



IME-USP

EP2 - Relatório

MAC0210 - Laboratório de Métodos Numéricos

Autor: Vítor Carvalho de Melo

nUSP: 10723753

São Paulo - 31/05/2023



Enunciado do exercício-programa: [link](#)

Repositório GitHub com todos os scripts e imagens: [link](#)

Introdução

Este documento é um relatório cujo propósito é esclarecer as decisões de projeto, as dificuldades, os resultados de experimentos e o desenvolvimento do exercício-programa 2 da disciplina Laboratório de Métodos Numéricos. Inicia-se discutindo o desenvolvimento dos *function files* **compress.m**, **decompress.m** e **calculcateError.m**, responsáveis pelos cálculos e pela criação das imagens usadas nos experimentos. Em seguida, discute-se os dois experimentos requeridos, intitulados “Zoológico” e “Selva”. Neste relatório, algumas informações já esclarecidas no enunciado do projeto, como certas deduções matemáticas ou chamadas de funções (e seus argumentos), foram suprimidas em prol da brevidade.

Function Files

1) compress.m

a) Objetivo

Esta é claramente a *function file* mais simples das três implementadas neste projeto. Trata-se de um *script* que recebe uma imagem e um argumento k , para em seguida remover k linhas e k colunas de pixels uniformemente.

b) Implementação

Segundo as instruções do enunciado, para comprimir uma imagem (quadrada de dimensão $n \times n$), devemos manter as linhas e colunas de índice i tais que $i \equiv 0 \pmod{k+1}$. Essa relação de congruência [pode ser utilizada](#) ao iterar $p = n + (n-1)k$ (a nova resolução) para manter as linhas e colunas de índice w tais que $w = p(k+1)$.

2) calculateError.m

a) Objetivo

Esta *function file* tem o objetivo de comparar as imagens original e descomprimidas. Isso é realizado tomando-se a média simples entre os erros das três cores, vermelho, azul e verde.

b) Implementação

O erro de uma cor é calculado, segundo instruções do enunciado, pela norma l^2 da diferença entre o valor dessa cor na imagem original e o valor dessa cor na imagem descomprimida, dividida pela norma l^2 do valor da cor na imagem original. Naturalmente, essa fórmula é calculada para todos os pixels das imagens:

$$errR = \frac{\|origR - decR\|_2}{\|origR\|_2}$$



Como vamos ver no próximo item, a descompressão de uma imagem comprimida, usando o mesmo valor de k , resulta em uma imagem com k linhas e colunas de pixels a menos, por isso no *script* decompress.m, as k primeiras linhas e colunas da imagem original são descartadas. Nos experimentos, notou-se que dessa maneira, adquiri-se um erro menor do que se a última linha e coluna forem descartadas, sendo assim possível inferir que é esta linha que é descartada no processo de descompressão.

3) decompress.m

a) Objetivo

Esta *function file* tem o papel de expandir a resolução de uma imagem ao se utilizar de funções interpoladoras para calcular o valor dos novos pixels que serão gerados. No contexto desta atividade, a usamos exclusivamente para expandir imagens previamente comprimidas, portanto chamamos a função de *decompress*.

De forma comum aos dois métodos implementados, inicia-se uma matriz de tamanho $n \times n$, com $n = p + (p - 1)k$, com p = número de linhas ou colunas da imagem comprimida e k = número inteiro > 1 fornecido pelo usuário, equivale à quantas vezes vamos multiplicar p (ex. se $p = 350$ e $k = 1$, n será igual à 699).

b) Implementação da função *bilinear*

Caso seja selecionada o método bilinear para interpolação, utilizamos a matriz H definida pelo enunciado como:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & h & 0 \\ 1 & h & 0 & 0 \\ 1 & h & h & h^2 \end{bmatrix}$$

Para cada cor da nova imagem descomprimida, definimos um vetor F como:

$$F = \begin{bmatrix} f(x_i, y_j) \\ f(x_i, y_{i+1}) \\ f(x_{i+1}, y_j) \\ f(x_{i+1}, y_{j+1}) \end{bmatrix}$$

em que a função f possui como domínio os pixels já conhecidos da imagem comprimida. Para execuções com $k > 1$, haverão mais de um pixel a ser interpolado para cada quadrado Q_{ij} .

Dessa forma, podemos calcular a matriz A , que representa as constantes a serem utilizadas no polinômio interpolador a partir da equação:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(x_i, y_j) \\ f(x_i, y_{i+1}) \\ f(x_{i+1}, y_j) \\ f(x_{i+1}, y_{j+1}) \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & h & 0 \\ 1 & h & 0 & 0 \\ 1 & h & h & h^2 \end{bmatrix}^{-1}$$



Em seguida, calcula-se os pontos interpolados a partir da equação:

$$p_{ij}(x, y) = a_0 + a_1(x - x_i) + a_2(y - y_j) + a_3(x - x_i)(y - y_j)$$

O script foi programado de forma a calcular estes pontos de forma individual para cada cor (vermelho, verde e azul) para facilitar o entendimento, mas seria possível calcular as três cores utilizando-se de uma matriz A de dimensões 3 e 4.

c) Implementação da função *bicubic*

O cálculo da matriz de constantes A para o método bicúbico é substancialmente mais complexa, sendo necessário o cálculo de derivadas parciais e mistas. O objetivo desse cálculo é (ao se assumir que a função geradora da imagem seja C¹) obter um polinômio interpolador mais *smooth*, e assim obter uma imagem menos quadriculada. O polinômio é definido como:

$$p_{ij}(x, y) = [1 \ (x - x_i) \ (x - x_i)^2 \ (x - x_i)^3] * A * \begin{bmatrix} 1 \\ (y - y_j) \\ (y - y_j)^2 \\ (y - y_j)^3 \end{bmatrix}$$

E a matriz A pode ser calculada da seguinte maneira:

$$A = B^{-1} * F * B^{-1T}$$

Sendo,

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & h & h^2 & h^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2h & 3h^2 \end{bmatrix}$$

e,

$$F = \begin{bmatrix} f(x_i, y_j) & f(x_i, y_{j+1}) & \frac{\partial f}{\partial y}(x_i, y_j) & \frac{\partial f}{\partial y}(x_i, y_{j+1}) \\ f(x_{i+1}, y_j) & f(x_{i+1}, y_{j+1}) & \frac{\partial f}{\partial y}(x_{i+1}, y_j) & \frac{\partial f}{\partial y}(x_{i+1}, y_{j+1}) \\ \frac{\partial f}{\partial x}(x_i, y_j) & \frac{\partial f}{\partial x}(x_i, y_{j+1}) & \frac{\partial f}{\partial x \partial y}(x_i, y_j) & \frac{\partial f}{\partial x \partial y}(x_i, y_{j+1}) \\ \frac{\partial f}{\partial x}(x_{i+1}, y_j) & \frac{\partial f}{\partial x}(x_{i+1}, y_{j+1}) & \frac{\partial f}{\partial x \partial y}(x_{i+1}, y_j) & \frac{\partial f}{\partial x \partial y}(x_{i+1}, y_{j+1}) \end{bmatrix}$$

Para calcular as derivadas mistas e parciais necessárias para montar a matriz F, foram escritas as funções *partialDeriv*, que calcula a primeira derivada no ponto (x, y) em relação a x ou a y, e *mixedDeriv*, que utiliza a função *partialDeriv* para calcular a derivada mista no ponto (x, y). Em ambas as funções, toma-se o cuidado de não calcular as derivadas de forma centrada nos pontos extremos (bordas) da imagem.



Zoológico

Nesta seção da atividade, construí imagens utilizando funções matemáticas simples com o objetivo de testar as *function files* escritas em um ambiente controlado e responder às seguintes questões, como descritas no enunciado:

- Funciona bem para imagens preto e branco?
- Funciona bem para imagens coloridas?
- Funciona bem para todas as funções de classe C2?
- E para funções que não são da classe C2?
- Como o valor de h muda a interpolação?
- Como se comporta o erro?

Estas perguntas serão respondidas continuamente ao analisarmos os resultados individuais do experimentos. No final, entretanto, responderemos às questões de forma geral.

1) Função 1 - do enunciado

$$f_1(x, y) = (\text{sen}(x), \frac{\text{sen}(y) + \text{sen}(x)}{2}, \text{sen}(x))$$

A função acima foi utilizada para gerar a seguinte imagem (código disponível no [repositório](#) do GitHub), de resolução 600x600:

(continua abaixo)

