Libyan Journal of Medical and Applied Sciences LJMAS
Online ISSN: 3006-1113
Volume , Issue , June-December 2025, Page No: 48-56
Website: https://ljmas.com/index.php/journal/index

Analysis and Simulation of Lucas Distribution Using Markov Chain Monte Carlo Methods

Abd Anasir Salem Edabaa [1 *], Goksel Bilgici [2]
[1] Department of Statistics, Faculty of Science, Gharyan, University of Gharyan, Gharyan, Libya
[2] Department of Mathematics, Kastamonu University, Kastamonu, Turkya
[*]*Corresponding author: salemn136@gmail.com*

Received: June 10, 2025 Accepted: August 06, 2025 Published: August 10, 2025

Cite this article as: A, S, Edabaa., G, Bilgici. (2025) Analysis and Simulation of Lucas Distribution Using Markov Chain Monte Carlo Methods. Libyan Journal of Medical and Applied Sciences (LJMAS). 2025;3(3):48-56.

Abstract:
This study aims to investigate the properties of the Lucas distribution and apply the Markov Chain Monte Carlo (MCMC) method to simulate it and estimate its statistical characteristics. The work begins by defining the probability mass function of the Lucas distribution and establishing its mathematical connection to Birth–Death processes in continuous-time Markov chains. A generator matrix is constructed, and the corresponding discrete-time transition matrix is derived, demonstrating that the stationary distribution matches the target Lucas distribution. The Metropolis–Hasting's algorithm is then implemented in the R programming environment to generate samples from this distribution. The simulated results are analyzed and compared with theoretical values through graphical and statistical summaries. The findings reveal a high degree of agreement between the estimated and theoretical values over most of the range, with noticeable underrepresentation in the upper tail, suggesting the need for improved proposal mechanisms or longer chains. This research provides both a mathematical framework and an applied methodology for using MCMC to simulate uncommon discrete distributions and offers methodological enhancements to overcome the observed limitations.

Keywords: Markov Chain Monte Carlo "MCMC ", Lucas distribution, Lucas sequence.

التحليل والمحاكاة باستخدام سلاسل ماركوف مونت كارلو لتوزيع لوكاس

عبدالناصر سالم الضبع [1*]، غوكسل بيلجيجي.[2]
[1] قسم الإحصاء، كلية العلوم غريان، جامعة غريان، غريان، ليبيا
[2] قسم الرياضيات، جامعة كستامونو، كستامونو، تركيا

الملخص
يهدف هذا البحث إلى دراسة خصائص توزيع لوكاس وتطبيق أسلوب سلاسل ماركوف مونت كارلو (MCMC) لمحاكاة هذا التوزيع وتقدير خواصه الإحصائية. يبدأ العمل بتعريف دالة كتلة الاحتمال لتوزيع لوكاس وتوضيح العلاقة الرياضية التي تربطه بعمليات الولادة والوفاة (Birth–Death Processes) في سلاسل ماركوف الزمنية المستمرة. تم تطوير نموذج للمصفوفة المولدة (Generator Matrix) واستنتاج مصفوفة الانتقال المناظرة في الزمن المتقطع، مع برهان أن التوزيع الثابت للنظام يتطابق مع توزيع لوكاس المستهدف. بعد ذلك، طُبِّق خوارزمية ميتروبوليس–هاستينغز لتوليد عينات من هذا التوزيع باستخدام بيئة البرمجة R، وتم تحليل النتائج ومقارنتها بالقيم النظرية من خلال الرسوم البيانية والجداول الإحصائية. أظهرت النتائج تطابقًا مرتفعًا بين القيم التقديرية والنظرية في معظم المجال، مع ملاحظة قصور في تمثيل القيم الطرفية الكبيرة، ما يشير إلى الحاجة لتحسين آلية الاقتراح أو زيادة طول السلسلة. يوفر البحث إطارًا رياضيًا وتطبيقيًا يمكن تطويره لاستخدام MCMC في محاكاة توزيعات عددية غير شائعة، ويقترح تحسينات منهجية للتغلب على القيود الملحوظة.

الكلمات المفتاحية: سلسلة ماركوف مونت كارلو، توزيع لوكاس، متسلسلة لوكاس.

Introduction

The Lucas sequence, a close mathematical relative of the well-known Fibonacci sequence, has attracted the attention of researchers due to its intriguing algebraic properties, combinatorial interpretations, and potential applications in number theory, cryptography, and stochastic modeling. Defined by a simple linear recurrence relation yet exhibiting rich structural behavior, the Lucas sequence generates a discrete probability distribution when interpreted through a normalized probability mass function (pmf). This Lucas distribution can serve as a model for phenomena where recurrence-based growth patterns and integer-valued outcomes are present. Despite

its mathematical appeal, the statistical properties of the Lucas distribution have not been extensively explored in applied stochastic simulation frameworks.

