BULETINUL ACADEMIEI DE ŞTIINŢE
A REPUBLICII MOLDOVA. MATEMATICA
Number 1(92), 2020, Pages 75–88
ISSN 1024–7696

New Algorithms for Finding the Limiting and Differential Matrices in Markov Chains

Alexandru Lazari, Dmitrii Lozovanu

Abstract. New algorithms for determining the limiting and differential matrices in Markov chains, using fast matrix multiplication methods, new computation procedure of the characteristic polynomial and algorithms of resuming matrix polynomials, are proposed. We show that the complexity of finding the limiting matrix is $O(n^3)$ and the complexity of calculating differential matrices is $O(n^{\omega+1})$, where n is the number of the states of the Markov chain and $O(n^\omega)$ is the complexity of the used matrix multiplication algorithm. The theoretical computational complexity estimation of the algorithm is governed by the fastest known matrix multiplication algorithm, for which $\omega < 2.372864$.

Mathematics subject classification: 65C40, 60J22, 90C40, 65F05, 65F15, 15B51.
Keywords and phrases: Discrete Markov Process, The Matrix of Limiting States Probabilities, Differential Matrices, Matrix Multiplication Complexity.

1 Introduction and problem formulation

In this paper we propose new algorithms for determining the limiting and differential matrices in a homogenous discrete Markov process with finite set of states. The proposed algorithms represent a modification of the algorithms from [6, 7], based on results from [2–4, 9–12, 14–16], concerned with fast matrix multiplication methods, efficient computation of the characteristic polynomial of a transition probability matrix and fast algorithms for resuming matrix polynomials. We show that the proposed modified algorithms determine the limit matrix with running time $O(n^3)$ and the differential matrices with running time $O(n^{\omega+1})$, where n is the number of the states of the Markov chain and $O(n^\omega)$ is the complexity of the used matrix multiplication algorithm. We ground all these results for a discrete Markov process with given set of states $X = \{x_1, x_2, \ldots, x_n\}$ and given transition probability matrix $P = (p_{x,y})_{x,y \in X}$. In such a process at every discrete moment of time $t = 0, 1, 2, \ldots$ the stochastic system passes from the current state $x \in X$ to the next state $y \in X$ with probability $p_{x,y}$, where $\sum\limits_{y \in X} p_{x,y} = 1, \ \forall x \in X$.

An important role in studying the evolution of a discrete Markov process is played by the t-step transition matrix $P(t) = P^t$, where an element $p_{x,y}(t)$ of matrix $P(t)$ expresses the probability of the stochastic system to occupy the state y after t transitions if it starts transition in x at the time moment $t = 0$. In [17] Howard presented a general scheme how the matrix $P(t)$ can be expressed via limiting and

differential matrices, however a detailed algorithm how to calculate these matrices has not been exposed. We will consider the matrix of limiting state probabilities $Q = (q_{x,y})_{x,y\in X}$ of the Markov process in the sense of Cesaro limit, i.e. $Q = \lim\limits_{t\to\infty} \frac{1}{t}\sum\limits_{k=1}^{t} P(k)$. An element $q(x,y)$ of matrix Q can be treated as the probability with which the process will occupy the state y after a large number of transitions when the initial state is x, $\forall x, y \in X$.

Algorithms for calculating the limiting and differential matrices have been proposed and grounded in [6, 7]. The computational complexity of these algorithms are $O(n^4)$. Special algorithms with running time $O(n^3)$ that determine only the limiting matrix in a Markov chain can be found in [1, 13]. In this paper we shall use the approach and the corresponding algorithms for determining the limiting and differential matrices from [1, 6, 7]. First a general approach for determining these matrices has been proposed in [6] and then such an approach has been specified in [6, 7]. According to [6], the limiting and differential matrices in the discrete Markov process represent the coefficients $\beta_k(y)$, $y \in (\mathbb{C}\backslash\mathcal{D})\backslash\{1\}$, $k = \overline{0, m(y)-1}$, in the decomposition

$$P(t) = \sum_{y\in\mathbb{C}\backslash\mathcal{D}} \sum_{k=0}^{m(y)-1} \frac{t^k}{y^t}\beta_k(y),\ \forall t \geq n - r, \tag{1}$$

where $\Delta(z) = |I - zP|$, $r = deg(\Delta(z))$, $\mathcal{D} = \{z \in \mathbb{C} \mid \Delta(z) \neq 0\}$ and $m(y)$ is the multiplicity of the root y of the polynomial $\Delta(z)$, $\forall y \in \mathbb{C}\backslash\mathcal{D}$. Also, it was noted that, for each $y \in (\mathbb{C}\backslash\mathcal{D})\backslash\{1\}$, $k = \overline{0, m(y)-1}$, the matrix $\beta_k(y)$ is differential, i.e. the sum of elements across to each row is equal to zero. The unique non-differential component matrix in representation (1) is the stochastic matrix $\beta_0(1) = Q$, the rest of the matrices $\beta_k(1)$, $k = \overline{1, m(1)-1}$, being null.

