete-time (DTMC) Markov chains in [25] and [24], respectively. De Sterck, Manteuffel, McCormick, Nguyen and Ruge proposed a multilevel adaptive aggregation (MAA) method to calculate the stationary probability vector of Markov matrices in [18]. As already showed in [18], the multilevel method is a special case of the adaptive smoothed aggregation [8] and adaptive algebraic multigrid methods [7] for sparse linear systems.

Thereafter, De Sterck et al. proposed several strategies to accelerate the convergence of the multilevel aggregation methods. Such strategies include the application of a smoothing technique to the interpolation and restriction operators [17], and analyzing a recursive accelerated multilevel aggregation method by computing quadratic programming problems with inequality constraints [20]. In particular, a top-level acceleration of adaptive algebraic multilevel method was considered by finding a linear combination of previous fine-level iterates, so that it minimizes a functional over a subset of the probability vectors in [19]. The active set method from matlab’s quadprog function [22] was used in their implementations.

Here, we consider a class of new accelerated multilevel aggregation methods by the use of two polynomial-type vector extrapolation methods: the reduced rank extrapolation (RRE) and the generalization of quadratic extrapolation (GQE) methods proposed by Sidi [38]. In fact, the idea to improve iteration methods by combining with vector extrapolation methods is not new; see [10, 11, 28, 39, 45]. This paper shows how to combine the multilevel aggregation methods with RRE and GQE algorithms on the finest level in order to speed up the numerical calculation of the stationary probability vector for an irreducible Markov chain. In numerical experiments, the accelerated multilevel aggregation methods are tested using three representative Markov chain problems. The problems include the nearly completely decomposable (NCD) Markov chains, which are difficult to solve since they consist
of groups of states that are strongly interacted among each other but weakly interacted among the groups themselves. In particular, the necessary iteration counts increase rapidly with the size of the weak interactions approaching zero [24, 25].

The remainder of this paper is organized as follows. In Section 2, we briefly review the multilevel aggregation methods for the Markov chains [6, 42, 43]. Procedures to combine the multilevel aggregation methods with GQE and RRE algorithms on the finest level are presented in Section 3. Numerical simulations for three Markov chain problems are given in Section 4. Finally, conclusions are reported in Section 5.

2. Multilevel aggregation for Markov chains

In this section, we briefly review the multilevel aggregation methods to compute the stationary probability vector of Markov chains [6, 42, 43]. The main idea of a multilevel aggregation method is to transfer a large linear system into a smaller one by some aggregation strategies so that the numerical solution can be computed in an efficient way.

The basic multigrid algorithm for solving linear equations $Ax = b$ has been presented in [6, 42, 43]. Clearly, system (2) is a special case with $b = 0$. Using the same notations as in [6, 42, 43], let $P_l$ be the full rank prolongation matrices of size $n_l \times n_{l+1}$, and $R_l$ be the restriction operators of size $n_{l+1} \times n_l$, $l = 1, \ldots, L - 1$. Here, the operators $P_l$ and $R_l$ are created by an automatic coarsening process, in which the coarse-level matrices are constructed by $A_{l+1} = R_l A_l P_l$, where $A_1 = A$. It is natural to number the levels so that the finest level is 1 and the coarsest level is $L$. Therefore, starting from the descriptions of the basic multigrid method for $Ax = b$, we have the following multilevel aggregation method for Markov chains, which is similar to that reported in [17–20].

Algorithm 1: Multilevel Aggregation Method, $x \leftarrow MA(A, x, \nu_1, \nu_2, \alpha)$

1. Pre-smoothing: Apply $\nu_1$ times $x \leftarrow N(Relax(A, x))$.
2. Construct $Q$ according to the automatic coarsening process described here. Obtain $R \leftarrow Q^T$ and $P \leftarrow \text{diag}(x)Q$.
3. Form the coarse-level matrix $A_c \leftarrow \text{RAP}$, and compute $x_c \leftarrow Q^T x$.
4. If on the coarsest level, solve $A_c x_c = 0$ by a direct method. Otherwise apply $\alpha$ iterations of this algorithm $x_c \leftarrow MA(A_c\text{diag}(Q^T x)^{-1}, x_c, \nu_1, \nu_2, \alpha)$.
5. Coarse-level correction: $x \leftarrow P(\text{diag}(Q^T x)^{-1}) x_c$.
6. Post-smoothing: Apply $\nu_2$ times $x \leftarrow N(Relax(A, x))$.

