Dynamical Behavior of Cellular Automata

Miles Kenyon and Matthew Driscoll

December 4, 2009

1 Introduction

Cellular Automata are simple computational models which can be leveraged to model a wide variety of dynamical systems. Composed of a lattice of discrete cells that take finite number of states based on previous iterations these models differ greatly from dynamic systems that vary continuously in space or time. However given their ability to model many of continuous systems, it could be postulated that certain cellular automata exhibit chaotic dynamics. This paper examines the the chaotic analysis of a specific class of Cellular Automata, Elementary Cellular Automata (ECA), using the Lyapunov Exponent as our primary measure of chaos. The Lyapunov Exponent for these cellular automata does not have a standardized definition, given the unique nature of this model and its relative youth as an area of study. We focused on two methods used to calculate the Lyapunov Exponent, both of which generally define the Lyapunov Exponent to be the rate of error propagation in time, yet differ in specific method. Given these two theoretical frameworks we stove to create an efficient numerical application to calculate the Lyapunov Exponent of ECA of arbitrary rule number.

2 Elementary Cellular Automata

Cellular Automata are computational models comprised of a set of cells which evolve in discrete time steps through a finite number of states governed by the states of neighboring cells. For simplicity's sake we only analyzed a subset of this class of models, referred to as Elementary Cellular Automata. This subset is composed of one dimensional cellular automata where the state of a given cell is governed only by the previous states of the cell itself and its two immediate neighbors, further we will restrict the number of accessible states to 2. This results in a set of 256 possible rules, that is 256 distinct manners in which cells evolve.

Though this class seems arbitrarily limited, extremely intricate dynamic behavior can occur. Steven Wolfram studied cellular automata extensively and developed a system of classes to describe the dynamics of elementary cellular automata: [3]

I Evolves to homogeneous state.

II Evolves to simple, separated periodic structures.

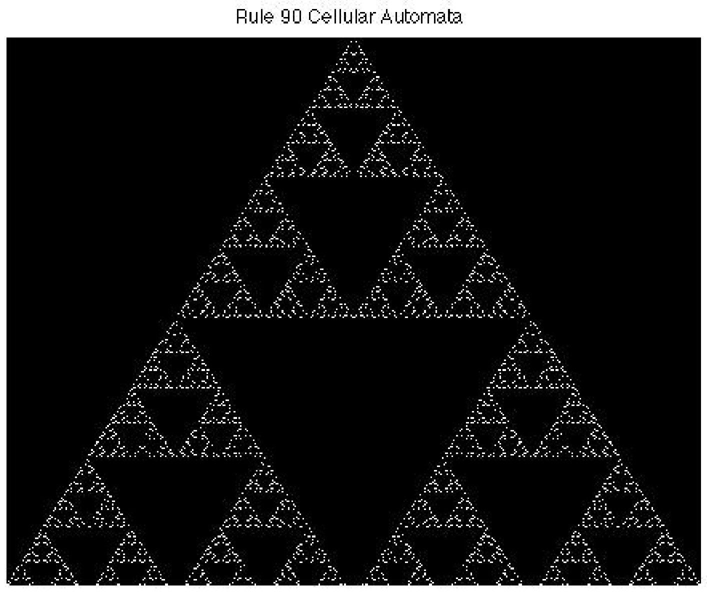
III Yields chaotic aperiodic patterns.

IV Yields chaotic pattern of localized structure.

Rules 90, 30, and 110 demonstrate the behavior of classes 2, 3, and 4 respectively. We will use these rules as examples of some of the types of behavior we set out to classify and analyze.

