UNIVERSITI TEKNOLOGI MARA

PROJECT MATRIX THEORY (MAT 723)

SIERPINSKI FRACTALS

GROUP 1

NUR IMAN NAZIRAH BINTI NASIR (2018469592)

USWATI ARWAEKAJI BINTI ABDULLAH (201887154)

NURUL AINA SYAFIQAH ZULKAFLI (2018632812)

DUE DATE:18JANUARY 2019

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

INTRODUCTION

Many people are fascinated by the lovely pictures called fractals. In addition to the typical perception of mathematics as a body of complicated, long-lasting formulas, fractal geometry combines art with mathematics to show that equations are more than just a collection of numbers. The fact that fractals are the best mathematical descriptions of many natural forms, such as coastlines, mountains or parts of living organisms, is even more interesting.

It is very interesting to study the properties of general Sierpinski fractals, including dimensions, measurements, Lipschitz equivalence, etc. Like other fractals, general Sierpinski fractals are so complicated and irregular that they are hopelessly modelled using classical geometry objects only.

While fractal geometry is closely linked to computer techniques, some people worked on fractals long before computers were invented. These people were British cartographers who came up against the problem of measuring the length of the Britain coast. On a large-scale map, the coastline measured was about half the length of the coastline measured on a detailed map. The closer you looked, the longer and more detailed the coastline became. They didn't realize they had found one of the main characteristics of fractals.

Geometric modelling of fractal objects plays an important role not only in the study of fractal theory but also in computer graphics. Computer simulation is based on Fractal 's basic theory. However, drawing strange and amazing fractal images by computer can provide us with the most intuitive discussion and explanation at the same time, which greatly encourages the development of fractal theory. Computer graphics provide many algorithms to generate fractal images, such as IFS (Iterated Function System), L- system, time-scaped recursive algorithm, etc. There are different source programs based on each algorithm. To our knowledge, however, most algorithms and source programs are intended for graphic purposes and are often carried out by non- mathematical investigators (Zhu, 2017).

OBJECTIVE

This project is mainly focus on fractals which is one of the topics in Matrix Theory. Thus, the objectives of this project are as follows:

i) to introduce the fractal concept by means of a classic example, as the Sierpinski carpet.

ii) to familiarize the student with his / her self-similarity.

iii) to develop manual and visual activity.

iv) to encourage interchange of knowledge and experiences between the participants of the project, in the social networks, as well as in e-Twinning.

v) to highlight the cooperative work, and positive interdependence, as a way of getting a sizeable construction.

SIERPINSKI CARPET

i. Geometric description

Begin with solid(filled) C(0) square. Divide into nine smaller squares. Remove the center square interior(i.e. do not remove the border) to get C(1). Now divide each of the remaining eight solid squares into nine congruent squares and remove the center square for C(2) from each one. Continue repeating the construction to achieve a decreasing set sequence

$$C(0)\supset C(1)\supset C(2)\supset C(3)\supset\cdots C(0)\supset C(1)\supset C(2)\supset C(3)\supset\cdots$$

The Sierpinski carpet is the intersection of all sets in this sequence, i.e. the set of points that remain after this construction is infinitely often repeated. The following figures show the first four iterations. The red squares represent some of the smaller congruent squares used in building.

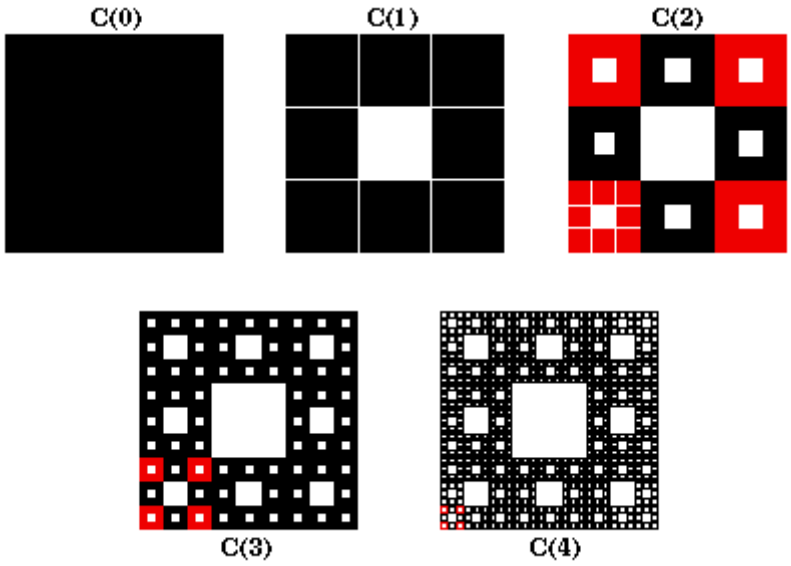


Figure 1: The first four iteration of Sierpinski carpet

ii. IFS Transformations

The original square has a r=1/3 factor. This is done eight times, followed by the necessary translations to arrange the eight squares as shown in C(1). If we consider the original square to be a unit square with opposite corners at (0,0) and (1,1), the following functions would be provided to the IFS.

$$f_1(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x$$

$$f_2(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/3 \end{bmatrix}$$

$$f_3(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 0 \\ 2/3 \end{bmatrix}$$

$$f_4(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 1/3 \\ 0 \end{bmatrix}$$

$$f_5(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$$

$$f_6(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 2/3 \\ 0 \end{bmatrix}$$

$$f_7(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix}$$

$$f_8(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 2/3 \\ 2/3 \end{bmatrix}$$

