

Projeto B: Conjunto de Mandelbrot

Vinicio Patriarca Miranda Miguel RA: 260731

9 de julho de 2025

Sumário

1	Introdução	2
1.1	Representação Visual do Conjunto de Mandelbrot	2
1.2	Algoritmo para Gerar o Conjunto de Mandelbrot	2
2	Exercícios	3
2.1	Exercício 1: Exploração do Conjunto de Mandelbrot	3
2.2	Exercício 2: Estudando f_c e sua dinâmica	4
2.3	Exercício 3: Construção e Exploração do Conjunto de Mandelbrot	7
3	Projeto	11
3.1	Motivação da Escolha da Linguagem	11
3.2	Definições e Flags de Execução	11
3.3	Exemplo de Uso com e sem a Flag <code>--julia</code>	11
3.4	Constantes e Domínio do Problema	12
3.5	Resolução da Imagem	12
3.6	Intervalo do Plano Complexo	12
3.7	Kernel do Código e Tentativas de Otimização	12
3.8	Paralelismo	13
4	Considerações Finais	14

1 Introdução

O conjunto de Mandelbrot é um dos exemplos mais conhecidos de fractais. Ele é definido matematicamente como o conjunto de números complexos $c \in \mathbb{C}$ para os quais a sequência definida por:

$$z_{n+1} = z_n^2 + c, \quad z_0 = 0$$

permanece limitada, ou seja, não diverge para o infinito.

Em termos de órbitas, cada ponto c no plano complexo gera uma órbita, que é a sequência de valores $\{z_0, z_1, z_2, \dots\}$ obtida pela iteração da fórmula acima[1]. Se a órbita permanece confinada dentro de uma região finita do plano complexo, dizemos que o ponto c pertence ao conjunto de Mandelbrot. Caso contrário, se a órbita diverge para o infinito, o ponto não pertence ao conjunto.

Visualmente, o conjunto de Mandelbrot é representado no plano complexo, onde cada ponto c é colorido de acordo com o comportamento de sua órbita. Se a órbita não diverge, o ponto pertence ao conjunto e é geralmente colorido de preto. Caso contrário, o ponto é colorido de acordo com a rapidez com que a órbita diverge.

1.1 Representação Visual do Conjunto de Mandelbrot

Abaixo está uma representação visual do conjunto de Mandelbrot. Para gerar essa imagem, utilizamos um algoritmo que verifica se cada ponto no plano complexo pertence ao conjunto, iterando a fórmula acima e analisando o comportamento da órbita até um número máximo de iterações.

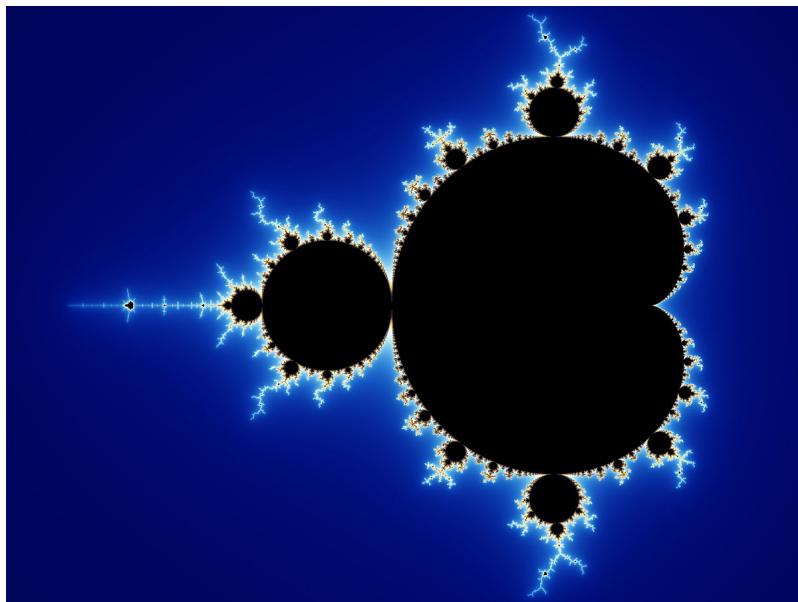


Figura 1: Representação visual do conjunto de Mandelbrot.

1.2 Algoritmo para Gerar o Conjunto de Mandelbrot

O algoritmo básico para gerar o conjunto de Mandelbrot pode ser descrito como:

1. Para cada ponto c em uma grade do plano complexo:
 - (a) Inicialize $z_0 = 0$.
 - (b) Itere a fórmula $z_{n+1} = z_n^2 + c$ até um número máximo de iterações ou até $|z_n| > 2$.
 - (c) Se $|z_n| \leq 2$ após o número máximo de iterações, considere o ponto como pertencente ao conjunto.
2. Atribua cores aos pontos com base no número de iterações necessárias para divergir.

Essa abordagem permite criar imagens detalhadas e coloridas do conjunto de Mandelbrot, revelando sua estrutura fractal fascinante.

2 Exercícios

2.1 Exercício 1: Exploração do Conjunto de Mandelbrot

Considerando o conjunto de Mandelbrot como descrito na introdução, seja M o conjunto de pontos que o compõem, $f_c : \mathbb{C} \rightarrow \mathbb{C}$ a função geradora desses pontos, dada por $f_c(z) = z^2 + c$, para todo $c \in \mathbb{C}$, de forma que $|f_c^n(z)| < \infty$, quando $n \rightarrow \infty$. Em outras palavras, a órbita de c não diverge a ∞ . No caso de M , temos a condição inicial $z_0 = 0$, ou seja, todas as órbitas partem do mesmo ponto.

- (a) **Explicita os primeiros termos da órbita de f_c para um c fixo.** Teste se os seguintes pontos estão em M : $c_1 = -2$, $c_2 = -2i$, $c_3 = 0.35e^{i\pi/4}$.

Para $c_1 = -2$, temos $z_0 = -2$, $z_1 = 2$, $z_2 = 2$, $z_3 = 2$, $z_4 = 2$, $z_5 = 2$, $z_6 = 2$, $z_7 = 2$, $z_8 = 2$ e $z_9 = 2$. Conclusão: A órbita não diverge, mas não está em M pois não satisfaz a condição de convergência.

Para $c_2 = -2i$, temos $z_0 = -2i$ e $z_1 = -4 - 2i$. Conclusão: A órbita diverge para ∞ . $c_2 \notin M$.

