A Project Report

On

Chaotic Dynamical Systems

BY

Vibhu Agrawal

2014B4AA0745H

Under the supervision of

Dr. Sharan Gopal

SUBMITTED IN FULLFILLMENT OF THE REQUIREMENTS OF

MATH F366: LABORATORY ORIENTED PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(March 2018)

Acknowledgements

I would like to express my immense gratitude to my teacher and mentor, Dr. Sharan Gopal, for his continued guidance, support and scrutiny, and for giving me the opportunity to take up this project. I thank Dr. Manish Kumar for giving me the guidance and opportunity to extend my knowledge to practical applications of the same. I would also like to thank my fellow classmates, especially Vaibhav Goyal, for helping me throughout the project in gaining a better understanding of the topics covered within.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled "**Chaotic Dynamical Systems**" submitted by Mr. Vibhu Agrawal (ID No. 2014B4AA0745H) in fulfillment of the requirements of the course Math F366, Laboratory Oriented Project Course, embodies the work done by him under my supervision and guidance.

Date: 20 April, 2018 **(Dr. Sharan Gopal)**

BITS- Pilani, Hyderabad Campus

Abstract

Chaos theory is a branch of mathematics that is focused on the behavior of dynamical systems that are highly sensitive to initial conditions. Chaos exists in many natural systems, like weather and climate, and also in artificial systems like road traffic. Chaos theory finds its application in a wide range of fields, from meteorology and anthropology to engineering and pure sciences.

This project is an attempt to gain insight and an in-depth understanding of the topic while revisiting and using the knowledge of differential equations and topology.

CONTENTS

Concepts in Dynamical Systems

Important Definitions

A discrete time dynamical system consists of a non-empty set X and a map $f: X \to X$.

For $n \in \mathbb{N}$, $f^n = f \circ \cdots \circ f,$ and is the n-th iterate of f or the n-fold composition.

If f is invertible, then $f^{-n} = f^{-1} \circ \cdots \circ f^{-1}$.

A continuous-time dynamical system consists of a space X and a one-parameter family of maps $\{f^t: X \to X\}$, $t \in \mathbb{R}$, or $t \in \mathbb{R}^{+}{}_{0}$, that forms a one-parameter group or subgroup such that $f^{t+s} = f^t \circ f^s$ and $f^0 = \mathrm{Id}$. The dynamical system is called a flow if the time t ranges over $\mathbb{R}$.

For $x \in X,$ the positive semi-orbit is defined as $O_f{}^{+}(x) = \cup_{t\geq 0} f^t(x)$ and in the invertible case the negative semi-orbit is defined as $O_f{}^{-}(x) = \cup_{t\leq 0} f^t(x)$. The orbit is defined as $(x) = O_f{}^{+}(x) \cup O_f{}^{-}(x) = \cup_t f^t(x)$.

A point $x \in X$ is said to be a periodic point of period $T > 0$ if $(x) = x,$ and its orbit is called periodic orbit. x is said to be a fixed point if $f^t(x) = x$ for all t. If (x) is periodic for some $s > 0$, we say that x is eventually periodic.

If $f^t: X \to X$ and $g^t: Y \to Y$ are dynamical systems, a semiconjugacy from (Y, g) to (X, f) is the surjective map $\pi: Y \to X$ such that $f^t \circ \pi = \pi \circ g^t$, for all t. It is called a conjugacy if the semiconjugacy is invertible.

Circle Rotation

We consider the circle $S^1 = [0,1]/\sim$, where ~ represents that 0 and 1 are identified. We define the distance metric on S^1 as

$$(x, y) = \min(|x - y|, 1 - |x - y|).$$

For $\alpha \in \mathbb{R}$, let R_α be the rotation of S^1 by angle $2\pi\alpha$.

$R_\alpha = x + \alpha \bmod 1$ R_α preserves the distance d.

If α is rational, then every orbit is periodic. If α is irrational, then every orbit is dense in S^1.

Expanding Endomorphisms of the Circle

For $m \in \mathbb{Z}$, $|m| > 1$, we define the times-m map $E_m: S^1 \to S^1$ by $E_m x = mx \bmod 1$.

E_m expands the arc length and distance between two neighbouring points, by a factor of m i.e. if

$$d(x, y) \leq \frac{1}{2m}, \text{then } \mathrm{d}(E_m x, E_m y) = md(x, y).$$

Now, let $\Sigma = \{0, \dots, m-1\}^{\mathbb{N}}$ be the set of sequences of elements in $\{0, \dots, m-1\}$. We define shift $\sigma: \Sigma \to \Sigma$ as a function that discards the first element of a sequence and shifts the remaining elements to one place to the left.

$$\sigma\ ((x_1, x_2, x_{3,\dots})) = (x_2, x_3, x_4, \dots)$$

We define a map,

$$\phi: \Sigma \to [0,1], \phi((x_i)_{i\in\mathbb{N}}) = \sum_{i=1}^{\infty} \frac{x_i}{m_i}$$

which is a base-m expansion of a series in Σ.

We can now consider ϕ as a map into S^1 by identifying 0 and 1.

$\phi \circ \sigma = E_m \circ \Sigma$, so ϕ is a semiconjugacy from σ to Σ.

Generalizing the notion of shift space, for an integer $m > 1$, set $A_m = \{1, \dots, m\}$.

A_m is called an alphabet and its elements are called symbols. A finite sequence of symbols is called a word. $\Sigma_m = A^{\mathbb{Z}}{}_m$ is the set of infinite two-sided sequences of symbols in A_m, and $\Sigma_m^+ = A_m^{\mathbb{N}}$ is the set of infinite one-sided sequences in A_m.

A sequence $x = (x_i)$ is said to contain a word $w = w_1 w_2 \dots w_k$ if there is some j such that $w_i = x_{j+i}$ for $i = 1, \dots k$.

The pair (Σ_m, σ) is called the full two-sided shift and the pair $(\Sigma_m^+, \sigma)$is called the full onesided shift.

Topological Dynamics

Definitions

A topological dynamical system is a topological space X and a continuous map $f: X \to X$.

