Exercício de paralelização do algoritmo que gera o fractal de Buddhabrot

Raphael Y. de Camargo e Anderson Gonçalves Marco

Quadrimestre: 2017-1

Sumário

1 Oque é o fractal de Buddhabrot **3**

2 Algoritmo para gerar o fractal de Buddhabrot **4**
2.1 Descrição do algoritmo 4
2.2 Ideia para parelizar o algoritmo 4
2.3 Exemplo 5

3 Formato de imagem PGM **12**
3.1 Exemplo de imagem PGM 13

4 Objetivo do exercício **14**

1 Oque é o fractal de Buddhabrot

O fractal de Buddhabrot [2] é um fractal 2d, sendo assim o Buddhabrot é o *plot* de uma matriz 2d para um arquivo de imagem. O fractal de Buddhabrot é baseado no fractal de Mandelbroth [3] e em fractais *IFS* [5]. A figura 1 mostra a imagem do fractal de Buddhabrot.

O fractal de Buddhabrot foi descrito pela primeira vez em 1993 por Melinda Green. O nome Buddhabrot é uma combinação das palavras Buda e Mandelbroth, isto é porque o Buddhabrot se parece com a forma de um Buda [1] e também porque é baseado no fractal de Mandelbroth.

A figura 1 mostra um fractal de Buddhabrot.

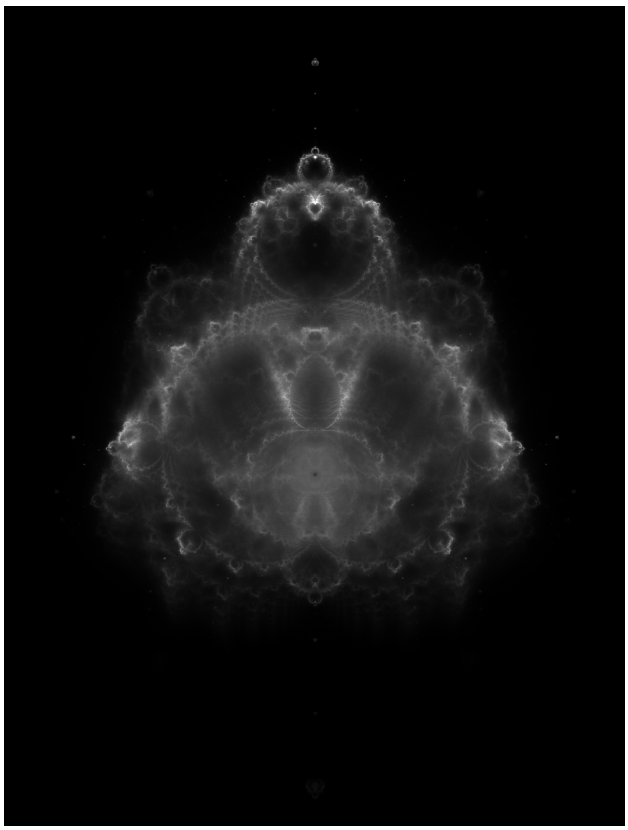

Figura 1: Fractal de Buddhabrot.

2 Algoritmo para gerar o fractal de Buddhabrot

A subseção 2.1 possui uma descrição do algoritmo de que gera o fractal de Buddhabrot. Esta descrição foi extraída da Wikipédia [2].

2.1 Descrição do algoritmo

Matematicamente o conjunto de Mandelbrot consiste no conjunto de pontos c no plano complexo para os quais a seqüência iterativa definida por $Z_{n+1} = {Z_n}^2 + C$ com n tendendo ao infinito.

O Buddhabrot, entretanto, é representado ao se criar um vetor bidimensional de contadores, um para cada pixel da tela. Então um conjunto aleatório (ou homogêneo, dependendo da implementação) de pontos de amostragem c é iterado na função de Mandelbrot. Para os pontos que escapam num dado número de iterações (ou seja pontos que não fazem parte do conjunto de Mandelbrot) os contadores correspondentes aos pontos zn de cada iteração são incrementados para cada vez que zn for igual a posição do contador.

Após a iteração sobre um número grande de pontos em c as cores (ou as cores e a intensidade) dos pixels são escolhidos com base nos valores de cada contador.

