Random Number Generation using the Standard Map

Ahmad Alfani Handoyo - 13520023
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520023@std.stei.itb.ac.id

***Abstract*—This paper aims to develop an implementation of the standard map for random number generation. The design of the random number generator algorithm is based on the values generated by the standard map equations, with careful considerations to avoid predictable patterns and ensure consistency. The resulting algorithm displays high randomness without discernible periodicity or repeating patterns, and the distribution of the random numbers approximates a uniform distribution with occasional deviations.**

***Keywords*—Random number generation; Standard map; Chaotic systems**

I. Introduction

Random number generation is crucial for uses and applications such as simulation, modeling, statistical analysis, cryptography, gaming, machine learning, and finance. In some of these scenarios, there is a desire for a truly random number generator, meaning that each value is equally likely to occur without a perceivable pattern or periodicity. Achieving true randomness is no small feat and is challenging, as it often requires specialized hardware or sophisticated algorithms. In the field of computer science itself, researchers extensively seek random number generators that are efficient, easy to implement, and unpredictable.

One approach to random number generation involves incorporating chaotic systems. The study of chaotic systems dates back to the late 1880s when Henri Poincaré attempted to solve the long-standing three-body problem in classical mechanics. Poincaré observed that orbits can exhibit chaotic behavior, characterized by their inability to converge or diverge towards a specific point. Building upon Poincaré's groundbreaking work, other scientists such as Edward Lorenz and Mitchell Feigenbaum delved deeper into the subfields of chaotic systems, including dynamical systems, bifurcation theory, and chaotic maps.

Given the apparent ubiquity of randomness in the universe, chaos theory offers a suitable model for describing complex systems characterized by seemingly unpredictable behavior. Chaotic systems can be harnessed for random number generation, offering several advantages over traditional pseudo-random number generators. They generate values iteratively based on their previous states, making them highly sensitive to changes in initial conditions. Consequently, even the slightest perturbation in the initial conditions can lead to substantial differences in future system states, rendering chaotic systems inherently unpredictable.

One prototypical example of a chaotic system that has been extensively studied since its introduction in the 1960s is the standard map. The standard map is a mathematical model that has found diverse applications and has been the subject of significant research. Its true chaotic nature has been well-explored, and it has been observed that various systems can be effectively reduced to the standard map. Examples include the whisker map, which represents the chaotic layer around the separatrix of a nonlinear resonance induced by a monochromatic force, as studied by Chirikov. Additionally, the standard map has been used to analyze particle dynamics in accelerators, microwave ionization of Rydberg atoms, and electron magnetotransport in resonant tunneling diodes, among others.

Furthermore, the standard map has been widely discussed in the context of cryptography. Several papers have explored its potential for cryptographic purposes, highlighting its value as a tool for secure information transmission. For instance, Lian et al. proposed a symmetric block cipher using the standard map, analyzing the sensitivity of its parameters. Building upon this work, K.W. Wong et al. proposed an image encryption scheme that employed the block cipher suggested by Lian et al., albeit with a reduced number of diffusion rounds. In another study, Hamdi et al. proposed a selective compression-encryption scheme for images, utilizing the standard map in the substitution and permutation processes.

The primary objective of this paper is to develop an implementation of the standard map for random number generation. The paper aims to analyze the chaotic nature of the standard map and assess its suitability for producing random numbers. The ultimate goal is to construct an efficient random number generator based on the standard map.

By exploring the potential of chaotic systems, specifically the standard map, for random number generation, this research aims to contribute to the development of robust and reliable methods for producing random numbers. The analysis of the standard map's chaotic behavior and its impact on the generated numbers will provide valuable insights into the efficacy of using chaotic systems as a source of randomness. The findings of this study can have implications for various fields in cryptography, where secure and unpredictable random numbers are essential for ensuring the integrity of sensitive information.

II. Basic Theory

A. Random Number

A random number is a numerical value that cannot be predicted in terms of its value or occurrence. These numbers can take various forms including integers, real numbers, or binary strings. Examples of a series of random numbers include integers [285, 1241, 599, 2349, 7], real numbers [0.297, 0.6726, 0.0302, 0.4401], and the binary string ‘11010100110’.