The Sierpinski carpet consists of eight pieces that are similar to the eight functions in the iterated system. We apply the IFS transformation into Matlab

Coding IFS

```
clear all
iterations=100000;
%the number of iterations
% recall, the transformations are of the form A*[x,y]+t
A1=[1/3 0 ; 0 1/3];
A2=[1/3 0 ; 0 1/3];
A3=[1/3 0 ; 0 1/3];
A4=[1/3 0 ; 0 1/3];
A5=[1/3 0 ; 0 1/3];
A6=[1/3 0 ; 0 1/3];
A7=[1/3 0 ; 0 1/3];
A8=[1/3 0 ; 0 1/3];

t1=[0 ; 0];
t2=[0 ; 1/3];
t3=[0 ; 2/3];
t4=[1/3 ; 0];
t5=[1/3 ; 2/3];
t6=[2/3 ; 0];
t7=[2/3 ; 1/3];
t8=[2/3 ; 2/3];


% probabilities with which each transformation is applied
% notice that p1+p2+p3+p4+p5+p6=1
p1=1/8;
p2=1/8;
p3=1/8;
p4=1/8;
p5=1/8;
p6=1/8;
p7=1/8;
p8=1/8;

% the initial point
x(1)=0;
y(1)=0;
% (x,y) points as a vector, v
v=[x(1);y(1)]; % here is the intial point
for n=2:iterations
% choose a random number, k, between 0 and 1
k=rand;
% depending on your random number ...
% do one of the four transformations to get a new point
if k<p1
v=A1*v+t1;
elseif k<p1+p2
v=A2*v+t2;
elseif k<p1+p2+p3
v=A3*v+t3;
elseif k<p1+p2+p3+p4
v=A4*v+t4;
elseif k<p1+p2+p3+p4+p5
v=A5*v+t5;
elseif k<p1+p2+p3+p4+p5+p6
v=A6*v+t6;
```

```
elseif k<p1+p2+p3+p4+p5+p6+p7
v=A7*v+t7;
else
v=A8*v+t8;
end
%(x,y) point as elements of the vector v
x(n)=v(1);
y(n)=v(2);

end
opengl software
hold off
% plot graph
plot(x,y,'square','Color',[0.45 0.67 0.75])
title('Sierpinski's carpet')
hold on
axis(x);
set(gcf, 'color','white');
```