2.2 Exemplo

Esta subseção possui um exemplo de uso para o algoritmo que gera o fractal do Buddhabrot. Neste exemplo foram considerados apenas 4 pontos inicias, mas o Buddhabrot requer milhões de evoluções para ser *plotado* em uma imagem. Não existe nenhum regra dizendo qual deve ser quantidade de pontos inicias, mas ela deve ser grande.

Uma parte importante deste exemplo são as figuras 3, 4, 5 e 6, que mostram números complexos em planos cartesianos. Para representar números complexos em um plano cartesianos foi considerado que eles são pontos do plano, a componente y do ponto seria a parte imaginaria do número complexo e a componente x do ponto seria a parte real do número complexo.

Para ilustrar o uso do algoritmo descrito na subseção 2.1 vamos fazer um exemplo com 4 evoluções. Vamos considerar os pontos $0.3 + 0.6i$,$-0.3 + 0.6i$, $-0.2 + 0.3i$ e $-0.7 + -0.6i$ como sendo os pontos iniciais das evoluções. As figuras 3, 4, 5 e 6 mostram a "pseudo-evolução" destes pontos iniciais quando iterados pela regra $Z_{n+1} = {Z_n}^2 + C$, sendo $Z_1 = C$ o ponto inicial.

É importante dizer que a "evolução" dos pontos mostrados nas imagens das figuras 3, 4, 5 e 6 é meramente ilustrativa e não são uma evolução verdadeira

dos pontos iniciais Z_1 descrito nelas.

Vamos considerar que o termo escapar, utilizado na subseção 2.1, corresponde a dizer $|Z_n| > 2$. O termo escapar também pode ser descrito geometricamente como qualquer ponto fora dos círculos que estão nas figuras 3, 4, 5 e 6.

Seguindo com o exemplo excluiremos os pontos em que $|Z_n| > 2$, para qualquer valor de n. Também serão excluídos os pontos que pertencem a uma evolução em que $|Z_\alpha| > 2$. Os pontos que pertencem a uma evolução em que $|Z_n| \leq 2$, para $n > \beta$, também serão excluídos.

Com regras que foram definidas anteriormente, com $\alpha = 3$ e com $\beta = 7$ apenas sobram os pontos que estão dentro círculos nas figuras 3 e 4.

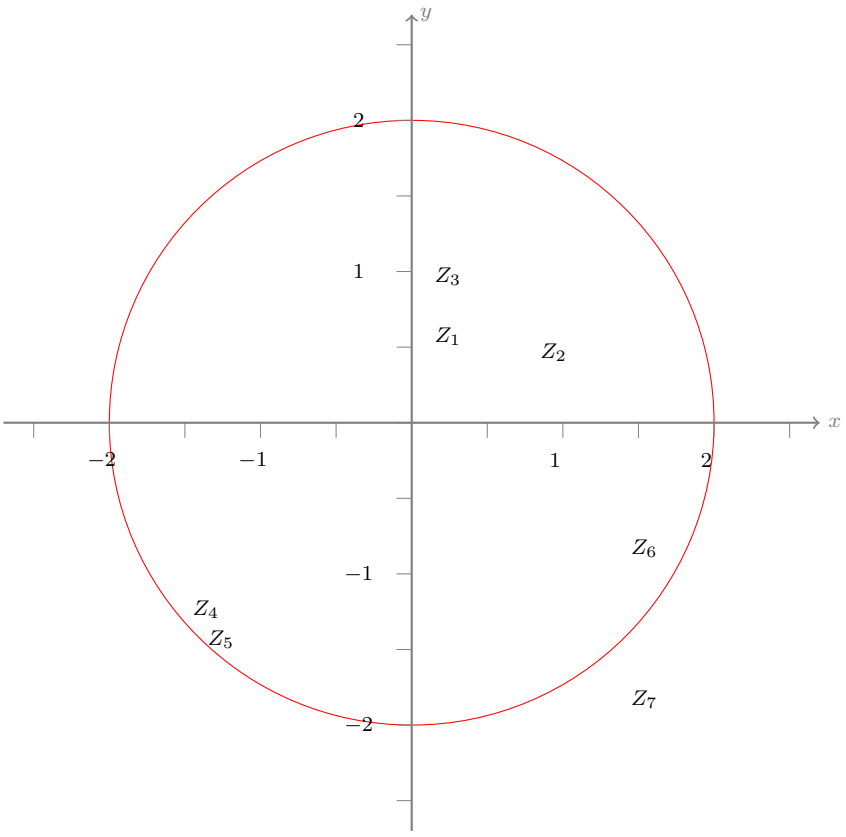

Figura 2: Exemplo de evolução de Z_n com $C = 0.3 + 0.6i$, como $|Z_3| \leq 2$ e $|Z_7| > 2$ então esta evolução será considera. Valores de Z_n são meramente ilustrativos.

