NUMERICAL PRECISION AND TRAJECTORY DEGRADATION IN THE CHAOTIC SKEW TENT MAP

Alexandru George Vaduva[1], Adriana Vlad[2], Bogdan Badea[3]

The research investigates the degradation of the chaotic skew tent map dynamics, specifically, the impact of numerical precision on iterating the chaotic signal, which is commonly referred to as trajectory degradation. In the first part of the study, we determine the minimum precision necessary to iterate the chaotic signal K times without degradation and investigate possible optimized solutions to this problem.

In the second part, the experimental findings indicate a correlation between the control parameter of the skew tent map and the minimum number of bits required to perform the iterative calculations without experiencing trajectory degradation. We assess the trajectory degradation phenomenon under the scenario of double precision.

Keywords: numerical precision, tent map chaotic systems, trajectory degradation, arbitrary precision

1. Introduction

In the field of chaos-based applications, a chaotic signal's defining formula is initialized with an initial condition and then iterated step by step. This paper analyzes the effects of using different numerical precisions when iterating the chaotic skew tent map signal. The problem of accurately reproducing chaotic processes is well documented in literature [1–6] and a few specific solutions have been previously explored [7, 8]. This study focuses on the skew tent map chaotic signal and explores the connection between trajectory degradation and the control parameter p, as well determining the minimum numerical precision required to iterate a given number of times accurately.

[1] PhD Student, Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology Politehnica of Bucharest, Romania, e-mail: alexandru.george.vaduva@gmail.com

[2]Professor, Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology Politehnica of Bucharest, Romania; The Research Institute for Artificial Intelligence, Romanian Academy, Bucharest, Romania, e-mail: avlad@racai.ro, adriana_vlad@yahoo.com

[3] Collaborator, Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology Politehnica of Bucharest, Romania

The skew tent map signal is a chaotic system in discrete time with a one-dimensional structure, widely used in various fields such as [9–13]. The definition of the skew tent map signal under consideration is as follows:

$$z_{k+1} = \begin{cases} \frac{z_k}{p} & 0 \leq z_k < p \\ \frac{1-z_k}{1-p} & p \leq z_k \leq 1 \end{cases} \tag{1}$$

where p represents the control parameter defined in interval (0;1) \ 0.5. The generated values denoted as z_k, k denotes iteration, belong in the (0;1) domain. Given a fixed value for the control parameter p and initial conditions chosen uniformly in (0;1), the system 1 generates an ensemble of trajectories which represent the ergodic random process that is assigned to skew tent map function. Each trajectory is identified by its initial condition and represents a particular sample of the random process.

The first order probability law that governs this random process is a uniform law in the (0;1) range. Given an initial condition z_0 and a control parameter p, the value z_k at iteration k can be computed with the precision given by the data format used. However, the chosen precision can significantly alter a trajectory as rounding errors accumulate in the iteration process. Fig. 1 illustrates two trajectories computed using p=0.35 for the control parameter and starting from the same initial condition z_0=0.7, but iterated using two different data formats and thus with two numerical precisions.

