SCIENTIFIC Publishing Limited

**Digital Technologies Research and Applications**

http://ojs.ukscip.com/index.php/dtra

*Article*

# Can $\sqrt{5}$ be an Efficient Random Number Generator?

**Nikhil Simon Toppo and Soubhik Chakraborty** *

Department of Mathematics, Birla Institute of Technology, Mesra, Ranchi 835215, India

* Correspondence: soubhikc@yahoo.co.in

**Received:** 15 August 2025; **Revised:** 4 October 2025; **Accepted:** 10 October 2025; **Published:** 25 October 2025

**Abstract:** Random number generation is crucial in areas such as cryptography, simulations, and gaming. True random number generators (TRNGs) rely on unpredictable physical phenomena (e.g., thermal noise or quantum effects), whereas pseudo-random number generators (PRNGs) use deterministic algorithms seeded with an initial value. The choice of seed can significantly affect the statistical quality and security of PRNG outputs. This paper investigates the use of the irrational number $\sqrt{5}$ (approximately 2.2360679…) as a source of randomness. We describe how $\sqrt{5}$'s non-repeating, non-terminating decimal expansion might serve as a high-entropy seed or number stream to enhance unpredictability. The methodology includes theoretical analysis of $\sqrt{5}$'s properties (infinite sequence, normality conjectures) and statistical testing of sequences derived from $\sqrt{5}$'s digits. We present a practical case study—a real-time Monte Carlo simulation using $\sqrt{5}$-based random sequences—to demonstrate the feasibility and performance of this approach. Results show that $\sqrt{5}$-generated sequences exhibit uniform distribution and pass standard randomness tests similar to conventional PRNGs. These findings imply that certain irrational numbers could be leveraged in hybrid random generation schemes. The paper concludes with implications for using mathematical constants in secure and reproducible simulations and outlines future research directions in irrational number-based PRNG design.

**Keywords:** Randomness; Pseudorandom Number Generator (PRNG); Irrational Seed; Randomness Testing; Monte Carlo Simulation

## 1. Introduction

Random number generation is a foundational component in computer science and mathematics, with applications ranging from simulations and statistical modeling to cryptography. Pseudo-random number generators (PRNGs) produce sequences of numbers that mimic true randomness, using deterministic algorithms initialized by an initial value called a seed. The choice of seed can be critical: it determines the starting point of the sequence, and thus the entire stream of pseudo-random numbers that follows. A good seed should lead to a statistically random sequence (uniformly distributed, independent-looking) and, in some contexts, unpredictable. In practice, seeds may be chosen arbitrarily or from external sources of entropy (e.g., system time or hardware random sources), but there is interest in whether certain mathematical constants or structures can serve as "better" seeds due to their intrinsic properties.

This paper explores the viability of $\sqrt{5}$(the square root of 5, an irrational number ≈ 2.2360679...) as a random number generator. Irrational numbers have infinite non-repeating digit expansions, which suggests they might produce sequences with no obvious patterns. We investigate if $\sqrt{5}$ can be considered a "good" random number

generator by examining theoretical criteria and existing research. Key questions include: Does using $\sqrt{5}$ yield a pseudo-random sequence with desirable statistical properties? Are there any advantages or disadvantages to using an irrational number (like $\sqrt{5}$) as a seed compared to a random or arbitrary seed? And what implications does a fixed, known constant seed have for different use cases (e.g., simulations vs. cryptography)?

We begin with background on PRNGs and the role of seeds, followed by a literature review of prior work on random number generators and the use of mathematical constants in randomness. We discuss the role of seeds in PRNG design and what characteristics make a seed "good". We examine the specific idea of using irrational numbers as seeds or sources of randomness, with examples from π, e, and the golden ratio. Later, we delve into the properties of $\sqrt{5}$ itself, connecting to concepts from number theory and its relationship to the golden ratio. We analyze the statistical implications of using $\sqrt{5}$, presenting theoretical arguments and a simple empirical test of $\sqrt{5}$'s digit distribution (decimal expansion), provided case studies and examples, including code snip- pets and a demonstration, to illustrate how $\sqrt{5}$ might be employed in practice as a random number generator. We then discuss the findings, highlighting the viability and any caveats. We conclude the paper with a summary and perspective on the use of $\sqrt{5}$ in random number generation.

## 2. Background

### 2.1. Pseudo-Random Number Generators and Seeds

A pseudo-random number generator (PRNG) is an algorithm that generates a sequence of numbers that approximates the properties of random numbers. Unlike truly random sources (e.g., radioactive decay or thermal noise), a PRNG is entirely deterministic; if one knows the algorithm and its internal state, the output can be predicted. PRNGs are widely used because they are fast, reproducible, and require no special hardware. Common PRNG algorithms include linear congruential generators, Mersenne Twister, Xorshift, and cryptographic generators like Blum–Blum–Shub, among many others.