In the following we present a modification of the algorithms from [6, 7] based on recent optimized methods for matrix multiplication, computation of the characteristic polynomial of transition probability matrix and algorithms for resuming matrix polynomials.

2 Preliminaries

In this section we present some preliminary results regarding the optimal methods for solving matrix multiplication problem, determining the characteristic polynomial and resuming the matrix polynomials that we shall use for our new algorithms for determining the limiting matrix Q and the differential matrices $\beta_k(y)$, $y \in (\mathbb{C}\backslash\mathcal{D})\backslash\{1\}$, $k = \overline{0, m(y)-1}$.

2.1 Fast matrix multiplication

The classical method that multiplies two $n \times n$ matrices (the naive algorithm) needs $O(n^3)$ elementary operations. Actually to multiply two matrices more efficient

algorithms can be used, especially in the case of large n. The first algorithm of matrix multiplication with computation complexity $O(n^\omega)$, where $\omega < 3$, was proposed by Strassen [16] in 1969. He developed an algorithm with $\omega \approx 2.802$. Later, this algorithm has been improved by others (see reference from [3]) and a series of algorithms with $\omega \approx 2,796$ (Pan 1978), $\omega \approx 2,2522$ (Bini 1981), $\omega \approx 2.479$ (Strassen 1986) have been obtained. The last more efficient algorithms was elaborated by Coppersmith-Winograd [14] with $\omega < 2.376$ in 1990 and Williams [3] with $\omega < 2.372873$ in 2014. The fastest known algorithm at the moment is Le Gall version [2], developed in 2014 as an improvement of Coppersmith-Winograd algorithm. He proved that $\omega = 2.3728639\ldots$. The fastest known matrix multiplication algorithms (Coppersmith–Winograd, Williams, Le Gall) are frequently used as a building block in other algorithms to prove theoretical time bounds. However, unlike the Strassen algorithm, they are considered as galactic algorithms and are not used in practice due to their advantage only for matrices very large, which are not able to be processed by modern hardware. Despite of above mentioned disadvantages, there exist parallel algorithms for computing the product of two $n \times n$ matrices. D'Alberto and Nicolau studied the adaptive Winograd's matrix multiplications in [9] and Ballard, Demmel, Holtz, Lipshitz and Schwartz described a communication-optimal parallel algorithm for Strassen's matrix multiplication in [5].

2.2 Matrix inversion

Similar as matrix multiplication operation, the matrix inversion is one of the most basic problems in mathematics and computer science. There exist multiple algorithms for finding the inverse of an invertible matrix: Gauss-Jordan elimination, LU decomposition, Newton's iterative method, Cayley-Hamilton method, eigendecomposition, Cholesky decomposition, reciprocal basis vectors method, blockwise inversion and others. The most simple method to inverse an invertible matrix is the Gauss-Jordan elimination. According to this method, the identity matrix is augmented to the right of given matrix and, after that, through application of elementary row operations, the reduced echelon form is found, the obtained left block being identity matrix and the right block being the inverse of the given invertible matrix. If the algorithm is unable to reduce the left block to identity matrix, then the initial matrix is not invertible. The disadvantage of this method is its $O(n^3)$ computational complexity, which is the same as for naive matrix multiplication algorithm. However, due to its simplicity, it can be considered as part of more complex algorithms, at least in the case when there are also another algorithm parts with complexity bigger than $O(n^3)$, since the entire algorithm complexity is not such much affected by this method.

In contrast to the Gauss-Jordan elimination, the blockwise inversion method [8, 11] is a divide and conquer algorithm, which allows to reduce the algorithm complexity, due to its relationship with matrix multiplication complexity. It was shown in [8] that the blockwise inversion method runs with the same time complexity as the matrix multiplication algorithm that is used internally. Since the best known

matrix multiplication complexity is $O(n^\omega)$, this means that the fastest known matrix inversion algorithm runs with same time complexity $O(n^\omega)$.

