Taking randomness for granted: the complexities of applying random number streams in simulation modelling

Maximilian Selmair

Tesla Manufacturing Brandenburg SE
15537 Grünheide (Mark), Germany
mselmair@tesla.com

***Abstract*—Uncertainty, as a constant companion of our world, is one major reason why simulation modelling takes precedence over static calculations to achieve accurate predictions. Computational random number generators are able to algorithmically determine values on the basis of random distributions, which utilise seed values to calculate streams of random numbers. This deterministic approach to replicating seemingly non-deterministic numbers ensures stochastic models to be reproducible at any time – one of the major requirements of simulation models. However, there are some pitfalls in the application of random number streams in modelling and simulation, which may even mislead experienced developers. In addition to a general introduction of the history of random number generators, this article shares empirical considerations and means by which the utilisation of random number streams can be improved to deliver valid and reliable results.**

Keywords*—Randomness; Modelling and Simulation; Seed Values; Random Number Generators

I. Introduction and the Historical Perspective

The digital replication of any system or process involving randomness requires a method to generate or obtain numbers that are both random and reproducible. Practical examples for random occurrences in the field of production and logistics are service times, interarrival times and maintenance occurrences. This article focuses on the history of random number generators as well as the potential drawbacks of utilising deterministic random number generators for simulation modelling. The first paragraph introduces how random values can be generated efficiently from a predetermined probability distribution in order to provide data sets for simulation modelling.

The modest beginnings of generating random numbers date back to over a century ago. (Hull and Dobell, 1962; Dudewicz and Dalal, 1971; Morgan, 1984). The earliest samples were not generated by computer, but literally carried out by hand: flipping coins, throwing dice, dealing out game cards or the lottery draws. Even today, most lotteries are still operated in this manner to avoid fraud allegations. As early as the 20th century, gamblers were joined by statisticians in their quest to explore random numbers and mechanised devices were built to generate random numbers more efficiently. Particular examples from the late 1930s are e. g. Kendall and Smith (1938) – the use of a spinning disk to select values from a turntable containing a hundred thousand random digits. Only two years later, electric circuits based on randomly pulsating vacuum tubes were used to deliver random digits at much higher rates of up to 50 per second. Such a device, referred to as a random number machine, was used by the British General Post Office to pick the winners of the Premium Savings Bond lottery (Thomson, 1959). Electronic devices were also used by the Rand Corporation (2001) to produce a sequence of a million random numbers. Some past examples of different approaches were picking numbers randomly out of phone books and using digits in an expansion of π, such as 0.1415926535, 0.8979323846, 0.2643383279, etc. (Tu and Fischbach, 2005). Another notable example of retrieving random numbers was proposed by Yoshizawa et al. (1999), who described a physics-based random number generator that relied on the radioactive decay of the nuclide Americium-241.

As computers and, later on, simulation modelling became more relevant, computational random number generators began to gain popularity. A first attempt in this direction was the use of the previously mentioned Rand Corporation's table in a computer's memory (Rand Corporation, 2001). This solution depended on substantial memory requirements and a vast amount of time to retrieve new values. Research in the 1940s and 1950s developed towards more deterministic strategies of generating random numbers. These values were sequential, which means that each new number was determined by one or sometimes several of its predecessors according to a fixed algorithm. The first known deterministic generator was proposed by von Neumann (1951). The well-known mid-square method is demonstrated in the subsequent example.

A researcher may begin with a four-digit positive integer $Z_0 = 1234$ and square it to obtain an integer with at least eight digits. The central four digits of this eight-digit number constitutes the next four-digit number, Z_1. By placing a decimal point on the left of Z_1, the first random number between 0 and 1 (U_1) is yielded. Following this procedure, an unlimited sequence of deterministic "random" values can be created. Table I lists the first few examples.