2.1 Rule 90

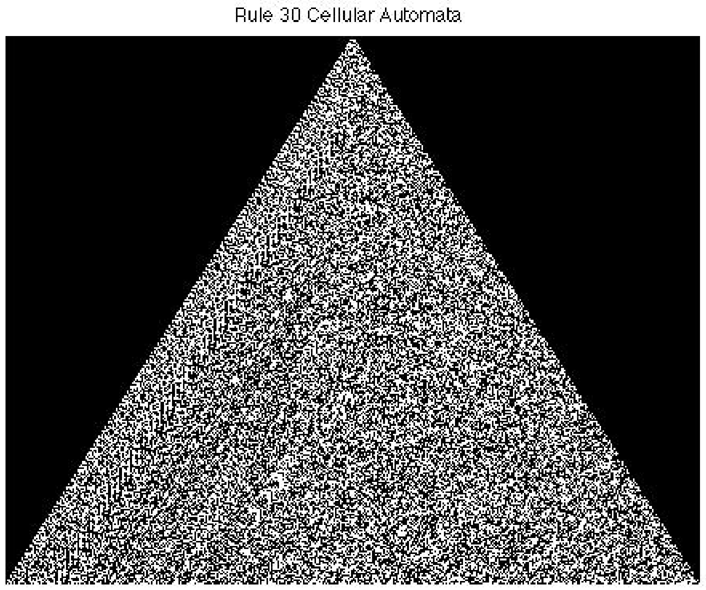
Rule 90 generates the Sierpinski Gasket, that is a fractal composed of self similar equilateral triangles. The following image was generated using a single "on" cell as an initial state for rule 90 by our vectorized ECA code:



Despite the highly regular structure, this complexity of the pattern seems to lend to a high level of error propagation under perturbation, regulated structure might then not be directly related to chaotic behavior.

2.2 Rule 30

Rule 30 seems to have little or no structure on initial inspection, this is consistent with class III cellular automata which ought to yeild “chaotic, aperiodic patterns” The following figure, also generated with our vectorized ECA code, illustrates the disordered nature of Rule 30 and Class III ECA.

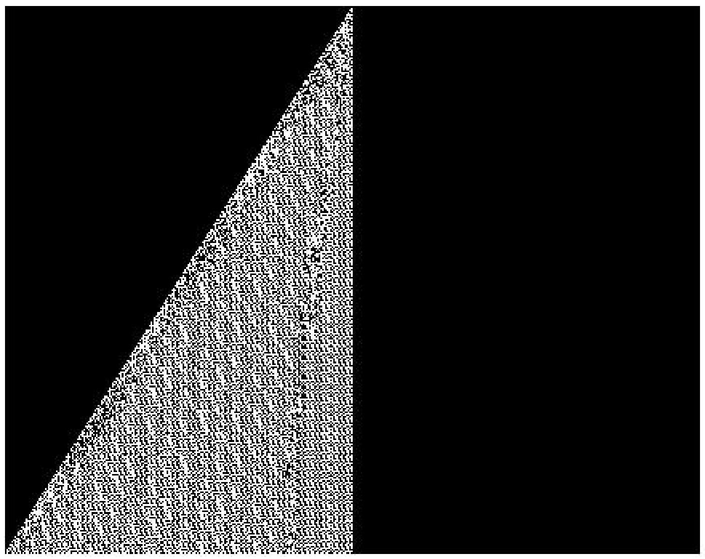


Ultimately we can confirm the chaotic nature of this rule with both of our methods of calculation.

2.3 Rule 110

Rule 110 is an incredibly interesting example of the wildly complicated dynamics that can be found in ECA. Rule 110 is Class IV, and therefore characterized by randomized areas of local structure. This specific Rule has further been proven to be Turing Complete, that is capable of universal computation, any computable operation can be encoded in a set of initial conditions.

Rule 110 Cellular Automata



We confirm that this rule is also chaotic in general structure, and that error propagates quickly.