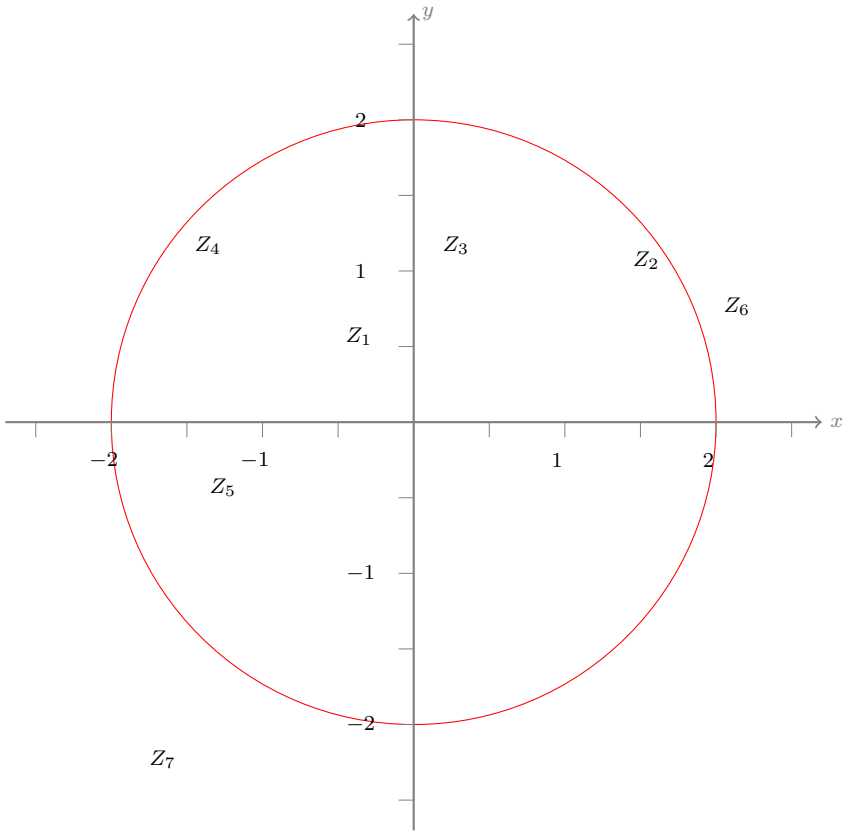

Figura 3: Exemplo de evolução de Z_n com $c = -0.3 + 0.6i$, como $|Z_3| \leq 2$ e $|Z_6| > 2$ então esta evolução será considera. Valores de Z_n são meramente ilustrativos.