In Algorithm 1, $A = A_1$ is given in (2), $x$ is an initial guess vector, $A$, denotes the coarse-level matrix, and $x_c$ is the corresponding coarse-level vector, $Q$ is the aggregation matrix generated by the aggregation method, $P$ and $R$ are the prolongation and restriction operators, respectively. For the pre- and post-smoothing, the following weighted Jacobi method with weight $\omega$ is employed. Let the matrix $A$ in (2) is split into the form

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

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

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

with the weight $\omega \in (0, 1)$. Here, we let $N(\cdot)$ denote the normalization operator defined by $N(x) := x/\|x\|_1 \forall x \neq 0$. Note that, a normalization is performed after
each relaxation process to ensure that the fine-level iterates $x_i$ can be interpreted as approximations to the stationary probability vector.

It is necessary to apply the direct solver for the coarsest level. The direct method used at step 4 of Algorithm 1 is based on the following theorem.

**Theorem 1** (Theorem 4.16 in [4]). If $A$ is an irreducible singular $M$-matrix, then each of its principal submatrices other than $A$ itself is a nonsingular $M$-matrix.

If $A_L$ is the coarsest-level operator, then we use the direct method presented in the coarsest-level algorithm below to solve the coarsest-level equation $A_Lx_L = 0$.

**Coarsest-level algorithm**

1. Compute $N_L := \text{size}(A_L, 1)$;
2. Determine $A_{Lp} := A_L(1 : N_L - 1, 1 : N_L - 1)$;
3. Determine $b_{Lp} := -A_L(1 : N_L - 1, N_L)$;
4. Compute $x_{Lp} := A_{Lp}^{-1}b_{Lp}$; let $x_{Lp}(N_L) = 1$; and set the coarsest-level solution $x_L = x_{Lp}/\|x_{Lp}\|_1$.

It is of vital importance to understand how the nodes are aggregated in step 2 of Algorithm 1, that is, we need to understand which nodes should be aggregated into a block and which nodes should be split between their neighbors. Here, we apply a neighborhood-based technique as our aggregation method since it is able to provide well-balanced aggregates of approximately equal size, and leads to a more regular coarsening throughout the automatic coarsening process; see [19, 20, 43].

Note that the neighborhood-based aggregation for Markov chains is based on the matrix scaled by the current iterate, i.e., $\bar{A} = A \text{diag}(x_k)$, rather than the original coefficient matrix $A$ (for details see [18, 24, 25]), where $\text{diag}(\cdot)$ denotes a diagonal matrix formed with the current iterate $x_k$. Here, a node $i$ is said to be strongly connected to node $j$ in the graph of $\bar{A}$ if

$$-\bar{a}_{ij} \geq \theta \max_{k \neq i} \{-\bar{a}_{ik}\} \quad \text{or} \quad -\bar{a}_{ji} \geq \theta \max_{k \neq j} \{-\bar{a}_{jk}\},$$

where $\theta$ is a strength of connection parameter. Suppose $\mathcal{N}_i$ is the set of all points which are strongly connected to $i$ in the graph of $\bar{A}$ including node $i$ itself. Then the neighborhood-based algorithm is given as follows [19, 20].

**Algorithm 2: Neighborhood-Based Aggregation,**\{\(Q_j\)\}_{j=1}^J \leftarrow \text{NBA}(\bar{A}, \theta)

1. Set $R = \{1, \cdots, n\}$ and $J = 0$.
2. Assign entire neighborhoods to aggregates:
   - for $i \in \{1, \cdots, n\}$, construct strong neighborhoods $\mathcal{N}_i$ based on (5),
   - if $\mathcal{N}_i \subset R$ then $J \leftarrow J + 1$, $Q_j \leftarrow \mathcal{N}_i$, $Q_j \leftarrow \mathcal{N}_i$, $R \leftarrow R \setminus \mathcal{N}_i$.
3. Determine the remaining points in the most connected aggregates:
   - while $R \neq \emptyset$, pick $i \in R$ and set $j = \text{argmax}_{k=1, \cdots, J} \text{card}(\mathcal{N}_i \cap Q_k)$,
   - set $Q_j \leftarrow Q_j \cup \{i\}$ and $R \leftarrow R \setminus \{i\}$.
4. Construct the aggregation matrix $Q$:
   - if $i \in Q_j$, $j = 1, \cdots, J$, then $Q_{ij} = 1$, otherwise $Q_{ij} = 0$.\]
Note that the aggregation matrix $Q$ in Algorithm 2 has the following form

