A Brief Discourse on Ergodic Theory

Khalil El Achi

Final Project of PHYS 310

Under the supervision of
Dr. Sara Najem

Beirut, Lebanon

Table of Content

1 Introduction **3**

2 Dynamical System **3**

2.1 Demarcation of Dynamical Systems 5

2.1.1 Deterministic 5

2.1.2 Stochastic 6

3 Study Random Trajectories (Walkers) **6**

3.1 Limitless Phase Space 8

3.2 Limited Phase Space 8

3.3 Random Walkers with Bias 12

4 Ergodicity **13**

4.1 Time Average 14

4.2 Ensemble Average 15

5 Illustrating Non-Ergodic Systems **16**

5.1 Non-Ergodic Systems 16

5.1.1 Polya's Urn Model 16

6 Chaos **21**

6.1 Double Pendulum 23

6.1.1 Non-Chaotic Regime 23

6.1.2 Chaotic Regime 24

6.2 Quadruple Pendulum 24

6.2.1 Non-Chaotic Regime 25

6.2.2 Chaotic Regime 25

7 Baker's Map **26**

7.1 Geometric Illustration 26
7.2 Mathematical Illustration 26
7.3 Fractal Dimensions 27
7.4 Fixed Point 29

8 Class Notes **30**

1 Introduction

This project will be centered on utilizing and harnessing the power of computational tools such as MATLAB and Python to visualize and study the dynamics of a different systems and illustrate their characteristics.
The paper will start by exploring randomness and bias and how it impacts the trajectories in their perspective phase spaces. Then illustrate how these systems provide statistical power through these characteristics. Then finally, explore one interesting mapping and studying its dynamics.
The code that carry out a simulation will have an accompanied video posted on Github. The final shot will be inserted here as a picture.

2 Dynamical System

A dynamical system is a mathematical framework used to describe the evolution of a system over time. It is a concept used in various scientific disciplines, including physics, engineering, biology, economics, and more. In essence, a dynamical system consists of a set of rules or equations that govern the behavior of a system as it evolves from one state to another.
The most remarkable of these systems is a a simple harmonic oscillator. To attempt to simulate a harmonic oscillator's transition through a discretized time frame the following MATLAB code is executed, the video on GitHub is titled: "harmonic-oscillator-animation.mov"

```
% Parameters
amplitude = 1; % Amplitude of the oscillation
frequency = 5; % Frequency of the oscillation
time = linspace(0, 5, 300); % Time vector

% Calculate position using a sine function
position = amplitude * sin( frequency * time);

% Create a figure
figure;

% Create a line object for animation
lineHandle = plot(time(1), position(1), 'b', 'LineWidth', 2);

% Customize plot appearance
title('Harmonic Oscillator Animation');
```

```
xlabel('Time');
ylabel('Position');
grid on;

% Set initial axis limits
xlim([0, max(time)]);
ylim([-amplitude, amplitude]);
% Create a VideoWriter object
writerObj = VideoWriter('harmonic_oscillator_animation.mov', 'MPEG-4');
writerObj.FrameRate = 20; % Adjust the frame rate as needed

% Open the video file for writing
open(writerObj);
% Loop to animate the position
for t = 1:length(time)
% Update line data for the current time
set(lineHandle, 'XData', time(1:t), 'YData', position(1:t));

% Update the x-axis limit dynamically
xlim([time(1), time(end)]);

% Pause to control animation speed
pause(0.005);

% Update the plot for the next iteration
drawnow;
% Capture the frame and write to the video file
frame = getframe(gcf);
writeVideo(writerObj, frame);
end
% Close the video file
close(writerObj);
```

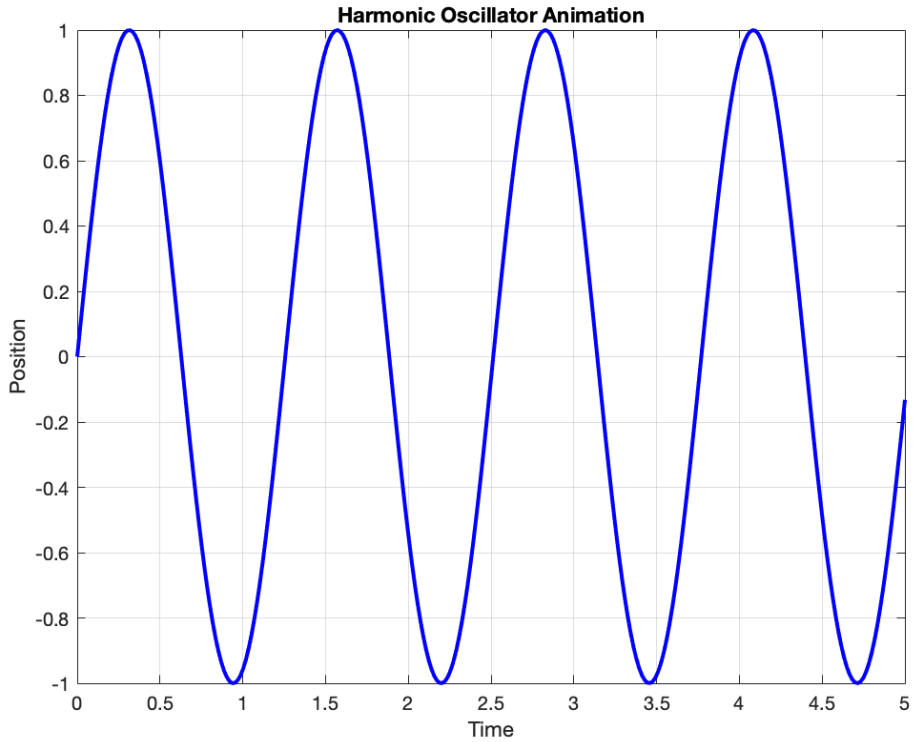

Figure 1: The Position of a Free harmonic Oscillator

2.1 Demarcation of Dynamical Systems

Taking into consideration the Harmonic Oscillator, one knows with ultimate certainty where the oscillator will be at any moment in time. This makes the system deterministic. In fact, many dynamical systems show this behavior. But many are random and we have very little understanding of there dynamics due to their stochastic behavior.

2.1.1 Deterministic

In a deterministic dynamical system, the future states of the system are completely determined by its current state and the evolution rule.

There is no randomness or uncertainty in the evolution of the system.

Given the initial conditions, the same set of rules will always lead to the same trajectory or path in the state space.

Examples of deterministic systems include many classical physics models, such as the motion of planets in the solar system described by Newton's laws.

2.1.2 Stochastic

In a stochastic dynamical system, randomness or uncertainty is inherent in the evolution rules.

The future states of the system are not uniquely determined by the current state and the rules; instead, they are influenced by random or probabilistic factors.

The evolution of the system is described using probability distributions rather than exact trajectories.

Stochasticity can arise from external random forces, inherent system noise, or a combination of both.

Examples of stochastic systems include Brownian motion, which describes the random movement of particles suspended in a fluid, or certain aspects of financial markets where randomness plays a role in price movements.

3 Study Random Trajectories (Walkers)

The first simulation work of this project will center around random walkers and studying their dynamics.

Initially, one wants to create a random trajectory. That is start from an arbitrary initial point, take the origin for an instance, and as time diverges, the random walkers will make a random step in a 2D phase space.

The MATLAB code that will be used to execute a random unbiases Random Walk is the following:

```
function [x,y] = Random_Walk(x,y)
int_x=x;
int_y=y;
j=randi(4,1,1);
if j==1
int_x=int_x-1;
elseif j==2
int_x=int_x+1;
elseif j==3
int_y=int_y+1;
else
int_y=int_y-1;
end
x=int_x;
y=int_y;
```

```
end
```

```
Nw=2;
Ns=200;
x=zeros(Nw,Ns+1);
y=zeros(Nw,Ns+1);
hold on
lineHandle1 = plot(x(1,1), y(1,1), 'mo-', 'LineWidth', 2);
lineHandle2 = plot(x(2,1), y(2,1), 'ko-', 'LineWidth', 2);
grid on
xlabel('x position')
ylabel('y position')
title('Evolution of Random Walkers')
%axis equal

writerObj = VideoWriter('Random Walkers.mov', 'MPEG-4');
writerObj.FrameRate = 20;

%Open the video file for writing
open(writerObj);
for j=1:Ns
i=1;
[x(i,j+1),y(i,j+1)]=Random_Walk(x(i,j),y(i,j));
i=2;
[x(i,j+1),y(i,j+1)]=Random_Walk(x(i,j),y(i,j));

set(lineHandle1, 'XData', x(1, 1:j), 'YData', y(1, 1:j));
set(lineHandle2, 'XData', x(2, 1:j), 'YData', y(2, 1:j));
% xlim([-10,10])
% ylim([-10,10])
pause(0.05)
drawnow;
frame = getframe(gcf);
writeVideo(writerObj, frame);
end
close(writerObj);
```

3.1 Limitless Phase Space

When placing the random walkers in a 2D limitless phase space, they evolve without any bias and any perturbation. The animation will be uploaded as "Random-Walkers.mov".