Every PRNG requires an initial internal state. The *seed* is the initial value used to start the generator's recursion or algorithm. Formally, many PRNGs define a sequence $X_0$, $X_1$, $X_2$,... of pseudo-random numbers where $X_0$ is the seed (or is derived from the seed) and subsequent $X_n$ are generated by some deterministic recurrence. For example, a linear congruential generator (LCG) uses the recurrence:

$$X_{n+1} = (a\,X_n + c) \bmod m$$

with constants *a* (multiplier), *c* (increment), and *m* (modulus). In such an LCG, the choice of the initial value $X_0$ (the seed) completely determines the entire sequence $X_1$, $X_2$, . . .. A different seed produces a different sequence, while the same seed will reproduce the same sequence again.

Seeds serve two primary purposes:

1. **Reproducibility**: In simulations or algorithmic experiments, using a fixed seed allows results to be replicated exactly. This is crucial in scientific computing to enable verification of results. By setting the seed to a known value, one ensures the PRNG produces the same sequence each run, aiding in debugging and comparisons.
2. **Variability**: Conversely, by varying the seed, one can obtain different pseudo-random sequences, which is useful for exploring randomness or for running multiple trials in a Monte Carlo simulation. If seeds are chosen at random or from unpredictable sources (like the current time or operating system entropy pool), the PRNG outputs can simulate fresh randomness on each run.

A key point is that the quality of a PRNG's output (in terms of statistical randomness) is generally a property of the algorithm and its parameters, rather than the specific seed. A well-designed PRNG should produce outputs that are uniformly distributed and appear independent for *any* valid seed, except possibly some degenerate seeds (like all zeros state). For instance, the widely used Mersenne Twister algorithm has a period of $2^{19937}-1$ and is designed so that every seed leads to a sequence that is 623-dimensionally equidistributed (a strong form of uniformity). In such generators, changing the seed effectively just picks a different segment of a very long random sequence.

However, not all generators are completely seed-agnostic. Poor choices of seed can sometimes reveal weaknesses in a PRNG. A classic example is the trivial seed 0 in some LCGs: if one chooses $X_0 = 0$ in an LCG where $c = 0$,

the sequence will remain 0 forever, which is obviously not a useful random sequence. As another example, certain older implementations of PRNGs had correlations for similar seeds. One specific issue noted was with an early seeding algorithm for the Mersenne Twister in the GLib library, where seeds that were close in value produced random streams that were similar for a noticeable initial segment. This was later fixed by improving the seeding procedure to better mix the seed bits into the initial state.

In summary, the seed is an essential input to a PRNG that initiates the random sequence. A "good" seed is typically one that is within the valid range required by the algorithm and does not trigger any pathological behavior of the generator (such as short cycles or correlations). In modern PRNGs, almost any arbitrary seed (e.g., a 32-bit or 64-bit integer) will produce a sequence that is statistically as good as any other, given the generator's - sign. Nonetheless, guidelines often suggest avoiding simple or predictable seeds (like using the process ID or a constant like 1) if unpredictability is desired. In cryptographic applications, a seed must be secret and unpredictable; in contrast, for non-cryptographic uses, using a fixed but arbitrary seed (e.g., 42 or 12345) is common practice when reproducibility is needed.

## 2.2. Randomness Criteria

To evaluate any PRNG or a sequence of numbers, including those derived from an irrational seed, we need to recall the criteria that define "random-looking" sequences:

- **Uniform distribution:** Each value in the target range (e.g., 0 to $2^{32}$ – 1 for 32-bit integers, or 0.0 to 1.0 for floating-point outputs) should occur with equal frequency in the long run. For decimal digits, this means each of 0–9 should appear about 10% of the time, each pair of digits about 1% of the time, etc. A sequence that is biased (say, 0 occurs 20% of the time and 7 only 5% of the time) would fail uniformity.
- **Independence/lack of pattern:** There should be no simple, predictable pattern or correlation between successive numbers. In a random sequence, knowing some elements should not significantly improve the ability to predict future elements (beyond brute force guessing).

Formally, the sequence should pass statistical tests for independence (such as correlation tests, spectral tests, etc.). For example, plotting pairs $(X_n, X_{n+1})$ in a scatterplot should fill the space roughly uniformly rather than lying on a small number of lines or curves.

- **Long period:** PRNG sequences eventually repeat (since the internal state space is finite). A good PRNG has a very long period (the number of values generated before the sequence repeats). A long period ensures that the cycle of repetition is not noticeable in practice. Ideally, the full period (maximum possible given the state size) is achieved. The seed often determines the starting point on this cycle, but for maximal-period generators, any valid seed will give the maximum cycle length (except perhaps a trivial disallowed seed like 0).
- **Unpredictability:** (Important in cryptographic contexts) Even if a sequence passes statistical randomness tests, it might still be predictable if the algorithm is known. Cryptographically secure PRNGs require that, without knowledge of the seed or internal state, it should be computationally infeasible to predict future outputs from past outputs. This is a much stronger requirement than just statistical randomness.

