Fractals

Krzysztof Gdawiec

kgdawiec@ux2.math.us.edu.pl

1 Introduction

In the 1970's Benoit Mandelbrot introduced to the world new field of mathematics. He named this field fractal geometry (*fractus* – from Latin divided, fractional). Fractal geometry breaks the way we see everything. It provides a new idea of modeling natural objects, such as clouds, plants, landscapes, galaxies. Fractals existed considerbaly earlier, but they were perceived as exceptional objects, mathematical monsters. Many of the early fractals came into being by effort of deep understanding of the basic mathematical notions. We can mention many names among mathematicians connected with fractals (e.g. George Cantor, Giuseppe Peano, David Hillbert, Helge von Koch, Waclaw Sierpinski). Their creatures such as Koch curve, Sierpinski gasket, Cantor set, Peano curve are called classical fractals. Mandelbrot turned the official interpretation of these objects upside down. He observed that what seemed to be an exception was a rule.

Currently the application of fractals is very wide. Their application can be found in image compression, generating shore lines, mountains, clouds or in medicine and economy.

Now we will try to present two basic methods which allow to generate fractals. Also we will present Julia sets and Mandelbrot set.

2 Iterated function system

In this section we will present one of methods to generate fractals – generating through iterated function system. But first we introduce several fundamental definitions.

Definition 2.1. Let X be a nonempty set. We say that function $\rho : X \times X \to [0, +\infty)$ is a *metric (distance)* on space X when

1. $\forall_{x,y \in X} \quad \rho(x, y) = 0 \iff x = y$
2. $\forall_{x,y \in X} \quad \rho(x, y) = \rho(y, x)$
3. $\forall_{x,y,z \in X} \quad \rho(x, y) + \rho(y, z) \geq \rho(x, z)$

This nonempty set X with metric ρ is called *metric space*.

For example $(\mathbb{R}^2, \rho)$, where ρ is Euclidean distance

$$\forall_{(x_1,y_1),(x_2,y_2) \in \mathbb{R}^2} \quad \rho((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

is metric space.

Definition 2.2. We say that a metric space (X, ρ) is *complete* if every sequence $(a_n)_{n\in\mathbb{N}}$ satisfying following condition

$$\forall_{\varepsilon>0}\exists_{N\in\mathbb{N}}\forall_{n,m>N} \quad \rho(a_n, a_m) < \varepsilon$$

is convergent.

In our further considerations we will use complex subset space in $\mathbb{R}^2$ which we denote as $H(\mathbb{R}^2)$ and as metric we take the Haussdorf distance $h : H(\mathbb{R}^2) \times H(\mathbb{R}^2) \to [0, +\infty)$

$$h(R, S) = \max\{D(R, S), D(S, R)\}$$

where $D : H(\mathbb{R}^2) \times H(\mathbb{R}^2) \to [0, +\infty)$ is defined as follows

$$D(R, S) = \max_{x\in R} \min_{y\in S} \rho(x, y)$$

for every $R, S \in H(\mathbb{R}^2)$ and any metric ρ on $\mathbb{R}^2$.

Theorem 2.3. *$(H(\mathbb{R}^2), h)$ is complete metric space.*

Proof. See [1]. □

Definition 2.4. Let (X, ρ) be a metric space. We say that transformation $f : X \to X$ is a *contraction mapping* if there exists $0 \leq c < 1$ such that

$$\forall_{x,y\in X} \quad \rho(f(x), f(y)) \leq c\rho(x, y).$$

Let us consider a contraction mapping $f : \mathbb{R}^2 \to \mathbb{R}^2$ and our space $(H(\mathbb{R}^2), h)$. Then the mapping f defines mapping on $H(\mathbb{R}^2)$ as follows

$$\forall_{A\in H(\mathbb{R}^2)} \quad f(A) = \{f(a) : a \in A\}.$$

Now we are ready to define iterated function system.

Definition 2.5. Let (X, ρ) be a complete metric space. We say that a set of contraction mappings $W = \{f_1, \ldots, f_n\}$, $n \in \mathbb{N}$ is an *iterated function system* (IFS).

If $W = \{f_1, \ldots, f_n\}$ is IFS, then W defines transformation on $(H(\mathbb{R}^2), h)$ as follows

$$\forall_{A\in H(\mathbb{R}^2)} \quad W(A) = \bigcup_{i=1}^{n} f_i(A).$$

Mapping W on the space $(H(\mathbb{R}^2), h)$ also is contraction mapping.

Let $A \in H(\mathbb{R}^2)$ and W be IFS. Consider following sequence

$$\begin{cases} W^0(A) = A \\ W^n(A) = W(W^{n-1}(A)), \quad n \geq 1 \end{cases} \tag{1}$$

According to the Banach fixed-point Theorem sequence given by (1) will be convergent to some $B \in H(\mathbb{R}^2)$. Moreover B will be independent on any choice of A.

Definition 2.6. We say that a set $B \in H(\mathbb{R}^2)$ being a limit of sequence given by (1) is an *attractor* of mapping W.

Let us consider an affine mapping $f : \mathbb{R}^2 \to \mathbb{R}^2$

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

where $a, b, c, d, e, f \in \mathbb{R}$. Using homogenous coordinates that mapping can be presented in the matrix form

$$f\left([x, y, 1]\right) = [x, y, 1] \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ e & f & 1 \end{bmatrix} = [x, y, 1]F$$

where

$$F = \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ e & f & 1 \end{bmatrix}$$

From now we will be using this affine mappings as a contraction mappings.

Let us consider following IFSs $W_1 = \{f_1, f_2, f_3\}$ where

$$\begin{aligned} f_1([x, y, 1]) &= [x, y, 1] \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0.25 & 0 & 1 \end{bmatrix} \\ f_2([x, y, 1]) &= [x, y, 1] \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0.5 & 1 \end{bmatrix} \\ f_3([x, y, 1]) &= [x, y, 1] \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix} \end{aligned}$$

and $W_2 = \{g_1, g_2, g_3\}$ where

$$\begin{aligned} g_1([x, y, 1]) &= [x, y, 1] \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0.75 & 0 & 1 \end{bmatrix} \\ g_2([x, y, 1]) &= [x, y, 1] \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0.5 & 1 \end{bmatrix} \\ g_3([x, y, 1]) &= [x, y, 1] \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 1 & 0.5 & 1 \end{bmatrix} \end{aligned}$$

Both of IFSs have the same attractor shown in fig.1. We can see that any $B \in H(\mathbb{R}^2)$ may be an attractor of different IFSs.

3 The chaos game

In this section we will present next method that allows us to generate fractals – the chaos game.

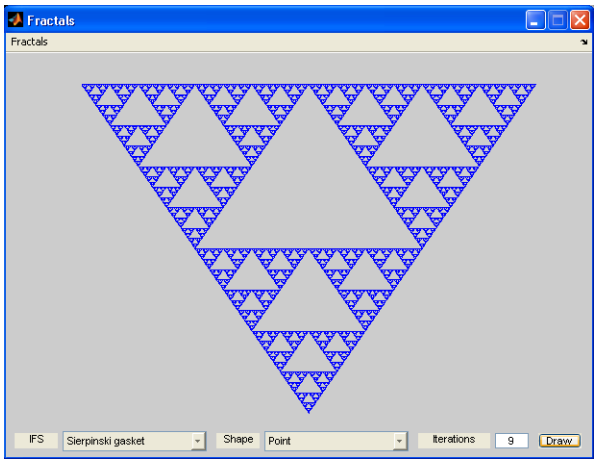

Fig. 1: Sierpinski gasket

Algorithm 1 Generating fractal through IFS

Require: $\{f_1, \ldots, f_k\}$ – affine mappings, S – starting set (point, triangle), n – number of iterations

Ensure: Approximation of an attractor of $\{f_1, \ldots, f_k\}$

$temp = A$
for $i = 1$ to n
  $temp1 = \emptyset$
  for $j = 1$ to k
    $temp1 = temp1 \cup f_j(temp)$
  end for
  $temp = temp1$
end for
Plot $temp$

First we will explain the rules of our game. We choose any three points of the plane and mark them as 1, 2, 3. For the game we need a dice with numbers 1, 2, 3 drawn on it. Finally we pick one more point of the plane (starting point). Now we throw the dice. After throwing up a number $n \in \{1, 2, 3\}$ we plot a point in a half of a distance between the starting point and the point marked as n. That new point becomes initial point for the next step, in which again we draw a number and plot new point in the half of the distance between the initial point and the drawed one. And again the new point becomes initial point for the next step. We repeat that procedure again and again. Fig.2 shows us the result of our game for 500, 2000 and 10000 steps. We see that slowly Sierpinski's gasket appears.

Now we consider an IFS $W = \{f_1, \ldots, f_n\}$, where f_i is an affine mapping given by the formula

$$f_i([x, y, 1]) = [x, y, 1]F_i$$

for $i = 1, \ldots, n$.

