# EP2 - Comprimir e Conquistar

MAC0210 - Laboratrio de Mtodos Numricos

Setembro 2016

Este enunciado foi elaborado e escrito por Ernesto G. Birgin (professor), Gustavo Estrela (monitor) e pelos alunos Nathan Benedetto, Rodrigo Enju, Victor da Matta, Victor Sprengel e Victor Molero. Agradecemos ao aluno Gabriel Capella por notar que a matriz do sistema bicúbico estava errada.

## 1 Introdução

Este exercício-programa tem como objetivo generalizar dois métodos de interpolação por polinômios para o caso bivariado. O experimento consiste em começar com uma imagem grande, fazer uma amostra pequena de seus pixels e depois reconstruí-la.

Você deve comparar os resultados de interpolação de duas maneiras. A primeira parte consistirá em observar os resultados desse experimento quando a imagem interpolada é gerada por uma função (o zoológico), e a outra em testá-lo com imagens reais (a selva).

## 2 Parte 0 - O Laboratório

Primeiro, vamos considerar que todo nosso trabalho de interpolação pode ser feito independentemente para cada uma das 3 cores da paleta RGB. Uma imagem de $p^2$ pixels fornece, então, 3 matrizes de pixels, as quais estamos interessados em estudar e interpolar. Para interpolar essa imagem, vamos imaginar que os pixels conhecidos foram amostrados de uma imagem original.

Seja $(i, j)$ um pixel da imagem. Consideraremos que esse pixel é o valor de uma função $f : \mathbb{R}^2 \to \mathbb{R}$ avaliada em um ponto da forma:

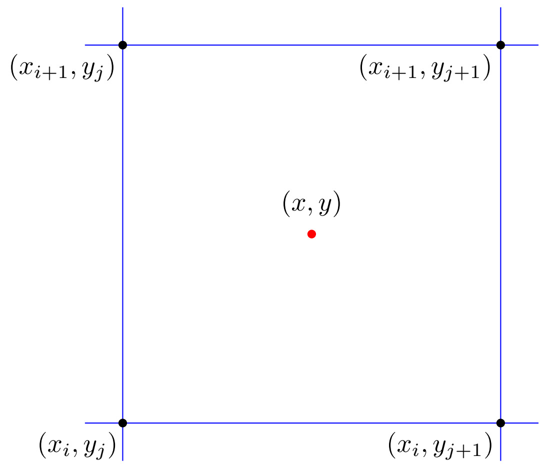
$$\begin{aligned} (x_i, y_j), &\quad \text{tal que} \\ x_i = \overline{x} + ih, &\quad \overline{x} \in \mathbb{R} \\ y_j = \overline{y} + jh, &\quad \overline{y} \in \mathbb{R} \end{aligned}$$

para algum $h \in \mathbb{R}$. Isto significa que estamos associando uma imagem quadrada de $p \times p$ pixels a um quadrado no $\mathbb{R}^2$ com canto inferior esquerdo em $(\overline{x}, \overline{y}) = (x_0, y_0)$ e canto superior direito em $(\overline{x}+(p-1)h, \overline{y}+(p-1)h) = (x_{p-1}, y_{p-1})$.

Nossa função interpolante deve ser capaz de, dado $(x, y) \in [\overline{x}, \overline{x} + (p-1)h] \times [\overline{y}, \overline{y} + (p-1)h]$, aproximar $f(x, y)$. Iremos supor que $(x, y)$ não é um dos pontos amostrados, isto é,

$$(x, y) \neq (x_i, y_j) \ \forall \ 0 \leq i < p, 0 \leq j < p.$$

Para aproximar $f(x, y)$, considere os quatro pontos amostrados que definem um quadrado de lado $h$ que contém o ponto $(x, y)$.




Note que, da maneira que definimos, existe mais de um quadrado que contém o ponto $(x, y)$ se ele estiver na fronteira de um quadrado. Você tem a liberdade de criar uma regra para definir qual quadrado será usado para interpolar o ponto nessas condições. Para este enunciado, chamaremos o quadrado com canto inferior esquerdo em $(x_i, y_j)$ de quadrado $Q_{ij}$.

