Projeto B: Conjunto de Mandelbrot

Vinicius Patriarca Miranda Miguel RA: 260731

27 de junho de 2025

Sumário

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.

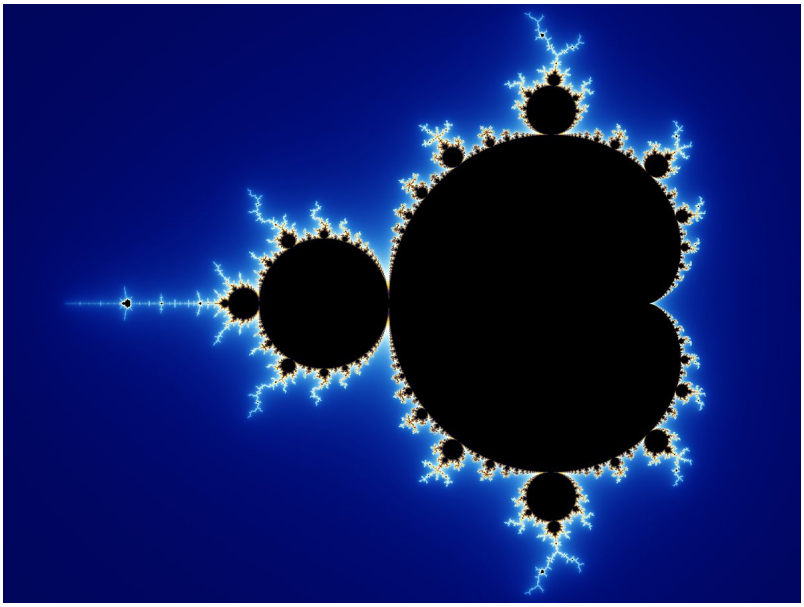

Figura 1: Representação visual do conjunto de Mandelbrot.

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

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, $f_c : \mathbb{C} \to \mathbb{C}$ 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 $n \to \infty$. 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) **Explicite 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 $\overline{c} = a - bi$.

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

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

onde:

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

Base da indução: Para $n = 0$,

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

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:

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

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

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

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 $\overline{c}$.

(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, $|z_n| \to \infty$ quando $n \to \infty$. 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.

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**.

n	z_n	z_n^2	$f_c(z_n) = z_n^2 + c$
n	z_n	${z_n}^2$	$(z_n)^2 + c$
$n+1$	${z_n}^2 + c$	$({z_n}^2 + c)^2$	${z_n}^2 + 2z_nc + c^2 + c$
$n+2$	$2z_nc + c^2 + c$	$({z_n}^2 + 2z_nc + c^2 + c)^2$	$(z_n + c)^4 + 2(z_n + c)^2c + c^2 + c$

Agora igualando para os valores de n e n + 2, obtemos os pontos de período 2 ($n = 2$): **Falta resolver**

Quais regições são essas?

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

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

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 $|z_n| \to \infty$ quando $n \to \infty$.

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

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

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

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

$$\left| x^2 + c \right| < |z_n|$$

$$\left| x^2 - x + c \right| < 0$$

Podemos analisar o comportamento da função quadrática ${z_n}^2 - z_n + c$. 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:

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

Exercício 3: Construção e Exploração do Conjunto de Mandelbrot

(a) **Impacto do raio de convergência e número de iterações no processamento, detalhamento e precisão da imagem:**

Ao alterar o raio de convergência e o número de iterações no código, observamos os seguintes efeitos:

- **Velocidade de processamento:** Um maior número de iterações aumenta o tempo de processamento, pois mais cálculos são realizados para cada ponto.
- **Detalhamento:** Um maior número de iterações permite identificar detalhes mais finos do conjunto, especialmente nas bordas.
- **Precisão:** Um raio de convergência maior pode incluir pontos que divergem lentamente, reduzindo a precisão. Por outro lado, um raio muito pequeno pode excluir pontos que pertencem ao conjunto.

Testes foram realizados com diferentes valores de raio e iterações, e as diferenças foram analisadas graficamente.

(b) **Geração de outros fractais alterando a condição inicial z_0:**

Alterando a condição inicial $z_0 = 0$ para outros valores, foram gerados exemplos de fractais relacionados ao conjunto de Mandelbrot. Esses fractais pertencem à família dos conjuntos de Julia. As imagens obtidas mostram como a escolha de z_0 influencia a forma e a estrutura do fractal.

