

Софийски УНИВЕРСИТЕТ "Св. Климент Охридски"

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА



Фрактал на Нютон

Даниел Халачев, №62547, група II, курс II

Съдържание

1 Въведение	2
1.1 Метод на Нютон	2
1.2 От Метод на Нютон до фрактал	2
1.3 Наистина ли е фрактал?	5
2 Математическа формулировка	6
2.1 Извеждане	6
2.2 Генерализация	8
2.3 Нова фрактал	9
3 Програма за генериране на фрактали на езика C#	10
4 Инструкции за употреба на програмата за генериране на фрактали на Нютон	19

1 Въведение

Полиноми... Срещаме ги навсякъде. Използваме ги навсякъде - почти всяко математическо изчисление се базира на тях. Но за да имаме потребност от един полином, ние трябва да можем да намерим неговите корени. Когато степента на полинома е не по-голяма от 4, той може да бъде решен чрез формула. Но за полиномите от 5-та и по-голяма степен не съществуват формули за решението им.

Теорема (Абел-Руфини). *Не съществува решение в радикали на общи полиномиални уравнения с произволни коефициенти от степен n , за $n \geq 5$.*

Затова, за да се изчислят корените на полиноми от висока степен се използват алгоритми и методи, които да намерят достатъчно точни приближения до корените. Един от тях е...

1.1 Метод на Нютон

Нека $f(z) : \mathbb{R} \rightarrow \mathbb{R}$ е функция, а $f'(z)$ е нейната производна. Нека z_0 е избрана от нас начална стойност. Ако са изпълнени определени условия, редицата $\{z_i\}_{1}^{\infty}$ генерира все по-точни приближения z_i до стойността на някой корен \bar{z} на $f(z)$, м.e. $\lim_{i \rightarrow \infty} z_i = \bar{z}$, където

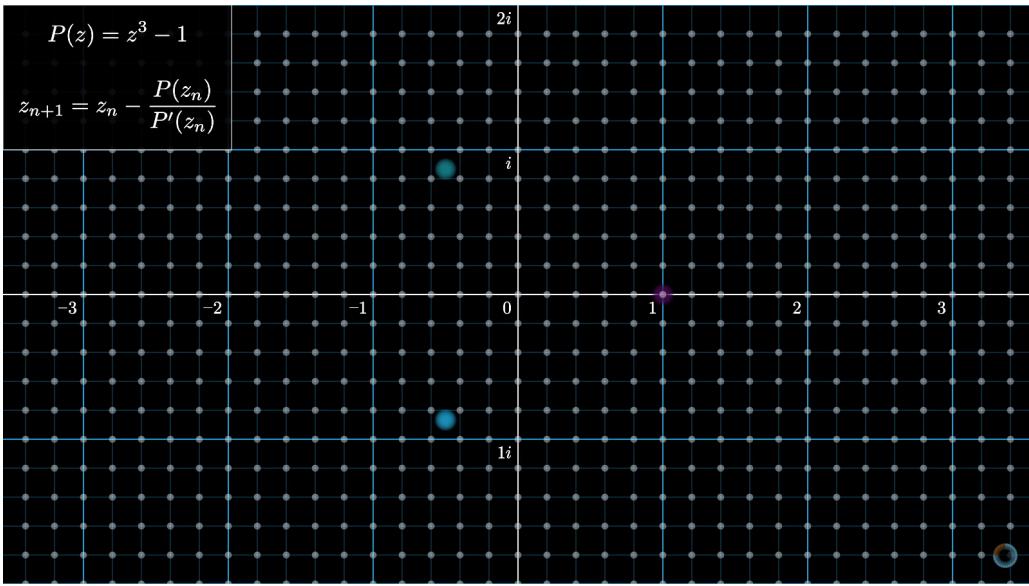
$$z_i = z_{i-1} - \frac{f(z_{i-1})}{f'(z_{i-1})}, i = 1, 2, \dots$$

$$f(\bar{z}) = 0$$

Когато необходимите условия не са изпълнени (изборът z_0 не е достатъчно близък до \bar{z} или $f'(z)$ не е добре дефинирана в околността на z_i за някое $i \in (0, \infty)$), то $\{z_i\}_{1}^{\infty} \not\rightarrow \bar{z}$. Тоест, всяка следваща итерация не ни приближава по-близо до корена \bar{z} . Това е особено видимо при графично представяне.

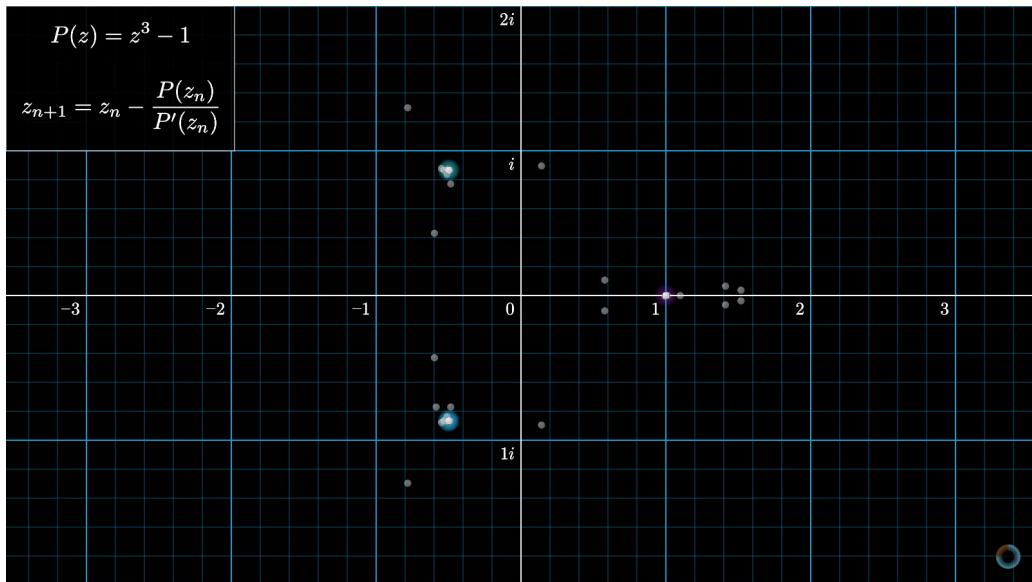
1.2 От Метод на Нютон до фрактал

Без ограничение на общността, нека вземем за пример полиномът $P(z) = f(z) = z^3 - 1$ с корени $z_1 = 1, z_{2,3} = \frac{-1 \pm \sqrt{3}i}{2}$. Нека приложим методът на Нютон за голям брой точки от комплексната равнина (наприимер 120 на брой).



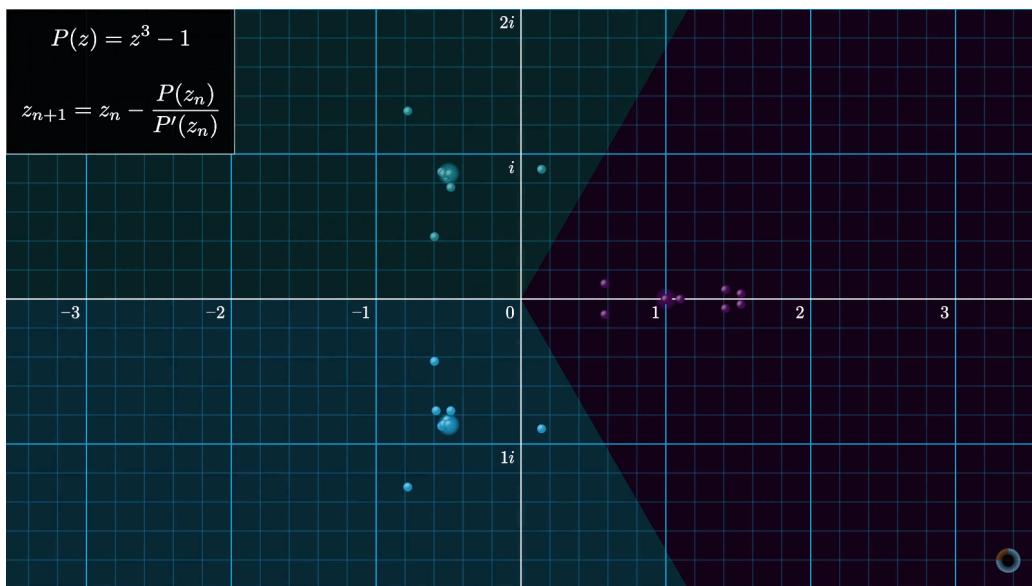
Фигура 1: Комплексната равнина. В сиво са означени точките, върху които ще приложим метода на Нютон, а с алтернативни цветове са обозначени корените z_1, z_2 и z_3 на $f(z) = z^3 - 1$