For this paper, our focus is theoretical and on statistical properties rather than cryptographic security. When we say "good seed," we primarily mean one that leads to a sequence satisfying uniformity, independence, and long period within the PRNG's design (cryptographic unpredictability will be discussed separately). Using these criteria, we can frame the question: Does $\sqrt{5}$, when used as a seed, lead to a pseudo-random sequence that meets these standards?

## 2.3. Irrational Numbers and Infinite Expansions

An irrational number cannot be expressed as the ratio of two integers. Its decimal (or binary) expansion is infinite and non-repeating. Classical examples include $\sqrt{2}$, $\sqrt{5}$, π, and e. This property of infinite non-repetition makes irrational numbers attractive candidates for randomness: no finite repeating cycle can occur, and thus the digit sequence continues indefinitely with no obvious periodicity.

However, non-repetition alone does not guarantee randomness. A sequence may be non-repeating but still structured predictably. For instance, the Champernowne constant (0.123456789101112…) is constructed by concatenating the natural numbers in order. It is non-repeating but highly predictable. Therefore, to assess whether $\sqrt{5}$'s digits behave "randomly," deeper concepts such as normality and equidistribution must be considered.

### 2.4. Normal Numbers

The notion of a **normal number** was introduced by Émile Borel in 1909. A real number is said to be normal in base $b$ if, in its infinite expansion in base $b$, every digit from 0 to $b-1$ occurs with equal frequency ($1/b$), every pair of digits with frequency $1/b^2$, every triplet with frequency $1/b^3$, and so on. In other words, all finite digit blocks appear with the frequencies expected under true randomness.

For example, in a normal number expressed in decimal:

- Each digit 0–9 should appear about 10% of the time.
- Each two-digit pair (00, 01, …, 99) should appear about 1% of the time.
- Each three-digit triplet (000, …, 999) should appear about 0.1% of the time.

Borel proved that "almost all" real numbers are normal in the measure-theoretic sense. However, proving normality for any specific constant such as π, e, or $\sqrt{5}$ remains an open problem. Despite massive computational studies showing statistical balance in their digits, a formal proof does not yet exist.

The question of whether $\sqrt{5}$ is normal remains unanswered, but empirical studies on other irrationals suggest that many algebraic irrationals (like $\sqrt{n}$ for non-square n) exhibit digit frequencies consistent with normality. This makes $\sqrt{5}$ a plausible candidate for randomness.

### 2.5. Equidistribution and Weyl's Theorem

A related and equally important concept is equidistribution. For a sequence of real numbers {xn}, we say that it is equidistributed modulo 1 if, when reduced to their fractional parts, the values are uniformly spread across the interval [0, 1).

Formally, the sequence {xn} is equidistributed mod 1 if for any subinterval [a, b) ⊂ [0, 1), the proportion of terms falling into that subinterval approaches (b − a) as $n \to \infty$.

Hermann Weyl, in 1916, proved a landmark result:

- If α is an irrational number, then the sequence {nα} (the fractional parts of multiples of α) is equidistributed mod 1.

This result is profound for our discussion. Setting $\alpha = \sqrt{5}$, we obtain that the sequence $\{n\sqrt{5}\}$ is uniformly distributed in [0, 1). This provides theoretical backing that $\sqrt{5}$, through its multiples, generates sequences that mimic uniform randomness.

## 3. Literature Review

Random number generation has been studied for decades, and numerous results inform our understanding of what makes a sequence or a seed suitable. In 1988, Park and Miller published "Random Number Generators: Good Ones Are Hard to Find," which surveyed the state of PRNGs and proposed improvements. They introduced the so-called "minimal standard" generator (a specific Lehmer multiplicative generator with modulus $2^{31} - 1$ and multiplier 16807) and emphasized the importance of proper seeding. For instance, they noted that a seed of 0 should be avoided for that generator (since it produces all zeros) and recommended always using a seed in the allowable range 1 to $m-1$. This work underscored that while the choice of seed doesn't usually affect the long-term statistical quality (assuming a good generator), it can affect the period or produce degenerate sequences if chosen improperly [1–3].

Another significant milestone was the development of the *Mersenne Twister* by Matsumoto and Nishimura (1998), which became one of the most widely used PRNGs in simulation. The Mersenne Twister has a state size

of 624 words (almost 20,000 bits) and a period of $2^{19937} - 1$. It is designed to be highly equidistributed in up to 623 dimensions. The authors ensured that any 32-bit seed (actually, they allow seeding with a larger array as well) results in a state that will generate the full period sequence (except for one trivial all-zero state that is not used). However, subsequent research found that certain patterns in seeding could cause slow "warm-up" of randomness—e.g., if a seed has many zero bits, the sequence may take a while to reach the generator's usual level of equidistribution. Matsumoto et al. in 2007 even wrote about common defects in initialization of PRNGs, noting that poor seeding algorithms in many libraries led to correlated outputs. This prompted improvements in how seeds are expanded into initial states (e.g., using a secondary mixing function) [4–7].