Para $c_3 = 0.35e^{i\pi/4}$ (aproximado como $c_3 = 0.247487 + 0.247487i$), temos $z_0 = 0.247487 + 0.247487i$, $z_1 = 0.247487 + 0.369987i$, $z_2 = 0.171847 + 0.430622i$, $z_3 = 0.091584 + 0.395489i$, $z_4 = 0.099463 + 0.319928i$, $z_5 = 0.155026 + 0.311129i$, $z_6 = 0.174719 + 0.343954i$, $z_7 = 0.159710 + 0.367678i$, $z_8 = 0.137808 + 0.364931i$ e $z_9 = 0.133304 + 0.348068i$. Conclusão: A órbita não diverge e parece convergir. $c_3 \in M$.

- (b) **Prove que M é simétrico em relação ao eixo x .**

Para provar que M é simétrico em relação ao eixo x , considere um ponto $c \in \mathbb{C}$ tal que $c = a + bi$, onde $a, b \in \mathbb{R}$. O conjugado complexo de c é dado por $\bar{c} = a - bi$.

Queremos provar, por indução matemática, que para toda $n \in \mathbb{N}$,

$$\overline{z_n} = w_n,$$

onde:

$$\begin{cases} z_0 = 0 \\ z_{n+1} = f_c(z_n) = z_n^2 + c \end{cases} \quad \text{e} \quad \begin{cases} w_0 = 0 \\ w_{n+1} = f_{\bar{c}}(w_n) = w_n^2 + \bar{c}. \end{cases}$$

Base da indução: Para $n = 0$,

$$\overline{z_0} = \bar{0} = 0 = w_0.$$

Portanto, a propriedade vale para $n = 0$.

Hipótese: Suponha que para algum $n \geq 0$,

$$\overline{z_n} = w_n.$$

Passo: Vamos mostrar que então

$$\overline{z_{n+1}} = w_{n+1}.$$

Calculando z_{n+1} :

$$z_{n+1} = f_c(z_n) = z_n^2 + c.$$

Tomando o conjugado:

$$\overline{z_{n+1}} = \overline{z_n^2 + c} = \overline{z_n}^2 + \bar{c}.$$

Pela hipótese de indução, $\overline{z_n} = w_n$, então:

$$\overline{z_{n+1}} = w_n^2 + \bar{c} = f_{\bar{c}}(w_n) = w_{n+1}.$$

Conclusão: Por indução matemática, a propriedade vale para todo $n \in \mathbb{N}$:

$$\overline{z_n} = w_n \quad \forall n \in \mathbb{N}.$$

Implicação geométrica: Isso significa que a sequência de iterações de f_c , partindo de 0, tem comportamento simétrico em relação ao eixo real (eixo x) quando comparamos c e \bar{c} .

- (c) **O que acontece com a órbita de um ponto que está fora de M ?**

Se um ponto está fora de M , sua órbita diverge para ∞ . Isso ocorre porque, por definição, os pontos em M são aqueles cujas órbitas permanecem limitadas. Para pontos fora de M , a sequência $\{z_n\}$ gerada por f_c cresce indefinidamente em magnitude, ou seja, $|z_n| \rightarrow \infty$ quando $n \rightarrow \infty$. Geometricamente, isso significa que esses pontos não pertencem ao conjunto de Mandelbrot.

- (d) **O que acontece com a órbita de um ponto que está dentro de M ?**

Se um ponto está dentro de M , sua órbita não diverge para ∞ . Em vez disso, ela permanece limitada e pode exibir diferentes comportamentos dinâmicos, dependendo do ponto c :

- **Convergência a um ponto fixo ou periódico:** A órbita pode convergir para um ponto fixo ou para um ciclo periódico. Nesse caso, o ponto fixo ou o ciclo periódico é chamado de atrator.
- **Comportamento caótico:** Em algumas regiões de M , a órbita pode exibir comportamento caótico, sem convergir para um ponto fixo ou ciclo periódico, mas ainda assim permanecendo limitada.

Geometricamente, isso significa que os pontos dentro de M estão associados a dinâmicas estáveis ou limitadas da função f_c .

2.2 Exercício 2: Estudando f_c e sua dinâmica

Pontos periódicos de uma função são aqueles cujos valores se repetem após um número finito de iterações, assim, existe um n (chamado de período) tal que $f_c^n(\hat{z}) = \hat{z}$. Se $n = 1$, \hat{z} é chamado de ponto fixo. Observe que, para cada c , os pontos periódicos de f_c serão diferentes dos pontos de sua órbita. Porém, se a órbita é convergente, existirá um subconjunto dela que converge a cada \hat{z} . Nesse caso, ele é chamado de atrator e a seguinte desigualdade é válida: $|(f_c^n)'(\hat{z})| < 1$.

- (a) **A função f_c^n é holomorfa? Se sim, qual sua derivada complexa?**

Sim, a função f_c^n é holomorfa para todo $n \in \mathbb{N}$. Como $f_c(z) = z^2 + c$ é um polinômio, ela é holomorfa em todo o plano complexo \mathbb{C} . Além disso, a composição de funções holomorfas também é holomorfa, logo a função iterada $f_c^n(z) = \underbrace{f_c \circ f_c \circ \cdots \circ f_c}_{n \text{ vezes}}(z)$ é holomorfa.

A derivada de f_c^n é obtida aplicando a regra da cadeia sucessivas vezes. Assim, temos:

$$(f_c^n)'(z) = f'_c(f_c^{n-1}(z)) \cdot f'_c(f_c^{n-2}(z)) \cdots f'_c(z) = \prod_{k=0}^{n-1} f'_c(f_c^k(z))$$

Como $f_c(z) = z^2 + c$, sua derivada é $f'_c(z) = 2z$. Substituindo:

$$(f_c^n)'(z) = \prod_{k=0}^{n-1} 2f_c^k(z) = 2^n \cdot \prod_{k=0}^{n-1} f_c^k(z)$$

Portanto, f_c^n é holomorfa e sua derivada é dada pelo produto acima.