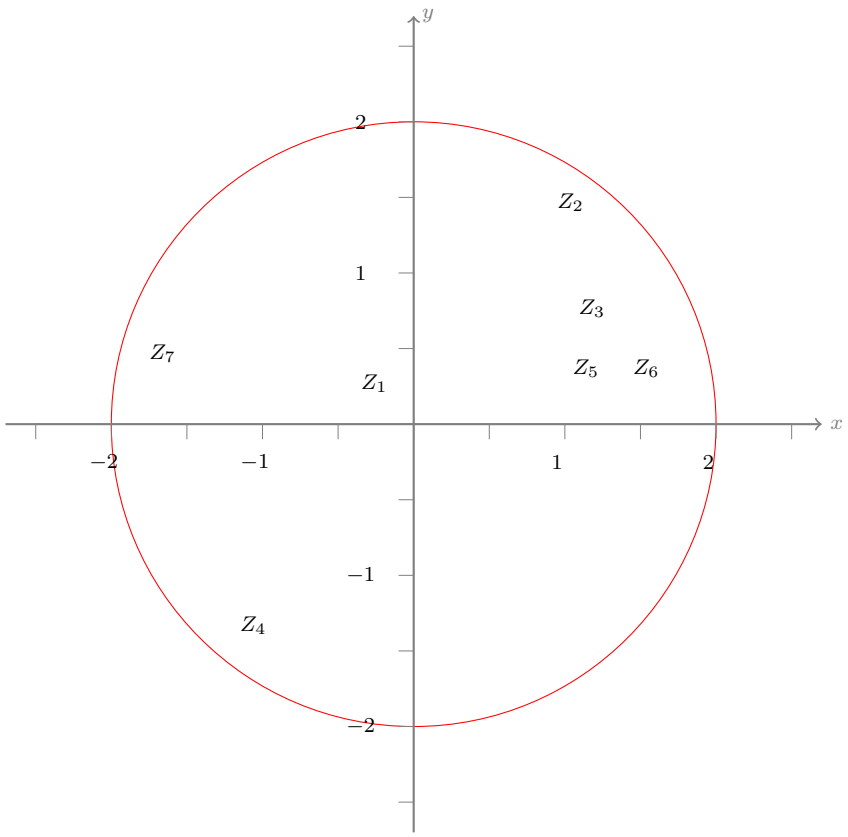

Figura 4: Exemplo de evolução de Z_n com $c = -0.3 + 0.6i$, como $|Z_7| \leq 2$ então esta evolução será desconsidera. Valores de Z_n são meramente ilustrativos.

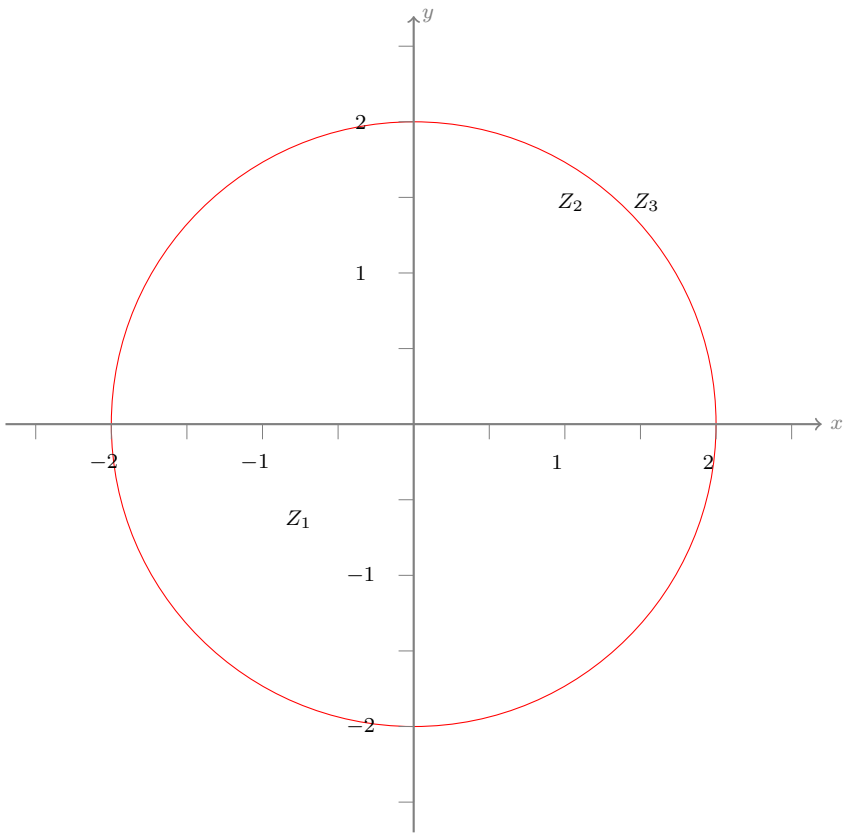

Figura 5: Exemplo de evolução de Z_n com $c = -0.3 + 0.6i$, como $|Z_3| > 2$ então esta evolução será desconsidera. Valores de Z são meramente ilustrativos.

Neste ponto é necessário se definir uma maneira de acessar os componentes real e imaginario que formam um número complexo. Considere que um número complexo $z = x + yi$, o operador $.i$ retorna o número real y e o operador $.r$ retorno o número real x. Deste modo para o número $b = 3.2 + 0.1i$ temos que $b.r = 3.2$ e $b.i = 0.1$.

Neste ponto do exemplo uma matriz deve ser definida. A matriz M foi definida como sendo uma matriz de números inteiros com γ linhas por ψ colunas, com $\gamma = 6$ e $\psi = 6$ e todos os elementos de M tendo o valor 0. Considere o operador $[\quad]$ como sendo a operação que arredonda números reais. Considere a função $f(z) = \left[\frac{z.r+2}{4}(\psi - 1)\right] + 1$ e a função $g(z) = \left[\frac{z.i+2}{4}(\gamma - 1)\right] + 1$, estas funções são responsáveis por mapear um número complexo para um ponto na matriz M.