3 Shreshevsky Lyapunov Exponent

3.1 Theory

To begin the discussion of Shereshevsky's development of Lyapunov exponents for ECA, we must fix a mathematical framework for discussion. Consider the set $S = \{0, 1\}$ of possible states that each cell in the ECA may take on. We let $X = S^{\mathbb{Z}}$ be the set al all bi-infinite sequences of 0 and 1, and call $x \in X$ a configuration, where $x = (x_i)_{i \in \mathbb{Z}}$. X along with the product topology is the configuration space for the CA (similarly, Tisseur develops a natural metric for X [?]), and we define a function $F : S^3 \mapsto S$ to be the rule for state switching for the given ECA. Finally, define a function $f : X \mapsto X$ by

$$(f x_i) = F(x_{i-1}, x_i, x_{i+1}) \qquad [1]$$

The ECA can then be considered as an ordered double (f, X)

Shereshevsky then proceeds to define two sets, $W_s^+(x)$ and $W_s^-(x)$, where $W_s^+(x) = \{y \in X : y_i = x_i \forall i \geq s\}$. Intuitively, if we fix a configuration x, $W_s^+(x)$ is the set of all configurations in X that are identical to x to the right of and including the s^{th} cell. $W_s^-(x)$ is defined similarly as the set of configurations in X that are identical to the left of the s^{th} cell, inclusively. He then goes on to define

$$\tilde{\Lambda}_n^+(x) = min\{s \geq 0 : f^n(W_0^+(x)) \subset W_s^+(f^n x)\}$$

$$\tilde{\Lambda}_n^-(x) = min\{s \geq 0 : f^n(W_0^-(x)) \subset W_s^-(f^n x)\}$$

$W_0^+(x)$ and $W_0^-(x)$ can be thought of as the set of all perturbations of x to the left and right of x_0, respectively. $\tilde{\Lambda}_n^+(x)$ then measures the distance that the perturbations propagate through x after n iterations of f in a manner that will be explained when our methods of numerical calculations are outlined. It may seem natural to take the limit of $\tilde{\Lambda}_n^+(x)$ divided by the number of iterations of f on x to be the average rate of propagation of the perturbations through x. However, we run into difficulties when considering the fact that such propagations may not travel the same distance if their front is located elsewhere than x_0. By introducing the shift operator τ we may construct a Lyapunov exponent of x that does not depend on where we perturb x. If we let $(\tau^j x)_i = x_{i+j}$, then shift invariance is introduced to the exponent by setting

$$\Lambda_n^+(x) = \max_{j \in \mathbb{Z}} \tilde{\Lambda}_n^+(\tau^j x) \qquad \Lambda_n^-(x) = \max_{j \in \mathbb{Z}} \tilde{\Lambda}_n^-(\tau^j x)$$

Finally, Shereshevsky gives the following theorem:
Theorem: The definition of Lyapunov exponents (Shereshevsky, 1992). *Fix a probability measure μ on the configuration space X. There exists a set $G \subset X$, where $\mu(G) = 1$, such that for every $x \in G$ the limits*

$$\lambda^+(x) = lim_{n\to\infty} \frac{1}{n} \Lambda_n^+(x) \qquad \lambda^-(x) = lim_{n\to\infty} \frac{1}{n} \Lambda_n^-(x)$$

exist. The functions $\lambda^+(x), \lambda^-(x)$ are called the left and right Lyapunov exponents of the cellular automaton (f,X) at x

[1]

Shereshevsky includes μ in order to prove the existence of the above limits, but makes no reference to the nature of μ other than it's being a probability measure. For the purposes of our calculations, all x that we measured had a finite number of ones in the initial configurations, and we postulate that such a condition generates the subset G of X for which the Theorem holds for ECA, which fits with the μ set forth in [2]. Shereshevsky extends his discussion of right and left Lyapunov exponents for CA by specifying conditions on μ for which λ^+ and λ^- are constant for almost every x. [1]

3.2 Numerical Estimation and Procedure

In order to numerically estimate λ^+ and λ^- a few immediate simplifications must be made in the calculation of $\tilde{\Lambda}^{\pm}$, namely:

1. Computation must occur on a finite grid rather than across an infinite state space

2. As a consequence of (1), a finite subset of $W_0^{\pm}$ must be used in the calculation of $\tilde{\Lambda}^{\pm}$

Condition (1) is supplemented by the provision that the boundary of the space be modeled as cells that are always zero. Calculations are aided by the fact that, given a configuration comprising a finite subsequence of ones and zeros within a bi-infinite sequence of zeros, an ECA will never grow the subsequence by more than one cell in the positive and negative directions upon successive iterations. With that in mind, we may initialize the calculation space as a finite grid of cells the width of the subsequence plus twice the number of planned iterations, and height set as the number of iterations (so that the horizontal axis can be thought of as a space dimension of the grid, and the vertical axis the time dimension. For convention's sake, we take the top row of the space to be time zero, and the middle of the original subsequence to be the spatial origin). If $x_{\pm i}$ are the sites of the cells on the boundary of the space for any time t, we perform our calculations is such a way that x_{i+1} and x_{-i-1} are zeros.