Markov Chain Monte Carlo (MCMC) methods have become a fundamental computational tool for simulating and analyzing probability distributions, especially those for which direct sampling is analytically intractable. These methods rely on constructing a Markov chain whose stationary distribution matches a target distribution of interest. The chain is then simulated for a sufficiently long time, allowing samples from the stationary regime to approximate the true distribution. Among MCMC methods, the Metropolis–Hastings (MH) algorithm stands out for its generality and simplicity, enabling sampling from complex distributions using only the ability to evaluate the target density up to a proportionality constant.

In this study, we bridge the gap between the theoretical structure of the Lucas distribution and the practical machinery of MCMC simulation. We first present a formal derivation of the pmf associated with the Lucas sequence and explore its connection to Birth–Death processes in continuous-time Markov chains (CTMCs). By appropriately defining the birth and death rates, we construct a generator matrix that admits the Lucas distribution as its stationary distribution. This formulation provides a probabilistic interpretation of the Lucas sequence and establishes a pathway for simulation via discrete-time Markov chain transitions.

The proposed approach involves two complementary steps. First, we analytically connect the generator matrix of the CTMC to its discrete-time counterpart, ensuring that the stationary distribution is preserved. Second, we apply the Metropolis–Hastings algorithm to simulate the Lucas distribution, using proposal mechanisms tailored to the discrete integer support of the pmf. The use of MCMC is particularly advantageous here because the Lucas distribution, though simple to define recursively, can be cumbersome to normalize for large integer ranges, especially in applied contexts requiring probabilistic inference.

Simulation experiments are conducted in the R programming environment, producing synthetic samples from the Lucas distribution. We then compare the empirical frequencies obtained from the MCMC output to the exact theoretical pmf. This comparison is carried out using graphical overlays and numerical summaries to assess the accuracy of the approximation. Our findings indicate that MCMC can reproduce the Lucas distribution with high fidelity in most regions of the support, though underrepresentation of extreme values in the upper tail highlights limitations in the choice of proposal distribution and chain length.

This research contributes to the growing body of literature that extends MCMC applications beyond standard statistical models into more specialized and less commonly encountered discrete distributions. By combining the number-theoretic elegance of the Lucas sequence with the flexibility of the Metropolis–Hasting's algorithm, we offer both a theoretical framework and an empirical demonstration that may inspire further exploration in related areas, such as generalized recurrence-based distributions, cryptographic randomization schemes, and stochastic models in discrete event systems. The methodology and findings presented here also point toward practical improvements in proposal design, convergence diagnostics, and tail behavior handling, which are essential considerations for future work.

Literature Review

The Lucas sequence, introduced by Édouard Lucas in the late 19th century, is defined by the recurrence relation $\mathbf{L_n=L_{n-1}+L_{n-2}}$ with initial conditions $\mathbf{L_0=2}$ **and** $\mathbf{L_1=1}$. It is closely related to the Fibonacci sequence, sharing many of its algebraic and combinatorial properties but differing in initial values, which leads to distinct numerical patterns and growth rates. Over time, the Lucas sequence has been studied extensively within number theory (Vajda, 1989; Koshy, 2001), with applications spanning primality testing, cryptographic algorithms, and Diophantine equations. Despite its prominence in pure mathematics, its role in probability and stochastic processes has been less explored. The notion of constructing a **Lucas distribution** arises when the integer sequence values are normalized to form a discrete probability mass function (pmf), enabling probabilistic interpretation and simulation.

In applied probability, discrete distributions arising from number-theoretic sequences have received sporadic attention. Fibonacci-based probability models have appeared in queuing theory, random walk problems, and branching processes (Hilton & Pedersen, 1991; Horadam, 1965). However, analogous developments for Lucas numbers remain scarce. The few works that touch upon Lucas numbers in stochastic contexts have primarily examined their combinatorial generation functions or their use as weights in optimization algorithms, leaving a gap in literature concerning their empirical simulation and statistical characterization.