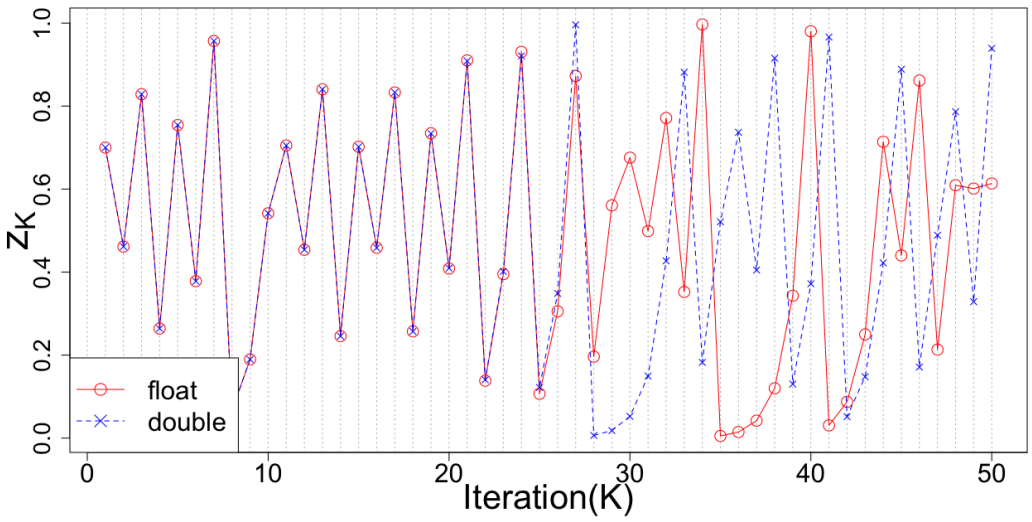

FIGURE 1. Iterating a tent map trajectory using an initial condition $z_0 = 0.7$ and control parameter p=0.35 with double and float precision.

The two trajectories seem to visually overlap for the first few iterations, but if they are compared numerically, they start to diverge sooner than it appears visually. In our study we define two trajectories as having diverged when the difference between their respective values at the same k iteration is greater than $\epsilon = 10^{-5}$. The chosen threshold ensures a balance between the visual differences when comparing two or more trajectories as shown in Fig. 1 and the numerical differences between trajectories, which we consider to be negligible below this value for practical purposes. In the current paper we want to evaluate the numerical precision required to iterate K times without being practically affected by the trajectory degradation phenomenon.

2. Background

The operands' data format precision is given by the used representation format. The IEEE 754 standard [14] is used for storing floating point numbers in the memory of modern computers. The most popular representation formats are float and double which use a total of 32 bits and 64 bits respectively. The number representations defined by the IEEE 754 standard are shown in Table 1.

In strongly typed programming languages, like C++, the type of a variable must be declared (e.g., float or double). In contrast, in loosely typed languages, such as Matlab or Python (which use double or float, respectively), the type of a variable is inferred. Not declaring the type of variables when working with chaotic signals can result in differing outcomes on different machines or even on the same machine.

In order to evaluate the minimum number of precision bits required to iterate without degradation K times, we have used an arbitrary precision library [15]. For the internal representation of numbers, this library employs a fixed number of 64 bits for the exponent and a variable number of bits for the mantissa, allowing for virtually arbitrary precision. Note that the highest precision defined by the IEEE 754 format, Binary256, cannot be used to iterate accurately for more than a few hundred iterations.

Table 1
IEEE 754 number representation formats.

Name	Popular name	Mantisa (bits)	Exponent (bits)
Binary16	Half precision	11	5
Binary32	Single precision	24	8
Binary64	Double precision	53	11
Binary128	Quadruple precision	113	15
Binary256	Octuple precision	237	19

With the help of [15] we can iterate any chaotic signal with sufficient precision and generate an arbitrary long trajectory for our chaotic signal without degradation, given a sufficient number of bits for the mantissa. However, this library has some drawbacks, such as increased memory usage and runtime that grows exponentially as precision increases.

The simulations performed for this study were done using an Intel Xeon W with 16 cores and 32 threads running at 3.5 GHz on system having 32 GB of RAM.

3. Methodology and experimental results

3.1. Algorithm for detecting trajectory degradation

In this study, our primary objective is to determine the minimum number of mantissa bits, b, that allow us to iterate K times without trajectory degradation. To do this, we start by generating an initial trajectory, $z_0^{4K}, z_1^{4K}, \ldots, z_{K-1}^{4K}$, using (1) and a given control parameter, p, an initial condition z_0, and $4K$ bits of precision for the iterative calculation. This trajectory serves as our reference trajectory. Empirically, we have found that using $4K$ bits of precision is well beyond what is necessary to iterate K times without degrading the trajectory.

We then iterate the same initial z_0 condition using M bits of precision, where $M < K$ and obtain a second trajectory with samples $z_0^M, z_1^M, \ldots, z_{K-1}^M$. If we compare the two trajectories sample by sample and $|z_i^{4K} - z_i^M| \leq \epsilon$ for all $i \in \{0, 1, \ldots K-1\}$, then we consider the 2 trajectories to be identical and iterating K times using M bits of precision starting from z_0 is done without degradation.