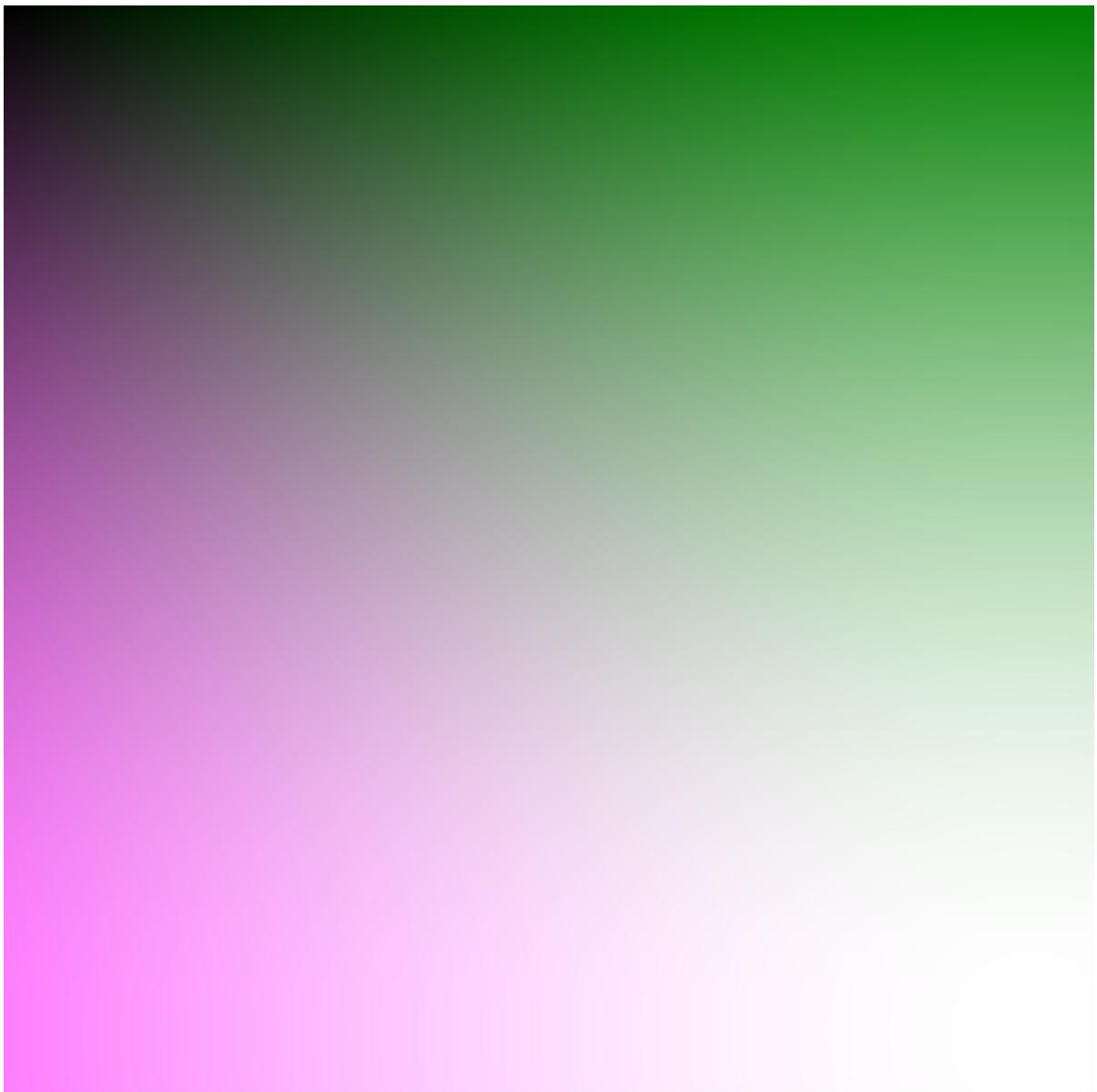


Figura 1. imagem gerada pela função f_1

Vamos agora comprimir a imagem com o parâmetro $k = 2$ e, em seguida, comparar os dois algoritmos de descompressão.



Figura 2. Figura 1 comprimida com $k = 2$



Utilizaremos agora, em ambos os algoritmos de descompressão, os parâmetros $k = 2$ e $h = 1$.

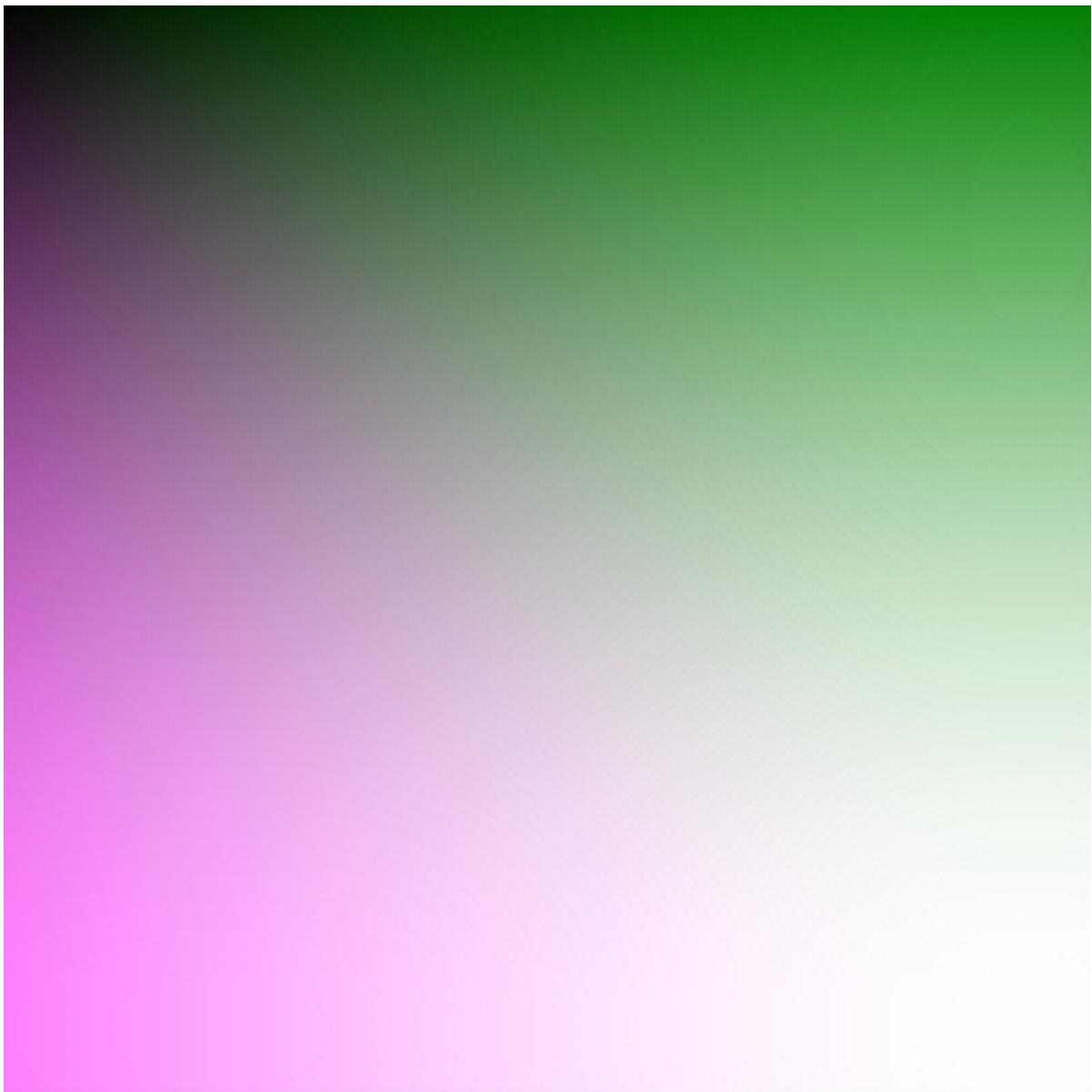


Figura 3. Figura 2 descomprimida pelo método BILINEAR com parâmetros $k = 2$ e $h = 1$

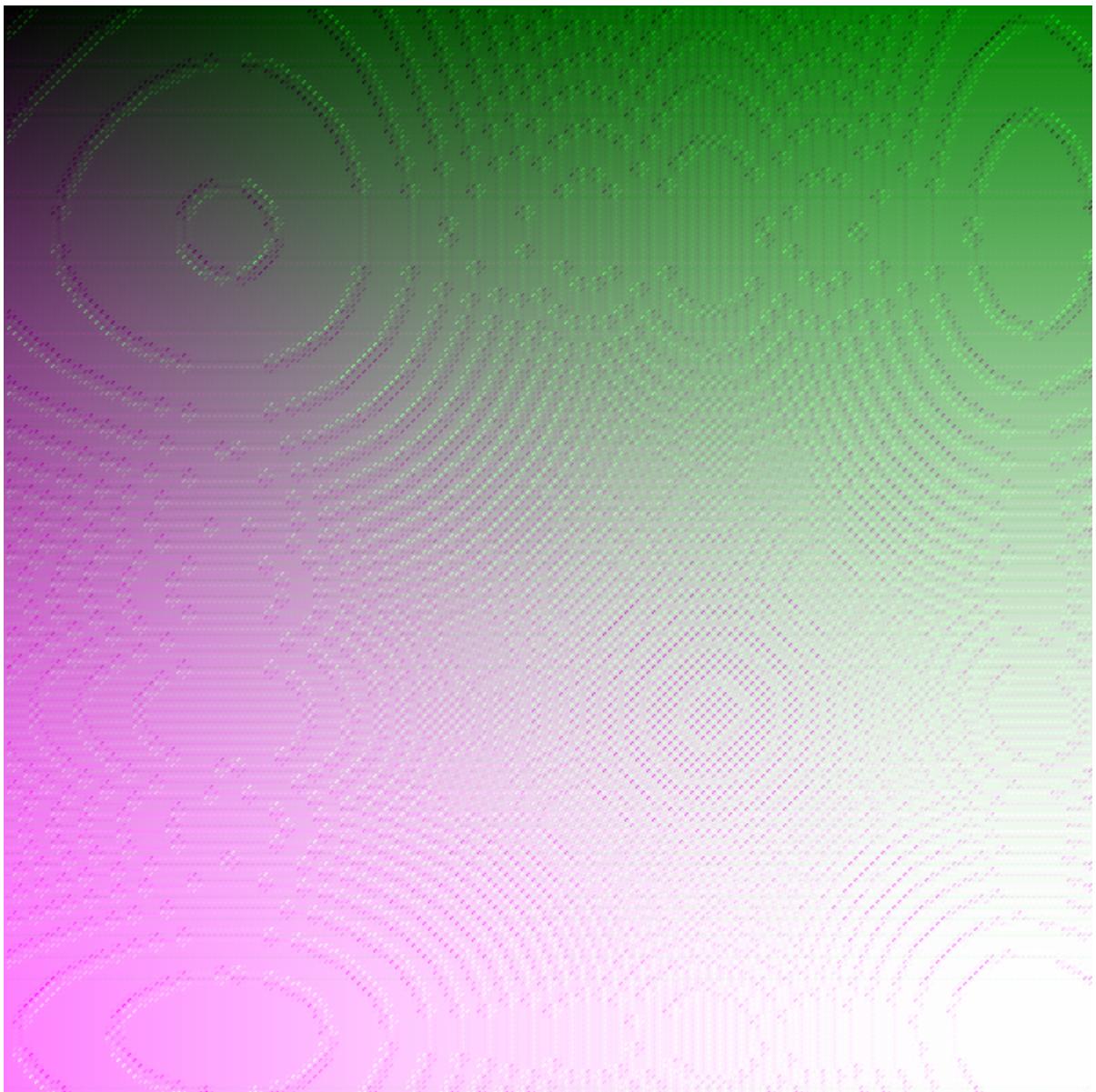


Figura 4. Figura 2 descomprimida pelo método BICÚBICO com parâmetros $k = 2$ e $h = 1$

Para estes parâmetros, está claro que ambos os métodos sofrem de problemas para descomprimir a imagem. O método bilinear tende a quadricular um pouco as “linhas” divisórias que separam as cores, enquanto o método bicúbico produz padrões e insere cores diferentes das esperadas nas regiões das imagens.