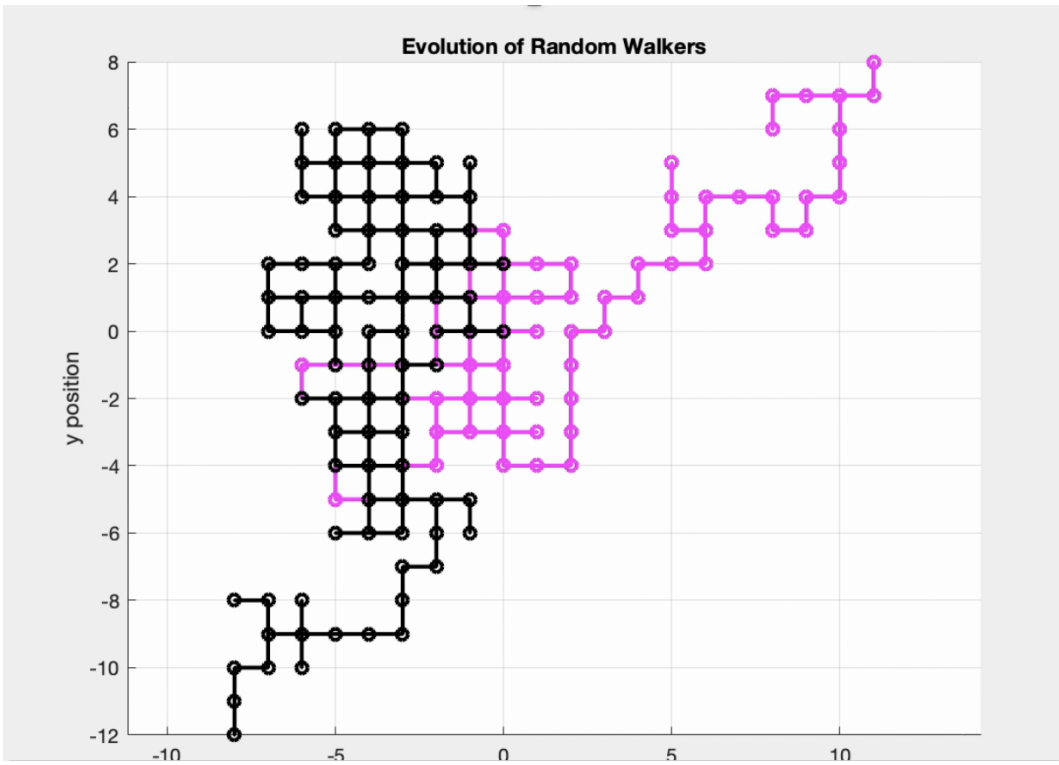

Figure 2: Trajectories of Two random walkers in a limitless Phase Space

For the sake of studying interesting dyanmics and how the system will fill up the phase-space, we must restrict the phase space volume to a 20x20 or a 10x10 square and see their behavior then.

To do that, we must deal with their perspective behavior on the boundaries.

3.2 Limited Phase Space

As the random walkers approach the boundaries I need to ensure they get repelled back into the phase space volume. For instance, when their x position reaches 10, I need to tell my code they can only proceed in the up, down, or left direction. And perpectively for the other 3 boundaries.

Now the Random_Walk function must be edited to 4 different functions.

```
function [x,y] = Random_Walk_Down(x,y)
int_x=x;
int_y=y;
j=randi(3,1,1);
```

```
if j==1
int_x=int_x+1;
elseif j==2
int_x=int_x-1;
else
int_y=int_y+1;
end
x=int_x;
y=int_y;
end
```

```
function [x,y] = Random_Walk_LEFT(x,y)
int_x=x;
int_y=y;
j=randi(3,1,1);
if j==1
int_x=int_x+1;
elseif j==2
int_x=int_y-1;
else
int_y=int_y+1;
end
x=int_x;
y=int_y;
end
```

```
function [x,y] = Random_Walk_RIGHT(x,y)
int_x=x;
int_y=y;
j=randi(3,1,1);
if j==1
int_x=int_y+1;
elseif j==2
int_x=int_x-1;
else
int_y=int_y-1;
end
```

```
x=int_x;
y=int_y;
end
```

```
function [x,y] = Random_Walk_UP(x,y)
int_x=x;
int_y=y;
j=randi(3,1,1);
if j==1
int_x=int_x+1;
elseif j==2
int_x=int_x-1;
else
int_y=int_y-1;
end
x=int_x;
y=int_y;
end
```

The final code is as follows:

```
Nw=2;
Ns=200;
x=zeros(Nw,Ns+1);
y=zeros(Nw,Ns+1);
hold on
lineHandle1 = plot(x(1,1), y(1,1), 'mo-', 'LineWidth', 2);
lineHandle2 = plot(x(2,1), y(2,1), 'ko-', 'LineWidth', 2);
grid on
xlabel('x position')
ylabel('y position')
title('Evolution of Random Walkers')
%axis equal
max = 10;
% writerObj = VideoWriter('Random Walkers_10.mov', 'MPEG-4');
% writerObj.FrameRate = 20;

%Open the video file for writing
```

```
% open(writerObj);
for j=1:Ns
i=1;
[x(i,j+1),y(i,j+1)]=Random_Walk(x(i,j),y(i,j));
i=2;
[x(i,j+1),y(i,j+1)]=Random_Walk(x(i,j),y(i,j));
i=1;
if x(i,j)==max
[x(i,j+1),y(i,j+1)]=Random_Walk_RIGHT(x(i,j),y(i,j));
elseif x(i,j) == -max
[x(i,j+1),y(i,j+1)]=Random_Walk_LEFT(x(i,j),y(i,j));
elseif y(i,j) == max
[x(i,j+1),y(i,j+1)]=Random_Walk_UP(x(i,j),y(i,j));
elseif y(i,j) == -max
[x(i,j+1),y(i,j+1)]=Random_Walk_Down(x(i,j),y(i,j));
else
[x(i,j+1),y(i,j+1)]=Random_Walk(x(i,j),y(i,j));
end
i=2;
if x(i,j)==max
[x(i,j+1),y(i,j+1)]=Random_Walk_RIGHT(x(i,j),y(i,j));
elseif x(i,j) == -max
[x(i,j+1),y(i,j+1)]=Random_Walk_LEFT(x(i,j),y(i,j));
elseif y(i,j) == max
[x(i,j+1),y(i,j+1)]=Random_Walk_UP(x(i,j),y(i,j));
elseif y(i,j) == -max
[x(i,j+1),y(i,j+1)]=Random_Walk_Down(x(i,j),y(i,j));
else
[x(i,j+1),y(i,j+1)]=Random_Walk(x(i,j),y(i,j));
end


set(lineHandle1, 'XData', x(1, 1:j), 'YData', y(1, 1:j));
set(lineHandle2, 'XData', x(2, 1:j), 'YData', y(2, 1:j));

xlim([-10,10])
```

```
            ylim([-10,10])
            pause(0.05)
            drawnow;
                    frame = getframe(gcf);
                writeVideo(writerObj, frame);
            end
                close(writerObj);
```

In a restricted 20x20 or 10x10 phase space volume the random walkers have such behaviors:

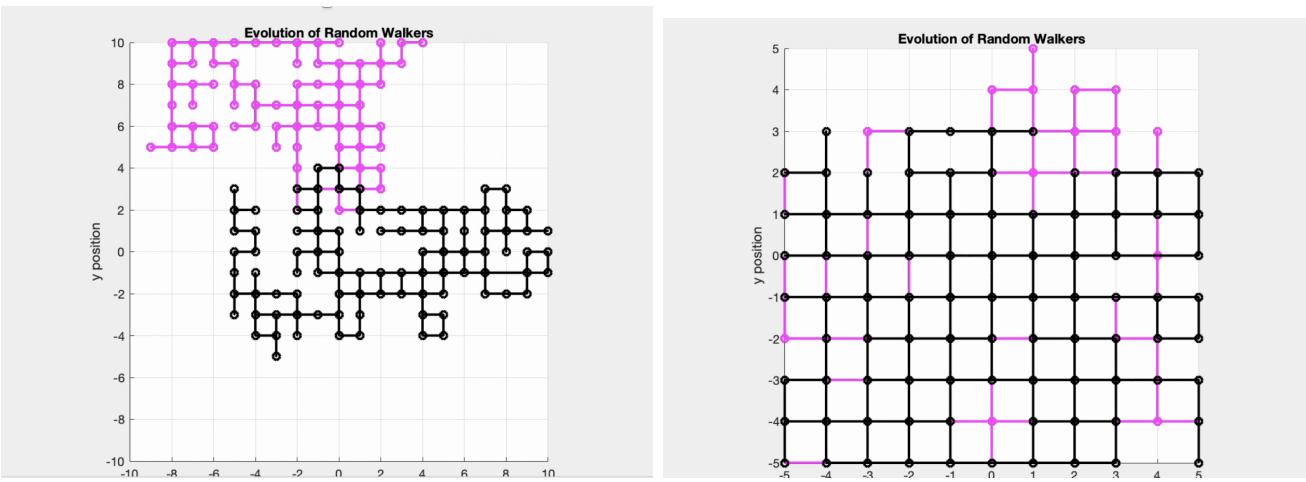

Figure 3: Trajectories of Two random walkers in a Restricted Phase Space

This way we have noticed interesting behavior. With a restricted phase space, the random walkers were able to fill the entire phase space volume and their trajectories become almost identical and indistinguishable.

This will be a corner axiom in Ergodicity.

3.3 Random Walkers with Bias

If some biases were introduced, such as giving the random walkers more weights to go up and right or left and down. The trajectories will look very interesting.

The Random_Walk function must be edited as such:

```
    function [x,y] = Random_Walk(x,y)
    int_x=x;
    int_y=y;
    j=randi(6,1,1);
    if j==1 || j==2
```

```
        int_x=int_x-1;
        elseif j==3 || j==4
        int_y=int_y-1;
        elseif j==5
        int_x=int_x+1;
        else
        int_y=int_y+1;
        end
        x=int_x;
        y=int_y;
        end
```

The trajectories will look prospectively as such:

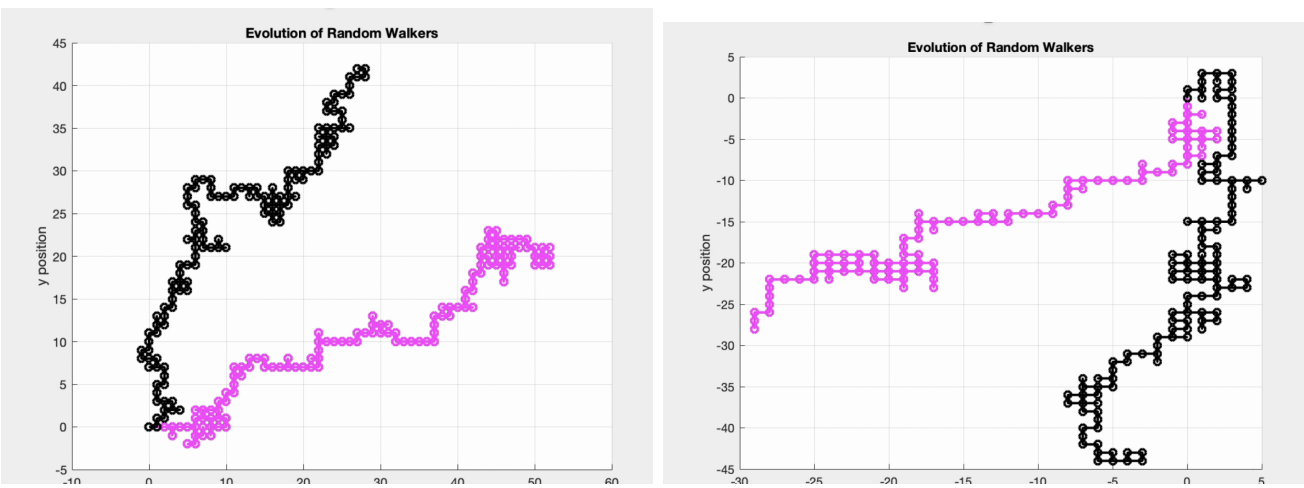

Figure 4: Trajectories of Two random walkers with Bias

This breaks Ergodicity for the the trajectories won't fill up the entire phase space volume with equal weights and creates distinguishable paths. Thus creating almost forbidden regions in the phase-space volume.

4 Ergodicity

Ergodicity is a concept that arises in the study of dynamical systems and probability theory. In a broad sense, ergodicity refers to the property of a system or process that, over time, explores and covers its entire state space, and statistical properties measured along its trajectory converge to their time averages. This concept has applications in physics, statistical mechanics, economics, and other fields.

In the context of dynamical systems, an ergodic system is one in which, over time, a system's trajectory visits all accessible states and spends a significant amount of time in each state.

Mathematically, a dynamical system is said to be ergodic if, over long periods, the time average of any observable quantity for a single trajectory converges to the ensemble average over all possible trajectories.

This property is crucial for the study of statistical mechanics and the foundation of the ergodic hypothesis, which asserts that, over time, a system in equilibrium will explore all accessible microstates.

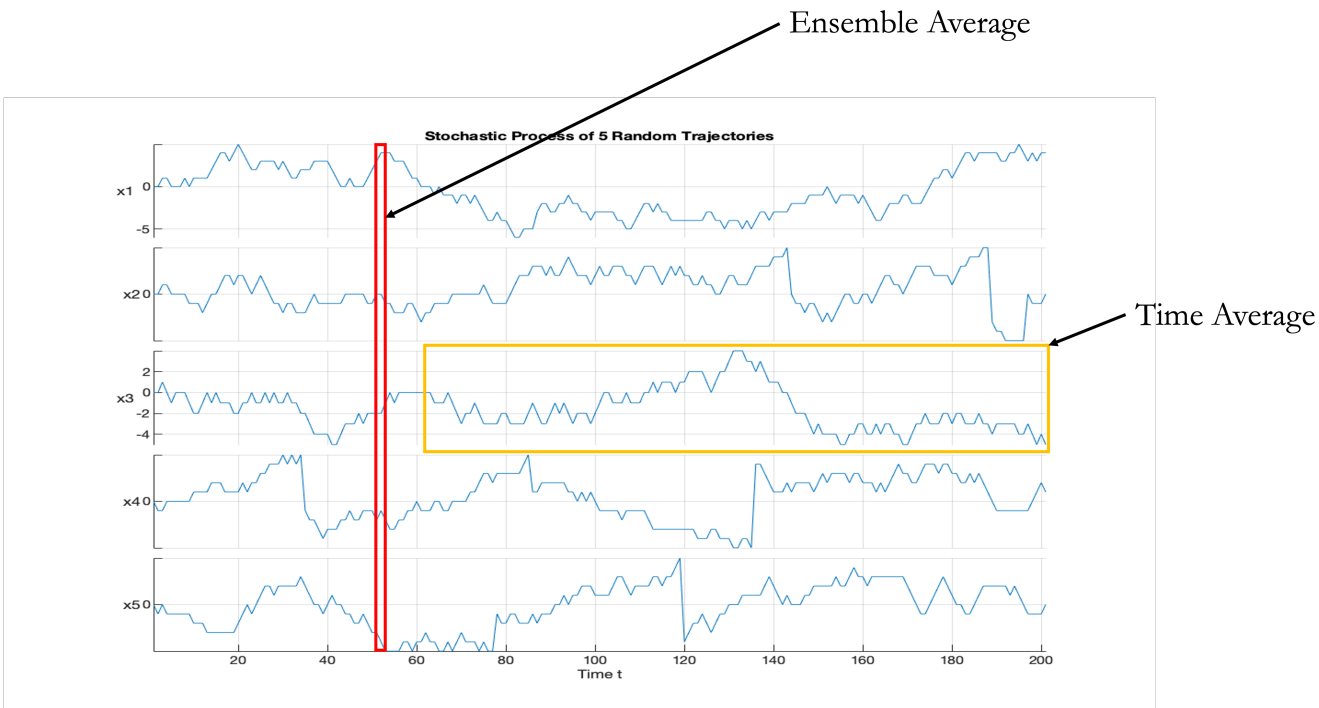

Figure 5: Trajectories of five Ergodic Random Walkers with Equivalent Time and Ensemble Averages

4.1 Time Average

The time average of a function f over a trajectory of a dynamical system is obtained by averaging the values of the function along the trajectory as time progresses.