2.3 Determining the characteristic polynomial

The characteristic polynomial for a given matrix can be found using algorithms from [10, 15]. In [10] several ideas were combined to get a new Las Vegas randomized algorithm for computing the characteristic polynomial. The complexity of this randomized algorithm is $O(n^\omega)$. The Keller Gehrig's deterministic algorithm [15] works for all inputs and in the worst case has the complexity $O(n^\omega lg(n))$.

2.4 Resuming matrix polynomial

The problem of resuming the matrix polynomial is the following: For a given $n \times n$ matrix P and a polynomial $T(z) = \sum\limits_{k=0}^{m} a_k z^k$ with numeric coefficients, to compute the matrix $T(P) = \sum\limits_{k=0}^{m} a_k P^k$.

This problem can be solved by using the naive algorithm recursively computing the matrices $P^0 = I_n,\ P^1 = P^0 P,\ P^2 = P^1 P,\ \ldots\ P^m = P^{m-1} P$, performing m matrix multiplications, multiply each matrix with the corresponding coefficient a_k, $k \in \{1, 2, \ldots, m\}$ and sum by k. The complexity of such an algorithm is $O(mn^\omega)$.

The $r \times r$ scheme presented in [12] shows that we can use only $O(r)$ matrix multiplications, where $r = [\sqrt{m+1}]$. This means that we have to store the matrices $P^0, P^1, \ldots, P^r$ and reuse them along polynomial subdivision into a set of sub-polynomials of degree less than or equal to r. So, the complexity of the algorithm is $O(\max\{mn^2, rn^\omega\})$. On the conditions $m \approx n$ and $\omega <= 2.5$, the running time of this algorithm is $O(n^3)$.

For resuming matrix polynomials there exist also parallel algorithms. For example, Alonso, Boratto, Peinado, Ibáñez and Sastre in [4] evaluated matrix polynomials using several GPGPUs.

2.5 Determining the limiting matrix in a Markov chain

The main results concerned with determining the limiting matrix in a finite state space Markov chain can be found in [1, 6, 7, 14]. In our investigations we shall use the following algorithm for determining the limiting matrix from [7]:

Algorithm 1.

Input: *the transition matrix $P \in \mathcal{M}_n(\mathbb{R})$;*
Output: *the limit matrix $Q \in \mathcal{M}_n(\mathbb{R})$;*

1. *Set $\beta_0 = 1$ and $P^0 = I_n$;*

2. *For each* $k = \overline{1,n}$ *calculate:*

$$P^k = PP^{k-1},\ s_k = trP^k,\ \beta_k = -\frac{1}{k}\left(s_k + \sum_{j=1}^{k-1}\beta_j s_{k-j}\right);$$

3. *Divide the polynomial* $\Delta(z) = \sum_{k=0}^{n}\beta_k z^k$ *by polynomial* $(z-1)^{m(1)}$ *and obtain the quotient* $V(z)$ *using Horner scheme, where* $m(1)$ *is the multiplicity of root* $z_0 = 1$ *for* $\Delta(z)$*. Additionally preserve the coefficients* $\beta_k^*,\ k = \overline{0,N}$*, of the polynomial* $V'(z) = (z-1)V(z)$ *obtained at the last step;*

4. *Compute* $S^{(0)} = \beta_0^* I$ *and* $S^{(k)} = \beta_k^* I_n + PS^{(k-1)},\ k = \overline{1,N-1}$*;*

5. *Determine the matrix* $S = \sum_{k=0}^{N-1} S^{(k)}$ *and the value* $V(1)$*;*

6. *Find the limit matrix* $Q = -S/V(1)$*.*

Based on methods and calculation procedures from Sections 2.1 - 2.4 we can conclude that the complexity of Algorithm 1 is $O(n^{\omega+1})$.

3 A new approach and an algorithm to find the limiting matrix

In this section we propose an approach for determining the limiting matrix in a Markov chain that represents a modification of Algorithm 1. Based on such an approach we ground a new algorithm for determining the limiting matrix Q. The complexity of the algorithm is $O(n^3)$.