Output IFS Sierpinski Carpet

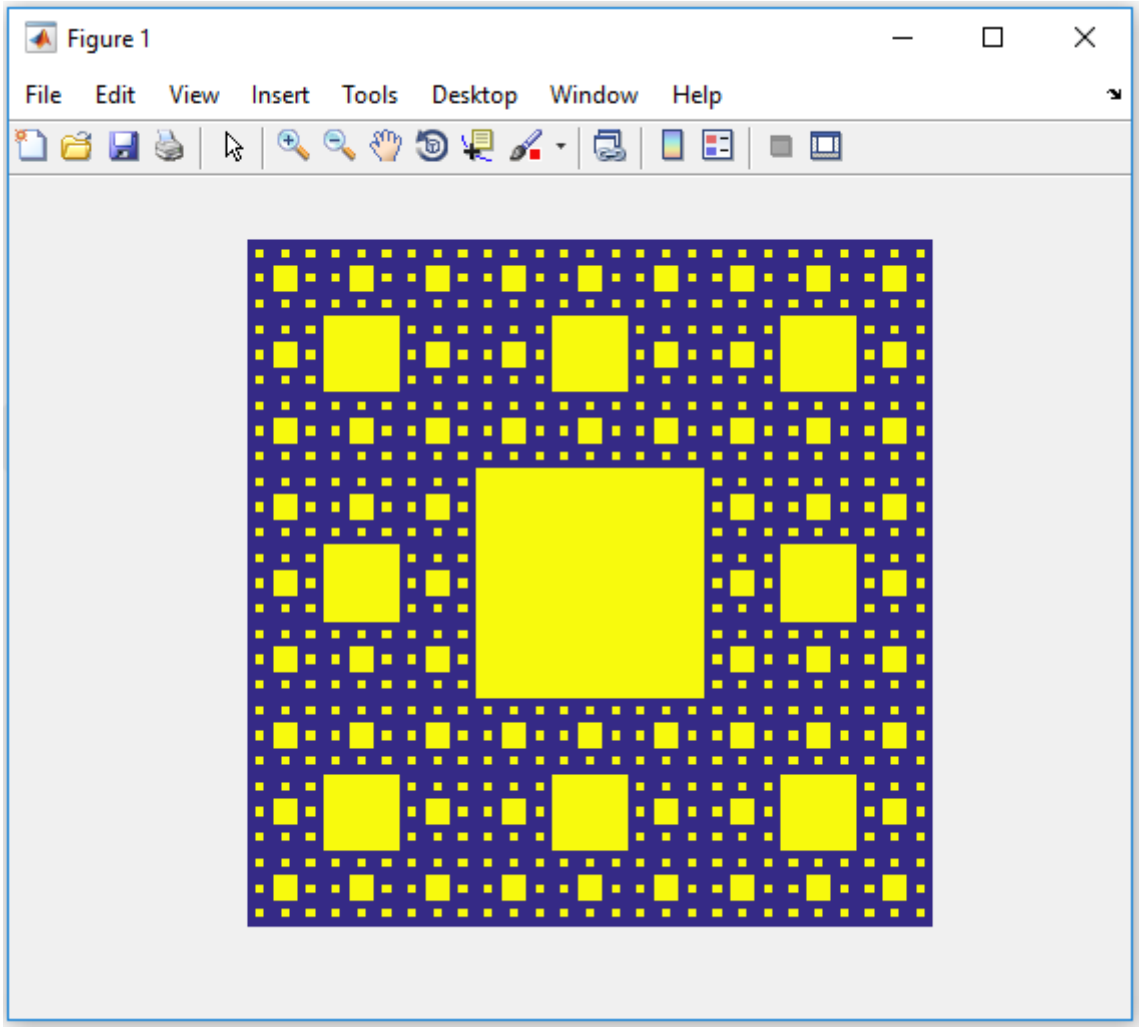


Figure 2: Output for Sierpinski Carpet

iii. Lindenmayer System (L-system)

The code for generating the fractal via a Lindenmayer system (L-system)

Angle 90

Axiom F

F = F+F−F−F−G+F+F+F−F

G= GGG

This results in a 45 ° rotation of the Sierpinski carpet. In addition, there are lines through the " holes " square. To avoid these lines, you can change the L-system to lift the pen by using

F = F+F-F-F-G+F+F+F-F

where F is moving forward by line length drawing a line, + means turn left by turning angle, - means turn right by turning angle. Below are the results using L-system.

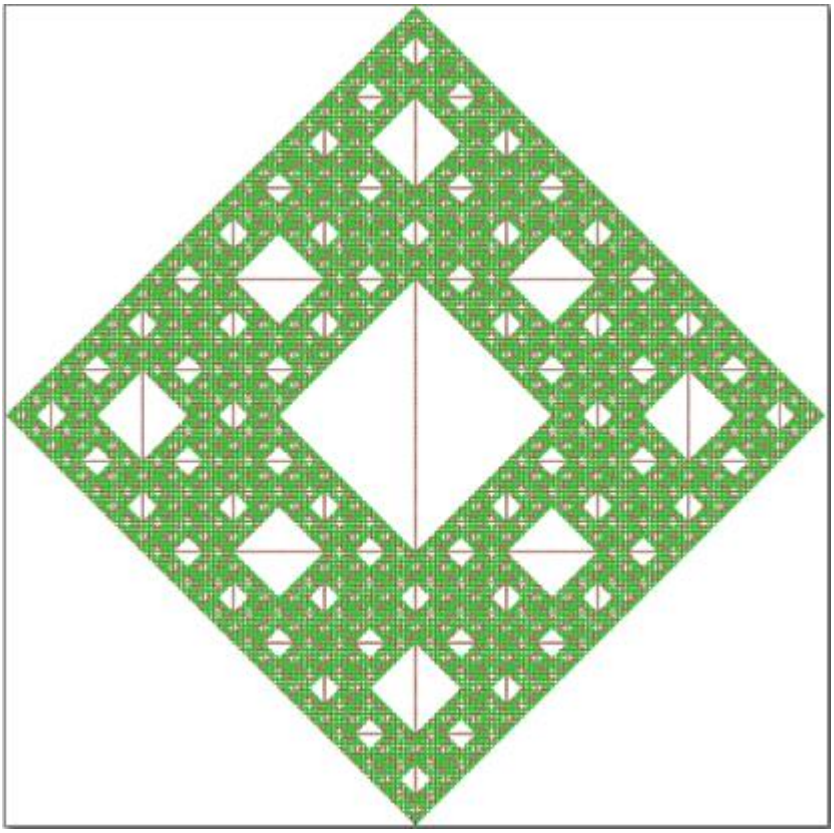


Figure 3: Result of Sierpinski Carpet using L-System

iv. Similarity Dimension

The Sierpinski carpet is similar to 8 non-overlapping copies, each with a factor r<1. The similarity dimension, d, of the unique attractor of the IFS is therefore the solution.

$$\sum_{k=1}^{8} r^{d} = 1 \Rightarrow d = \frac{\log(1/8)}{\log(r)} = \frac{\log(1/8)}{\log(1/3)} = \frac{\log(8)}{\log(3)} = 1.89279$$

v. Special Properties

Assume the area of original C(0) square is equal to 1. In order to get C(k+1), we scale C(k) by 1/3, reducing the area by 1/9 = 1/3. But we make 8 copies of C(k+1) of this scaled version. The area of C(k+1) must therefore be (8/9) of the area of C(k). That means that C(n) area is (8/9) for all n.

Notice that these areas reach 0 while n reaches infinity. This means that in the construction of the Sierpinsky carpet we have removed " all " from the original square area. But there are still many points in the carpet. This is one reason why this set doesn't have a useful dimension.

Sierpinski used the carpet to catalogue all compact one-dimensional objects in the plane (from a topological point of view). What this basically means is the Sierpinski carpet contains a topologically equivalent copy of any compact one-dimensional object in the plane. Thus for any curve contained in the plane, there is a set homeomorphic to the Sierpinski carpet that contains the curve.

