СОФИЙСКИ УНИВЕРСИТЕТ
"СВ. КЛИМЕНТ ОХРИДСКИ"

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

Изобразяване на фрактал

проект по
Системи за паралена обработка

Изготвил: Калоян Стоилов, ф.н. 81609, КН,курс 3, група 5

Ръководител: проф. Васил Цунижев

8 юни 2020 г.

1 Въведение

Целта на проекта е изобразяване на фрактал, използвайки многонишкови изчисления, при решаването на следната задача от "Манделбродов" тип:

Дефинираме евклидова норма $\|\cdot\|$ в комплексната $\mathbb{C}$ така - $\|z\| = \sqrt{Re(z)^2 + Im(z)^2}$. Нека са фиксирани $z_0 \in \mathbb{C}$, и фунцкия $f : \mathbb{C} \mapsto \mathbb{C}$. За произволна точка от равнината $c \in \mathbb{C}$ търсим дали редицата $\{z_n\}_{n=0}^{\infty}$, която ще наричаме орбита на z_0, получена по следната рекурентна зависимост:

$$\forall n(z_{n+1} = f(z_n)) \tag{1}$$

притежава следното свойство:

$$\exists M_{\in\mathbb{R}} \forall n(\|z_n\| \leq M) \tag{2}$$

Точките с горното свойство образуват фрактално множество. Изобразяването на точките е в зависимост дали притежават свойството (2), или колко бързо е достигнато достатъчно условие за неналичието му.

Поставени са математическите условията $z_0 = 0$ и $f(z) = e^{z^2+c}$.

Поставени са програмните условия да е възможно използването на произволен брой нишки в изчисленията, да се изобразява произволен правоъгълник в $\mathbb{C}$, да се създава изображение с произволна резолюция, избор на грануларност. Тези променливи стойности да се задават като командни параметри във форматиран вид, а при липсата на някой от тях се взимат подразбиращи се стойности: максимално една нишка се използва, за изобразяването на $D = \{x, y \ : \ x \in [-2, 2] \wedge y \in [-2, 2]\}$, създавайки изображение 640x480.

Обсъждат се:

- Математическите възможности за "лесно"определяне точките, принадлежащи M и тези принадлежащи на $\mathbb{C} \setminus M$.

- Разпределението на изчисленията и възможните грануларности.
- Избора на оцветяването.

2 Математически анализ на проблема

Първо ще направим кратък анализ на широко известното множество на Манделброт, тъй като доста от идеите, свързани с компютърното изобразяване на фрактали, са следствие от опити за неговото представяне. Множеството на Манделброт (ще го бележим с $\mathfrak{M}$) се състои от точките $c \in \mathbb{C}$ равнината, изпълняващи условията 1 и 2, при условия $z_0 = 0$ и $f(z) = z^2 + c$.

Твърдение: $\exists n(\|z_n\| > 2) \implies c \notin \mathfrak{M}$
Доказателство:

1. Да допуснем, че $\|c\| > 2$. Ще покажем с индукция, че:
   $\forall n_{>0} \, \|z_n\| \geq 2^{n-1} \|c\|$:
   - База: $z_1 = f(z_0) = z_0^2 + c = 0^2 + c = 0 + c = c$, т.е. $\|z_1\| = \|c\|$
   - Нека е вярно за произволно n. Но тогава $z_{n+1} = z_n^2 + c = z_n^2 - (-c)$ и съответно:
     $\|z_{n+1}\| \geq \left\|z_n^2\right\| - \|-c\| = \|z_n\|^2 - \|c\| \geq (2^n \|c\|)^2 - \|c\| = 2^{2n}\|c\|^2 - \frac{\|c\|^2}{\|c\|} = (2^{2n} - \frac{1}{\|c\|})\|c\|^2 \geq (2^{2n} - \frac{1}{2})\|c\|^2 \geq 2^{2n-1}\|c\|^2 > 2^{2n} \|c\| \geq 2^n \|c\|$

   Така $\|c\| > 2 \implies \lim_{n\to\infty} \|z_n\|$ и $c \notin \mathfrak{M}$.

2. Да допуснем, че $\|c\| \leq 2$ и $\exists n \exists a_{>0} \, \|z_n\| = 2 + a$. Тогава ще покажем с индукция, че $\forall k \, \|z_{n+k}\| \geq 2 + (k+1)a$:
   - База: $\|z_{n+0}\| = 2 + (0+1)a$

- Да допуснем, че е вярно за някое k. Тогава: $\|z_{n+k+1}\| \geq \left\|z_{n+k}^2\right\| - \|-c\| \geq \left\|z_{n+k}^2\right\| - 2 = \|z_{n+k}\|^2 - 2 = (2+(k+1)a)^2 - 2 = (k+1)^2a^2 + 4(k+1)a + 2 > 2 + (k+2)a$

Редицата от норми отново е разходяща и съответно $c \notin \mathfrak{M}$.

Това твърдение се използва при определянето дали точка е от множеството на Манделброт или не, както ще покажем по-късно.