$$Q = \begin{bmatrix}
1 & 0 & 0 & \cdots \\
1 & 0 & 0 & \cdots \\
0 & 1 & 0 & \cdots \\
0 & 0 & 1 & \cdots \\
0 & 0 & 0 & 1 \\
\vdots & \vdots & \vdots & \ddots \\
\end{bmatrix}_{n \times J},$$

where $J$ denotes the number of aggregates. Clearly, from (6), the matrix $Q$ has the properties that there exists only one element $Q_{ij} = 1$ in each row, but each column may have several elements $Q_{ij} = 1$, and the sum of the elements in the $j$th column denotes the number of the nodes combined into the $j$th aggregate.

3. Accelerated multilevel aggregation for Markov chains

In this section, we first give a short introduction to two polynomial-type vector extrapolation methods: the reduced rank extrapolation (RRE) and the generalization of quadratic extrapolation (GQE) methods. The key idea of our work is then presented. Finally, we show how to combine the multilevel aggregation methods with RRE and GQE algorithms.

Various kinds of vector extrapolation methods have been discussed in [39], and this includes the polynomial-type vector extrapolation methods such as the minimal polynomial extrapolation (MPE) of Cabay and Jackson [13] and RRE method of Eddy [21], and the epsilon vector extrapolation methods utilizing the scalar and vector epsilon methods of Wynn [46, 47], and the topological epsilon method of Brezinski [9].

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

$$x_{j+1} = F(x_j), \quad j = 0, 1, \cdots; \quad F: \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

where $x_0$ is an initial guess. In recent years, applications of the vector extrapolation methods to compute the stationary probability vector of Markov chains have been reported in [10, 11, 28, 38, 45]. Numerical simulations have also illustrated 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, two polynomial-type vector extrapolation methods, the RRE and GQE methods proposed in [38], are considered in this paper.

Here we give the definition of the RRE method.

**Definition 1** (see [38]). Let $\{x_j\}$ be a given sequence in $\mathbb{R}^n$, and $u_j = x_{j+1} - x_j$, choose $k \leq n$ arbitrarily and define a matrix $U$ via $U = [u_0, u_1, \cdots, u_k] \in \mathbb{R}^{n \times (k+1)}$. Let $z$ be the least squares solution to the linear system $Uz = 0$ subject to the constraint $\sum_{j=0}^k z_j = 1$; this means that $z$ is the solution to the problem

$$\min_{z_0, z_1, \ldots, z_k} \| \sum_{j=0}^k z_j u_j \|_2, \quad \text{subject to } \sum_{j=0}^k z_j = 1.$$
Then the RRE approximation for the exact solution of system (2) is given by

\[ \tilde{x}_k = \sum_{j=0}^{k} z_j x_j. \]

Clearly, the approximate solution \( \tilde{x}_k = Xz \) has the property

\[ ||\tilde{x}_k||_1 = \sum_{i=0}^{n} (\tilde{x}_k)_i = \sum_{i=1}^{n} \sum_{j=0}^{k} X_{ij} z_j = \sum_{j=0}^{k} z_j \sum_{i=1}^{n} X_{ij} = \sum_{j=0}^{k} z_j = 1 \]

once it is checked to satisfy the condition that \( \tilde{x}_k \geq 0 \). This implies that the updated iterate is also a probability vector of Markov chains. Hence, starting from this point, we find that both the acceleration process considered in this paper and that reported in [1] are based on a linear combination of previous fine-level iterates, i.e., \( \tilde{x}_k = Xz \), which is similar in principle to a preconditioned flexible Krylov wrapper applied to an additive iteration for a linear problem. The main difference between these accelerated methods lies in the computation of the vector \( z \).

Note that the definition of RRE here is the most direct approach for our computational purposes. It is different from the original definition in [21]. We give the algorithm of RRE as follows [38].