Markov Chain Monte Carlo (MCMC) methods, pioneered by Metropolis et al. (1953) and later generalized by Hastings (1970), have become indispensable for sampling from complex probability distributions. The **Metropolis–Hastings (MH)** algorithm, in particular, enables simulation from target distributions known only up to a proportionality constant, a feature that makes it especially useful for distributions like the Lucas distribution, where normalization constants may be computationally intensive for large supports. MCMC's core principle is to design a Markov chain with a stationary distribution matching the target distribution, then simulate the chain for enough iterations to approximate expectations under the target.

Over the past three decades, the theoretical foundations of MCMC have been thoroughly established (Tierney, 1994; Robert & Casella, 2004), and its applications have expanded into nearly all areas of computational statistics, including Bayesian inference (Gelman et al., 2013), statistical physics (Newman & Barkema, 1999), and computational biology (Lartillot & Philippe, 2006). Discrete distributions with non-standard forms have been effectively simulated using MCMC, particularly in cases where inverse transform sampling or direct rejection methods are inefficient. The challenge in discrete settings often lies in designing efficient proposal distributions that ensure rapid mixing and low autocorrelation in generated samples.

Continuous-time Markov chains (CTMCs) and their discrete-time counterparts (DTMCs) provide a natural mathematical framework for interpreting certain MCMC setups. Birth–death processes, a subclass of CTMCs, are particularly relevant for integer-valued distributions due to their tridiagonal generator matrices and well-studied stationary distributions (Norris, 1997). By defining state-dependent birth and death rates that reflect the structure of a target distribution's pmf, it is possible to construct a Markov process whose equilibrium coincides with the desired distribution. This approach has been widely used for modeling queueing systems (Gross & Harris, 1998) and population dynamics but has rarely been applied to number-theoretic distributions such as Lucas.

In recent computational literature, there has been a push to explore **special distributions** inspired by mathematical sequences—motivated partly by their potential applications in random number generation and cryptography. For example, Fibonacci and Pell distributions have been simulated to test pseudo-random number generators or to model discrete-event systems with structured state spaces. The Lucas distribution, with its unique growth pattern and divisibility properties, offers a similar opportunity for both theoretical analysis and simulation-based experimentation.

To date, there is no substantial body of work applying MCMC methods to the Lucas distribution. Existing studies on Lucas numbers focus on closed-form identities, congruences, and combinatorial interpretations, rather than probabilistic modeling or empirical simulation. This gap in the literature presents an opportunity to contribute new insights into both the statistical behavior of the Lucas distribution and the methodological adaptations needed to sample from it efficiently. By integrating MCMC with birth–death process modeling, the present research seeks to bridge theoretical number theory and applied stochastic simulation, offering a framework that can be extended to other recurrence-based discrete distributions.

Matrix Development for Lucas

In (1), Edabaa & Bilbici [essentially] determined the Lucas distribution to have a probability mass function.

$$\boldsymbol{l(x) = \frac{L_x}{2^{x+2}}} \qquad \boldsymbol{(1)}$$

$$\boldsymbol{for\ x = 1, 2, 3, \ldots where\ L_1 = 1\ , L_2 = 3\ and}$$
$$\boldsymbol{L_x = L_{x-1} + L_{x-2},} \qquad \boldsymbol{for\ x = 3, 4, 5\ \ldots . are\ the\ Lucas\ numbers}$$

Suppose we want to use MCMC to simulate observations from the Lucas distribution. First of all, we compute the ratio of successive Lucas probabilities:

$$\boldsymbol{\frac{l(x+1)}{l(x)} = \frac{L_{x+1}}{2^{x+3}} \Big/ \frac{L_x}{2^{x+2}} = \frac{L_{x+1}}{2L_x}} \qquad \boldsymbol{(2)}$$

Instead of using the Metropolis-Hastings algorithm method, we use a special choice from the Rosenbluth-Hastings method, to obtain the transition probabilities for given limiting ratios.

We enter to create a Markov chain of state-space 1,2,3, ... with the probability of state change. We will use one of the Markov chain processes called the Birth and Death process, which provides us with the required limit probability distribution.

From Eq (2), we have:

$$\boldsymbol{2L_x l(x+1) = L_{x+1} l(x)}$$

We use this equation as the equilibrium equation for a continuous-time Markov process. Our Birth and Death transition diagram looks like

$$\underset{(2)}{\overset{L_2}{\rightarrow}}\ \underset{(3)}{\overset{L_3}{\rightarrow}}\ \underset{(4)}{\overset{L_4}{\rightarrow}}\ \underset{(5)}{\overset{L_5}{\rightarrow}}\ \underset{(6)}{\overset{L_6}{\rightarrow}}\ \underset{(7)}{\overset{L_7}{\rightarrow}}\ (8)$$

$$\underset{2L_1}{\leftarrow} \quad \underset{2L_2}{\leftarrow} \quad \underset{2L_3}{\leftarrow} \quad \underset{2L_4}{\leftarrow} \quad \underset{2L_5}{\leftarrow} \quad \underset{2L_6}{\leftarrow}$$