След всяка итерация на метода на Нютон точките ще се изместват, стремейки се да се доближат възможно най-много до корените на полинома. След достатъчно голям брой итерации забелязваме, че по-голямата част от тези точки са "оцелили" корените, докато останалите се движат "хаотично" и не могат "да намерят посоката". При тях условията на метода на Нютон не са изпълнени и затова приближението е неуспешно.



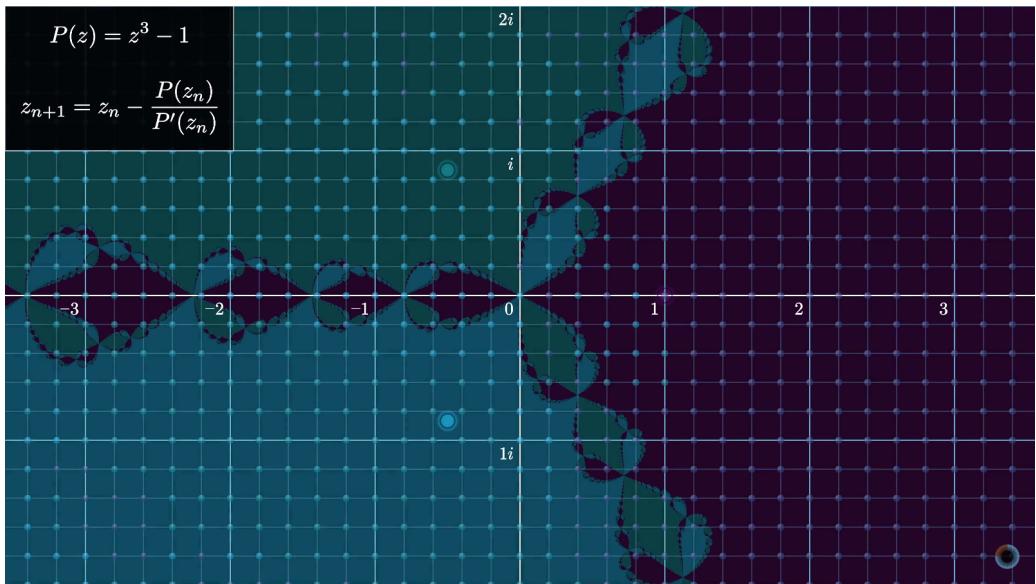
Фигура 2: Изместване на точките след p на брой итерации

Нека след p на брой итерации прекратим метода на Нютон и оцветим всяка точка в един от 3 цвята, в зависимост дали е най-близко разположена до z_1 , z_2 или z_3 .



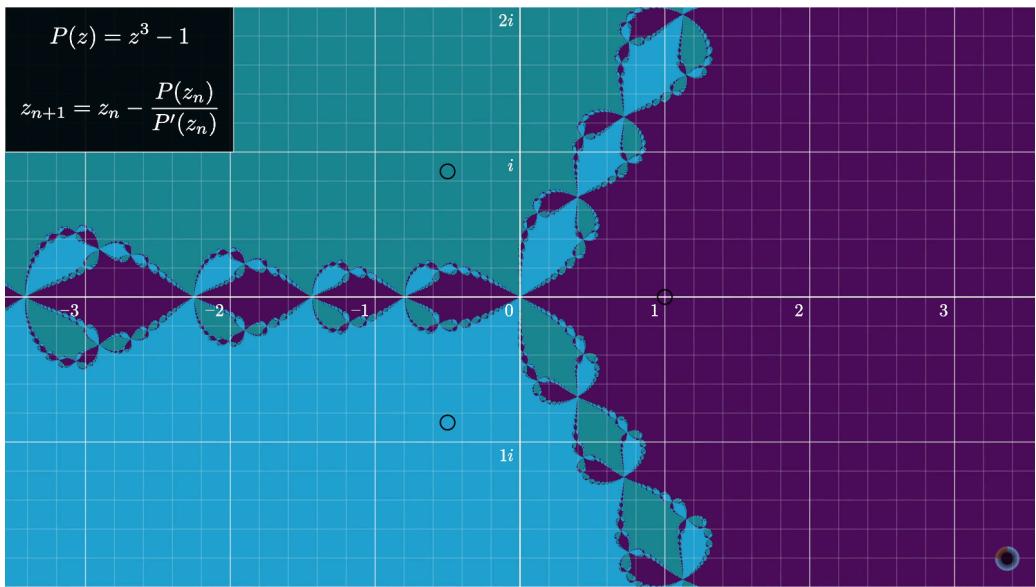
Фигура 3: Точките след p итерации, оцветени в зависимост до кой корен се намират най-близко

Когато върнем тези точки в началните им позиции, но запазим оцветяването им, ще получим следното изображение:



Фигура 4: Оцветените точки от комплексната равнина, върнати в изходното им положение

Ако повторим същите стъпки не за 120, а за няколко милиона точки (например 1920×1080 , колкото е резолюцията (броя пиксели) на един монитор, или дори за $n \rightarrow \infty$ на брой точки), изображението би изглеждало така:



Фигура 5: Изображението, генерирано чрез няколко милиона точки

Резултатът не е просто една красива картина, а изображение, притежаващо множество характерни особености. Най-важната от тях е, че колкото и да увеличаваме мащаба на изображението, характерната форма с тристранина симетрия се запазва и пресъздава отново и отново рекурсивно – получили сме фрактал!

Дефиниция (Фрактал). *Фракталът е структура, за която се наблюдава самоподобие със собствените ѝ части.*

1.3 Наистина ли е фрактал?

Нека разгледаме повторно резултатното изображение. Какво означава всеки един от тези цветове? Областта, оцветена в един от трите дадени цвята, представлява затвореното множество от точки, които след n на брой итерации се намират най-близко до z_i , където $f(z_i) = 0$. Наричаме всяка от тези области басейн на атракция, защото корените атрактират, т.e привличат точките в дадения цвят към себе си с всяка итерация. Границата на това множество е абсолютно същата като границата на множествата от точки за другите два цвята.

Дефиниция (Граница на множество). *Нека M е множество. Граница на M се нарича съвкупността от всички точки, такива, че във всяка тяхна околност попадат точки и извън, и вътре в M .*

Ако разгледаме границата на едно от трите множества от точки, ще установим, че не можем да открием точка, околността на която да съдържа точно 2 цвята. Всички околности на всички точки в комплексната равнина съдържат 1 или 3 цвята. Ако околността на точка съдържа един цвят, това означава че точката се намира вътре в множеството. Ако съдържа два цвята, това означава, че точката се намира на границата на две множества, която представлява плътна крива. Но околностите на точките от границата в това изображение винаги съдържат 3 цвята. Това означава, че границата между две множества от точки не е гладка крива, а рекурсивното повторение на изображението.