3.1 The main results and an algorithm to find the limiting matrix

Let $K_P(z)$ be the characteristic polynomial for the transition probability matrix P. Then, based on results from [7] and Algorithm 1, the following equalities

$$\Delta(z) = (-1)^n z^n K_P(z^{-1}) = (z-1)^{m(1)}V(z)$$

hold. Substituting z with z^{-1}, we obtain

$$(-1)^n z^{-n} K_P(z) = (z^{-1}-1)^{m(1)}V(z^{-1})$$

This implies the relation

$$V(z^{-1}) = \frac{K_P(z)}{(-z)^n(z^{-1}-1)^{m(1)}}. \tag{2}$$

Additionally, from step 4 of the Algorithm 1 we have

$$S^{(k)} = \sum_{j=0}^{k}\beta_{k-j}^* P^j,\ k = \overline{0,N-1}. \tag{3}$$

Taking into account that

$$V'(z) = (z-1)V(z),\ V(z) = \sum_{k=0}^{N-1} \gamma_k z^k \ and\ V'(z) = \sum_{k=0}^{N} \beta_k^* z^k$$

and dividing the polynomial $V'(z)$ by $z-1$ using Horner schema, we obtain

$$\gamma_k = \sum_{j=k+1}^{N} \beta_j^*,\ k = \overline{0, N-1}.$$

Since $\sum\limits_{j=0}^{N} \beta_j^* = V'(1) = 0$, we have

$$\gamma_k = -\sum_{j=0}^{k} \beta_j^*,\ k = \overline{0, N-1}. \tag{4}$$

So, using the equalities (3) and (4), we obtain the following formula

$$S = \sum_{k=0}^{N-1} S^{(k)} = -\sum_{k=0}^{N-1} \gamma_{N-1-k} P^k. \tag{5}$$

Now, using the equalities

$$V(z^{-1}) = \sum_{k=0}^{N-1} \gamma_k z^{-k} = z^{1-N} \sum_{k=0}^{N-1} \gamma_k z^{N-1-k} = z^{1-N} \sum_{k=0}^{N-1} \gamma_{N-1-k} z^k,$$

and applying the relation (2), we obtain

$$\sum_{k=0}^{N-1} \gamma_{N-1-k} z^k = z^{N-1} V(z^{-1}) = z^{n-m(1)} V(z^{-1}) =$$

$$= \frac{z^{n-m(1)} K_P(z)}{(-z)^n (z^{-1}-1)^{m(1)}} = (-1)^{N-1} \frac{K_P(z)}{(z-1)^{m(1)}}.$$

Taking into account that $K_P(z) \vdots (z-1)^{m(1)}$, we can represent $K_P(z)$ as follows $K_P(z) = (z-1)^{m(1)} T(z)$. This means that the previous relation becomes

$$\sum_{k=0}^{N-1} \gamma_{N-1-k} z^k = (-1)^{N-1} T(z). \tag{6}$$

Formula (6) represents an identity for two polynomials. Substituting z with P, we obtain an equality of two matrix polynomials

$$\sum_{k=0}^{N-1} \gamma_{N-1-k} P^k = (-1)^{N-1} T(P). \tag{7}$$

Formula (5) becomes

$$S = (-1)^N T(P), \tag{8}$$

i.e. the limit matrix is

$$Q = -\frac{S}{V(1)} = \frac{(-1)^{N-1}}{V(1)}\, T(P). \tag{9}$$

So, the limit matrix Q represents the product of the matrix $R = T(P)$ with a constant $c = \dfrac{(-1)^{N-1}}{V(1)}$. Because the limit matrix Q is a stochastic matrix, we can determine the constant c through normalization operation, i.e. c^{-1} represents the sum of the elements from an arbitrary row of the matrix R.

So, the results presented above ground the following algorithm for determining the limiting matrix Q:

Algorithm 2.

Input: *the transition matrix $P \in \mathcal{M}_n(\mathbb{R})$;*
Output: *the limit matrix $Q \in \mathcal{M}_n(\mathbb{R})$;*

1. *Determine the characteristic polynomial $K_P(z) = |P - zI_n|$;*
2. *Divide the polynomial $K_P(z)$ by $(z-1)^{m(1)}$, where $m(1)$ is the multiplicity of the eigenvalue $z_0 = 1$, obtaining the quotient $T(z)$;*
3. *Compute the matrix $R = T(P)$;*
4. *Determine the limit matrix Q by dividing the matrix R by the sum of the elements of an arbitrary its row.*

Remark 1. For a better performance of Algorithm 2 it is necessary to use the fast methods from Sections 2.1 - 2.4. So, to obtain algorithm with complexity $O(n^3)$ for determining the limiting matrix Q it is necessary to use the fast matrix multiplication algorithms from Section 2.1 with complexity $O(n^\omega)$, the algorithm from Section 2.3 with complexity $O(n^\omega)$ at the Step 1, the $O(n \cdot m(1))$ Horner schema at the Step 2 and the $O(n^{\max\{\omega+1/2,\ 3\}})$ algorithm from Section 2.4 at the Step 3.