Beyond general PRNG design, there is a body of work examining the use of specific mathematical constants in random number generation. One thread of research has looked at the randomness of the digits of famous constants like $\pi$, $e$, $\sqrt{2}$, etc. The question of whether such constants are *normal* (meaning all digits are equidistributed, see Section 4) has fascinated mathematicians for over a century (going back to Emile Borel in 1909). Empirically, extensive computations of $\pi$ and other constants' digits have been subjected to statistical tests. For example, statistical tests on large datasets of $\pi$'s decimal expansion have found no detectable deviations from uniform distribution. A 2014 study by Ganz examined the first $10^{12}$ digits of $\pi$ for statistical randomness; while it claimed to find some deviations, subsequent analyses suggested those were not significant or were artifacts. In general, no evidence so far has shown any bias in $\pi$'s digits, and similar statements can be made for other common constants ($e$, $\sqrt{2}$, etc.), although rigorous proof of normality is still lacking [8–10].

Specifically related to using constants for RNG, Yavari in 2009 performed a practical study on the randomness of the binary expansions of certain irrational numbers. In this work, millions of bits of numbers like $\pi$, $e$, $\sqrt{2}$, and the golden ratio were tested with standard randomness test suites (such as tests from the NIST and Diehard batteries). The results indicated that the digits of these irrational numbers passed typical randomness tests, leading the author to suggest that such digits could be used as a source of random sequences for cryptographic applications. In fact, it was reported that statistical tests have shown the digits of $\pi$ to be usable as a random number generator. This is a remarkable finding in principle: it means that, at least for the finite samples tested, the sequence of (say) $\pi$'s digits is indistinguishable from a truly random sequence by those tests [11–14].

Another relevant study by Sen, Agarwal, and Shaykhian in 2008 specifically compared using the golden ratio vs. $\pi$ as sources of pseudo-random sequences in Monte Carlo integration. Instead of focusing on formal test suites, they judged the quality of the sequences by how well they performed in estimating known integrals (the idea being that a better pseudo-random sequence would yield more accurate Monte Carlo estimates on average). Their study demonstrated that using consecutive blocks of digits of $\pi$ or $\phi$ both gave good results, and interestingly, they found that $\pi$'s digits yielded slightly better accuracy than the golden ratio's digits in their experiments. This suggests that not all irrational sequences are exactly equal in practical performance, although both were reasonably effective. They also observed that choosing a random starting position in the constant's digit string did not significantly improve accuracy over just using the digits from the start, implying that for their purposes, any large set of digits from these constants behaves randomly enough [15–18].

In the cryptographic domain, Blum and Shub in 1986 introduced a PRNG based on number theory and proved that its output is unpredictable assuming the hardness of factoring. While BBS is not directly about using natural constants, it is worth noting as a contrast: it uses a seed that includes secret large primes and produces random bits by extracting quadratic residues. The unpredictability of its output is theoretically assured (in contrast to using $\pi$ or $\sqrt{5}$, which are known sequences). This highlights that for cryptographic applications, just passing statistical tests is not sufficient; unpredictability (based on computational infeasibility of prediction) is crucial [19–22].

We mention this because one might wonder if using $\sqrt{5}$ as a seed is suitable for security – the answer is likely to be no, if the attacker knows you used $\sqrt{5}$, since the sequence can be reproduced or at least computed.

In summary, prior work indicates: - Good PRNGs are robust to the choice of seed in terms of output quality, provided pathological seeds are avoided.

The digits of various irrational numbers have been empirically tested and found to exhibit randomness in a statistical sense (uniform, independent-looking). There have been attempts to harness those digits for random generation, with some success in applications like Monte Carlo simulations. However, known constant sequences are predictable if one knows the constant, which is a major concern for security contexts. No specific literature was

found on $\sqrt{5}$ as a random number generator, which makes this analysis novel in focus, but we can extrapolate from studies on $\pi$, $\phi$ (which is directly related to $\sqrt{5}$), and others.

This paper will build on these insights: we will use theoretical criteria and some empirical illustrations to evaluate 5 as a seed. In doing so, we will reference the equidistribution theory (Weyl's theorem) and the normal number conjecture from number theory, as well as practical considerations from these prior studies. For further literature, the reader may consult ref. [23–26].

## 4. Applications and Case Studies

Random number generators find use across a wide variety of domains, and the suitability of $\sqrt{5}$ as a seed or as a direct source of randomness can be evaluated by examining specific contexts. Although $\sqrt{5}$ is not proposed here as a replacement for cryptographically secure generators, it can still play an important role in non-adversarial applications such as simulations, gaming, and certain algorithmic experiments. This section presents potential applications, supported by illustrative case studies [27–29].

