Reservoir computing with a single time-delay autonomous Boolean node

Nicholas D. Haynes, Miguel C. Soriano, David P. Rosin, Ingo Fischer, and Daniel J. Gauthier

Reservoir computing with a single time-delay autonomous Boolean node

Nicholas D. Haynes,[1] Miguel C. Soriano,[2] David P. Rosin,[1] Ingo Fischer,[2] and Daniel J. Gauthier[1]

[1]*Department of Physics, Duke University, Durham, North Carolina, USA*
[2]*Instituto de Física Interdisciplinar y Sistemas Complejos, IFISC (UIB-CSIC),*
Campus Universitat de les Illes Balears, Palma de Mallorca E-07122, Spain

We demonstrate reservoir computing with a physical system using a single autonomous Boolean logic element with time-delay feedback. The system generates a chaotic transient with a window of consistency lasting between 30 and 300 ns, which we show is sufficient for reservoir computing. We then characterize the dependence of computational performance on system parameters to find the best operating point of the reservoir. When the best parameters are chosen, the reservoir is able to classify short input patterns with performance that decreases over time. In particular, we show that four distinct input patterns can be classified for 70 ns, even though the inputs are only provided to the reservoir for 7.5 ns.

Classical von Neumann machines cannot efficiently perform some tasks, such as pattern recognition and classification, that animal brains do with relative ease. This observation motivates the search for novel computational paradigms, particularly those with biological plausibility. One approach is reservoir computing (RC) [1, 2], a method of efficiently training recurrent neural networks to perform a given task. In conventional RC, computation is performed by a large recurrent network of nonlinear elements with arbitrary connection weights (the reservoir) [3, 4]. The reservoir performs a nonlinear transformation on a time-dependent input signal, mapping it onto a high-dimensional state space in which it is linearly separable from other inputs. During training, the weights of connections within the reservoir are kept fixed, and the weights of connections to an output layer are determined by linear regression with target outputs.

Remarkably, a large, complex network is not necessary for RC. In fact, any system that is able to map input states onto a high-dimensional state space and that fulfills three commonly cited sufficient conditions for RC [2] – separation of input states, generalization of similar inputs to similar outputs, and fading memory – can be used [5]. For example, previous experiments [5–10] have shown that a high-dimensional transient output can be generated with a single, physical nonlinear device that time-multiplexes "virtual nodes" using time-delay feedback. These experiments demonstrated that a RC with time-delay feedback can achieve excellent performance on time-dependent pattern recognition tasks such as speech recognition and time series prediction.

The success of these systems motivates the exploration of other physical time-delay systems that possess suitable characteristics for RC. In this vein, we introduce a RC consisting of a single autonomous Boolean logic element that executes the exclusive-or (XOR) function with two time-delay feedback lines. The state of an autonomous Boolean network with time-delay feedback depends on a continuous history of its past states, making the dynamics much more complex than in synchronous systems, which depend on only a discrete set of past values. This added complexity allows small, resource-efficient systems to perform the required high-dimensional transformation on inputs [11, 12].

Here, we demonstrate experimentally that this simple time-delay autonomous Boolean reservoir is capable of producing dynamics that fulfill the sufficient conditions for RC and can solve a classification task. We find that input signals produce chaotic transients with windows of consistency [13]. We then use a measure of effective dimensionality to show that optimal operating points of the reservoir can be found by varying the lengths of the time delays. Finally, we show that with an appropriate choice of parameters, the system is able to classify input signals with an accuracy that decreases with time.

We build the reservoir shown in Fig. 1 on an Altera Cyclone IV field-programmable gate array (FPGA), an integrated circuit comprised of $\sim 10^5$ distinct logic elements. Each logic element can be configured to compute an arbitrary Boolean function on several inputs and wired to other logic elements on the chip using a hardware description language. In addition, the FPGA architecture allows for asynchronous logic operations, and the inherent rise time of the logic elements can be used to realize physical time delays.
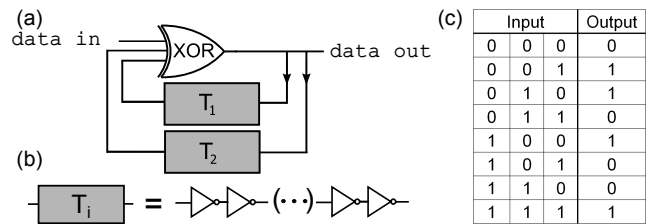

FIG. 1. (a) Circuit diagram of the single-node time-delay reservoir. (b) Time delays are realized as a cascade of pairs of inverter gates. (c) Truth table for the three-input XOR gate. Two stable fixed points exist – when all inputs are 0 (1), the output is 0 (1).