O erro do método **bilinear** nessas condições foi de: 0.0096744
O erro do método **bicúbico** nessas condições foi de: 0.022997

Vamos tentar, agora, repetir o experimento com os parâmetros $k = 2$ e $h = 2$.

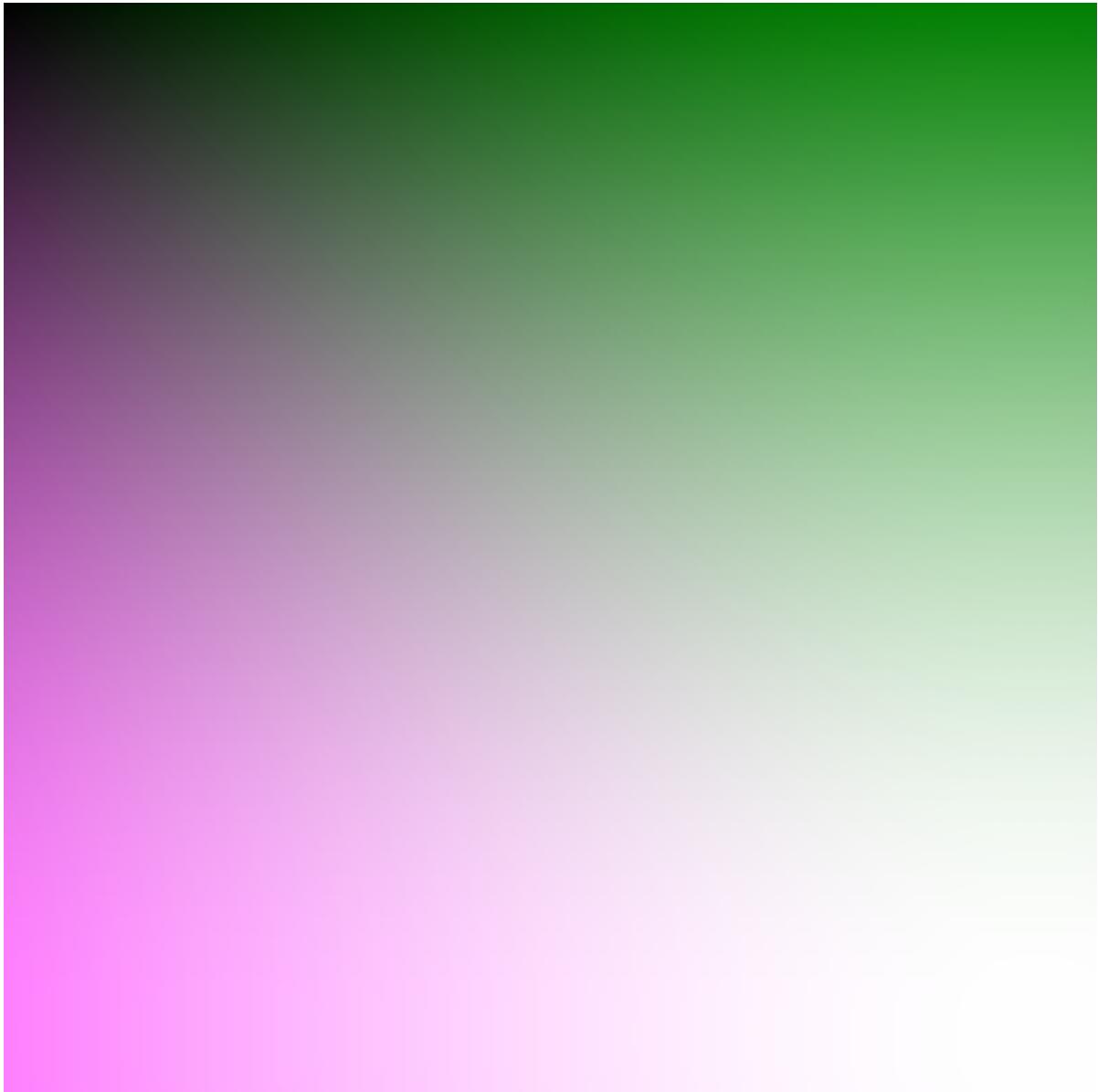


Figura 5. Figura 2 descomprimida pelo método BILINEAR com parâmetros $k = 2$ e $h = 2$

(continua abaixo)

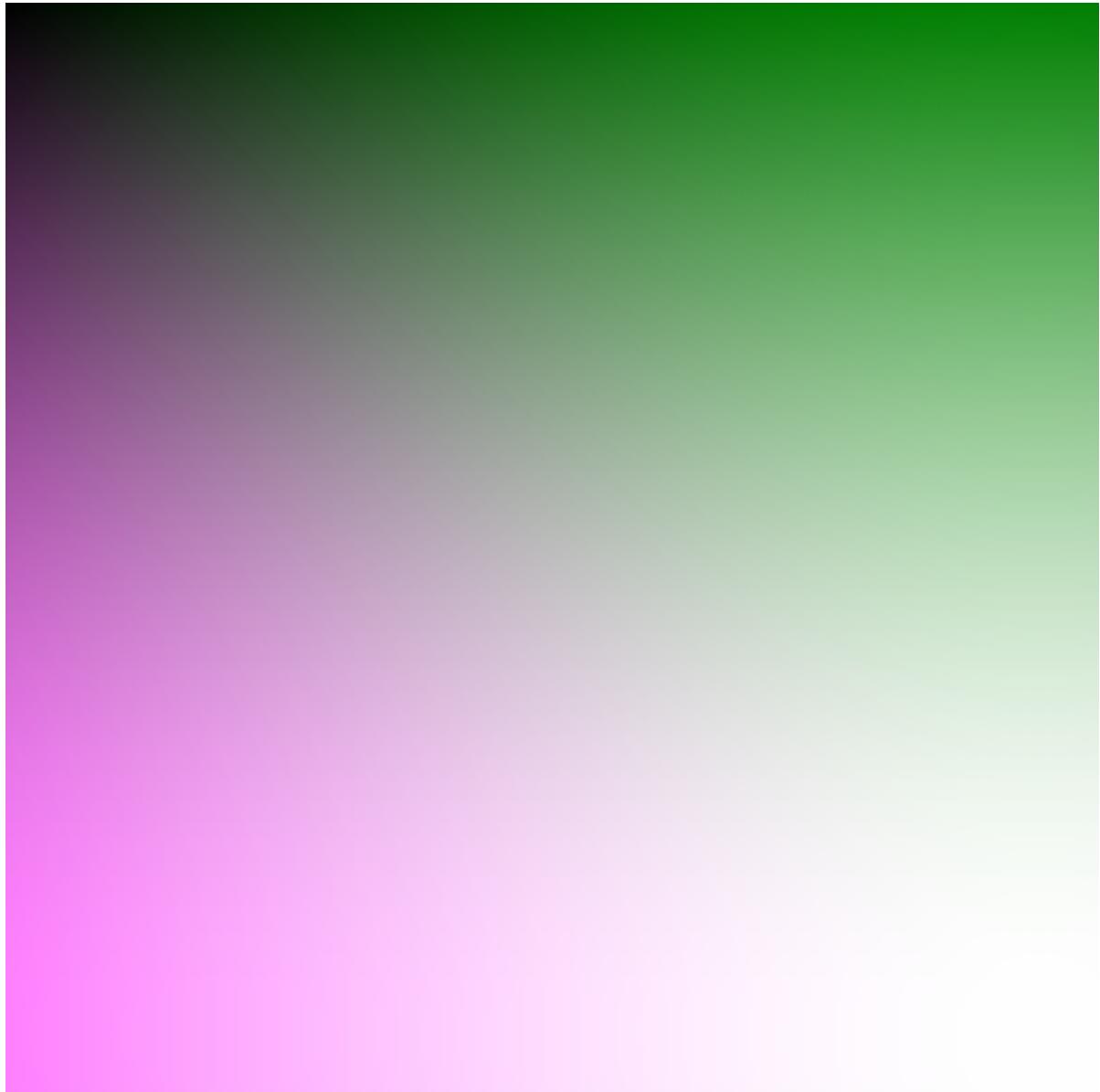


Figura 6. Figura 2 descomprimida pelo método BICÚBICO com parâmetros $k = 2$ e $h = 2$

Para estes parâmetros, ainda é possível notar, de forma muito mais sutil, padrões quadriculados quando usa-se o método bilinear. Para o método bicúbico, entretanto, as imagens originais e descomprimidas são quase indiferenciáveis.

