STUDIA UNIV. "BABEŞ–BOLYAI", MATHEMATICA, Volume **LI**, Number 1, March 2006

MONTE CARLO METHODS FOR SYSTEMS OF LINEAR EQUATIONS

NATALIA ROŞCA

Abstract. We study Monte Carlo methods for solving systems of linear equations. We propose three methods to generate the trajectories of the Markov chain associated to the system. We calculate the average complexity of generating the trajectories using these methods. From the complexity point of view, the proposed methods are better than other methods reported in the literature.

1. Introduction

We consider the system of linear algebraic equations:

$$Ax = b, \tag{1}$$

where $A = (a_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ is a given invertible matrix and $b \in \mathbb{R}^n$ is a given vector, $b = (b_1, \ldots, b_n)^t$. We are interested in estimating the solution $x = (x_1, \ldots, x_n)^t \in \mathbb{R}^n$ of system (1), using Monte Carlo methods. For this, we write the system in the following form:

$$x = Tx + c, \tag{2}$$

where $T = (t_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$, $c = (c_1, \ldots, c_n)^t \in \mathbb{R}^n$ and $I - T$ is an invertible matrix. The solution x admits the Neumann series representation:

$$x = c + Tc + T^2c + T^3c + \ldots \tag{3}$$

It is assumed that $\sum_{j=1}^n |t_{ij}| < 1,\ i = 1, \ldots, n$, which is a sufficient condition for the convergence of Neumann series to the solution.

Received by the editors: 09.11.2005.

2000 *Mathematics Subject Classification.* 65C05, 65C10.

Key words and phrases. Monte Carlo method, random number generation, linear systems, algorithm complexity.

The first Monte Carlo method for solving systems of linear equations was proposed by von Neumann and Ulam, and extended by Forsythe and Leibler [5]. For further details, see [6] and [9]. The method is efficient when we are interested in estimating one component of the solution.

2. **Monte Carlo methods to estimate the solution of the system**

There is also a Monte Carlo method for solving systems of linear equations, which allows to estimate the entire solution, by constructing unbiased estimators for the components of the solution.

To solve system (2), let $P = (p_{ij})_{i,j=1}^{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}$ be a matrix, whose elements satisfy the conditions:

1. $p_{ij} \geq 0$ such that $t_{ji} \neq 0 \Longrightarrow p_{ij} \neq 0$,
2. $\sum_{j=1}^{n} p_{ij} \leq 1$, $i = 1, \ldots, n$,
3. $p_{i,n+1} = 1 - \sum_{j=1}^{n} p_{ij}, \quad i = 1, \ldots, n$,
4. $p_{n+1,j} = 0$, $j < n+1$,
5. $p_{n+1,n+1} = 1$.

We also use the notation p_i for $p_{i,n+1}$. Furthermore, define the weights:

$$w_{ij} = \begin{cases} \frac{t_{ji}}{p_{ij}} & \text{if } p_{ij} \neq 0 \\ 0 & \text{if } p_{ij} = 0 \end{cases}, \quad i, j = 1, \ldots, n. \tag{4}$$

The matrix P describes a Markov chain with states $\{1, \ldots, n+1\}$, where $n+1$ is an absorbing state and p_{ij}, $i, j = 1, \ldots, n+1$ is the one step transition probability from state i to state j. Such a Markov chain is also called a *random walk*, as it is homogeneous and finite.

Denote by $\gamma = (i_0, i_1, \ldots, i_k, n+1)$ a trajectory that starts at the initial state $i_0 < n+1$ and passes successfully through the sequence of states $(i_1, \ldots, i_k)$, to finally get into the absorbing state $i_{k+1} = n+1$. Consider a vector $\alpha = (\alpha_1, \ldots, \alpha_n)$, where α_i, $i = 1, \ldots, n$ is the probability that a trajectory starts in state i, in other words,

$$P(i_0 = i) = \alpha_i, \qquad \alpha_i \geq 0, \ i = 1, \ldots, n, \qquad \sum_{i=1}^{n} \alpha_i = 1.$$

The probability to follow trajectory γ is $P(\gamma) = \alpha_{i_0} p_{i_0 i_1} \ldots p_{i_{k-1} i_k} p_{i_k}$.