The XOR operation is chosen as the Boolean node for several reasons. First, it exhibits maximum Boolean sensitivity – when any of the inputs to a XOR gate are flipped, the output is also flipped (see Fig. 1c), which produces dynamics with maximal spreading with respect to the input states. Second, the XOR operation has a

balanced output, meaning that, for the 2^n possible combinations of n-bit input patterns, half result in 0 and half result in 1. Choosing a balanced Boolean function prevents the system from being biased toward 0 or 1 over time.

We realize the physical time-delay feedback lines by exploiting the finite response time of the FPGA's logic elements. Delay line 1 (2) is constructed by wiring N_1 (N_2) pairs of inverters in series; we call a pair of inverters a *delay element*. A pair of inverters is used as a delay element instead of a single copier to correct for an observed asymmetry between the rise and fall times of the logic elements [14]. When the lines are implemented on the FPGA, we observe a certain degree of heterogeneity in the delays due to physical imperfections. To reduce the effect of the heterogeneity as much as possible, we manually fix each of the delay lines to specific logic elements next to each other on the FPGA, rather than allowing them to be automatically placed by a compiler.

The experimentally measured delay times T_1 (T_2) resulting from N_1 (N_2) between 7 and 20 delay elements are listed in Tab. 1. The average delay time due to a single delay element is 0.59 ns, but the variation between elements in the same line is large, ranging between 0.43 ns and 0.99 ns. By manually fixing the lines, however, much of the heterogeneity between the two lines is eliminated – the difference between T_1 and T_2 is, on average, 0.03 ns, and is never more than 0.1 ns.

TABLE I. Measured delay times for each delay line.

N_1, N_2	T_1 (ns)	T_2 (ns)	N_1, N_2	T_1 (ns)	T_2 (ns)
7	3.76	3.75	14	7.94	7.92
8	4.31	4.31	15	8.44	8.41
9	5.30	5.25	16	8.99	8.96
10	5.79	5.79	17	9.75	9.76
11	6.35	6.44	18	10.30	10.25
12	6.79	6.89	19	10.74	10.76
13	7.50	7.49	20	11.42	11.38

To demonstrate that this system fulfills the sufficient conditions for RC described above, we observe the transient response of the reservoir to different inputs. Input data are encoded in words of bits that are supplied to the reservoir in the form of sequential strings of little-endian (least-significant bit first), non-return-to-zero digital voltages at 400 MHz. After a word is completed, the `data in` wire of the reservoir is tied to 0. Similar initial conditions are assured by allowing the system to settle into the fixed point XOR$(0, 0, 0) = 0$ before inputting data. A header to the input is necessary to distinguish patterns with leading or trailing zeros – to a reservoir sitting in the fixed point XOR$(0, 0, 0) = 0$, the word 0100, for example, appears to be just a time-shifted version of 0010. Including a one-bit header changes 0100 to 01001 and 0010 to 00101, which appear as distinct inputs.