Condition (2) is employed by randomly generating perturbations of finite length on a fixed subsequence. Iterating the ECA on such perturbations gives a finite subset of $f^n(W_0^{\pm}(x))$. For convenience's sake, we may assume that $f^n(x) \in f^n(W_0^{\pm}(x)$ so that to calculate s such that $f^n(W_0^{\pm}(x)) \subset W_s^{\pm}(f^n x)$, we do the following (method outlined for $\tilde{Lambda}^+(x)$):

1. Calculate $f^n(W_0^+(x))$ and choose some element to be $f^n(x)$.

2. Begin at the right-boundary of the state space at time n, and step leftward.

3. Note the first position $i \geq 0$ where $(x')_i \neq (x)_i$ and $x' \in f^n(W_0^+(x))$.

Taking the maximum of all such i and adding 1 is then the s desired. This can be visualized as the first cell in which, after iterating all perturbations n times, all successive cells in each element of $f^n(W_0^+(x))$ are equal. By taking this cell's distance from $f(x_0)$, we have measured how far the right-moving "perturbation front" has traveled after n iterates. To calculate $\tilde{\Lambda}^-$ a similar method is employed, but with the left-moving perturbation front $f^n(W_0^-(x))$, and counting the cell differences from the left-hand boundary.

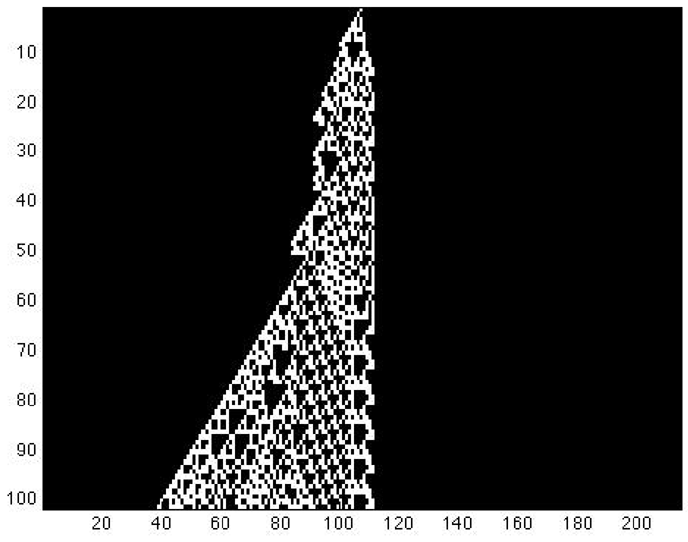
To introduce τ invariance and thus calculate $\Lambda_n^{\pm}(x)$, we repeat this process for the front located at each position x_i within the original finite subsequence. Finally, we sample our estimations of $\Lambda_n^{\pm}(x)$ for a fixed number of n and plot these against n in $\mathbb{R}^2$. We take the slope of the regression curve fit to these data points to be our final estimation of $\lambda^{\pm}$. Future research should consider further investigation into the statistical properties of ECA so that we may better estimate the number of needed perturbations and the size of the finite subsequences for our calculations, as well as how many iterations and at what sizes we should sample $\Lambda_n^{\pm}(x)$ to give an adequate estimation of $\lambda^{\pm}(x)$. As for now, we set our initial conditions to be a randomly generated sequence of evenly distributed ones and zeros of length between three and thirteen, with eight perturbations of this configuration. It may be the case that fewer perturbations are needed, and different distributions may lead to different results. However, all such configurations converge after 100-200 iterations of the ECA, most to either zero or one, with a small subset of ECA rules leading to conditions to converge to $\lambda^{\pm} = .5$.