- (b) **Ache a região de M que contém os pontos c de forma que f_c possua um ponto fixo. Faça o mesmo para os c que fazem f_c ter um ponto de período igual a 2. Identifique essas duas regiões em um mesmo gráfico.**

Iteração	Expressão	Função
1	z_n	$f_c^{(1)}(z_n) = z_n^2 + c$
2	$z_{n+1} = z_n^2 + c$	$f_c^{(2)}(z_n) = (z_n^2 + c)^2 + c$

- Para ponto fixo: Agora igualando para os valores de z_n e f_c^1 , obtemos os pontos fixos:

$$z_n = z_n^2 + c \quad (1)$$

Resolvendo a equação quadrática:

$$z^2 + c = z \implies z^2 - z + c = 0$$

As raízes da equação são dadas por:

$$z = \frac{1 \pm \sqrt{1 - 4c}}{2}$$

Para que o ponto fixo seja atrator, a condição $|f'_c(z)| < 1$ deve ser satisfeita. Como $f'_c(z) = 2z$, temos:

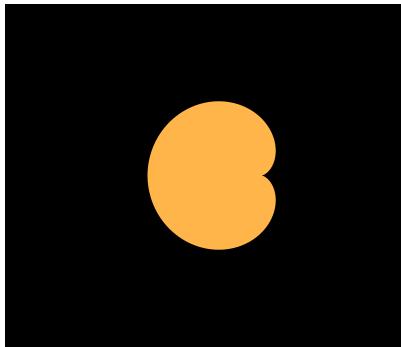
$$|2z| < 1 \implies |z| < \frac{1}{2}.$$

Substituindo $z = \frac{1 \pm \sqrt{1 - 4c}}{2}$:

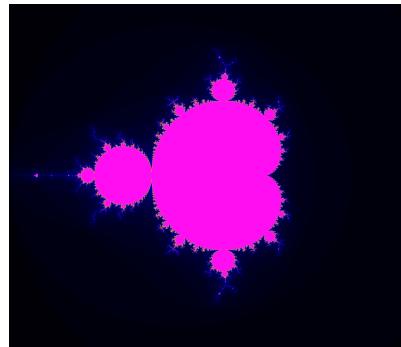
$$\left| \frac{1 \pm \sqrt{1 - 4c}}{2} \right| < \frac{1}{2} \implies |1 \pm \sqrt{1 - 4c}| < 1.$$

Portanto, a região de c que satisfaz essa condição pode ser determinada resolvendo a desigualdade acima. Para identificar essa região, foi implementado um código que gera o gráfico correspondente. Para executar o código, utilize o seguinte comando:

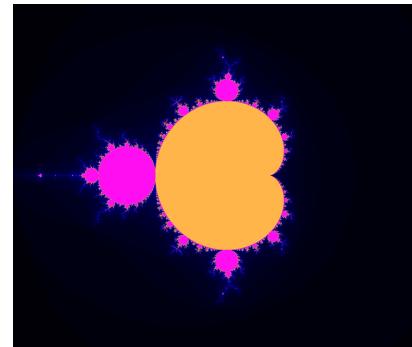
```
bin/mandelbrot --questoes
```



(a) Regiões de M contendo os pontos c para os quais f_c possui um ponto fixo.



(b) Regiões de M clássica.



(c) Sobreposição das regiões de M contendo os pontos c para os quais f_c possui um ponto fixo.

- Para período igual a 2: Agora igualando para os valores de z_n e f_c^2 , obtemos os pontos de período 2 ($n = 2$):

$$(z^2 + c)^2 + c - z = 0 \quad (2)$$

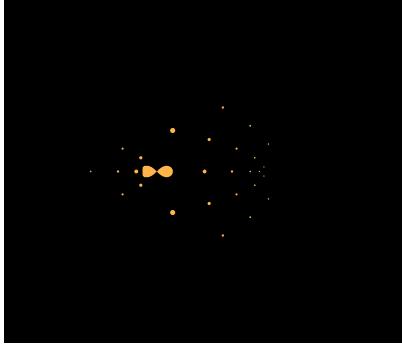
Para cada solução z , verificamos se $f_c(z) \neq z$ (ou seja, não é ponto fixo) e se é atrator, satisfazendo a condição $|4z(z^2 + c)| < 1$.

A condição para atrator de período 2 é:

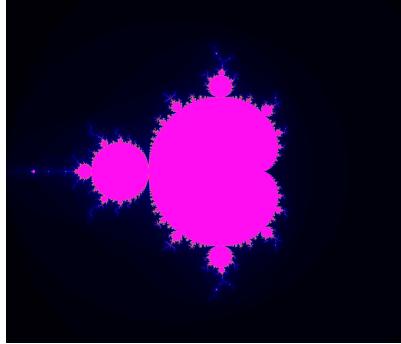
$$|4z(z^2 + c)| < 1 \quad (3)$$

Para encontrar a região dos c em que f_c tem um ponto de período 2 atrator, resolvemos numericamente a equação acima. O método consiste em discretizar o plano complexo em uma grade de pontos e, para cada ponto c , verificar se existe algum z que satisfaz Equação 3. Para isso, iteramos sobre valores iniciais de z e calculamos a função f_c até encontrar uma solução que satisfaz as condições de atrator. Para esse ponto verificamos se satisfaz a Equação 2 com uma tolerância de 0.01. Após o cálculo, o resultado é salvo como uma imagem que mostra as regiões de M contendo os pontos c para os quais f_c possui um ponto de período igual a 2.

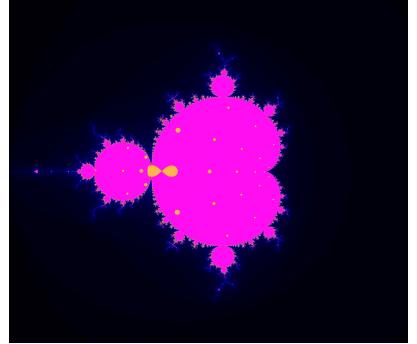
A figura abaixo mostra as regiões de M contendo os pontos c para os quais f_c possui um ponto fixo e um ponto de período igual a 2.



(d) Regiões de M contendo os pontos c para os quais f_c possui um ponto de período igual a 2.



(e) Regiões de M clássica.



(f) Sobreposição das regiões de M contendo os pontos c para os quais f_c possui um ponto fixo e um ponto de período igual a 2.

(c) Prove que um ponto c pertence a M se $|z_n| \leq 2$, para todo $n = 1, 2, \dots$

Suponha, por contraposição, que $|z_n| > 2$ para algum $n \geq 1$. Vamos mostrar que, nesse caso, a sequência $\{z_n\}$ diverge, ou seja, $|z_n| \rightarrow \infty$ quando $n \rightarrow \infty$.

Sabemos que $f_c(z) = z^2 + c$. Assim, para $|z_n| > 2$, temos:

$$|z_{n+1}| = |f_c(z_n)| = |z_n^2 + c| \geq |z_n|^2 - |c|.$$

Como $|z_n| > 2$, temos $|z_n|^2 > 4$. Além disso, $|c|$ é finito, então:

$$|z_{n+1}| > |z_n|^2 - |c| > 4 - |c|.$$

Portanto, $|z_{n+1}| > |z_n|$ para $|z_n| > 2$, o que implica que a sequência $\{|z_n|\}$ é estritamente crescente para $|z_n| > 2$. Como $|z_n|$ cresce indefinidamente, concluímos que $|z_n| \rightarrow \infty$ quando $n \rightarrow \infty$.

Por contraposição, se $|z_n| \leq 2$ para todo $n \geq 1$, então a sequência $\{z_n\}$ não diverge, ou seja, $c \in M$.