The circuit in Fig. 1a is predicted [11, 12, 15] to have aperiodic behavior when the time delays are incommensurate. Indeed, when data is fed to the experimental reservoir, a long, complex transient is produced. The first 100 ns of four complex transients captured by a high-speed oscilloscope is shown in Fig. 2 for $N_1 = 17$, $N_2 = 18$ elements. Each of the transients is generated by a unique 2-bit input word. Clear Boolean-like transitions between high and low voltages can be observed, albeit with a finite slew rate. By direct inspection, it is clear that the output patterns resulting from each of the distinct inputs are unique. We find that a simple event-based model including a low-pass filter to remove short pulses can reproduce the observed dynamics [15, 16], but it is not described further here.
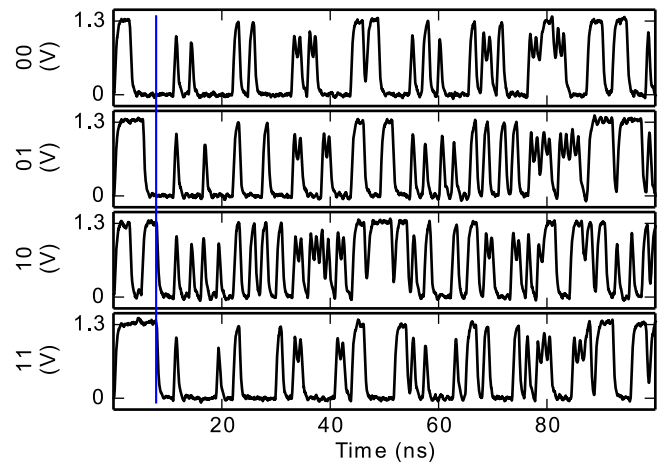

FIG. 2. (color online) Experimentally observed transient dynamics of four unique input patterns (noted on the y axes), generated by a reservoir with $N_1 = 17$, $N_2 = 18$. Solid blue (dark gray) vertical line marks $t = 7.5$ ns, where the input word ends.

In addition to output states that are unique to the inputs, reliable computation requires reproducibility of the outputs, *i.e.*, reservoir responses to the input signals must be consistent [13]. Starting with the reservoir in its fixed point, we repeatedly observe transients by providing a word to the reservoir, recording the output time series for 1 μs, and then forcing the system back into its fixed point before repeating the experiment.

To quantify the consistency of reservoir states, we define the measure of the output state space as the Boolean distance

$$d_{i,j}(t) = \frac{1}{\tau} \int_t^{t+\tau} x_i(t') \oplus x_j(t') dt', \quad (1)$$

where $x_i(t) \in \{0, 1\}$ is the Booleanized time-dependent output state corresponding to input i. We inject the four 2-bit input patterns ($i \in \{00, 01, 10, 11\}$) into a reservoir with $N_1 = 8$, $N_2 = 11$ elements. The average distances between the output time series from each of the input words is shown in Fig. 3. Consistency can be ob-

served by comparing the distances $d_{i,i}(t)$ between transients generated by the same inputs with those generated by different inputs ($d_{i,j}(t)$, $i \neq j$). For each i, $d_{i,i}(t = 0)$ is significantly smaller than $d_{i,j}(t = 0)$, but increases exponentially until it converges with $d_{i,j}(t)$. The window of exponential divergence in $d_{i,i}(t)$, a signature of a chaotic transient, is indicated by dashed black lines in Fig. 3. The slopes of these lines, which indicate how quickly $d_{i,i}(t)$ diverges, correspond to the local Lyapunov exponents of the system.
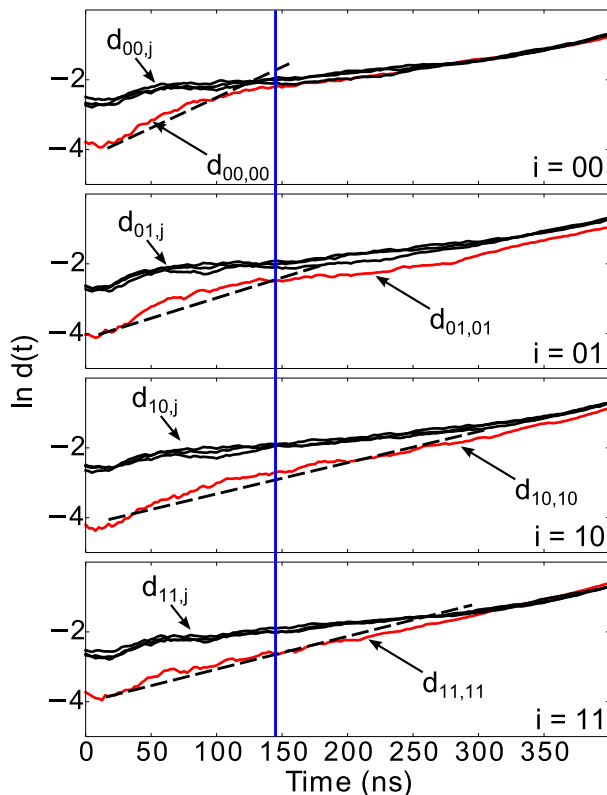

FIG. 3. (color online) Boolean distances ($\tau = 100$ ns) $d_{i,i}$ and $d_{i,j}$ ($i \neq j$) averaged over 50 output time series from a reservoir with $N_1 = 8$, $N_2 = 11$ elements for four distinct inputs. Solid blue (dark gray) vertical line represents the end of the consistency window, beyond which states are not distinguishable.