2 Математическа формулировка

2.1 Извеждане

Дефиниция (Симетричен полином). *Симетричен полином от степен v наричаме израза $z^v = 1$, $z \in \mathbb{C}, v \in \mathbb{N}, v > 2$*

Дефиниция. *Методът на Нютон за $f(z)$ е дефиниран чрез итерацията*

$$z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)} = g(z_k),$$

където z_k е k -тата итерация

Изчисляването на множество итерации е скъпа операция. Вместо да прилагаме формулата k пъти, тръгвайки от дадена начална точка, ще се запитаме кои точки $z_k = z$ се изобразяват в точка $z_{k+1} = z_0$ чрез формулата по-горе. Така дефинираме:

Дефиниция. *Комплексният Нютонов полином от ред v е дефиниран от*

$$(v-1)z^v - vz_0z^{v-1} + 1 = 0, \quad z, z_0 \in \mathbb{C}$$

Дефиниция. *Хаусдорфова мярка с размерност t на множеството \mathcal{A} наричаме величината*

$$H^t(\mathcal{A}) = \liminf_{\varepsilon \rightarrow 0} \left\{ \sum_{i=1}^M d^t(\mathcal{U}_i) : M \in \mathbb{N} \cup \infty; \mathcal{A} \subset \bigcup_{i=1}^M \mathcal{U}_i; 0 < d(\mathcal{U}_i) \leq \varepsilon, i = 1, 2, \dots \right\},$$

където $d(\mathcal{U}) = \sup \{d(x, y) : x, y \in \mathcal{U}\}$ - диаметър на множеството \mathcal{U} в метричното пространство (\mathbb{R}^n, d)

Дефиниция (Хаусдорфова размерност). *Хаусдорфова размерност на множеството \mathcal{A} се нарича величината*

$$\dim_H(\mathcal{A}) = \inf \{t : H^t(\mathcal{A}) > 0\} = \sup \{t : H^t(\mathcal{A}) < \infty\},$$

където H^t е t -измерната Хаусдорфова мярка.

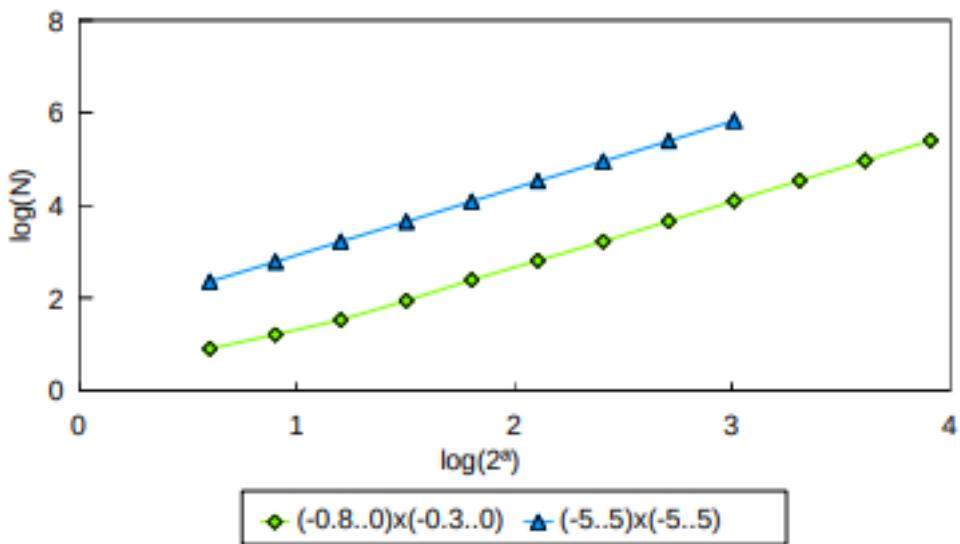
По-просто казано, ако изберем даден мащаб за "атомарен" (приемаме, че той съдържа всички елементарни обекти, необходими за построяване на фрактала), и впоследствие умножим мащаба и изброим колко са обектите в новото изображение, можем да получим размерността на фрактала, а именно:

$$d = \frac{\ln N}{\ln L}$$

Този метод, на подобието, е приложим само когато имаме фрактал, съставен от еднакви по размер свои копия. Фракталът на Нютон, който разглеждаме, в действителност е съставен от свои копия, но в различни размери. За да определим неговата размерност, трябва да използваме алтернативен подход, а именно *Броене на клетки*: Нанасяме фрактала в квадрат, т.e. "клетка" с определени размери - $\frac{1}{2^n}$. Нека след това изброим броя запълнени клетки $N_a(\mathcal{A})$. Нека сега намалим размера на квадрата, т.e. разбием го на по-малки квадратчета, и отново преброим броя запълнени клетки - сметката ще бъде по-точна. Ако повторим този експеримент два пъти с различни размери клетки и нанесем данните в таблица ще получим:

Резултати				
	[-0.8, 0] × [-0.3, 0]		[-5, 5] × [-5, 5]	
a	Брой клетки $N_a(\mathcal{A})$	Общо	Брой клетки $N_a(\mathcal{A})$	Общо
2	8	8	224	1600
3	16	21	600	6400
4	34	65	1636	25600
5	87	260	4469	102400

Нека съпоставим стойностите на $\ln N$ и $\ln 2^a$ за всяко a от експеримента и ги нанесем на графика:



Фигура 6: Съпоставяне стойностите на $\ln N$ и $\ln 2^a$ за всяко a от експеримента в двата му опита. Наклонът на двете прави е приблизително един и същ, колкото е и размерността на фрактала - 1.44

Получаваме и за двата опита прави с приблизително еднакъв наклон, а именно размерността на фрактала:

$$D \approx 1.44$$

Размерността на фрактала едробно число, за разлика от обектите в Евклидовата геометрия, които притежават целочислена размерност - 1, 2, 3,

Дефиниция (Фрактал). Фрактал е множества, чиято Хаусдорфова размерност е строго по-голяма от неговата топологична размерност.

Забележка. Когато дефинираме множество S като фрактал, приемаме, че то има следните свойства:

- S има детайли в определен мащаб
- S има твърде неправилна форма, за да бъде описан чрез традиционни геометрични подходи (напри мер от Евклидовата геометрия)
- S притежава някакво ниво на самоподобност
- Фракталната размерност на S е по-голяма от топологичната размерност на S
- S може да се дефинира математически рекурсивно или итеративно

Дефиниция. Фракталът на Нютон от ред v е определен от обединението на всички точки, които се изобразяват от Метода на Нютон

От тази дефиниция е очевидно, че ако z принадлежи на фрактала, то и z_0 също ще принадлежи на него. Трябва да отбележим, че фракталът е обединението на точките, които са на границата на басейните на привличане и не включват самите басейни.

Дефиниция. Множеството Жулия $\mathcal{J}(g)$ на функция g е затварянето на множеството от отблъскващи периодични точки на g

Дефиниция. Множеството Фату $\mathcal{F}(g)$ на функция g е допълнението на Множеството Жулия за g $\mathcal{J}(g)$

Така можем да дефинираме Фракталът на Нютон като обединението на всички точки от множеството на Жулия за метода на Нютон.

Дефиниция. Порядък на точка от множеството на Жулия се нарича броят итерации на Метода на Нютон, необходими за достигането на източника от тази точка.

Дефиниция. Басейн на атракция $\mathcal{A}(w)$ на атрактивна фиксирана точка w на функция g се дефинира така:

$$\mathbb{A}(w) = \{z \in \mathcal{C} : g^k(z) \rightarrow w, k \rightarrow \infty\}$$

Теорема. Нека w е атрактивна фиксирана точка на g . Тогава, ако $\partial\mathcal{A}(w)$ е границата на басейнът ѝ на атракция $\mathcal{A}(w)$, то

$$\mathcal{J}(g) = \partial\mathcal{A}(w)$$

Тоест, границата на атрактивните точки на функция g съвпада с Множеството на Жулия за g .

2.2 Генерализация

От казаното досега става ясно, че Фракталът на Нютон е чувствителен спрямо следните параметри:

- Изборът на полином, за който да приложим метода на Нютон
- Началните точки

Нека изменим малко формулата на Метода на Нютон:

Дефиниция. Генерализация на Метода на Нютон се нарича изразът

$$z_{k+1} = z_k - a \frac{f(z_k)}{f'(z_k)}, \quad a \in \mathbb{C}$$

Така фракталът на Нютон става чувствителен и по отношение на параметъра a , който се нарича коефициент на релаксация. Разгледан по този начин, фракталът на Нютон е частен случай на генерализирания фрактал за $a = 1$. Още по-общо фракталът на Нютон може да се разгледа като частен случай на множеството на Жулия.