We define b as the value for which $|z_i^{4K} - z_i^b| \leq \epsilon$ for all $i \in \{1, 2, ..., K-1\}$ and $|z_{K-1}^{4K} - z_{K-1}^{b-1}| \geq \epsilon$ for at least one value of i in the same range $\{1, 2, ..., K-1\}$. Usually the error cascades and checking the previous conditions at $i = K-1$ is a strong condition. For a given p control parameter and a set of K iterations, the minimum number of mantissa bits, b, varies with different initial conditions z_0. For each combination of (p, K), we evaluate a value b for each initial condition out of all 1 000 000 and retain the largest value for b. The p control parameter can take any value between (0;1) \ 0.5. In this paper we analyzed the following values for p: {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.45, 0.55, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95}. Regarding the K number of iterations, *i.e.* the length of the trajectory unaffected by degradation, we studied the following values: {100, 200, 300, 500, 1000, 5000}.

In order to determine b for each combination of (p, K) values we perform the following:

(1) We choose the next unused (p, K) combination.
(2) We generate 1 000 000 z_0 initial conditions uniformly distributed in (0;1). For each initial condition we perform the following steps:
    (a) We compute the reference trajectory of the skew tent map signal from (1) using $4K$ mantissa bits of precision.

TABLE 2
Minimum number of mantissa bits used to iterate without trajectory degradation for a given (p, K) pair.

Control parameter p	Number of iterations K					
	100	200	300	500	1000	5000
0.05	91	153	170	257	439	1766
0.1	110	175	224	349	626	2666
0.15	120	200	260	412	773	3335
0.2	101	157	208	329	596	2701
0.25	129	221	305	496	927	4263
0.3	129	229	324	520	979	4578
0.35	130	231	324	529	1016	4824
0.45	123	225	325	526	1030	5025
0.55	123	225	326	529	1030	5025
0.65	130	229	331	536	1006	4810
0.7	131	233	324	509	977	4594
0.75	128	221	302	494	921	4273
0.8	125	212	283	455	847	3877
0.85	125	206	265	414	755	3364
0.9	110	174	226	346	618	2648
0.95	95	141	177	267	443	1737

(b) We determine b, the minimum number of precision bits for which the trajectory does not degrade as when compared to its reference trajectory.

(3) From all 1 000 000 b values for the current (p, K) pair, we retain the maximum value then return to step 1 until all pairs have been analyzed.

The results in Table 2 show that the necessary minimum precision to perform K iterations accurately varies with p. As a general rule, to avoid trajectory degradation, it is recommended to use at least K+100 bits of precision in the mantissa as this empirically determined precision covers all cases tested. If we want to use slightly less bits of precision, then Table 2 should be consulted.

According with Table 2, for a given control parameter p=0.55 and K=200 iterations we must use 225 bits for the mantissa in order to avoid the phenomenon of trajectory degradation. For the same p parameter but for K=5 000 iterations, we need 5 025 bits of precision for the mantissa to iterate correctly and avoid trajectory degradations.

Table 2 demonstrates that when the p control parameter is closer to the limits of (0;1), fewer precision bits are needed to perform K iterations without loss of precision compared to when p is closer to 0.5.

One way to optimize the above algorithm is to use a binary search between

values 1 and $4K$ for step 2b from above instead of looking at each value $i \in \{1, 2..., K-1\}$. This makes sense given that we did not encounter any repeating values / cyclic trajectories for the number of iterating steps we have considered. Also, once a trajectory has diverged from its reference counterpart at iteration $i < K-1$ the outcome of bouncing back to the reference trajectory is highly unlikely, especially at iteration j where $i < j < K-1$.

The histograms of b values for various $(p,\ K)$ combinations from Table 2 can be seen in Fig. 2. These histograms were created using the b values obtained in steps 1-3, which were computed using 1 000 000 initial conditions.