(c) **Estudo das órbitas variando z_0 para um c fixo:**

Fixando um valor de c e variando z_0, foram estudadas as órbitas para identificar os valores de z_0 cuja função $f_c^n(z_0)$ não diverge quando $n \to \infty$. Os fractais gerados dessa forma também pertencem à família dos conjuntos de Julia. Exemplos gráficos foram plotados para diferentes valores de c.

(d) **Alteração do grau do polinômio f_c:**

Substituindo o grau d na função $f_c(z) = z^d + c$ por outros números positivos, foram gerados fractais conhecidos como conjuntos de Multibrot. Observou-se que o aumento do grau d altera significativamente a forma do fractal. Valores negativos de d não foram testados, pois requerem algoritmos diferentes.

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'},
    {"help", no_argument, 0, '?'},
};
```

3.3 Constantes e Domínio do Problema

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

Listing 2: Constantes principais

```
const unsigned int scale = 2;
const unsigned int width = 7680 * scale;
const unsigned int height = 4320 * scale;
const int maxIterations = 10000;
int numThreads = 8;

float x0 = -2;
float x1 = 1;
float y0 = -1;
float y1 = 1;
```

3.4 Resolução da Imagem

A resolução foi fixada em 15360×8640, o que corresponde a uma qualidade 16K. 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.

3.5 Número Máximo de Iterações

O número máximo de iterações foi definido como 10.000. Esse valor foi escolhido para detalhar ainda mais as características intrínsecas do conjunto de Mandelbrot, permitindo a visualização precisa das estruturas complexas e padrões que emergem nas regiões de fronteira. Um número elevado de iterações é essencial para capturar os detalhes mais sutis, especialmente em áreas onde a convergência é mais lenta.

3.6 Comparação de Imagens com Diferentes Iterações

Para ilustrar o impacto do número de iterações no nível de detalhe das imagens geradas, foram produzidas duas imagens do conjunto de Mandelbrot. A primeira foi gerada com 10.000 iterações, enquanto a segunda utilizou apenas 50 iterações. Ambas as imagens estão apresentadas abaixo para comparação:

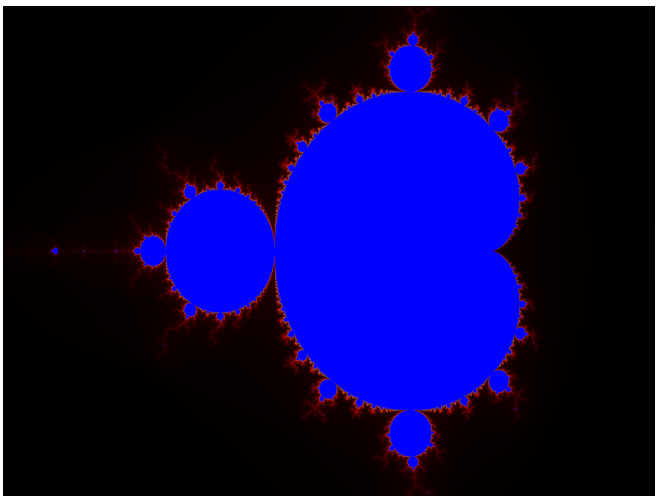