Дори и за една и съща функция $f(z)$, при различно a се получават значителни разлики в структурата на фрактала. Примери за това са дадени по-долу:

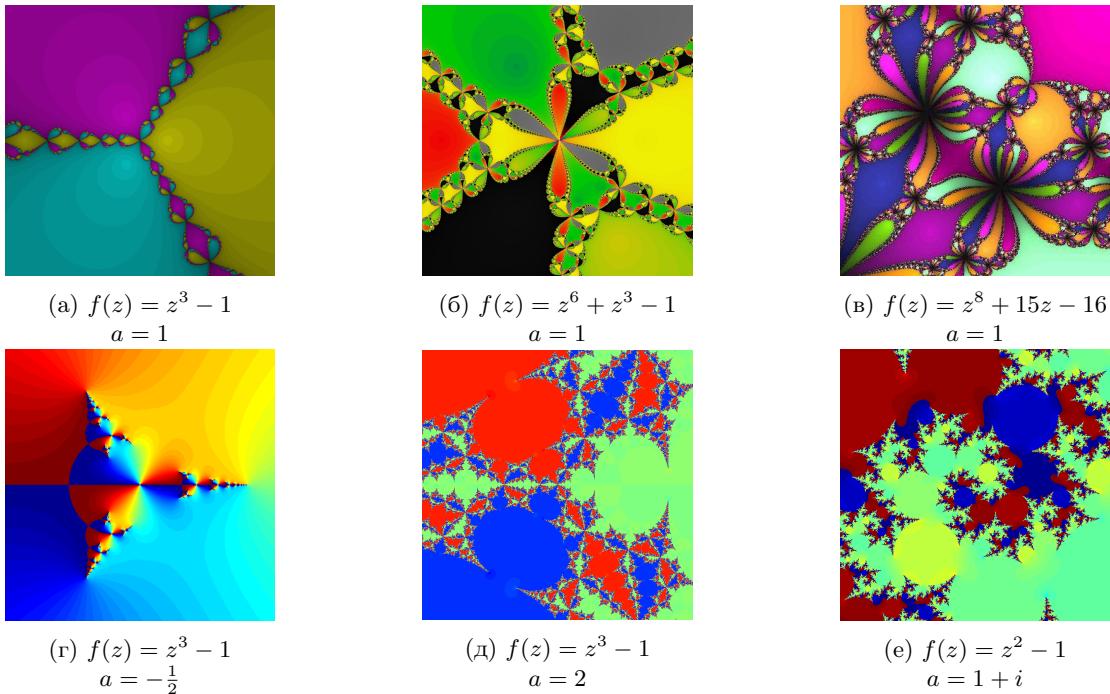


Таблица 1: Генерализирани фрактали на Нютон и техните басейни на атракция

2.3 Нова фрактал

Нека сега към формулата за генерализация прибавим константа c , променяща се на всяка итерация. Ще получим

$$z_{k+1} = z_k - a \frac{f(z_k)}{f'(z_k)} + c = G(a, c, z), \quad a, c \in \mathbb{C}$$

Резултатният фрактал се нарича Нова фрактал и е открит от Пол Дербишир в средата на 90-те в резултат от опитите му да подобри скоростта на сходимост на Метода на Нютон. Той се характеризира с още по-голяма "назъбеност" на формата:

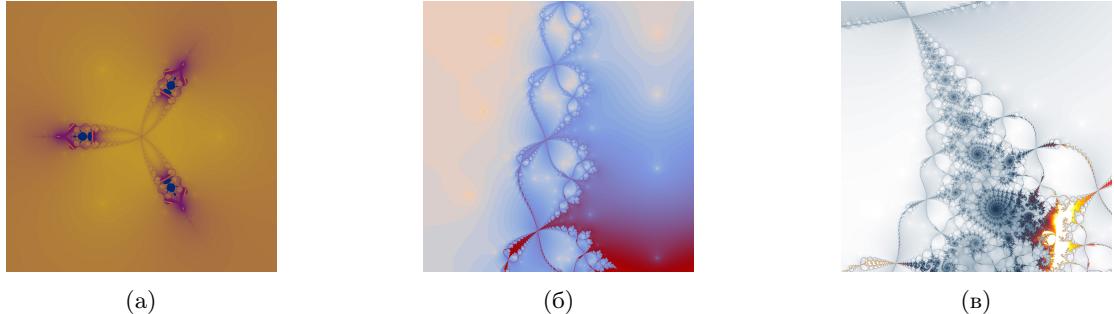


Таблица 2: Нова фрактали за $f(z) = z^3 - 1$, $a = 1$ и различни константи c

3 Програма за генериране на фрактали на езика C#

Listing 1: Имплементация на интерфейса Generic2DFractal

```
1  using ComplexMath;
2  using System;
3  using System.Threading;
4
5  namespace FractalRenderer
6  {
7      public class NewtonFractalByIterationsRequired : Generic2DFractal
8      {
9          public NewtonFractalByIterationsRequired()
10         {
11             fractalParameters.SetValue("NAME", "Newton Fractal by iterations required");
12             fractalParameters.SetValue("X", 0.0);
13             fractalParameters.SetValue("Y", 0.0);
14             fractalParameters.SetValue("W", 8.0);
15             fractalParameters.SetValue("H", 8.0);
16             fractalParameters.SetValue("A", 1.0);
17             fractalParameters.SetValue("WIDTH", 800);
18             fractalParameters.SetValue("HEIGHT", 800);
19             fractalParameters.SetValue("VERSION", "1.0.0");
20             fractalParameters.SetValue("ITERATIONS", 32);
21             fractalParameters.SetValue("NUM_THREADS", Environment.ProcessorCount);
22             int[] palette = { -16777088, -15662976, -14548864, -13434752, -12320640,
23                               -11206528, -10092416, -8978304, -7864192, -6750080, -5635968, -4521856,
24                               -3407744, -2293632, -1179520, -65408, -61576, -57488, -53400, -49312,
25                               -45224, -41136, -37048, -32960, -28872, -24784, -20696, -16608, -12520,
26                               -8432, -4344, -256, -12713984, -12318712, -11923440, -11528168, -11132896,
27                               -10671831, -10276559, -9881287, -9486015, -9024950, -8629678, -8234406,
28                               -7839134, -7443862, -6982797, -6587525, -6192253, -5796981, -5335916,
29                               -4940644, -4545372, -4150100, -3754828, -3293763, -2898491, -2503219,
30                               -2107947, -1646882, -1251610, -856338, -461066, -1, -16760832, -16235000,
31                               -15709168, -15183336, -14657504, -14066135, -13540303, -13014215,
32                               -12488383, -11897014, -11371182, -10845350, -10319518, -9793430, -9202061,
33                               -8676229, -8150397, -7624565, -7033196, -6507108, -5981276, -5455444,
34                               -4929612, -4338243, -3812411, -3286323, -2760491, -2169122, -1643290,
35                               -1117458, -591626, -1, -10526881, -10461088, -10395295, -10329502,
36                               -10263709, -10197916, -10132123, -10066330, -10000537, -9934744, -9868951,
37                               -9803158, -9737365, -9671572, -9605779, -9539986, -9474193, -9408400,
38                               -9342607, -9276814, -9211021, -9145228, -9079435, -9013642, -8947849,
39                               -8882056, -8816263, -8750470, -8684677, -8618884, -8553091, -8487298,
40                               -8421505, -8355712, -8289919, -8224126, -8158333, -8092540, -8026747,
41                               -7960954, -7895161, -7829368, -7763575, -7697782, -7631989, -7566196,
42                               -7500403, -7434610, -7368817, -7303024, -7237231, -7171438, -7105645,
43                               -7039852, -6974059, -6908266, -6842473, -6776680, -6710887, -6645094,
44                               -6579301, -6513508, -6447715, -6381922, -6316129, -6250336, -6184543,
45                               -6118750, -6052957, -5987164, -5921371, -5855578, -5789785, -5723992,
46                               -5658199, -5592406, -5526613, -5460820, -5395027, -5329234, -5263441,
47                               -5197648, -5131855, -5066062, -5000269, -4934476, -4868683, -4802890,
48                               -4737097, -4671304, -4605511, -4539718, -4473925, -4408132, -4342339,
49                               -4276546, -4210753, -4144960, -4079167, -4013374, -3947581, -3881788,
50                               -3815995, -3750202, -3684409, -3618616, -3552823, -3487030, -3421237,
51                               -3355444, -3289651, -3223858, -3158065, -3092272, -3026479, -2960686,
52                               -2894893, -2829100, -2763307, -2697514, -2631721, -2565928, -2500135,
53                               -2434342, -2368549, -2302756, -2236963, -2171170, -2105377, -2039584,
54                               -1973791, -1907998, -1842205, -1776412, -1710619, -1644826, -1579033,
55                               -1513240, -1447447, -1381654, -1315861, -1250068, -1184275, -1118482,
56                               -1052689, -986896, -921103, -855310, -789517, -723724, -657931, -592138,
57                               -526345, -460552, -394759, -328966, -263173, -197380, -131587, -1 };
58             fractalParameters.SetValue("PALETTE", palette);
59         }
60     }
61
62     public override void Configure()
63     {
64         Generic2DFractalSettings settingsDialog = new Generic2DFractalSettings();
65         settingsDialog.FractalType = "FractalRenderer.NewtonFractalByIterationsRequired";
66         settingsDialog.Parameters = (IFractalParameters)this.Parameters.Clone();
67         if (settingsDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
68         {
69             this.Parameters = settingsDialog.Parameters;
70         }
71     }
72 }
```

```

37         override protected void PartialRender(object P)
38     {
39         PartialRenderIterationsNeededToReachRoot(P);
40     }
41
42     protected void PartialRenderIterationsNeededToReachRoot(object P)
43     {
44
45         object[] o = (object[])P;
46         int offset = (int)o[0];
47         int lines = (int)o[1];
48         int[] dst = (int[])o[2];
49
50         RenderResult.RenderStatus renderStatus = (RenderResult.RenderStatus)o[3];
51         AutoResetEvent completed = (AutoResetEvent)o[4];
52
53         int width = (int)fractalParameters.GetValue("WIDTH");
54         int height = (int)fractalParameters.GetValue("HEIGHT");
55         int iterations = (int)fractalParameters.GetValue("ITERATIONS") - 1;
56         int[] Palette = (int[])fractalParameters.GetValue("PALETTE");
57
58         double a = (double)fractalParameters.GetValue("A");
59         double W = (double)fractalParameters.GetValue("W");
60         double H = (double)fractalParameters.GetValue("H");
61         double X = (double)fractalParameters.GetValue("X");
62         double Y = (double)fractalParameters.GetValue("Y");
63
64         int currentIteration = 0;
65         int idx = 0;
66         double xs = (X - W / 2.0);
67         double ys = (Y - H / 2.0);
68         double xd = W / (double)width;
69         double yd = H / (double)height;
70         double y1 = ys + yd * offset;
71
72
73         for (int y = offset; y < offset + lines; y++)
74     {
75             idx = y * width;
76             double x1 = xs;
77
78             for (int x = 0; x < width; x++)
79     {
80                 currentIteration = 0;
81
82                 // newton fractal formula:
83                 // zn+1 = zn - a* p(zn)/p'(zn)
84                 // where p(z):
85                 //           z^3-1   a: 1,0 ..... 0,5
86
87
88                 ComplexNumber zn = new ComplexNumber(x1, y1);
89                 ComplexNumber pz = new ComplexNumber(1, 0);
90                 ComplexNumber pzd = new ComplexNumber(1, 0);
91                 ComplexNumber Att1 = new ComplexNumber(1.25992, 0);
92                 ComplexNumber Att2 = new ComplexNumber(-0.629961, -1.09112);
93                 ComplexNumber Att3 = new ComplexNumber(-0.629961, 1.09112);
94
95                 if (x1 == 0 && y1 == 0)
96     {
97                     currentIteration = 0;
98                 }
99                 else
100     {
101                     while ((currentIteration < iterations) && pz.GetModulusSquared() >
102                         0.00000001)
103     {
104                     pz = ComplexNumber.Pow(zn, 3) - 2;
105                     pzd = 3 * ComplexNumber.Pow(zn, 2);
106                     zn -= a * pz / pzd;
107                     currentIteration++;
108                 }
109
110                 currentIteration = currentIteration % Palette.Length;
111                 dst[idx++] = Palette[currentIteration];

```

```

112             x1 += xd;
113         }
114         y1 += yd;
115         linesRendered++;
116         if (linesRendered % 40 == 0)
117         {
118             renderStatus(100.0f * ((float)linesRendered) / heigth);
119         }
120     }
121     completed.Set();
122 }
123 }
124 }
```

Listing 2: Код за действие на визуалната част на приложението

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Text;
7  using System.Windows.Forms;
8
9  namespace FractalRenderer
10 {
11     public partial class FractalForm : Form
12     {
13         #region MEMBERS
14         IAsyncResult renderResult = null;
15         IFractal frac = null;
16         string caption = String.Empty;
17         float CalculationProgress = 0;
18         Bitmap LastFractalImage = null;
19         int LastErrorCode = 0;
20         Point dragStart = Point.Empty;
21         Point dragEnd = Point.Empty;
22         Rectangle r = new Rectangle();
23         bool isDragging = false;
24         bool isZooming = false;
25         bool isRendering = false;
26         DateTime renderStart = DateTime.MinValue;
27         TimeSpan renderTime = new TimeSpan();
28     #endregion
29
30     /// <summary>
31     /// Get or set fractal associated with this form
32     /// </summary>
33     public IFractal Fractal
34     {
35         get { return frac; }
36         set { frac = value; }
37     }
38
39     public FractalForm()
40     {
41         InitializeComponent();
42         pictureBox2.MouseMove += new MouseEventHandler(pictureBox2_MouseMove);
43         pictureBox2.MouseDown += new MouseEventHandler(pictureBox2_MouseDown);
44         pictureBox2.MouseUp += new MouseEventHandler(pictureBox2_MouseUp);
45         splitContainer2.Panel1.SizeChanged += new System.EventHandler(this.
46             splitContainer2_Panel1_SizeChanged);
47     }
48
49     protected override void OnClosing(CancelEventArgs e)
50     {
51         if (isRendering && renderResult != null)
52         {
53             frac.EndRender(renderResult);
54         }
55         base.OnClosing(e);
56
57         private void splitContainer2_Panel1_SizeChanged(object sender, EventArgs e)
58         {
```