За нашата задача обаче не е толкова лесно да се изведе такова твърдение. Да забележим, че: $\|z_{n+1}\| = \left\|e^{z_n^2+c}\right\| = e^{Re(z_n^2+c)} = e^{Re(z_n^2)+Re(c)} = e^{Re(z_n)^2 - Im(z_n)^2 + Re(c)}$, тоест зависи и много от самото разположение на z_n и c. Нещо повече, в общия случай, ротацията на z_n влияе тази на z_{n+1} по следния начин: $z_{n+1} = \|z_{n+1}\|(cos(Im(z_n^2+c)) + isin(Im(z_n^2+c)) = \|z_{n+1}\|(cos(2Re(z_n)Im(z_n)+c) + isin(2Re(z_n)Im(z_n)+c))$ и прилагайки отново, виждаме доста сложна зависимост на z_{n+2} от z_n и c. Едва ли може да се каже нещо в общия случай.

Може да забележим обаче, че ако $c \in \mathbb{R}$, то $z_1 \in \mathbb{R}$, а оттам и $\forall n(z_n \in \mathbb{R})$.В такъв случай, ако $c \geq 0$ веднага получаваме, че $lim_{n\to\infty} z_n = \infty$. Ако пък $c < 0$, то $z_1 = e^c$,$z_2 = e^{e^{2c}+c}$ и т.н. Но $z_1 \xrightarrow[c\to\infty]{} 0$, $z_2 \xrightarrow[c\to\infty]{} 0$ и т.н. Тоест при достатъчно малки стойности на c, то ще е от нашето фрактално множество. При отрицателни стойности близо до 0 се вижда, че c не е в множеството ни. Точката, оказваща се повратна е решението на $e^{2c} + c = 0$.

Тъй като не можахме да намерим достатъчно условие за това дали точка е от търсеното множеството или не в общия случай, ще обсъдим евристични методи за приближеното му намиране по-долу.

3 Общ вид на алгоритъм за намиране на фрактал в $\mathbb{C}$

Първо да отбележим, че компютърът само може да приближи фрактално множество, но не и да го намери точно.Обаче това не е проблем, ако се цели главно естетиката на полученото изображение.

Обикновено се постъпва по следният начин за създаване на изображение:

1. На всеки пиксел се съпоставя число в комплексната равнина. Тъй като компютърните изображения са с правоъгълни измерения се съпоставят точки от даден правоъгълник в комплексната равнина. "Наслагвайки"растера върху правоъгълника получаваме множество от правоъгълни подобласти, като на всеки пиксел отговаря точка от съответния му регион. Обикновено се избира центърът, но ако желаем можем и горен,ляв край или друга точка.

2. За така определените точки се прилага многократно функцията $f(z)$ в цикъл. Цикълът продължава докато не е достигнат максимален определен брой итерации или достатъчно условие за неограниченост на нормите(затова и в много източници при изобазяване на $\mathfrak{M}$ това е $\|z_n\| > 2$).

3. В зависимост от типа на оцветяване:

   - Итерацията се подава като аргумент на оцветяваща функция, която определя цвета на съответния пиксел.
   - Итерациите се запазват в масив. След приключване изчисленията за всички точки, цветът се определя по "нормирана"форма на итерацията, например разделятено на броя итерации за пиксела, върху общия брой на цялото изображение.

4. В зависимост от имплементацията, информацията за пикселите

се запазва на момента от тяхното изчисление или след всички изчисления направо се запазва цялото изображение.

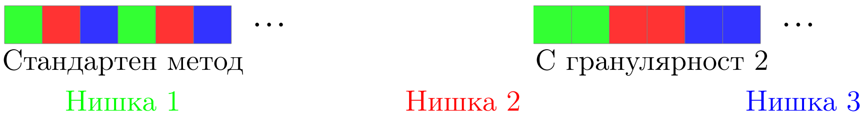
Ще отбележим, че такава е методологията и за създаване на фрактали от "Джулиев"тип.

4 Разпределение на работата

За решението на задачата трябва да направим еднотипни изчисления на множество от пиксели. Затова ще използваме SPMD. Работата може да се извърши асинхронно.
При голям брой нишки е възможно да се определят 1-2,които да се грижат само за оцветяването, като тогава ще се наложи синхронизиране - асинхронно подаване на съобщения. Ако се желае моментално записване на информацията в изображението след пресмятането на цвета, то може и да са необходими семафори(в зависимост от това как работи форматът на изображението и дадените функции за обработката му.
Ще разгледаме няколко общи метода на статично разпределение на работата, като ще се опитаме да дадем обща оценка за техните качества - ефикасност, възможност за гранулярност, сложност на имплементация.