Define the estimators $\theta_i,\ i = 1,\ldots,n$ and $\lambda_i,\ i = 1,\ldots,n$ on the space of trajectories as follows. For a trajectory $\gamma = (i_0, i_1, \ldots, i_k, n+1)$, the values of these estimators are defined as:

$$\theta_i(\gamma) = W_k(\gamma)\frac{\delta_{i_k i}}{p_{i_k}}, \qquad \lambda_i(\gamma) = \sum_{m=0}^{k} W_m(\gamma)\delta_{i_m i}, \qquad i = 1,\ldots,n,$$

where $W_m,\ m = 0,\ldots,k$ are random variables whose values are:

$$\begin{aligned} W_0(\gamma) &= \frac{c_{i_0}}{\alpha_{i_0}}, \\ W_m(\gamma) &= W_{m-1}(\gamma)w_{i_{m-1}i_m} \\ &= \frac{c_{i_0}}{\alpha_{i_0}} w_{i_0 i_1} w_{i_1 i_2} \ldots w_{i_{m-1}i_m}, \qquad m = 1,\ldots,k. \end{aligned}$$

The above values are taken with probability $P(\gamma)$ (δ_{ij} is the Kronecker symbol, i.e., $\delta_{ij} = 1$ if $i = j$ and 0 otherwise).

It can be proved that θ_i and λ_i are unbiased estimators of x_i, i.e.: $E(\theta_i) = E(\lambda_i) = x_i,\ i = 1,\ldots,n$.

The Monte Carlo Algorithm to estimate the solution of system (2) is the following:

Algorithm 1. *Monte Carlo Algorithm to estimate the solution x*

1. Input data: the matrix $T$and P, the vectors c and α, the integer n.
2. Generate N trajectories $\gamma_1,\ldots,\gamma_N$.
3. Compute the Monte Carlo estimate of the solution:

$$\hat{x} = \left[\frac{\theta_1(\gamma_1) + \ldots + \theta_1(\gamma_N)}{N}, \ldots, \frac{\theta_n(\gamma_1) + \ldots + \theta_n(\gamma_N)}{N}\right]^t. \tag{5}$$

or, the estimate:

$$\tilde{x} = \left[\frac{\lambda_1(\gamma_1) + \ldots + \lambda_1(\gamma_N)}{N}, \ldots, \frac{\lambda_n(\gamma_1) + \ldots + \lambda_n(\gamma_N)}{N}\right]^t. \tag{6}$$

3. Complexity of the Monte Carlo Algorithm

To compute the complexity of Algorithm 1, we assume that:

1. The costs of all arithmetical operations are equal, i.e., $CP(+) = CP(-) = CP(*) = CP(:) = 1$.

2. The cost of testing any of the inequalities $x < y$, $x > y$, $x \leq y$ or $x \geq y$ or the equality $x = y$ is d arithmetical operations.

3. The cost of generating one random number uniformly distributed on $[0, 1)$ is 3 arithmetical operations, as we use the linear congruential generator to generate random numbers.

Next, we analyse the complexity of each step of Algorithm 1.

3.1. **Complexity of generating the trajectories.** To start a trajectory, we sample from the following discrete distribution:

$$Y_\alpha : \begin{pmatrix} 1 & 2 & \dots & n \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \end{pmatrix}$$

described by the probability vector α, in order to get the initial state $i_0 \in \{1, 2, \dots, n\}$. Once the trajectory is in state $i_m = i$, $i \in \{1, 2, \dots, n\}$, we sample from the discrete distribution:

$$Y_i : \begin{pmatrix} 1 & 2 & \dots & n & n+1 \\ p_{i1} & p_{i2} & \dots & p_{in} & p_i \end{pmatrix}$$

described by the i-th line of matrix P, in order to determine the next state i_{m+1}. We repeat this procedure till absorbtion takes place.

The total number of steps before absorption is $\sum_{i=1}^{n} C_i$, where C_i denotes the number of times a trajectory visits the non-absorbing state i. Let $z = (z_1, \dots, z_n)$ be the solution of system $z = \bar{P}z + \alpha$, where $\bar{P}$ is the transpose of matrix $(p_{ij})_{i,j=1}^{n}$. The expectation of the random variable C_i is $E(C_i) = z_i$, $i = 1, \dots, n$ ([7]).