$$\bar{f}_T = \frac{1}{T}\int_0^T x_n(t)dt$$

As T diverges to infinity the time average becomes:

$$\bar{f_T} = \lim_{T\to\infty} \bar{f_T} = \lim_{T\to\infty} \frac{1}{T} \int_0^T x_n(t)dt$$

4.2 Ensemble Average

The ensemble average of a function f is obtained by averaging the values of the function over all possible trajectories of the system.

$$\langle f_N \rangle = \frac{1}{N} \sum_0^N x_n(t)$$

As N diverges to infinity the time average becomes:

$$\langle f_N \rangle = \lim_{N\to\infty} \bar{f_N} = \lim_{N\to\infty} \frac{1}{N} \sum_0^N x_n(t)$$

Since the Ergodic system will explore the entire phase space, the time average of one trajectory over infinite time space, or the ensemble average of infinite trajectories in one instant of time will be equal.

$$\bar{f_T} = \langle f_N \rangle$$

5 Illustrating Non-Ergodic Systems

5.1 Non-Ergodic Systems

5.1.1 Polya's Urn Model

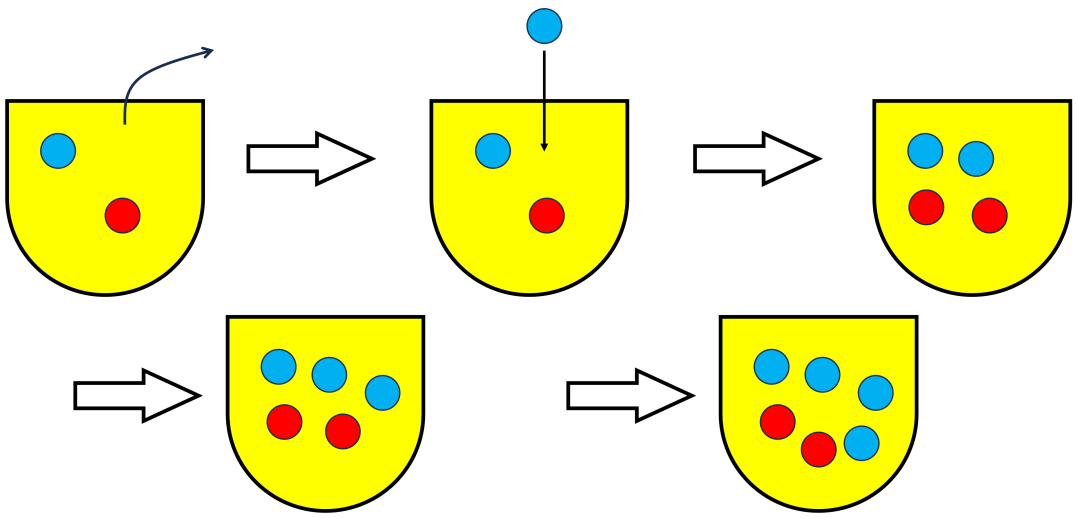

Figure 6: Illustrating Polya's Urn Model

Polya's Urn is a game that is drawn in the figure above. Take a hypothetical Urn that initial has one red ball and one blue ball. Each round one ball is drawn at random if the ball is blue, the drawn ball is returned back into the urn with an additional blue ball and the same happens if the ball was red where the drawn red ball will be returned to the urn along with an additional red ball. Finally after a set of rounds, one wishes to know the fraction of blue balls that are found in the Urn at each round. This is easily coded and graphed via MATLAB.

```
num_r=1;
num_b=1;
total=2;
prob_b=[];
prob_r=[];
for i=1:500
if rand<(num_r / total)
num_r=num_r+1;
total=total+1;
prob_r(i)=num_r/total;
prob_b(i)=num_b/total;
else
```

```
        num_b=num_b+1;
        total=total+1;
        prob_r(i)=num_r/total;
        prob_b(i)=num_b/total;
        end
        end
        i=1:500;
        plot(i,prob_b,'LineWidth',1.2)
        xlabel('Rounds Played')
        ylabel('Fraction of Blue Balls')
        title('Fraction of Blue Balls in the Urn at each Round')
        grid on
        hold on
```

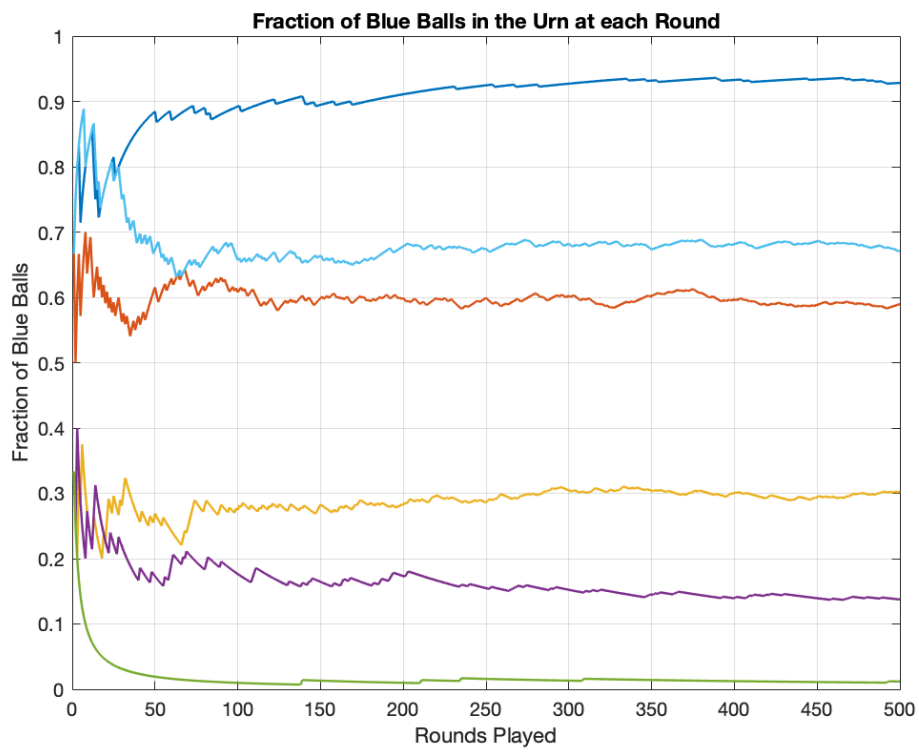

Figure 7: The Fraction of Blue Balls in the Urn after each round

Figure 7 shows the fraction of blue balls in the urn at each round or draw for multiple games played. For each game, one can see that a plateau forms, where the fraction of blue balls is not changing much but is fluctuating ever so slightly. Nonetheless, each game is categorically and conspicuously

different, and the outcome is very unpredictable. Sometimes the urn becomes dominated by blue balls and sometimes by red balls. Although again, there is no bias on initial draws. The probability to draw a blue ball or a red ball are the beginning of the game is precisely $P_b = P_r = \frac{1}{2}$. Since the balls are identical in shape and size. But still, one will hardly ever reach an outcome where there is a certain equilibrium between red or blue balls.

Here, certain numbers are impossible to achieve due to the nature of the game. Such as reaching 100% blue balls in the urn, since the game started with an existing red ball that will never be removed.

Therefore the system will resemble a non-ergodic dynamic where the sample can never pass through every mathematically possible state. Nonetheless, each system, i.e. each game, shows extreme path dependence. Thus the early results will be the indicative and determining step that will decide how the game will progress later. Whether the urn will be mostly blue or red balls is decided early on in the first few rounds of the game.

This is a good example of non-ergodic systems that have a penchant for dependency on the initial path taken. The example of Random Walkers above, are not impacted whatsoever by the previous position. That is the outcome of $T(x(t))$ doesn't depend on $x(t-1)$.

But in the Polya's Urn, the probability of drawing a blue ball changes after each round. Thus breaking ergodicity. In stochastic problems, the next position in the trajectory is not dependent on the previous realization since it is perfectly random.

Although this game is also considered "random," since there is no technical bias in choosing one ball over the other for they are identical, it unlocks a new definition and contraposition of the true idea of randomness.