The minimum time to the convergence of $d_{i,i}(t)$ with $d_{i,j}(t)$ in Fig. 3 is approximately 145 ns, and thus the transient $x_i(t)$ due to input i is distinguishable from $x_j(t)$ for 145 ns. We call this period of time the consistency window. For T_1, T_2 between 3.75 and 11.42 ns (N_1, N_2 between 7 and 20 elements), we find consistency windows between 30 ns and 300 ns. The existence of the consistency window suggests that the reservoir fulfills the separation and generalization requirements introduced above.

To quantify the ability of the reservoir to perform effective computation and characterize its performance as a function of the size of the delays, we consider the measures known as kernel quality and generalization ability

[17, 18]. Once the consistency window for particular values of N_1 and N_2 is measured, the $m \times m$ state matrix M_K is created using the following procedure. Data is collected from the reservoir by sampling the `data out` wire at 400 MHz to RAM blocks located on the FPGA. The size m of the matrix is defined as the number of samples that can be collected within the consistency window ($12 \leq m \leq 120$ samples for consistency windows of $30 \leq L \leq 300$ ns). The state matrix M_K is then formed by collecting m samples from the resulting time series of m distinct inputs. The normalized kernel quality $K = \frac{1}{m}\text{rank } M_K$ is defined as the rank of the normalized state matrix. A system with perfect separation will produce linearly independent outputs for all possible input states, *i.e.*, $K = 1$.

The normalized generalization ability $\Gamma = \frac{1}{m}\text{rank } M_\Gamma$ is calculated using the same procedure, except that M_Γ is formed by m outputs from m distinct inputs, each of which is preceded by the same string of constant bits. A system with perfect generalization has $\Gamma = 1/m$, meaning that perturbations to the constant string of input bits do not cause a distinguishable change to the output.

An effective reservoir simultaneously maximizes K and minimizes Γ, *i.e.*, maximizes their difference $K - \Gamma \equiv \Delta$. Thus, $-1 < \Delta < 1$ is interpreted as the fraction of L that is useful for computation. A perfect reservoir has $\Delta = 1 - 1/m$, indicating that almost the entire consistency window provides useful information about the input, and $\Delta \leq 0$ represents a reservoir that fails to perform any useful information processing. We can therefore define an effective computational dimensionality $D = L\Delta$. Thus, the best reservoir is the one that maximizes D, which is bounded above by the length of the consistency window.
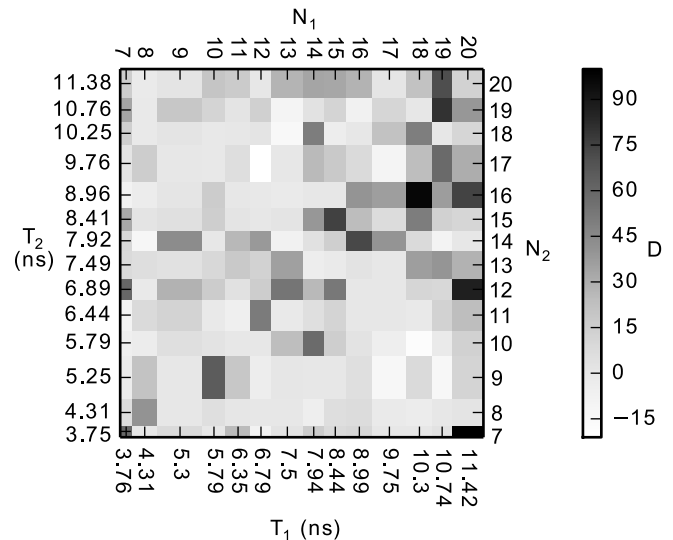

FIG. 4. Dependence of effective computational dimensionality D on the length of the delay lines. The tiles are different sizes due to the heterogeneity in the size of each delay element.