Figure (1) shows Birth and Death transition diagram

In the process of birth and death, all restricted possibilities are obtained based on a basic probability (usually π_0 but, in our setting, there is no π_0, our base probability is π_1). According to the condition that the sum of all probabilities is equal to 1, we determine the base probability. In this case, if we called the limiting probabilities as:

$$\boldsymbol{\pi_1 = l_{(1)}, \pi_2 = l_{(2)}, \pi_3 = l_{(3)}}, \dots$$

We get

$$\boldsymbol{\pi_2 = l_{(2)} = \frac{L_2}{2L_1} l_1 = \frac{L_2}{2L_1} \pi_1},$$

$$\boldsymbol{\pi_3 = l_{(3)} = \frac{L_3}{2L_2} l_{(2)} = \frac{L_3 L_2}{2^2 L_1 L_2} \pi_1},$$

$$\boldsymbol{\pi_4 = l_{(4)} = \frac{L_4}{2L_3} l_{(3)} = \frac{L_4 L_3 L_2}{2^3 L_3 L_2 L_1} \pi_1},$$

$$\boldsymbol{\pi_5 = l_{(5)} = \frac{L_5}{2L_4} l_{(4)} = \frac{L_5 L_4 L_3 L_2}{2^4 L_4 L_3 L_2 L_1} \pi_1},$$

$$\textbf{So } \boldsymbol{\pi_2 = \frac{L_2}{2L_1}\pi_1, \pi_3 = \frac{L_3}{2^2 L_1}\pi_1, \pi_4 = \frac{L_4}{2^3 L_1}\pi_1, \pi_5 = \frac{L_5}{2^4 L_1}\pi_1}, \dots, \boldsymbol{and}$$

$$\boldsymbol{1 = \pi_1 + \pi_2 + \pi_3 + \cdots}$$

$$\boldsymbol{1 = \pi_1 \left[1 + \frac{L_2}{2L_1} + \frac{L_3}{2^2 L_1} + \frac{L_4}{2^3 L_1} + \frac{L_5}{2^4 L_1} + \cdots\right]}$$

$$\boldsymbol{1 = \pi_1 [1 + \frac{1}{L_1} \sum_{i=1}^{\infty} \frac{L_{i+1}}{2^i}]}$$

As long as the ratio of the pairwise ratios is maintained, we will obtain the same restricted probability.
So that, we divide each pair by the sum of the two components (leaving the same ratio), in order for the rates do not to become too arbitrarily large, a new state transition diagram is obtained with the same limiting probabilities.

$$\underset{}{\overset{\boldsymbol{\frac{L_2}{L_2+2L_1}}}{\rightarrow}} \quad \underset{}{\overset{\boldsymbol{\frac{L_3}{L_3+2L_2}}}{\rightarrow}} \quad \underset{}{\overset{\boldsymbol{\frac{L_4}{L_4+2L_3}}}{\rightarrow}} \quad \underset{}{\overset{\boldsymbol{\frac{L_5}{L_5+2L_4}}}{\rightarrow}} \quad \boldsymbol{\frac{L_6}{L_6+2L_5}}$$

$$\rightarrow \quad \boldsymbol{(2)} \quad \underset{}{\overset{\leftarrow}{}} \quad \boldsymbol{(3)} \quad \underset{}{\overset{\leftarrow}{}} \quad \boldsymbol{(4)} \quad \underset{}{\overset{\leftarrow}{}} \quad \boldsymbol{(5)} \quad \underset{}{\overset{\leftarrow}{}} \quad \boldsymbol{(6)} \quad \boldsymbol{(7)}$$

$$\leftarrow \quad \boldsymbol{\frac{2L_1}{L_2+2L_1}} \quad \boldsymbol{\frac{2L_2}{L_3+2L_2}} \quad \boldsymbol{\frac{2L_3}{L_4+2L_3}} \quad \boldsymbol{\frac{2L_4}{L_5+2L_4}} \quad \boldsymbol{\frac{2L_5}{L_6+2L_5}}$$

Figure (2) shows a new state transition diagram with the same limiting probabilities