```

59             UpdatePictureBox();
60         }
61
62
63     void UpdatePictureBox()
64     {
65         if (InvokeRequired)
66         {
67             this.Invoke(new MethodInvoker(UpdatePictureBox));
68         }
69         else
70         {
71             if (LastFractalImage != null)
72             {
73                 pictureBox2.Width = LastFractalImage.Width + 20;
74                 pictureBox2.Height = LastFractalImage.Height + 20;
75
76                 Point location = pictureBox2.Location;
77
78                 if (pictureBox2.Width < splitContainer2.Panel1.Width)
79                 {
80                     location.X = (splitContainer2.Panel1.Width - pictureBox2.Width) /
81                         2;
82                 }
83                 else
84                 {
85                     location.X = -splitContainer2.Panel1.HorizontalScroll.Value;
86                 }
87
88                 if (pictureBox2.Height < splitContainer2.Panel1.Height)
89                 {
90                     location.Y = (splitContainer2.Panel1.Height - pictureBox2.Height) /
91                         2;
92                 }
93                 else
94                 {
95                     location.Y = -splitContainer2.Panel1.VerticalScroll.Value;
96                 }
97
98                 pictureBox2.Location = location;
99             }
100         }
101     void OnImageOriginDragged(int dx, int dy)
102     {
103         ((Generic2DFractal)frac).SetOrigin(dx, dy);
104
105         RenderFractal();
106     }
107
108     void OnControlPointDragged(int dx, int dy)
109     {
110         ((Generic2DFractal)frac).SetControlParameter(dx, dy);
111
112         RenderFractal();
113     }
114
115     bool ScreenToFractal(double X, double Y, out double XF, out double YF)
116     {
117         if (pictureBox2.Image != null)
118         {
119             int off_x = pictureBox2.Width - pictureBox2.Image.Width;
120             int off_y = pictureBox2.Height - pictureBox2.Image.Height;
121
122             off_x /= 2;
123             off_y /= 2;
124
125             if (X > off_x && X < (off_x + pictureBox2.Image.Width) &&
126                 Y > off_y && Y < (off_y + pictureBox2.Image.Height))
127             {
128                 X -= off_x;
129                 Y -= off_y;
130                 IFractalParameters pars = frac.Parameters;
131
132                 double FracX = (double)pars.GetValue("X");

```

```

133             double FracY = (double)pars.GetValue("Y");
134             double FracWidth = (double)pars.GetValue("W");
135             double FracHeight = (double)pars.GetValue("H");
136             double FracX1 = FracX - 0.5 * FracWidth;
137             double FracY1 = FracY - 0.5 * FracHeight;
138
139             // PicW : X = FracWidth : dx
140             // pictureBox2.Image.Width : X = FracWidth : dx
141             double dxf = FracWidth * X / pictureBox2.Image.Width;
142             double dyf = FracHeight * Y / pictureBox2.Image.Height;
143             XF = FracX1 + dxf;
144             YF = FracY1 + dyf;
145             return true;
146         }
147     }
148     XF = 0;
149     YF = 0;
150     return false;
151 }
152
153 void OnImageZoomed(Point p1, Point p2)
154 {
155     IFractalParameters pars = frac.Parameters;
156
157     double x1f,y1f;
158     double x2f, y2f;
159
160     ScreenToFractal(p1.X, p1.Y, out x1f, out y1f);
161     ScreenToFractal(p2.X, p2.Y, out x2f, out y2f);
162
163     double XN = (x2f + x1f) / 2.0;
164     double YN = (y2f + y1f) / 2.0;
165     double WFN = (x2f - x1f);
166     double HFN = (y2f - y1f);
167     pars.SetValue("X", XN);
168     pars.SetValue("Y", YN);
169     pars.SetValue("W", WFN);
170     pars.SetValue("H", HFN);
171     RenderFractal();
172 }
173
174 void pictureBox2_MouseUp(object sender, MouseEventArgs e)
175 {
176     if (IsMouseOverImage(e.X, e.Y) &&
177         dragStart != Point.Empty &&
178         e.Button == MouseButtons.Right)
179     {
180         // begin drag
181         dragEnd = new Point(e.X, e.Y);
182         OnImageOriginDragged(dragEnd.X - dragStart.X, dragEnd.Y - dragStart.Y);
183     }
184     else if (IsMouseOverImage(e.X, e.Y) &&
185              dragStart != Point.Empty &&
186              e.Button == MouseButtons.Middle)
187     {
188         // begin drag
189         dragEnd = new Point(e.X, e.Y);
190         OnControlPointDragged(dragEnd.X - dragStart.X, dragEnd.Y - dragStart.Y);
191     }
192     else if (IsMouseOverImage(e.X, e.Y) && dragStart != Point.Empty && e.Button ==
193              MouseButtons.Left)
194     {
195         double ar = 1.0;
196
197         if (pictureBox2.Image != null)
198         {
199             ar = (double)pictureBox2.Image.Width / (double)pictureBox2.Image.Height
200             ;
201         }
202
203         double drw = e.X - dragStart.X;
204         double drh = e.Y - dragStart.Y;
205
206         if (drw > drh) drh = drw / ar;
207         else drw = drh * ar;

```

```

207             dragEnd = new Point((int)(dragStart.X + drw), (int)(dragStart.Y + drh));
208             OnImageZoomed( dragStart , dragEnd );
209         }
210
211         isZooming = false;
212         isDragging = false;
213         dragStart = Point.Empty;
214         dragEnd = Point.Empty;
215         this.Cursor = Cursors.Default;
216     }
217
218     void pictureBox2_MouseDown(object sender, MouseEventArgs e)
219     {
220         if (IsMouseOverImage(e.X, e.Y) &&
221             dragStart == Point.Empty &&
222             e.Button == MouseButtons.Right &&
223             !isRendering)
224         {
225             // begin drag
226             dragStart = new Point(e.X, e.Y);
227             Cursor = Cursors.SizeAll;
228             isDragging = true;
229         }
230         else if (IsMouseOverImage(e.X, e.Y) &&
231             dragStart == Point.Empty &&
232             e.Button == MouseButtons.Middle &&
233             !isRendering)
234         {
235             // begin drag
236             dragStart = new Point(e.X, e.Y);
237             Cursor = Cursors.Cross;
238             isDragging = true;
239         }
240         else if (IsMouseOverImage(e.X, e.Y) &&
241             dragStart == Point.Empty &&
242             e.Button == MouseButtons.Left &&
243             !isRendering)
244         {
245             dragStart = new Point(e.X, e.Y);
246             Cursor = Cursors.IBeam;
247             isZooming = true;
248             // begin zoom
249             Point startPoint = pictureBox2.PointToScreen(new Point(e.X, e.Y));
250             r = new Rectangle(startPoint.X, startPoint.Y, 0, 0);
251         }
252         else
253         {
254             dragStart = Point.Empty;
255             dragEnd = Point.Empty;
256             Cursor = Cursors.Default;
257         }
258     }
259
260     void pictureBox2_MouseMove(object sender, MouseEventArgs e)
261     {
262
263         if (!isDragging && !isZooming && !isRendering)
264         {
265             if (IsMouseOverImage(e.X, e.Y))
266             {
267                 this.Cursor = Cursors.Hand;
268             }
269             else
270             {
271                 this.Cursor = Cursors.Default;
272             }
273         }
274         else if (isZooming && !isRendering)
275         {
276             ControlPaint.DrawReversibleFrame(r, this.BackColor, FrameStyle.Dashed);
277
278             Point endPoint = pictureBox2.PointToScreen(new Point(e.X, e.Y));
279             Point startPoint = pictureBox2.PointToScreen(dragStart);
280             double ar = 1.0;
281
282             if (pictureBox2.Image != null)

```