Denote by CP the (computational) complexity of generating a trajectory, defined as the number of arithmetical operations needed to generate it. For a trajectory, we sample from Y_α once, at the beginning of the generation process. The number of times we sample from Y_i is C_i. Let CP_{Y_α} and CP_{Y_i} denote the number of operations needed to generate a sample from Y_α and Y_i, respectively. It follows that the average complexity of generating a trajectory is given by:

$$E(CP) = E(CP_{Y_\alpha}) + \sum_{i=1}^{n} z_i E(CP_{Y_i}). \tag{7}$$

There are several methods for sampling from discrete distributions (see [3] or [4]). In [7] three such methods are used: the inversion, the acceptance-rejection and the alias method. The following results for the average complexities of generating a trajectory were obtained:

$$E(CP_{inv}) \leq (d+1)(\|z\|_1+1)n, \tag{8}$$

$$E(CP_{rej}) \leq (d+9)(\|z\|_1+1)n, \tag{9}$$

$$E(CP_{alias}) = (d+9)(\|z\|_1+1). \tag{10}$$

In the following, we use three methods: the decomposition, the economical and the table look-up method to sample from Y_α and Y_i, $i = 1, \ldots, n$. We calculate the average complexity of generating a trajectory and compare our results with the results (8)-(10).

3.2. **Generating trajectories using decomposition method.** We describe how we can generate a trajectory using the decomposition method to sample from Y_α and Y_i, $i = 1, \ldots, n$. Decomposition method is based on the following result (see [3]):

Theorem 1. *Any discrete distribution Y with m possible values can be written as the weighted sum of m distributions $\xi_1, \xi_2, \ldots, \xi_m$, each taking two possible values and having weight $1/m$.*

Next, we consider $Y = Y_\alpha$ and we describe how we can construct the distributions $\xi_1, \ldots, \xi_n$ (in this case $m = n$). For the sake of simplicity, we denote the values $1, 2, \ldots, n$ of distribution Y_α by $y_1, \ldots, y_n$, respectively. Thus, Y_α has the following form:

$$Y_\alpha = \begin{pmatrix} y_1 & y_2 & \ldots & y_n \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \end{pmatrix}.$$

We assume that $\alpha_1 \leq 1/n$ and $\alpha_2 \geq 1/n$, otherwise we look for two such probabilities in distribution Y_α and re-index them to 1 and 2, respectively. First, we decompose the distribution Y_α into the two-point distribution ξ_1 and the $n-1$ point distribution η_1 with weights $1/n$ and $(n-1)/n$ respectively, i.e.,

$$Y_\alpha = \frac{1}{n}\xi_1 + \frac{n-1}{n}\eta_1. \tag{11}$$

It can be shown that these distributions have the following form:

$$\xi_1 = \begin{pmatrix} y_1 & y_2 \\ q_1 & q_2 \end{pmatrix} \qquad \eta_1 = \begin{pmatrix} y_2 & y_3 & \dots & y_n \\ \alpha_2^{'} & \alpha_3^{'} & \dots & \alpha_n^{'} \end{pmatrix},$$

where $q_1 = n\alpha_1$ and $q_2 = 1 - n\alpha_1$ and

$$\alpha_2^{'} = \frac{n(\alpha_1 + \alpha_2) - 1}{n-1}, \qquad \alpha_j^{'} = \frac{n}{n-1}\alpha_j, \qquad j = 3, \dots, n.$$

Distribution η_1 is further decomposed into the two-point distribution ξ_2 with weight $1/(n-1)$ and the $(n-2)$-point distribution η_2 with weight $(n-2)/(n-1)$, i.e.,

$$\eta_1 = \frac{1}{n-1}\xi_2 + \frac{n-2}{n-1}\eta_2. \tag{12}$$

These distributions can be constructed as described above. Substituting (12) into (11), one obtains that the weight of ξ_2 in the decomposition of Y_α is $1/n$ as well. In a similar way distributions $\xi_3, \dots, \xi_n$ are constructed. Their weights are $1/n$. Thus, Y_α can be written as:

$$Y_\alpha = \frac{1}{n}\xi_1 + \frac{1}{n}\xi_2 + \ldots + \frac{1}{n}\xi_n, \tag{13}$$

where the distributions ξ_i, $i = 1, \dots, n$ have the following form :

$$\xi_i = \begin{pmatrix} y_{i1} & y_{i2} \\ q_{i1} & q_{i2} \end{pmatrix}$$

with $y_{i1}, y_{i2} \in \{y_1, \dots, y_n\}$ (i.e. $y_{i1}, y_{i2} \in \{1, \dots, n\}$) , $i = 1, \dots, n$.