The next corresponding generator matrix for states 1,2... where the rates pairs appear at off-diagonal positions are:

$$G = \begin{bmatrix}
a & \frac{L_2}{L_2+2L_1} & 0 & 0 & 0 & 0 & 0 \dots \\
\frac{2L_1}{L_2+2L_1} & b & \frac{L_3}{L_3+2L_2} & 0 & 0 & 0 & 0 \dots \\
0 & \frac{2L_2}{L_3+2L_2} & c & \frac{L_4}{L_4+2L_3} & 0 & 0 & 0 \dots \\
0 & 0 & \frac{2L^3}{L_4+2L_3} & d & \frac{L_5}{L_5+2L_4} & 0 & 0 \dots \\
0 & 0 & 0 & \frac{2L_4}{L_5+2L_4} & e & \frac{L_6}{L_6+2L_5} & 0 \dots \\
. & . & . & . & . & . & . \dots \\
. & . & . & . & . & . & . \dots \\
. & . & . & . & . & . & . \dots \\
. & . & . & . & . & . & . \dots \\
. & . & . & . & . & . & . \dots
\end{bmatrix}$$

Matrix 1. shows the next corresponding generator matrix for states 1,2...

where a, b, c, d, e are negative values chosen that satisfy the conditions of a rate matrix of a continuous-time Markov process (CTMP) so that each row sum to 0.

We define $\vec{\pi}$ represents the limiting row vector (Lucas probabilities), and $\vec{0}$ be the row vector of zero, then results for Markov processes give $\overrightarrow{\mathbf{0}} = \vec{\boldsymbol{\pi}}\boldsymbol{G}$.

We want to convert our matrix to a discrete-time Markov chain. So, we want to adjust in Q. Since some values of the non-negative could be greater than 0.5, and two be the same row, then the sum of the row to be greater than 1. So, assume that X where Y still has a rate matrix with the same limiting probability vector. Then $\overrightarrow{\mathbf{0}} = \vec{\boldsymbol{\pi}}\,\boldsymbol{G}^*$.

New we add $\vec{\pi}$ to both sides to get $\vec{\boldsymbol{\pi}} = \vec{\boldsymbol{\pi}} + \vec{\boldsymbol{\pi}}\boldsymbol{G}^* = \vec{\boldsymbol{\pi}}(\boldsymbol{I} + \boldsymbol{G}^*)$.

Define $\boldsymbol{P} = \boldsymbol{I} + \boldsymbol{G}^*$, so P satisfies $P = \vec{\boldsymbol{\pi}}\,\boldsymbol{P}$, and the sum of the rows of P is 1 and the entries are non-negative. Then, P is a discrete-time probability transition matrix. Now that we have converted our setting to a discrete-time Markov chain, we can specify the precise Markov transition matrix that we will use.

Here $\boldsymbol{P} = \boldsymbol{I} + \boldsymbol{G}^*$

$$P = \begin{bmatrix}
1+0.5a & \frac{0.5L_2}{L_2+2L_1} & 0 & 0 & 0 & 0 & 0 & \cdots \\
\frac{L_1}{L_2+2L_1} & 1+0.5b & \frac{0.5L_3}{L_3+2L_2} & 0 & 0 & 0 & 0 & \cdots \\
0 & \frac{L_2}{L_3+2L_2} & 1+0.5c & \frac{0.5L_4}{L_4+2L_3} & 0 & 0 & 0 & \cdots \\
0 & 0 & \frac{L_3}{L_4+2L_3} & 1+0.5d & \frac{0.5L_5}{L_5+2L_4} & 0 & 0 & \cdots \\
0 & 0 & 0 & \frac{L_4}{L_5+2L_4} & 1+0.5e & \frac{0.5L_6}{L_6+2L_5} & 0 & \cdots \\
0 & 0 & 0 & 0 & \frac{L_5}{L_6+2L_5} & 1+0.5f & \frac{0.5L_7}{L_7+2L_6} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots
\end{bmatrix}$$

Matrix 2. shows a discrete-time probability transition matrix

$$\approx \begin{bmatrix}
0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0.2 & 0.6 & 0.2 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0.3 & 0.4667 & 0.2333 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0.2667 & 0.5133 & 0.22 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0.28 & 0.495 & 0.225 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0.275 & 0.50193 & 0.22307 & \cdots, \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots
\end{bmatrix}$$

Matrix 3. shows the probability transition matrix for a vector of Lucas probabilities.

The state is 1, 2, 3, 4, … this is a probability transition matrix, which has its limiting probability vector of Lucas probabilities. We can use it to generate random values from the Lucas distribution.

Then we apply the system $\vec{\pi} = \vec{\pi}P$ and compare the estimated probability from transition matrix P and the true probabilities.