At first sight, the mid-square method seems to provide a seemingly suitable set of random numbers. However, a main disadvantage of this method is the strong tendency of the

Communications of the ECMS, Volume 36, Issue 1,
Proceedings, ©ECMS Ibrahim A. Hameed, Agus Hasan,
Saleh Abdel-Afou Alaliyat (Editors) 2022
ISBN: 978-3-937436-77-7/978-3-937436-76-0(CD) ISSN 2522-2414

i	Z_i	U_i	Z_i^2
0	1234	—	01522756
1	5227	0.5227	27321529
2	3215	0.3215	10336225
3	3362	0.3362	11303044
4	3030	0.3030	09180900
5	1809	0.1809	03272481

Table I: Sample calculation for a set of random numbers with the mid-square method based on value 1234

generated values to converge to zero.

A more fundamental objection to the mid-square method is that it does not yield "random" values at all, if one considers random values to be unpredictable in nature. That is, if we know one number, we have to acknowledge that it determines the succeeding numbers as the method stipulates. In more recent times, the first number in a sequence or stream of random numbers is referred to as *seed value*. With a given Z_0, the whole sequence of Z_i's and U_i's is determined. This characteristic applies to all deterministic generators. These deterministic random generators are often faulted to be generating pseudo-random numbers. In the author's opinion, pseudo-random is a misnomer, indeed it is an oft-quoted remark by von Neumann (1951), who declared that *"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number – there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method... We are here dealing with mere "cooking recipes" for making digits... "* (von Neumann, 1951).

Although this quote dates back to over 70 years ago, it does not seem to have lost its relevance today. Furthermore, it is rarely stated that von Neumann proceeds in the same paragraph that these "recipes" *"probably can not be justified, but should merely be judged by their results. Some statistical study of the digits generated by a given recipe should be made, but exhaustive tests are impractical. If the digits work well on one problem, they seem usually to be successful with others of the same type"* (von Neumann, 1951). Lehmer (1951) offered a more practice-oriented definition: *"A random sequence is a vague notion embodying the idea of a sequence in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests traditional with statisticians and depending somewhat on the use to which the sequence is to be put"* (Lehmer, 1951).

Since Lehmer's proposal, further formal definitions and empirical considerations about randomness were formulated in the subsequent decades:

- *"Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number – there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method"* von Neumann (1951).
- *"[A] sequence is random if it has every property that is shared by all infinite sequences of independent samples of random variables from the uniform distribution"* Franklin (1963).
- *"[...] random numbers should not be generated with a method chosen at random. Some theory should be used"* Knuth (1968).
- *"The generation of random numbers is too important to be left to chance"* Coveyou (1969).
- *"[In statistics] you have the fact that the concepts are not very clean. The idea of probability, of randomness, is not a clean mathematical idea. You cannot produce random numbers mathematically. They can only be produced by things like tossing dice or spinning a roulette wheel. With a formula, any formula, the number you get would be predictable and therefore not random. So as a statistician you have to rely on some conception of a world where things happen in some way at random, a conception which mathematicians don't have"* LeCam (1988).
- *"Sequences of random numbers also inevitably display certain regularities. [...] The trouble is, just as no real die, coin, or roulette wheel is ever likely to be perfectly fair, no numerical recipe produces truly random numbers. The mere existence of a formula suggests some sort of predictability or pattern"* Peterson (1998).
- *"The practical definitions of randomness – a sequence is random by virtue of how many and which statistical tests it satisfies and a sequence is random by virtue of the length of the algorithm necessary to describe it [...]"* Bennett (1999).

The author agrees with most researchers that deterministic generators, if appropriately designed, can replicate numbers which would have been generated by independent draws from the $U(0, 1)$ distribution and would be also pass a series of statistical tests. In summary, in the domain of simulation modelling, deterministic random number generators are used to replicate stochastic occurrences and measure their influence on a system. These generators allow simulation modellers to achieve reproducible scenarios with an unlimited set of seemingly non-determined values. In simulation modelling, a replication is defined as having the same set of parameters, but different stochastic influences. Each replication is based on a predetermined *seed value* (e. g. 1, 2, 3, ...), defined as a specific reproducible stream of random numbers. The purpose of this article is to address the complexities of simulation modelling with random number streams, which are associated with the risk of generating skewed, unreliable and invalid results.