For multiple Balls, consider an Urn with Green, Yellow, Red, and Blue balls. The code becomes:

```
num_r=1;
num_b=1;
num_g=1;
num_y=1;
total=4;

for i=1:600
prob_r=num_r(i)/total;
prob_b=num_b(i)/total;
prob_g=num_g(i)/total;
prob_y=num_y(i)/total;
x = rand;
if x < prob_r
```

```
num_r(i+1) = num_r(i) + 1;
num_g(i+1) = num_g(i);
num_y(i+1) = num_y(i);
num_b(i+1) = num_b(i);
total=total+1;
elseif x > prob_r && x< (prob_r + prob_b)
num_b(i+1) = num_b(i) + 1;
num_r(i+1) = num_r(i);
num_g(i+1) = num_g(i);
num_y(i+1) = num_y(i);
total=total+1;
elseif x > (prob_r + prob_b) && x < (prob_r + prob_b + prob_g)
num_g(i+1) = num_g(i) + 1;
num_y(i+1) = num_y(i);
num_b(i+1) = num_b(i);
num_r(i+1) = num_r(i);
total=total+1;
else
num_y(i+1) = num_y(i) + 1;
num_b(i+1) = num_b(i);
num_r(i+1) = num_r(i);
num_g(i+1) = num_g(i);
total = total +1 ;
end
end
i=1:601;
hold on
grid on
plot(i,num_r,'r','LineWidth',1.2)
plot(i,num_b,'b','LineWidth',1.2)
plot(i,num_g,'g','LineWidth',1.2)
plot(i,num_y,'y','LineWidth',1.2)
xlabel('Rounds Played')
ylabel('Fraction of Blue Balls')
title('Number of Red, Blue, Green, and Yellow Balls')
```

The evolution of the number of balls in the Urn as round progress is the following:

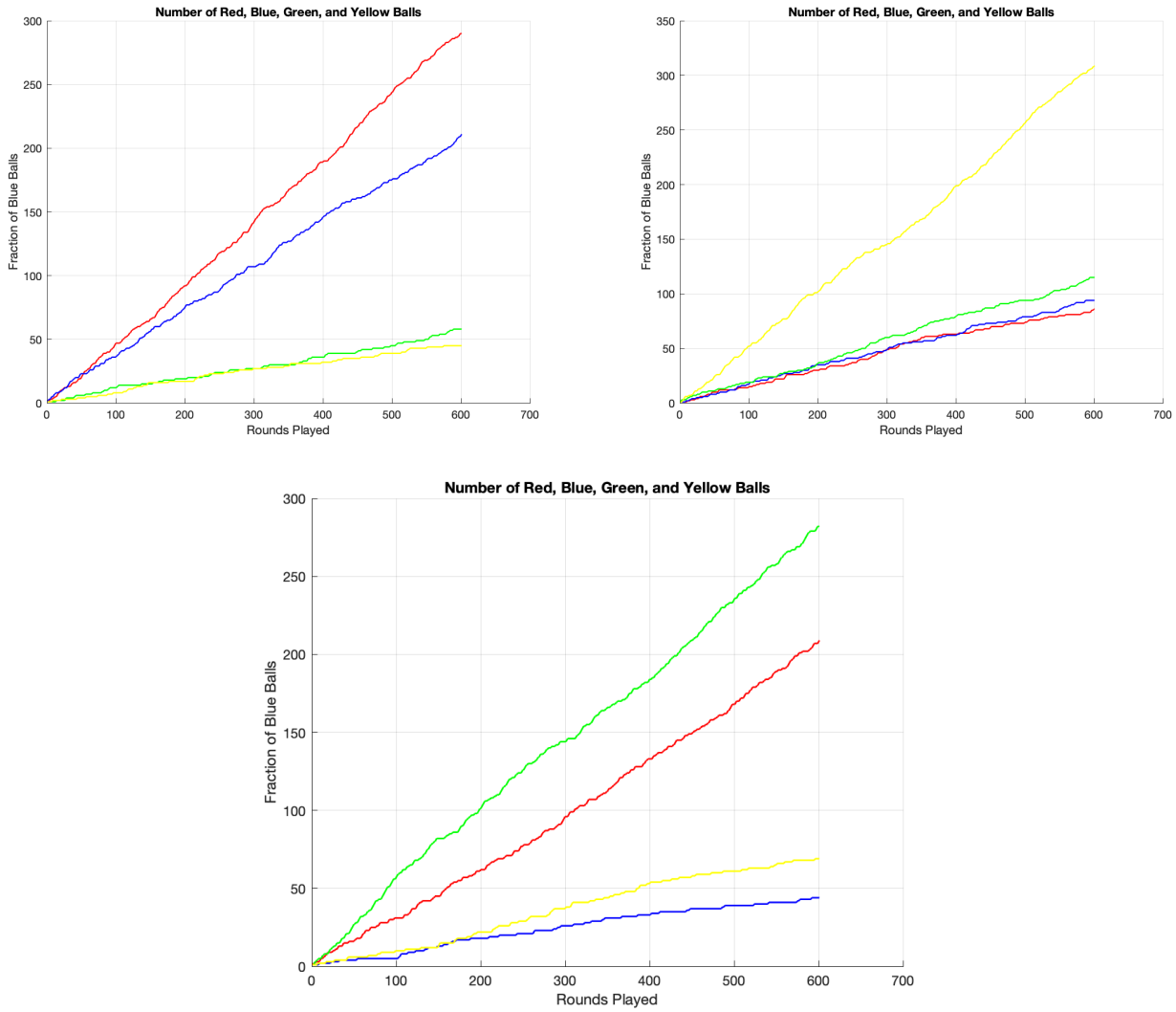

Figure 8: The Dominance of One or two Balls in the Urn

After the evolution of the game over a considerable amount of time, the Urn will be dominated by one or two balls. This is not coincidental and is a reality of the "random" game play.

This progression is similar to the wealth gap between different wealth groups shown in figure 9. The way that resources are divided in this current economic climate, is proportional to the amount of wealth that you already have.

Consider the addition of wealth or resources in society as round of the game. Once the added resources in this "positive-sum" world is added to a society, it will be mostly distributed to the group of people who already boast a large wealth. Leaving the poor left behind.

This model is also a demonstration how abstract mathematical models can mirror realistic world phenomena. This model is also called why the rich will become richer and why the poor will remain poorer and stuck in a cycle of poverty.

If the current economic system remains, there will be no middle grown or rich folks. There will be only two distinguishable groups of people. The 1%, or even less, and the rest.

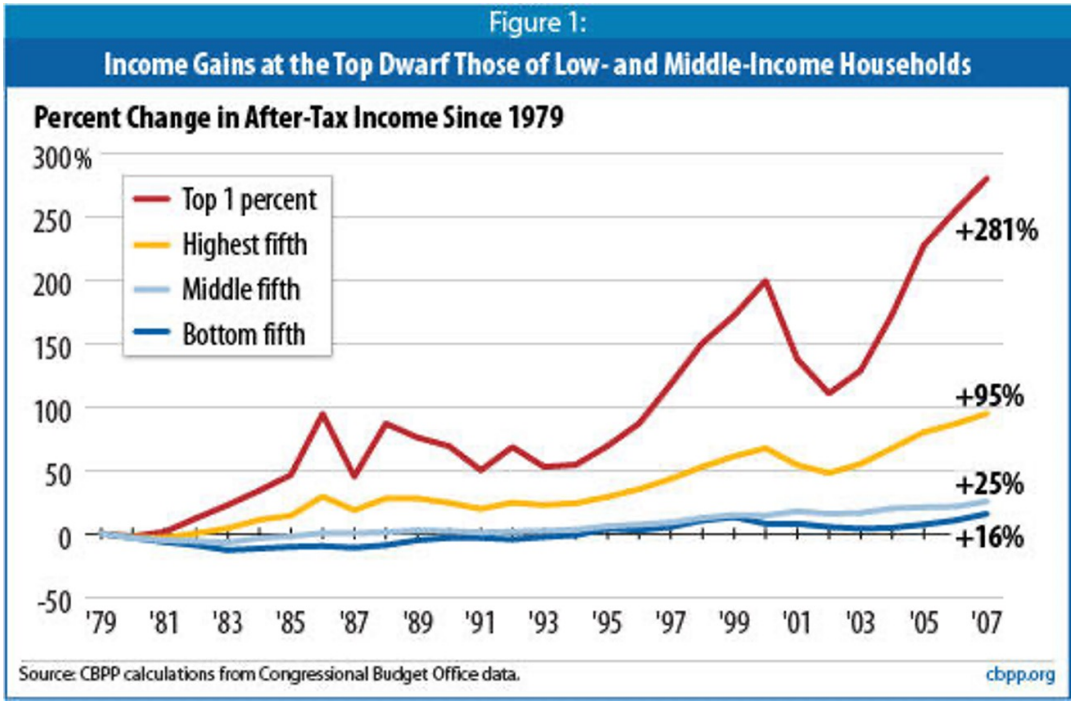

Figure 9: Showing the Income Gains of different Wealth Groups

6 Chaos

This section is only intended to illustrate how deterministic chaotic systems behave. Assume coupled harmonic oscillators. The system is deterministicly chaotic. Thus it depends insurmountably on its initial condition. That is $\theta, \dot{\theta}$.