If X and Y are topological spaces, then the continuous map $f: X \to Y$ is called a homeomorphism if it's a one-one and the inverse is continuous.

Let $f: X \to X$, and $g: Y \to Y$ be topological dynamical systems. A topological semiconjugacy from g to f is a surjective continuous map $h: Y \to X$ such that $f \circ h = h \circ g$. If h is a homeomorphism, it is called a topological conjugacy.

The ω limit set of x is defined as

$$\omega(x) = \bigcap_{n\in\mathbb{N}} \overline{\bigcup_{i\geq n} f^i(x)}$$

If f is invertible, the ω limit set of x is $\overline{(x) = (x) =} \bigcap_{n\in\mathbb{N}} \bigcup_{i\geq n} f^{-}(x)$.

A point x is called positively recurrent if $x \in \omega$. Periodic points are recurrent points.

A point x is called non-wandering if for any neighbourhood U of x there exists $n \in \mathbb{N}$ such that $(U) \cap U \neq \phi$.

A topological dynamical system $f: X \to X$ is said to be topologically transitive if there is a point $x \in X$ whose forward orbit is dense in X.

A topological dynamical system $f: X \to X$ is said to be a topological mixing if for any two non-empty open subsets $U, V \subset X,$ there is $N > 0$ such that $f^n(U) \cap V \neq \phi$ for $n \geq N$.

A homeomorphism $f: X \to X$ is expansive if there is $\delta > 0$ such that for any two distinct points $x, y \in X$ there is some $n \in \mathbb{Z}$ such that $d(\, f^n(x), f^n(y)) \geq \delta$. Here, δ is called the expansiveness constant.

A dynamical system (X, f) is called Devaney chaotic if:

1. f has sensitive dependence on initial conditions
2. (X, f) is topologically transitive
3. The set of periodic points is dense in X.

Non-Linear Systems

Definitions

If $f: E \to \mathbb{R}^n$ is differentiable on E, then $f \in C^1(E)$ if the derivative $Df: E \to L(\mathbb{R}^n)$ is continuous on E.

If E is an open subset of $\mathbb{R}^n$, and $f: E \to \mathbb{R}^n$, then $f \in C^1(E)$ iff the partial derivatives $\frac{\partial f_i}{\partial x_j}$ $i, j = 1, 2, 3, \dots n$, exist and are continuous on E.

Let E be an open subset of $\mathbb{R}^n$ containing x_0 and let $f \in C^1(E)$. Then there exists an $a > 0$ such that the initial value problem

$$\dot{x} = f(x)$$
$$x(0) = x_0$$

has a unique solution $x(t)$ on the interval $[-a, a]$.

Flow: Let E be an open subset of $\mathbb{R}^n$ and $f \in C^1(E)$, and for $x_0 \in E$, let $\phi(t, x_0)$ be the solution of the initial value problem given above defined on its maximal interval of existence $I_0(x)$. Then for $t \in I_0(x)$, the set of mappings ϕ_t defined by

$$\phi_t(x_0) = \phi(t, x_0)$$

is called the flow of the differential equantion $\dot{x} = f(x)$.

Let E be an open subset of $\mathbb{R}^n$ and $f \in C^1(E)$, and let $\phi_t: E \to E$ be the flow of the differential equation $\dot{x} = f(x)$ defined for all $t \in \mathbb{R}$. Then a set $S \subset E$ is called invariant with respect to the flow ϕ_t if $\phi_t\ (S) \subset\ S$ for all $t \in \mathbb{R}$ and S is called positively (or negatively) invariant with respect to the flow ϕ_t if $\phi_t(S) \subset\ S$ for all $t\ >\ 0$ (or $t\ <\ 0$).

A point $x_0 \in \mathbb{R}^n$ is called an **equilibrium point** or critical point of $\dot{x} = f(x)$ if $f(x_0) = 0$.

An equilibrium point x_0 is called a **hyperbolic equilibrium point** if none of the eigenvalues of the matrix $Df(x_0)$ have a zero real part.

Near a hyperbolic equilibrium point x_0, the nonlinear system

$$\dot{x} = f(x)$$

has stable and unstable manifolds S and U tangent to x_0 to the stable and unstable subspaces E^s and E^u of the linearized system

$$\dot{x} = Ax$$

where $A = Df(x_0)$.

Manifold: An n-dimensional differentiable manifold M is a connected metric space with an open covering $\{U_\alpha\}$ such that

1. For all $\alpha,\ U_\alpha$ is a homeomorphic to the open unit ball in $\mathbb{R}^n$, $B = \{x \in \mathbb{R}^n : |x| < 1\}$

2. If $U_\alpha \cap U_\beta \neq \phi$ and $h_\alpha: U_\alpha \to B$, $h_\beta: U_\beta \to B$ are homeomorphisms, then $h_\alpha(U_\alpha \cap U_\beta)$ and $h_\beta(U_\alpha \cap U_\beta)$ are subsets of $\mathbb{R}^n$ and the map

$$h = h_\alpha \circ h_\beta^{-1}: h_\beta(U_\alpha \cap U_\beta) \to h_\alpha(U_\alpha \cap U_\beta)$$

is differentiable and for all $x \in h_\beta(U_\alpha \cap U_\beta)$, the Jacobian determinant $detDh(x) \neq 0$.