Since the Lucas and the Fibonacci numbers have the same recursive relationship, we can use the same method to get the probability transition matrix of Fibonacci probability.

R program for Lucas and results

If we are in state x at step i, then we move to state $\boldsymbol{x-1, x\ or\ x+1}$ at step i+1, the probability is

$\frac{L_{x-2}}{L_{x-1}+2L_{x-2}}$, $\mathbf{1} - \frac{L_{x-2}}{L_{x-1}+2L_{x-2}} - \frac{0.5L_x}{L_x+2L_{x-1}}$, $\frac{0.5L_x}{L_x+2L_{x-1}}$, respectively, for x = 2,3…

where we define $L_0 = 0$. That is easy to do in R.

We use R to generate 1000000 uniform (0,1) values. As we simply obtain x [i + 1] from x[i] by subtracting 1 from x[i] if our uniform (0,1) value is less than $\frac{L_{x[i]-2}}{L_{x[i]-1}+2L_{x[i]-2}}$, by adding 1 if our uniform (0,1) value lies in

$\left(1 - \frac{0.5L_{x[i]}}{L_{x[i]}+2L_{x[i]-1}}, \mathbf{1}\right)$, we don't do any things otherwise.

We can note that there is no chance of decreasing if the state is =2. We can implement the R command to implement a Markov chain based on the transition matrix P. To show typical output, we print out the first 100 (of the 1000000) dependent values of a single run of the commands.

[1] 2 2 3 3 3 4 4 3 3 3 3 4 5 5 5 5 5 6 7 7 7 7 7 7 8 8 8 9 9 9 10
[32] 10 10 11 11 10 10 10 10 10 10 11 11 12 12 12 11 11 11 11 11 11 11 11 11 11
[57] 11 12 11 12 12 12 11 12 12 12 11 11 12 12 13 14 14 15 15 15 14 14 15 15 14
[82] 14 14 15 14 14 13 13 13 12 12 12 13 13 12 13 13 13 14 14

We can see the dependence of the state of each step on the state of the previous step. We can use the MCMC values that we have created to estimate the probability distribution function. Although these values are not independent, the ergodic theory [3] states that the long-run proportion of steps at each state matches the limiting distribution.

We show three graphs together; to check whether our method has the required limit probability. The first graph provides us with the true Lucas probability. The second plot gives us the relative frequencies of random values generated with our MCMC method. The third plot gives us random values generated from the true probabilities using the inverse function method and taking advantage of R's \sample" command. We can't almost differentiate between the three plots. Thus, MCMC did a very well job. plots appear in the appendix.

The estimated probabilities from MCMC simulation and the true probabilities of { 2,3,4,5,6,7,8,9} are:

Table 1. The estimated probabilities from MCMC simulation and the true probabilities

x	2	3	4	5	6	7	8	9
Est.Prob.	0.128	0.191	0.127	0.1100	0.08661	0.06906	0.05462	0.04370
TrueProb.	0.125	0.1875	0.125	0.10937	0.08593	0.07031	0.05664	0.04589

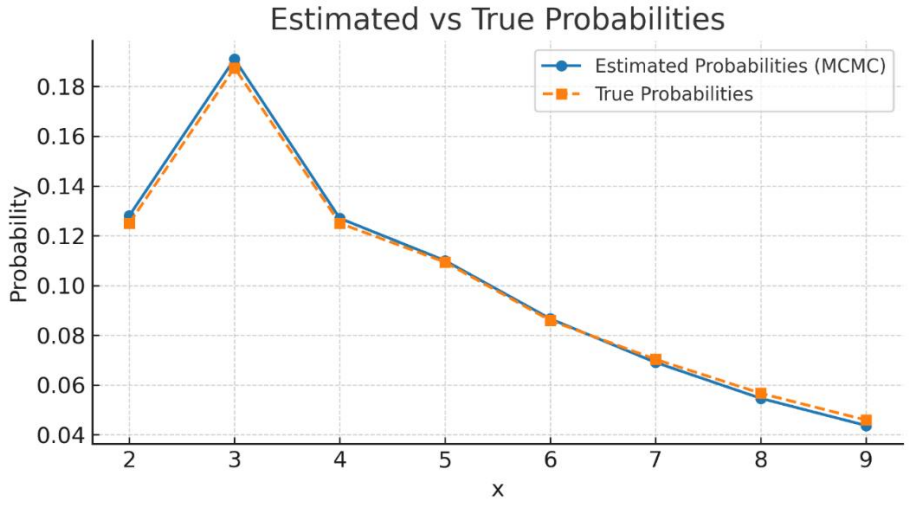

Figure 1. shows the estimated probabilities from MCMC simulation and the true probabilities