In an article written by (Ciesielski & Pogoda, 1996): Note that as early as one year ago Mr. Mazurkiewicz found an example of a curve which was simultaneously a Jordan curve and a Cantor curve...Mazurkiewicz forms this curve by dividing the square into nine smaller squares using lines parallel to the sides and removing the interior of the center square, performing the same procedure on each of the remaining eight square, and iterating this procedure ad infinitum.

So the Sierpinski carpet was actually invented by Stefan Mazurkiewicz, who in 1913 wrote his Ph.D. thesis under the supervision of Sierpinski on curves filling a square.

SIERPINSKI FISH

i. Geometric description

Start with a solid (filled) square F(0). Divide this into 9 smaller congruent squares. Remove interior of the center square and the last square to get F(1). Now, subdivide each of the seven remaining solid squares into 9 congruent squares and remove the center square and last square to obtain F(2). Continue to repeat the construction to obtain a decreasing sequence of sets

$$F(0) \supset F(1) \supset F(3) \ldots$$

The Sierpinski Fish is the intersection of all the sets in this sequence, that is, the set of points that remain after this construction is repeated infinitely often. The figures below show the first four iterations.

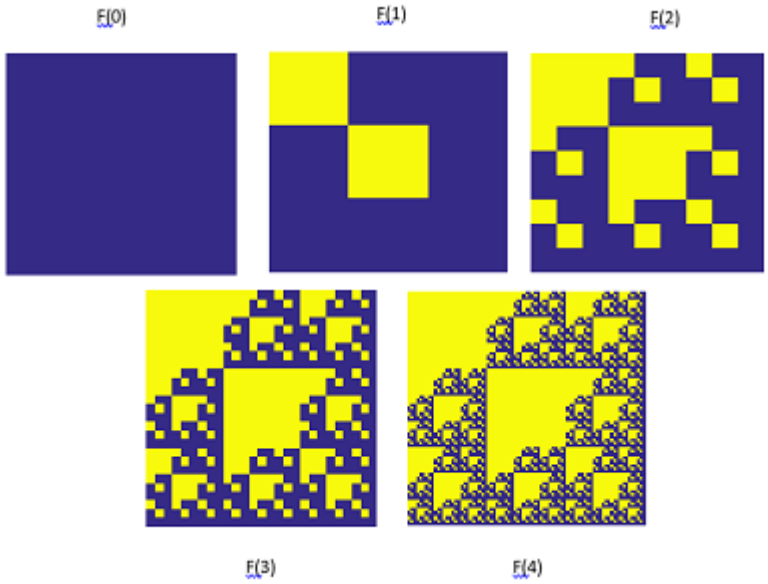


Figure 4: The first four iteration of Sierpinski Fish

ii. IFS Transformations

The original square has a r=1/3 factor. This is done seven times, followed by the necessary translations to arrange the eight squares as shown in C(1). If we consider the original square to be a unit square with opposite corners at (0,0) and (1,1), the following functions would be provided to the IFS.

$$f_1(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x$$

$$f_2(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/3 \end{bmatrix}$$

$$f_3(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 1/3 \\ 0 \end{bmatrix}$$

$$f_4(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$$

$$f_5(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix}$$

$$f_6(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 2/3 \\ 2/3 \end{bmatrix}$$

$$f_7(x) = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/3 \end{bmatrix} x + \begin{bmatrix} 2/3 \\ 0 \end{bmatrix}$$

The Sierpinski fish consists of seven pieces that are similar to the seven functions in the iterated system. We apply the IFS transformation into Matlab

Coding IFS

```
clear all;clc();
iterations=100000; %the number of iterations
% recall, the transformations are of the form A*[x,y]+t
% and, there are seven such transformations ...
A1=[1/3 0 ;0 1/3];
A2=[1/3 0 ;0 1/3];
A3=[1/3 0 ;0 1/3];
A4=[1/3 0 ;0 1/3];
A5=[1/3 0 ;0 1/3];
A6=[1/3 0 ;0 1/3];
A7=[1/3 0 ;0 1/3];

t1=[0 ; 0];
t2=[0; 1/3];
t3=[1/3; 0];
t4=[1/3 ; 2/3];
t5=[2/3; 1/3];
t6=[2/3 ; 2/3];
t7=[2/3 ; 0];

%the probabilities with which each transformation is applied
% notice that p1+p2+p3+p4+p5+p6+p7=1
p1=1/7;
p2=1/7;
p3=1/7;
p4=1/7;
p5=1/7;
p6=1/7;
p7=1/7;

% the initial point
x(1)=0;
y(1)=0;
% but, let's write the (x,y) points as a vector, v
v=[x(1);y(1)]; % here is the intial point
for n=2:iterations
% choose a random number, k, between 0 and 1
k=rand;
% depending on your random number ...
% do one of the seven transformations to get a new point
if k<p1
v=A1*v+t1;
elseif k<p1+p2
v=A2*v+t2;
elseif k<p1+p2+p3
v=A3*v+t3;
elseif k<p1+p2+p3+p4
v=A4*v+t4;
elseif k<p1+p2+p3+p4+p5
v=A5*v+t5 ;
elseif k<p1+p2+p3+p4+p5+p6
v=A6*v+t6 ;
else
 v=A7*v+t7 ;
end
```

```
%(x,y) point as elements of the vector v
x(n)=v(1);
y(n)=v(2);
end
opengl software
hold off
% plot graph
plot(x,y,'.','Color',[0.1 0.3 0])
hold on
axis('equal','off');
set(gcf, 'color','white');
```