3.2 Numerical examples

We present some numerical examples which illustrate the main details of the elaborated algorithm from Section 3.1.

Example 1. Consider the Markov process with the transition probability matrix $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. We can see that $P^{2t} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $P^{2t+1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\forall t \geq 0$, i.e. the Markov chain is 2-periodic.

So, in this case, the limit $\lim\limits_{t\to\infty} P^t$ does not exist, but there exists the limit matrix Q which can be found by using the Algorithm 2. If we apply this algorithm, then we have:

$$K_P(z) = |P - zI_n| = \begin{vmatrix} -z & 1 \\ 1 & -z \end{vmatrix} = (z-1)(z+1) \;\Rightarrow$$

$$T(z) = z + 1 \;\Rightarrow\; T(P) = I_2 + P = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \;\Rightarrow$$

$$Q = \frac{1}{1+1}\, T(P) = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}.$$

So, we obtained the same result as for numerical example 1 from [7]. Note that the Markov process for the considered process is not ergodic, however the rows of the limit matrix are the same and the vector of limiting probabilities is $\pi^* = (0.5,\ 0.5)$, i.e., regardless of the initial state of the Markov process, it will occupy the both states with the same probability 0.5 after a large number of transitions.

Example 2. Consider the Markov process with the transition probability matrix $P = \begin{pmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \end{pmatrix}$. We find the matrix Q applying Algorithm 2:

$$K_P(z) = |P - zI_n| = \begin{vmatrix} 0.5 - z & 0.5 \\ 0.4 & 0.6 - z \end{vmatrix} = (z-1)(z-0.1) \;\Rightarrow$$

$$T(z) = z - 0.1 \;\Rightarrow\; T(P) = -0.1 \cdot I_2 + P = \begin{pmatrix} 0.4 & 0.5 \\ 0.4 & 0.5 \end{pmatrix} \;\Rightarrow$$

$$Q = \frac{1}{0.4 + 0.5}\, T(P) = \begin{pmatrix} 4/9 & 5/9 \\ 4/9 & 5/9 \end{pmatrix}.$$

So, we obtained the same result as in numerical example 2 from [7]. The rows of the limit matrix are the same and the Markov process is ergodic, with the vector of limiting probabilities $\pi^* = (4/9,\ 5/9)$.

Example 3. We consider a non-ergodic Markov process with the transition probability matrix $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$. If for this example we apply Algorithm 2 for determining the limiting matix then we obtain:

$$K_P(z) = |P - zI_n| = \begin{vmatrix} 1 - z & 0 & 0 \\ 0 & 1 - z & 0 \\ 1/3 & 1/3 & 1/3 - z \end{vmatrix} = (z-1)^2(1/3 - z) \;\Rightarrow$$

$$T(z) = 1/3 - z \;\Rightarrow\; T(P) = (1/3)I_3 - P = \begin{pmatrix} -2/3 & 0 & 0 \\ 0 & -2/3 & 0 \\ -1/3 & -1/3 & 0 \end{pmatrix} \;\Rightarrow$$

$$Q = \frac{1}{-\frac{2}{3}+0+0} \cdot T(P) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 1/2 & 0 \end{pmatrix}.$$

So, we obtained the same result as in numerical example 3 from [7]. In this case all rows of the matrix Q are different. It is easy to observe that for the considered example there exists the limit $\lim\limits_{t\to\infty} P^t$ and this limit is Q.

4 A new approach and algorithm to find differential matrices

The aim of this section is to ground a new algorithm for determining the differential matrices in a finite state space Markov chain, the computational complexity of which is $O(n^{\omega+1})$. We show that such an algorithm can be grounded if the roots of characteristic polynomial of matrix P are known. The computational complexity of the algorithm is $O(n^{\omega+1})$.