II. Modelling Pitfalls

In simulation modelling, the term *iteration* refers to different combinations of parameter values. A single iteration is composed of a series of *replications*, their number determined by a sensitivity analysis, which differ in terms of their internal seed of randomness. Models that do not contain any internal stochastic are referred to as *deterministic*; however, these lie outside the focal point of this article. In *stochastic* modelling,

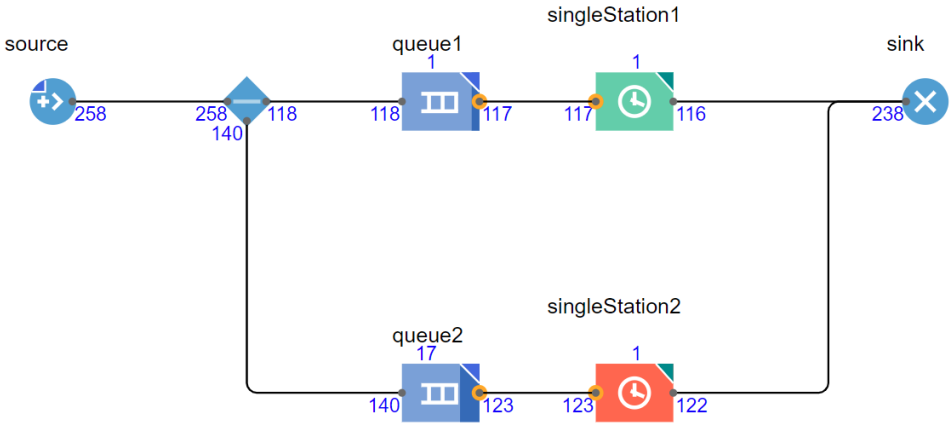

Figure 1: Examined process flow modeled with *AnyLogic*

a set of replications is computed in order to ascertain the influence of internal randomness on the final results. This methodology is referred to as the *Monte Carlo Experiment* (Shonkwiler and Mendivil, 2009).

In order to be able to compare two iterations with each other, it is vital that both are subject to the the same random stochastic behaviour. These can be e.ġ. machine maintenance occurrences, interarrival times of supply or interfering influences. If such random influences differ between iterations, the results, for instance differences in a Key Performance Indicator (KPI) of interest, may well be caused be the actual internal randomness. As such, the risk of attributing a result to a certain combination of parameters is considerable. Additionally, the adjustment of parameters (e. g. a different rule set or policy) may not correlate with any improvement of a KPI. The following conditions have been identified as potential causes of deviation in random behaviour in two or more iterations with the same seed of randomness:

1) When decisions in a model are stipulated by parameters, and a particular decision causes consumers of randomness, that is, every level of every variable, to require random numbers in a different sequence, e. g. job shop machine scheduling models
2) When parameters affect the number of consumers of randomness, e. g. Automated Guided Vehicle (AGV) fleet with an initially randomly distributed battery level
3) When parameters manipulate the model's initialisation sequence, e. g. data-driven models vs. applied random distributions

The following sections are intended to illustrate how these conditions can lead to different stochastic behaviour, even if the same seed of randomness is utilised. An example is provided to illustrate each item.

The first example presented is a basic process flow consisting of a source, two queues with subsequent servers and a sink, see Figure 1. The model's parameters were set as follows:

- Source, interarrival time:
  30 seconds, exponentially distributed
- Single stations 1 & 2
  process time: 1.5 - 2.5 minutes, uniformly distributed
  mean time between failures: 60 hours, exp. distributed
  mean time to repair: 15 hours, based on the Erlang distribution with $n = 2$
- There are two policies that decide whether an agent, for instance a customer, chooses the first or the second server. Policy 1 is a random-only decision with a 50 : 50 chance. With Policy 2, arriving agents will always choose the shorter queue. If both queues are of the same length, a random 50 : 50 chance is calculated.

If Policy 1 were to yield a higher throughput than the "smarter" Policy 2, it is deemed self-evident that the results were skewed by the difference in stochastic behaviour, caused by any one or combination of the above mentioned conditions. To examine this scenario, a Monte Carlo simulation trial run was performed on both iterations (Policy 1 and 2), each with 10^5 replications. The number of replications was determined by the results of a sensitivity analysis carried out at an earlier point. A different seed value was allocated to each replication, which provides a stream of random numbers for the consumers source, policy and both servers. Each simulated iteration covers a duration of 24 actual hours.