The estimated probabilities from the inverse function simulation and the true probabilities of {2, 3, 4, 5, 6, 7, 8, 9} are:

Table 2. The estimated probabilities from the inverse function simulation and the true probabilities

x	2	3	4	5	6	7	8	9
Est.Prob.	0.1248	0.1879	0.1253	0.10955	0.08634	0.06970	0.05626	0.04546
TrueProb.	0.1250	0.1875	0.1250	0.10937	0.08593	0.07031	0.05664	0.04589

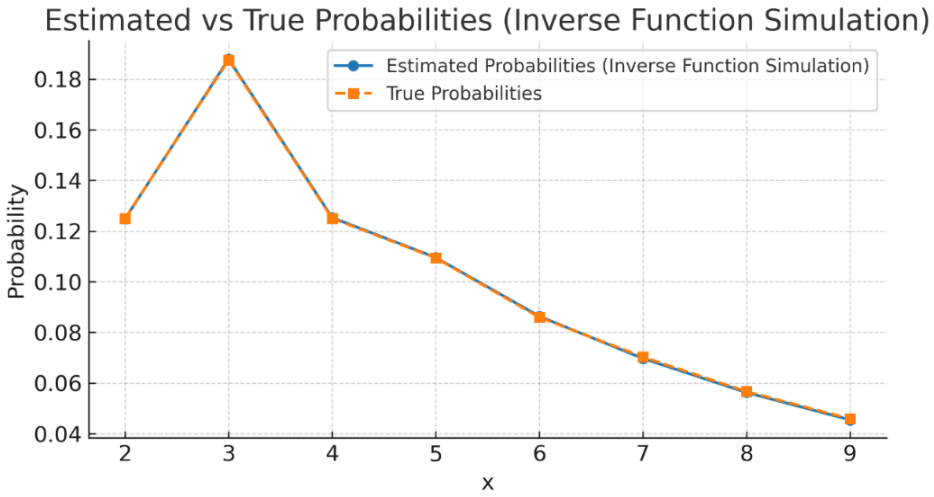

Figure 2. shows the estimated probabilities from the inverse function simulation and the true probabilities

Again, we can see how close the simulated probabilities are to the true probabilities.
Also of interest is the right tail of the two simulations. Of the 1000000 values generated by the two methods; we get counts of the highest 4 values of x as follows. In this case, we have

Table 3. The highest 4 values from MCMC and the inverse function

MCMC (x,count)	(57,12)	(58,13)	(59,4)	(60,2)
InvFunctionCount	(58,3)	(59,3)	(61,2)	(63,2)

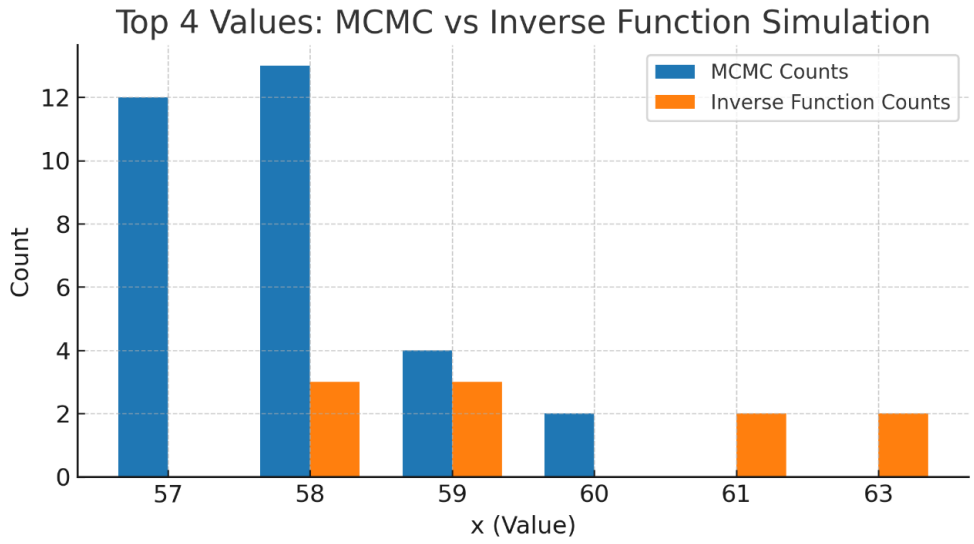

Figure 3. shows the highest 4 values from MCMC and the inverse function

Both methods of simulation are somewhat less than satisfactory here. The MCMC method fails to pick up any value over 60 when one expects that some value should be there. (If MCMC did find large values, every smaller value must also be present). The inverse function method, which is an exact method, has large gaps in the upper values, which suggest the pattern at the tail would be highly unlike to reappear for some time and could not be reliably used to represent the upper tail.