**Algorithm 3: The Reduced Rank Extrapolation Method (RRE)**

1. Input vectors \( x_0, x_1, \ldots, x_{k+1} \).
2. Compute \( u_i = x_{i+1} - x_i, i = 0, 1, \ldots, k \), and set \( U_k = [u_0, u_1, \ldots, u_k] \).
3. Compute the QR-factorization of \( U_k \), namely, \( U_k = Q_k R_k \).
4. Determine \( R_{k-1} := R_k(1 : k, 1 : k), Q_{k-1} := Q_k(:, 1 : k) \).
5. Solve the linear system \( R_k^T R_k d = e, d = [d_0, d_1, \ldots, d_k]^T, e = [1, 1, \ldots, 1]^T \in R^{k+1} \).
6. Set \( \lambda = \sum_{i=0}^{k} d_i \), compute \( \gamma = [\gamma_0, \gamma_1, \ldots, \gamma_k]^T \) by \( \gamma_i = d_i / \lambda, i = 0, 1, \ldots, k \).
7. Compute \( \xi = [\xi_0, \xi_1, \ldots, \xi_{k-1}]^T \) by \( \xi_0 = 1 - \gamma_0, \xi_j = \xi_{j-1} - \gamma_j, j = 1, \ldots, k - 1 \).
8. Compute \( \tilde{x}_{k+1} = x_0 + Q_{k-1} (R_{k-1} \xi) \).

Next, we investigate another vector extrapolation method - the quadratic extrapolation method presented by Kamvar, Haveliwala, Manning and Golub for accelerating the computations of the dominant eigenvector for the PageRank; see [28]. Wu and Wei discussed the close connection with the Arnoldi’s method [23] based on Ritz values in [45]. Moreover, Sidi reported that it was closely related to the MPE of Cabay and Jackson [13] and proposed a generalization of the quadratic extrapolation method along with the implementation of MPE; see [38]. Thus, for completeness, the algorithm of GQE is provided as follows [38].

**Algorithm 4: The Generalization of Quadratic Extrapolation (GQE)**

1. Input the vectors \( x_0, x_1, \ldots, x_{k+1} \).
2. Compute \( u_i = x_{i+1} - x_0, i = 0, 1, \ldots, k \), set \( U_k = [u_0, u_1, \ldots, u_k] \).
3. Compute the QR-factorization of \( U_k \), namely, \( U_k = Q_k R_k \).
4. Determine \( R_{k-1} := R_k(1 : k, 1 : k), Q_{k-1} := Q_k(:, 1 : k) \).
5. Solve the linear system \( R_{k-1} d = -Q_{k-1}^T d_k, d = [d_0, d_1, \ldots, d_{k-1}]^T \).
6. Set \( d_k = 1 \) and compute \( c = [c_0, c_1, \ldots, c_k]^T \) by \( c_i = \sum_{j=0}^{k} d_j, i = 0, 1, \ldots, k \).
(5) Compute \( \tilde{x}_{k+1} = (\sum_{i=0}^{k} c_i)x_0 + Q_k(R_k c) \).

Sidi has given the numerically fast, stable and efficient storage MPE and RRE algorithms in [37]. A common feature of the Algorithms 3 and 4 is that they both contain a QR-factorization in step 2 for \( U_k = Q_k R_k \), where \( Q_k \in \mathbb{R}^{n \times (k+1)} \) is unitary, and \( R_k \in \mathbb{R}^{(k+1) \times (k+1)} \) is an upper triangular matrix with positive diagonal elements. Precisely, we have

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

\[
R_k = \begin{pmatrix} r_{00} & r_{01} & \cdots & r_{0k} \\ r_{11} & r_{12} & \cdots & r_{1k} \\ r_{22} & \cdots & \cdots & r_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ r_{kk} & \end{pmatrix}, \quad r_{ii} > 0 \forall i.
\]

To develop an efficient implementation for the QR-factorization, we apply the modified Gram-Schmidt process (MGS) to vectors \( u_0, u_1, \cdots, u_k \), see [37, 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 Algorithms 3 and 4, we observe that the updated iterate \( \tilde{x}_{k+1} \) is given only in terms of the previous iterates \( x_j \), \( j = 0, 1, \cdots, k + 1 \), and no other input is required. This is similar to the analytic properties of RRE and GQE methods [28, 38]. Hence, RRE and GQE methods can be used as effective accelerators to improve the convergence of the vector sequence \( \{x_j\} \). In particular, given the vectors \( x_0, x_1, \cdots, x_{k+1} \) and the QR-factorization in the MGS algorithm, the operation counts of Algorithms 3 and 4 consist of \( \frac{1}{2}(k^2 + 5k + 2) \) vector additions, \( \frac{1}{2}(k^2 + 5k) \) scalar-vector multiplications and \( \frac{1}{2}(k^2 + 3k + 2) \) inner products [38].