4.1 The main results and an algorithm to find differential matrices

As we have already noted, based on results from [6], the decomposition (1) holds and the coefficients

$$\beta_k(y),\ y \in (\mathbb{C}\backslash\mathcal{D})\backslash\{1\},\ k = \overline{0, m(y)-1},$$

represent the differential matrices of the Markov process. Since the set $\mathbb{C}\backslash\mathcal{D}$ consists of all inverses of eigenvalues of the transition probability matrix P, the decomposition (1) is equivalent to the decomposition

$$P(t) = \sum_{y\in\sigma(P)} \sum_{k=0}^{m(y)-1} t^k y^t \overline{\beta}_k(y),\ \forall t \geq n-r,$$

where $K_P(z) = |P - zI|$ is the characteristic polynomial of the transition probability matrix P and $\sigma(P) = \{z \in \mathbb{C} \mid K_P(z) = 0\}$ is the spectrum of the transition probability matrix P. Note that $r = deg(K_P(z))$, $m(y)$ is the multiplicity of the root y of the polynomial $K_P(z)$ and $\overline{\beta}_k(y) = \beta_k(y^{-1})$, $\forall y \in \sigma(P)$. In particular, we have

$$P(t) = \sum_{y\in\sigma(P)} \sum_{k=0}^{m(y)-1} t^k y^t \overline{\beta}_k(y),\ t = \overline{n-r, n-1}.$$

On components, this relation can be written as follows

$$p_{i,j}(t) = \sum_{y\in\sigma(P)} \sum_{k=0}^{m(y)-1} t^k y^t \overline{\beta}_{i,j,k}(y),\ t = \overline{n-r, n-1},\ i,j = \overline{1,n}.$$

If we denote

$$\beta_t = (t^k y^t)_{(y,k)\in\sigma(P)\times\{0,\ldots,m(y)-1\}},\ \forall t \geq 0,\ B = (\beta_j)_{j=\overline{0,r-1}},\ 0^0 \equiv 1, \qquad (10)$$

and

$$p_{i,j} = (p_{i,j}(t))_{t=n-r}^{n-1},\ \overline{\beta}_{i,j} = (\overline{\beta}_{i,j,k}(y))_{(y,k)\in\sigma(P)\times\{0,\dots,m(y)-1\}},\ i,j=\overline{1,n},$$

then we obtain

$$p_{i,j} = \overline{\beta}_{i,j}B^T,\ i,j=\overline{1,n},$$

where B^T is the transpose of matrix B. Since the vectors $\beta_j,\ j=\overline{1,r}$, are linearly independent, the matrix B is invertible. As consequence,

$$\overline{\beta}_{i,j} = p_{i,j}(B^T)^{-1},\ i,j=\overline{1,n}.$$

Finally, if we consider $\triangle = (\overline{\beta}_{i,j})_{(i,j)\in\{1,\dots,n\}\times\{1,\dots,n\}}$ and $\Pi = (p_{i,j})_{(i,j)\in\{1,\dots,n\}\times\{1,\dots,n\}}$ then we obtain

$$\triangle = \Pi(B^T)^{-1}. \tag{11}$$

Since Π is an $n^2\times r$ matrix and B is an $r\times r$ matrix, then the result $\triangle$ is a $n^2\times r$ matrix, which contains all elements of r $n\times n$ coefficient matrices $\overline{\beta}_k(y),\ y\in\sigma(P)$. A row of $\triangle$ corresponding to a pair index (i,j) represents the elements from position (i,j) of the limit and differential matrices. This means that the columns of $\triangle$ represent the flattened version of the limit and differential matrices. In similar way we can refer this to the matrix Π whose columns represent the flattened version of the powers $P^t,\ t=\overline{n-r,n-1}$.

Now, if we analyze the computational complexity of calculating $\triangle$ according to (11), then we can notice the following. We can obtain the better performance for calculating (11) if we split the set of rows of Δ and Π into about $\left[\frac{n^2}{r}\right]$ parts of dimension r. After that the equation (11) can be regarded as about $\left[\frac{n^2}{r}\right]$ matrix multiplications of dimension $r\times r$. This means that we have the complexity $O(r^{\omega-1}n^2)$ for determining $\triangle$. Since, in the worst case, we have $r=n$, the complexity of $\triangle$ according to formula (11) is $O(n^{\omega+1})$.