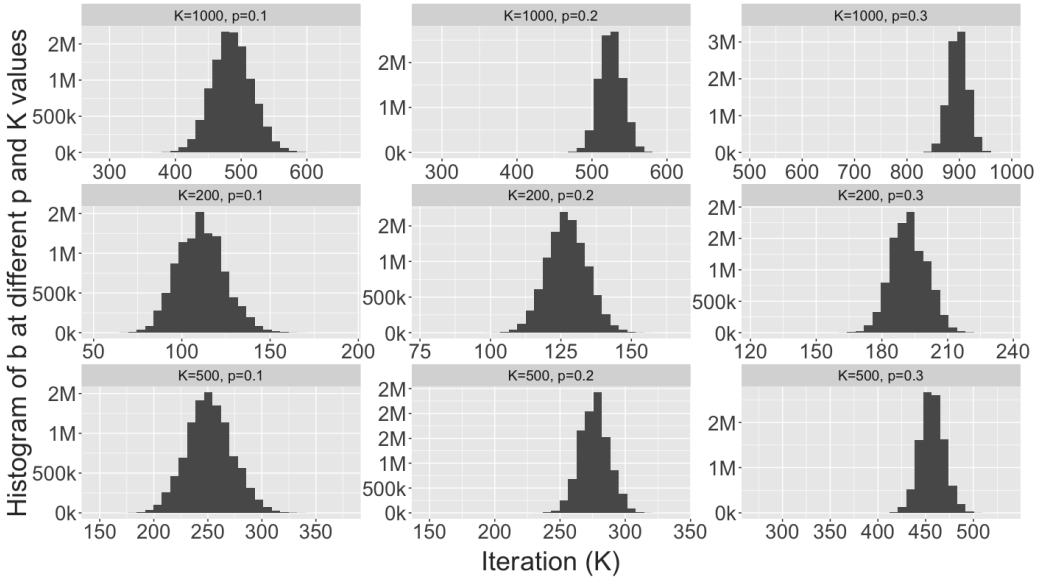

FIGURE 2. Histogram for b at different $(p,\ K)$ values.

3.2. Double precision vs extended arbitrary precision

The most used numerical precision in the field of chaotic signals is the *double precision* [16–19] and we want to evaluate the degradation of the skew tent map signal when iterated using double precision and with a variable control parameter p which takes values in the (0;1) \ 0.5 interval. We study the same values for the control parameter p proposed in the previous section: {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.45, 0.55, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95}. For all proposed values of p, we observed that the degradation happens before iteration K=300. In order to iterate without degradation for K=300 iterations, we used a precision of 512 bits for the mantissa, which is considerably higher than the minimum required precision to avoid degradation, as it can be verified in Table 2.

For evaluating the degradation of the skew tent map signal when using double precision, we applied the following steps:

a) We choose a value for the control parameter p.

b) We generate 1 000 000 z_0 initial conditions uniformly distributed in (0;1). For each initial condition we perform the following steps:

   i Compute the trajectory starting from z_0 seed by iterating (1) using double precision for K=300 iterations, resulting in $trajectory_{double\ precision}$.

   ii Compute a reference trajectory by iterating (1) starting from z_0 seed with an extended precision of 512 bits for K=300 iterations, resulting in $trajectory_{extended\ precision}$.

   iii We compare $trajectory_{double\ precision}$ with $trajectory_{extended\ precision}$ element-wise and check if the difference is larger than the established threshold ϵ. If the difference is larger than ϵ we store 1 and 0 otherwise. We obtain a sequence of 300 binary values with as many leading zeros as the number of iterations for which the two trajectories are overlapping, then followed by ones until iteration 300 as the trajectories have diverged.

c) For the current control parameter p, we count how many trajectories among all 1 000 000 have diverged from their corresponding reference trajectory at each $K \in \{0, 1, \ldots, 300\}$ and store these proportions. We plot the recorded proportions at each k for the respective p value in Fig. 3 and Fig. 4.

We repeat starting with step a) until all control parameters have been analyzed.

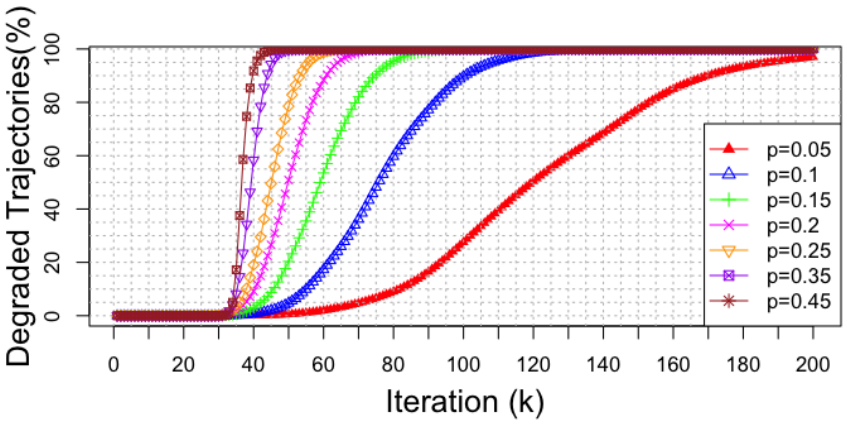

FIGURE 3. Trajectory degradation (%) at iteration K when using double precision for $p < 0.5$.