The expected results were that the policy which considers the length of the queues generated a greater throughput than expected. In order to ascertain that this is the case in every replication, a further analysis showed that in 14.8 % of replications, Policy 1 (random queue) lead to a higher throughput than Policy 2 (shortest queue). In order to investigate this further, the number of server maintenance occurrences was analysed and the findings showed that, in some cases, two iterations with the same random seed and stream were correlated with different numbers of maintenance occurrences per server. This seems to indicate that we are, in fact, comparing not only two different policies, but also two different number consumption patterns. Even though these differing patterns

are only obvious in the 14.8 % of the cases, it is suggested that this methodological shortcoming affects all cases. It has a certain influence on all the other scenarios, but in these cases the impact is less substantial and not noticeable when one only regards the number of maintenance occurrences. However, an assessment of the down-time duration is likely to detect stochastic differences in all replications. Focusing on those 14.8 % of cases where the throughput of Policy 1 (random queue) is higher, it is notable that the number of maintenance occurrences is lower than when applying Policy 2 with the same seed value.

Why does the randomness differ between two iterations with the same random seed? As mentioned above, the mid-square method generates random numbers in such a manner that each random number is based on the preceding one. Therefore, the sequence of numbers is considered to be predetermined. In the process flow described above, there are several consumers of random numbers. Depending on the chosen policy, all the consumers of random numbers (source, policy, servers) retrieve a random number whenever the stochastic influence necessitates it. For instance, Policy 1 (random queue) consumes a random number every time an agent needs to decide whether it joins Queue 1 or 2. In contrast, Policy 2 (shortest queue) only consumes a random number when both queues are of the same length, which only rarely occurs. Here, the sequence of random number consumers differs substantially and this leads to a different pattern of maintenance occurrences and process times at all subsequent processes.

The question that arises from these deliberations is how can researchers achieve the same stochastic influence for two different parameter settings, in this case iterations with the same random seed value. The solution appears to lie within the grasp of the researcher, who can separate all random occurrences within the model that are actually independent of each other. In the presented scenario, this refers to the interarrival time of the source, the random decision of the applied policy, the process times of both servers and their maintenance occurrences. More specifically, each random number consumer needs to retrieve a new random number from its own stream. If this is the case, the sequence of consumption will no longer influence the generated random numbers for the other consumers.

Keeping random streams separate from each other is also proposed to solve the issue addressed in condition 2) where parameters affect the consumed random numbers during the initialisation phase of a simulation model. A logistics scenario considering the number of AGVs that are involved in a specific material flow system lends itself to illustrate the case in point. If the AGV initially retrieves a random number to establish its initial battery level, the sequence is directly influenced by the number of AGVs. Consequently, this leads to very different random behaviour from the very beginning of the simulation. This unwanted deviation can be prevented by either allocating one specific stream of random numbers for each AGV or one stream that only provides the random initial battery levels.

Point 3 directly relates to the topic of model initialisation. Here, the issue arises when a model is capable of replicating both, evaluating actual historical data or applied random distributions. Both usually result in a different sequence in the initialisation phase and therefore in a different sequence of random numbers.

The conditions described apply intrinsically to simulation modelling in general, regardless of the software used. Depending on the software utilised for modelling, the described pitfalls must be tackled differently. *FlexSim*, for example, offers a total of 100 random streams by default. They can be allocated by only using digits from 1 to 100 as the respective parameters in any distribution function. Beyond that, *FlexSim* allows the user to create a new stream by using the command `getstream(current)`, where `current` refers to the individual consumer of random numbers whom a single stream is dedicated to commencing as of the first retrieval. In contrast, there is an unlimited number of seed values available in *AnyLogic*, which can be used to initialise an unlimited number of Java objects by using, for instance, `Random r = new Random(1);` for a seed value of 1. These objects can be used as a parameter in any distribution function instead of the default random generator of the model. *PlantSimulation*, on the other hand, designates an individual stream for each consumer automatically. Despite the advanced settings and suggestions that *PlantSimulation* offers, the developer requires a firm understanding of the design of their simulation environment and the requirements to maintain an empirically sound simulation when utilising random numbers. In summary, while not explicitly naming all common simulation tools, they all offer functions to maintain control over random streams.