The code that will animate couple harmonic oscillators is as follows:

```
function doublependelum
close all
t=0:0.005:10;
g=9.81;
l1=2;
l2=2;
m1=2;
m2=2;
x0=[1 -3 1 -3];
function dx=eq(t,x)
```

```
dx_1= x(2);
dx_2= (-m2*l1*(x(2)^2) *sin(x(1) - x(3))*cos(x(1) - x(3)) + g*m2*sin(x(3)
    ↪ )*cos(x(1) - x(3)) -m2*l2*(x(4)^2)*sin(x(1) - x(3)) - (m1+m2)*g*sin
    ↪ (x(1)))/(l1*(m1+m2) - m2*l1*(cos(x(1) - x(3)))^2);
dx_3= x(4);
dx_4= (m2*l2*(x(4)^2) *sin(x(1) - x(3))*cos(x(1) - x(3)) +g*sin(x(1))*cos
    ↪ (x(1) - x(3))*(m1+m2) + l2*(x(2)^2)*sin(x(1) - x(3))*(m1+m2) -g*sin
    ↪ (x(3))*(m1+m2)) / (l1*(m1+m2) - m2*l2*(cos(x(1) - x(3)))^2);
dx=[dx_1;dx_2;dx_3;dx_4];
end
[t,x] = ode45(@eq,t,x0);

plot(t,x(:,1),t,x(:,3))
xlabel('Time (s)')
ylabel('Angle (rad)')
X1=l1*sin(x(:,1));
Y1=-l1*cos(x(:,1));
X2=X1 + l2*sin(x(:,3));
Y2=Y1 - l2*cos(x(:,3));
X0=0*X1;
Y0=0*Y1;
X=[X0 X1 X2];
Y=[Y0 Y1 Y2];
n=1;
writerObj = VideoWriter('pendulum_fast.mov', 'MPEG-4');
writerObj.FrameRate = 30; % Adjust the frame rate as needed
open(writerObj);
while n<length(X(:,1))
P=plot(X(n,:),Y(n,:),'ro-');
set(P,'LineWidth',1);
set(P,'MarkerFaceColor','r');
plot(X1(n),Y1(n),'m.');
plot(X2(n),Y2(n),'c.');
hold on
pbaspect([1 1 1])
xlim([-(5) (5)])
```

```
        ylim([-(5) (5)])
        pause(0.005)
        frame = getframe(gcf);
        writeVideo(writerObj, frame);
        delete(P)
        n=n+10;
        end
        close(writerObj);
        end
```

6.1 Double Pendulum

6.1.1 Non-Chaotic Regime

Choosing intiial conditions where the pendulums have zero launching speed and low angle with the vertical, show a transition that is calm and non-chaotic. The animation is uploaded as "pendulum_slow."

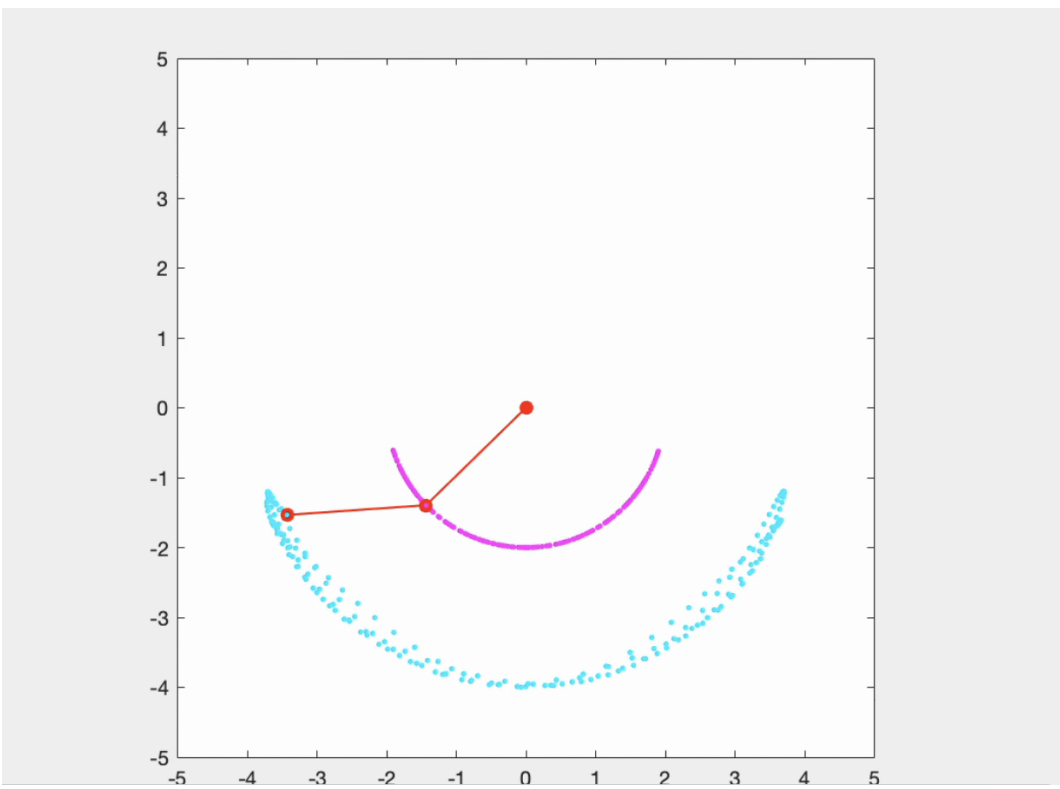

Figure 10: Non-Chaotic Double Pendulum

6.1.2 Chaotic Regime

By choosing a new set of initial conditions where by the pendulums are given a vast launching speed the projection becomes very chaotic and messy. The animation is uploaded as "pendulum_fast."

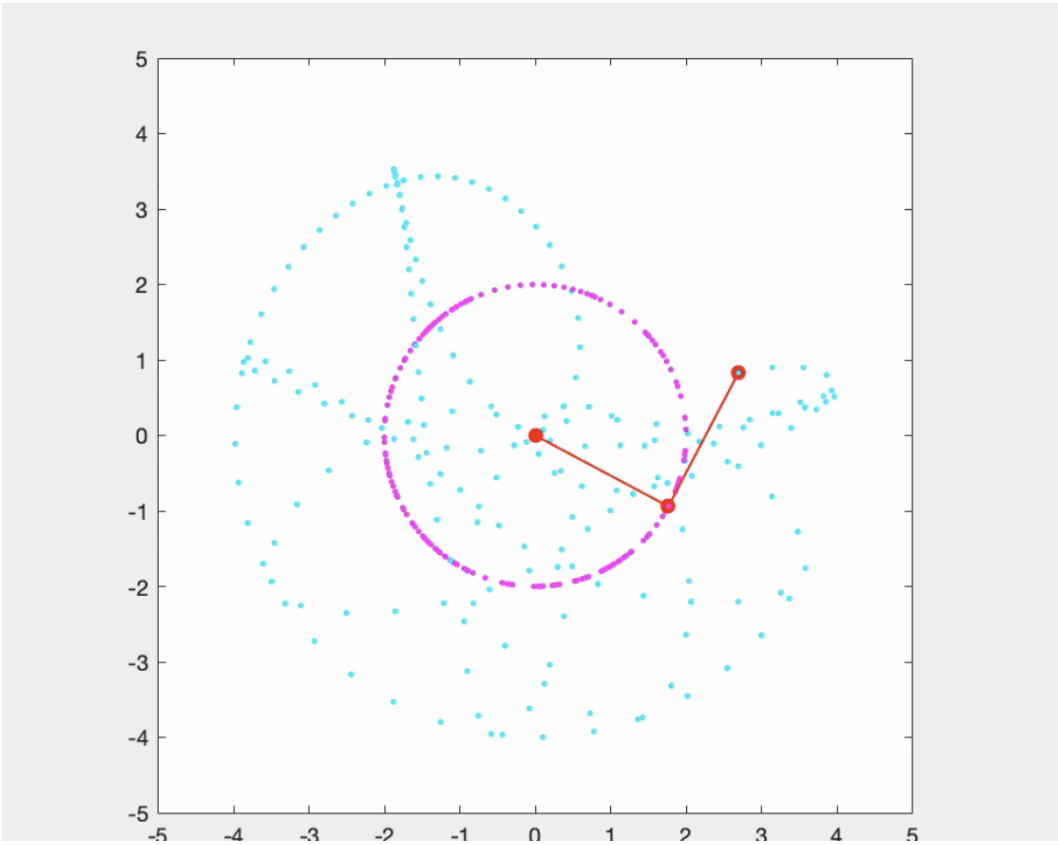

Figure 11: Chaotic Double Pendulum

6.2 Quadruple Pendulum

The quadruple pendulum was done in the same way as the double pendulum. Below are illustrations of both Chaotic and non-chaotic regimes. The animations will be uploaded as "pendulum_4_chaos" and "pendulum_4_no_chaos."