Results, Comments, and Comparison of Methods

Our MCMC method is a special case of the Rosenbluth-Hasting MCMC method. We should know only the ratio of probabilities of interest because our MCMC method of generating random values requires us that. However, the programming required for our method needs only uniform random numbers generated. It is a simple computation.

The inverse function method is the standard Technique to simulate many random variables. However, it requires us to know or to be able to compute the form of the density function or mass function. That is known for the Lucas distribution programming to become more difficult than our MCMC method in most packages, except for R.

1- Since the state of each step depends on the state of the previous step. We will use the \ sample " command in the R package as a random subset, which means that for a large population and a small sample size, we obtain values selected behave essentially as being independent.

In our "sample" command, we show approximately 20 independent values from the Lucas distribution. the output sample(x)[1:20] is

6 7 3 2 9 9 10 2 7 6 3 3 10 7 6 5 3 9 8 13

These values indeed look like independent values from the Lucas distribution.

2- Although our results are for the Lucas distribution, the MCMC method works well if the ratio of the probabilities is easily obtained from discrete distributions.

3- one might wonder why simulation of the distribution when we know the probability mass function, where we can easily obtain moments and other measures. We will answer this question that if we want to test a new strategy, then we need to have actual values from the distribution, not just the probabilities, so we can perform a simulation to see the effect of the strategy.

4- The MCMC results appear more reasonable because the estimated probabilities based on MCMC will not have an estimated probability of zero with there being no zero neighboring estimates on both sides. That is an advantage of MCMC over other simulation methods.

Conclusion:

1- From our MCMC method, a probability transition matrix was obtained, with a probability vector of the Lucas probabilities.

2- Random values were generated from a Lucas distribution that behaved essentially independently from our probability transition matrix. In the "sample" command, the resulting sample (x) [1, 20] is 6 7 3 2 9 9 10 2 7 6 3 3 10 7 6 5 3 9 8 13

3- The programming required for our method requires generating only regular random numbers. This is a simple calculation.

4- The probabilities estimated by our method were very close to the true probabilities

5- The MCMC method is somewhat unsatisfactory, failing to capture any value above 60 when a value is expected to exist.

6- If the MCMC method finds large values, every smaller value must also exist.

Recommendations:

After constructing a Markov transition matrix and applying it to the Lucas probability distribution using Markov chain Monte Carlo (MCMC), the researcher recommends the following:

1. Construct a new Markov transition matrix and apply it to a discrete probability distribution.
2. Identify the causes of the gaps in the upper values of the MCMC method and the inverse function method.
3. Study the potential errors in both methods and ways to address them.

References

1. BİLGİCİ, G. (2021), On waiting time distribution of runs in a Fibonacci and Lucas sequences. *Avrupa Bilim ve Teknoloji Dergisi*, (25), 774-781.
2. Hlynka, M. (2017). MCMC and the Fibonacci distribution. *Communications in Statistics-Simulation and Computation*, *46*(5), 3375-3382.
3. Medhi, J. (1994). Stochastic processes. New Age International.
4. Herbei, R., & Berliner, L. M. (2014). Estimating ocean circulation: an MCMC approach with approximated likelihoods via the Bernoulli factory. *Journal of the American Statistical Association*, *109*(507), 944-954.

5. Papamarkou, T., Mira, A., & Girolami, M. (2014). Zero variance differential geometric Markov chain Monte Carlo algorithms. *Bayesian Analysis*, *9*(1), 97-128.
6. Sadegh, M., & Vrugt, J. A. (2014). Approximate bayesian computation using markov chain monte carlo simulation: Dream (abc). *Water Resources Research*, *50*(8), 6767-6787.
7. Xiang, F., & Neal, P. (2014). Efficient MCMC for temporal epidemics via parameter reduction. *Computational Statistics & Data Analysis*, *80*, 240-250.
8. Beck, J. L., & Zuev, K. M. (2013). Asymptotically independent Markov sampling: a new Markov chain Monte Carlo scheme for Bayesian inference. *International Journal for Uncertainty Quantification*, *3*(5).
9. Albert,J. and Rizzo, M. (2012). R by Example Chapter 13 Springier.

Appendix: plots