Now, we give the procedure that generates a sample from Y_α.

Algorithm 2. *Decomposition Algorithm*

1. [*Set-up step*] Construct distributions $\xi_1, \dots, \xi_n$.

2. [*Selecting the distribution* ξ_i] Generate u uniformly distributed on $[0, 1)$ and set $i = [nu] + 1$ (i is uniformly distributed over $\{1, 2, \dots, n\}$).

3. [*Generating a sample from the distribution* ξ_i] Generate v uniformly distributed on $[0, 1)$, if $v < q_{i1}$ then return y_{i1}, otherwise return y_{i2}.

A similar algorithm can be written for sampling from $Y_i, i = 1, \dots, n$.

Concerning the complexity, we obtain the following main result.

Theorem 2. *The average complexity of generating a trajectory using the decomposition method is:*

$$E(CP_{dec}) = (d+8)(\|z\|_1 + 1). \tag{14}$$

Proof. Algorithm 2 requires the generation of 2 random numbers, 1 comparison, 1 multiplication and 1 addition. We omitted the integer part operation and the complexity of the set-up step. From formula (7), we obtain that the average complexity of generating a trajectory with decomposition method is given by:

$$\begin{aligned} E(CP_{dec}) &= E(CP_{Y_\alpha}) + \sum_{i=1}^{n} z_i E(CP_{Y_i}) = (d+8) + \sum_{i=1}^{n} z_i (d+8) \\ &= (d+8)(\|z\|_1 + 1). \end{aligned}$$

□

Corollary 3. *The average complexity of generating N trajectories using the decomposition method is equal to $(d+8)(\|z\|_1 + 1)N$.*

Remark. From (14) and (10), we obtain $E(CP_{dec}) < E(CP_{alias})$, which is an improvement from the complexity point of view.

3.3. **Generating trajectories using economical method.** We describe how to generate a trajectory using the economical method to sample from Y_α and Y_i, $i = 1, \ldots, n$. The economical method is a variant of the acceptance-rejection method, where no generated value is rejected. This will lead to a decrease in the complexity of generating a trajectory.

As previously illustrated, distribution Y_α can be written as:

$$Y_\alpha = \frac{1}{n}\xi_1 + \frac{1}{n}\xi_2 + \ldots + \frac{1}{n}\xi_n,$$

where the distributions ξ_i have the following form:

$$\xi_i = \begin{pmatrix} y_{i1} & y_{i2} \\ q_{i1} & q_{i2} \end{pmatrix}, \quad i = 1, \ldots, n.$$

Recall that y_{i1}, y_{i2}, $i = 1, \ldots, n$ are among the values of distribution Y_α.

We assume that $q_{i1} \leq q_{i2}, i = 1, \ldots, n$, otherwise q_{i1} and q_{i2} are inverted. In the economical method, degenerated distributions with $P(\xi_i = y_{i2}) = 1$ have to be transformed into $P(\xi_i = y_{i1}) = 1/2$, $P(\xi_i = y_{i2}) = 1/2$, where $y_{i1} = y_{i2}$.

The probabilities $q_{i1}, q_{i2}, i = 1, \ldots, n$ will be arranged into a vector $r = (r_1, \ldots, r_{2n})$, and correspondingly the values $y_{i1}, y_{i2}, i = 1, \ldots, n$ will be placed into a vector $v = (v_1, \ldots, v_{2n})$ as described below.