6.2.1 Non-Chaotic Regime

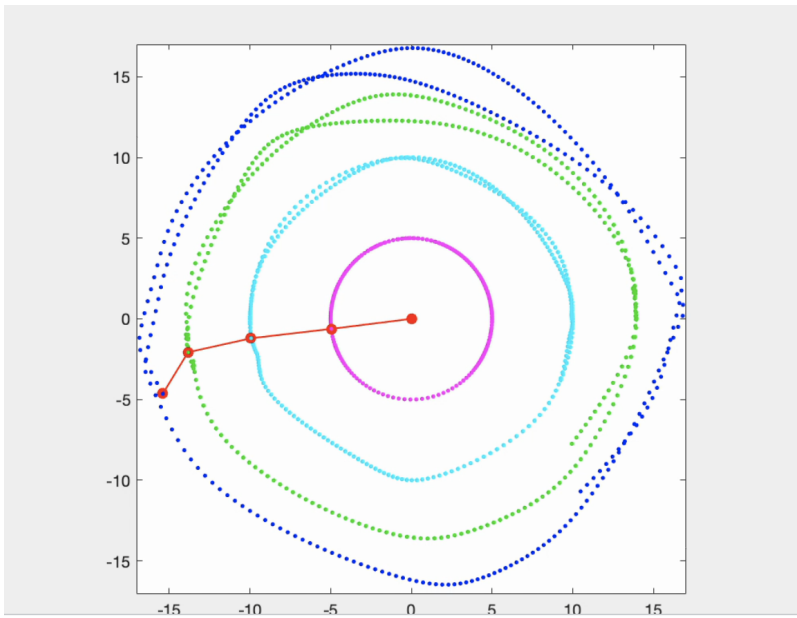

Figure 12: Non-Chaotic Double Pendulum

6.2.2 Chaotic Regime

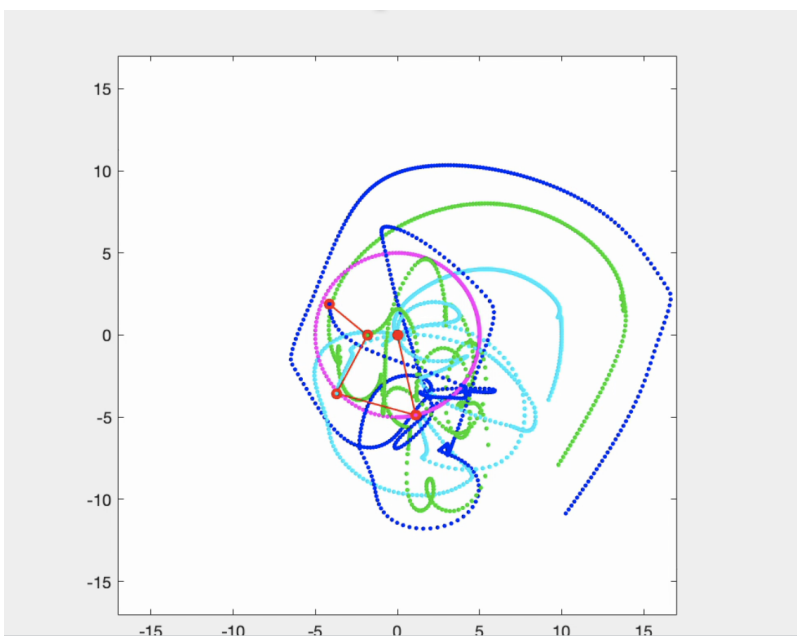

Figure 13: Non-Chaotic Double Pendulum

7 Baker's Map

7.1 Geometric Illustration

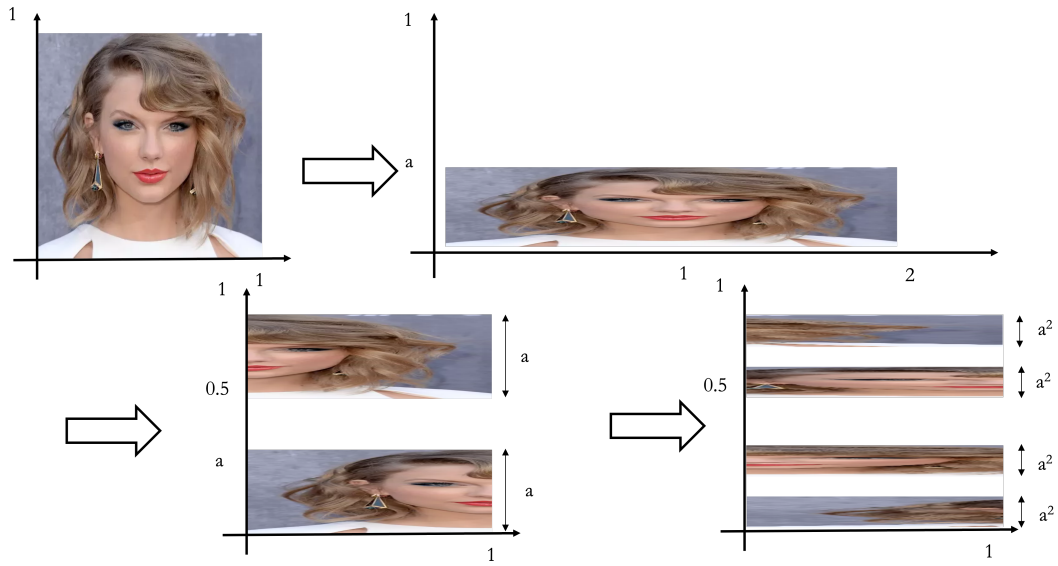

Figure 14: Explaining the Baker's Map

To fully understand a Baker's Map trasnformation. Take the illustration above.

Start by restricting a picture by a 1 x 1 square. Then stretch the picture to double its width and multiply its length by a factor of a. Where a $\in [0, 0.5]$. Then cut the stretched picture in half. Put the first picture in a region of 0.5 x 1 and the other picture stacked above in the remaining region. Maintain the empty space between a and 0.5 on the y-axis. After each iteration, there will be 2 bars each having a length of a. Thus after N iterations, there will be 2^N bars each having a^N width. Notice how the picture, that took the whole phase-space at the beginning, created allowed and forbidden region. The allowed regions being the bars and the empty spaced being the forbidden region.

7.2 Mathematical Illustration

The above modeling basically allows us to visualize this following transformation. For x $\in [0, 1]$ and y $\in [0, 1]$ the following transformation is done:

$$[x_{n+1}, y_{n+1}] = \begin{cases} (2x_n, ay_n) & \text{if } 0 \leq x < 0.5, \\ (2x_n - 1, ay_n + \frac{1}{2}) & \text{if } 0.5 \leq x < 1. \end{cases}$$

The above transformation is beautifully Ergodic. Thus, if one trajectory, i.e. one initial point, underwent this transformation for large number of iterations, the point's trajectory will create a fractal figure that mirror's that illustrated in figure 14. Thus, the entire phase space that will filled up the picture, can be filled up as well by a single trajectory.

This illustrates that the system of the Baker's Attractor is Ergodic. Thus, it explores the entire allowed phase space volume.

The system is deemed to be an attractor since it is a subset of the state space to which the system evolves over time. It represents a set of values or states toward which a dynamic system tends to move, regardless of the initial conditions. Attractors are essential concepts in the study of chaotic and nonlinear systems, providing insights into the long-term behavior and stability of such systems.

It is worth noting that the 1x1 phase space is maintained by restricting the dissipative parameter a to [0,0.5]. As mentioned above in the random walkers, restricting the phase space volume is an important axiom in studying ergodicity.