Now we try to create a new game. For every mapping f_i we define a probability $p_i > 0$ such that $\sum_{i=1}^{n} p_i = 1$. Like in our previous game we pick any point of the plane. Now we draw mapping f_i with probability p_i, where $i \in \{1, \ldots, n\}$ and we transform the point

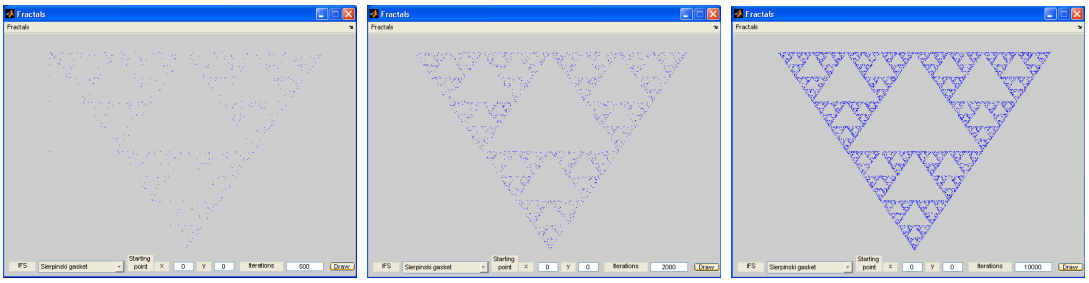

Fig. 2: Chaos game for 500, 2000, 10000 steps

with this mapping. This new point is initial point for the next step which is exactly the same as the first step. So we draw mapping and transform the point with this mapping. The new point is inital point for the next step. And we repeat that procedure again and again.

Now a question appears. How to choose the probabilities? We can take an uniform probability distribution, but we can obtain the best result if we choose them as follows

$$p_i = \frac{|\det F_i|}{\sum_{j=1}^{n} |\det F_j|}$$

for $i = 1, \ldots, n$.

Few words about the starting point. If the starting point does not belong to the attractor, then finite number of points created at the beginning of the game will lie outside the attractor. But if the starting point belongs to the attractor, then all new created points will lie in the attractor.

Algorithm 2 Generating fractal through chaos game

Require: n – number of steps, $\{f_1, \ldots, f_k\}$ – affine mappings, $\{p_1, \ldots, p_k\}$ – probabilities (probability distribution), $[x, y]$ – starting point

Ensure: Approximation of an attractor of $\{f_1, \ldots, f_k\}$

for $i = 1$ to n
  draw number $m \in \{1, \ldots, k\}$ according to given probability distribution $\{p_1, \ldots, p_k\}$

  $[x, y, 1] = f_m([x, y, 1])$
  Plot $[x, y]$
end for

4 Julia sets, Mandelbrot set

The name ,,Julia set" comes from the surname of a French mathematician Gaston Julia (1893-1978). His work about Julia sets appeared in 1918. The starting point of his work was article from a Britsh mathematican Sir Arthur Cayley entitled ,,The Newton-Fourier Imaginary Problem" (1879).

Let us consider following function $f : \mathbb{C} \to \mathbb{C}$, $f(z) = z^2 + c$, where $c \in \mathbb{C}$ is given parameter. Now we will iterate function f

$$\begin{cases} f^0(z) = z \\ f^n(z) = f(f^{n-1}(z)), \quad n \geq 1 \end{cases}$$

It turns out that for given $z \in \mathbb{C}$ we have two posibilities that happen in the iteration process with z when $n \to \infty$. It escapes to infinity, or it remains in a bounded area.

The points that escapes in the iteration process for given parameter $c \in \mathbb{C}$ are said to be in an escape set E_c

$$E_c = \{z \in \mathbb{C} : \lim_{n\to\infty} |f^n(z)| = +\infty\}$$

And the points that remains in a bounded area in the iteration process for given parameter $c \in \mathbb{C}$ are said to be in a prisoners set P_c

$$P_c = \{z \in \mathbb{C} : z \notin E_c\}$$

Definition 4.1. Let $c \in \mathbb{C}$. The boundary of the escape set E_c is called the *Julia set* J_c.

Now a question appears. How to mark the escape set E_c? The answer for that question is given by following lemma.

Lemma 4.2. *Let $c \in \mathbb{C}$ be a given parameter and $z \in \mathbb{C}$ be such that $|z| \geq |c|$ and $|z| > 2$. Then $z \in E_c$.*

Proof. See [5]. □

Let us define

$$r(c) = \max(|c|, 2)$$

According to Lemma (4.2) if $|f^k(z)| > r(c)$ for some $k = 0, 1, 2, \ldots$ then we are sure that $z \in E_c$. Now we define

$$Q_c^{(-k)} = \{z \in \mathbb{C} : |f^k(z)| \leq r(c)\}$$

for every $k = 0, 1, 2, \ldots$.

This set defines next approximations of the escape set, so

$$\lim_{k\to\infty} Q_c^{(-k)} = P_c$$

Using algorithm 3 we can visualize next approximations of the Julia sets.

Now we will concentrate on a Mandelbrot set. It was discovered in 1979 by French mathematician Benoit Mandelbrot. He studied whether the Julia set J_c is connected or disconnected set for given $c \in \mathbb{C}$. Let us remind the definition of connected set.

Definition 4.3. We say that a metric space (X, ρ) is *connected* when there do not exist nonempty, open sets $U, V \subset X$ such that $U \cap V = \emptyset$ and $X = U \cup V$.

Definition 4.4. We say that a subset W of the metric space (X, ρ) is *connected* if a metric space (W, ρ) is connected.

Algorithm 3

Require: $c \in \mathbb{C}$ – parameter in function f, $z \in \mathbb{C}$ – given point, k – the number of approximation

Ensure: TRUE $\iff$ $z \in Q_c^{(-k)}$, FALSE $\iff$ $z \in P_c$

$R = \max(|c|, 2)$
$i = 0$
while $i < k$
  if $|z| > R$
    Return FALSE
  end if
  $z = z^2 + c$
  $i = i + 1$
end while
Return TRUE

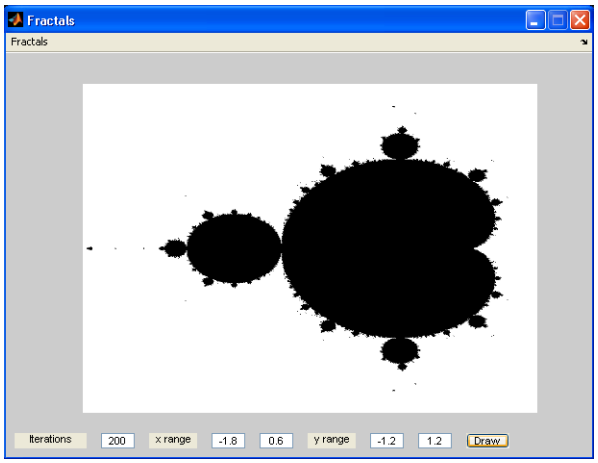

Fig. 3: The result of Mandelbrot's experiment

The definition of a Mandelbrot set looks as follows

$$M = \{c \in \mathbb{C} : J_c \text{ – is connected set}\}$$

Mandelbrot in his experiment marked on the complex plane with the black color every point $c \in \mathbb{C}$ if Julia set J_c was connected and if J_c was disconnected he marked that point with the white color. The result of this experiment is shown in the fig.3.

Here is an alternative definition of the Mandelbrot set

$$M = \{c \in \mathbb{C} : \lim_{n\to\infty} f_c^n(0) < \infty\}$$

where $f_c(z) = z^2 + c$.

Now a question appears. How to state whether Julia set J_c is connected or disconnected for given $c \in \mathbb{C}$? Like in the case of Julia sets we will define sets that will approximate the Mandelbrot set and will be convergent to it. The definition of this sets is following

$$R^{(-m)} = \{c \in \mathbb{C} : |f_c^m(0)| \leq 2\}$$

for $m = 0, 1, 2, \ldots$ and

$$\lim_{m\to\infty} R^{(-m)} = M$$

Algorithm 4

Require: $c \in \mathbb{C}$ – given parameter, m – the number of approximation
Ensure: TRUE $\iff$ $c \in R^{(-m)}$, FALSE $\iff$ J_c – disconnected set $\iff$ $c \notin M$

$k = 0$
$z = c$
while $k < m$
  if $|z| > 2$
    Return FALSE
  end if
  $z = z^2 + c$
  $k = k + 1$
end while
Return TRUE

Using algorithm 4 we can visualize next approximations of the Mandelbrot set.

5 Program and examples

In this section we will present implementations of the four algorithms that we have presented in previous sections and also some images that we have achieved using these functions.