Como exemplo de uso das funções $g(z)$ e $f(z)$ considere os números com-

plexos $a_1 = 0.5 + 0.2i$ e $a_2 = -0.7 + 0.8i$. Seguindo o exemplo para uso das funções $g(z)$ e $f(z)$ temos que $g(a_1) = g(0.5+0.2i) = \left[\frac{0.2+2}{4}6\right] + 1 = 4$, $f(a_1) = f(0.5 + 0.2i) = \left[\frac{0.5+2}{4}6\right] + 1 = 5$, $g(a_2) = g(-0.7 + 0.8i) = \left[\frac{0.8+2}{4}6\right] + 1 = 5$ e $f(a_2) = f(-0.7 + 0.8i) = \left[\frac{-0.7+2}{4}6\right] + 1 = 3$.

Utilizando para todos os pontos que restaram para incrementar os valores dos elementos da matriz M temos:

- $M_{g(0.3+0.6i),f(0.3+0.6i)} := 1 + M_{g(0.3+0.6i),f(0.3+0.6i)}$.
- $M_{g(1+0.5i),f(1+0.5i)} := 1 + M_{g(1+0.5i),f(1+0.5i)}$.
- $M_{g(-1.3-1.2i),f(-1.3-1.2i)} := 1 + M_{g(-1.3-1.2i),f(-1.3-1.2i)}$.
- $M_{g(-1.2-1.4i),f(-1.2-1.4i)} := 1 + M_{g(-1.2-1.4i),f(-1.2-1.4i)}$,
- $M_{g(1.6-0.8i),f(1.6-0.8i)} := 1 + M_{g(1.6-0.8i),f(1.6-0.8i)}$,
- $M_{g(-0.3+0.6i),f(-0.3+0.6i)} := 1 + M_{g(-0.3+0.6i),f(-0.3+0.6i)}$,
- $M_{g(1.6+1.1i),f(1.6+1.1i)} := 1 + M_{g(1.6+1.1i),f(1.6+1.1i)}$,
- $M_{g(0.34,1.2),f(0.34,1.2)} := 1 + M_{g(0.34,1.2),f(0.34,1.2)}$,
- $M_{g(-1.3,1.2),f(-1.3,1.2)} := 1 + M_{g(-1.3,1.2),f(-1.3,1.2)}$
- $M_{g(-1.2-0.4i),f(-1.2-0.4i)} := 1 + M_{g(-1.2-0.4i),f(-1.2-0.4i)}$.

O estado atual da matriz M é como o mostrado pela figura 7.

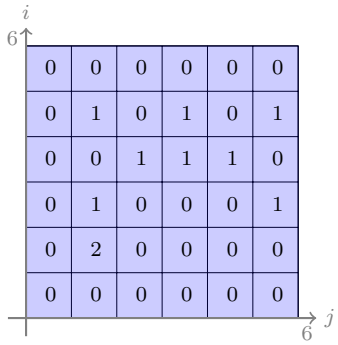

Figura 6: Estado da matriz M após seus elementos terem sido incrementados.

Agora é necessário normalizar os valores de M para que eles fiquem entre 0 e 255. Para isto vamos considerar que a variável ω guarda o valor máximo que os

elementos de M possuem, neste exemplo $\omega = 2$. Vamos também considerar uma variável ρ que vai guardar o valor mínimo que os elementos de M possuem, neste exemplo $\rho = 0$. A função $d(x, \theta)$ da equação 1 é responsável pela normalização.

$$d(x,\theta) = \begin{cases} \left[\frac{x}{\theta\omega - \rho}\right] & , \frac{x}{\theta\omega-\rho} \leq 255.5 \\ 255 & , \text{c.c} \end{cases} \tag{1}$$

Na função função $d(x, \theta)$ da equação 1 a variável x representa o número da matriz sendo normalizado e a variável θ representa um coeficiente para o brilho da normalização. É importante observar que quanto maior o valor θ mais "clara" a imagem resultante no final de todo o processo vai ser.

Fazendo $M_{i,j} := d(M_{i,j}, 1)$ para i de 0 a γ e j de 0 a ψ obtem-se a matriz da figura 8.