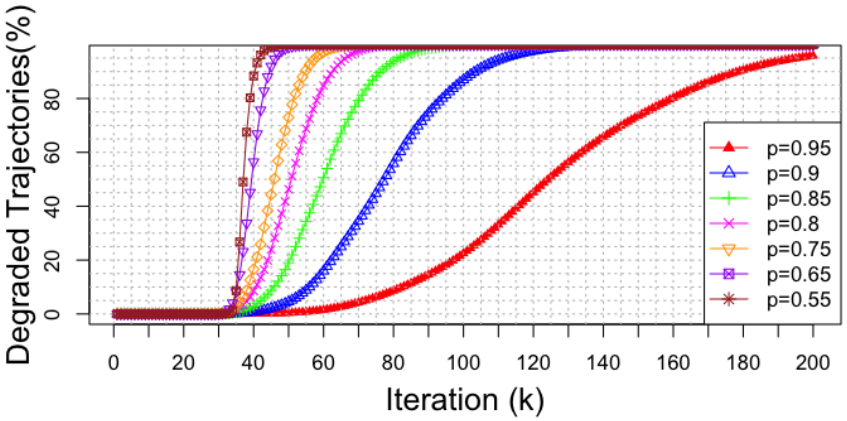

FIGURE 4. Trajectory degradation (%) at iteration K when using double precision for $p > 0.5$.

Fig. 3 and Fig. 4 show that for all p parameters tested there is no visible degradation until iteration $K = 30$. After this value the degradation manifests different depending on the control parameter p. For example, in Fig 3 for a control parameter $p = 0.1$ we can observe that after $k = 100$ iterations 80% of all tested trajectories have degraded.
We also observe that trajectories iterated using a control parameter p closer to 0.5 degrade much faster than trajectories iterated with a control parameter p that is much closer to the extremities of the (0;1) interval. This observation is closely related to the dependency between b and p observed in Table 2, where b represents the minimum number of mantissa bits required to iterate K times without precision loss and p is the control parameter for the skew tent map signal. All trajectories studied degraded after at most 200 iterations.

The potential use of this technique extends to several communication fields, especially cryptography, where chaotic system's dynamic state changes and/or control parameters are leveraged. Knowing the duration K in which trajectories remain unaffected by degradation can help identify the best moments to intervene with dynamic changes of the system [20, 21].

For the chaotic system under study, regardless of its parameter p and the initial conditions used, digital degradation when using the double data format occurs after approximately 30 iterations. This means that, for the first 30 iterations of the skew tent map signal, the first 5 decimals of the values computed by using double precision coincide with those computed theoretically.

4. Conclusions

We have proposed an algorithm for evaluating the *trajectory degradation* then applied it to the skew tent map chaotic signal described by (1) and we

have determined that, in order to iterate K times without encountering the trajectory degradation phenomenon, the rule of thumb proposed is to use $K+100$ bits of precision for the mantissa of the number representation, by using the arbitrary precision library. Table 2 offers a better result for the number of precision bits that should be used to avoid trajectory degradation, depending on the control parameter p of the skew tent map chaotic signal and the number of iterations K.

We have determined that the required precision to iterate K times without trajectory degradation varies with the control parameter p and the trajectories start to degrade after approximately $K = 30$ iterations. Moreover, trajectories iterated with a control parameter closer to the limits of (0;1) interval require using less bits of precision than trajectories iterated with a control parameter p which is closer to the middle of the (0;1) interval, for the same number of iterations K.

The findings from this research could be utilized in situations where chaotic signals are used to exchange shared secrets, or in scenarios where introducing randomness at optimal points can offset the degradation effect by altering the system's dynamic state [20, 21].