We examine the effect of T_1 and T_2 on reservoir performance by measuring D for different N_1 and N_2, shown

in Fig. 4. The lack of symmetry about the diagonal suggests that the heterogeneity in delay times between lines 1 and 2, seen in Tab. 1, plays a significant role in determining the dynamics, and that the performance of the reservoir is highly dependent on the ratio T_1/T_2. This result is consistent with [11, 19], where it was shown that T_1/T_2 determines an upper bound on the period length of possible dynamics, giving rise to windows with extremely long periods. The parameters that maximize D are found to be $N_1 = 20$, $N_2 = 7$.

Using the parameters $N_1 = 20, N_2 = 7$ that maximize reservoir performance, we now demonstrate the reservoir's ability to identify input patterns. To train the reservoir, we collect the dynamics for the 2^n possible n-bit input patterns by saving S samples of the output to on-chip RAM blocks at 400 MHz. Once the samples $x(t_0), x(t_1), ...x(t_{S-1})$ have been collected, they are transferred to a PC and used to form linear combinations

$$C_i(t_j) = \sum_{k=0}^{W-1} w_k^{ij} x(t_{j+k}) \qquad (2)$$

for each word $i = 0, 1, ..., 2^n - 1$ beginning at each sample time $t_j = t_0, t_1, ...t_{S-W-1}$.

The linear combinations $C_i(t_j)$ are called classifiers, and the weights w_k^{ij} that define them are computed by linear regression. The length of the classification window W defines the number of samples used to compute each classifier. The target of classifier C_i is 1 for the output pattern corresponding to input i and -1 for all other patterns. Thus, when additional time series are provided to the 2^n classifiers, each computes a scalar "score" for every sample point. We regard successful classification as an event when the classifier with the largest score is the classifier corresponding to the actual input.
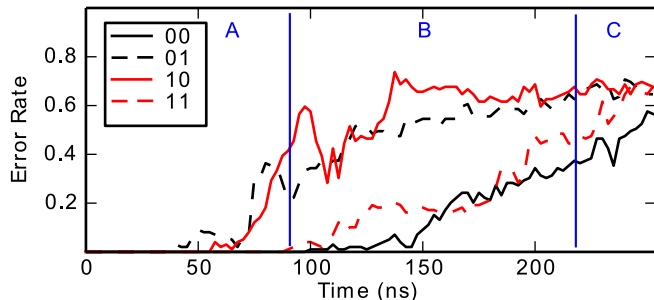

FIG. 5. (color online) Word recognition error rate for 2-bit input sequences with $N_1 = 20$, $N_2 = 7$ elements. Solid blue (dark gray) vertical lines separate training regions A, B, and C, defined in the text.

We present the classification performance of the reservoir in Fig. 5, which also demonstrates the effect of fading memory in the system. Error rates for each input word with $n = 2$ bits, obtained by testing the classifiers with 100 time series for each input, are shown in Fig. 5 for $N_1 = 20$, $N_2 = 7$ elements (the optimal parameters in Fig. 4). A time series length of $S = 200$ samples (500

ns) is collected, and a classification window of $W = 50$ samples (125 ns) is used. The consistency window for this system is measured to be 215 ns. Thus, there are three training regions. For classifiers $C_i(t_j)$ that begin at times t_j less than 90 ns (region A), the entire window of classification lies within the region of consistency. Classifiers that begin at times greater than 215 ns (region C) are trained with only samples from outside of the consistency window. Classifiers starting between 90 and 215 ns (region B) are trained with a mix of data from inside and outside the consistency window.

The classification performance is the strongest in region A, with the error rate remaining below 10% for approximately 70 ns, despite the input word being only 7.5 ns. In other words, the classifiers are able to faithfully reconstruct the reservoir's input state with reasonable accuracy even after the reservoir was no longer receiving information. Performance in region C plateaus at an error rate of 0.75, i.e., chance, meaning that no useful information about the input remains in the output after the consistency window. Additionally, a rise in classification error can be observed throughout region B, indicating that the inclusion of data from outside the consistency window decreases performance. We observe a similar trend for words of $n = 3$ and $n = 4$ bits – the error rate is lowest in region A, but gradually rises throughout region B to a plateau of $1 - 2^{-n}$ in region C. The error rate at a given time is generally higher for larger words, and the 32 possible words for $n = 5$ bits cannot be accurately classified at any time. This behavior is similar to that found in a reservoir that used electrical recordings from neurons of the primary visual cortex of an anesthetized cat [20].