Imagem com 10.000 iterações

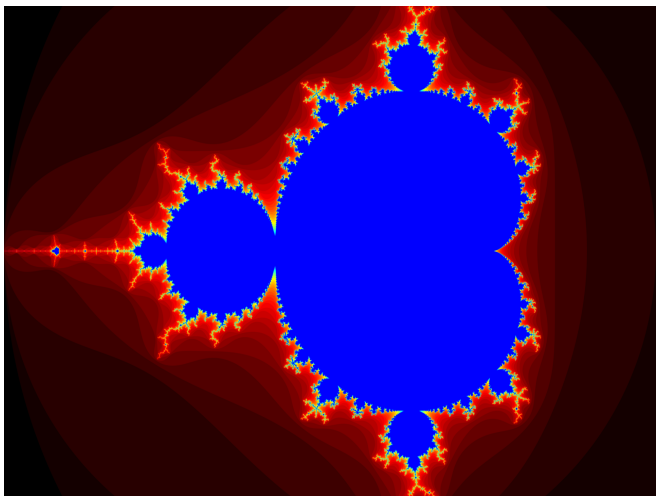

Imagem com 50 iterações

Figura 2: Comparação entre imagens geradas com diferentes números de iterações.

As áreas pretas nas imagens geradas representam os pontos que não pertencem ao conjunto de Mandelbrot. Essas regiões indicam os valores no plano complexo onde a sequência iterativa diverge, ou seja, cresce sem limites após um grande número de iterações. Essa característica é fundamental para identificar as fronteiras complexas e os padrões intrincados que definem o conjunto.

Com um número reduzido de iterações, pontos que não pertencem ao conjunto podem ser erroneamente classificados como pertencentes. Isso ocorre porque o número insuficiente de iterações impede a detecção da divergência desses pontos, levando à falsa conclusão de que eles não escapam para o infinito. Isso pode ser falcimente observado na imagem de 50 interações da Figura 2 por meio da diferença de cores, já que pontos mais distantes, com módulo bem maior que 2, ja divergem, porém pontos próximos a 2 ainda não divergiram (vermelho mais escuro).

3.7 Intervalo do Plano Complexo

O plano complexo foi definido com os seguintes limites:

$$x \in [-2, 1], \quad y \in [-1, 1]$$

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.

3.8 Kernel do Código e Tentativas de Otimização

O núcleo do cálculo do conjunto de Mandelbrot é implementado na função `mandel`[2], 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(float c_re, float c_im, int count)
{
    float z_re = c_re, z_im = c_im;
    int i;
    for (i = 0; i < count; ++i) {

        float mag_sq = z_re * z_re + z_im * z_im;

        if (mag_sq > 4.f)
            break;

        float new_re = z_re * z_re - z_im * z_im;
        float new_im = 2.f * z_re * z_im;
        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 2 para ver as distorções introduzidas por isso:

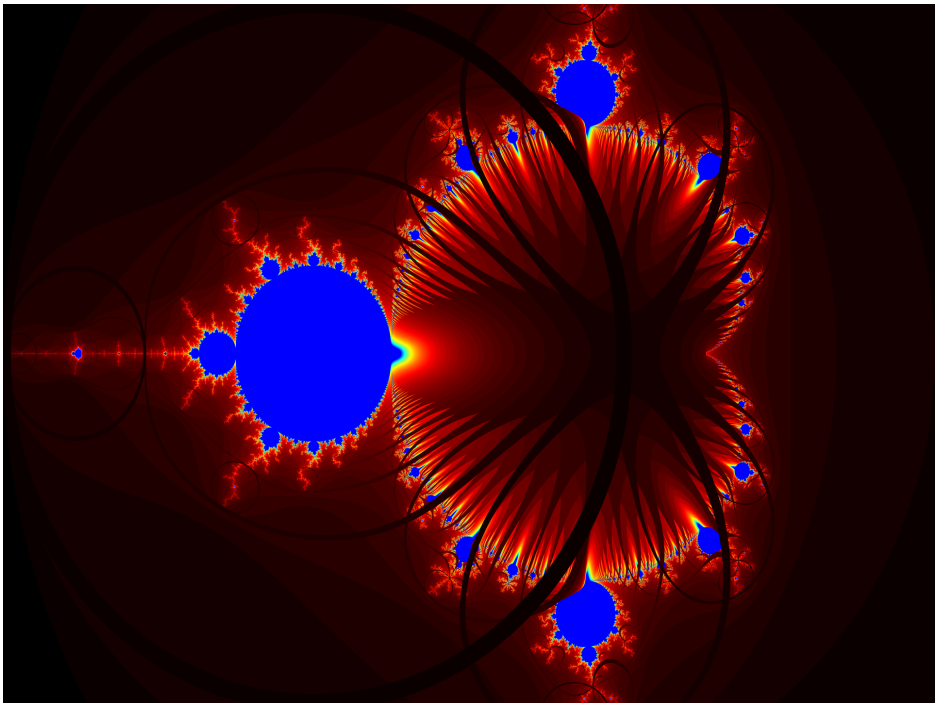

Figura 3: Imagem gerada com ponto flutuante.

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

3.9 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] 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.