4 Wolfram Lyapunov Exponent

4.1 Theory

Stephen Wolfram proposed a somewhat simpler method for calculating the Lyapunov Exponent of ECA. Wolfram similarly equates the Lyapunov Exponents of ECA to the rate of information transmission through a cellular automaton. His method of generating perturbations is much simpler than that of Sherevsky.

This Lyapunov Exponent can be estimated by reversing a single cell in an arbitrary initial state. By then plotting the cells where the cellular automata differ it is possible to calculate an average rate of transmission of the perturbation. This is simply the rate at which the perturbation propagates from the single perturbed cell, visually it is the slope of the two edges of the propagating error:



there will then be clearly two values as the perturbation can propagate in both the positive and negative "x" direction as the automaton steps through time. Similar to the Shreshevsky method above we will refer to the left Lyapunov Exponent as λ_- and the right Lyapunov Exponent as λ_+.

The result of any single run of this method is highly variable given the dependence on initial conditions. We conjecture that taking the mean lyapunov exponent over many repeated iterations of this method using a randomly generated initial states would allow us to estimate the Lyapunov Exponent as the number of samples goes to infinity.

4.2 Numerical Estimation

Generally our algorithm to compute this Lyapunov Exponent was much simpler than that employed for the Shershevsky method. We generated the sample ECA using the method described above, with the horizontal axis as a single spatial dimension, and the vertical axis as a time dimension, with each iteration being a single time step with the initial state set as $t = 0$.

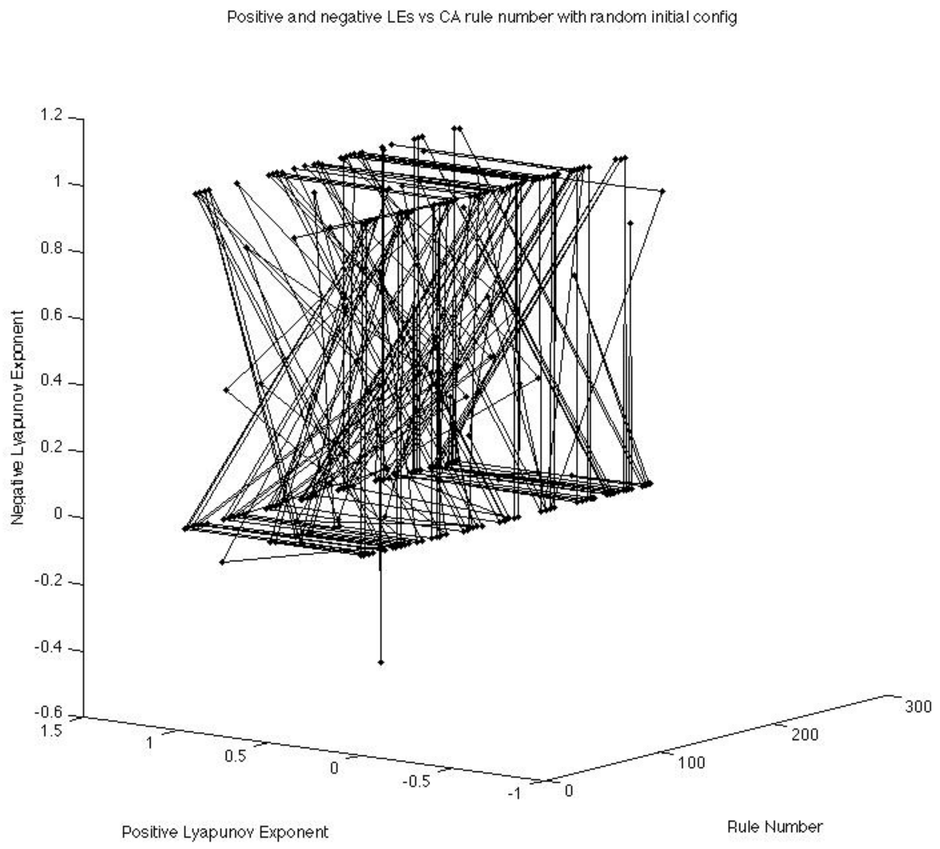
Using a randomly generated string as an initial state we generated an ECA, then switched the state of the $x = 0$ cell of the initial string. By superimposing these two ECAs and recording all cells where states differed, we were able to generate a matrix in which contained the error propagating through time. By measuring the displacement of error from the spatial origin in both directions we can generate a rough estimate for the Lyapunov Exponent, we ran this code an arbitrary number of times and took a simple average of our results for any single CA.