Output IFS Sierpinski Fish

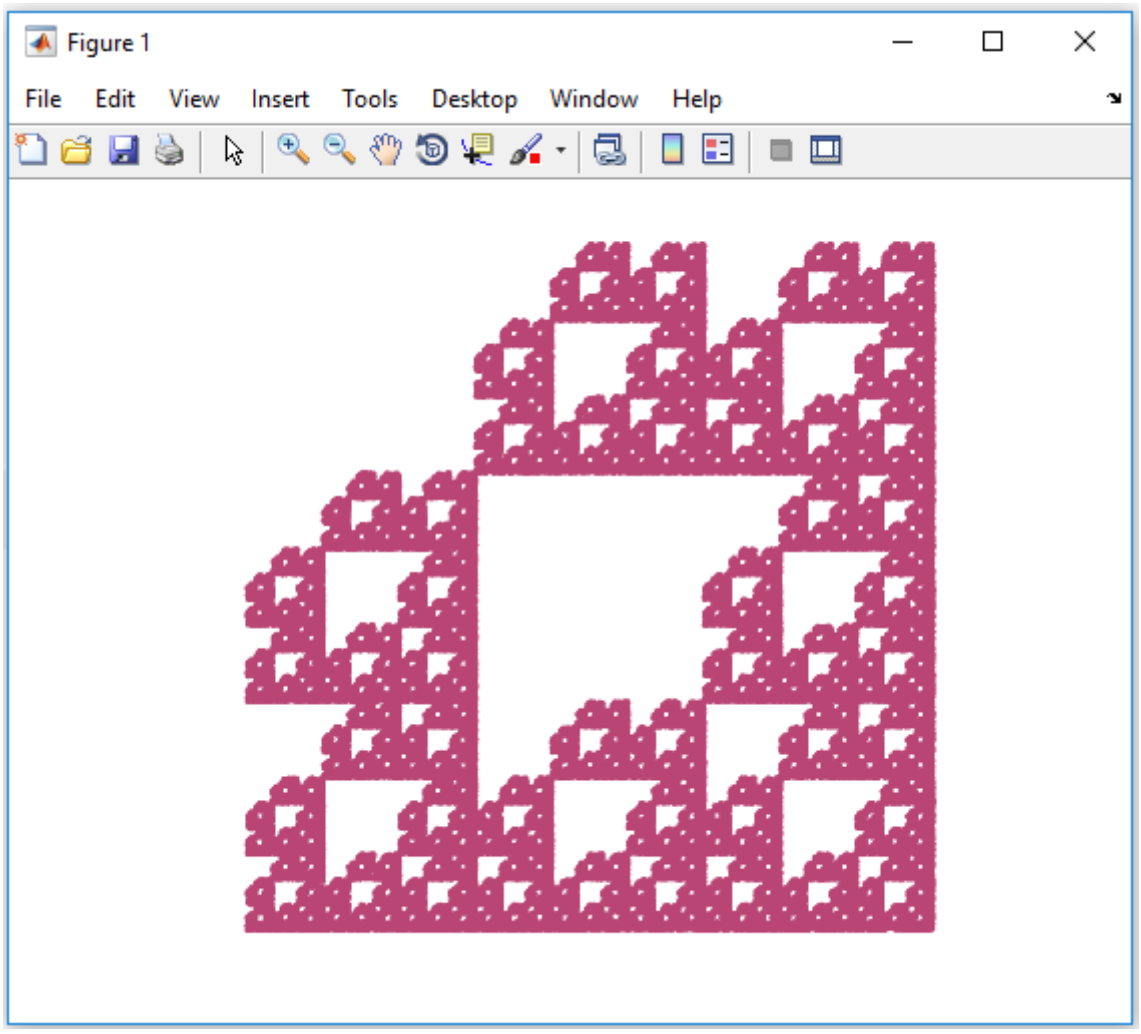


Figure 5: Output for Sierpinski Fish

iii. Lindenmayer System (L-system)

The code for generating the fractal via a Lindenmayer system (L-system)

Angle 90

Axiom F-F

F = C1FF-F-C0F-F-FF

This results in a reverse of starting shape and the inverse direction with the Sierpinski fish. Below are the results using L-system.