We conclude that, despite its simplicity, the single time-delay feedback XOR reservoir produces dynamics suitable for RC. In addition, the reservoir is shown to perform a pattern recognition task. The complexity of the dynamics produced by the relatively simple system described in this article suggest that FPGAs are an exceptionally versatile platform for realizing physical RCs.

Our results point to several future directions. For example, it is possible to build larger Boolean networks on FPGAs [14, 21, 22], which will allow for improved computation in two ways. First, recent work studying the computational ability of larger random Boolean networks without time delays [23, 24] motivates speculation that the performance of the number classification task described above can be improved with additional nodes. Second, the ability to run many parallel input nodes will allow a mix of spatial and temporal computation that could be attractive for speech and image recognition. The ability to train the reservoir using digital logic on the FGPA, instead of transmitting data to an external PC for post-processing, could also speed up training. As a result, the simple system studied here will be crucial to understanding the ability of much larger systems to process information.

[1] H. Jaeger, German National Research Center for Information Technology GMD Technical Report **148**, 34 (2001).

[2] W. Maass, T. Natschläger, and H. Markram, Neural Comput. **14**, 2531 (2002).

[3] M. Lukoševičius and H. Jaeger, Computer Science Review **3**, 127 (2009).

[4] B. Hammer, B. Schrauwen, and J. J. Steil, in *ESANN* (2009).

[5] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Nat. Commun. **2**, 468 (2011).

[6] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, Opt. Express **20**, 3241 (2012).

[7] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, Sci. Rep. **2** (2012).

[8] M. C. Soriano, S. Ortín, D. Brunner, L. Larger, C. R. Mirasso, I. Fischer, and L. Pesquera, Opt. Express **21**, 12 (2013).

[9] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, Nat. Commun. **4**, 1364 (2013).

[10] A. Dejonckheere, F. Duport, A. Smerieri, L. Fang, J. Oudar, M. Haelterman, and S. Massar, Opt. Express **22**, 10868 (2014).

[11] M. Ghil and A. Mullhaupt, J. Stat. Phys. **41**, 125 (1985).

[12] M. Ghil, I. Zaliapin, and B. Coluzzi, Physica D **237**, 2967 (2008).

[13] A. Uchida, R. McAllister, and R. Roy, Phys. Rev. Lett. **93**, 244102 (2004).

[14] D. P. Rosin, D. Rontani, D. J. Gauthier, and E. Schöll, Chaos **23**, 025102 (2013).

[15] H. L. D. d. S. Cavalcante, D. J. Gauthier, J. E. S. Socolar, and R. Zhang, Philos. Tr. R. Soc. A **368**, 495 (2010).

[16] R. Zhang, H. L. D. d. S. Cavalcante, Z. Gao, D. J. Gauthier, J. E. S. Socolar, M. M. Adams, and D. P. Lathrop, Phys. Rev. E **80**, 045202 (2009).

[17] R. Legenstein and W. Maass, Neural Networks **20**, 323 (2007).

[18] L. Büsing, B. Schrauwen, and R. Legenstein, Neural Comput. **22**, 1272 (2010).

[19] D. Dee and M. Ghil, SIAM J. Appl. Math. **44**, 111 (1984).

[20] D. Nikolić, S. Häusler, W. Singer, and W. Maass, PLoS Biol. **7**, e1000260 (2009).

[21] D. P. Rosin, D. Rontani, and D. J. Gauthier, Phys. Rev. E **89**, 042907 (2014).

[22] D. P. Rosin, D. Rontani, N. D. Haynes, E. Schöll, and D. J. Gauthier, Phys. Rev. E **90**, 030902 (2014).

[23] D. Snyder, A. Goudarzi, and C. Teuscher, in *Artificial Life*, Vol. 13 (2012) pp. 259–266.

[24] D. Snyder, A. Goudarzi, and C. Teuscher, Phys. Rev. E **87**, 042808 (2013).