O erro do método **bilinear** nessas condições foi de: 0.0034948

O erro do método **bicúbico** nessas condições foi de: 0.0033211

Curiosamente, o método bilinear teve um erro menor quando experimentamos um $h=1$, e o método bicúbico teve sucesso quando $h=2$.



2) Função 2 - preta e branca \mathbf{C}^2

$$f_2(x, y) = \left(\frac{\sin(x) + \cos(y)}{2}, \frac{\sin(x) + \cos(y)}{2}, \frac{\sin(x) + \cos(y)}{2} \right)$$

A imagem gerada por esta função, de dimensões 720x720, é a seguinte:

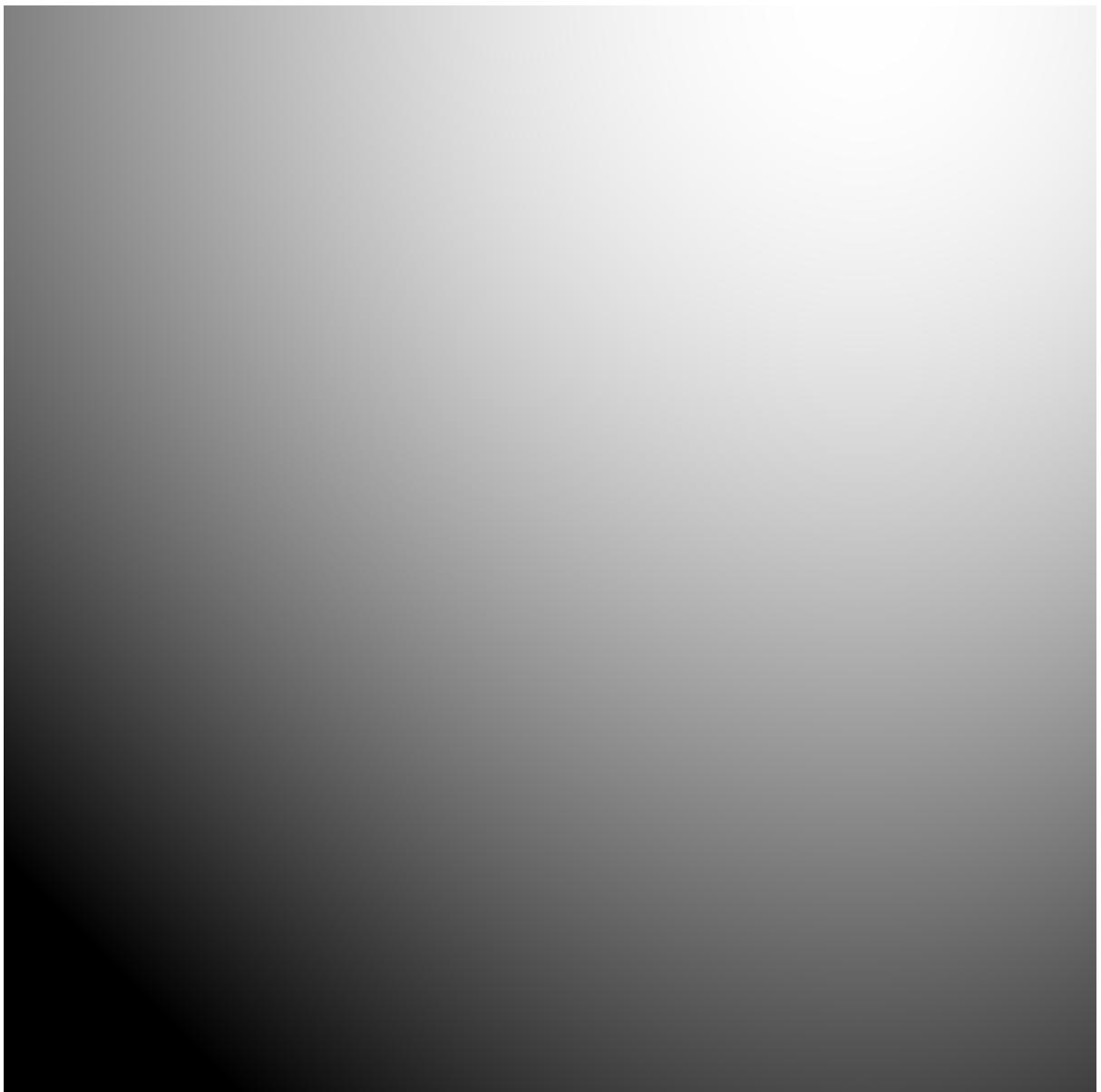


Figura 6. imagem gerada pela função f_2

Desta vez, vamos utilizar os parâmetro $k=3$ para comprimir a imagem.

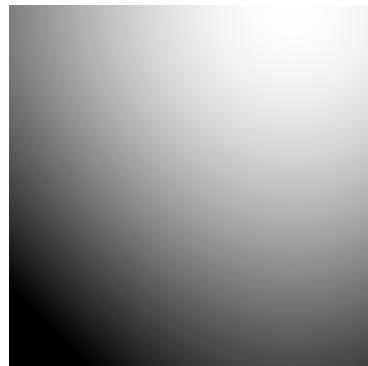


Figura 7. Figura 6 comprimida com $k = 3$

Utilizaremos agora, em ambos os algoritmos de descompressão, os parâmetros $k = 2$ e $h = 2$.

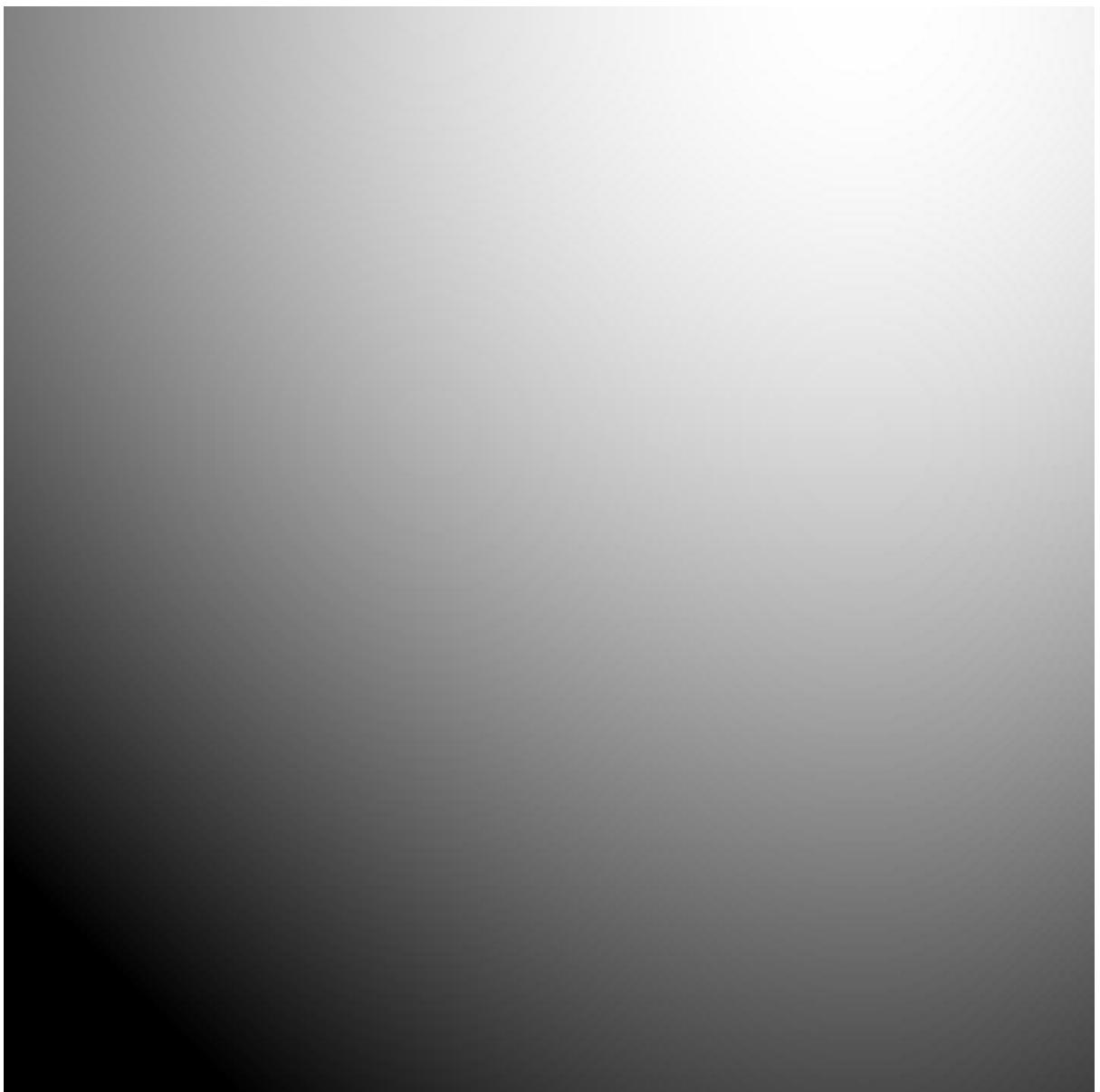


Figura 8. Figura 7 descomprimida pelo método BILINEAR com parâmetros $k = 3$ e $h = 2$