Random numbers play a crucial role in the field of cryptography. They are used in various cryptographic algorithms and systems. In public-key cryptography algorithms, random numbers are utilized for generating key parameter values. In the Diffie-Hellman Key Exchange, for example, a party may generate their private key number randomly. Similarly, the ElGamal encryption algorithm needs a random integer as part of its encryption process. Several other applications of random numbers are generating initialization vectors (IV) for a block cipher, random strings in challenge-and-response mechanisms used for authentication, and client session keys for the Secure Sockets Layer (SSL) protocol.

B. Random Number Generation

Random number generation is the creation of number sequences that lack predictable patterns. A program that does this task is called a random number generator (RNG). RNGs can generate numbers using two main methods, either exhibiting a pseudo-random or truly random behavior.

Pseudo-random number generators (PRNGs) imitate the properties of random numbers but are not genuinely random. They rely on an initial value called the seed to produce their sequence of numbers, which means their results can be reproduced. The speed and efficiency of PRNGs make them significant in various uses such as simulations and cryptography. However, cryptographic applications necessitate PRNG output that is unpredictable and not easily replicated. Therefore, more complex algorithms are required to ensure the output is not easily predictable. The output of a PRNG must possess good statistical properties, which necessitates careful mathematical analysis to guarantee its suitability. In some instances, common PRNGs may exhibit artifacts that fail statistical tests, such as having shorter-than-expected periods, lacking uniform distribution, showcasing correlation, poor dimensional distribution, and displaying differences in value occurrence distances.

Unlike their counterparts, true random number generators (TRNGs) produce random numbers through the utilization of physical processes instead of algorithms. These TRNGs are based on phenomena such as thermal noise and quantum effects, which are theoretically unpredictable. Hardware TRNGs utilize transducers, amplifiers, and analog-to-digital converters to convert physical phenomena into digital random numbers. However, it is important to note that hardware TRNGs often have limited output rates. Therefore, they are frequently employed to generate seeds for faster pseudorandom number generators.

Hardware random number generators find widespread use in the field of cryptography, especially in generating cryptographic keys for secure data transmission in protocols like Transport Layer Security (TLS). The inherent randomness of hardware TRNGs enhances the security of cryptographic systems by providing an unpredictable foundation for key generation. This unpredictability is crucial in cryptographic protocols to prevent adversaries from easily deducing the encryption keys and compromising the security of transmitted data.

C. Chaos Theory

Chaos theory examines the behavior of systems governed by deterministic laws that exhibit apparent randomness or unpredictability. By exploring deterministic chaos, we can reconcile the seemingly contradictory concepts of randomness and determinism. Previously, randomness was perceived as apparent and attributed to a lack of knowledge regarding the multiple underlying causes. However, the success of scientific predictions based on deterministic principles since Newton’s era has proven that a seemingly complex and random system can be predictable.

More recent investigations have revealed that even systems governed by well-understood physical laws can exhibit unpredictable behavior. Surprisingly, systems with simplicity can give rise to seemingly chaotic outcomes. What makes these systems behave unpredictably is their sensitivity to initial conditions and how these conditions are set in motion. This characteristic is a common thread among such systems.

The butterfly effect serves as a prominent illustration of the intrinsic unpredictability found within a simple model of heat convection. It demonstrates that a minute perturbation, such as the flapping of a butterfly’s wings, can have far-reaching consequences and lead to significant variations in the outcome of the system. Sensitivity to initial conditions in systems like this becomes the center of discussion in chaos theory.

Within classical mechanics, dynamical systems manifest motion on attractors. Traditionally, there are three types of attractors called single points, closed loops, and tori. However, in the 1960s, Stephen Smale made a groundbreaking discovery by uncovering a new class of attractors called strange attractors that exhibit chaotic dynamics. These strange attractors possess intricate structures at all scales, which has contributed to the development of fractal geometry and paved the way for advancements in computer graphics.

The advent of strange attractors and the subsequent exploration of their complexity have given rise to a deeper understanding of chaos theory. These intricate patterns, discovered within the dynamics of chaotic systems, have revealed a mesmerizing interplay between order and disorder in fractal structures. The examination of fractals and their self-similar properties has unlocked new avenues for studying complex phenomena in various fields, including physics, biology, economics, and even art.

D. *Standard Map*

The standard map, also referred to as the Chirikov standard map or Taylor-Greene-Chirikov map in older literature, is a chaotic map that preserves area and maps a square with a side length of 2π. Its origins can be traced back to Boris Chirikov who developed it to describe the dynamics of magnetic field lines. The standard map has been extensively studied as a paradigm for chaotic Hamiltonian dynamics and has played a major role in classical and quantum chaos research.