5.1 Iterated function system

Listing 1: IFS.m

```
function p = IFS(trans,shape,n)
% IFS(trans,shape,n)
%
% function returns points generated by IFS
%
% trans - list of affine transformations
% shape - starting shape
% n - number of iterations
%
%
% example:
% n = 10;
% trans = {[0.4194 0.3629 -0.0000; 0.0376 0.3306 0.0000; 0 0 1.0000],...
%          [0.5645 -0.2903 0; 0.0699 0.1855 0.0000; 0.8500 0.8250 1.0000]};
% shape = [0 0 1; 1 0 1; 0.5 1 1];
% p = IFS(trans,shape,n);

temp = shape;
[iw,ik] = size(temp);
tr = size(trans,2);
len = length(trans);
```

```
temp1 = zeros(iw,ik*len^n);
w = 1;
for i = 1:n
    k = 1;
    for j = 1:tr
        for m = 1:w
            temp1(:,1+(k-1)*ik:ik+(k-1)*ik) = temp(:,1+(m-1)*ik:ik+(m-1)*ik) * trans{j};
            k = k + 1;
        end
    end
    w = w * len;
    temp = temp1(:,1:ik*w);
end

p = temp;
```

In our program we have included the possibility to read IFS from a file. The format of the ifs-files is following: the first line contains a number $n \in \mathbb{N}$ of an affine mappings, next n lines describe the affine mappings

$$a\ b\ c\ d\ e\ f\ p$$

where $a, b, c, d, e, f \in \mathbb{R}$ are the coefficients of the affine mapping and $p > 0$ is the probability of drawing this affine mapping. For example following file

```
3
0.387  0.43  0.43  −0.387  0.256  0.522  0.694
0.441  −0.091  −0.009  −0.322  0.4219  0.5059  0.2962
−0.468  0.02  −0.113  0.015  0.4  0.4  0.0099
```

describes following IFS

$$f_1([x,y,1]) = [x,y,1]\begin{bmatrix} 0.387 & 0.43 & 0 \\ 0.43 & -0.387 & 0 \\ 0.256 & 0.522 & 1 \end{bmatrix} \quad p_1 = 0.694$$

$$f_2([x,y,1]) = [x,y,1]\begin{bmatrix} 0.441 & -0.009 & 0 \\ -0.091 & -0.322 & 0 \\ 0.4219 & 0.5059 & 1 \end{bmatrix} \quad p_2 = 0.2962$$

$$f_3([x,y,1]) = [x,y,1]\begin{bmatrix} -0.468 & -0.113 & 0 \\ 0.02 & 0.015 & 0 \\ 0.4 & 0.4 & 1 \end{bmatrix} \quad p_3 = 0.0099$$

5.2 The chaos game

Listing 2: chaos.m

```
function Points = chaos(trans,point,n)
% chaos(trans,point,n)
%
% function returns points generated by the chaos game
%
% trans - list of affine transformations
```

```
% point - starting point
% n - number of iterations
%
%
% example:
% point = [0 0 1];
% trans = {[0.4194 0.3629 -0.0000; 0.0376 0.3306 0.0000; 0 0 1.0000],...
%          [0.5645 -0.2903 0; 0.0699 0.1855 0.0000; 0.8500 0.8250 1.0000]};
% n = 1000;
% p = chaos(trans,point,n);

%computing the probabilities
det_i = zeros(1,length(trans));
pz = [];
det_p = 0;
for i = 1:length(trans)
    d = abs(det(trans{i}));
    det_i(i) = d;
    det_p = det_p + d;
    if d == 0
        pz = [pz i];
    end
end

if det_p == 0
    det_p = 1;
    det_i(:) = 1/length(trans);
end

pr = det_i / det_p;

if det_p > 0 && ~isempty(pz)
    pp = 0.008;
    pk = pp*length(pz)/(length(trans)-length(pz));
    for i = 1:length(trans)
        if ~isempty(find(pz == i))
            pr(i) = pp;
        else
            pr(i) = pr(i) - pk;
        end
    end
end

p{1} = pr(1);
for i = 2:length(pr)
    p{i} = p{i-1} + pr(i);
end

%setting the starting point
pkt_t = point;
point_size = length(point);
points = zeros(point_size,n);

%chaos game
for i = 1:n
    r = rand(1);
    for j = 1:size(p,2)
        if r < p{j}
```

```
            pkt_t = pkt_t * trans{j};
            points(:,i) = pkt_t(:);
            break;
        end
    end
end

Points = points;
```

5.3 Julia sets, Mandelbrot set

Listing 3: Julia_plot.m

```
function Julia_plot(c,k,Xr,Yr)
% Julia_plot(c,k,Xr,Yr)
%
% function plots Julia set for given parameter c
%
% c - parameter in equation z = z^2 + c
% k - number of iterations
% Xr - range of values on the x axis, Xr(1,1) - min value, Xr(1,2) - max
% value
% Yr - range of values on the y axis, Yr(1,1) - min value, Yr(1,2) - max
% value
%
%
% example:
% Julia_plot(i,100,[-2 2],[-2 2])

n = 400;
x = linspace(Xr(1,1),Xr(1,2),n);
y = linspace(Yr(1,1),Yr(1,2),n);
[X,Y] = meshgrid(x,y);
W = zeros(length(X),length(Y));
for m = 1:size(X,2)
    for j = 1:size(Y,2)
        [w,iter] = Julia(X(m,j)+Y(m,j)*i,c,k);
        W(m,j) = W(m,j) + iter;
    end
end
hold on;
%axis off;

colormap(jet);
pcolor(W);
shading interp;

hold off;

%**********************

function [pri,it] = Julia(z,c,k)

R = max(abs(c),2);
i = 0;
while i < k
    if abs(z) > R
        pri = 1;
        it = i;
```

```
        return;
    end

    z = z^2 + c;
    i = i + 1;
end
pri = 0;
it = i;
```

Listing 4: Mandelbrot_plot.m

```
function Mandelbrot_plot(k,Xr,Yr)
% Mandelbrot_plot(k,Xr,Yr)
%
% function plots the Mandelbrot set in given range
%
% k - number of iterations
% Xr - range of values on the x axis, Xr(1,1) - min value, Xr(1,2) - max
% value
% Yr - range of values on the y axis, Yr(1,1) - min value, Yr(1,2) - max
% value
%
%
% example:
% Mandelbrot_plot(100,[-2 2],[-2 2])

n = 400;
x = linspace(Xr(1,1),Xr(1,2),n);
y = linspace(Yr(1,1),Yr(1,2),n);
[X,Y] = meshgrid(x,y);
W = zeros(length(X),length(Y));
for m = 1:size(X,2)
    for j = 1:size(Y,2)
        [w,iter] = Mandelbrot(X(m,j)+Y(m,j)*i,k);
        W(m,j) = W(m,j) + iter;
    end
end
hold on;
%axis off;

colormap(jet);
pcolor(W);
shading interp;

hold off;

%*********************

function [pri,it] = Mandelbrot(c,m)

k = 0;
z = c;
while k < m
    if abs(z) > 2
        pri = 1;
        it = k;
        return;
    end
```

```
    z = z^2 + c;
    k = k + 1;
end
pri = 0;
it = k;
```

References

[1] M. Barnsley, *Fractals Everywhere*, Academic Press, Boston, (1993)

[2] Y. Fisher, *Fractal image compression*, SIGGRAPH '92 Course Notes, vol. 12 pp. 7.1-7.19, (1992)

[3] John C. Hart, *Fractal Image Compression and Recurent Iterated Function Systems*, IEEE Computer Graphics and Applications, vol. 16, no. 4, pp. 25-33, (1996)

[4] G. Neil, K. M. Curtis, *Shape recognition using fractal geometry*, Pattern Recognition, vol. 30, No. 12, pp. 1957-1969, (1997)

[5] H.-O. Peitgen, H. Jurgens, D. Saupe, *Fractals for the Classroom*, Springer-Verlag 1992