(d) Prove que o intervalo de números reais puros que pertencem a M é $[-2, 0.25]$.

Para provar que o intervalo de números reais puros que pertencem a M é $[-2, 0.25]$, iremos fazer o limite de n indo a ∞ e analisar o comportamento da função $f_c(z) = z^2 + c$.

Para isso, consideraremos a condição de que o módulo da derivada da função iterada deve ser menor que 1 para garantir que a órbita permaneça limitada:

$$\lim_{n \rightarrow \infty} \left| \frac{z_{n+1}}{z_n} \right| < 1$$

Como o limite $n \rightarrow \infty$ está no índice do termo, podemos fazer uma substituição para deixar a notação menos carregada, substituiremos z_n por x .

$$\left| \frac{x^2 + c}{x} \right| < 1$$

$$|x^2 + c| < |x|$$

$$|x^2 - x + c| < 0$$

Podemos analisar o comportamento da função quadrática $x^2 - x + c$. Para que essa função tenha raízes reais, o discriminante deve ser não negativo:

$$\Delta = (-1)^2 - 4 \cdot 1 \cdot c = 1 - 4c \geq 0$$

Resolvendo a desigualdade:

$$1 - 4c \geq 0 \implies c \leq 0.25$$

Pelo resultado do item anterior, sabemos que c deve ser maior ou igual a -2 para que a órbita não diverja. Assim, temos:

$$-2 \leq c \leq 0.25$$

2.3 Exercício 3: Construção e Exploração do Conjunto de Mandelbrot

- (a) Ao alterar o raio de convergência e o número de iterações no código, o que acontece com a velocidade de processamento, o detalhamento e a precisão da imagem? Para isso, teste valores no código e analise a diferença no resultado.

Falta a parte do raio de convergência

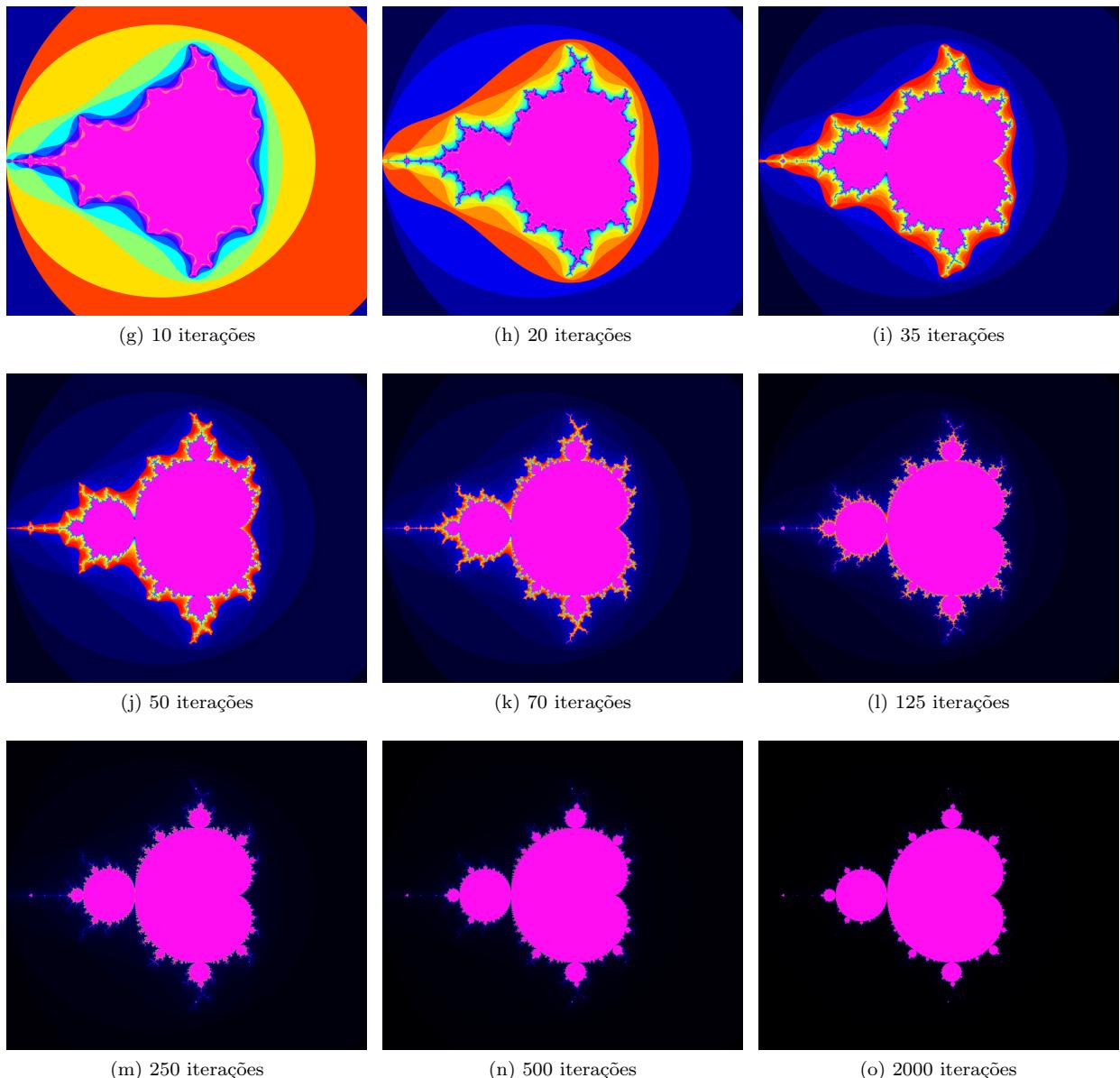


Figura 2: Conjuntos de Mandelbrot para diferentes números de iterações. Cada subfigura apresenta o conjunto gerado para o respectivo número de iterações.

As cores das imagens geradas representam a taxa de divergência dos números complexos no plano. Cada cor indica o número de iterações necessárias para que o valor escape para o infinito, ou seja, para que o módulo do número complexo ultrapasse o raio de convergência definido. Abaixo está a interpretação das cores:

- **Azul:** Representa os pontos que divergem rapidamente, escapando para o infinito em poucas iterações.
- **Verde:** Indica pontos que levam um número moderado de iterações para divergir.
- **Amarelo e Laranja:** Representam pontos que estão próximos à fronteira do conjunto de Mandelbrot, levando mais iterações para divergir.
- **Vermelho:** Indica pontos que estão ainda mais próximos da fronteira, divergindo muito lentamente.
- **Rosa:** Representa os pontos que não divergem, ou seja, pertencem ao conjunto de Mandelbrot.