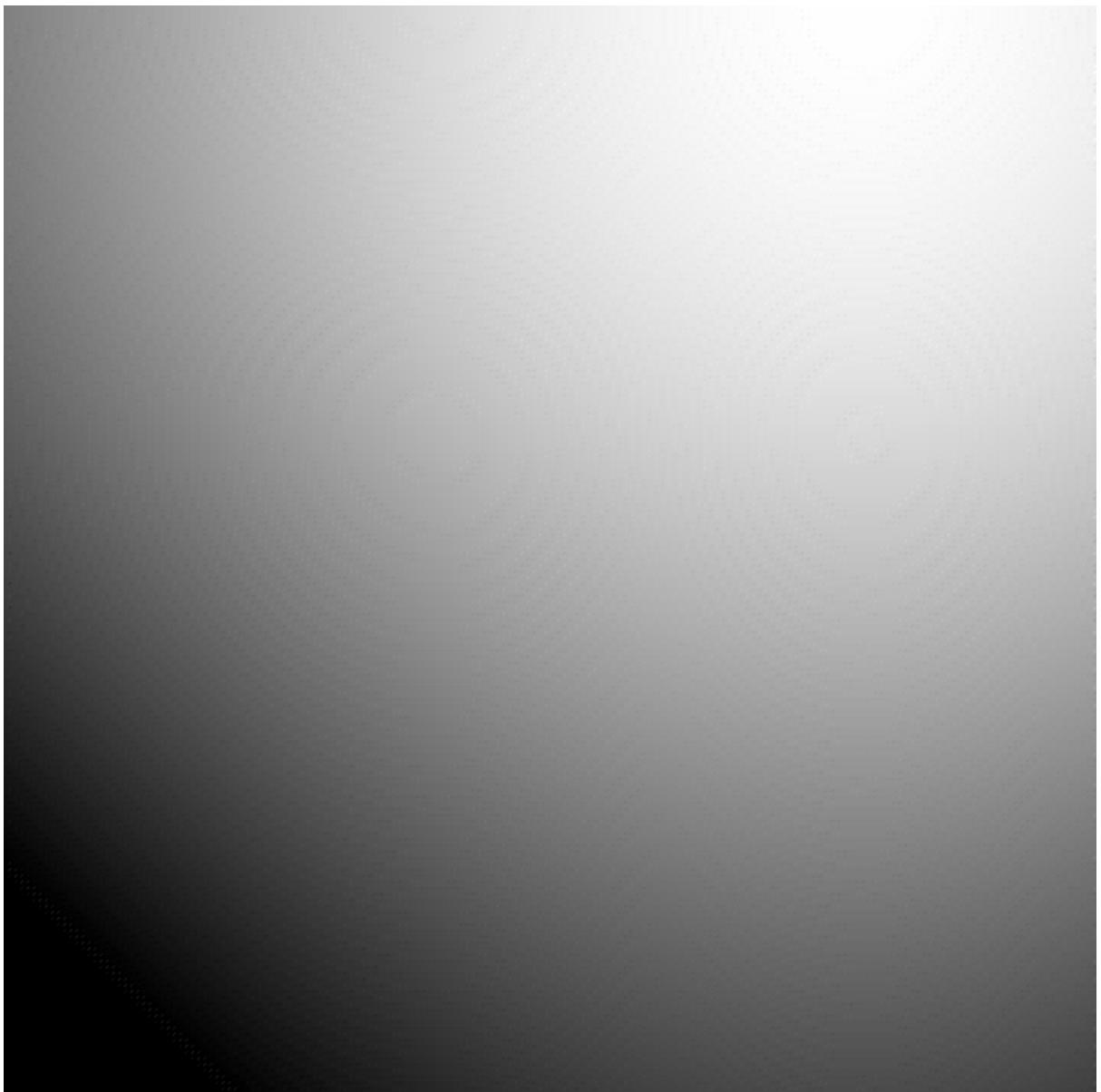


Figura 9. Figura 7 descomprimida pelo método BICÚBICO com parâmetros $k = 3$ e $h = 2$

Assim como na função 1, a descompressão utilizando o método bilinear, mesmo com o parâmetro $h = 2$, gera uma imagem com “linhas” quadriculadas. Caso não seja possível verificar isso neste relatório, pode acessar a Figura 8 neste [link](#).

O erro do método **bilinear** nessas condições foi de: 0.0026895
O erro do método **bicúbico** nessas condições foi de: 0.0037921

Vamos tentar, agora, repetir o experimento com os parâmetros $k = 3$ e $h = 3$.

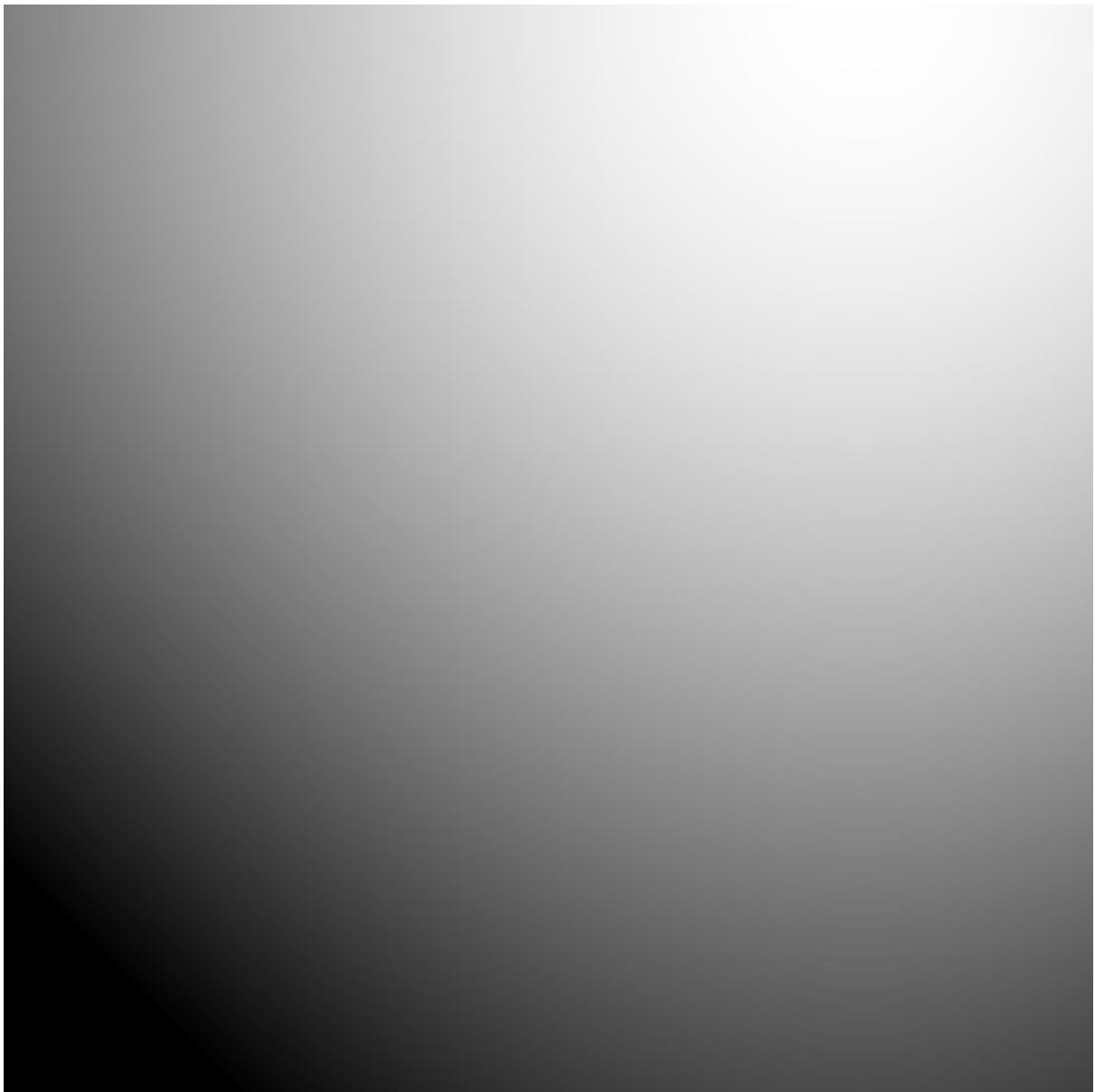


Figura 10. Figura 7 descomprimida pelo método BILINEAR com parâmetros $k = 3$ e $h = 3$