The Stable Manifold Theorem: Let E be an open subset of $\mathbb{R}^n$ and $f \in C^1(E)$, and let ϕ_t be the flow of the non-linear system $\dot{x} = f(x)$. Suppose $f(0) = 0$ and that $Df(0)$ has k eigenvalues with negative real part and $n - k$ eigenvalues with positive real part. Then there exists a k-dimensional differentiable manifold S tangent to the stable subspace E^s of the linear system $\dot{x} = Ax$ (defined above) at 0 such that for all $t \geq 0$, $\phi_t(S) \subset S$ and for all $x_0 \in S$

$$\lim_{t\to\infty} \phi_t(x_0) = 0$$

and there exists an n-k dimensional differentiable manifold U tangent to the unstable subspace E^u of the linear system $\dot{x} = Ax$ (defined above) at 0 such that for all $t \leq 0$, $\phi_t(U) \subset U$ and for all $x_0 \in U$

$$\lim_{t\to-\infty} \phi_t(x_0) = 0$$

Image Encryption Based on Henon Map and Spatiotemporal Chaos

Henon Map

The Henon map was put forward in 1976 by Henon and is a commonly used 2dimensional chaotic map. It is given by

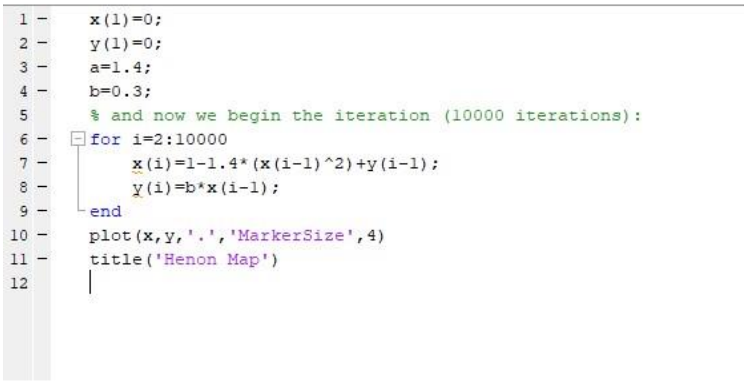
$$x_{d+1} = y_d + 1 - ax_{d2}$$

$$y_{d+1} = bx_d$$

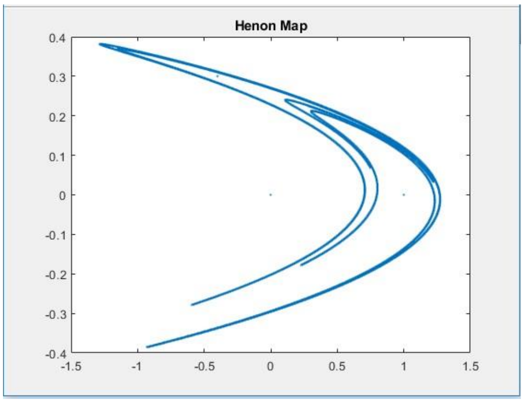
Where x and y are state variables, and the value of x and y at the dth iteration is x_d and y_d. Setting a=1.4 and b=0.3, we get.

$$x_{d+1} = y_d + 1 - 1.4x_{d2}$$

$$y_{d+1} = 0.3x_d$$



```
x(1)=0;
y(1)=0;
a=1.4;
b=0.3;
% and now we begin the iteration (10000 iterations):
for i=2:10000
    x(i)=1-1.4*(x(i-1)^2)+y(i-1);
    y(i)=b*x(i-1);
end
plot(x,y,'.','MarkerSize',4)
title('Henon Map')
```



One-way coupled map lattice

This is a spatiotemporal chaotic map, which is given by:

$$x_{n+1}^{i} = (1 - \varepsilon)f\left(x_n^i\right) + \varepsilon f\left(x_n^{i-1}\right)$$

Where $i = 0,1, \dots N - 1$ is the lattice site index, and ${x_n}^i$ represents the state value for ith iteration at time n; $\in (0,1)$ is a coupling coefficient. We take $= 0.2$. f is a chaotic map given by:

$$x_{n+1} = 1 - \lambda x_n^2$$

$\lambda \in [1.40115,2]$ is a control parameter, and $x_n \in [-1, 1]$.

To drive the OCML, we use Logistic and Chebyshev maps, to provide more randomness and more security.

A Logistic map is given by $x_{n+1} = (1 - x_n)$ where $\mu \in (3.57,4]$ is a constant and $x_n \in (0,1)$.

A Chebychev map is given by $x_{n+1} = \cos(k(arccos x_n))$, where $k \geq 2$ and $x_n \in (-1, 1)$. The initial state is set to (0,1).

The Chebychev map is used to fill the 0^{th} lattice, and the initial states for all other lattices (x_0^i) are generated using the Logistic map.

Algorithm

Step 1. Read the image of size MxN in A, where A_{ij} represents the pixel at ith row and jth column.

Step 2. Set appropriate secret keys.

Step 3. Define modulation key (MK) and interception key (IK) from plain image to enhance security.

$$IK = (\sum_{i=1}^{M} \sum_{J=1}^{N} A(i,j)) \mod K$$

$$MK = \frac{IK}{K}$$

Where K is an integer key ≥ 10001.

Step 4. Key modulation:

$$x'_{H0} = x_{H0} \times MK$$

$$y'_{H0} = y_{H0} \times MK$$

$$x'_{L0} = x_{L0} \times MK$$

$$x_{C'0} = x_{C0} \times MK$$

Step 5. Pixel position permutation

Now we iterate the Henon map $\max(M, N) + IK$ times and obtain two chaotic sequences x'_H and $y_{H'}$. Using these, we generate X_H and Y_H.

$$X_H(a) = x'_H(IK + a - 1)$$

$$Y_H(a) = y'_H(IK + b - 1)$$

where $X_H(a)$ is the ath element of X_H and $Y_H(b)$ denotes the bth element of Y_H; $1 \leq a \leq M$ and $1 \leq b \leq N$.

We define two chaotic index sequences $index1$ and $index2$.

$$index1 = (|X_H|)$$

$$index2 = (|Y_H|)$$

Where $|.|$ denotes the absolute value and (x) sorts the elements of x in an ascending order.

We now obtain a permutated image A_1:

$$A_1(p, q) = A(index(p), index(q))$$

Where $1 \leq p \leq M$ and $1 \leq q \leq N$

Step 6. Pixel values shuffling

We iterate the Logistic map IK+N-1 times and the Chebyshev map IK+M-1 times. Then, similar to step 5, two driving sequences , d_L and d_C are obtained from Logistic and Chebyshev maps respectively.