7.3 Fractal Dimensions

The following Python code will be used to illustrate the Baker's Map:

```
def BakersMap(a,x,y):
        y = a * y
        if x > 0.5:
                y = y + 0.5
                x = 1.9999999999999 * x
        while x > 1.0:
                x = x - 1.0
        return x,y

from pylab import *
a = 0.3
x = [0.1]
y = [0.3]
N = 100000
for n in range(0,N):
        xtemp, ytemp = BakersMap(a,x[n],y[n])
        x.append( xtemp )
        y.append( ytemp )
```

```
xlabel('x(n)')
ylabel('y(n)')
title("Dissipative Baker's Map with a =" + str(a))
plot(x,y, 'r,')
axis([0.0, 1.0, 0.0, 1.0])
grid()
show()
```

The code outputs the following Fractal Dimension in the Baker's Map:

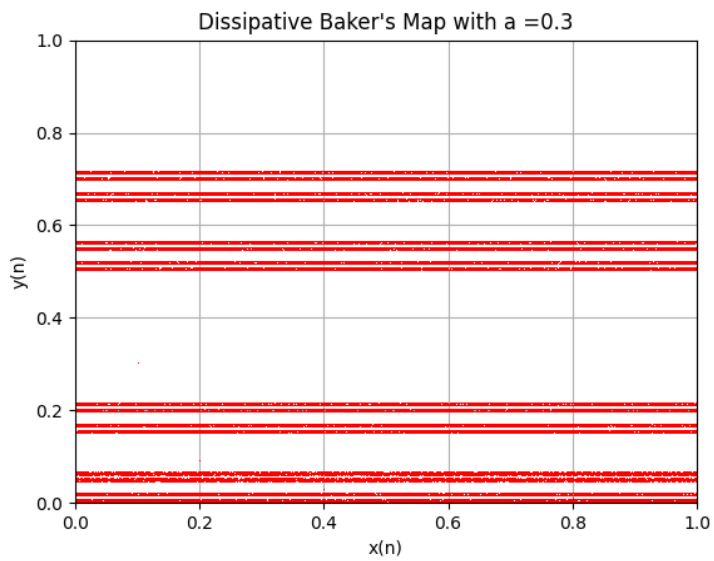

Figure 15: The Fractal Dimension of the Baker's Map at a=0.3

.

As expected, the single point starting at x = 0.3 and y = 0.4 was able to create a diagram similar to what was created by the procedure of shrinking, stretching, cutting, and stacking a full picture.

Thus this one point is able to explore the entirety of the allowed phase space.

Notice how the allowed and forbidden regions are directly linked to the dissipative term a. When $a < 0.5$, then the process is not volume preserving. And their will exist forbidden regions. As a approached 0.5, then the forbidden regions shrink and the trasnformation is able to fill the entirety of the 1x1 square region. Shown in figure 16.

This is a beautiful example of an ergodic trasnformation. One can perform more mathematical modeling to understand its dynamics.

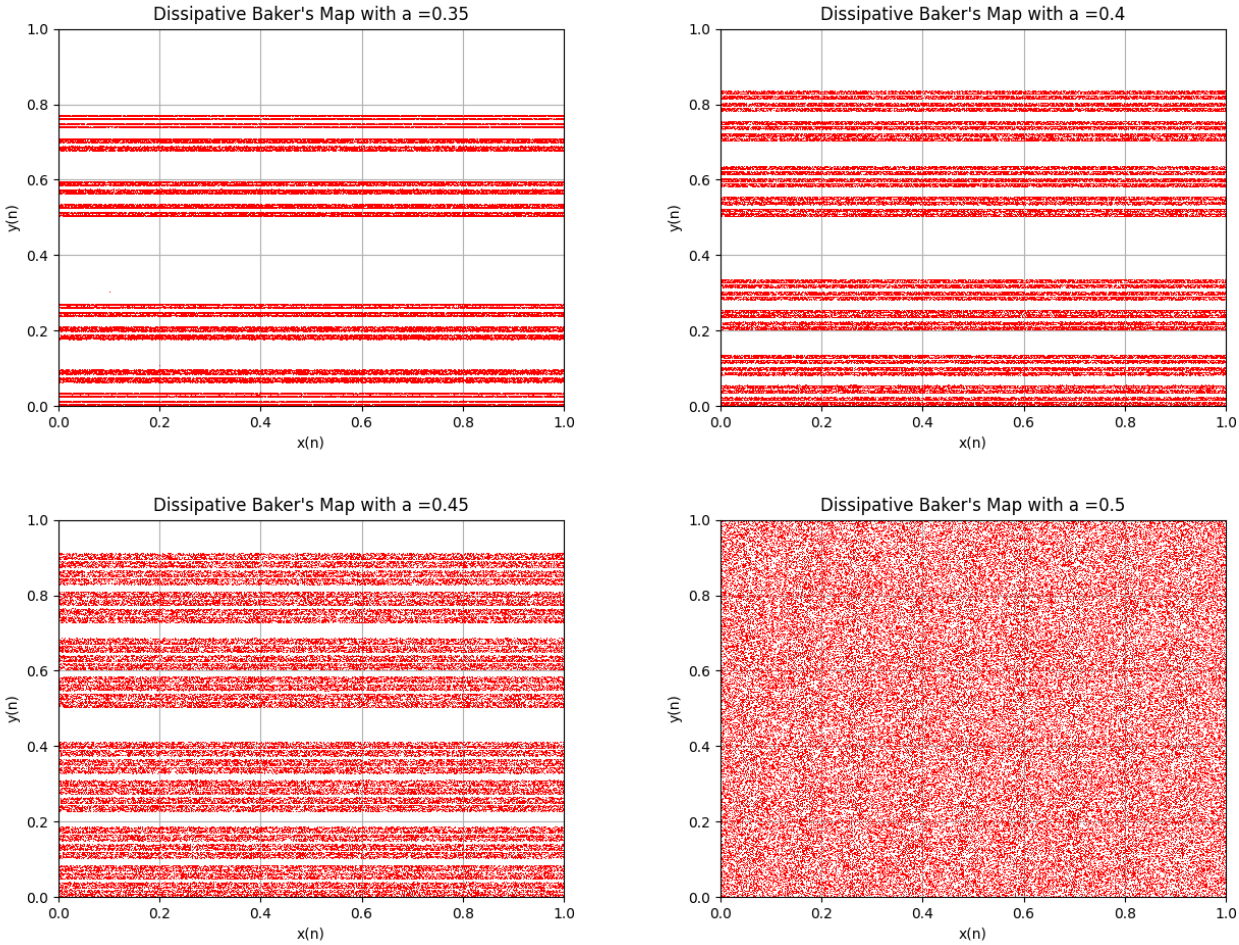

Figure 16: The Baker's Map at different dissipative term a

7.4 Fixed Point

The Jacobian Factor of the matrix can be written as such:

$$J = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

This is the same for each regime where x $\in [0, 0.5]$ and x $\in [0.5, 1]$. This systems boasts two fixed points. Each fixed point is on the edge. The first being the origin at (0,0) and the other being the edge point at (1,1). Since the eigenvalues of the system are 2 and c. Both are real and one is greater than 1 and the other is less than one. Thus the system has two unstable hyperbolic fixed points at (0,0) and (1,1).

Orbital study an be done by writing the transformation with linear combinations. Take B as the operation done on a point. The more operations one does, the more fixed points the system

has. Note that the original fixed points at (0,0) and (1,1), corresponded to only one iteration.

$$B_-(x) = \begin{pmatrix} 2 & 0 \\ 0 & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$B_+(x) = \begin{pmatrix} 2 & 0 \\ 0 & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

Then

$$B_+B_-(x) = \begin{pmatrix} 4 & 0 \\ 0 & a^2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

The Applying more transformation will increase the number of fixed points in the system. Thus the number of fixed points for B^N iterations corresponds to 2^N fixed points.

8 Class Notes

Here are the notes I have taken on my classmates presentations:

- Jad work centered around: Calculate the curvature of a sphere using discrete gravity, unify general relativity, and quantum mechanics. Using canonical quantum theory. Manifolds are continuous discreet, manifolds depend on the neighbors distance is not enough to define curvature. The vector changes direction in curved space, but stays the same in flat space. $\langle R \rangle$ represents smoothness of the manifold.

- Roua: Complex networks tackling the problem of communicability to notes can communicate through the shortest path, discover disorders and brains and networks the time series of the small world using the Lorenz System.

- Isar: recovering the dynamics of an Attractor, the Takens Embeddings. Along with studying the topological characteristics of it. Using time delays, get the mutual information between the nodes and reduce them to minimize false information. Getting the Lyaponov exponent to test for Chaos.

- Garo: Using course screening removing deformation on making the network simply replacing computing is in a class, and of course, screening, motivated, normalization groups and statistical physics.

- Fahid: studying the gravitational potential at a galactic center. Integral of motion of an isolating and non-isolating system of a Henon heiles system. Getting the Poincare Map and studying ergodicity.

- Semaan: recovering the solution of a diffusion equation using the Chebychev Polynomial. Setting the initial condition as a Gaussian, and using RK4 method.