The standard map describes the Poincaré section of the motion of the kicked rotator. The kicked rotator consists of a frictionless stick that can rotate in a plane around one of its tips, while also being periodically kicked on the other tip. Mathematically, the standard map is defined by the equations

$$I_{n+1} = I_n + k\,sin(\theta_n)$$

$$\theta_{n+1} = \theta_n + I_{n+1}$$

or by incorporating the previous iteration of *I*, θ_{n+1} can also be formulated as

$$\theta_{n+1} = \theta_n + I_n + k\,sin(\theta_n)$$

where *I* and θ are computed with respect to mod 2π and *k* is a positive constant. In the context of the system, *I* and θ represent the angular momentum and angular position of the stick respectively while *k* represents the strength of the nonlinear kick and ultimately the system's degree of chaos.

Below, examples of the Poincaré sections of the standard map with different values of *k* are illustrated, plotting the values of *I* and θ on a two-dimensional plane. These visual representations showcase the different behaviors and structures that emerge as *k* varies.

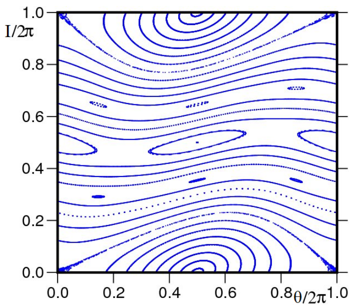


Fig. 2.1. Poincaré section for standard map of *k* = 0.5

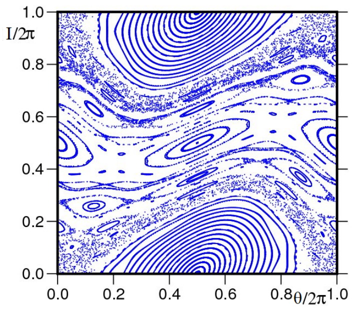


Fig. 2.2. Poincaré section for standard map of *k* = 0.971635

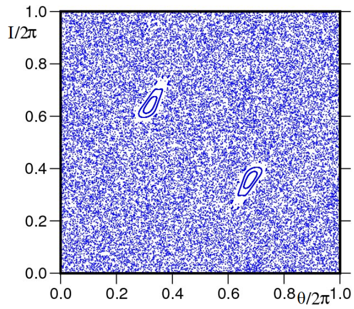


Fig. 2.3. Poincaré section for standard map of *k* = 5

Larger values of *k* increase the nonlinearity of the map and thus allow for more chaotic behaviors when coupled with appropriate initial conditions. Various authors have bounded the value of *k* at which global chaos occurs. Hermann provides a lower bound of $k > \frac{1}{34}$, while MacKay and Percival suggest an upper bound of $k < \frac{63}{64}$. However, the most tested and accurate estimation to date comes from Greene's method, which approximates $k \approx 0.971635$.

III. Design and Algorithm

A. *Design*

The design of the random number generator algorithm is based on the values generated by the standard map, specifically the values *I* and θ. In each iteration, the values of *I* and θ are calculated using the standard map equations and are bounded by the modulo of 2π. This ensures that the values stay within a defined range for the periodicity of the sine function in the formula of *I*.

To avoid getting stuck in a predictable pattern, a check is implemented to ensure that the initial values of *I* and θ cannot both be 0. If both values are 0, subsequent iterations will always yield 0 as the sine function will always return 0, resulting in a non-random sequence made up entirely of zeros. Therefore, the check prevents this scenario from occurring.

Another important check is performed on the parameter *k*, which represents the crucial constant determining the chaotic behavior in the standard map equations. It is necessary to ensure that *k* is a positive value. If *k* is not greater than 0, the behavior of the random number generator may become unpredictable or undesirable.

Furthermore, since the iterations of the standard map calculation will always produce values of *I* and θ bounded between 0 and 2π, it is necessary to check that the initial values of *I* and θalso fall within this range. This ensures consistency and prevents unexpected results.

As mentioned before, the random numbers generated by the algorithm are derived from the values of *I* and θ in each iteration of the standard map. Two numbers are obtained from each iteration by applying the mod 1 operation to the values of *I* and θ. These values are then multiplied by 10 to the power of *exp* to shift the decimal point. Finally, the resulting numbers are converted to integers, retaining only the integer part. This process allows the user to specify the number of digits the random numbers should have. By providing the *exp* parameter, users can control the precision and scale of the generated random numbers. This flexibility allows for customized usage of the random number generator in various applications. As an example of these operations, when $exp = 5$, the value 1.52486421 becomes the integer value of $(1.52486421 \; mod \; 1) \times 10^{4}$ which is 52486.