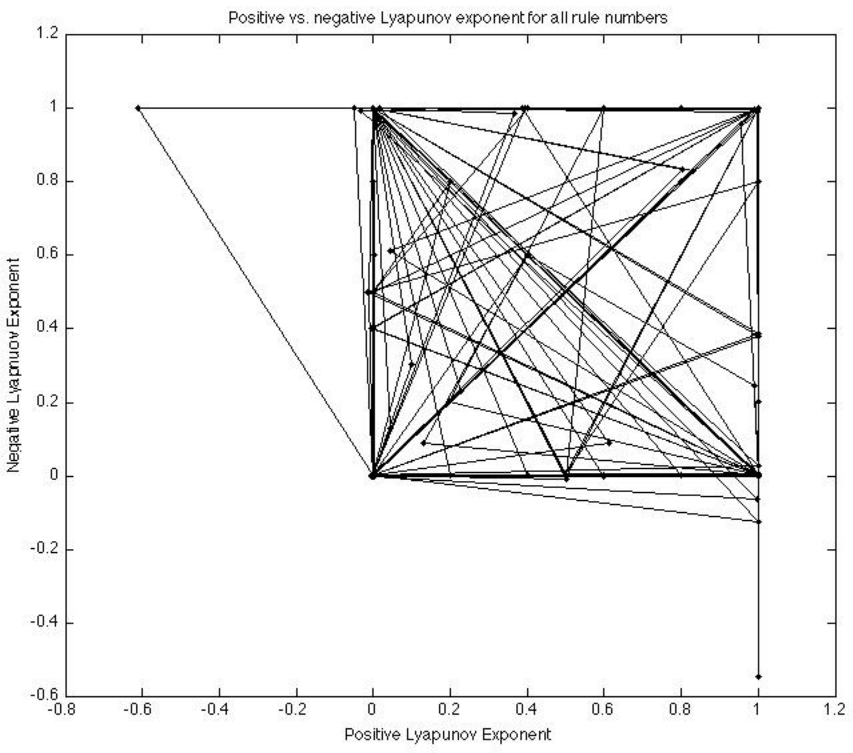
4.3 Analysis

Our simple statistical estimate of Lyanpunov Exponent using the Wolfram method seemed to generally match our results from the Shreshevsky method, both methods generally indicated that ECA either exhibit strongly chaotic behavior and error propagates with velocity equal to that of the general structure, that is the $\lambda_\pm = 1$, or that variance in initial conditions did not effect general sturcture and the Lyapunov Exponent was $\lambda_\pm = 0$. There were a handful of interesting rules which took on intermediate values, however we cannot effectively comment on the actual values of these Lyapunov Exponents given the simplicity of our method of estimation of the Wolfram method. In taking a simple mean, we discount any behavior where a given error structure falls back more than one cell in a single time step. Wolfram accounts for this behavior through constructing a Green's function based on the specific rule being examined, this method however was beyond the scope of this project. We will postulate that these intermediate values would converge to .5 given an effective means of analysis.

5 Conclusions

In our analysis of the dynamic behavior of ECA we were able to make some general observations concerning the chaotic behavior of randomly generally configurations of individual rules. While Lyapunov Exponents are ultimately a local property of specific configurations we found that most random configuration within each rule converge to a similar value. Further the vast majority of rules generate Lyapunov Exponents of approximately 0 or 1. This indicates that initial perturbations will most often either propagate with speed 1, that is to say propagate with the same speed as the general structure or not propagate at all. We must also note that there is a fairly distribution between rules which propagate in both, one of two, or no spatial directions. We also observed that certain rules had either a left or right Lyapunov Exponent that seemed to approach .5. The following figures show the distribution of left and right Lyapunov Exponents calculated using the Shreshevsky method. The first shows positive and negative exponents plotted against their rule number, while the second collapses the rule number dimension. Negative exponents are an artifact of error generated on the boundary conditions.