```

283             {
284                 ar = (double)pictureBox2.Image.Width / (double)pictureBox2.Image.Height
285                 ;
286             }
287
288             double drw = endPoint.X - startPoint.X;
289             double drh = endPoint.Y - startPoint.Y;
290
291             if (drw > drh)      drh = drw / ar;
292             else                  drw = drh * ar;
293
294             r = new Rectangle(startPoint.X, startPoint.Y, (int)drw, (int)drh);
295
296             ControlPaint.DrawReversibleFrame(r, this.BackColor, FrameStyle.Dashed);
297         }
298     }
299
300     bool IsMouseOverImage(int X, int Y)
301     {
302         if (pictureBox2.Image != null)
303         {
304             int off_x = pictureBox2.Width - pictureBox2.Image.Width;
305             int off_y = pictureBox2.Height - pictureBox2.Image.Height;
306
307             off_x /= 2;
308             off_y /= 2;
309
310             if (X > off_x && X < (off_x + pictureBox2.Image.Width) &&
311                 Y > off_y && Y < (off_y + pictureBox2.Image.Height))
312             {
313                 return true;
314             }
315         }
316         return false;
317     }
318
319     private void OnChangeFractalSettings(object sender, EventArgs e)
320     {
321         try
322         {
323             frac.Configure();
324         }
325         catch { }
326     }
327
328     private void UpdateFormContent()
329     {
330         if (InvokeRequired)
331         {
332             this.Invoke(new MethodInvoker(UpdateFormContent));
333         }
334         else
335         {
336             try
337             {
338                 this.Text = caption;
339                 if (LastFractalImage == null)
340                 {
341                     toolStripStatusLabel1.Visible = true;
342                     toolStripStatusLabel1.Text = "\CE\F2\EA\E0\E7\E0\ED\EE";
343                     toolStripProgressBar1.Visible = false;
344                     // LastErrorCode = 1;
345                 }
346                 else
347                 {
348                     toolStripStatusLabel1.Visible = true;
349                     toolStripStatusLabel1.Text = "\C3\E5\ED\E5\F0\E8\F0\E0\ED\EE: " +
350                         renderTime.TotalSeconds + "s";
351                     toolStripProgressBar1.Visible = false;
352                 }
353                 stopToolStripMenuItem.Enabled = false;
354                 renderToolStripMenuItem.Enabled = true;
355                 editToolStripMenuItem.Enabled = true;
356                 if (LastErrorCode == 0)
357                 {

```

```

357                     pictureBox2.Image = LastFractalImage;
358                     pictureBox2.SizeMode = PictureBoxSizeMode.CenterImage;
359                     UpdatePictureBox();
360                 }
361             }
362         }
363     }
364 }
365 }
366
367     private void UpdateFormCaption()
368     {
369         if (InvokeRequired)
370         {
371             this.Invoke(new MethodInvoker(UpdateFormCaption));
372         }
373         else
374         {
375             int Pcnt = (int)CalculationProgress;
376             this.Text = caption + "[" + Pcnt + "%]";
377             this.toolStripProgressBar1.Value = Pcnt;
378         }
379     }
380
381     private void RenderComplete(Bitmap bmp, int ErrorCode)
382     {
383         try
384         {
385             if (renderResult != null)
386             {
387                 renderTime = DateTime.Now - renderStart;
388                 LastErrorCode = ErrorCode;
389
390                 frac.EndRender(renderResult);
391                 LastFractalImage = bmp;
392
393                 UpdateFormContent();
394
395                 renderResult = null;
396                 isRendering = false;
397             }
398         }
399         catch { }
400     }
401
402     private void RenderStatusUpdated(float pcnt)
403     {
404         try
405         {
406             CalculationProgress = pcnt;
407             UpdateFormCaption();
408         }
409         catch { }
410     }
411
412     private void InterruptRender()
413     {
414         if (renderResult != null)
415         {
416             frac.EndRender(renderResult);
417         }
418
419         renderToolStripMenuItem.Enabled = true;
420         stopToolStripMenuItem.Enabled = false;
421     }
422
423     private void RenderFractal()
424     {
425         if (renderResult == null)
426         {
427             caption = this.Text;
428             isRendering = true;
429
430             renderStart = DateTime.Now;
431             renderResult = frac.BeginRender(new RenderResult.RenderComplete(
432                 RenderComplete),

```