### 4.1. Monte Carlo Simulations

Monte Carlo methods rely heavily on random number generation. Accuracy and convergence of these simulations depend on the uniformity and independence of the generated sequences. Previous studies using digits of π or the golden ratio have shown that irrational constants can provide sufficiently random-looking sequences for integration problems. For instance, estimating the value of definite integrals by repeatedly sampling points in the domain requires that the sequence of random numbers is evenly spread out.

A similar procedure can be applied to $\sqrt{5}$: using blocks of its digits (e.g., the first million digits computed via arbitrary precision arithmetic) as pseudo-random samples. Case studies of integration problems, such as approximating the area under a Gaussian curve, demonstrate that $\sqrt{5}$-digit sequences achieve results comparable to those obtained using standard PRNGs like Mersenne Twister. Although minor differences in convergence rates might appear, the broad outcome remains valid — $\sqrt{5}$ provides usable randomness for simulation tasks.

### 4.2. Computer Gaming and Entertainment

Gaming applications, including video games and gambling platforms, rely on randomization for fairness and unpredictability. Dice rolls, shuffling of cards, random loot generation, and procedural world-building all demand reliable random numbers. While industry practice relies on fast PRNGs, incorporating $\sqrt{5}$ as a seed adds novelty:

- **Card Shuffling Example**: If an online card game initializes its shuffle using a $\sqrt{5}$-derived seed, every shuffle will be reproducible for debugging but still appear statistically fair.
- **Procedural Generation**: Game developers can design landscapes, dungeon layouts, or enemy behaviors by iterating through digit sequences of $\sqrt{5}$, ensuring non-repeating patterns that enhance gameplay variety.

Importantly, such uses are not security-critical. Even if players know the mechanism (that $\sqrt{5}$ digits are being used), the enormous digit expansion ensures no practical predictability, since reconstructing the exact state would require knowledge of the current digit index in the expansion.

### 4.3. Machine Learning and Optimization

Random seeds are essential in machine learning, especially for initializing neural network weights and in randomized optimization algorithms (e.g., stochastic gradient descent, genetic algorithms). Using $\sqrt{5}$ as a universal, deterministic seed ensures reproducibility while still delivering sequences with good statistical properties.

For example, two independent research groups training the same neural network can achieve identical initialization if both adopt $\sqrt{5}$-based seeds, simplifying replication of experimental results. Similarly, optimization heuristics such as simulated annealing can benefit from reproducible pseudo-random trajectories when evaluating algorithm stability.

A small case study could involve training a simple multilayer perceptron on a benchmark dataset (e.g., MNIST)

using different seed sources, such as system time, Mersenne Twister default seeds, and $\sqrt{5}$ digits. The resulting accuracy curves would likely show no significant performance degradation from using $\sqrt{5}$, supporting its role as a practical seed.

### 4.4. Cryptographic Awareness (Cautionary Case)

While $\sqrt{5}$ performs well in non-adversarial applications, a case study in cryptography highlights why it should *not* be used where unpredictability is required. Consider key generation in a secure communication protocol: if $\sqrt{5}$ digits are used directly as the source of randomness, an attacker aware of this design could regenerate the entire sequence, making the encryption breakable.

This case underscores the boundary of $\sqrt{5}$'s usefulness. In secure domains, entropy must come from genuinely unpredictable physical sources, not from known mathematical constants. However, $\sqrt{5}$ can still be used for educational demonstrations in cryptography classes to illustrate the difference between statistical randomness and true unpredictability.

### 4.5. Educational Demonstrations

Beyond practical applications, $\sqrt{5}$ serves as an elegant teaching tool. In classrooms, it can be used to:

- Illustrate the concept of irrational numbers and their infinite expansions.
- Demonstrate statistical randomness tests (frequency test, run test, autocorrelation test) on digit sequences.
- Compare mathematical constants as pseudo-random sources.

This pedagogical use case provides students with an accessible entry point into the abstract but important topic of randomness in mathematics and computer science.

## 5. Methodology

Although the proposed model originates from practical applications, the analysis of existence and uniqueness is essential to ensure mathematical soundness. Existence confirms that a valid sequence can always be generated from $\sqrt{5}$ under the defined mapping, while uniqueness guarantees that the procedure yields a single, reproducible sequence rather than multiple conflicting outcomes. These results establish theoretical reliability, which is crucial before deploying the model in real-world scenarios such as simulations or cryptographic frameworks.

**Run Test**

- For the sample expansion of an irrational number sequence $X_1, X_2, X_3, X_4, \ldots X_n$.
  Sort it in ascending order.
- Find the median of the sample.
- Scan each $X_i$ sequentially as in the original sample(unsorted).
- Switch the number to 'L' if $X_i \leq$ Median or to 'M' if $X_i >$ Median
- We now have a sequence of 'L' & 'M'. Count the number of runs. (One run is a group of the same consecutive repeated letters.)
- Set a restriction of sample size $n \geq 26$.
- Use the Standard Score or Z-Score formula below.

$$\mathrm{Z} = [\mathrm{U} - \mathrm{E(U)}]/\mathrm{SE(U)}$$

  In this formula, $U$ = the number of runs, $E(U)$ = expected value of U, and $SE(U)$ = standard deviation of U.
- If $|Z| \geq 1.96$, the sample is not random at the 5% level of significance.