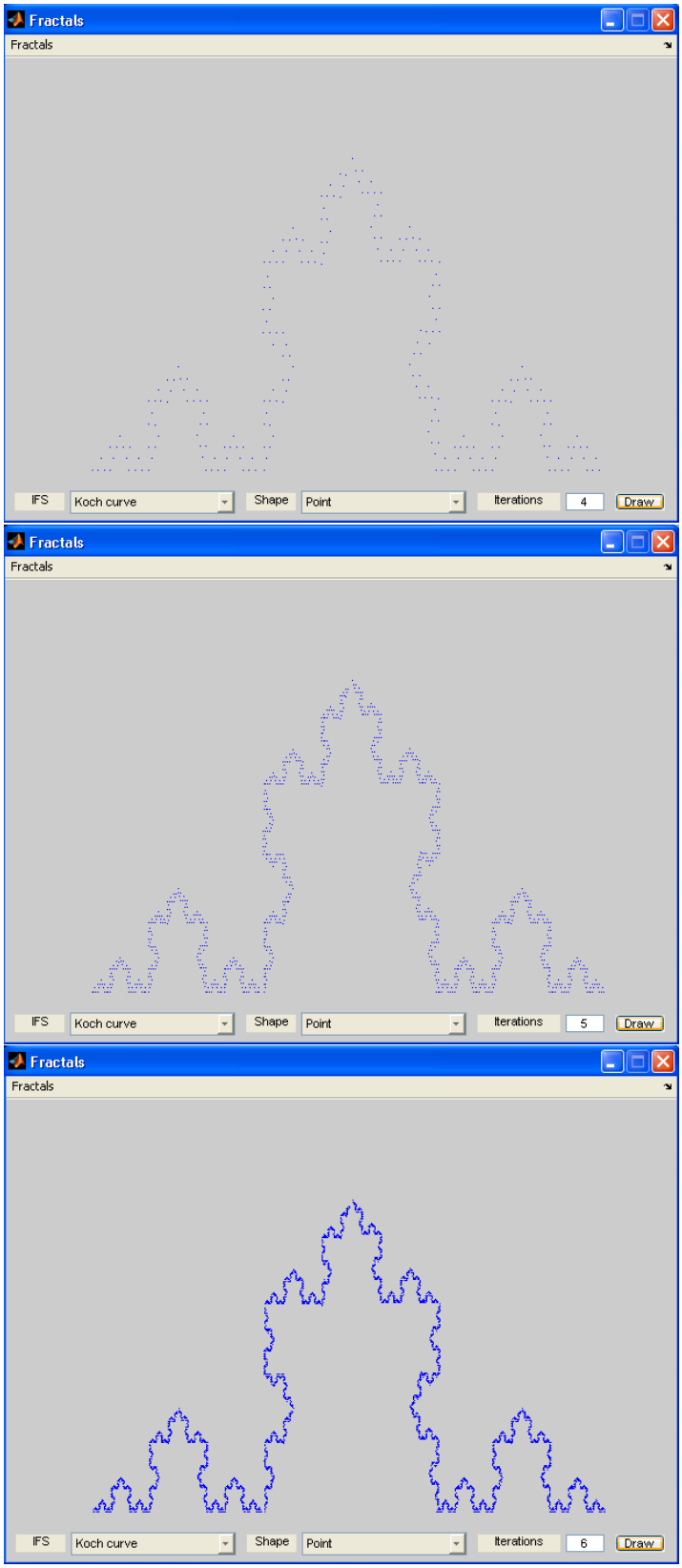

Fig. 4: Koch curve – starts from point, iteration 4, 5, 6

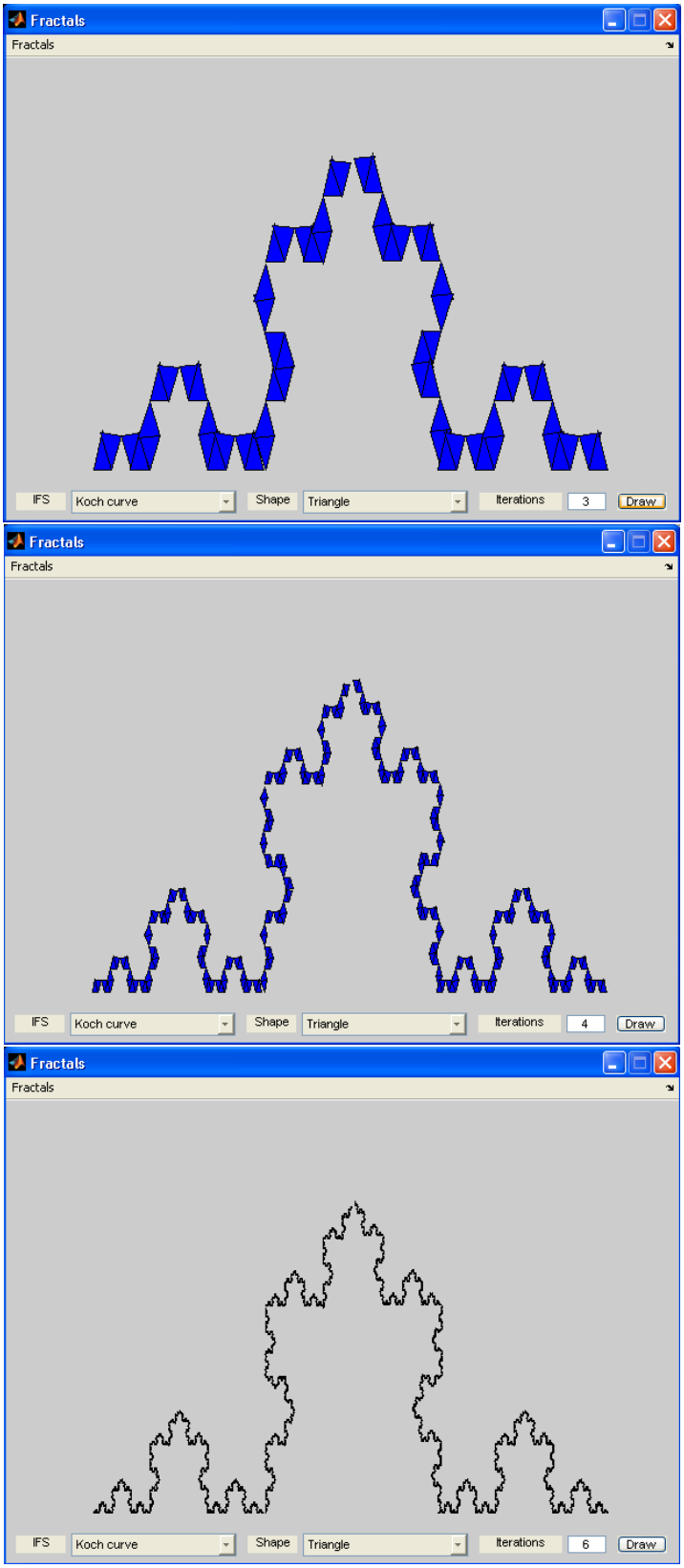

Fig. 5: Koch curve – starts from triangle, iteration 3, 4, 6

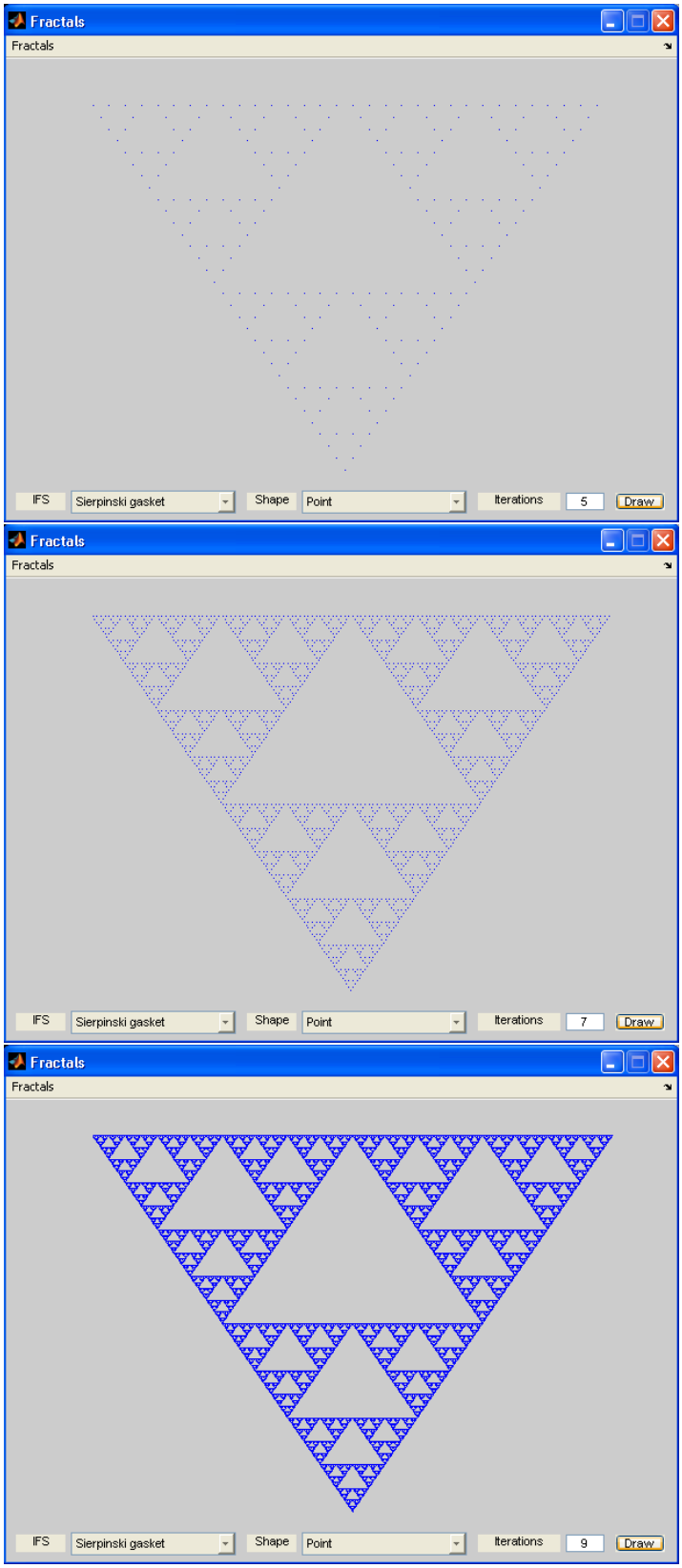

Fig. 6: Sierpinski gasket – starts from point, iteration 5, 7, 9