I. Поелементен метод

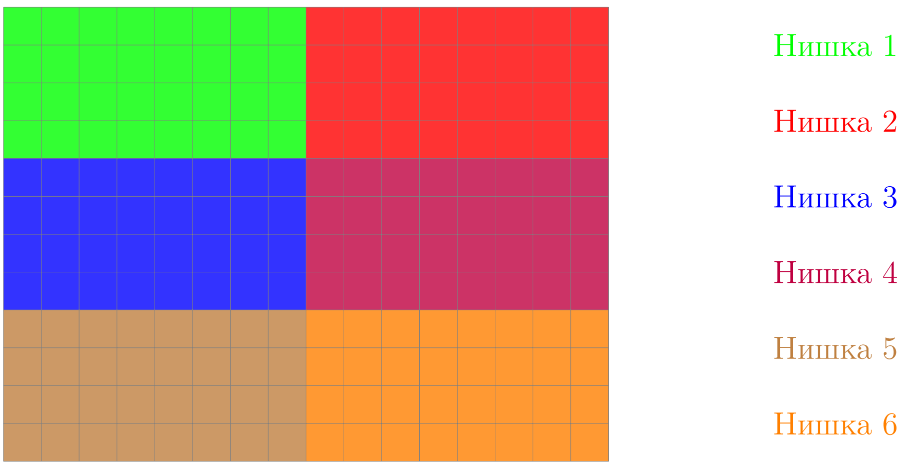
Да разгледаме как би било добре да се разпредели работата, ако всяка нишка обработва "блокове"от единични пиксели. В хода на програмата ще се наложи да пазим поне едно от следните: масив за итерации, масив за пиксели, отворено изображение. Такъв тип структури обикновено се реализират чрез разпределение в паметта на една последователност от данни, образуваща линеен масив от елементите на двумерния, като информацията се пази ред по ред. Имайки предвид това и асоциативността на кеша на процесора, при разглеждане на пикселите "последователно"би било най-добре да правим изчисления за тях ред по ред, като зададем статично циклично работа на нишките по начина показан по-долу.



Възможна е гранулация - например число k, така че блоковете да не са от по един, а от по k пиксела. Така големите стойности на k водят до груба гранулация.

II. Регионен метод

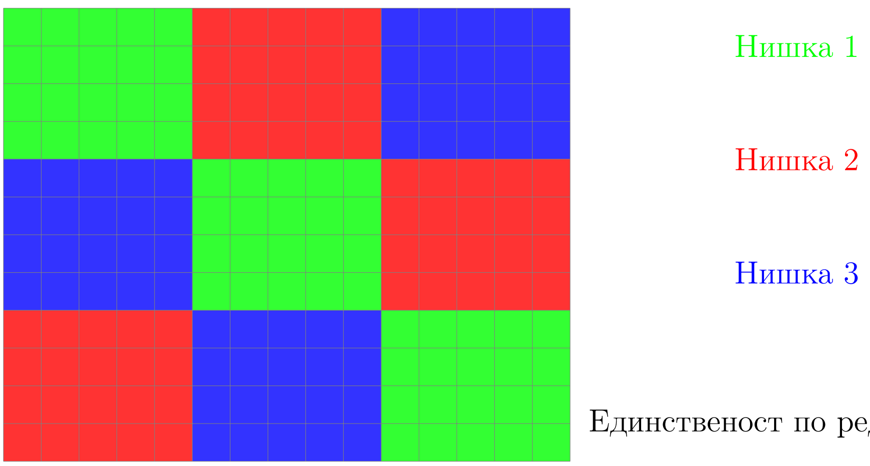
Желаем да разпределим изображението на правоъгълни региони, като всяка нишка прави изчисления само за зададения ѝ регион. Достигаме много бързо до проблем - какво правим, ако имаме нечетен брой нишки? Единият вариант е да не използваме предоставения максимален брой нишки, но това би било неразумно. Възможно е също да изчислим размера на "нечетния"регион. Отчитайки казаното за кеша би било по-разумно да го разположим с по-дългата част да обхване широчината, а по-късата да е по височината на изображението. Тъй като регионите ще са със сравнително голяма дължина, не би трябвало да имаме проблеми със запазването на информацията в кеша, колкото в предния подход. Заради зависимостта от четност, имплементацията ще се усложни. Този подход не се подлага на гранулярност.



III. Квадратен метод

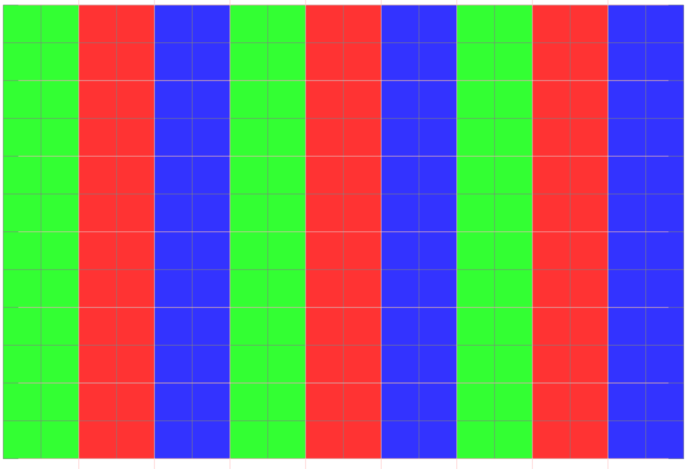
За него беше споменато на лекции. При n нишки разделяме работата на квадратна мрежа $n \times n$. Разпределянето им по нишки може да стане по следните начини:

- Произволно - не много добра идея, защото може да се окаже, че някоя от нишките не върши почти нищо, а за сметка на това поне една друга ще трябва да извърши повече изчисления.
- Циклично - по редове или колони на мрежата. Тук не би трябвало да има значение от разпределението за по-ниските нива на кеша, но с оглед например L3 вероятно е по-добре да разпределим по редове. Получаваме проблем, защото може да се окаже, че нишка може да смята голям свързан регион, където е възможно да има много числа от фракталното множество и да отнеме повече време.
- Единственост по ред и стълб - подсигуряваме всяка от нишките да изчислява в определена двойка $(ред, стълб)$ по само веднъж. Например задаваме работата последователно на първия ред, след което "превъртаме"последователността и прилагаме за долния ред. Пресмятането на "превъртането"е лесна задача използвайки смятане $(mod\ n)$.



Единственост по ред и стълб

Този метод е по-разумент от другите два. Също така лесно може да се въведе гранулярност като число k и мрежата да се разпада на $kn \times kn$(т.е. големите числа водят до фина грануларност). Тогава обаче последният начин е неприложим. Този метод не е сложен за имплементация.



Цикличноно разпределение
с гранулярност 2

IV. Линеен метод

Това е реализираният метод в проекта. С оглед казаното по-горе, разделяме изчисленията ред по ред. Всеки ред се изчислява от единствена нишка, като разпределянето е циклично, започвайки от първата създадена. Малко вероятно е една и съща нишка да попадне многократно на по-големи(в сравнение с другите нишки) участъци от по-сложни изчисления,т.е. би трябвало работата да е добре разпределена.

Прост линеен метод

Линеен метод с гранулярност 2

Методът позволява грануларност - някакво чиско k, т.ч. изчисленията стават не ред по ред, а в групи от k реда(т.е. базовият случай е k=1). Така по-големи k водят до по-груба гранулярност. Методът е елементарен за имплементация.

5 Имплементация

За изобразяване на фрактала е направена имплементация на C++, използваща главно стандартни библиотеки и дефинирането на структура RGB, пазеща по един байт съответно за червен, зелен и син цвят на пиксела. Избраният формат на изображението е PPM - това е тип bitmap, с прост и кратък header. Изображението се запазва ред по ред, започвайки от най-горния, като в използвания тип P6, файлът е в двоичен запис. Програмата работи по следният начин:

1. Попълват се дадените стойности по подразбиране в новосъздадени променливи.

2. Приемат се стойности на съответните командни параметри, ако са зададени.

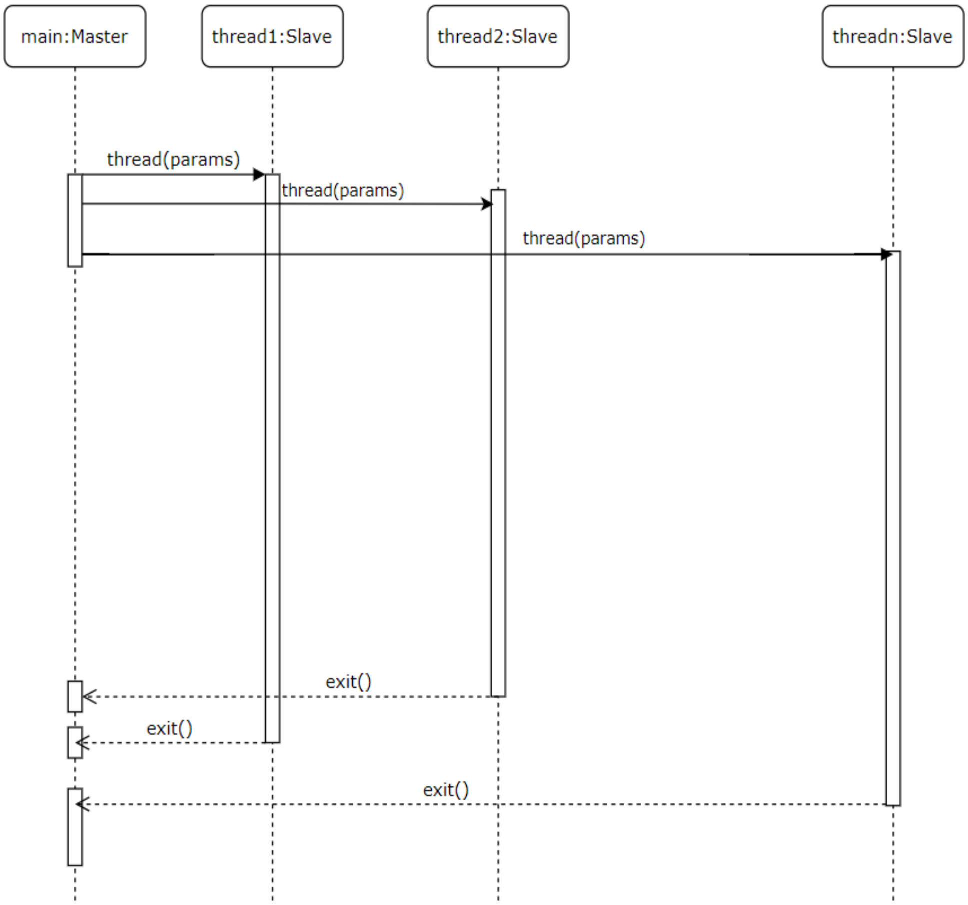
3. Създават се масив за нишките и указател към двумерен масив от RGB.