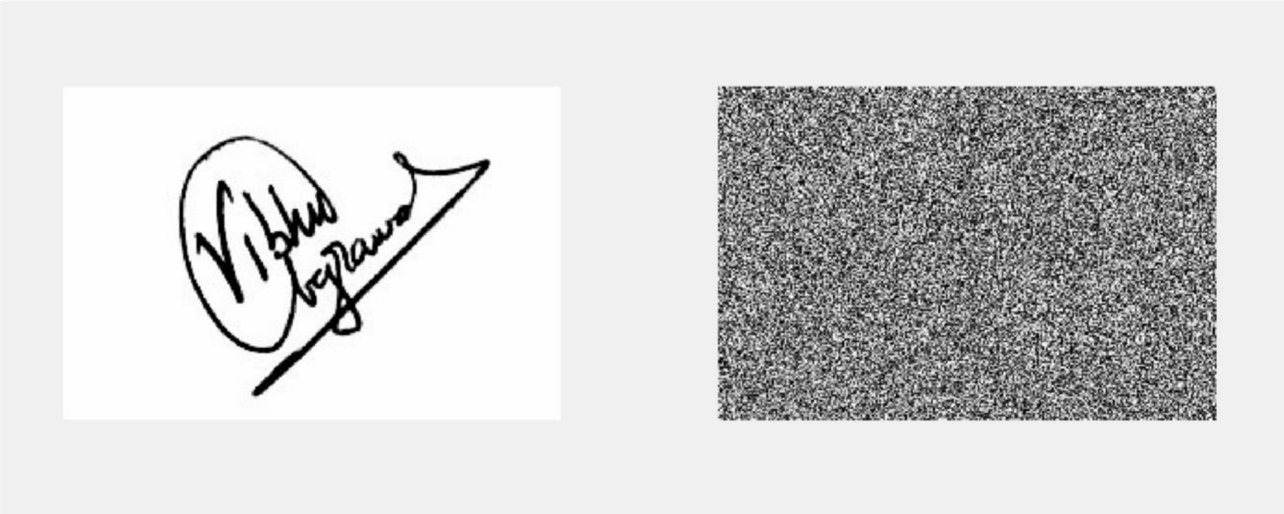
An OCML is generated by letting d_C fill the 0th lattice in order, and letting each value of d_L be the initial state value for the first to Nth lattice. The system is iterated $M - 1$ times, giving an image of size MxN. We calculate an updated image S as follows:

$$(i, j) = (r(10^{14}S(i, j)))mod256$$

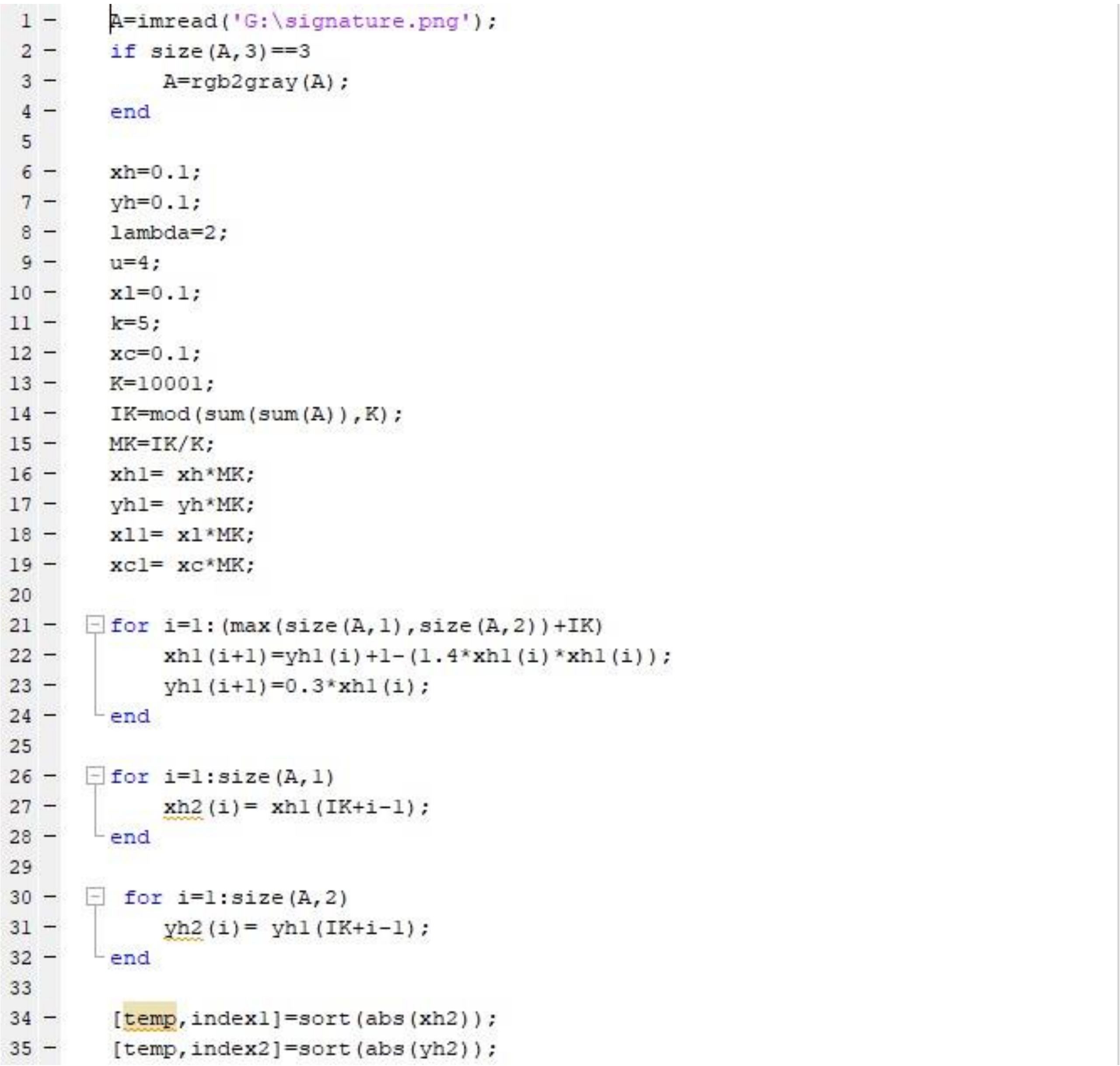
The final shuffled image A_2 is calculated as:

$$A_2 = A_1 \oplus S$$

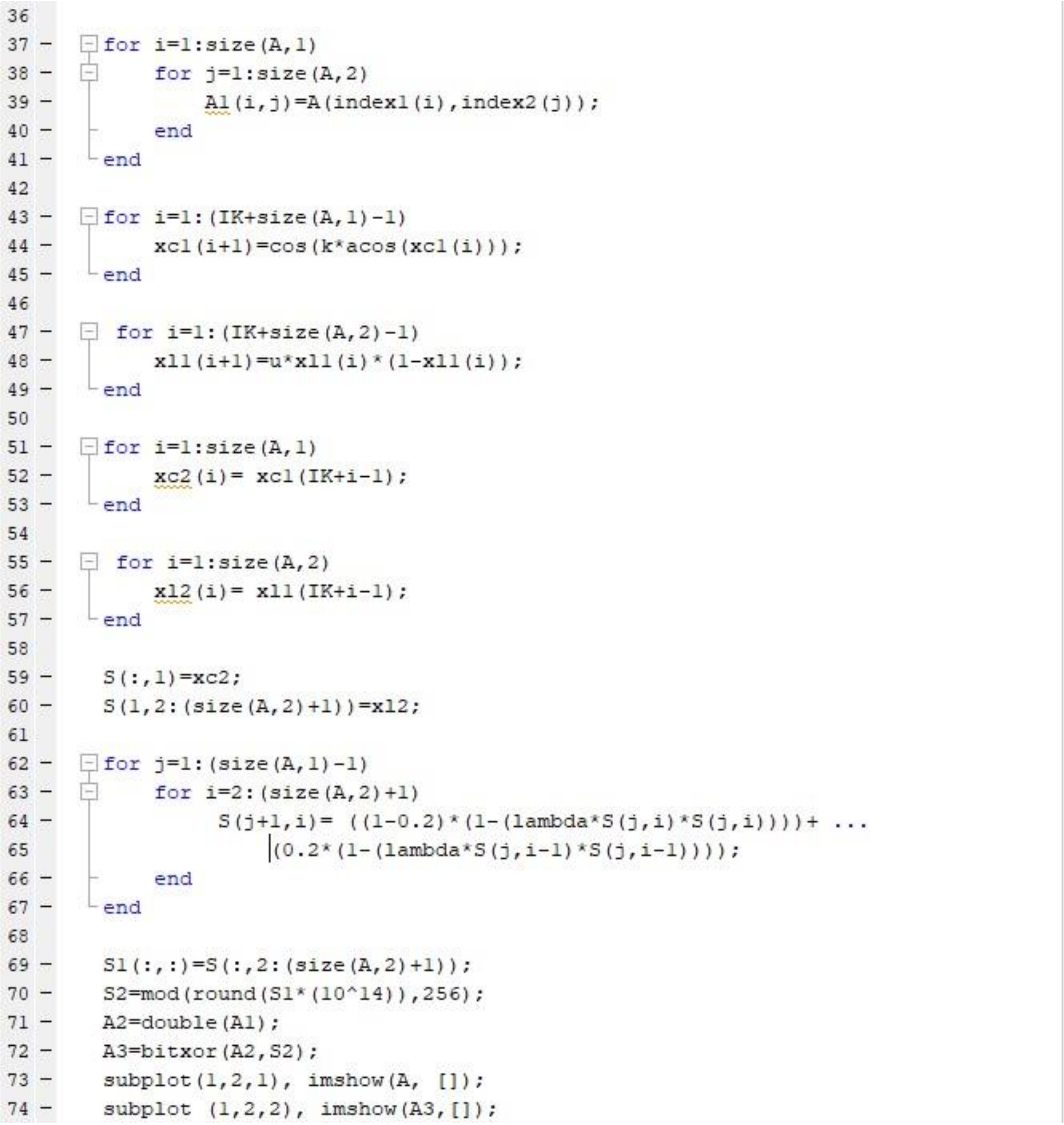
Where $\oplus$ is the logical XOR operation, bit-by-bit.



Original image (left) and the encrypted image (right)
(For this case, the correlation coefficient is 0.0035)



```
A=imread('G:\signature.png');
if size(A,3)==3
    A=rgb2gray(A);
end

xh=0.1;
yh=0.1;
lambda=2;
u=4;
xl=0.1;
k=5;
xc=0.1;
K=10001;
IK=mod(sum(sum(A)),K);
MK=IK/K;
xh1= xh*MK;
yh1= yh*MK;
xl1= xl*MK;
xc1= xc*MK;

for i=1:(max(size(A,1),size(A,2))+IK)
    xh1(i+1)=yh1(i)+1-(1.4*xh1(i)*xh1(i));
    yh1(i+1)=0.3*xh1(i);
end

for i=1:size(A,1)
    xh2(i)= xh1(IK+i-1);
end

 for i=1:size(A,2)
    yh2(i)= yh1(IK+i-1);
end

[temp,index1]=sort(abs(xh2));
[temp,index2]=sort(abs(yh2));
```



```
for i=1:size(A,1)
    for j=1:size(A,2)
        A1(i,j)=A(index1(i),index2(j));
    end
end

for i=1:(IK+size(A,1)-1)
    xc1(i+1)=cos(k*acos(xc1(i)));
end

 for i=1:(IK+size(A,2)-1)
    xl1(i+1)=u*xl1(i)*(1-xl1(i));
end

for i=1:size(A,1)
    xc2(i)= xc1(IK+i-1);
end

 for i=1:size(A,2)
    xl2(i)= xl1(IK+i-1);
end

S(:,1)=xc2;
S(1,2:(size(A,2)+1))=xl2;

for j=1:(size(A,1)-1)
    for i=2:(size(A,2)+1)
         S(j+1,i)= ((1-0.2)*(1-(lambda*S(j,i)*S(j,i))))+ ...
             (0.2*(1-(lambda*S(j,i-1)*S(j,i-1))));
    end
end

S1(:,:)=S(:,2:(size(A,2)+1));
S2=mod(round(S1*(10^14)),256);
A2=double(A1);
A3=bitxor(A2,S2);
subplot(1,2,1), imshow(A, []);
subplot (1,2,2), imshow(A3,[]);
```

Improvements to the Algorithm

We propose a robust improvement to the algorithm given in the previous section. As we know, the algorithm only works for 2-dimensional images. But in most practical applications, 3-dimensional images are used, eg. RGB images.