B. Algorithm

With the aforementioned design considerations, a random number generator algorithm is constructed and implemented in the form of a function. The function receives parameters i, o, n, k, and exp which respectively represent the initial I and θ value, how many random numbers are generated, the k constant, and the maximum amount of digits a random number could have. The parameter k has a default value of 0.971635 which is the significant point of global chaos in the standard map, while the parameter exp has a default value of 4 so that the generated numbers by default have a maximum of 4 digits. The function returns an array of random numbers with length n. The algorithm follows these steps:

1. Do initial checks on function parameters.
   a. Check that $k > 0$. If $k \leq 0$, value k is invalid and exit.
   b. Check that i and o are not both equal to 0. If both $i - 0$ and $o = 0$, the combination of i and o are invalid and exit.
   c. Check that $0 \leq i < 2\pi$. If $i < 0$ or $i \geq 2\pi$, value i is invalid and exit.
   d. Lastly, check that $0 \leq o < 2\pi$. If $o < 0$ or $o \geq 2\pi$, value o is invalid and exit.
2. Initialize an empty array called $numbers$.
3. Loop $\lceil \frac{n}{2} \rceil$ times.
   a. Calculate $i = (i + k \, sin(o)) \; mod \; 2\pi$.
   b. Then, calculate $o = (o + i) \; mod \; 2\pi$ with i using the value calculated in the previous step.
   c. Calculate $num1$ which is the integer value of $(i \; mod \; 1) \times 10^{exp}$.
   d. Then, calculate $num2$ which is the integer value of $(o \; mod \; 1) \times 10^{exp}$.
   e. Append $num1$ and $num2$ to the array $numbers$.
4. Return the array $numbers$, taking the first n elements. This means that for an odd value of n, the last generated number is discarded.

As an example, if the user inputs $i = 0.2$, $o = 0.1$, $n = 5$ while leaving the default values $k = 0.971635$ and $exp = 4$, the algorithm will return a sequence of random numbers 2970, 3970, 6726, 696, and 5248.

IV. Analysis

With the random number generator algorithm proposed, a few aspects are analyzed which are deemed relevant and important to assess the quality of the random numbers. These aspects are the periodicity and distribution of the random numbers generated by the algorithm.

A. Periodicity

To analyze the periodicity of the random number generator, a line plot is used with the x-axis representing the index of the number and y-axis representing the number generated for a given index. There are two variable conditions which are by varying the initial value of I and θ, and also by varying the k constant.

A sequence of 100 random three-digit numbers is generated and examined to see whether any repeating patterns or cycles emerge. By observing the generated sequence over a large number of iterations, it can be determined whether the system exhibits any periodic behavior. The absence of discernible patterns or repetitions indicates a higher degree of randomness. Ideally, the random number generator should not have any period or repetition.

1) Initial value of I and θ

The values $k = 0.971635$ and $exp = 3$ are used to analyze the effect of different values of I and θ when generating 100 random numbers.