The present work is motivated from the study of the excellent papers in [19, 20, 38]. In particular, De Sterck, Miller, Manteuffel and Sanders have improved the convergence of the multilevel aggregation methods by selecting a linear combination of previous iterates to minimize a functional within the space of probability vector (for details see [19]). Similar to the theoretical analysis in [19], we suppose there is a sequence of iterates \( \{x_i\}_{i=1}^{\infty} \) obtained by Algorithm 1, where \( x_i \geq 0 \). Then at the \( k \)th outer iteration, let

\[
X = [x_k, x_{k-1}, \cdots, x_{k-m+1}] \in \mathbb{R}^{n \times m}
\]

be a matrix consisting of the last \( m \) iterates with \( x_k \) being the newest, where \( m \) is called the window size. As mentioned above, all the probability vectors \( x_i \) are nonnegative, thus the matrix \( X \) has the following properties:

\[
x_i \geq 0 \text{ and } ||x_i||_1 = 1, \quad i = 1, 2, \cdots .
\]

Combining with Definition 1, the problem to be solved has been transformed into computing the vector \( z \) satisfying \( \sum_{i=1}^{m} z_i = 1 \). Thus we obtain an updated probability vector \( \tilde{z}_k = X z = z_1 x_k + z_2 x_{k-1} + \cdots + z_m x_{k-m+1} \), which is a linear combination of the last \( m \) iterates. It has been reported in [19] that computing the vector \( z \) is
equivalent to solving a small quadratic programming problem (QPP) for any window size \( m \geq 2 \), in which the active set method from matlab’s quadprog function [22] has been used in the implementations.

Therefore, it is natural to consider a new accelerated multilevel aggregation method for the window size \( m \geq 2 \). Our main contribution is to show how to combine the multilevel aggregation method with RRE and GQE algorithms on the finest level to accelerate the numerical computation of the stationary probability vector for the Markov chains.

Algorithm 5: Accelerated Multilevel Aggregation Methods by RRE and GQE, \( x \leftarrow \text{AMA}(A, \tilde{x}_0, m, \epsilon) \)

1. Set \( k = 1 \), if no initial guess is given, choose \( \tilde{x}_0 \).
2. Implement the multilevel aggregation method, \( x_k \leftarrow \text{MA}(A, \tilde{x}_{k-1}, \nu_1, \nu_2, \alpha) \).
3. Set \( m \leftarrow \min\{M, k\} \).
4. Set \( X \leftarrow [x_k, x_{k-1}, \cdots, x_{k-m-1}] \).
5. Apply RRE and GQE algorithms to obtain \( \tilde{x}_k \) from \( X \), respectively.
6. If \( \|Ax_k\|_1 > \|Ax_0\|_1 \), then \( \tilde{x}_k \leftarrow x_k \).
7. Check convergence, if \( \|Ax_k\|_1/\|Ax_0\|_1 < \epsilon \), then \( \tilde{x}_k \leftarrow \tilde{x}_k/\|\tilde{x}_k\|_1 \), otherwise set \( k \leftarrow k + 1 \) and go to step 2.

Note that, in Algorithm 5 \( m \) is the window size and \( \epsilon \) is a prescribed tolerance. In particular, there exists a difference in constructing the matrix \( X \) between our Algorithm 5 and the algorithm of [1]. From Algorithms 3 and 4, the RRE and GQE methods require \( k + 2 \) vectors \( x_0, x_1, \cdots, x_{k+1} \) as inputs. That is to say, when the window size is \( m = 2 \), four approximate probability vectors given in the matrix \( X = [x_0, x_1, x_2, x_3] \in \mathbb{R}^{n \times 4} \) are needed as input in the GQE and RRE algorithms. The window size \( m = 2 \) corresponds to the quadratic vector extrapolation method presented in [28]. Hence, the matrix \( X \) is given as the form of the step 4 in Algorithm 5. The efficiency of the accelerated multilevel aggregation algorithms will be demonstrated shortly.