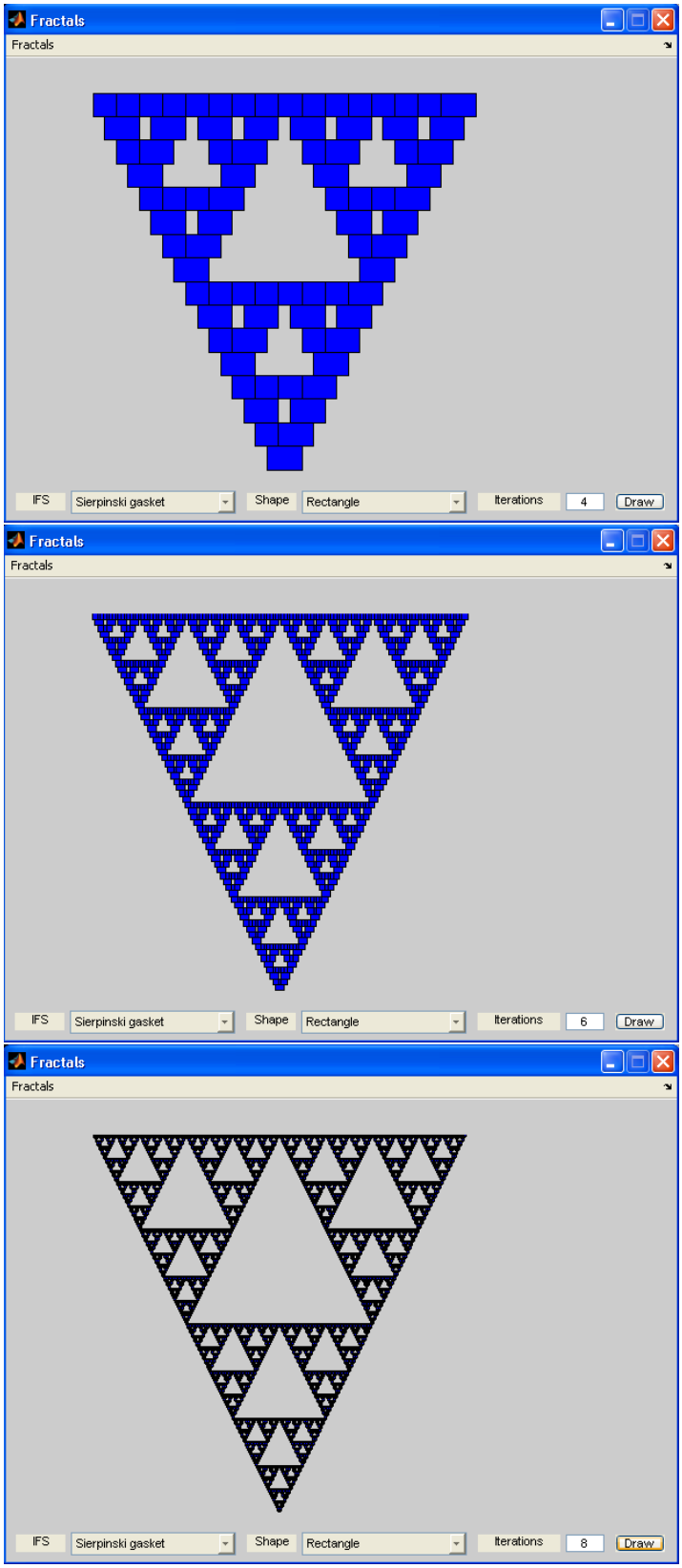

Fig. 7: Sierpinski gasket – starts from rectangle, iteration 4, 6, 8

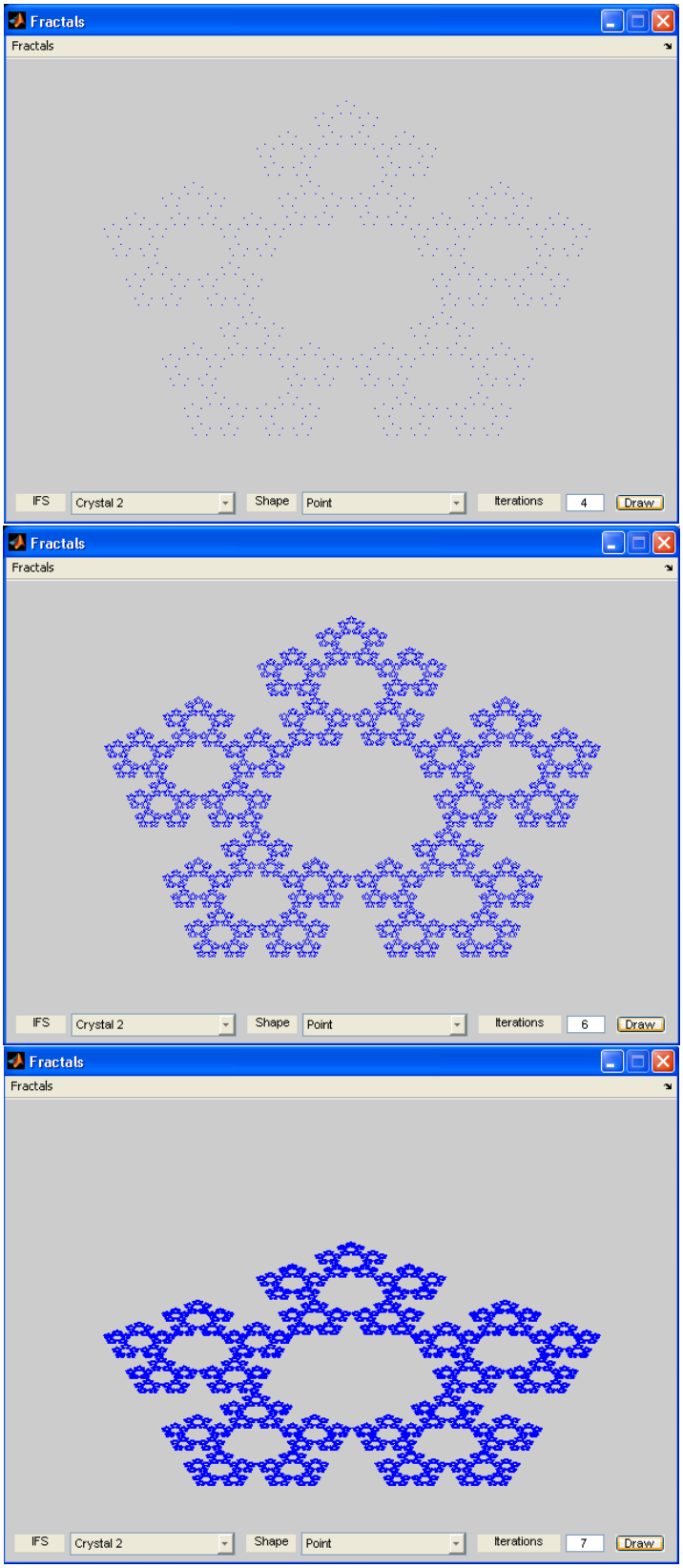

Fig. 8: Crystal – starts from point, iteration 5, 6, 7

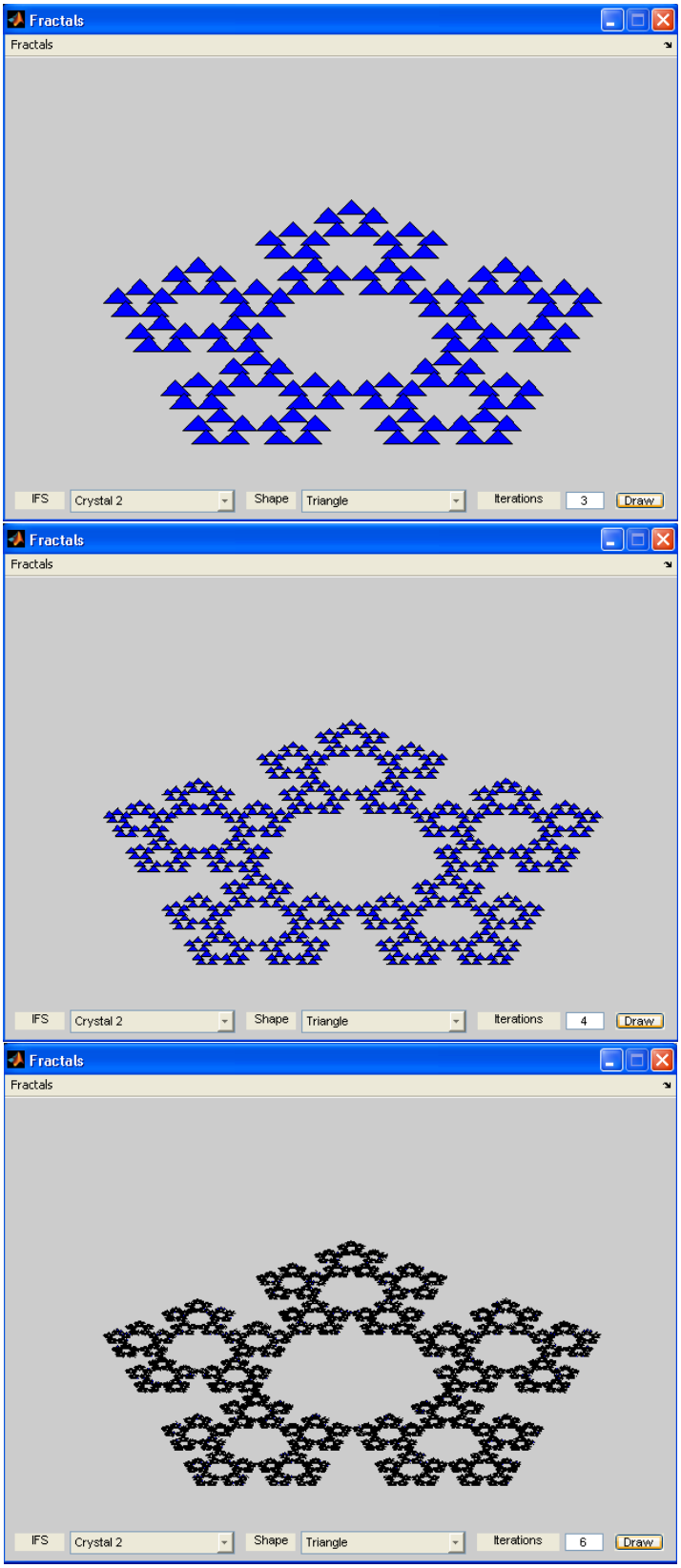