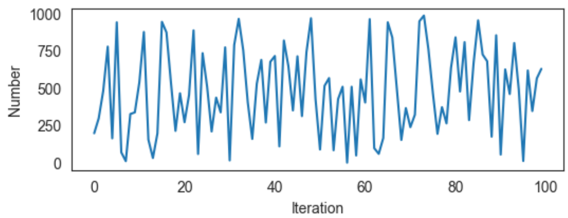


Fig. 4.1. Periodicity of initial $I = 0.1$ and $\theta = 0.1$

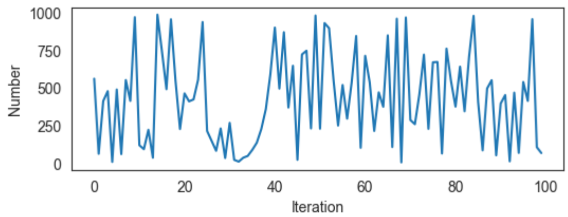


Fig. 4.2. Periodicity of initial $I = 0.1$ and $\theta = 0.5$

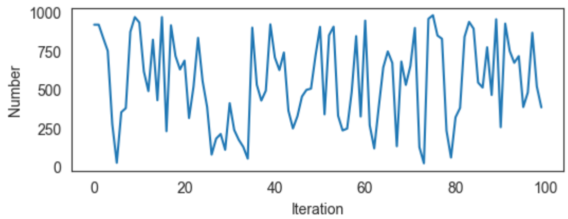


Fig. 4.3. Periodicity of initial $I = 0.1$ and $\theta = 1$

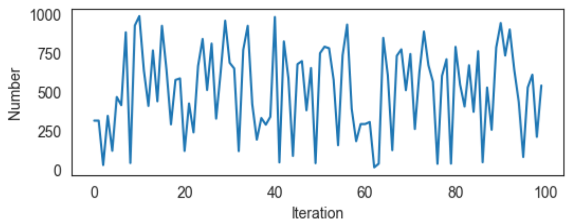


Fig. 4.4. Periodicity of initial $I = 0.5$ and $\theta = 1$

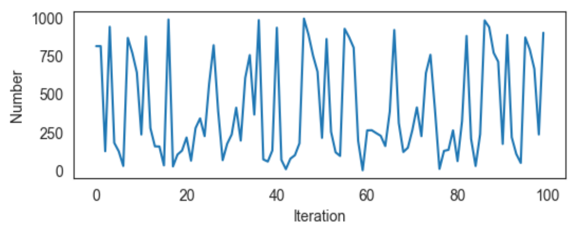


Fig. 4.5. Periodicity of initial $I = 1$ and $\theta = 1$

The graphs generated for each set of initial values I and θ appear completely distinct from one another. This observation suggests that even slight changes in the initial conditions result in significantly different sequences of random numbers. It indicates the sensitivity of the standard map to initial conditions and the chaotic nature of the system.

None of the graphs exhibit any discernible periodicity or repeating patterns. This absence of regularity implies that the random numbers generated by the algorithm do not follow a predictable cyclic behavior. The system appears to explore the entire phase space chaotically, providing a high degree of randomness. Regardless of the specific initial conditions tested, all the resulting sequences of random numbers exhibit similar levels of chaotic behavior.

2) k constant

The values $I = 0.1$, $\theta = 0.5$, and $exp = 3$ are used to analyze the effect of different values of k when generating 100 random numbers. The value $k = 0.971635$ is a crucial value to be analyzed as that is the point when the standard map descends into global chaos.

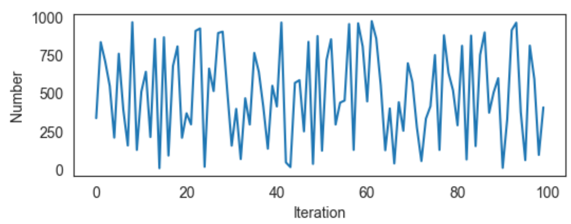


Fig. 4.6. Periodicity of $k = 0.5$

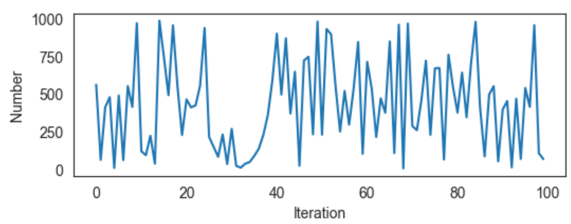


Fig. 4.7. Periodicity of $k = 0.971635$

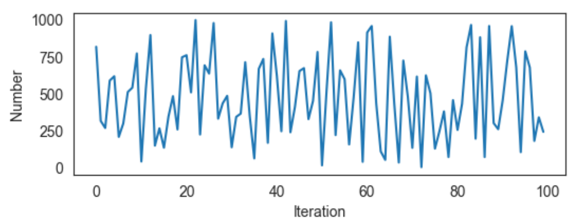


Fig. 4.8. Periodicity of $k = 1.5$

Similar to the previous analysis, the graphs generated for each value of k appear distinct from one another. Each graph represents a unique sequence of random numbers, highlighting the sensitivity of the standard map to changes in the parameter k.

The analysis demonstrates that varying the value of the k constant has a significant impact on the behavior of the algorithm. Even when k is not set to the commonly associated value of global chaos of 0.971635, the algorithm continues to exhibit chaotic behavior for generating random numbers.