Additionally to this complexity, we need to take into account the complexity of calculating the matrices Π, B, B^T and $(B^T)^{-1}$. For obtaining the matrix Π we need to have all matrices P^t, $t=\overline{n-r,n-1}$, i.e. in the worst case we need $O(n^{\omega+1})$ operations. If the eigenvalues of the matrix P are known, then the matrices B and B^T can be computed by using $O(n^2)$ elementary operations and the computation of $(B^T)^{-1}$ takes $O(n^\omega)$ arithmetic operations. Taking into account all above remarks, we can conclude that the entire algorithm complexity is $O(n^{\omega+1})$. So, having the eigenvalues of the transition probability matrix P, we can compute the matrix of limiting state probabilities and the differential matrices using $O(n^{\omega+1})$ arithmetic operations.

Thus, based on mentioned above results we can propose the following algorithm for determining the differential matrices.

Algorithm 3.

Input: *the transition probability matrix* $P \in \mathcal{M}_n(\mathbb{R})$*;*
Output: *the limit and differential matrices;*

1. *Calculate* $P^0 = I_n,\ P^1 = P^0P,\ P^2 = P^1P,\ \ldots\ P^{n-1} = P^{n-2}P$;
2. *Build the matrix* Π*, by collecting elements from the powers of matrix* P*;*
3. *Determine the matrix* B*, according to* (10)*;*
4. *Compute the matrix* $(B^T)^{-1}$*;*
5. *Determine matrix* $\triangle$*, according to* (11)*;*
6. *Build the limit and differential matrices, by collecting elements from* Δ*.*

4.2 Numerical Examples

We present numerical examples that illustrate the details of Algorithm 3 for periodic and aperiodic Markov chains.

Example 4. Let be given the Markov chain with transition probability matrix $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix}$ and consider the problem of determining the differential components. We apply Algorithm 3:

1) $P^0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix}$, $P^2 = \begin{pmatrix} 1 & 0 & 0 \\ 0.25 & 0.25 & 0.5 \\ 0.75 & 0 & 0.25 \end{pmatrix}$;

2-4) $K_P(z) = |P - zI_n| = \begin{vmatrix} 1-z & 0 & 0 \\ 0 & 0.5-z & 0.5 \\ 0.5 & 0 & 0.5-z \end{vmatrix} = (1-z)(0.5-z)^2 \ \Rightarrow$

$\Rightarrow r = deg(K_P(z)) = 3,\ \sigma(P) = \{1, 0.5\},\ m(1) = 1,\ m(0.5) = 2 \ \Rightarrow$

$\Rightarrow \beta_0 = (1,1,0),\ \beta_1 = (1,0.5,0.5),\ \beta_2 = (1,0.25,0.5) \Rightarrow B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0.5 & 0.5 \\ 1 & 0.25 & 0.5 \end{pmatrix} \Rightarrow$

$$\Rightarrow \quad (B^T)^{-1} = \begin{pmatrix} 1 & 0 & -2 \\ -4 & 4 & 6 \\ 4 & -4 & -4 \end{pmatrix}; \qquad \Pi = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -- & -- & -- \\ 0 & 0 & 0.25 \\ 1 & 0.5 & 0.25 \\ 0 & 0.5 & 0.5 \\ -- & -- & -- \\ 0 & 0.5 & 0.75 \\ 0 & 0 & 0 \\ 1 & 0.5 & 0.25 \end{pmatrix};$$

$$5)\ \triangle = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -- & -- & -- \\ 0 & 0 & 0.25 \\ 1 & 0.5 & 0.25 \\ 0 & 0.5 & 0.5 \\ -- & -- & -- \\ 0 & 0.5 & 0.75 \\ 0 & 0 & 0 \\ 1 & 0.5 & 0.25 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -2 \\ -4 & 4 & 6 \\ 4 & -4 & -4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -- & -- & -- \\ 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -- & -- & -- \\ 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix};$$

$$6)\ \overline{\beta}_0(1) = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},\ \overline{\beta}_0(0.5) = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix},\ \overline{\beta}_1(0.5) = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

So, the t-step transition probability matrix can be represented as follows:

$$P(t) = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \left(\frac{1}{2}\right)^t + \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} t \left(\frac{1}{2}\right)^t,\ \forall t \geq 0.$$