Fig. 9: Crystal – starts from triangle, iteration 3, 4, 6

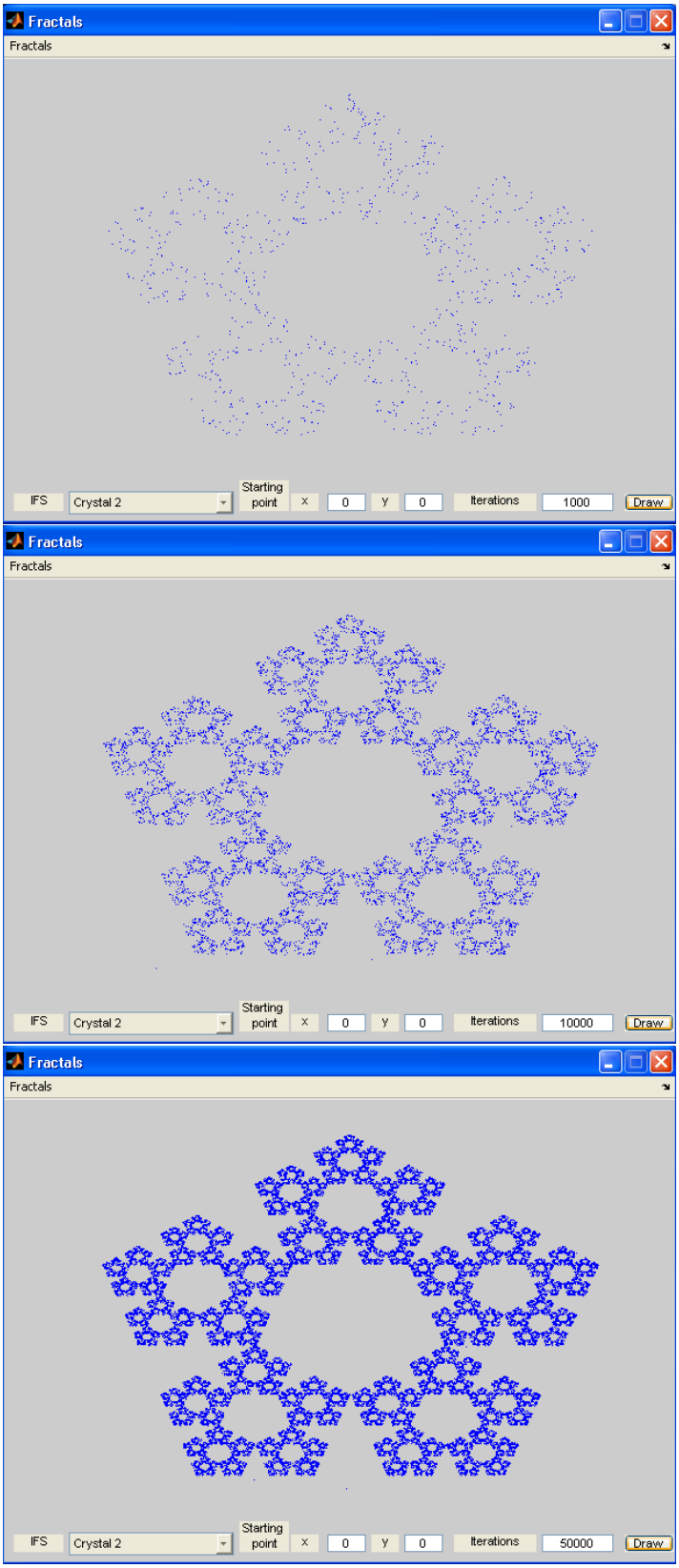

Fig. 10: Crystal – the chaos game, iteration 1000, 10000, 50000

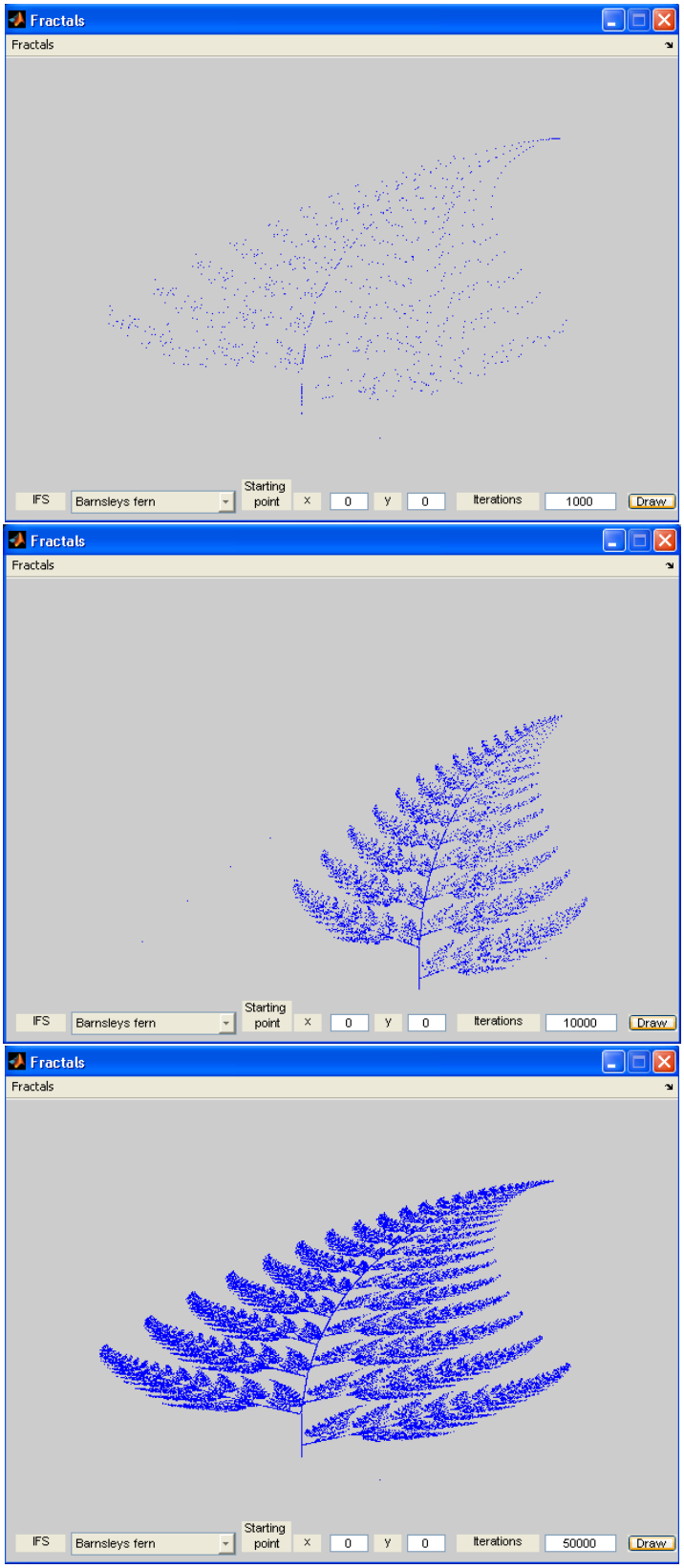

Fig. 11: Barnsley fern – the chaos game, iteration 1000, 10000, 50000