B. *Distribution*

To assess the overall distribution of the random number generator, a histogram is plotted with respect to a sample of 1000 random numbers generated by the algorithm. Similar to the previous periodicity subsection, there are two variable conditions which are by varying the initial value of I and θ, and also by varying the k constant.

From the histogram, it is hoped that the observed distribution is close to uniform. A uniform distribution means that the algorithm can produce random numbers fairly across the range of possible values without any significant clustering. This means that a number to another number will have the same chance of being generated.

1) Initial value of I and θ

The values $k = 0.971635$ and $exp = 3$ are used to analyze the effect of different values of I and θ when generating 1000 random numbers.

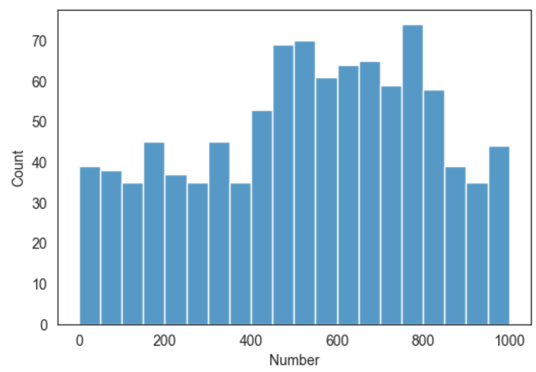


Fig. 4.9. Distribution of initial $I = 0.1$ and $\theta = 0.1$

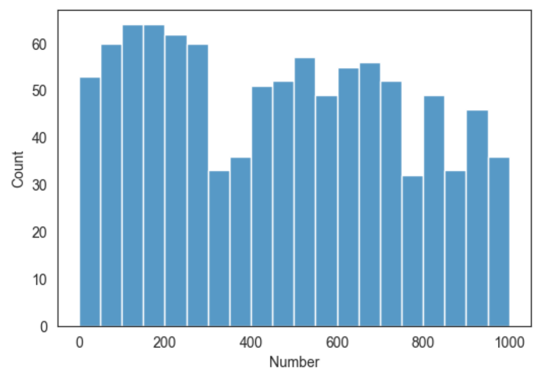


Fig. 4.10. Distribution of initial $I = 0.1$ and $\theta = 0.5$

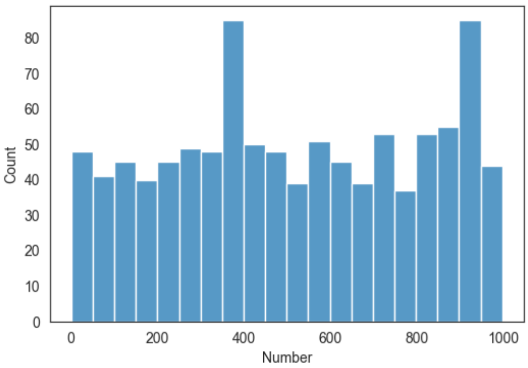


Fig. 4.11. Distribution of initial $I = 0.1$ and $\theta = 1$

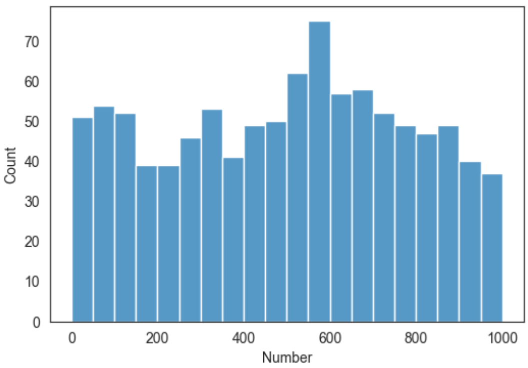


Fig. 4.12. Distribution of initial $I = 0.5$ and $\theta = 1$

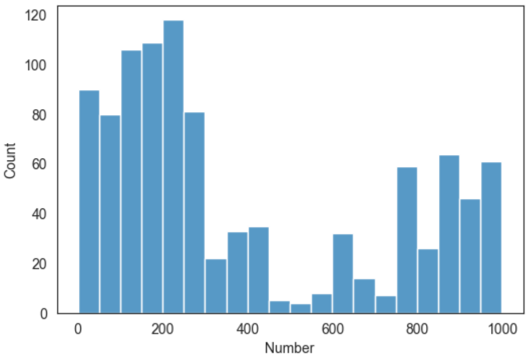


Fig. 4.13. Distribution of initial $I = 1$ and $\theta = 1$

For the test case with $I = 0.1$ and $\theta = 0.1$, the histogram exhibits a somewhat bell-curve shape in the higher numbers. This indicates a concentration of random numbers towards the middle range of values. While the distribution deviates slightly from perfect uniformity, it still maintains a relatively balanced spread of numbers throughout the range.