We propose a change in the step 6 of the previous algorithm. Instead of using two predefined maps (Chebyshev and Logistic) for OCML for pixel value shuffling, we add functionality to the encryption algorithm to automatically select two maps per layer from a collection of maps based on the image pixel values. This makes the key space larger as well.

We use six different maps for the algorithm to choose any two from:

1. Logistic Map

$$x_{n+1} = rx_n(1 - x_n)$$

2. Chebyshev Map

$$x_{n+1} = \cos(\alpha \cos^{-1}(x_n))$$

3. Gaussian Map

$$x_{n+1} = x_n e^{-\alpha x_n^2} + \beta$$

4. Circle Map

$$x_{n+1} = x_n + \Omega - \frac{K}{2\pi}\sin(2\pi\theta_n)$$

5. Bernoulli Map

$$x_{n+1} = 2x_n \; mod \; 1$$

6. Sine Map

$$x_{n+1} = \alpha \sin(\pi x_n)$$

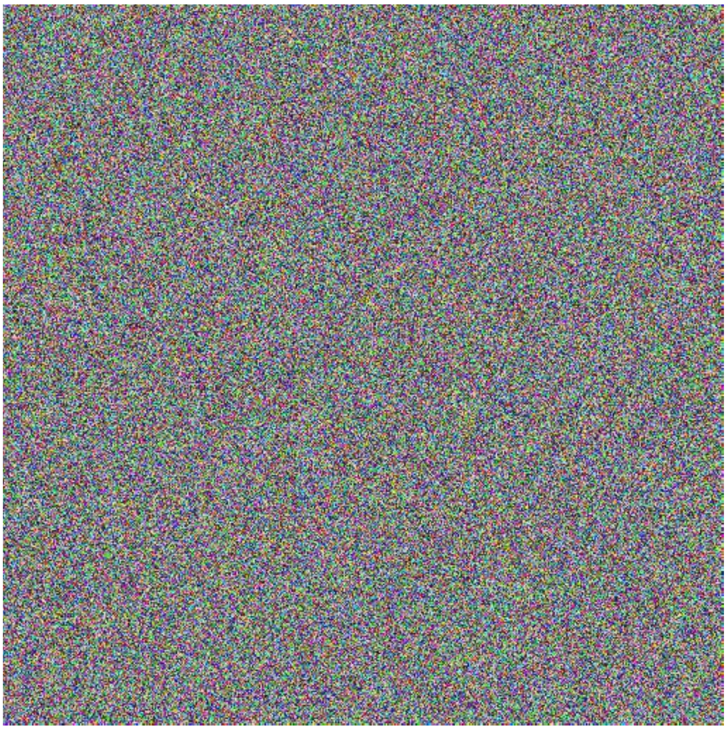
The sums of the pixel values in first two layers of the image are calculated separately and are stored for using in the key. The result of the sums modulus 6 is then used to choose the first pair of maps i.e., for the first layer.

For the second layer, we use the sums of digits in both sums modulus 6 for choosing the next pair of maps.

For the third layer, the two sums are converted to binary, and the sums of digits of the two binary numbers are then used to calculate the third pair of maps.

Then, for each layer, the OCML is generated as mentioned in the algorithm step 6, and the rest of steps are followed.