Positive and negative LEs vs CA rule number with random initial config
Negative Lyapunov Exponent
Positive Lyapunov Exponent
Rule Number
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
1.5
1
0.5
0
-0.5
-1
0
100
200
300


Positive vs. negative Lyapunov exponent for all rule numbers
Negative Lyapnuov Exponent
Positive Lyapunov Exponent
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2

Though Shreshevsky's theory is well developed there is little hard data by which to judge our numerical results. We were loosely compare our results to the simpler Wolfram method and generally confirmed our result.

We extend Shreshevsky's condition for the probability measure on the state space in his proof of the convergence of the limits defining the Lyapunov exponents to postulate that the set for which the limits converge is the set of all configurations with a finite number of ones (or zeros). This may be tested by adding a noise term along the boundary that imitates the effects of an infinite initial state interacting with our finite defined configuration. We suspect that the left and right Lyapunov exponents under such conditions would not converge.

6 Code

6.1 Cellular Automata Generator

```
function CA = one_dim_CA(rule_num, input_vec, iteration_num)

bin = dec2bin(rule_num) - 48;
if length(bin) ≠ 8
    a = 8 - length(bin);
    bin = [zeros(1,a),bin];
end


CA = zeros(1+iteration_num, length(input_vec)+2*iteration_num);
CA(1,iteration_num+1:iteration_num+length(input_vec)) = input_vec;

        for m = 1:iteration_num
         x = [0 CA(m,1:end-1)]*4 + 2*CA(m,:) + [CA(m,2:end) 0];
        CA(m+1,:) = bin(8 - x);
        end


end
```

6.2 CA Plot Function

```
function CA = CAdraw(rule_num, input_vec, iteration_num)
```

```
bin = dec2bin(rule_num)*1−48;
if length(bin) ≠ 8
    a = 8 − length(bin);
    bin = [zeros(1,a),bin];
end


CA = zeros(1+iteration_num, length(input_vec)+2*iteration_num);
CA(1,iteration_num+1:iteration_num+length(input_vec)) = input_vec;

        for m = 1:iteration_num
         x = [0 CA(m,1:end−1)]*4 + 2*CA(m,:) + [CA(m,2:end) 0];
        CA(m+1,:) = bin(8−x);
        end

    imagesc(CA)
    colormap gray
    disp('See figure')
end
```

6.3 Positive Lyapunov Exponent Calculation

```
function h = lyapCA_plus(rule_num, input_vec, iteration_vec)

w = length(input_vec);
t = length(iteration_vec);
initial = cell(1,w);
pet_num = 8;
S = zeros(w,t);

for z = 1:w
for i = 1:pet_num
initial{z}(i,:) = [round(rand(1,w−1)), input_vec(z:end)];
end

p = size(initial{z});
CA = zeros(1+max(iteration_vec), p(2)+2*max(iteration_vec),p(1));

for j = 1:p(1)
CA(:,:,j) = one_dim_CA(rule_num, initial{z}(j,:),max (iteration_vec));
end

q = size(CA);
s = zeros(pet_num, q(2));
```

```
for iteration_num = iteration_vec
        for n = 1:pet_num
            s(n,:) = CA(iteration_num,:,1) ≠ CA(iteration_num,:,n);
        end

     m = sum(s);
     r = find(m>0);
     if isempty(r)
        S(z, iteration_num == iteration_vec) = 0;
     elseif r(end) ≥ ceil(q(2)/2)
            S(z, iteration_num == iteration_vec) = r(end) - ceil(q(2)/2);
     end

end
end

lambda = max(S);
fitline = polyfit(iteration_vec,lambda,1);

subplot(2,1,1)
imagesc(one_dim_CA(rule_num, input_vec, max(iteration_vec)))
colormap gray
x = polyval(fitline, [0:.5:max(iteration_vec)]);

subplot(2,1,2)
plot(iteration_vec, lambda, 'cx', [0:.5:max(iteration_vec)], x, 'k-')

h = fitline(1);
end
```