In the case of $I = 0.1$ and $\theta = 0.5$, the histogram reveals a high count in the lower and middle numbers, indicating a clustering of random numbers in these regions. However, there are also several ranges with low counts in the middle and upper ranges, resulting in an uneven distribution. This non-uniform behavior indicates the presence of patterns in the generated random numbers, compromising the desired ideal distribution.

When $I = 0.1$ and $\theta = 1$, the histogram appears mostly uniform, indicating a closer approximation to the ideal distribution. However, there are a few regions with exceptionally high counts, specifically around the values of 400 and 900. These localized peaks in frequency indicate some deviations from perfect uniformity, but overall, the distribution displays a satisfactory level of randomness.

For $I = 0.5$ and $\theta = 1$, the histogram exhibits a bell-curve shape in the higher numbers. This suggests a concentration of random numbers towards the upper end of the range, indicating a clustering in that region. While not adhering strictly to the ideal uniform distribution, the overall spread of numbers appears reasonably balanced.

The case of $I = 1$ and $\theta = 1$ presents a severely non-uniform distribution. The histogram shows the highest counts in the lower numbers, with very few counts in the higher numbers. The middle numbers exhibit a strikingly low count, creating a distinct valley-like pattern. This behavior strongly deviates from the ideal uniform distribution, indicating unseen patterns in the generated random numbers.

Varying initial values of I and θ demonstrates the algorithm's ability to approach a uniform distribution of random numbers in some cases, while not so in others. The ideal uniform distribution is achieved in some instances, but clustering or non-uniform patterns can also emerge.

2) k constant

As discovered before, the initial values of I and θ to be 0.1 and 1 respectively produce the closest to a uniform distribution. Thus, those values are used to analyze the effect of varying values of the k constant with $exp = 3$ when generating 1000 random numbers.

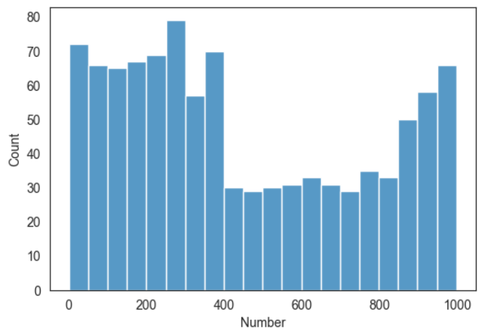


Fig. 4.14. Distribution of $k = 0.5$

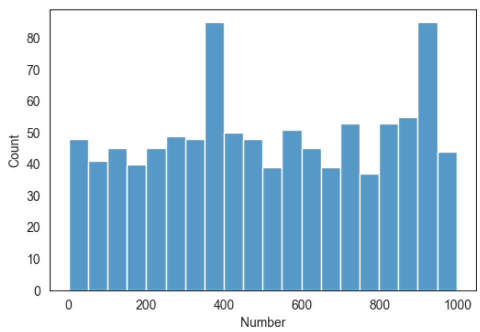


Fig. 4.15. Distribution of $k = 0.971635$

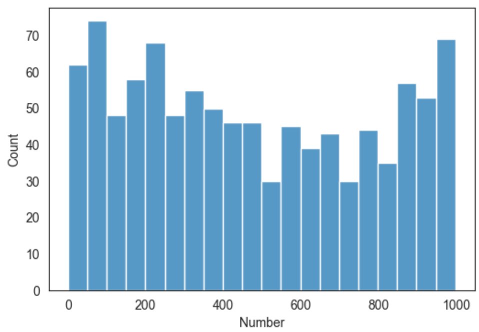


Fig. 4.16. Distribution of $k = 1.5$

The test case with $k = 0.5$ reveals a distribution pattern in the histogram where the lower and middle numbers have high counts, followed by a valley of low counts from the middle to the higher range. While this distribution deviates from perfect uniformity, it still demonstrates a degree of randomness with a slight bias towards the lower and middle ranges.

When $k = 0.971635$, the histogram reveals a mostly uniform distribution, which closely approximates the ideal. The majority of the bins in the histogram display similar counts, indicating an even spread of random numbers across the entire range as previously seen.