Essa paleta de cores ajuda a visualizar a complexidade e os detalhes do conjunto de Mandelbrot, destacando as regiões de transição entre os pontos que pertencem ao conjunto e os que não pertencem.

Ao alterar o número de iterações, conseguimos identificar com maior precisão os números complexos que divergem. Um número maior de iterações permite capturar mais detalhes do conjunto de Mandelbrot, evidenciando regiões de transição entre os pontos que pertencem ao conjunto e os que não pertencem. No entanto, isso também aumenta o tempo de processamento, pois mais cálculos são necessários para determinar a convergência ou divergência de cada ponto.

- (b) A Outros fractais podem ser gerados a partir do Mandelbrot, uma família de exemplos é obtida ao alterar a condição inicial $z_0 = 0$ para outros valores. Gere alguns exemplos desses conjuntos.

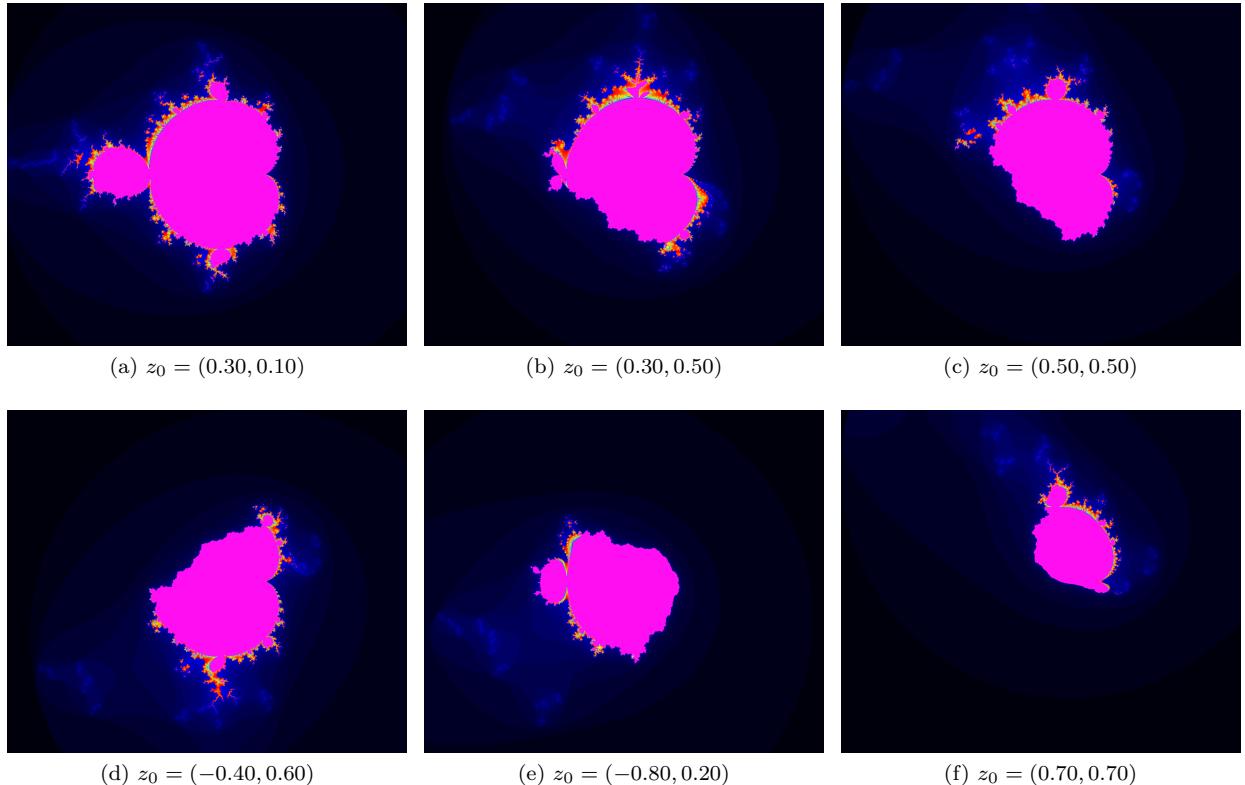


Figura 3: Conjuntos de Mandelbrot para diferentes valores de z_0 e iterações. Cada subfigura apresenta o conjunto gerado para o respectivo valor de z_0 .

Tabela 1: Índice das imagens, valores de z_0 e módulo de z_0 .

Índice	z_0	$ z_0 $
(a)	$(0.30, 0.10)$	$\sqrt{0.30^2 + 0.10^2} \approx 0.316$
(b)	$(0.30, 0.50)$	$\sqrt{0.30^2 + 0.50^2} \approx 0.583$
(d)	$(0.50, 0.50)$	$\sqrt{0.50^2 + 0.50^2} \approx 0.707$
(c)	$(-0.40, 0.60)$	$\sqrt{(-0.40)^2 + 0.60^2} \approx 0.721$
(e)	$(-0.80, 0.20)$	$\sqrt{(-0.80)^2 + 0.20^2} \approx 0.824$
(f)	$(0.70, 0.70)$	$\sqrt{0.70^2 + 0.70^2} \approx 0.990$

Uma análise interessante é que, à medida que o valor de $|z_0|$ aumenta, o conjunto de Mandelbrot aparenta diminuir em tamanho (redução da área de cor rosa na figura). Isso possivelmente ocorre devido à maior probabilidade de os valores iterados escaparem para o infinito, reduzindo a região de convergência.

- (c) Pode-se também alterar a forma de observar a dinâmica: ao invés de fixar a condição inicial z_0 , escolha um valor fixo para c e estude as órbitas variando os valores de z_0 , escolhendo aqueles para os quais a função $f_c^n(z_0)$ não diverge quando $n \rightarrow \infty$. Plote alguns exemplos desse novo fractal. Os fractais gerados nessa questão e na anterior são exemplos dos conhecidos conjuntos de Julia.