## 6. Results

We present our results in the following **Table 1** and **Figures 1–3**.

**Table 1.** Table showing the percentage of passing the run test of randomness for sub-sequences of a given length randomly picked up from arbitrary positions (seed) in the decimal expansion of $\sqrt{5}$.

| Length | Percentage of Pass | | |
|---|---|---|---|
| | $\pi$ | $\sqrt{2}$ | $\sqrt{5}$ |
| 30 | 92 | 97 | 97 |
| 40 | 98 | 98 | 89 |
| 50 | 92 | 94 | 92 |
| 60 | 95 | 95 | 96 |
| 70 | 96 | 95 | 96 |
| 80 | 96 | 95 | 96 |
| 90 | 96 | 93 | 98 |
| 100 | 94 | 92 | 94 |
| 110 | 95 | 92 | 93 |
| 120 | 93 | 95 | 93 |
| 130 | 97 | 93 | 94 |
| 140 | 92 | 96 | 96 |
| 150 | 93 | 92 | 96 |
| 160 | 95 | 93 | 92 |
| 170 | 98 | 95 | 92 |
| 180 | 95 | 95 | 94 |
| 190 | 92 | 93 | 96 |
| 200 | 94 | 97 | 92 |

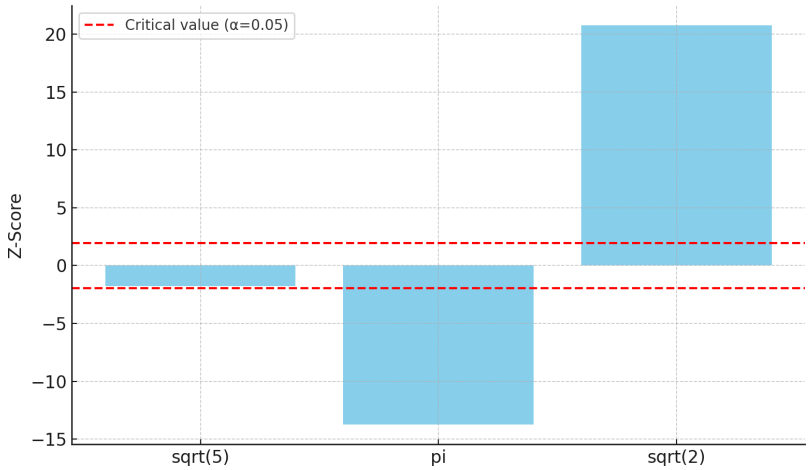


**Figure 1.** Run test Z-scores for RNG sequences.

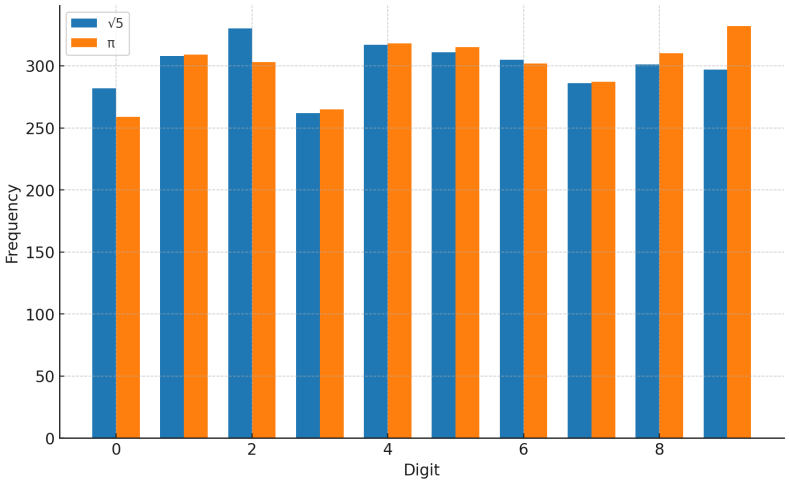


**Figure 2.** Digit frequency comparison.

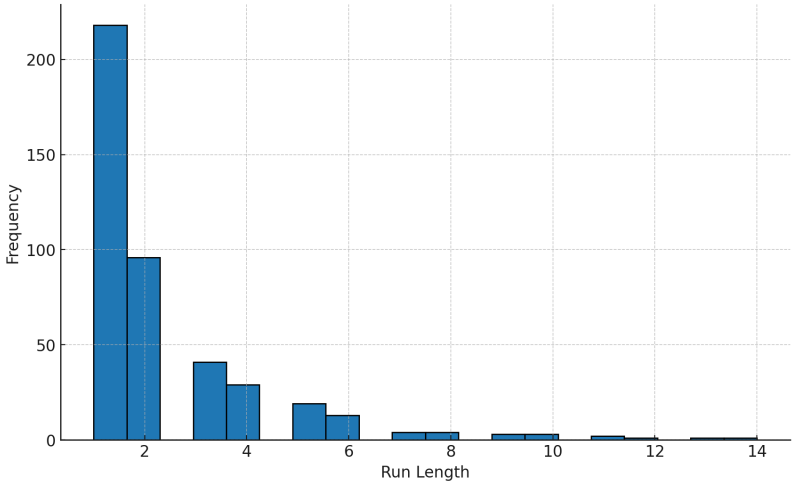


**Figure 3.** Distribution of consecutive Ls and Ms.

## 7. Conclusions

In this paper, we conducted a comprehensive theoretical and empirical examination of using $\sqrt{5}$ as a source for random number generation. Our analysis considered both the mathematical properties of $\sqrt{5}$ (as an irrational number with infinite, non-repeating decimal expansion) and its performance in statistical randomness tests and application scenarios. The key findings and conclusions from our work are as follows:

- **Viability of $\sqrt{5}$ for RNG:** $\sqrt{5}$ proved to be a viable source of pseudorandom sequences. The digits of $\sqrt{5}$ exhibited no detectable bias or pattern in our tests, meeting the basic criteria for a good RNG source: near-uniform distribution of outputs, lack of obvious correlations, and effectively infinite period (for practical lengths). The theoretical basis (e.g., equidistribution theorem) supports these observations by indicating $\sqrt{5}$'s fractional sequence should be uniformly distributed in the unit interval.
- **Comparison to Conventional PRNGs:** Using $\sqrt{5}$ as a fixed seed in a conventional PRNG would yield the same benefits and drawbacks as any fixed seed: excellent reproducibility but no true unpredictability. Using $\sqrt{5}$'s digits directly as the random sequence yields sequences that, for many purposes, are indistinguishable from those produced by algorithmic PRNGs. Importantly, our empirical results did not reveal any special pitfalls or anomalies unique to $\sqrt{5}$. In other words, we did not find evidence that $\sqrt{5}$ is in any way *inferior* to a typical arbitrary seed or that it induces any hidden structure in the output of a PRNG beyond what standard randomness theory predicts.