Algorithm 3. *Set-up Step for Economical Algorithm*

Initialize $j = 1$, $m = 1$, $i = 1$.
WHILE $i \leq n$ DO
IF $q_{i1} < q_{i2}$ THEN [case of a non-degenerated distribution ξ_i]
Set $r_j \leftarrow q_{i1}$, $r_{2n-j+1} \leftarrow q_{i2}$, $v_j \leftarrow y_{i1}$, $v_{2n-j+1} \leftarrow y_{i2}$,
Increase $j \leftarrow j + 1$.
ELSE [case of a degenerated distribution ξ_i]
Set $r_{n-m+1} \leftarrow q_{i1}$, $r_{n+m} \leftarrow q_{i2}$, $v_{n-m+1} \leftarrow y_{i1}$, $v_{n+m} \leftarrow y_{i2}$,
Increase $m \leftarrow m + 1$.
END IF
Increase $i \leftarrow i + 1$.
END WHILE
Save $n_1 \leftarrow j$, $n_2 \leftarrow n + m$.

Note that the probabilities $q_{i1} = q_{i2} = 1/2$ occupy the positions $r_s, s = n_1, n_1 + 1, \ldots, n_2 - 1$, which are central positions of vector r. The probabilities $q_{i1} < q_{i2}$ occupy symmetrical positions in vector r.
The procedure that generates a sample from Y_α is the following:

Algorithm 4. *Economical Algorithm*

Generate u_1 uniformly distributed on $[0, 1)$.
Compute $j \leftarrow [2nu_1] + 1$ (j is uniformly distributed over $\{1, \ldots, 2n\}$).
IF $j \geq n_1$ THEN RETURN v_j
ELSE Generate u_2 uniformly distributed on $[0, 1)$.
IF $\frac{u_2}{2} < r_j$ THEN RETURN v_j
ELSE RETURN v_{2n-j+1}

END IF

END IF.

A similar algorithm can be written for sampling from $Y_i, i = 1, \ldots, n$.

Concerning the complexity, we obtain the following main result.

Theorem 4. *The average complexity of generating a trajectory using the economical method is bounded by:*

$$E(C_{econ}) \leq (2d + 12)(\|z\|_1 + 1). \tag{15}$$

Proof. In Algorithm 4, the worst case scenario is the situation when both ELSE instructions are executed. In this case, we have 2 random numbers generated, 2 multiplications (we count the multiplication $2n$ only once), 2 additions, 1 substraction, 1 division and 2 comparisons. We omitted the integer part operation and the complexity of the set-up step. From formula (7), we get that the average complexity of generating a trajectory with the economical method is:

$$\begin{aligned} E(CP_{econ}) &= E(CP_{Y_\alpha}) + \sum_{i=1}^{n} z_i E(CP_{Y_i}) \\ &\leq (2d + 12) + \sum_{i=1}^{n} z_i (2d + 12) = (2d + 12)(\|z\|_1 + 1). \end{aligned}$$

□

Corollary 5. *The average complexity of generating N trajectories using the economical method is bounded by* $(2d + 12)(\|z\|_1 + 1)N$.

Remark. In the economical method the size n of matrix T is not included in the upper bound, whereas in the acceptance-rejection method, the complexity is proportional to n. As a consequence, the computing time is substantially reduced in the economical method, comparing to the acceptance-rejection method.

3.4. **Generating trajectories using table look-up method.** The table look-up method is a fast method to sample from Y_α, in the particular case when the probabilities α_i are rational numbers with common denominator M, i.e., $\alpha_i = m_i/M$, with $\alpha_i > 0,\ i = 1, \ldots, n$ and $\sum_{i=1}^{n} m_i = M$.

First, we construct a vector D of size M with m_1 entries 1, m_2 entries 2, ..., m_n entries n. Then, one element of this vector is picked up randomly (uniformly). Obviously, this element is a sample from distribution Y_α. The algorithm that generates a sample from Y_α is:

Algorithm 5. *Table Look-up Algorithm*

1. [*Set-up step*] Construct a vector $D = (D(1), \ldots, D(M))$, where m_i entries are i, $i = 1, \ldots, n$.
2. Generate u uniformly distributed on $[0, 1)$ and set $j = [Mu] + 1$ (j is uniformly distributed over $\{1, \ldots, M\}$).
3. Return $D(j)$.

If the transition probabilities are rational numbers with common denominator, a similar algorithm can be written to sample from $Y_i, i = 1, \ldots, n$.

Concerning the complexity, we get the following theorem.