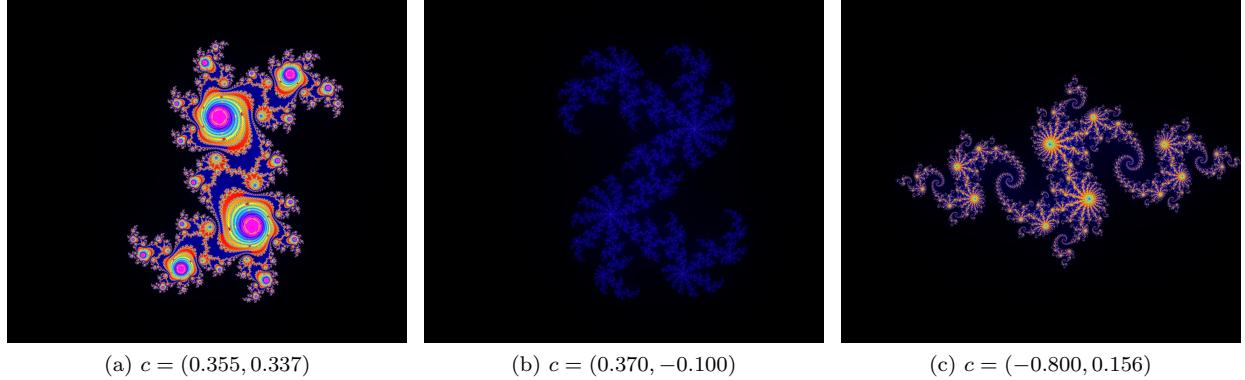


Figura 4: Conjuntos de Julia para diferentes valores de c utilizando o expoente $d = 2$. Cada subfigura apresenta o conjunto gerado para o respectivo valor de c .

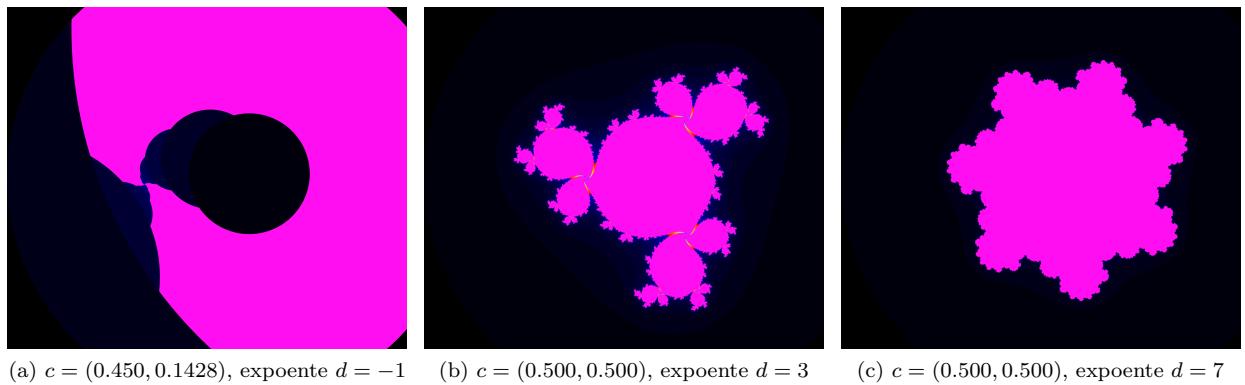


Figura 5: Conjuntos de Julia para diferentes valores de c e expoentes.

Para todos os exemplos apresentados, foram utilizadas 125 iterações, e os valores iniciais estão indicados nas legendas das figuras correspondentes. Essa uniformidade nos parâmetros permite uma análise comparativa mais clara entre as diferentes variações dos conjuntos de Mandelbrot, Julia e Multibrot, destacando como as alterações nos valores de z_0 , c e d influenciam as formas e os padrões gerados.

- (d) Uma terceira forma de alterar M é mudando o grau do polinômio f_c , substituindo d na função $f_c(z) = z^d + c$ por outros números positivos. Veja o que acontece com a figura ao usar diferentes valores. Esses conjuntos são conhecidos como *Multibrot*. Experimente também valores negativos de d .

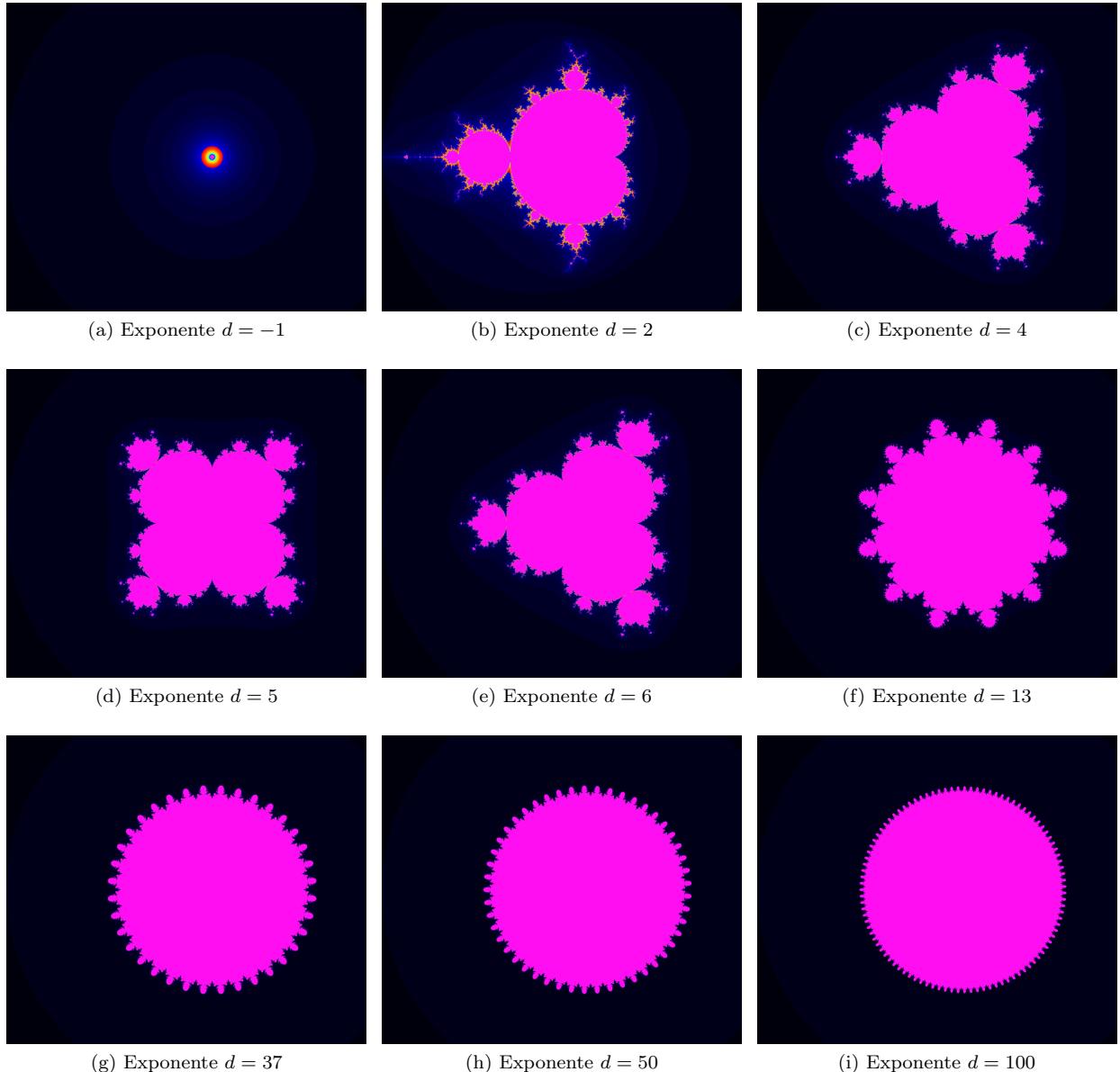


Figura 6: Conjuntos de Mandelbrot para diferentes valores do exponente d . Cada subfigura apresenta o conjunto gerado para o respectivo valor de d .