4. Main пуска максималния брой нишки, след което изчаква завършването им.

5. Нишките правят изчисления по "линейния метод" от предната глава и сами попълват информация в RGB масива всеки път щом приключат с итерирането за съответния пиксел.

6. Main създава PPM файла, попълва заглавната му част, а след това записва информацията от попълнения масив с RGB стойности.

7. Терминация

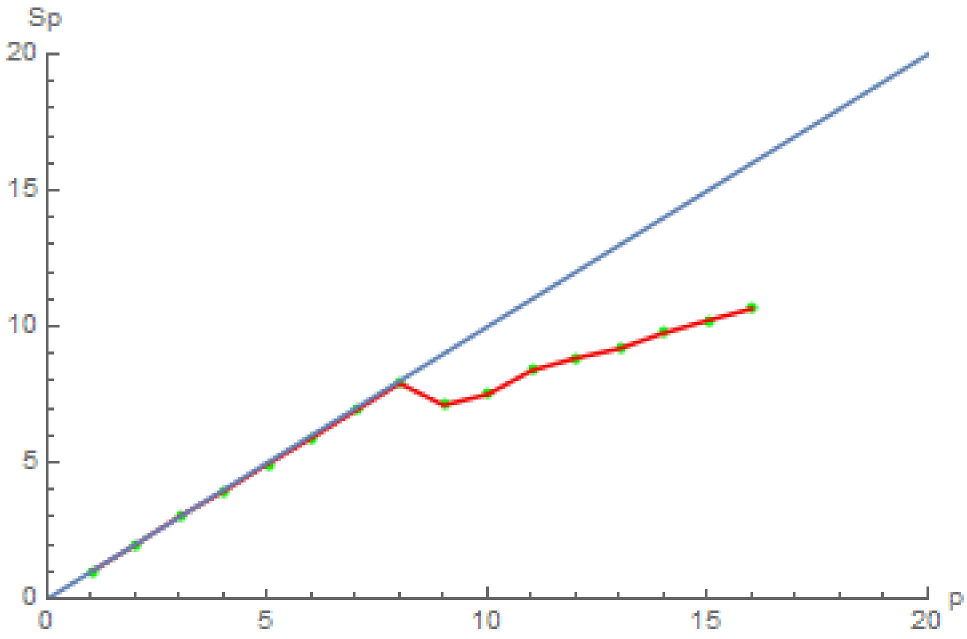


Понеже не бе намерено достатъчно условие, което да поставим в цикъла за пикселите, е поставено следното чисто евристично условие: Ако за някоя итерация z_n напусне кръга, определен от описаната около правоълната област окръжност(т.е. при $\left\|z_n - \frac{(x_{min}+x_{max})+(y_{min}+y_{max})i}{2}\right\| > \frac{\sqrt{(x_{max}-x_{min})^2+(y_{max}-y_{min})^2}}{2}$, то ще приключваме итерирането. Друго възможно условие би било например да се разглежда разстоянието м/у c и z_n.