Example 5. Let be given the 2-periodic Markov process determined by the matrix of probability transition $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and consider the problem of determining the limit and differential components. If we apply Algorithm 3, then we obtain:

$$1)\ P^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$2\text{-}4)\ K_P(z) = |P - zI_n| = \begin{vmatrix} -z & 1 \\ 1 & -z \end{vmatrix} = z^2 - 1 = (z-1)(z+1)\ \Rightarrow$$

$$\Rightarrow r = deg(K_P(z)) = 2,\ \sigma(P) = \{1, -1\},\ m(1) = m(-1) = 1\ \Rightarrow$$

$$\Rightarrow \beta_0 = (1, 1),\ \beta_1 = (1, -1) \Rightarrow B = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \Rightarrow$$

$$\Rightarrow \quad (B^T)^{-1} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{pmatrix}; \qquad \Pi = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -- & -- \\ 0 & 1 \\ 1 & 0 \end{pmatrix};$$

$$5)\ \triangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -- & -- \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \\ -- & -- \\ 0.5 & -0.5 \\ 0.5 & 0.5 \end{pmatrix};$$

$$6)\ \overline{\beta}_0(1) = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix},\ \overline{\beta}_0(-1) = \begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{pmatrix}.$$

So, the t-step transition probability matrix can be represented as follows:

$$P(t) = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} + \begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{pmatrix} (-1)^t,\ \forall t \geq 0.$$

5 Conclusions

Based on results presented in this article, new algorithms for determining the limiting and differential matrices for finite state space Markov chains have been elaborated. The proposed algorithms represent a modification of the algorithms from [6,7] by using new calculation methods and procedures for fast matrix multiplication, matrix inversion, finding characteristic polynomial and resuming polynomial. The computational complexity of finding the limiting matrix is $O(n^3)$ and the complexity of calculating differential matrices is $O(n^{\omega+1})$, where n is the number of the states of the Markov chain and $O(n^{\omega})$ is the complexity of the used fast matrix multiplication algorithm.

References

[1] Lozovanu D., Pickl S. *Optimization of Stochastic Discrete Systems and Control on Complex Networks*. Springer, 2015.

[2] Le Gall F. *Powers of tensors and fast matrix multiplication*, Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, 2014, 296–303.

[3] Williams V. *Multiplying matrices in $O(n^{2.373})$ time*, Stanford University, 2014.

[4] Alonso P., Boratto M., Peinado J., Ibáñez J., Sastre J. *On the evaluation of matrix polynomials using several GPGPUs*, 2014.

[5] Ballard G., Demmel J., Holtz O., Lipshitz B., Schwartz O. *Communication-Optimal Parallel Algorithm for Strassen's Matrix Multiplication.* Cornell University Library, 2012.

[6] Lazari A. *Algorithms for determining the transient and differential matrices in finite Markov processes*, Bul. Acad. Ştiinţe Repub. Moldova, Mat., 2010, No. 2(63), 84–99.

[7] Lazari A., Lozovanu D. *An approach for determining the matrix of limiting state probabilities in discrete Markov processes*, Bul. Acad. Ştiinţe Repub. Moldova, Mat., 2010, No. 1(62), 77–91.

[8] Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. *Introduction to Algorithms*, 3rd ed., MIT Press, Cambridge, MA, 2009.

[9] D'Alberto P., Nicolau A. *Adaptive Winograd's Matrix Multiplications*, ACM Transactions on Mathematical Software, Vol. V, 2008.

[10] Pernet C., Storjohann A. *Faster algorithms for the characteristic polynomial*, David R. Cheriton School of Computer Science University of Waterloo, Ontario, Canada N2L 3G1, 2007.

[11] Bernstein D. *Matrix Mathematics*, Princeton University Press, 2005.

[12] WanZhen L., Roi B., Chandra S., Yihan S., Alexis T., Martin H. *Fast methods for resumming matrix polynomials and Chebyshev matrix polynomials*, Journal of Computational Physics, 194, 2004, 575–587.

[13] Puterman M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 2005.

[14] Coppersmith D., Winograd S. *Matrix multiplication via arithmetic progressions*, J. Symbolic Computation, 1990, **9(3)**, 251–280.

[15] Keller-Gehrig W. *Fast algorithms for the characteristic polynomial*, Theoretical Computer Science, 1985, **36**, 309–317.

[16] Strassen V. *Gaussian elimination is not optimal*, Numer. Math., 13, 1969, 354-356.

[17] Howard R. A. *Dynamic Programming and Markov Processes*, Wiley, 1960.

Alexandru Lazari, Dmitrii Lozovanu
Vladimir Andrunachievic Institute of Mathematics
and Computer Science
5 Academiei str., Chişinău, MD−2028, Moldova
E-mail: *alexan.lazari@gmail.com, lozovanu@math.md*

Received March 4, 2020