Uma análise interessante sobre os conjuntos de Mandelbrot gerados para diferentes valores do expoente d é a simetria presente nas figuras. Para valores positivos de d , observa-se que os conjuntos apresentam simetria rotacional de ordem d . Isso significa que, ao girar a figura em torno da origem por um ângulo de $360^\circ/d$ [2], a imagem resultante será idêntica à original.

Por exemplo:

- Para $d = 2$, a figura apresenta simetria rotacional de 180° .
- Para $d = 5$, a figura apresenta simetria rotacional de 72° .
- Para $d = 7$, a figura apresenta simetria rotacional de 51.43° .
- Para $d = 13$, a figura apresenta simetria rotacional de 27.69° .
- Para $d = 37$, a figura apresenta simetria rotacional de 9.73° .
- Para $d = 50$, a figura apresenta simetria rotacional de 7.2° .

Essa simetria é uma característica fundamental dos conjuntos de Multibrot e está diretamente relacionada ao grau do polinômio utilizado na função $f_c(z) = z^d + c$. À medida que o valor de d aumenta, a complexidade e o número de "braços" ou "pétais" na figura também aumentam, criando padrões mais intrincados e visualmente interessantes.

3 Projeto

Este projeto foi inicialmente desenvolvido como parte da disciplina MC970, que estou cursando neste semestre. O objetivo principal do programa é gerar imagens de fractais, um processo que exige um número elevado de iterações para alcançar um nível considerável de detalhe e permitir a percepção das características intrínsecas dessas estruturas. Devido à natureza altamente paralelizável do problema, a implementação em linguagens como C ou CUDA é ideal. No entanto, optou-se por não desenvolver uma versão em CUDA para evitar dificuldades na execução em diferentes ambientes.

3.1 Motivação da Escolha da Linguagem

Embora a linguagem Python pudesse ser utilizada para implementar o projeto, sua escolha seria ineficiente para este tipo de aplicação. Python, sendo uma linguagem interpretada e de alto nível, apresenta desempenho inferior em comparação com C para operações intensivas em processamento, como o cálculo de fractais. Além disso, o gerenciamento de threads em Python é limitado pelo Global Interpreter Lock (GIL), o que restringe a execução paralela em múltiplos núcleos de CPU. Assim, a escolha de C foi mais adequada para garantir eficiência e desempenho.

Embora a linguagem CUDA fosse uma alternativa para execução altamente paralela na GPU, sua adoção implicaria em dependências de hardware (placas compatíveis com NVIDIA CUDA), além de limitar a portabilidade e aumentar a complexidade da depuração e desenvolvimento. Assim, optou-se por manter a implementação em C, permitindo execução em qualquer sistema com suporte a threads POSIX.

3.2 Definições e Flags de Execução

A estrutura para tratamento de parâmetros via linha de comando utiliza a biblioteca `getopt_long`. As opções disponíveis estão descritas a seguir:

Listing 1: Flags de linha de comando

```
static struct option long_options[] = {
    {"threads", required_argument, 0, 't'},
    {"view", required_argument, 0, 'v'},
    {"z0", required_argument, 0, 'z'},
    {"maxIterations", required_argument, 0, 'm'},
    {"exponent", required_argument, 0, 'e'},
    {"julia", required_argument, 0, 'j'},
    {"questoes", no_argument, 0, 'q'},
    {"help", no_argument, 0, '?'},
    {0, 0, 0, 0}
};
```

3.3 Exemplo de Uso com e sem a Flag --julia

A flag `--julia` ativa o modo de geração do conjunto de Julia, que é uma variação do conjunto de Mandelbrot. No modo Julia, o valor inicial z_0 é fixado, e o cálculo é realizado para diferentes valores de c no plano complexo. Sem a flag `--julia`, o programa gera o conjunto de Mandelbrot, onde c é fixo e z_0 varia.

Exemplo sem a Flag --julia: O comando abaixo gera o conjunto de Mandelbrot com 4 threads, 1000 iterações máximas, e uma visualização padrão:

```
./programa -t 4 -m 1000
```

Exemplo com a Flag --julia: O comando abaixo gera o conjunto de Julia com 8 threads, 500 iterações máximas, e o valor inicial $z_0 = 0.355 + 0.355i$:

```
./programa -t 8 -m 500 -j -z 0.355,0.355
```

Comparação: Sem a flag `--julia`, o programa calcula o conjunto de Mandelbrot, que é baseado na variação de z_0 para um valor fixo de c . Com a flag `--julia`, o programa calcula o conjunto de Julia, que é baseado na variação de c para um valor fixo de z_0 . A escolha entre os dois modos depende do tipo de fractal que se deseja visualizar.

3.4 Constantes e Domínio do Problema

A resolução e a área do plano complexo são definidas pelas seguintes constantes:

Listing 2: Constantes principais

```
struct Config {
    long double x0 = -2.0;
    long double x1 = 1.5;
    long double y0 = -1.5;
    long double y1 = 1.5;
    long double j_re = 0.0;
    long double j_im = 0.0;
    long double z0_re = 0.0;
    long double z0_im = 0.0;
    bool juliaMode = false;
    unsigned int scale = 1;
    int maxIterations = 750;
    int exponent = 2;
    int numThreads = sysconf(_SC_NPROCESSORS_ONLN);
} config;

const unsigned int width = static_cast<unsigned int>((config.x1 - config.x0) * 1000 * config.scale;
const unsigned int height = static_cast<unsigned int>((config.y1 - config.y0) * 1000 * config.scale;
```

3.5 Resolução da Imagem

A resolução foi fixada em 7000×6000 , o que corresponde a uma qualidade 8K. Tal definição visa garantir a visualização detalhada da fronteira do conjunto de Mandelbrot, onde os padrões complexos são mais interessantes e exigem alta densidade de pixels para serem adequadamente renderizados.¹

3.6 Intervalo do Plano Complexo

O plano complexo foi definido com os seguintes limites:

$$x \in [-2, 1.5], \quad y \in [-1.5, 1.5]$$

Esse intervalo foi escolhido devido ao fato de que, para o conjunto de Mandelbrot, os valores de z que pertencem ao conjunto estão limitados a $|z| \leq 2$. Isso garante que a visualização esteja focada na região relevante do plano complexo, onde as iterações convergem e os padrões característicos do conjunto emergem.