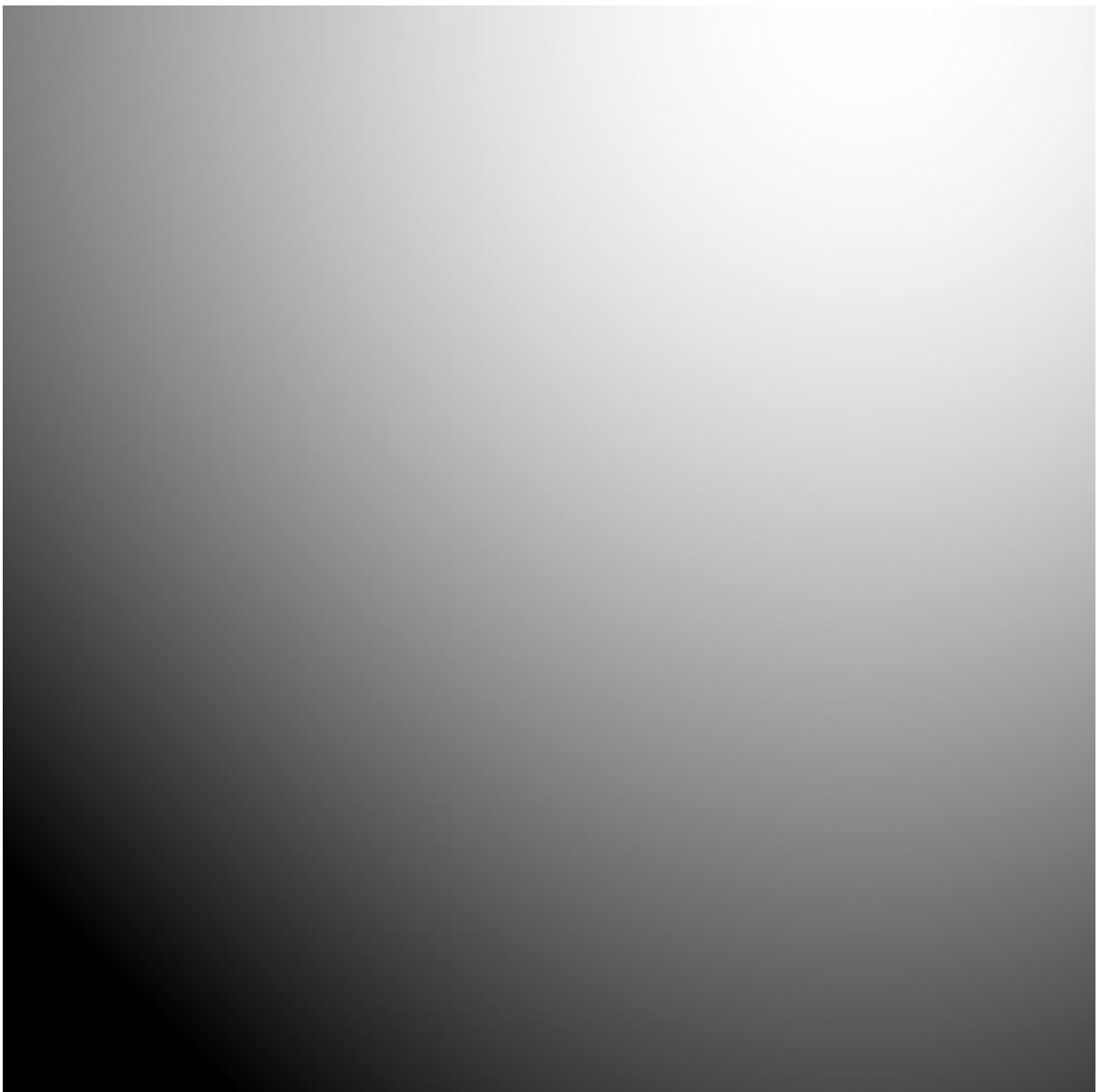


Figura 11. Figura 7 descomprimida pelo método BICÚBICO com parâmetros $k = 3$ e $h = 3$

Mesmo para valores de h mais altos, o padrão se repete: inicialmente o método bilinear gera imagens quadriculadas porém com muito menos erros. À medida que h aumenta, o método bicúbico passa a gerar imagens mais nítidas e suaves. O mesmo acontece com o método bilinear, porém as imagens continuam um pouco quadriculadas.

O erro do método **bilinear** nessas condições foi de: 0.0011051

O erro do método **bicúbico** nessas condições foi de: 0.0010097

3) Função 3 - não da classe C^2

No intervalo $1 \leq x \leq 300$, $f_3 = (2x + \frac{y}{2}, x + y, \frac{x}{2} + 2y)$

No intervalo $301 \leq x \leq 500$, $f_3 = (\frac{x}{2}, (600 - x) + (600 - y), 2x)$

Nesta próxima etapa de testes, é estabelecida uma função para o intervalo de pixels do eixo x [1,300] e outra para o intervalo [301, 600], garantindo a descontinuidade. O objetivo é testar o comportamento das *function files* para este tipo de função. A imagem gerada pela função, de dimensões 600x600, é a seguinte:

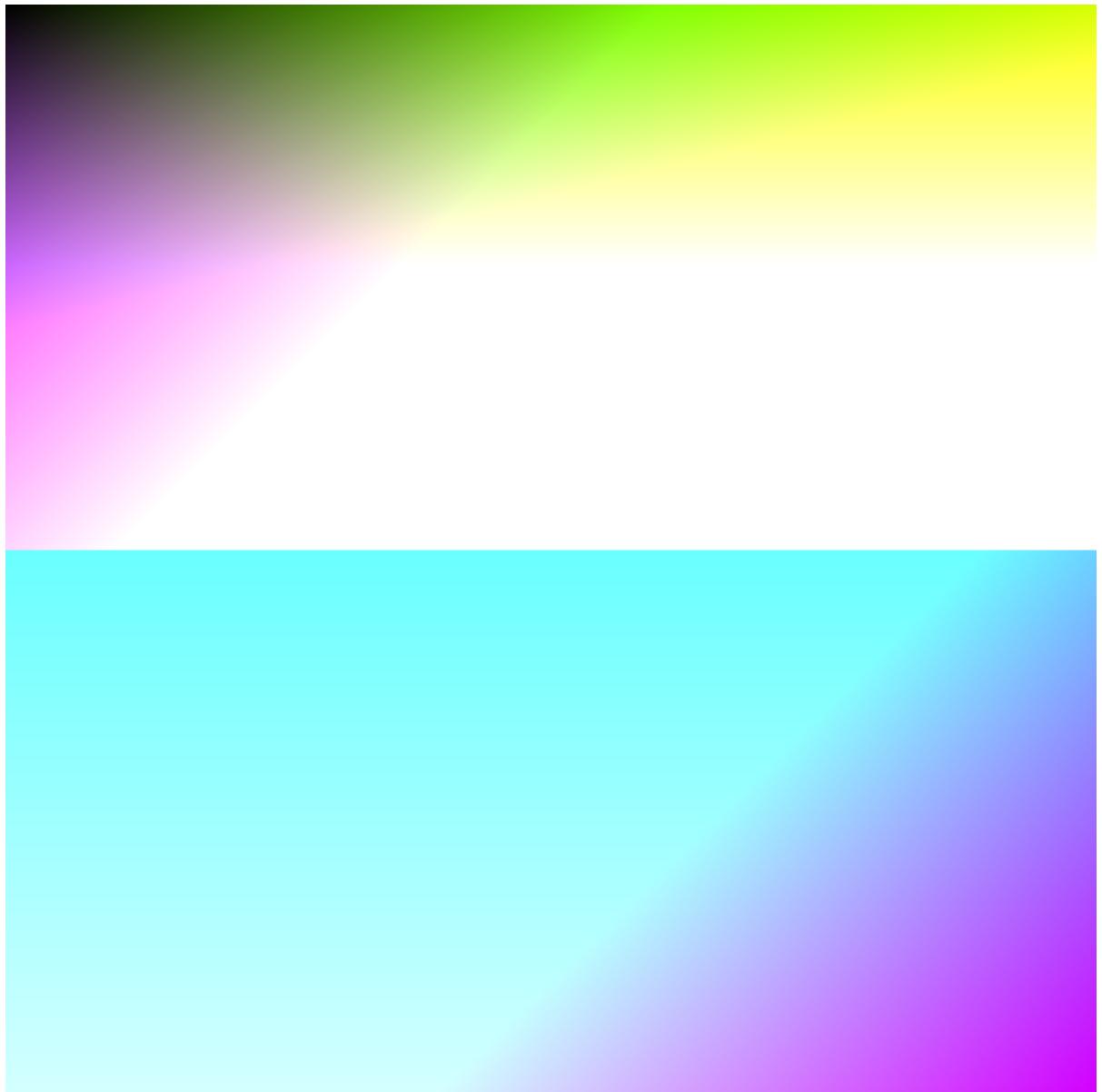


Figura 12. imagem gerada pela função f_3

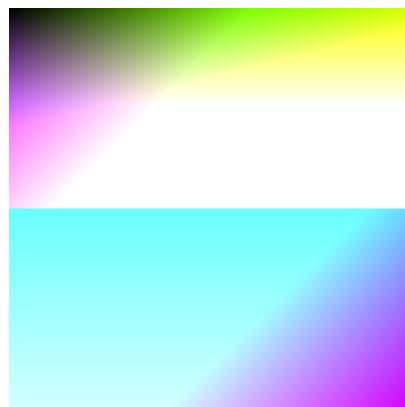


Figura 13. Figura 12 comprimida com $k = 2$

Utilizaremos agora, em ambos os algoritmos de descompressão, os parâmetros $k = 2$ e $h = 1$.