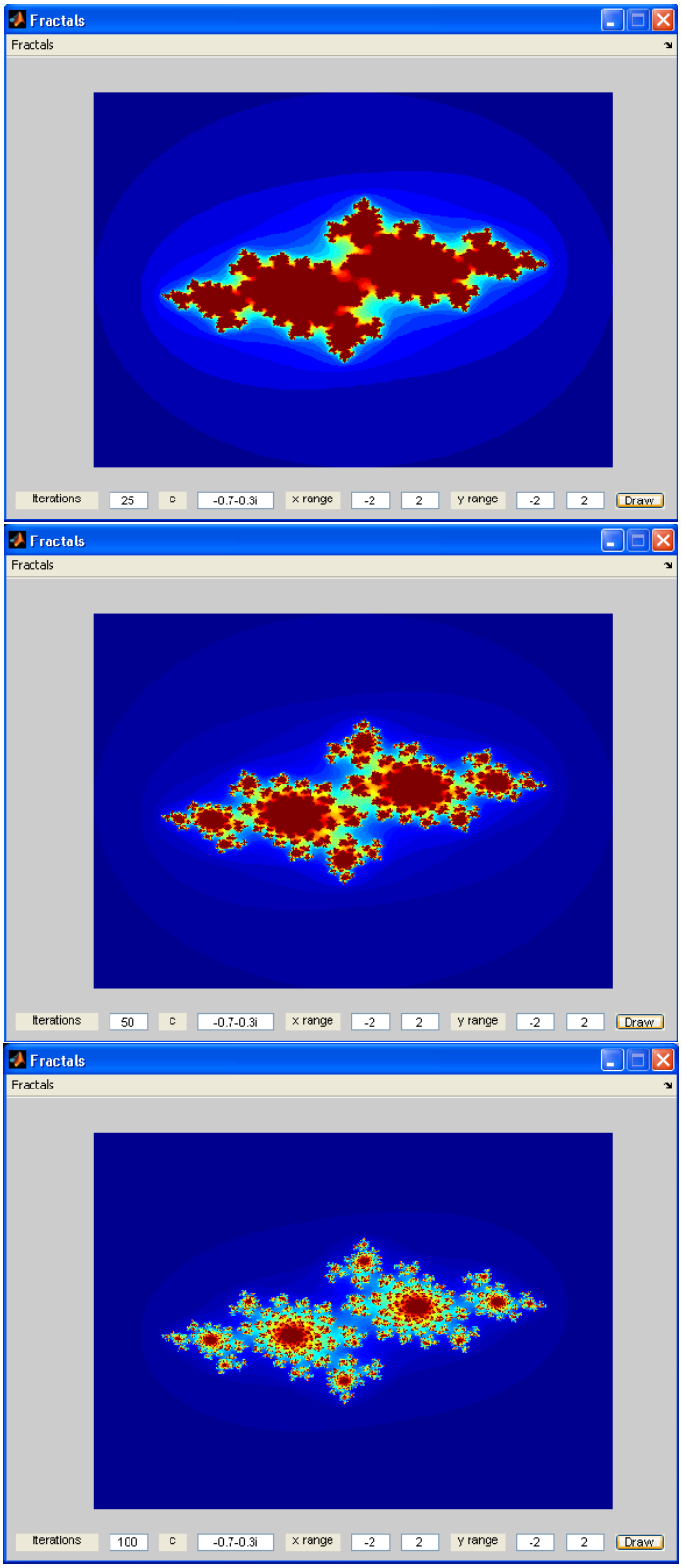

Fig. 12: Julia set – $c = -0.7 - 0.3i$, iterations 25, 50, 100

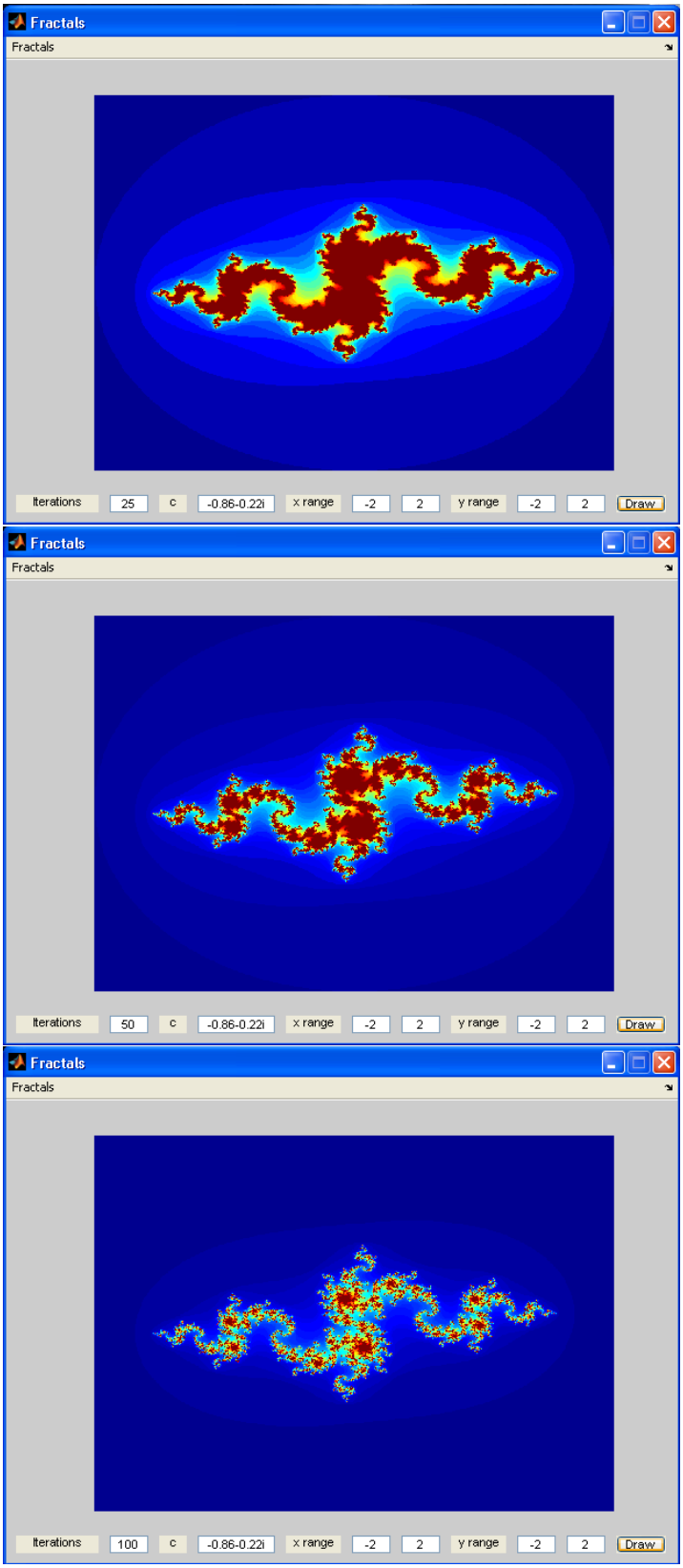

Fig. 13: Julia set – $c = -0.86 - 0.22i$, iterations 25, 50, 100

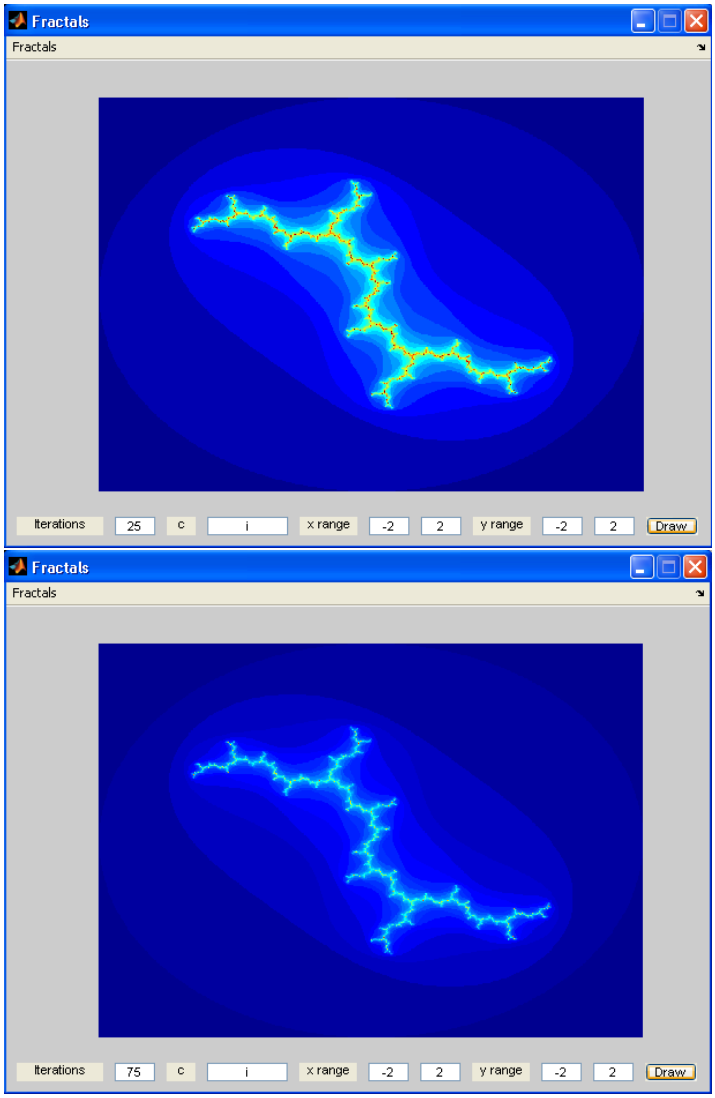

Fig. 14: Julia set – $c = i$, iterations 25, 75