3.7 Kernel do Código e Tentativas de Otimização

O núcleo do cálculo do conjunto de Mandelbrot é implementado na função `mandel`[3], que realiza as iterações necessárias para determinar se um ponto no plano complexo pertence ao conjunto. A função utiliza a seguinte lógica:

Listing 3: Kernel do cálculo do conjunto de Mandelbrot

```
static inline int mandel(long double c_re, long double c_im, int count, long double z0_re = 0.0,
{
    long double z_re = z0_re;
    long double z_im = z0_im;
    int i;

    for (i = 0; i < count; ++i) {
        long double r2 = z_re * z_re + z_im * z_im;

        if (r2 > 4.0L)
            break;

        if (e == 2) {
            long double new_re = z_re * z_re - z_im * z_im;
```

¹Para incluir as imagens no relatório, a resolução foi reduzida para 1400×1200 para evitar que o arquivo PDF ficasse muito pesado.

```

        long double new_im = 2.0L * z_re * z_im;
        z_re = c_re + new_re;
        z_im = c_im + new_im;
    } else {
        long double r = std::sqrt(r2);
        long double theta = std::atan2(z_im, z_re);
        long double r_e = std::pow(r, e);
        long double new_re = r_e * std::cos(e * theta);
        long double new_im = r_e * std::sin(e * theta);
        z_re = c_re + new_re;
        z_im = c_im + new_im;
    }
}

return i;
}

```

Durante o desenvolvimento, foram realizadas tentativas de otimização para encontrar os pontos fixos, porém devido ao float, não consegui encontrar um intervalo de erro que conseguia parar a execução antes e encontrei imagens distorcidas.

Abaixo, apresentamos um exemplo de imagem gerada com a tentativa de otimização tentando encontrar pontos fixos, basta comparar com [Figura 1](#) para ver as distorções introduzidas por isso:

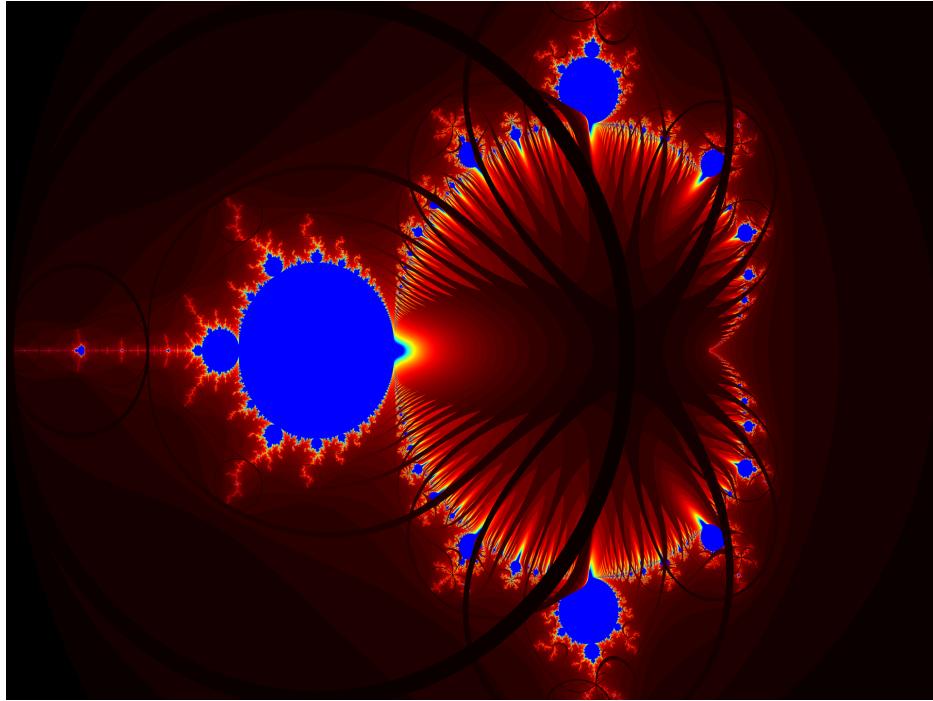


Figura 7: Imagem gerada com ponto flutuante.

Como resultado, a abordagem de encontrar pontos fixos foi descartada para preservar a qualidade das imagens geradas.

3.8 Paralelismo

A paralelização é feita utilizando múltiplas threads, controladas pelo parâmetro `--threads`. Cada thread é responsável por calcular uma fatia da imagem, dividida por linhas. O número de threads pode ser ajustado conforme o número de núcleos disponíveis na máquina.

Devido à alta resolução da imagem (15360×8640) e ao elevado número de iterações (10.000), o tempo de execução ainda pode ser significativo, mesmo com paralelismo. Isso ocorre porque o cálculo de cada ponto no conjunto de Mandelbrot é computacionalmente intensivo, especialmente nas regiões de fronteira, onde a convergência é mais lenta e exige mais iterações para determinar se o ponto pertence ao conjunto. Além disso, a sobrecarga de criação e gerenciamento de threads também contribui para o tempo total de execução.

4 Considerações Finais

O projeto exemplifica a aplicação prática de conceitos de paralelismo para resolver um problema computacionalmente intensivo. A escolha da linguagem C permitiu um controle eficiente do processo, enquanto a opção de não utilizar CUDA manteve a simplicidade e portabilidade do projeto, respeitando as restrições de infraestrutura dos alunos da disciplina.

Além disso, os exercícios realizados foram fundamentais para determinar os intervalos de interesse e compreender o conceito de pontos fixos, permitindo uma análise mais precisa e fundamentada do problema abordado. Eles também permitiram observar a simetria vertical do problema, o que possibilitou uma otimização significativa ao processar apenas metade dos dados.

Referências

- [1] M. V. d. S. Pereira, “Uma abordagem elementar do fractal árvore pitagórica.” Disponível em: <https://repositorio.ufpa.br/jspui/handle/2011/15695>. Acesso em: jun. 2025., 2023. Trabalho de Conclusão de Curso (Licenciatura em Matemática) – Faculdade de Matemática, Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém.
- [2] H.-O. Peitgen and P. H. Richter, *The Beauty of Fractals: Images of Complex Dynamical Systems*. Berlin, Heidelberg: Springer-Verlag, 1986.
- [3] Intel Corporation, “Example fractal code modified from intel’s original source.” Código-fonte, 2011. Código modificado fornecido pela Intel, sob licença BSD. Disponível sob termo de licença aberto Intel.