References

[1] *J. Oteo and J. Ros*, "Double precision errors in the logistic map: Statistical study and dynamical interpretation," *Physical Review E*, vol. 76, no. 3, p. 036214, 2007.

[2] *S.M. Hammel, J. A. Yorke and C. Grebogi*, "Do numerical orbits of chaotic dynamical processes represent true orbits?" *Journal of Complexity*, vol. 3, no. 2, pp. 136–145, 1987.

[3] *T. E. Nazaré, E. G. Nepomuceno, S.A. Martins, and D.N. Butusov*, "A note on the reproducibility of chaos simulation," *Entropy*, vol. 22, no. 9, p. 953, 2020.

[4] *S. Qin, and S. Liao*, "Influence of numerical noises on computer-generated simulation of spatio-temporal chaos," *Chaos, Solitons & Fractals*, vol. 136, p. 109790, 2020.

[5] *S. Li, G. Chen, and X. Mou*, "On the dynamical degradation of digital piecewise linear chaotic maps," *International Journal of Bifurcation and Chaos*, vol. 15, no. 10, pp. 3119–3151, 2005.

[6] *G. Alvarez, and S. Li*, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos*, vol. 16, pp. 2129–2151, Aug. 2006.

[7] *N. Nagaraj*, "The unreasonable effectiveness of the chaotic tent map in engineering applications," *Chaos Theory and Applications*, vol. 4, no. 4, pp. 197–204, 2022.

[8] *S. Zhou, X. Wang and Y. Zhang* , "Novel image encryption scheme based on chaotic signals with finite-precision error," *Information Sciences*, vol. 621, pp. 782–798, 2023.

[9] *M. Frunzete, L. Yu, J.-P. Barbot and A. Vlad.* "Compressive sensing matrix designed by tent map, for secure data transmission," in *Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2011*, IEEE, 2011, pp. 1–6.

[10] *M. Kennedy, R, Rovatti and G. Setti. Chaotic electronics in telecommunications.* CRC press, 2000.

[11] *D.M. Kato and M. Eisencraft.* "On the power spectral density of chaotic signals generated by skew tent maps," in *2007 International Symposium on Signals, Circuits and Systems*, IEEE, vol. 1, 2007, pp. 1–4.

[12] *C. Macovei, A. Vaduva, A. Vlad and M. Zamfir.* "A mean test on the autocorrelation function of a chaotic signal aiming to support the statistical independence sampling distance," in *2019 International Symposium on Signals, Circuits and Systems (ISSCS)*, IEEE, 2019, pp. 1–4.

[13] *A. Vlad, A. Luca, O. Hodea, and R. Tataru.* "Generating chaotic secure sequences using tent map and a running-key approach," *relation*, vol. 1, 2013.

[14] *D.G. Hough* "The IEEE standard 754: One for the history books," *Computer*, vol. 52, no. 12, pp. 109–112, 2019.

[15] *T. Granlund, Torbjrn. GNU MP 6.0 Multiple precision arithmetic library.* Samurai Media Limited, 2015.

[16] *K.J. Persohn, and R. Povinelli.* "Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floating-point representation," *Chaos, Solitons & Fractals*, vol. 45, no. 3, pp. 238-245, 2012.

[17] *L.G. Nardo, E.G. Nepomuceno, J. Arias-Garcia, and and D.N. Butusov.* "Image encryption using finite-precision error," *Chaos, Solitons & Fractals*, vol. 123, pp. 69-78, 2019.

[18] *G. Zhou, D. Zhang, Y. Liu, Y. Yuan, and Q. Liu.* "A novel image encryption algorithm based on chaos and Line map," *Neurocomputing*, vol. 169, pp. 150-157, 2015.

[19] *A. Luca, A. Vlad, B. Badea, and M. Frunzete.* "A study on statistical independence in the tent map," in *2009 International Symposium on Signals, Circuits and Systems*, IEEE, 2009, pp. 1–4.

[20] *O. Datcu, C. Macovei, and R. Hobincu.* "Chaos based cryptographic pseudo-random number generator template with dynamic state change," *Applied Sciences*, vol. 10, no. 2, 2020.

[21] *C. Fan and Q. Ding.* "Counteracting the dynamic degradation of high-dimensional digital chaotic systems via a stochastic jump mechanism," *Digital Signal Processing*, vol. 129, p. 103651, 2022.