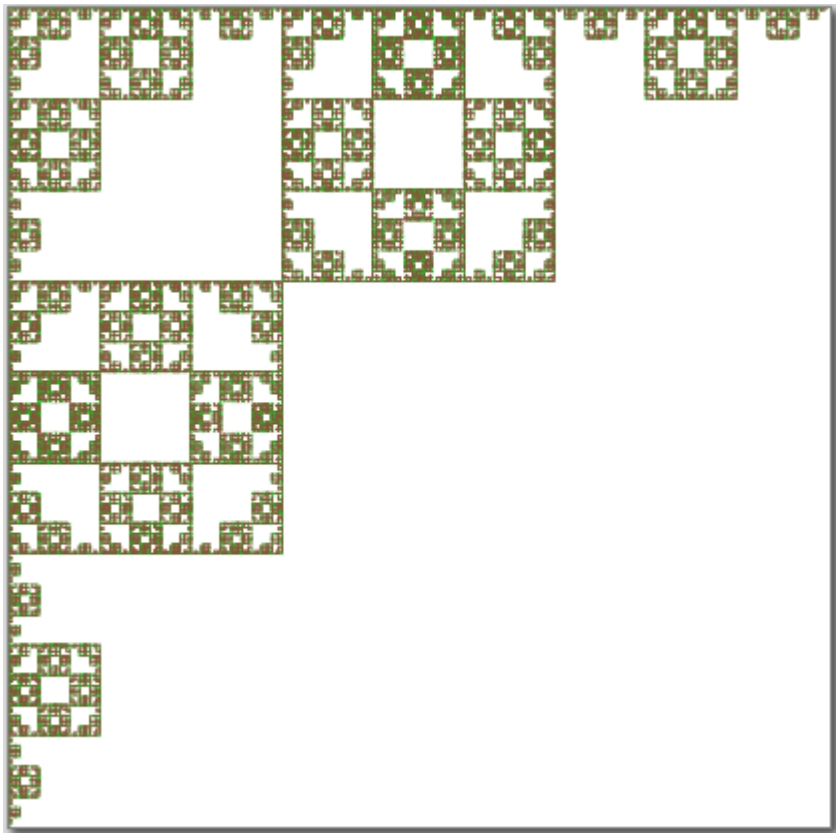


Figure 6: Result of Sierpinski Fish using L-System

iv. Similarity Dimension

The Sierpinski fish is similar to 7 non-overlapping copies, each with a factor r<1. The similarity dimension, d, of the unique attractor of the IFS is therefore the solution.

$$\sum_{k=1}^{7} r^{d} = 1 \Rightarrow d = \frac{\log(1/7)}{\log(r)} = \frac{\log(1/7)}{\log(1/3)} = \frac{\log(7)}{\log(3)} = 1.77124$$

v. Special Properties

Assume the area of original F(0) square is equal to 1. In order to get F(k+1), we scale F(k) by 1/3, reducing the area by 1/9 = 1/3. But we make 7 copies of F(k+1) of this scaled version. The area of F(k+1) must therefore be (7/9) of the area of F(k). That means that F(n) area is (8/9) for all n.

Notice that these areas reach 0 while n reaches infinity. This means that in the construction of the Sierpinsky fish we have removed " all " from the original square area. But there are still many points in the fish. This is one reason why this set doesn't have a useful dimension.

Sierpinski used the fish to catalogue all compact one-dimensional objects in the plane (from a topological point of view). What this basically means is the Sierpinski fish contains a topologically equivalent copy of any compact one-dimensional object in the plane. Thus for any curve contained in the plane, there is a set homeomorphic to the Sierpinski fish that contains the curve.

DISCUSSION

In this section , we briefly discuss the results that we were obtained. The fractal that we have made is most likely a fish shape. The name of the fractal we obtained is Sierspinski Fish. Sierspinski Fish is based on the implementation from the basic concept of Sierspinski carpet. As you can see that the scale factor in Iterated Funstion System (IFS) method of Sierspinski Fish is also the same as Sierspinski Carpet which is 1/3. We set 1/3 as scale factor for Sierspinski Fish because we have done some observation with different value of scale factor and the result that was obtained is not satisfying our fractal shape. The repetitions and the similitudes for Sierspinski Fish is not the same as Sierspinski Carpet. By referring the IFS of both fractals, note that the similitudes in Sierspinski carpet are $\binom{0}{0}, \binom{0}{\frac{1}{3}}, \binom{0}{\frac{2}{3}}, \binom{\frac{1}{3}}{0}, \binom{\frac{1}{3}}{\frac{2}{3}}, \binom{\frac{2}{3}}{0}, \binom{\frac{2}{3}}{\frac{1}{3}}, \binom{\frac{2}{3}}{\frac{2}{3}}$ whereas similitudes of Sierspinski Fish are $\binom{0}{0}, \binom{0}{\frac{1}{3}}, \binom{\frac{1}{3}}{0}, \binom{\frac{1}{3}}{\frac{2}{3}}, \binom{\frac{2}{3}}{\frac{1}{3}}, \binom{\frac{2}{3}}{\frac{1}{3}}, \binom{\frac{2}{3}}{0}$. It is clearly that the Sierspinski carpet have 8 similitudes while Sierspinski Fish have 7 similitudes. In order to get the Sierspinski Fish, we apply the IFS method with 100000 iterations and the result is obtained in Figure 5. Then, we construct the code of L-system so that we can make the comparison between these two method. In Figure 6, we can see that the result is not exactly as in Figure 5. However, we notice that the shape in Figure 6 especially the white area in the large region is the same as the shape in Figure 5. Due to the limitation of time, it is very difficult to do the L-System to get our fractal. The L-system is very tedious to do, hence we strongly suggest to use the IFS Transformation Method. Since our fractal is require 100000 iterations , hence we need to use computer software such as Matlab for computing the shape of our fractal. Once we know the algorithm for IFS transformation, it is easier for us to write the coding of this method.