6.4 Negative Lyapunov Exponent Calculation

```
function h = lyapCA_minus(rule_num, input_vec, iteration_vec)

w = length(input_vec);
t = length(iteration_vec);
initial = cell(1,w);
pet_num = 8;
S = zeros(w,t);

for z = 1:w
for i = 1:pet_num
initial{z}(i,:) = [input_vec(1:z), round(rand(1,w-1)) ];
```

```
end

p = size(initial{z});
CA = zeros(1+max(iteration_vec), p(2)+2*max(iteration_vec),p(1));

for j = 1:p(1)
CA(:,:,j) = one_dim_CA(rule_num, initial{z}(j,:),max (iteration_vec));
end

q = size(CA);
s = zeros(pet_num, q(2));

for iteration_num = iteration_vec
        for n = 1:pet_num
            s(n,:) = CA(iteration_num,:,1) ≠ CA(iteration_num,:,n);
        end
     m = sum(s);
     r = find(m>0);
     if isempty(r)
         S(z, iteration_num == iteration_vec) = 0;
     elseif r(1) ≤ ceil(q(2)/2)
     S(z, iteration_num == iteration_vec) = ceil(q(2)/2)−  r(1);
     end

end


end
lambda = max(S);
fitline = polyfit(iteration_vec,lambda,1);

subplot(2,1,1)
imagesc(one_dim_CA(rule_num, input_vec, max(iteration_vec)))
colormap gray
x = polyval(fitline, [0:.5:max(iteration_vec)]);

subplot(2,1,2)
plot(iteration_vec, lambda, 'cx', [0:.5:max(iteration_vec)], x, 'k−')

h = fitline(1);
end
```

6.5 Wolfram Lyapunov Exponent Calculation

```
function h = lyapcode(rule_num, input_vec, iteration_vec)

x= [0 0 0; 1 0 0; 0 1 0; 0 0 1; 1 1 0; 1 0 1; 0 1 1; 1 1 1];
y = input_vec;

for i = 1:8
initial(i,:) = [x(i,:) y];
end                                                    %creates peturbations

p=size(initial);
S = zeros(1,length(iteration_vec));

for iteration_num = iteration_vec
    CA = zeros(1+iteration_num, p(2)+2*iteration_num);
for j = 1:p(1)
CA(:,:,j) = one_dim_CA(rule_num, initial(j,:), iteration_num);
end                                          %generates array of CA for each peturbation

q = size(CA);
r = [];

for n = 1:ceil(q(2)/2)
     r = CA(end, (end+1) − n, 1);

    for m = 1:length(x)

       if CA(end, (end+1)−n, m) ≠ r
           S(iteration_vec == iteration_num) = (q(2)+2−n) −ceil(q(2)/2);
           break
       end

    end
        if S(iteration_vec == iteration_num) ≠ 0
            break
        end
end

h = zeros(1,length(iteration_vec));
for l = 1:length(iteration_vec)
h(l) = S(l)/iteration_vec(l);
end

end
S
h
end
```

References

[1] M. A. Shreshevsky. Lyapunov exponents for one-dimensional cellular automata. *Journal of Nonlinear Science*, 2:1–8, 1992.

[2] P. Tisseur. Cellular automata and lyapunov exponents.

[3] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3), July 1983.

[4] Stephen Wolfram. Twenty problems in the theory of cellular automata. *Physica Scripta*, T9:170–183, 1985.

[5] Stephen Wolfram. Random sequence generation by cellular automata. *Advances in Applied Mathematics*, 7:123–169, 1986.

[6] Stephen Wolfram. *Theory and Applications of Cellular Automata*, volume 1 of *Advanced Series on Complex Systems.* World Scientific Publishing Company, 1986.