III. Conclusion

Based on this author's previous reviews of the pertinent literature in this domain, it is suggested that few simulation studies tend to the exact assessment of random occurrences from one replication to the another. Modellers frequently rely on the software's default random generator, as its main purpose is to simply provide random numbers. Yet, depending on the design of the simulation study, default settings may not suffice when empirically reliable results require a more thorough approach.

In this article, a brief historical review of random number generators was provided and some substantial pitfalls in their application to simulation modelling were highlighted. In this particular context, it may appear as if some modellers do not separate their random streams as their design may require it. This oversight can lead to considerable differences in the behaviour of random number consumers as well as the results of the simulation, even if the same initial seed value is used. As such, assuming the same stochastic behaviour may lead to imprecise results and their comparison does not lead to reliable conclusions. However, for many cases this difference of the model's stochastics may go unnoticed or perhaps be attributed to the parameter settings. Finally, a number of suggestions were made to remedy these procedural shortcomings of replicating randomness.

References

Bennett, D. (1999), *Randomness*, Harvard University Press, ISBN: 978-0674107465.

Coveyou, R.R. (1969), 'Random number generation is too important to be left to chance', *Applied Probability and Monte Carlo Methods and modern aspects of dynamics*, pp. 70–111.

Dudewicz, E.J. and Dalal, S.R. (1971), 'Allocation of observations in ranking and selection with unequal variances', *Optimizing Methods in Statistics*, Elsevier, pp. 471–474, ISBN: 9780126045505.

Franklin, J.N. (1963), *Mathematics of Computation* (8), pp. 28–59.

Hull, T.E. and Dobell, A.R. (1962), 'Random Number Generators', *SIAM Review* 4 (3), pp. 230–254, ISSN: 0036-1445.

Kendall, M.G. and Smith, B.B. (1938), 'Randomness and Random Sampling Numbers', *Journal of the Royal Statistical Society* 101 (1), p. 147, ISSN: 09528385.

Knuth, D.E. (1968), *The Art of Computer Programming*, Stanford University: Addison-Wesley, ISBN: 0-201-89684-2.

LeCam, L. (1988), 'Interview', *More Mathematical People* (8), p. 174.

Lehmer, D.H. (1951), 'Mathematical methods in large-scale computing units', *Annu. Comput. Lab. Harvard University* 26, pp. 141–146.

Morgan, B.J.T. (1984), *Elements of simulation*, London and New York: Chapman and Hall, 351 pp., ISBN: 978-0412245909.

Peterson, I. (1998), *The Jungles of Randomness: A Mathematical Safari*, Penguin Books Ltd, ISBN: 978-0140271720.

Rand Corporation (2001), *A million random digits with 100,000 normal deviates*, Santa Monica, Calif.: Rand Corporation.

Shonkwiler, R.W. and Mendivil, F. (2009), *Explorations in Monte Carlo methods*, Undergraduate texts in mathematics, Dordrecht and Heidelberg: Springer, ISBN: 978-0-387-87837-9.

Thomson, W.E. (1959), 'Ernie - A Mathematical and Statistical Analysis', *Journal of the Royal Statistical Society. Series A (General)* 122 (3), p. 301, ISSN: 00359238.

Tu, S.-J. and Fischbach, E. (2005), 'A Study on the Randomness of the Digits of π', *International Journal of Modern Physics C* (16), pp. 281–294.

Von Neumann, J. (1951), 'Various Techniques Used in Connection with Random Digits', *Monte Carlo Method*, ed. by A.S. Householder, G.E. Forsythe and H.H. Germond, vol. 12, National Bureau of Standards Applied Mathematics Series, Washington, DC: US Government Printing Office, pp. 36–38.

Yoshizawa, Y., Kimura, H., Inoue, H., Fujita, K., Toyama, M. and Miyatake, O. (1999), 'Physical random numbers generated by radioactivity', *Journal of the Japanese Society of Computational Statistics* 12 (1), pp. 67–81, ISSN: 0915-2350.