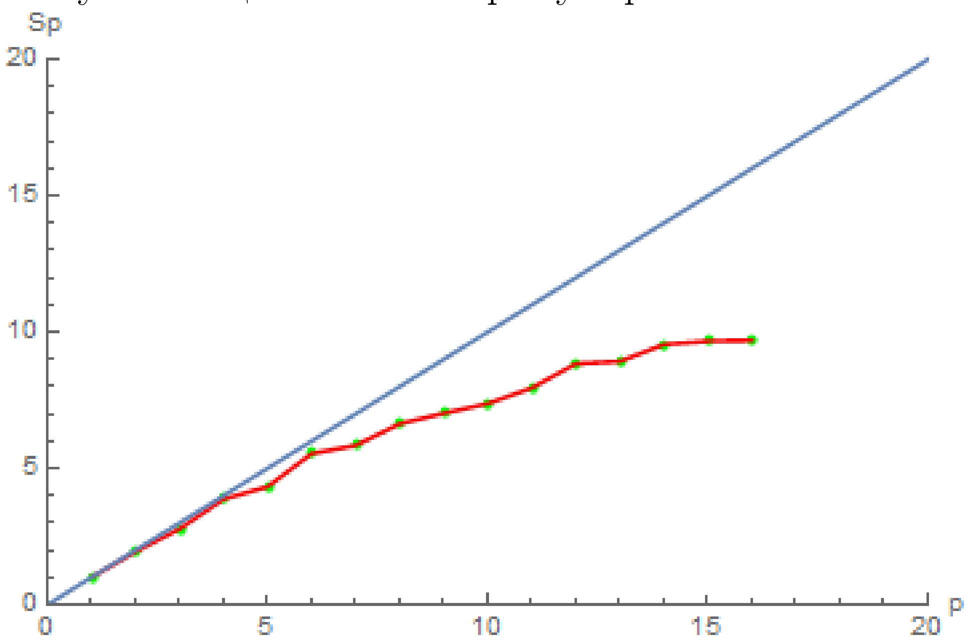
6 Тестове

Тестовете са изпълнени на машина със следната конфигурация: CPU - AMD Ryzen 1700 3.0GHz, 8 ядра, 16 нишки, общо кеш L1-768KB, L2-4MB ,L3-16MB; RAM - DDR4 16GB ок.3000MHz.

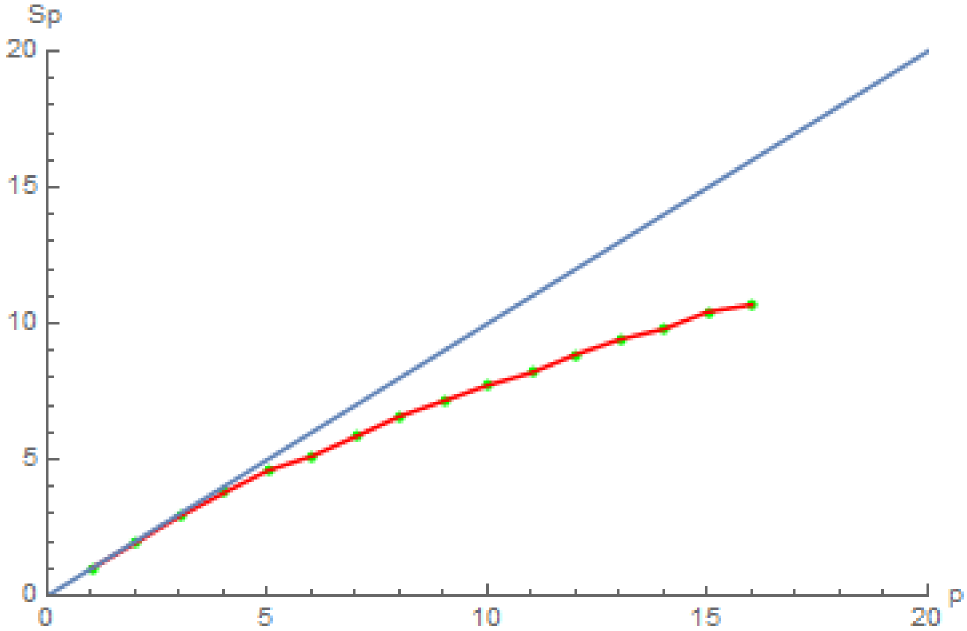
1. Grayscale оцветяване с гранулярност 1



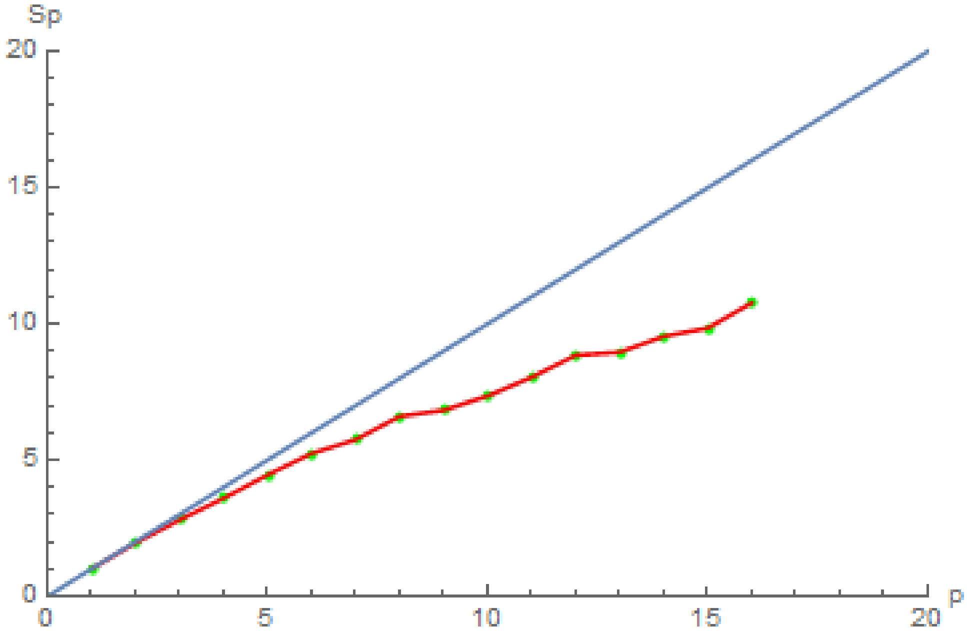
2. Grayscale оцветяване с гранулярност 5