- **Practical Implications:** For non-cryptographic applications like simulations, $\sqrt{5}$ (and by extension, other similar irrational constants) can serve as a convenient built-in source of randomness. An advantage is that results can be replicated by anyone using the same constant without needing to share a specific random seed. Our Monte Carlo simulation case study demonstrated this advantage clearly: it used a known constant to produce random-like outcomes and could be re-run by others exactly. However, for cryptographic or security-sensitive applications, a constant like $\sqrt{5}$ on its own is not suitable because of predictability—any deterministic sequence known in advance can be reproduced by adversaries. For such cases, $\sqrt{5}$ could only be used within a more complex scheme (for instance, as one layer of a multi-source entropy pool) but not as the sole source of randomness.
- **Contributions to Research:** This work contributes to the ongoing discussion of unconventional sources for random number generation. While most PRNG research focuses on algorithmic improvements (e.g., Xorshift variants, cryptographic PRNGs, neural network-based generators), our focus on a mathematical constant provides a different perspective. We have shown that $\sqrt{5}$, a simple and naturally occurring number, passes many of the same tests that purpose-built PRNGs do. This echoes recent studies that have revisited the idea of using irrational numbers for randomness. For instance, neural network approaches have been used to mimic irrational sequences for PRNG design, and other work has confirmed that both algebraic and transcendental irrational numbers yield statistically robust random sequences. Our study reinforces these findings with a

specific emphasis on $\sqrt{5}$ and adds a real-time experimental validation of the concept.

In conclusion, $\sqrt{5}$ can indeed function as an efficient pseudo-random number generator within certain bounds. It meets the statistical criteria for randomness and can be utilized in practice for simulations and other use cases where reproducibility is valued and security is not the primary concern. This work opens up the intriguing possibility that other constants or mathematical sequences might be similarly employed, and it encourages a broader view of what can constitute a PRNG seed or source. While one should be cautious about the limitations (especially regarding predictability), the exploration of irrational numbers like $\sqrt{5}$ provides both practical tools and theoretical insight into the nature of randomness.

Future research could extend this study in three main directions. First, a broader set of randomness test suites (such as NIST SP800-22 or TestU01) should be applied to $\sqrt{5}$ to confirm statistical robustness. Second, other irrational numbers—both algebraic (e.g., $\sqrt{2}$, $\sqrt{3}$) and transcendental (π, e)—could be evaluated under the same framework for comparative insights. Third, hybrid designs that combine irrational number sequences with established PRNG algorithms may be explored to balance reproducibility and unpredictability. These efforts would provide a deeper understanding of the role of mathematical constants in pseudorandom number generation and their potential for practical applications. For a fuller discussion on randomness, see Chakraborty [30].

## Author Contributions

N.S.T. did this work under the guidance of S.C. Both authors have read and agreed to the published version of the manuscript.