Original Image



Encrypted Image

Code for encryption

```
cd ('C:\Users\vaibh\Desktop');
A=imread('lena512color.jpg');
K=10001;
lambda=2;
IK=mod(sum(sum(sum(A))),K);
MK=IK/K;
S=zeros(size(A,1),size(A,2),3);
for i=1:2
    su(i)=sum(sum(A(:,:,i)));
end

 for k=1:2
        n=mod(su(k),6)+1;
        switch n
            case 1
                xl=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xl(i+1)=4*xl(i)*(1-xl(i));
                end
                for i=1:size(A,k)
                    x(1,i)= xl(IK+i-1);
                end
            case 2
                xc=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xc(i+1)=cos(5*acos(xc(i)));
                end
                for i=1:size(A,k)
                    x(2,i)= xc(IK+i-1);
                end
            case 3
                xg=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
                end
                for i=1:size(A,k)
                    x(3,i)= xg(IK+i-1);
                end
            case 4
                xl=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xl(i+1)=4*xl(i)*(1-xl(i));
                end
                for i=1:size(A,k)
                    x(4,i)= xl(IK+i-1);
                end
            case 5
                xb=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xb(i+1)=mod(2*xb(i),1);
                    if xb(i+1)==0
                        xb(i+1)=0.1*MK;
                    end
                end
                for i=1:size(A,k)
                    x(5,i)= xb(IK+i-1);
                end
```

```
           case 6
               xs=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xs(i+1)=0.9*sin(pi*xs(i));
               end
               for i=1:size(A,k)
                   x(6,i)= xs(IK+i-1);
               end
       end
               switch k
                   case 1
                       for i=1:size(A,1)
                           S(i,1,1)=x(n,i);
                       end
                   case 2
                       for i=1:size(A,1)
                           S(1,i+1,1)=x(n,i).';
                       end
               end
end

for k=1:2
       n=mod(sum(str2num(num2str(su(k)).')),6)+1;
       switch n
           case 1
               xl=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xl(i+1)=4*xl(i)*(1-xl(i));
               end
               for i=1:size(A,k)
                   x(1,i)= xl(IK+i-1);
               end
           case 2
               xc=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xc(i+1)=cos(5*acos(xc(i)));
               end
               for i=1:size(A,k)
                   x(2,i)= xc(IK+i-1);
               end
           case 3
               xg=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
               end
               for i=1:size(A,k)
                   x(3,i)= xg(IK+i-1);
               end
           case 4
               xl=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xl(i+1)=4*xl(i)*(1-xl(i));
               end
               for i=1:size(A,k)
                   x(4,i)= xl(IK+i-1);
               end
           case 5
               xb=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xb(i+1)=mod(2*xb(i),1);
                   if xb(i+1)==0
                       xb(i+1)=0.1*MK;
                   end
               end
               for i=1:size(A,k)
                   x(5,i)= xb(IK+i-1);
               end
           case 6
               xs=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xs(i+1)=0.9*sin(pi*xs(i));
               end
               for i=1:size(A,k)
                   x(6,i)= xs(IK+i-1);
               end
       end
               switch k
                   case 1
                       for i=1:size(A,1)
                           S(i,1,2)=x(n,i);
                       end
                   case 2
                       for i=1:size(A,2)
                           S(1,i+1,2)=x(n,i).';
                       end
               end
end

for k=1:2
       n=mod(sum(de2bi(su(k))),6)+1;
       switch n
           case 1
               xl=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xl(i+1)=4*xl(i)*(1-xl(i));
               end
               for i=1:size(A,k)
                   x(1,i)= xl(IK+i-1);
               end
           case 2
               xc=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xc(i+1)=cos(5*acos(xc(i)));
               end
               for i=1:size(A,k)
                   x(2,i)= xc(IK+i-1);
               end
           case 3
               xg=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
               end
               for i=1:size(A,k)
                   x(3,i)= xg(IK+i-1);
               end
           case 4
               xl=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xl(i+1)=4*xl(i)*(1-xl(i));
               end
```

```
                for i=1:size(A,k)
                    x(4,i)= xl(IK+i-1);
                end
            case 5
                xb=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xb(i+1)=mod(2*xb(i),1);
                    if xb(i+1)==0
                        xb(i+1)=0.1*MK;
                    end
                end
                for i=1:size(A,k)
                    x(5,i)= xb(IK+i-1);
                end
            case 6
                xs=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xs(i+1)=0.9*sin(pi*xs(i));
                end
                for i=1:size(A,k)
                    x(6,i)= xs(IK+i-1);
                end
        end
                switch k
                    case 1
                        for i=1:size(A,1)
                            S(i,1,3)=x(n,i);
                        end
                    case 2
                        for i=1:size(A,2)
                           S(1,i+1,3)=x(n,i).';
                        end
                end
 end

for k=1:3
    for j=1:(size(A,1)-1)
        for i=2:(size(A,2)+1)
            S(j+1,i,k)= ((1-0.2)*(1-(lambda*S(j,i,k)*S(j,i,k))))+ (0.2*(1-
(lambda*S(j,i-1,k)*S(j,i-1,k))));
        end
    end
    S1(:,:,k)=S(:,2:(size(A,2)+1),k);
end
S1=mod(round(S1*(10^14)),256);

xh=0.1*MK; yh=0.1*MK;
for i=1:(max(size(A,1),size(A,2))+IK)
    xh(i+1)=yh(i)+1-(1.4*xh(i)*xh(i));
    yh(i+1)=0.3*xh(i);
end
for i=1:size(A,1)
    xh1(i)= xh(IK+i-1);
end
 for i=1:size(A,2)
    yh1(i)= yh(IK+i-1);
end
[temp,index1]=sort(abs(xh1));
[temp,index2]=sort(abs(yh1));
for k=1:3
```

```
    for i=1:size(A,1)
        for j=1:size(A,2)
            A1(i,j,k)=A(index1(i),index2(j),k);
        end
    end
end

encrypt=bitxor(A1,uint8(S1));
imwrite(encrypt,'encrypt1.tiff');
imshow(encrypt,[]);
```

Decryption

The key is made up of the following components:

1. All the appropriate secret keys for the various maps.
2. MK and IK
3. Sums of the first two layers of the original image

Using the key, we recreate the S matrix as was done in the encryption process. Now, this S matrix and the encrypted image are operated on by a bit-XOR operator. The resultant matrix A_2 is the same as the one that was obtained after shuffling the pixels using the Henon map during the encryption process.

Now, using the key, the Henon map is reiterated and the index matrices are generated. Using these index matrices, A_2 is deshuffled back to the original image A.