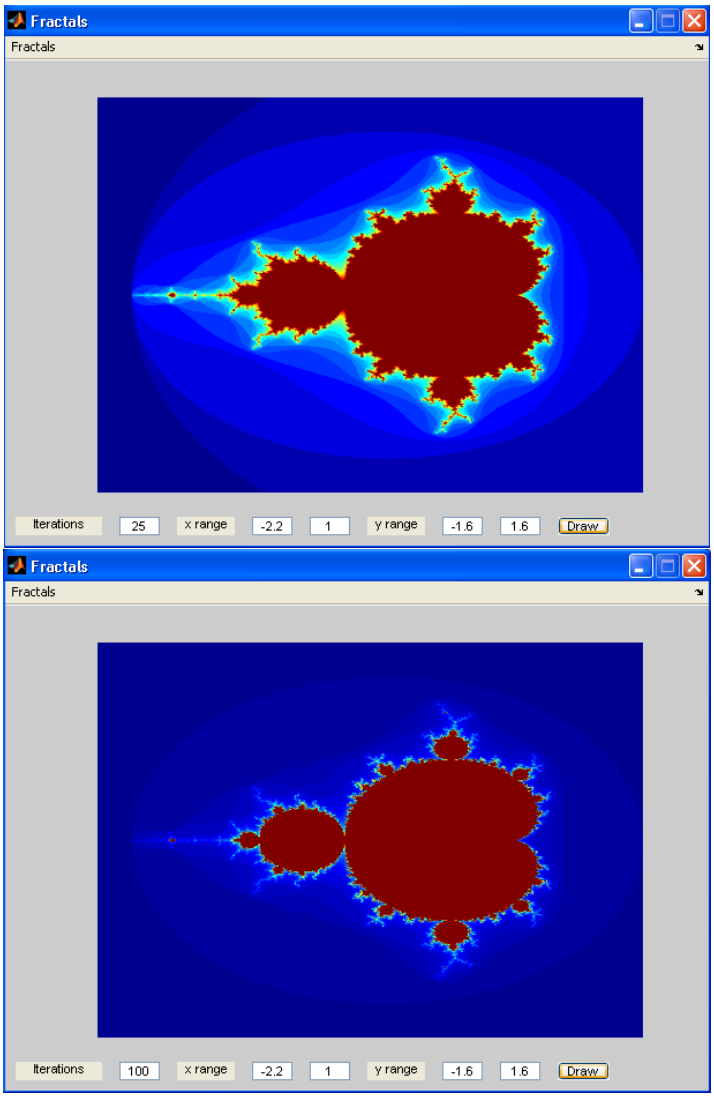

Fig. 15: Mandelbrot set – iterations 25, 100

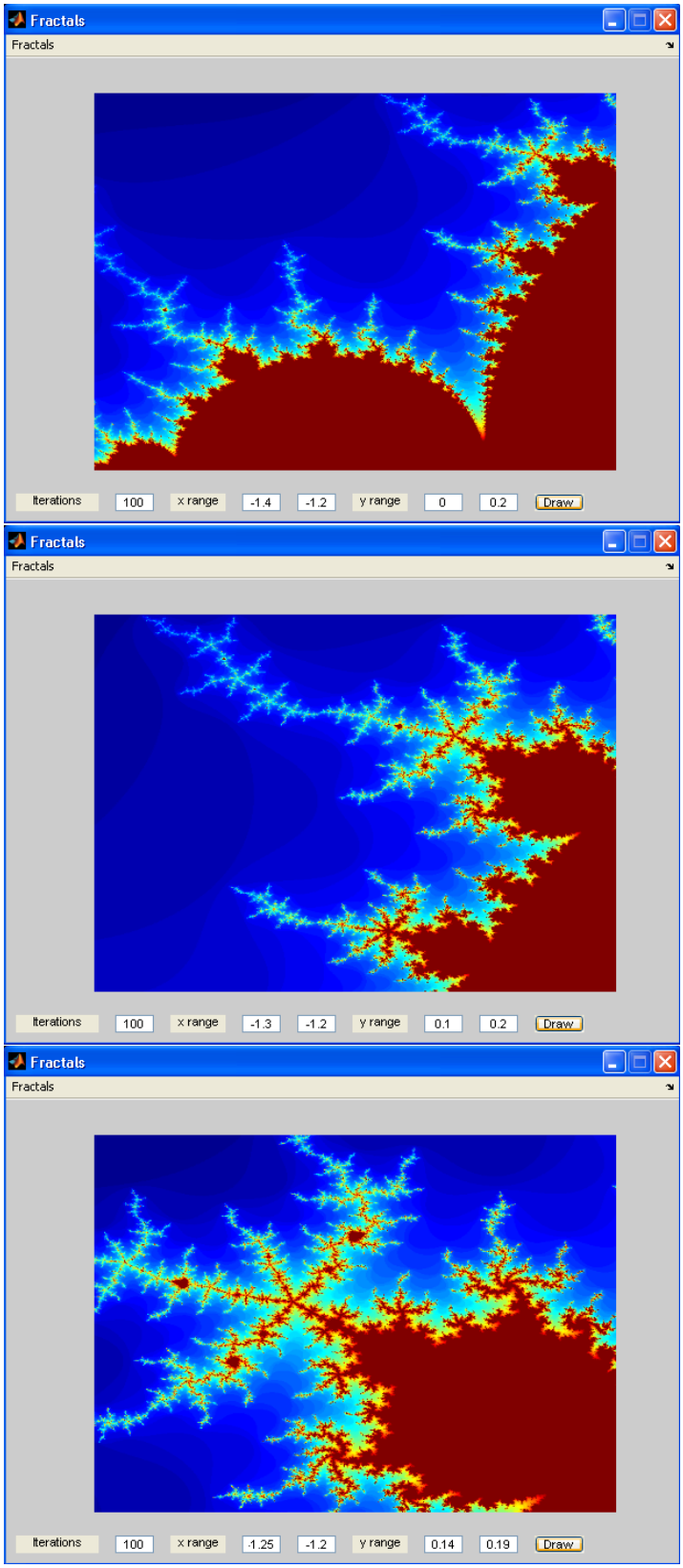

Fig. 16: Parts of the Mandelbrot set – iterations 100

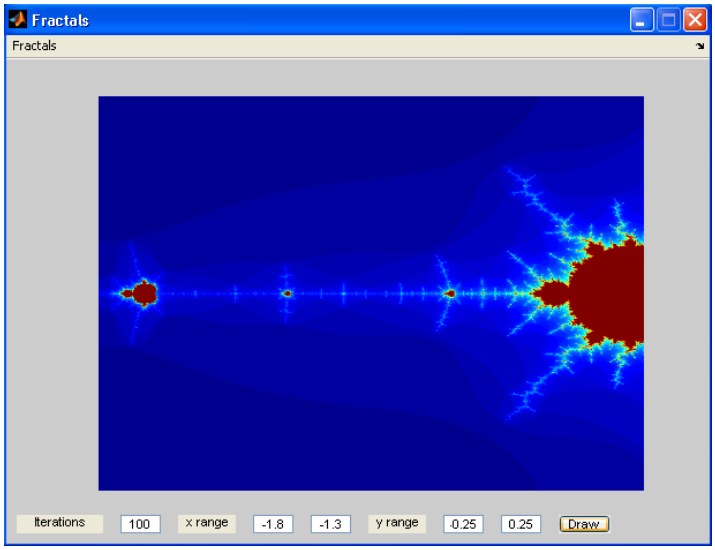

Fig. 17: Another part of the Mandelbrot set – iterations 100