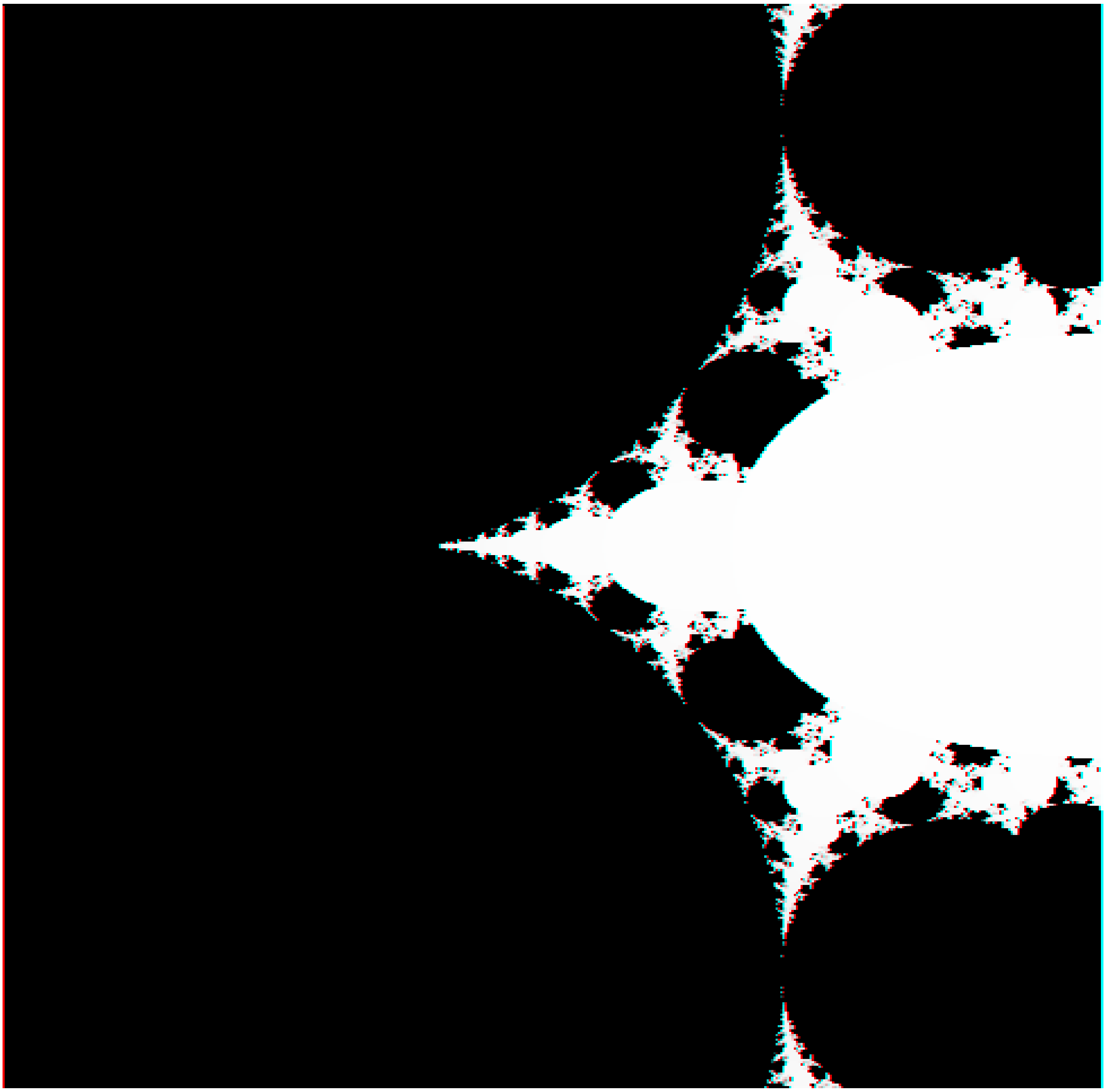


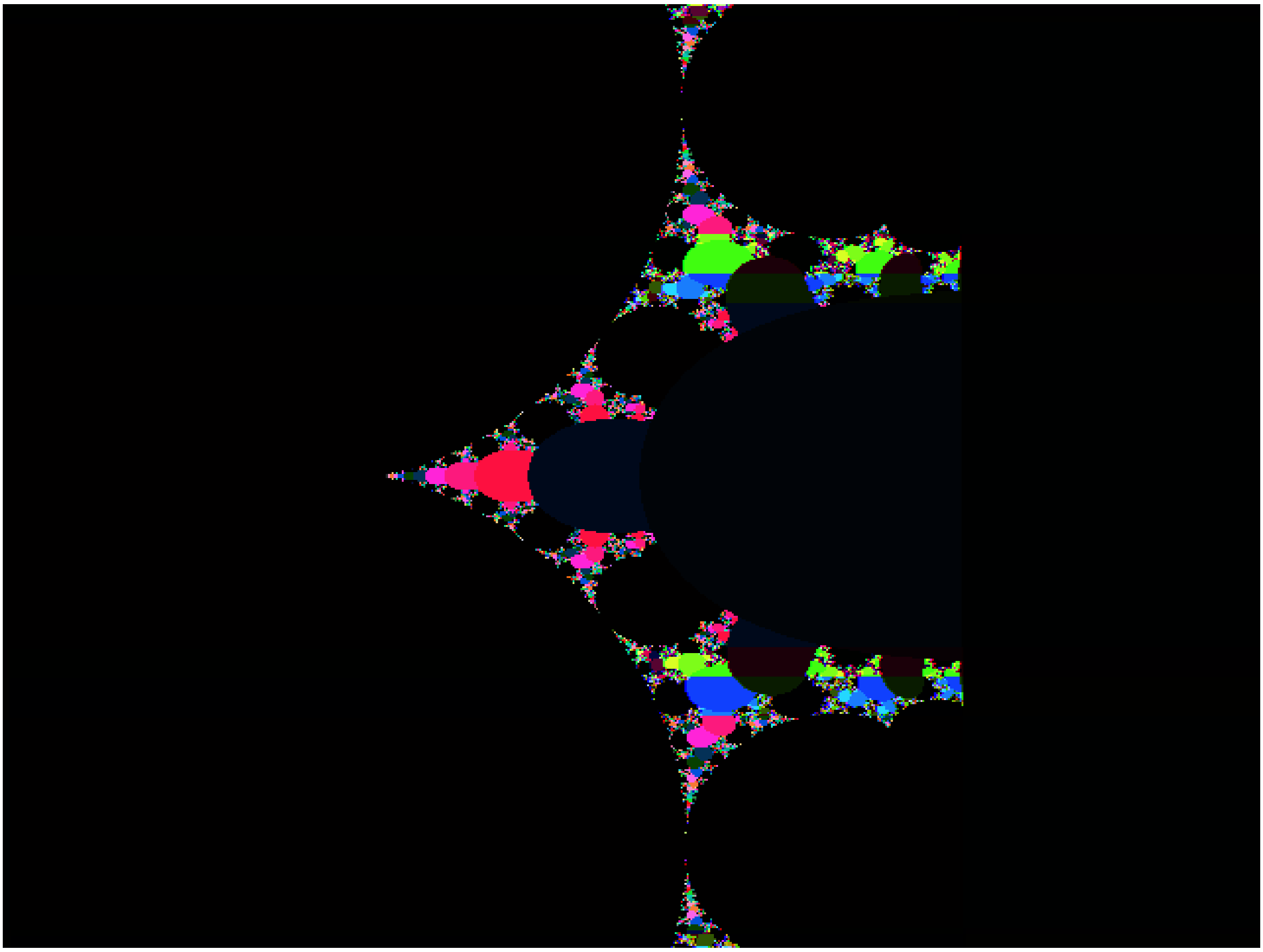
3. Просто многоцветно оцветяване с гранулярност 1



4. Просто многоцветно оцветяване с гранулярност 5







Литература

[1] Andrew Williams, *Computing the Mandelbrot set*, plus.maths.org, 01.09.1999, `https://plus.maths.org/content/os/issue9/features/mandelbrot/index`

[2] Robert L. Devaney, *Unveiling the Mandelbrot set* , plus.maths.org, 01.09.2006, `https://plus.maths.org/content/os/issue9/features/mandelbrot/index`

[3] Peter Afled, *The Mandelbrot Set*, math.utah.edu, 10.08.1998, `https://www.math.utah.edu/~alfeld/math/mandelbrot/mandelbrot.html`

[4] Simon Bridge, *Parallel Programming & the Mandelbrot Set*, codeproject.com,01.06.2012, `https://www.codeproject.com/articles/395627/parallel-programming-the-mandelbrot-set`

[5] Carmen Pughineanu, *Parallel processing and the Mandelbrot set*, Technical Sciences and Applied Mathematics , "Ştefan cel Mare" University, Suceava, Romania, 2008, `http://www.afahc.ro/ro/revista/Nr_2_2008/ART_CARMEN.pdf?fbclid=IwAR2c0UvCGDygEIzHT5oNFzDStPo5T5g6xJ8MLG-xTRVpkeSkxTGV6ADYG5A`

[6] Ian Foster, *Designing and Building Parallel Programs*, 1995, *3.6 Evaluating Implementations*

[7] Javier Barrallo , Santiago Sanchez, *Fractals and multi layer coloring algorithms*,The University of the Basque Country,2009