```

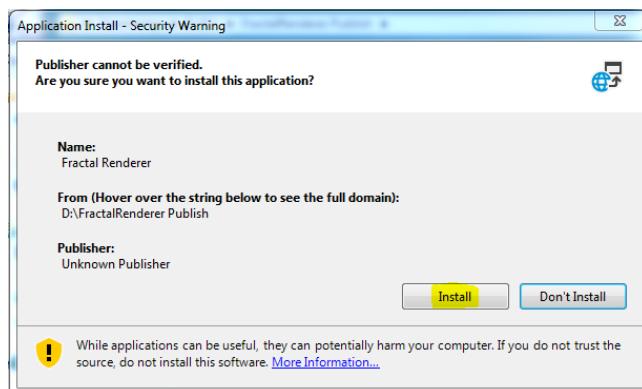
432                                     new RenderResult.RenderStatus(
433                                         RenderStatusUpdated));
434
435             renderToolStripMenuItem.Enabled = false;
436             editToolStripMenuItem.Enabled = false;
437             stopToolStripMenuItem.Enabled = true;
438             toolStripStatusLabel1.Visible = false;
439             toolStripProgressBar1.Visible = true;
440         }
441     }
442
443     private void OnrenderToolStripMenuItemClicked(object sender, EventArgs e)
444     {
445         RenderFractal();
446         stopToolStripMenuItem.Enabled = false;
447     }
448
449     private void OnstopToolStripMenuItemClicked(object sender, EventArgs e)
450     {
451         InterruptRender();
452     }
453
454     private void RenderToolStripMenuItem1_Click(object sender, EventArgs e)
455     {
456         RenderFractal();
457     }
458
459     private void CancelToolStripMenuItem1_Click(object sender, EventArgs e)
460     {
461         InterruptRender();
462     }
463
464     private void ExitToolStripMenuItem_Click(object sender, EventArgs e)
465     {
466         this.Close();
467     }
468 }
469 }
```

4 Инструкции за употреба на програмата за генериране на фрактали на Нютон

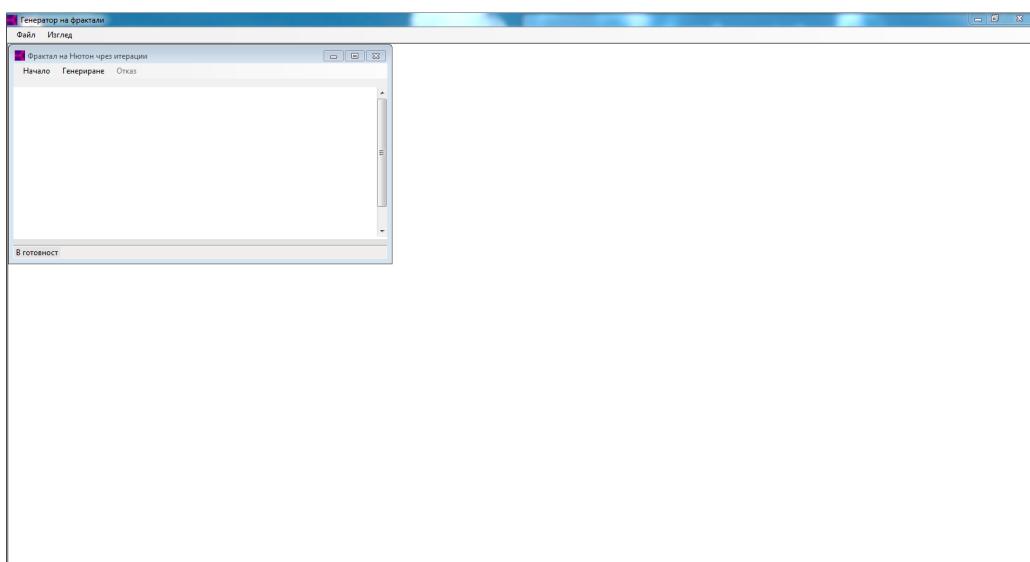
1. Поставете диска
2. Отворете програмния файл FractalRendererer

Име	Дата на промяна	Тип
Application Files	23.4.2022 г. 10:57 ч.	Папка с файлове
FractalRendererer	23.4.2022 г. 10:57 ч.	ClickOnce Applica...
FractalRendererer	23.4.2022 г. 10:57 ч.	Приложение
setup	23.4.2022 г. 10:57 ч.	Приложение

3. Възможно е готовият програмен файл да не работи на Вашия компютър. В този случай ще трябва да инсталirate приложението, като отворите инсталационния файл setup и следвате инструкциите в него. Ако програмата се е отворила успешно, пропуснете тази стъпка



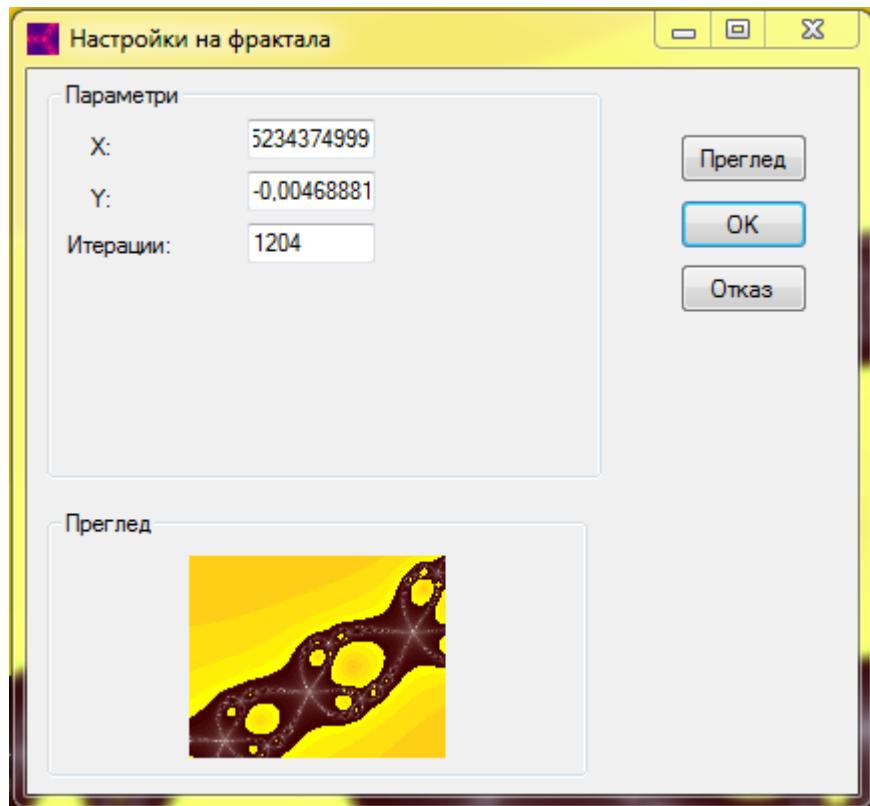
4. Препоръчително е да разширите програмата на цял еcran за най-добра видимост
5. Изберете Файл -> Нов -> Фрактал на Нютон
6. В новия вътрешен прозорец натиснете Генериране



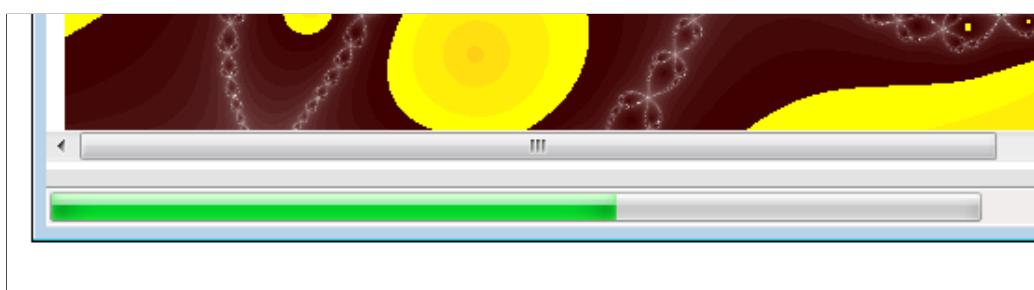
7. Ако искате да увеличите машаба на фрактала и да "задълбаете" в него, изберете с левия бутон на мишката правоъгълник по Ваш избор (натиснете ляв бутон на мишката и без да отпускате я

влачете докато не постигнете избора си, след това отпуснете левия бутон). Програмата автоматично ще генерира новото изображение

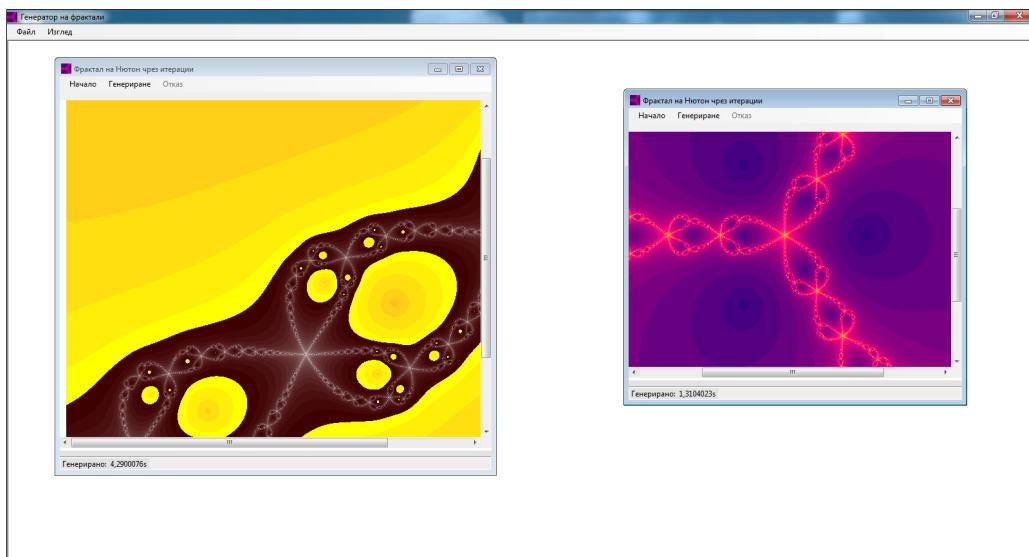
8. Можете да промените настройките за генерирането от менюто Начало -> Настройки. Ако забелязвате неясни детайли във фрактала, увеличите броя итерации - например на 512, а после потвърдете с OK и натиснете Генериране отново. Изображението ще се създаде повторно.



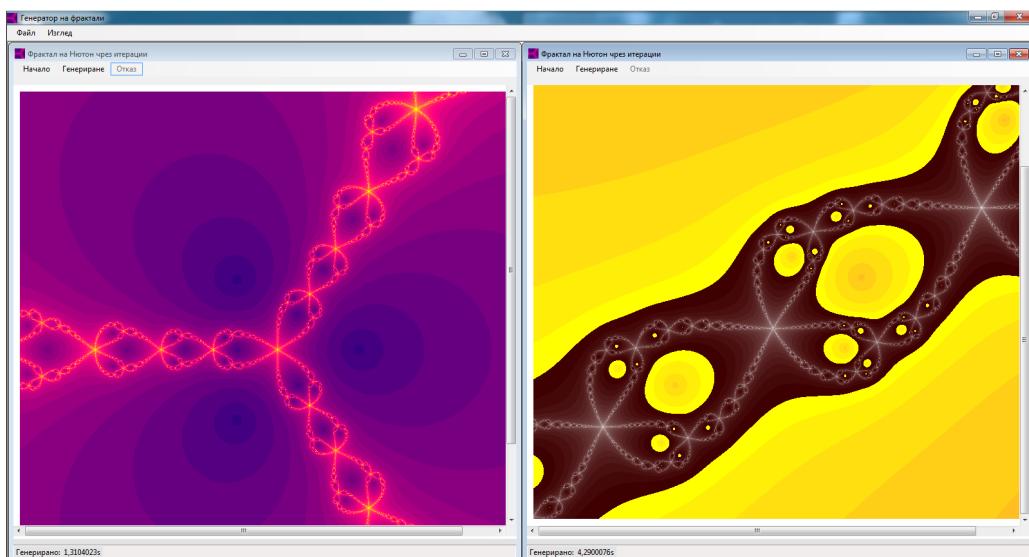
9. В долния край на програмата може да видите статуса на текущото изчисление



10. Може да преместите или преоразмерите вътрешния прозорец и дори да отворите нови до него



11. В менюто Изглед -> Подреди може да изберете готово подреждане (полезно при повече от един прозорец)



12. Може да затворите програмата от бутона за изход в горния десен ъгъл или от менюто Файл -> Изход или Начало -> Изход