4. Numerical simulations

In this section, we report numerical results obtained using a Matlab 7.0.1 implementation on Window-XP with 2.93GHz 64-bit processor and 2GB memory. The main goal is to examine the new accelerated multilevel aggregation algorithms and to show the speed up for numerical solutions of stationary probability vector for the Markov chains. Here we set the window size as \( m = 2, 3, 4 \). Three typical Markov chain problems studied in [17, 24, 25] have been used in our experiments. We compare the efficiency of the unaccelerated V-cycles (\( \alpha = 1 \) in Algorithm 1) and the unaccelerated W-cycles (\( \alpha = 2 \) in Algorithm 1). For convenience, we let \( W_{\text{GQE}} \) and \( W_{\text{RRE}} \) denote the applications of GQE and RRE methods with W-cycles.

Without loss of generality, special sets of the parameters are employed and they are taken from [17, 19, 20]. As mentioned early, the weighted Jacobi method is used as the pre- and post-smoothing in Algorithm 1. Let \( \nu_1 = \nu_2 = 1 \) and set the relaxation parameter \( \omega = 0.7 \) in the experiments. Note that, even though the values work well for all tests considered here, they are likely to be problem-dependent. The coarsest-level solution is implemented by the coarsest-level algorithm and the strength of connection parameter is chosen as \( \theta = 0.25 \). The initial guess in Algorithm 5 is generated by random sampling with a uniform \((0,1)\) distribution and normalized to one in the one norm. All iterations are terminated when \( \text{res} = \|Ax_k\|_1/\|Ax_0\|_1 \leq 10^{-5} \) with \( x_k \) the current approximate solution.
Numerical results for various accelerated multilevel aggregation methods are now reported, where “n” is the problem size, “lev” is the number of levels in the multilevel aggregation methods, “it” denotes the iteration counts less than 100, “CPU” denotes the computing time, and “$C_{op}$” is the operator complexity of the last cycle, which is defined as the sum of the nonzero entries in all operators on all levels, divided by the nonzero entries in the fine-level operator, i.e., $C_{op} = \sum_{l=1}^{L} \frac{\text{nnz}(A_l)}{\text{nnz}(A_1)}$ with nnz($A$) is the number of nonzero entries in $A$. In addition, if the convergence factor for the $k$th iteration is defined by $\gamma_k := \|Ax_k\|_1/\|Ax_k-1\|_1$, then the effective convergence factor can be defined by $\gamma_{\text{eff}} := \gamma^1/C_{op}$, where $\gamma$ is the geometric mean of the sequence $\{\gamma_k\}$.

The first problem is a one-dimensional (1D) Markov chain with identical transition rates, which often results from a $M/M/1$ queueing system with identical arrival and service rates. The simplest graph of this test problem is illustrated in Fig. 1. In fact, the iteration counts needed to solve this problem increases sharply when the problem size is increasing, since in principle the probability has to be distributed evenly throughout the chain and a number of iterations are required to propagate [24, 25]. Numerical results are presented in Tables 1, 2 and 3.

![Graph for one-dimensional Markov chain with identical transition rates.](image)

Fig. 1: Graph for one-dimensional Markov chain with identical transition rates.

<table>
<thead>
<tr>
<th>n</th>
<th>lev</th>
<th>$C_{op}$</th>
<th>$\gamma_{\text{eff}}$</th>
<th>$\text{it}$</th>
<th>res</th>
<th>$C_{op}$</th>
<th>$\gamma_{\text{eff}}$</th>
<th>$\text{it}$</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>243</td>
<td>2</td>
<td>1.5007</td>
<td>0.7138</td>
<td>20</td>
<td>9.6963e-6</td>
<td>2.0014</td>
<td>0.7766</td>
<td>20</td>
<td>9.6963e-6</td>
</tr>
<tr>
<td>729</td>
<td>3</td>
<td>1.7506</td>
<td>0.9177</td>
<td>64</td>
<td>9.9026e-6</td>
<td>3.0018</td>
<td>0.9231</td>
<td>41</td>
<td>9.0712e-6</td>
</tr>
<tr>
<td>2187</td>
<td>4</td>
<td>1.8754</td>
<td>0.9570</td>
<td>&gt;100</td>
<td>2.8835e-5</td>
<td>4.0017</td>
<td>0.9602</td>
<td>58</td>
<td>9.5623e-6</td>
</tr>
<tr>
<td>4374</td>
<td>5</td>
<td>1.9378</td>
<td>0.9641</td>
<td>&gt;100</td>
<td>9.1344e-5</td>
<td>5.0015</td>
<td>0.9789</td>
<td>87</td>
<td>9.7252e-6</td>
</tr>
</tbody>
</table>

Table 1: Unaccelerated V-cycles and W-cycles for one-dimensional Markov chain.