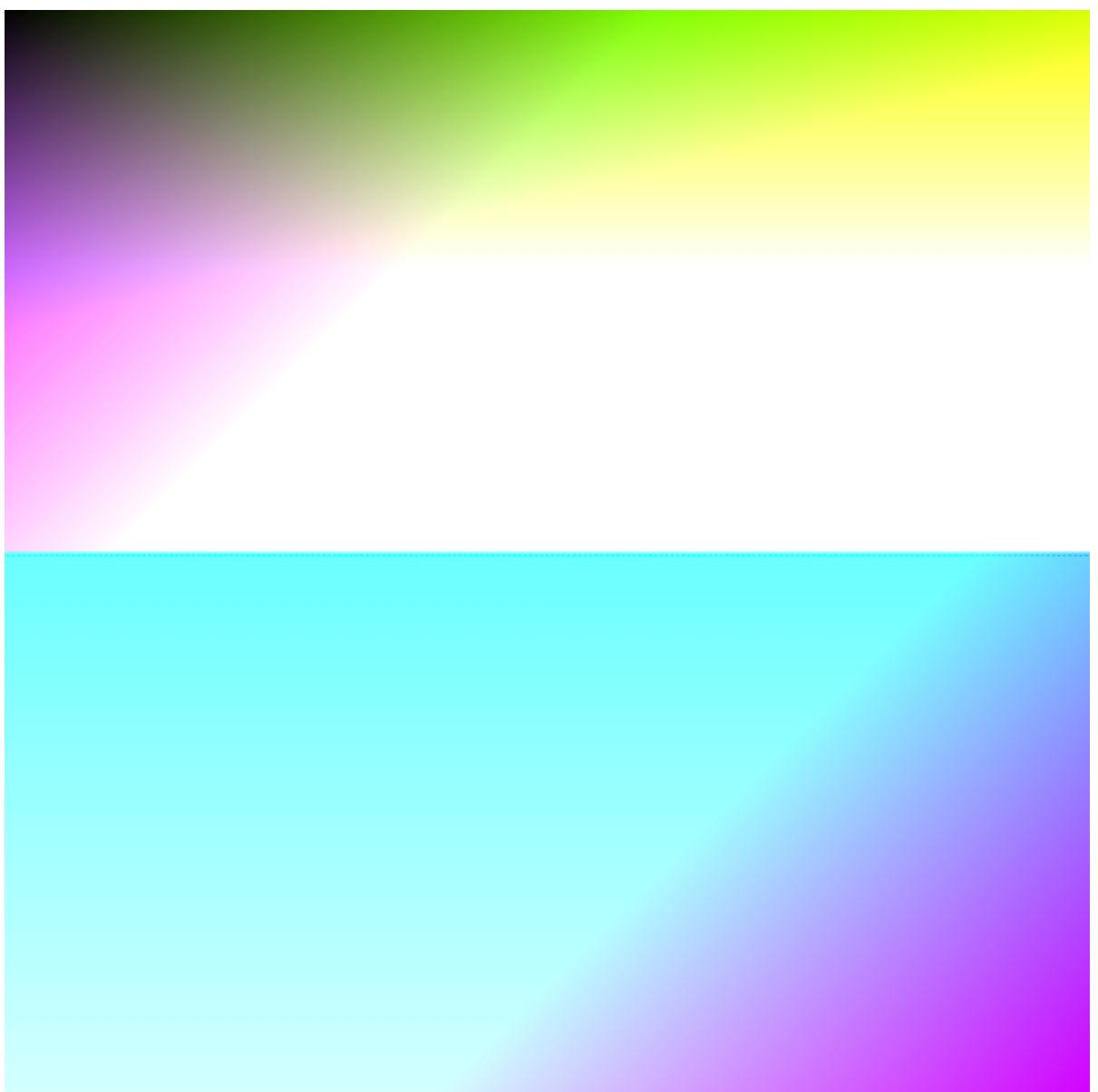


Figura 14. Figura 13 descomprimida pelo método BILINEAR com parâmetros $k = 2$ e $h = 1$

Fica claro, por essa imagem, que na região de descontinuidade, a descompressão bilinear com esses parâmetros gera valores incorretos. Isso se dá pelo fato de que, neste caso, a interpolação para os pontos nas regiões de inflexão deveriam apenas levar em consideração alguns pontos vizinhos, mas pelo fato de que considera todos os pontos vizinhos, os valores interpolados são bastante incorretos.

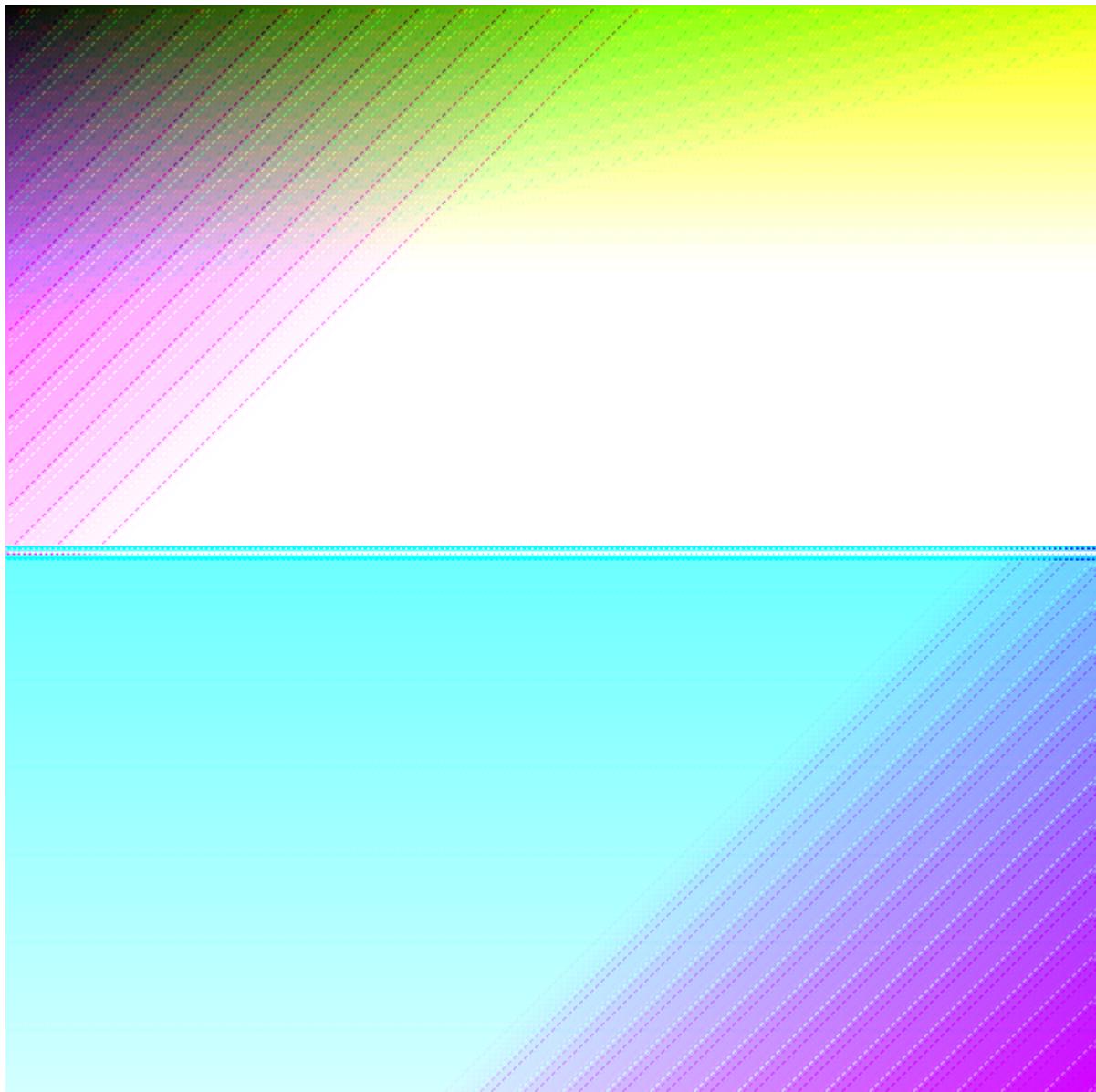


Figura 15. Figura 13 descomprimida pelo método BICÚBICO com parâmetros $k = 2$ e $h = 1$

Esta diferença se mostra ainda mais nítida pelo método bicúbico. Todos os pontos próximos à região de descontinuidade possuem um erro altíssimo.

O erro do método **bilinear** nessas condições foi de: 0.028535
O erro do método **bicúbico** nessas condições foi de: 0.037610

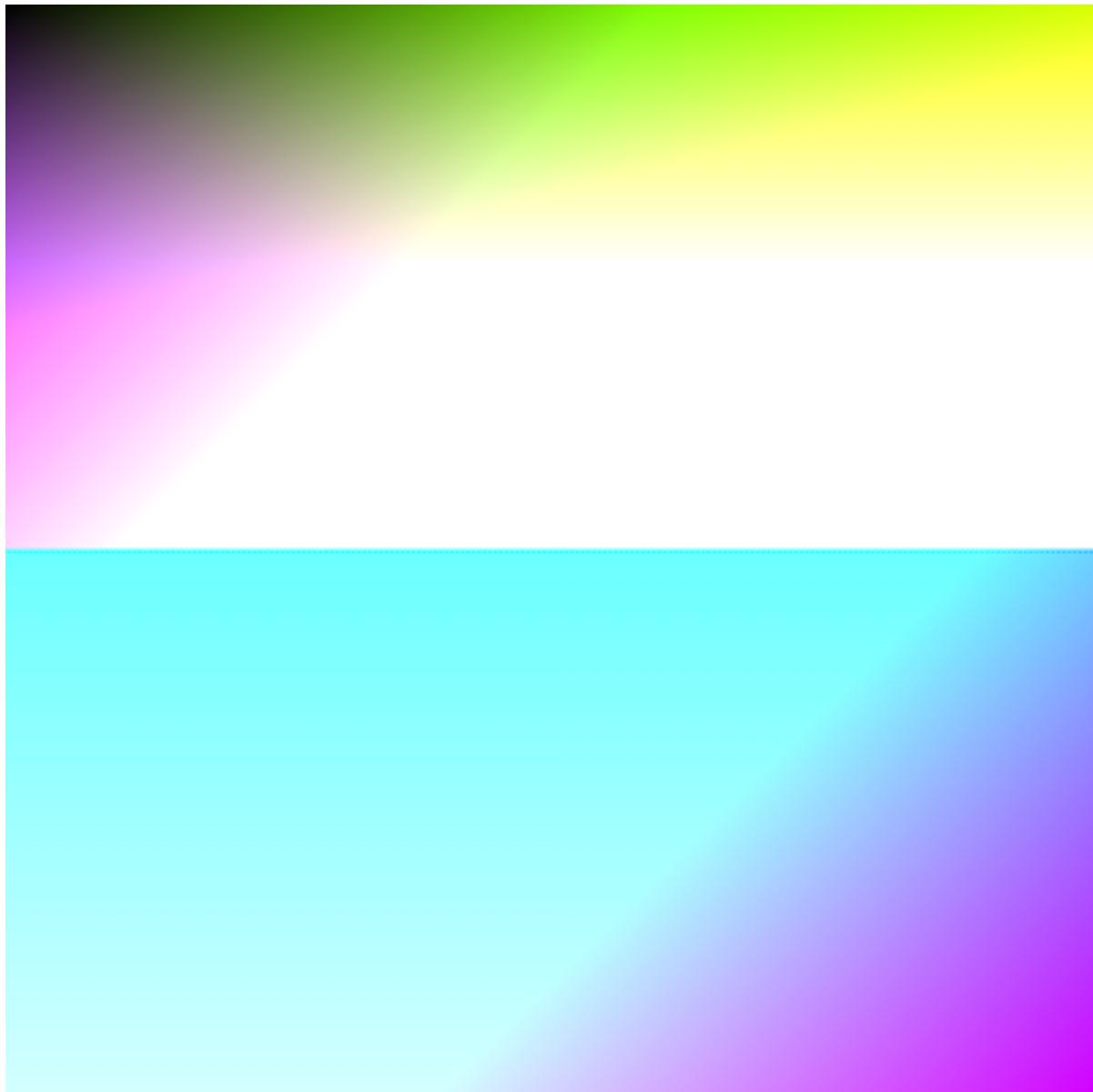


Figura 16. Figura 13 descomprimida pelo método BILINEAR com parâmetros $k = 2$ e $h = 2$

Embora os demais pontos sejam melhor interpolados ao aumentar o valor de h , a região das bordas continua sendo interpolada incorretamente.