Theorem 6. *The average complexity of generating a trajectory using the table look-up method is:*

$$E(CP_{tab}) = 5 + \sum_{i=1}^{n} 5z_i = 5(\|z\|_1 + 1). \tag{16}$$

Proof. Algorithm 5 requires the generation of 1 random number, 1 multiplication and 1 addition. We omitted the integer part operation and the complexity of the set-up step. From formula (7), we obtain the average complexity of generating a trajectory with the table look-up method:

$$\begin{aligned} E(CP_{tab}) &= E(CP_{Y_\alpha}) + \sum_{i=1}^{n} z_i E(CP_{Y_i}) = 5 + \sum_{i=1}^{n} 5z_i \\ &= 5(\|z\|_1 + 1). \end{aligned}$$

□

Corollary 7. *The average complexity of generating N trajectories using the table look-up method is equal to $5(\|z\|_1 + 1)N$.*

3.5. **Complexity of evaluating the estimators.** The average complexity of computing (5) is equal to $(2\|z\|_1+1)N+n$ ([7]). The average complexity of computing (6) is bounded by $\big(d(\|z\|_1-1)+3\big)\|z\|_1 N+n$.

3.6. **Total Complexity.** The average complexity of the Monte Carlo Algorithm 1 is the sum of the average complexity of generating N trajectories and the average complexity of evaluating the estimator.

The following table contains bounds for the average complexity of the Monte Carlo Algorithm 1, when the decomposition (DEC), the economical (ECON) and the table-look up (TAB) methods are used to generate the trajectories.

Method	Est.	Upper bound for the average complexity
DEC	θ_i	$(d+8)(\|z\|_1+1)N+(2\|z\|_1+1)N+n$
ECON	θ_i	$(2d+12)(\|z\|_1+1)N+(2\|z\|_1+1)N+n$
TAB	θ_i	$5(\|z\|_1+1)N+(2\|z\|_1+1)N+n$
DEC	λ_i	$(d+8)(\|z\|_1+1)N+\big(d(\|z\|_1-1)+3\big)\|z\|_1 N+n$
ECON	λ_i	$(2d+12)(\|z\|_1+1)N+\big(d(\|z\|_1-1)+3\big)\|z\|_1 N+n$
TAB	λ_i	$5(\|z\|_1+1)N+\big(d(\|z\|_1-1)+3\big)\|z\|_1 N+n$

Thus, the total average complexity of Algorithm 1 is $O(N)+n$.

4. **Concluding remarks**

1. We described how to generate the trajectories using decomposition method and calculated the average complexity of this procedure. We found this is less then the average complexity for the alias method, which is an improvement from the complexity point of view.

2. We used the economical method to generate the trajectories. This leads to a substantial decrease in the average complexity of generating the trajectories, comparing to the acceptance-rejection method.

3. We used the table look-up method to generate the trajectory, in the case when the initial and transition probabilities are rational numbers with a common denominator. This leads to the smallest complexity.

References

[1] Blaga, P., *Probability Theory and Mathematical Statistics*, II, Babes-Bolyai University Press, Cluj-Napoca, 1994 (In Romanian).

[2] Coman, Gh., *The Complexity of Algorithms*, Proceedings of the Conference in Differential Equations, Cluj-Napoca, November 21-23, 1985, 25-42.

[3] Deak, I., *Random Number Generators and Simulation*, Akademiai Kiado, Budapest, 1990.

[4] Devroye, L., *Non-Uniform Random Variate Generation*, Springer-Verlag, New York, 1986.

[5] Forsythe, G.E., and Leibler, R.A., *Matrix Inversion by a Monte Carlo Method*, Mathematical Tables and Other Aids to Computation 4(1950), 127-129.

[6] Hammersley, J.M., and Handscomb, D.C., *Monte Carlo Methods*, Methuen, London, 1964.

[7] Okten, G., *Solving Linear Equations by Monte Carlo Simulation*, SIAM Journal on Scientific Computing, in process of editing, 2005.

[8] Spanier, J., and Gelbard, E.M., *Monte Carlo Principles and Neutron Transport Problems*, Addison-Wesley, 1969.

[9] Wasow, W., *A Note on the Inversion of Matrices by Random Walks*, Mathematical Tables and Other Aids to Computation 6(1952), 78-81.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Str. Kogalniceanu 1, Cluj-Napoca, Romania
E-mail address: natalia@math.ubbcluj.ro