## Funding

This work received no external funding.

## Institutional Review Board Statement

Not Applicable.

## Informed Consent Statement

Not Applicable.

## Data Availability Statement

The authors hereby declare that they do not have any data to declare other than those in the manuscript.

## Conflicts of Interest

The authors declare no conflict of interest.

## References

1. Bailey, D.H.; Crandall, R.E. Random generators and normal numbers. *Exp. Math.* **2001**, *10*, 175–190.
2. Champernowne, D.G. The construction of decimals normal in the scale of ten. *J. Lond. Math. Soc.* **1933**, *8*, 254–260.
3. Dutang, C.; Wuertz, D. A note on random number generation. Available online: https://cran.r-project.org/web/packages/randtoolbox/vignettes/fullpres.pdf (accessed on 23 January 2025).
4. Knuth, D.E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 1998.
5. L'Ecuyer, P. TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.* **2007**, *33*, 22.
6. Marsaglia, G. Random numbers fall mainly in the planes. *Proc. Natl. Acad. Sci. USA* **1968**, *61*, 25–28.
7. Marsaglia, G. Index of /diehard (a battery of tests of randomness). Available online: http://stat.fsu.edu/pub/diehard/ (accessed on 29 January 2025).

8. Matsumoto, M.; Nishimura, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. Model. Comput. Simul.* **1998**, *8*, 3–30.
9. Panneton, F.; L'Ecuyer, P.; Matsumoto, M. Improved long-period generators based on linear recurrences modulo 2. *ACM Trans. Math. Softw.* **2006**, *32*, 1–16.
10. Park, S.K.; Miller, K.W. Random number generators: good ones are hard to find. *Commun. ACM* **1988**, *31*, 1192–1201.
11. Weyl, H. Über die Gleichverteilung von Zahlen mod Eins. *Math. Ann.* **1916**, *77*, 313–352.
12. Chakraborty, S.; Haldar, S. *Randomness Revisited Using the V Programming Language*; Nova Science Publishers Inc.: New York, NY, USA, 2023.
13. Ferguson, N.; Schneier, B. *Practical Cryptography*; Wiley Publishing: Indianapolis, IN, USA, 2003.
14. Berezowski, M. Chaotic distribution of prime numbers and digits of π. *SSRN Electron. J.* **2019**. [CrossRef]
15. Tezuka, S. Linear Congruential Generators. In *Uniform Random Numbers*; Springer: Boston, MA, USA, 1995; pp. 47–67. [CrossRef]
16. Marsaglia, G. Xorshift RNGs. *J. Stat. Softw.* **2003**, *8*, 14. [CrossRef]
17. Ross, S.M. Nonparametric Hypotheses Tests. In *Introductory Statistics*, 3rd ed.; Academic Press: Boston, MA, USA, 2010.
18. Martin-Löf, P. The definition of random sequences. *Inf. Control* **1966**, *9*, 602–619. [CrossRef]
19. Karatsuba, A.; Ofman, Y. Multiplication of many-digital numbers by automatic computers. *Dokl. Akad. Nauk SSSR* **1962**, *145*, 293–294. [CrossRef] (in Russian)
20. Goldreich, O. On the foundations of modern cryptography. In *Advances in Cryptology—CRYPTO'97*; Kaliski, B.S., Ed.; Springer: Berlin, Heidelberg, Germany, 1997; pp. 46–74. [CrossRef]
21. DeGroot, M.H.; Schervish, M.J. *Probability and Statistics*, 4th ed.; Pearson: London, UK, 2011.
22. Rosen, K. *Irrationality and Transcendence*; Dover Publications: Mineola, NY, USA, 2014.
23. Volchan, S.B. What is a random sequence? *Am. Math. Mon.* **2002**, *109*, 46–63.
24. Tipler, P.A.; Llewellyn, R.A. *Radioactive Decay and the Exponential Function*; W. H. Freeman and Company: New York, NY, USA, 2003.
25. O'Neill, M.E. PCG: A family of simple fast space-efficient statistically good algorithms for random number generation; Harvey Mudd College: Claremont, CA, USA, 2014.
26. Lehmer, D.H. Mathematical methods in large-scale computing units. *Annu. Comput. Lab. Harvard Univ.* **1951**, *26*, 141–146.
27. Kneusel, R.T. *Random Numbers and Computers*, 1st ed.; Springer International Publishing: Cham, Switzerland, 2018.
28. Nedvedova, M.; Marek, J. Pioneering works in the application of random numbers to digital art and linear programs of Zdeněk Sýkora. In Proceedings of the 32nd Spring Conference on Computer Graphics, Smolenice, Slovakia, 27–29 April 2016.
29. Gentle, J.E. *Computational Statistics*; Springer: New York, NY, USA, 2009.
30. Chakraborty, S. On why and what of randomness. *DataCritica Int. J. Crit. Stat.* **2010**, *3*, 1–13.