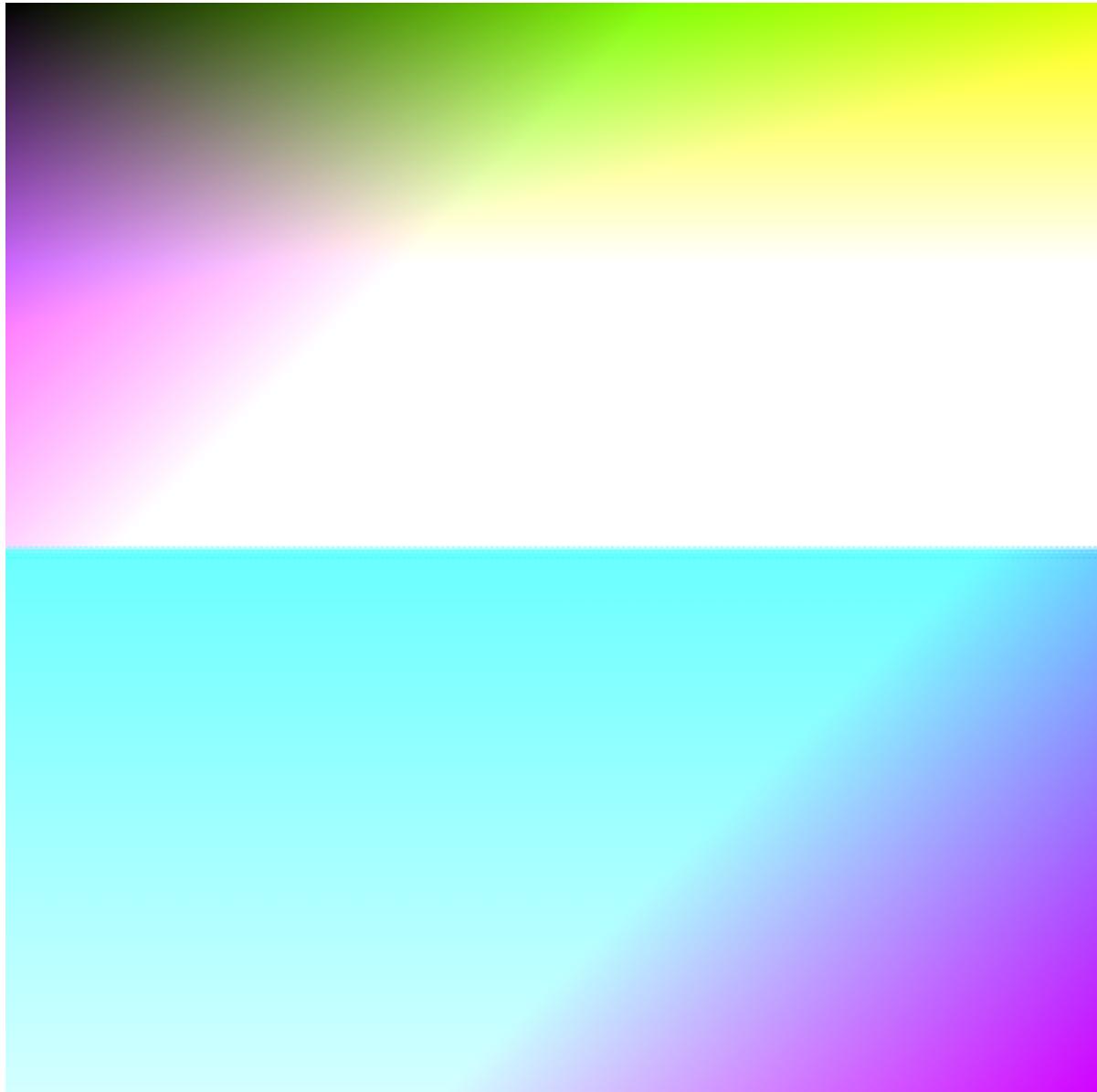


Figura 16. Figura 13 descomprimida pelo método BILINEAR com parâmetros $k = 2$ e $h = 2$

Ao aumentar o valor de h , o método bicúbico se comporta consideravelmente melhor, embora a região de descontinuidade continue visivelmente incorreta.

O erro do método **bilinear** nessas condições foi de: 0.014529

O erro do método **bicúbico** nessas condições foi de: 0.014529



Respondendo às perguntas do enunciado:

- **Q: Funciona bem para imagens preto e branco?**
A: As mesmas características encontradas com imagens coloridas tendem a se repetir para imagens em preto e branco. Curiosamente, entretanto, o método bicúbico produz um interessante erro ao produzir certos padrões circulares na imagem. Esses erros diminuem à medida que o parâmetro h é incrementado.
- **Q: Funciona bem para imagens coloridas?**
A: Sim, o principal problema do método bilinear é produzir linhas ou curvas quadriculadas, e o principal problema do método bicúbico é produzir padrões (geralmente circulares) onde eles não existem inicialmente.
- **Q: Funciona bem para todas as funções de classe C2?**
A: Sim, levando-se sempre em consideração os problemas supracitados.
- **Q: E para funções que não são da classe C2?**
A: Para funções descontínuas, problemas maiores ocorrem nas regiões onde estão presentes descontinuidades. As especificidades destes problemas foram discutidas em seções anteriores.
- **Q: Como o valor de h muda a interpolação?**
A: O valor de h , invariavelmente, melhora a qualidade das descompressões à medida que é incrementado, além de aumentar consideravelmente o tempo de execução dos scripts.
- **Q: Como se comporta o erro?**
A: Nos experimentos realizados, percebe-se que, para valores pequenos de h , o método bilinear apresenta erros menores que o método bicúbico, mas, à medida que h é incrementado, este segundo método torna-se mais eficaz que o primeiro.

Selva

Nesta parte do trabalho, vamos testar nosso método de descompressão com uma imagem real, ou seja, não definida por função. A imagem escolhida é uma renderização do personagem [G-Man](#), um antagonista da série de video-games [Half-Life](#). A imagem original possui resolução 700x700.



Figura 17. G-Man

Inicialmente, vamos comprimi-la com o parâmetro $k = 1$.



Figura 18. Figura 17 comprimida com $k = 1$

Vamos descomprimir essa imagem com parâmetros $h = 1$ e $k = 1$.

(continua abaixo)



Figura 19. Figura 18 descomprimida pelo método BILINEAR com parâmetros $k = 1$ e $h = 1$

(continua abaixo)



Figura 20. Figura 18 descomprimida pelo método BICÚBICO com parâmetros $k = 1$ e $h = 1$

Os problemas já notados na seção zoológico também estão presentes neste caso e novas observações podem ser feitas. O efeito de tornar regiões mais quadriculadas é evidente no método bilinear, principalmente em regiões de descontinuidade (entre a face e o fundo preto, por exemplo). Um novo fenômeno observado é que o método bicúbico, nestes parâmetros, parece gerar uma imagem com efeito de desfoque, porém a aproximação, já para um h pequeno, é visualmente melhor do que a realizada pelo método bilinear.

O erro do método **bilinear** nessas condições foi de: 0.036867

O erro do método **bicúbico** nessas condições foi de: 0.062937

Vamos repetir o experimento com $h = 2$.



Figura 21. Figura 18 descomprimida pelo método BILINEAR com parâmetros $k = 1$ e $h = 2$

(continua abaixo)



Figura 22. Figura 18 descomprimida pelo método BICÚBICO com parâmetros $k = 1$ e $h = 2$

Para um valor de h levemente maior, as qualidades das descompressões aumentou de forma impressionante. Embora problemas de quadriculação ainda aconteçam, ambos os métodos de descompressão conseguiram resultados excelentes, muito próximos à imagem original.

O erro do método **bilinear** nessas condições foi de: 0.016389
O erro do método **bicúbico** nessas condições foi de: 0.016389

Vamos repetir o experimento com $h = 3$.



Figura 23. Figura 18 descomprimida pelo método BILINEAR com parâmetros $k = 1$ e $h = 3$

(continua abaixo)



Figura 22. Figura 18 descomprimida pelo método BICÚBICO com parâmetros $k = 1$ e $h = 3$

De forma interessante porém previsível, no método bilinear, embora a qualidade aumente de forma geral, o problema de quadriculação aumenta nas regiões de descontinuidade a medida que h é incrementado.

O erro do método **bilinear** nessas condições foi de: 0.017412

O erro do método **bicúbico** nessas condições foi de: 0.018258