Nos dois métodos de interpolação que implementaremos nesse exercício programa, o valor interpolado em $(x, y)$ dependerá apenas de informações dos vértices do quadrado de lado $h$ como definimos. Isto é, para cada quadrado $Q_{ij}$ vamos definir um polinômio $p_{ij}$ que interpola pontos dentro deste quadrado.

## 2.1 Interpolação Bilinear Por Partes

Para interpolar $f(x, y)$ para $(x, y)$ do quadrado $Q_{ij}$, o método bilinear define $p_{ij}(x, y)$ tal que:

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

O polinômio $p_{ij}$ deve satisfazer as restrições de igualdade que nos levam a determinar os coeficientes de $p_{ij}$. Veja:

- $f(x_i, y_j) = p_{ij}(x_i, y_j)$

  $$f(x_i, y_j) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- $f(x_i, y_{j+1}) = p_{ij}(x_i, y_{j+1})$

  $$f(x_i, y_{j+1}) = \begin{bmatrix} 1 & 0 & h & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- $f(x_{i+1}, y_j) = p_{ij}(x_{i+1}, y_j)$

  $$f(x_{i+1}, y_j) = \begin{bmatrix} 1 & h & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- $f(x_{i+1}, y_{j+1}) = p_{ij}(x_{i+1}, y_{j+1})$

  $$f(x_i, y_j) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Portanto, podemos escrever o sistema determinado:

$$\begin{bmatrix} f(x_i, y_j) \\ f(x_i, y_{j+1}) \\ f(x_{i+1}, y_j) \\ f(x_{i+1}, y_{j+1}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & h & 0 \\ 1 & h & 0 & 0 \\ 1 & h & h^2 & h^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Leia mais em:
https://en.wikipedia.org/wiki/Bilinear_interpolation

### 2.2 Interpolação Bicubica

Na interpolação bilinear cada ponto interpolado depende apenas dos valores dos 4 pontos mais próximos, como definimos anteriormente. Esse tipo de técnica pode causar uma aparência quadrada na imagem interpolada, como vemos abaixo.

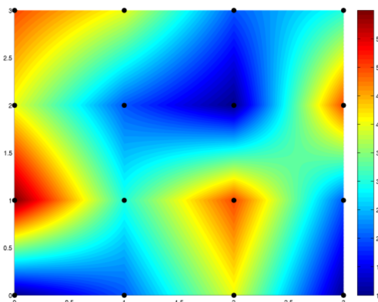


Figure 1: Figura gerada por uma interpolação bilinear. Fica aparente na imagem como a falta de suavidade na interpolação "mostra" quadrados delimitados pelas bordas de cada interpolação. Fonte: http://www.codecogs.com/library/maths/approximation/interpolation /multivariate.php

Para diminuir esse efeito usamos a interpolação bicubica, que exige que a função interpoladora $v(x, y)$ seja contínua e também que as primeiras derivadas em $x$ e $y$ e a derivada primeira mista sejam contínuas. Note que em nosso exemplo não faz sentido exigir que a imagem de entrada tenha informação das derivadas parciais (e será que faz sentido perguntar o valor da derivada parcial em um ponto da imagem?) portanto teremos que aproximar essas derivadas parciais.

Iremos calcular as derivadas parciais de forma aproximada, por diferenças finitas. Dado um ponto $(x_i, y_j)$, é interessante fazer nossa aproximação ao redor do ponto, de forma centralizada:

$$g'(x) \approx \frac{g(x+h) - g(x-h)}{2h}$$

As derivadas parciais são calculadas variando apenas um dos termos, e, portanto, podem ser calculadas da mesma forma que se calcula no caso de funções de $\mathbb{R}$ pra $\mathbb{R}$. Assim, as fórmulas para as derivadas primeiras são:

$$\frac{\partial f}{\partial x}(x_i, y_j) \approx \frac{f(x_{i+1}, y_j) - f(x_{i-1}, y_j)}{2h}$$
$$\frac{\partial f}{\partial y}(x_i, y_j) \approx \frac{f(x_i, y_{j+1}) - f(x_i, y_{j-1})}{2h}$$

Resta então calcular as derivadas mistas. Isto será feito utilizando as já calculadas derivadas parciais. Ou seja, o método é equivalente a calcular a derivada parcial com relação à $x$ na função dada pela derivada parcial com relação à $y$. Formalmente,

$$\frac{\partial^2 f}{\partial x \partial y}(x_i, y_j) \approx \frac{\frac{\partial f}{\partial y}(x_{i+1}, y_j) - \frac{\partial f}{\partial y}(x_{i-1}, y_j)}{2h}$$