From Table 1, we observe that the unaccelerated W-cycles is more effective than the unaccelerated V-cycles, and it requires less iterations for the one-dimensional Markov chain. In addition, the convergence factor of the W-cycles is comparable to that of the V-cycles, but the operator complexity of the W-cycles is larger than that of the V-cycles.

Table 2 presents the numerical results of the accelerated W-cycles with GQE and RRE methods for the one-dimensional Markov chain with the window size $m$: $m = 2, 3, 4$. Clearly, both the effective convergence factor and the iteration counts of the accelerated W-cycles are less than those based on the unaccelerated W-cycles. However, the operator complexities are identical. In particular, window size $m = 3$ and $m = 4$ provide the most effective acceleration for the GQE and RRE methods. The accelerated W-cycles with GQE method is superior to that with RRE method. For instance, when the problem size is 2187 or 4374, comparing with the results using the unaccelerated W-cycles shown in Table 1, the iterations has been reduced by 63% for $W_{\text{GQE}}$, and 57% for $W_{\text{RRE}}$. Hence, we conclude that the accelerated
Table 2: Accelerated W-cycles with GQE and RRE methods for one-dimensional Markov chain.

<table>
<thead>
<tr>
<th>Method</th>
<th>Window size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>W_GQE</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>W_RRE</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>lev</th>
<th>C_op</th>
<th>( \gamma_{eff} )</th>
<th>it</th>
<th>( \gamma_{eff} )</th>
<th>it</th>
<th>( \gamma_{eff} )</th>
<th>it</th>
</tr>
</thead>
<tbody>
<tr>
<td>243</td>
<td>2</td>
<td>2.0014</td>
<td>0.7014</td>
<td>12</td>
<td>0.6716</td>
<td>11</td>
<td>0.6818</td>
<td>12</td>
</tr>
<tr>
<td>729</td>
<td>3</td>
<td>3.0018</td>
<td>0.8342</td>
<td>18</td>
<td>0.8553</td>
<td>21</td>
<td>0.8523</td>
<td>19</td>
</tr>
<tr>
<td>2187</td>
<td>4</td>
<td>4.0017</td>
<td>0.9162</td>
<td>28</td>
<td>0.9041</td>
<td>24</td>
<td>0.8947</td>
<td>22</td>
</tr>
<tr>
<td>4374</td>
<td>5</td>
<td>5.0015</td>
<td>0.9573</td>
<td>43</td>
<td>0.9606</td>
<td>47</td>
<td>0.9450</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 3: CPU of unaccelerated W-cycles and accelerated W-cycles with GQE and RRE methods for one-dimensional Markov chain.

<table>
<thead>
<tr>
<th>n</th>
<th>lev</th>
<th>C_op</th>
<th>( \gamma_{eff} )</th>
<th>it</th>
<th>( \gamma_{eff} )</th>
<th>it</th>
<th>( \gamma_{eff} )</th>
<th>it</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>2</td>
<td>0.2500</td>
<td>0.2030</td>
<td>0.1870</td>
<td>0.2030</td>
<td>0.2030</td>
<td>0.2030</td>
<td>0.2030</td>
</tr>
<tr>
<td>81</td>
<td>2</td>
<td>0.5470</td>
<td>0.3600</td>
<td>0.3390</td>
<td>0.3280</td>
<td>0.3750</td>
<td>0.3500</td>
<td>0.4540</td>
</tr>
<tr>
<td>243</td>
<td>2</td>
<td>1.3280</td>
<td>0.8440</td>
<td>0.7810</td>
<td>0.8440</td>
<td>0.9060</td>
<td>0.9060</td>
<td>0.9690</td>
</tr>
<tr>
<td>1458</td>
<td>4</td>
<td>293.30</td>
<td>142.49</td>
<td>137.47</td>
<td>113.53</td>
<td>166.49</td>
<td>132.22</td>
<td>112.63</td>
</tr>
</tbody>
</table>

multilevel aggregation methods with GQE and RRE produce faster convergence than that of the unaccelerated methods.

In Table 3, we report the CPU time using the unaccelerated W-cycles and accelerated W-cycles with GQE and RRE methods. It is clear that compared to the unaccelerated W-cycles, significant saving in computing time ranging from 20%–62% is achieved using the accelerated W-cycles with GQE and RRE methods.