Lastly, at $k = 1.5$, the histogram appears kind of uniform but exhibits a valley-like shape with lower counts in the middle-high regions. This indicates a slight clustering towards the lower and higher ends of the range, resulting in a dip in frequency in the middle region. While the distribution deviates slightly from the ideal uniform distribution, it still maintains a relatively balanced spread of numbers.

By varying the values of the k constant, the algorithm's capability to approximate a uniform distribution under specific conditions is observed. Notably, when $k = 0.971635$, the distribution is predominantly uniform with minimal deviations. However, other values of k show non-uniform distributions. It is uncertain to say whether the result of the uniformity is because of the value of k by itself or because of the combination of it with values I and θ as well.

V. Conclusion

The successful implementation of the standard map as a random number generator demonstrates its potential for producing high-quality random numbers. The algorithm's sensitivity to initial conditions, combined with the absence of discernible periodicity or repeating patterns, ensures a high degree of randomness, making it a promising candidate for random number generation. Regardless of the specific initial conditions tested, the algorithm consistently exhibits similar levels of chaotic behavior, further enhancing its reliability.

A random number generator should generate a number with the same probability as another number, or in other words have a uniform distribution of generated values. However, the algorithm fails to consistently generate uniformly distributed values with varying parameters k, I, and θ, necessitating further investigation into the complex interaction between k, I, and θ. Future research can delve into these relationships, unraveling the underlying mechanisms and enabling the optimization of the algorithm for everyday applications.

Source Code

The source code for the standard map random number generator algorithm implementation can be found in the following GitHub repository:

https://github.com/blueguy42/Standard-Map-RNG

Acknowledgment

The author would like to bestow the highest gratitude to Dr. Ir. Rinaldi Munir, MT. as the lecturer of IF4020 Cryptography for having taught his knowledge as well as giving the author the opportunity to write this paper. The author would also like to warmly thank friends and family that have played an integral role in helping the author in any way to finish this paper.

References

[1] Munir, R. (2019) 'Pembangkit Bilangan Acak', in *Kriptografi*. 2nd edn. Penerbit Informatika, pp. 359–388.

[2] Rasbad, S.N. (1997) 'The Standard Map', in *Chaotic Dynamics of Nonlinear Systems*. New York, NY: Wiley, pp. 178–179.

[3] Chirikov, B.V. (1979) 'A universal instability of many-dimensional oscillator systems', *Physics Reports*, 52(5), pp. 263–379. doi:10.1016/0370-1573(79)90023-1.

[4] Pisarchik, A.N. and Zanin, M., 2010. Chaotic map cryptography and security. *Encryption: Methods, Software and Security*, pp.1-28.

[5] Lian, S., Sun, J. and Wang, Z. (2005) 'A block cipher based on a suitable use of the chaotic standard map', *Chaos, Solitons & Fractals*, 26(1), pp. 117–129. doi:10.1016/j.chaos.2004.11.096.

[6] Wong, K.-W., Kwok, B.S.-H. and Law, W.-S. (2008) 'A fast image encryption scheme based on Chaotic Standard Map', *Physics Letters A*, 372(15), pp. 2645–2652. doi:10.1016/j.physleta.2007.12.026.

[7] Hamdi, M., Rhouma, R. and Belghith, S. (2017) 'A selective compression-encryption of images based on SPIHT coding and Chirikov Standard Map', *Signal Processing*, 131, pp. 514–526. doi:10.1016/j.sigpro.2016.09.011.

[8] L'Ecuyer, P. (2017) 'History of uniform random number generation', *2017 Winter Simulation Conference (WSC)* [Preprint]. doi:10.1109/wsc.2017.8247790.

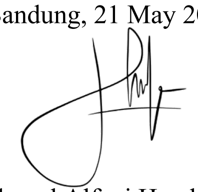
[9] Hasselblatt, B. and Katok, A.B. (2010) *A first course in dynamics: With a panorama of recent developments*. Cambridge: Cambridge University Press.

[10] Bisong, E. (2019) 'Matplotlib and Seaborn', *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 151–165. doi:10.1007/978-1-4842-4470-8_12.

Declaration

I hereby declare that this paper I have written is my own work, not a summary or translation of someone else's paper, and is not plagiarized.

Bandung, 21 May 2023



Ahmad Alfani Handoyo