Code for decryption

```
cd ('C:\Users\vaibh\Desktop');
A=imread('encrypt1.tiff');
K=10001;
lambda=2;
IK=1181;
MK=IK/K;
su=[16686096,9502365];

for k=1:2
        n=mod(su(k),6)+1;
        switch n
            case 1
                xl=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xl(i+1)=4*xl(i)*(1-xl(i));
                end
                for i=1:size(A,k)
                    x(1,i)= xl(IK+i-1);
                end
            case 2
                xc=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xc(i+1)=cos(5*acos(xc(i)));
                end
                for i=1:size(A,k)
                    x(2,i)= xc(IK+i-1);
                end
            case 3
                xg=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
                end
                for i=1:size(A,k)
                    x(3,i)= xg(IK+i-1);
                end
            case 4
                xl=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xl(i+1)=4*xl(i)*(1-xl(i));
                end
                for i=1:size(A,k)
                    x(4,i)= xl(IK+i-1);
                end
            case 5
                xb=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xb(i+1)=mod(2*xb(i),1);
                    if xb(i+1)==0
                        xb(i+1)=0.1*MK;
                    end
                end
                for i=1:size(A,k)
                    x(5,i)= xb(IK+i-1);
                end
            case 6
                xs=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xs(i+1)=0.9*sin(pi*xs(i));
```

```
               end
               for i=1:size(A,k)
                   x(6,i)= xs(IK+i-1);
               end
       end
               switch k
                   case 1
                       for i=1:size(A,1)
                           S(i,1,1)=x(n,i);
                       end
                   case 2
                       for i=1:size(A,1)
                           S(1,i+1,1)=x(n,i).';
                       end
               end
end

for k=1:2
       n=mod(sum(str2num(num2str(su(k)).')),6)+1;
       switch n
           case 1
               xl=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xl(i+1)=4*xl(i)*(1-xl(i));
               end
               for i=1:size(A,k)
                   x(1,i)= xl(IK+i-1);
               end
           case 2
               xc=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xc(i+1)=cos(5*acos(xc(i)));
               end
               for i=1:size(A,k)
                   x(2,i)= xc(IK+i-1);
               end
           case 3
               xg=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
               end
               for i=1:size(A,k)
                   x(3,i)= xg(IK+i-1);
               end
           case 4
               xl=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xl(i+1)=4*xl(i)*(1-xl(i));
               end
               for i=1:size(A,k)
                   x(4,i)= xl(IK+i-1);
               end
           case 5
               xb=0.1*MK;
               for i=1:(IK+size(A,k)-1)
                   xb(i+1)=mod(2*xb(i),1);
                   if xb(i+1)==0
                       xb(i+1)=0.1*MK;
                   end
               end
```

```
                for i=1:size(A,k)
                    x(5,i)= xb(IK+i-1);
                end
            case 6
                xs=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xs(i+1)=0.9*sin(pi*xs(i));
                end
                for i=1:size(A,k)
                    x(6,i)= xs(IK+i-1);
                end
        end
                switch k
                    case 1
                        for i=1:size(A,1)
                            S(i,1,2)=x(n,i);
                        end
                    case 2
                        for i=1:size(A,2)
                            S(1,i+1,2)=x(n,i).';
                        end
                end
end

for k=1:2
        n=mod(sum(de2bi(su(k))),6)+1;
        switch n
            case 1
                xl=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xl(i+1)=4*xl(i)*(1-xl(i));
                end
                for i=1:size(A,k)
                    x(1,i)= xl(IK+i-1);
                end
            case 2
                xc=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xc(i+1)=cos(5*acos(xc(i)));
                end
                for i=1:size(A,k)
                    x(2,i)= xc(IK+i-1);
                end
            case 3
                xg=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
                end
                for i=1:size(A,k)
                    x(3,i)= xg(IK+i-1);
                end
            case 4
                xl=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xl(i+1)=4*xl(i)*(1-xl(i));
                end
                for i=1:size(A,k)
                    x(4,i)= xl(IK+i-1);
                end
            case 5
```

```
                xb=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xb(i+1)=mod(2*xb(i),1);
                    if xb(i+1)==0
                        xb(i+1)=0.1*MK;
                    end
                end
                for i=1:size(A,k)
                    x(5,i)= xb(IK+i-1);
                end
            case 6
                xs=0.1*MK;
                for i=1:(IK+size(A,k)-1)
                    xs(i+1)=0.9*sin(pi*xs(i));
                end
                for i=1:size(A,k)
                    x(6,i)= xs(IK+i-1);
                end
        end
                switch k
                    case 1
                        for i=1:size(A,1)
                            S(i,1,3)=x(n,i);
                        end
                    case 2
                        for i=1:size(A,2)
                           S(1,i+1,3)=x(n,i).';
                        end
                end
 end

 for k=1:3
    for j=1:(size(A,1)-1)
        for i=2:(size(A,2)+1)
            S(j+1,i,k)= ((1-0.2)*(1-(lambda*S(j,i,k)*S(j,i,k))))+ (0.2*(1-
(lambda*S(j,i-1,k)*S(j,i-1,k))));
        end
    end
    S1(:,:,k)=S(:,2:(size(A,2)+1),k);
 end

 S1=mod(round(S1*(10^14)),256);

 xh=0.1*MK; yh=0.1*MK;
for i=1:(max(size(A,1),size(A,2))+IK)
    xh(i+1)=yh(i)+1-(1.4*xh(i)*xh(i));
    yh(i+1)=0.3*xh(i);
end
for i=1:size(A,1)
    xh1(i)= xh(IK+i-1);
end
 for i=1:size(A,2)
    yh1(i)= yh(IK+i-1);
end
[temp,index1]=sort(abs(xh1));
[temp,index2]=sort(abs(yh1));

 A1=bitxor(A,uint8(S1));
```

```
for k=1:3
    for i=1:size(A,1)
        for j=1:size(A,2)
            decrypt1(index1(i),index2(j),k)=A1(i,j,k);
        end
    end
end
imshow(decrypt1,[]);
```

Conclusion

After revisiting selected topics in topology, I studied about the topological dynamics. I studied the main ideas in dynamical systems along with examples.

I applied the knowledge gained about chaotic systems in the implementation of a robust image encryption algorithm. The algorithm uses several chaotic maps to increase the keyspace, make the algorithm more robust and to provide more randomness. It was implemented in MATLAB. The original algorithm was improved upon by adding a collection of 1-dimensional chaotic maps from which the algorithm automatically chooses a pair of maps to construct the OCML.

I explored the ideas of dynamical systems in the context of differential equations and linear algebra, and studied about calculating solutions, flows and the stable manifold theorem.

I feel confident about my knowledge in the topics, and am sure that I would be able to use the knowledge gained during this project in my future endeavors.

References

- Brin, Michael, and Garrett Stuck. Introduction to dynamical systems. Cambridge university press, 2002.

- Pareek, Narendra K., Vinod Patidar, and Krishan K. Sud. "Image encryption using chaotic logistic map." Image and vision computing 24, no. 9 (2006): 926-934

- Zheng, Yifeng, and Jianxiu Jin. "A novel image encryption scheme based on Hénon map and compound spatiotemporal chaos." Multimedia Tools and Applications 74, no. 18 (2015): 7803-7820.

- Sharan Gopal, Chaos and its ingredients, Proceedings of AP Academy of Sciences, Vol. 14, No. 2 (2012) 73-92. Published by AP Academy of Sciences.

- Perko, L., 2013. Differential equations and dynamical systems (Vol. 7). Springer Science & Business Media.