É importante observar que o método da diferença centralizada não pode ser aplicado nos pontos das bordas. Para obter estes valores, será necessário calcular utilizando a diferença unilateral correspondente. Os casos de borda são

$$\frac{\partial f}{\partial x}(x_0, y_j) \approx \frac{f(x_1, y_j) - f(x_0, y_j)}{h}$$
$$\frac{\partial f}{\partial x}(x_{p-1}, y_j) \approx \frac{f(x_{p-1}, y_j) - f(x_{p-2}, y_j)}{h}$$
$$\frac{\partial f}{\partial y}(x_i, y_0) \approx \frac{f(x_i, y_1) - f(x_i, y_0)}{h}$$
$$\frac{\partial f}{\partial y}(x_i, y_{p-1}) \approx \frac{f(x_i, y_{p-1}) - f(x_i, y_{p-2})}{h}$$

Novamente, definimos a função interpoladora por partes, e chamamos o polinômio que interpola $f$ no quadrado $Q_{ij}$ de $p_{ij}(x, y)$. Teremos que:

$$p_{ij}(x, y) = \begin{bmatrix} 1 & (x - x_i) & (x - x_i)^2 & (x - x_i)^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ (y - y_j) \\ (y - y_j)^2 \\ (y - y_j)^3 \end{bmatrix}$$

Para determinar os dezesseis coeficientes de $p_{ij}$ vamos determinar dezesseis condições de igualdade, que serão entre $f$ e $p_{ij}$ assim como suas derivadas parciais e primeira derivada mista, nos quatro pontos de $Q_{ij}$. Seguindo o mesmo procedimento tomado na seção anterior, para a interpolação bilinear, é possível escrever (e é um bom exercício que pode estar em seu relatório) essas dezesseis equações de modo a formar o sistema seguinte.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = B \begin{bmatrix} f(x_i, y_j) & f(x_i, y_{j+1}) & \frac{\partial f(x_i, y_j)}{\partial y} & \frac{\partial f(x_i, y_{j+1})}{\partial y} \\ f(x_{i+1}, y_j) & f(x_{i+1}, y_{j+1}) & \frac{\partial f(x_{i+1}, y_j)}{\partial y} & \frac{\partial f(x_{i+1}, y_{j+1})}{\partial y} \\ \frac{\partial f(x_i, y_j)}{\partial x} & \frac{\partial f(x_i, y_{j+1})}{\partial x} & \frac{\partial^2 f(x_i, y_j)}{\partial x \partial y} & \frac{\partial^2 f(x_i, y_{j+1})}{\partial x \partial y} \\ \frac{\partial f(x_{i+1}, y_j)}{\partial x} & \frac{\partial f(x_{i+1}, y_{j+1})}{\partial x} & \frac{\partial^2 f(x_{i+1}, y_j)}{\partial x \partial y} & \frac{\partial^2 f(x_{i+1}, y_{j+1})}{\partial x \partial y} \end{bmatrix} B^T$$

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

Leia mais em:
https://en.wikipedia.org/wiki/Bicubic_interpolation

# 3 O Programa

Você deve entregar três **function files** do **Octave**. Segue a descrição de cada um.

**Dica:** Para ler e escrever imagens leia a documentação do **Octave** para **imread** e **imwrite**. Estas duas funções bastam para o pedido. **Tome cuidado para não comprimir suas imagens ao usar as funções de manipulação de imagens do Octave. O imwrite, por padrão comprime as imagens em 75%.**

## 3.1 compress(originalImg, k)

Esta função deve comprimir uma imagem. Suponha que ela é quadrada e tem $p^2$ pixels e que $\exists n \in \mathbb{N}$ tal que $p = n + (n-1)k$. Comprima a imagem original para o tamanho $n \times n$ ao manter as linhas e colunas de índice $i$ tais que $i$ é da forma $i \equiv 0 \mod k+1$; remova o resto dos pixels.

### 3.1.1 Parâmetros

- **originalImg** - nome da imagem original que você irá comprimir
- **k** - taxa de compressão, ou seja, número de linhas e colunas que serão removidas respectivamente.