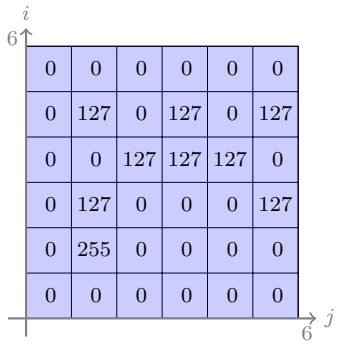

Figura 7: Estado da matriz M após seus elementos terem sido normalizados.

A matriz M pode agora ser gravada em um arquivo.

3 Formato de imagem PGM

O *PGM* é um formato de imagem, fácil implementação, sem qualquer compressão. O *PGM* pertence a uma família de formatos de imagem, todos de fácil implementação, chamado de Netpbm [4]. O Windows não possui suporte nativo ao *PGM* para poder visualizar imagens *PGM* no Windows é recomendavel o uso do programa editor de imagens *GIMP*.

O formato consiste em valores que são *strings ASCII* os valores são separados por espaços ou quebras de linhas, se uma linha começa com o caracter # esta linha vai ser desconsiderada.

O primeiro valor de um arquivo *PGM* é a *string P2*. Os dois próximos valores são a largura e a altura da imagem, estas *strings* devem ser escritas com os caracteres numéricos da tabela *ASCII*. Depois deve-se colocar o valor máximo que os *pixels* da imagem devem ter, esta *string* deve ser escrita com os caracteres numéricos da tabela *ASCII*. As próximas *strings*, que são escritas com os caracteres numéricos da tabela *ASCII*, são os valores dos *pixels* da imagem. O primeiro *pixel* do *PGM* é o *pixel* superior esquerdo e o ultimo *pixel* do *PGM* é o *pixel* inferior direito. Por fim o ultimo *byte* de um arquivo *PGM* deve ser uma quebra de linha.

3.1 Exemplo de imagem PGM

O código 1 é um arquivo *.pgm* da imagem 3.1.

Código 1: Arquivo texto de uma imagem no formato *PGM*

```
P2
#Comentario para imagem PGM
12 9
255
255 255 255 255 255 0   0   255 255 255 255 255
0   0   0   0   0   0   0   0   0   0   0   0
0   63  63  63  63  0   0   127 127 127 127 0
0   63  0   0   0   0   0   127 0   0   0   0
0   63  63  63  0   0   0   127 127 127 0   0
0   63  0   0   0   0   0   127 0   0   0   0
0   63  0   0   0   0   0   127 127 127 127 0
0   0   0   0   0   0   0   0   0   0   0   0
255 255 255 255 255 0   0   255 255 255 255 255
```

Figura 8: Imagem resultante do código 1 aumentada 26 vezes e meia.

4 Objetivo do exercício

Você deverá paralelizar o algoritmo que gera o fractal de Buddhabrot, analisando o desempenho obtido. Para tal, você deverá gerar um relatório onde explica como realizou a paralelização e apresenta resultados de testes de desempenho obtidos com a paralelização, comparando o desempenho das versões sequencial e paralela. Não esqueça que é necessário descrever qual a máquina utilizada nos experimentos e executar o experimentos múltiplas vezes, extraindo sua média e desvio padrão.

a) Parte I (Entrega até 23/03): Paralelize o código utilizando Pthreads.

b) Parte II (Entrega até 17/04): Paralelize utilizando MPI e OpenMP ao mesmo tempo, isto é, cada processo MPI poderá ter múltiplas threads. Aqui você deverá executar os experimentos usando pelo menos 2 processos e 2 threads por processo.

Referências

[1] Buda. `https://pt.wikipedia.org/wiki/Buda`. Accessed: 2015-10-30.

[2] Buddhabrot. `https://pt.wikipedia.org/wiki/Buddhabrot`. Accessed: 2015-10-30.

[3] Conjunto de mandelbrot. https://pt.wikipedia.org/wiki/Conjunto_de_Mandelbrot. Accessed: 2015-10-30.

[4] Netpbm format. https://en.wikipedia.org/wiki/Netpbm_format. Accessed: 2015-10-30.

[5] Sistemas de funções iterativas. https://pt.wikipedia.org/wiki/Sistemas_de_fun%C3%A7%C3%B5es_iterativas. Accessed: 2015-10-30.