CONCLUSION

Fractals are the best mathematical descriptions of various natural forms, such as coastlines, mountains or parts of living organisms, and even more fascinating. In this project, we able to understand the background of theory in fractal such as the classic example of fractal. Besides that, the application of Matrix Theory is very interesting to learn since we know the fractals are mostly made up based on the matrix form. In addition, the application of Matrix Theory especially in Fractals require the creativity of person to create a new fractal. For instance, Michael Barnsley developed a matrix for attaining the ferns. The fractals are everywhere and most of the researchers interest to develop a new method for attaining fractal. With the knowledge of Matrix Theory we able to develop manual and visual activity for earning a new fractal. We have been observe that some of the researcher obtained the same fractals like others but their method of findings is different. Eventhough the fractal can made up from any method, however when the shape of fractals is very complex, it is require to use the suitable method. All of these theories are very useful. Once we know the theory, we will be able to compute the fractals by using the algorithm that we obtain. The computer software is very effective when the problem require to solve a lot of iterations.

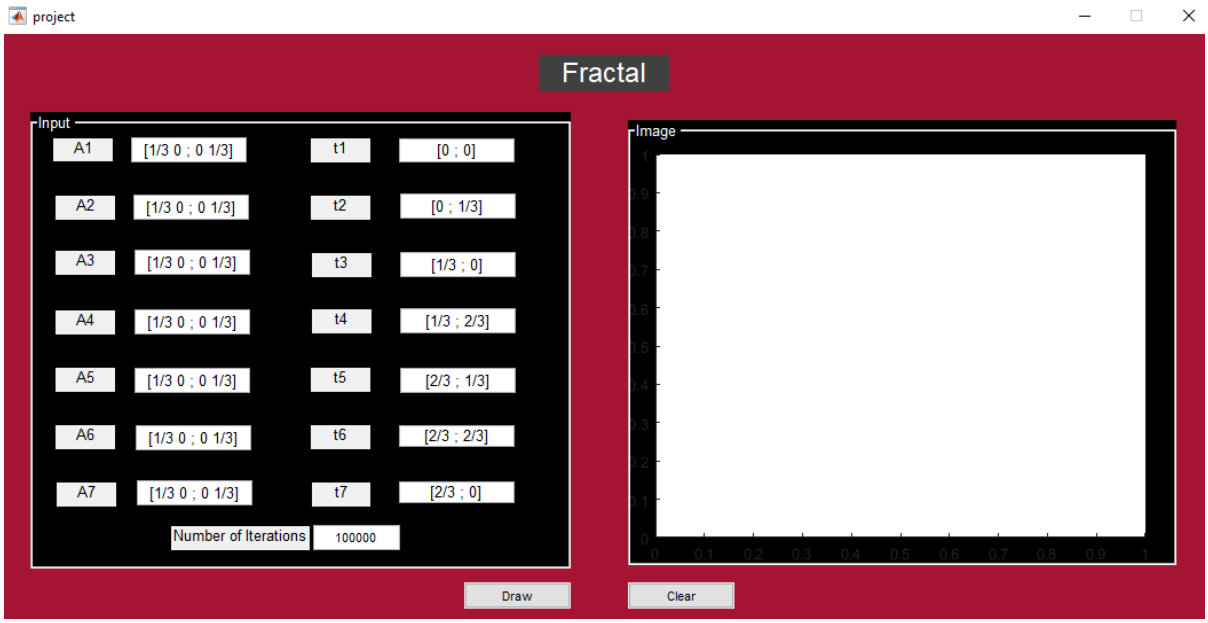
REFERENCE

Ciesielski, K., & Pogoda, Z. (1996). The Beginning of Polish Topology, *18*(3).

Zhu, Z. (2017). Geometric Modelling of General Sierpinski Fractals using iterated function system in Matlab, *14*, 43–57.

APPENDIX

We create GUI version in MATLAB so that the user be able to see the difference of input values will effect the image obtain. The user shall be able to input the value of Matrix A, Matrix t and also the number of iterations. The figure below shows the Graphical User Interface. Please note that the value of Matrix A and Matrix t are come from the IFS Transformation for Sierspinski Fish. The user can see when the number of iterations is less the output of image is not clearly shows the Sierspinski Fish.