For the second test problem, we consider a birth-death chain with invariant birth and death rates as shown in Fig. 2. Generally speaking, this kind of birth-death chain is used in the study of demographics, queueing theory, performance engineering or in biology [5, 40]. This test problem has also been discussed in [17, 24, 25]. Here, we choose \( \mu = 0.96 \) in the simulations. Numerical results for this problem are given in Tables 4, 5 and 6.

![Fig. 2: Graph for a birth-death chain with \( \mu = 0.96 \).](image)
Table 4: Unaccelerated V-cycles and W-cycles for the birth-death chain with $\mu = 0.96$.  

Table 5: Accelerated W-cycles with GQE and RRE methods for the birth-death chain.

The numerical simulations presented in Tables 4 - 6 for the birth-death chain problem clearly demonstrate the effectiveness and improvement using the accelerated W-cycles method compared with the unaccelerated version. It is observed that the operator complexity increases when the problem size is increasing. For a fixed and large problem size, the window size $m$ could affect the complexity. For example, when $n = 2048$, the operator complexity for the accelerated W-cycles with GQE method is given by 7.04, 7.67 and 16.48 for $m = 2, 3, 4$ respectively. The computing time listed in Table 6 also reveals that more time are required for the accelerated W-cycles with GQE and RRE methods when $m = 3$ compared to those needed with $m = 2$ and 4. For this test problem, the accelerated W-cycles with GQE and RRE using $m = 4$ produce the most efficient results, and significant reduction in CPU time is achieved compared to those needed using the unaccelerated W-cycles.

The final test problem is displayed in Fig. 3, which is a uniform chain with two weak links in the middle. The problem consists of groups of nodes that are strongly connected among each other but have two weak links in the middle of this chains. This is a typical nearly completely decomposable (NCD) Markov chain problem, and it has been discussed in [17, 24, 25]. Let $\tau = 1e - 3$ in the numerical simulation, and the results are given in Tables 7, 8 and 9.

From Table 7, we note that the number of iterations of the unaccelerated W-cycles is less than that of the unaccelerated V-cycles. In addition, the effective
Table 6: CPU of unaccelerated W-cycles and accelerated W-cycles with GQE and RRE for the birth-death chain.

Table 7: Unaccelerated V-cycles and W-cycles for the uniform chain with two weak links in middle (τ = 1e − 3).

convergence factor of the unaccelerated W-cycles is comparable to that of the unaccelerated V-cycles, but the operator complexity of the unaccelerated W-cycles increases linearly as the number of levels in the multilevel aggregation methods is increasing.

Table 8 gives the numerical results of the accelerated W-cycles with GQE and RRE methods for the uniform chain with two weak links in the middle. The effective convergence factor and the iteration counts of the accelerated W-cycles are less than those of the unaccelerated W-cycles, but their operator complexities are identical. The most effective acceleration for the accelerated W-cycles with GQE and RRE methods corresponds when the window size \( m = 4 \).

Based on the CPU reported in Table 9, we observe that the computing time of the unaccelerated W-cycles has been reduced tremendously by about 27%–61% when the accelerated W-cycles with GQE and RRE methods are employed.

5. Conclusions

In this paper, we present a new approach for the finest level acceleration of multilevel aggregation methods for the solution of the stationary probability vector of an irreducible Markov chain. Our work is inspired by the excellent papers reported in [19, 20, 38]. Using a linear combination of the previous fine-level iterates, we construct a class of new accelerated methods to improve the convergence of
the unaccelerated V-cycles and W-cycles methods. Here, we consider two vector extrapolation methods: the reduced rank extrapolation (RRE) [38] and the generalized quadratic extrapolation (GQE) methods [28, 38]. We discuss how to efficiently combine the multilevel aggregation methods with the RRE and GQE algorithms on the finest level. To validate and to demonstrate the effectiveness of the developed accelerated multilevel aggregation methods, numerical simulations are carried out for three typical Markov chain problems. The computational results confirm that the accelerated W-cycles method is very efficient, and it provides a significant saving in computing time compared to the corresponding unaccelerated method.

References


School of Mathematical Sciences/Institute of Computational Science, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P. R. China

E-mail: wchun17@163.com (C. Wen), tingzhuhuang@126.com (T.-Z. Huang)