Below these are the coding in Matlab for constructing GUI.

```
function varargout = project(varargin)
% PROJECT MATLAB code for project.fig
%      PROJECT, by itself, creates a new PROJECT or raises the existing
%      singleton*.
%
%      H = PROJECT returns the handle to a new PROJECT or the handle to
%      the existing singleton*.
%
%      PROJECT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJECT.M with the given input arguments.
%
%      PROJECT('Property','Value',...) creates a new PROJECT or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before project_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to project_OpeningFcn via varargin.
```

```
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help project

% Last Modified by GUIDE v2.5 17-Jan-2019 10:27:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @project_OpeningFcn, ...
                   'gui_OutputFcn',  @project_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before project is made visible.
function project_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to project (see VARARGIN)

% Choose default command line output for project
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes project wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = project_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function gA1_Callback(hObject, eventdata, handles)
% hObject    handle to gA1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA1 as text
%        str2double(get(hObject,'String')) returns contents of gA1 as a
double


% --- Executes during object creation, after setting all properties.
function gA1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gA2_Callback(hObject, eventdata, handles)
% hObject    handle to gA2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA2 as text
%        str2double(get(hObject,'String')) returns contents of gA2 as a
double


% --- Executes during object creation, after setting all properties.
function gA2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gA3_Callback(hObject, eventdata, handles)
% hObject    handle to gA3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA3 as text
```

```
%        str2double(get(hObject,'String')) returns contents of gA3 as a
double


% --- Executes during object creation, after setting all properties.
function gA3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gA4_Callback(hObject, eventdata, handles)
% hObject    handle to gA4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA4 as text
%        str2double(get(hObject,'String')) returns contents of gA4 as a
double


% --- Executes during object creation, after setting all properties.
function gA4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gA5_Callback(hObject, eventdata, handles)
% hObject    handle to gA5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA5 as text
%        str2double(get(hObject,'String')) returns contents of gA5 as a
double


% --- Executes during object creation, after setting all properties.
function gA5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gA6_Callback(hObject, eventdata, handles)
% hObject    handle to gA6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA6 as text
%        str2double(get(hObject,'String')) returns contents of gA6 as a
double


% --- Executes during object creation, after setting all properties.
function gA6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gt1_Callback(hObject, eventdata, handles)
% hObject    handle to gt1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt1 as text
%        str2double(get(hObject,'String')) returns contents of gt1 as a
double


% --- Executes during object creation, after setting all properties.
function gt1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function gt2_Callback(hObject, eventdata, handles)
% hObject    handle to gt2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt2 as text
%        str2double(get(hObject,'String')) returns contents of gt2 as a
double


% --- Executes during object creation, after setting all properties.
function gt2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gt3_Callback(hObject, eventdata, handles)
% hObject    handle to gt3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt3 as text
%        str2double(get(hObject,'String')) returns contents of gt3 as a
double


% --- Executes during object creation, after setting all properties.
function gt3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gt4_Callback(hObject, eventdata, handles)
% hObject    handle to gt4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt4 as text
```

```
%        str2double(get(hObject,'String')) returns contents of gt4 as a
double


% --- Executes during object creation, after setting all properties.
function gt4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gt5_Callback(hObject, eventdata, handles)
% hObject    handle to gt5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt5 as text
%        str2double(get(hObject,'String')) returns contents of gt5 as a
double


% --- Executes during object creation, after setting all properties.
function gt5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gt6_Callback(hObject, eventdata, handles)
% hObject    handle to gt6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt6 as text
%        str2double(get(hObject,'String')) returns contents of gt6 as a
double


% --- Executes during object creation, after setting all properties.
function gt6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in gdraw.
function gdraw_Callback(hObject, eventdata, handles)
% hObject    handle to gdraw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A1=str2num(get(handles.gA1,'String'));
A2=str2num(get(handles.gA2,'String'));
A3=str2num(get(handles.gA3,'String'));
A4=str2num(get(handles.gA4,'String'));
A5=str2num(get(handles.gA5,'String'));
A6=str2num(get(handles.gA6,'String'));
A7=str2num(get(handles.gA7,'String'));
t1=str2num(get(handles.gt1,'String'));
t2=str2num(get(handles.gt2,'String'));
t3=str2num(get(handles.gt3,'String'));
t4=str2num(get(handles.gt4,'String'));
t5=str2num(get(handles.gt5,'String'));
t6=str2num(get(handles.gt6,'String'));
t7=str2num(get(handles.gt7,'String'));
iterations=str2num(get(handles.giteration,'String'));
%the probabilities with which each transformation is applied
% notice that p1+p2+p3+p4+p5+p6+p7=1
p1=1/7;
p2=1/7;
p3=1/7;
p4=1/7;
p5=1/7;
p6=1/7;
p7=1/7;
% the initial point
x(1)=0;
y(1)=0;
% (x,y) points as a vector, v
v=[x(1);y(1)]; % here is the intial point
for n=2:iterations
% choose a random number, k, between 0 and 1
k=rand;
% depending random number ...
% do one of the four transformations to get a new point
if k<p1
v=A1*v+t1;
elseif k<p1+p2
v=A2*v+t2;
elseif k<p1+p2+p3
v=A3*v+t3;
elseif k<p1+p2+p3+p4
v=A4*v+t4;
elseif k<p1+p2+p3+p4+p5
v=A5*v+t5;
elseif k<p1+p2+p3+p4+p5+p6
v=A6*v+t6;
else
```

```
v=A7*v+t7;
end
x(n)=v(1);
y(n)=v(2);
end
%now, let's plot all those (x,y) points that we just computed!
opengl software %this is a fix to ensure that Matlab won't crash :)
hold off
%output of the image
plot(x,y,'.','Color',rand(1,3))
hold on
axis('equal','off');

% --- Executes on button press in gclear.
function gclear_Callback(hObject, eventdata, handles)
% hObject    handle to gclear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla



function gA7_Callback(hObject, eventdata, handles)
% hObject    handle to gA7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gA7 as text
%        str2double(get(hObject,'String')) returns contents of gA7 as a
double


% --- Executes during object creation, after setting all properties.
function gA7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gA7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gt7_Callback(hObject, eventdata, handles)
% hObject    handle to gt7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gt7 as text
%        str2double(get(hObject,'String')) returns contents of gt7 as a
double


% --- Executes during object creation, after setting all properties.
```

```
function gt7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gt7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function giteration_Callback(hObject, eventdata, handles)
% hObject    handle to giteration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of giteration as text
%        str2double(get(hObject,'String')) returns contents of giteration
as a double


% --- Executes during object creation, after setting all properties.
function giteration_CreateFcn(hObject, eventdata, handles)
% hObject    handle to giteration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```