### 3.1.2 Saída

Você deve gerar uma imagem chamada **compressed.png** como pedido acima.

## 3.2 decompress(compressedImg, method, k, $h$, $\overline{x}$, $\overline{y}$)

Esta função deve realizar o papel de decompressão. Você deve expandir uma imagem quadrada com tamanho de lado $n$ (pixels) de modo que o lado final tenha tamanho $n + (n-1)k$. Faça isso adcionando $k$ linhas e $k$ colunas entre cada uma das $n$ linhas e colunas.

Depois, para cada ponto que você precisar preencher, interpole três funções (uma para cada cor RGB) usando o método especificado em **method** e descrito nesta seção.

### 3.2.1 Parâmetros

- **compressedImg** - nome do arquivo da imagem que comprimida
- **method** - método de interpolação a ser usado
  - **1** - Bilinear
  - **2** - Bicubico
- **k** - Taxa de decompressão, como especificada na compressão e acima
- $h$ - O tamanho do lado do quadrado que interpola os pontos, como definido anteriormente. Se $F$ imagem a ser decomprimida, então assumimos que $(x_i, y_j) = (\overline{x} + ih, \overline{y} + jh)$.
- $\overline{x}$ como definido anteriormente.
- $\overline{y}$ como definido anteriormente.

### 3.2.2 Saída

Gere uma imagem chamada **decompressed.png**. Esta é a imagem gerada pela sua matriz de tamanho $n + (n-1)k$ após a interpolação das entradas criadas.

## 3.3 calculateError(originalImg, decompressedImg)

Este programa deve calcular o erro entre a imagem original e a imagem descomprimida do seguinte modo:

Considere $origR$ a matriz correspondente a cor RED da imagem original, e análogo para $B$ e $G$. Considere também $decR$ a matriz correspondente a cor RED da sua imagem descomprimida, e análogo para $G$ e $B$. Então definimos o erro $err$ como

$$err = \frac{errR + errG + errB}{3}$$

onde

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

e análogo para G e B.

### 3.3.1 Parâmetros

- **originalImg** - nome do arquivo que contém a imagem original
- **decompressedImg** - nome do arquivo que contém a imagem depois de comprimida e descomprimida

### 3.3.2 Saída

Imprima apenas um número na saída padrão: o erro calculado.

## 4 Parte 1 - O Zoológico

Nesta parte você deve usar uma função $f : \mathbb{R}^2 \to \mathbb{R}^3$ de classe $C^2$ para gerar uma imagem grande em RGB, no formato especificado na parte "Programa", para testar nosso método de compressão, isto é, comprimir, descomprimir e calcular o erro.

Teste com exemplos próprios, mas certifique-se de testar pelo menos com a função:

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

Faça experimentos e responda questões como:

- Funciona bem para imagens preto e branco?
- Funciona bem para imagens coloridas?
- Funciona bem para todas as funções de classe $C^2$?
- E para funções que não são de classe $C^2$?
- Como o valor de $h$ muda a interpolação?
- Como se comporta o erro?

Responda também a esta questão:

Considere uma imagem de tamanho $p^2$. Comprima-a com $k = 7$. Para obter a descompressão, podemos rodar **decompress** com $k = 7$. Experimente alternativamente usar **decompress** *três vezes* com $k = 1$ nas três. Compare os resultados. Escreva no relatório suas conclusões.

## 5 Parte 2 - A Selva

Agora, na selva, sua tarefa é realizar algo muito parecido com a parte 1, mas com uma imagem real (foto ou desenho). Note que agora a existência de uma $f$ de classe $C^2$ é uma suposição muito provavelmente falsa.

Responda no relatório todas as questões ainda pertinentes da Parte 1, e outras que você julgar boas.

## 6 Para entregar

Entregue um arquivo .tar com nome **ep2.tar**. Ao descompactá-lo devemos encontrar uma pasta **ep2** com os seguintes conteúdos:

- **decompress.m**, o programa em Octave como function file
- **compress.m**, o programa em Octave como function file
- **calculateError.m**, o programa em Octave como function file
- **relatorio.pdf**, um relatório completo do EP, em pdf, incluindo:
  - Decisões de projeto quanto à implementação dos métodos
  - Observações pedidas quanto aos experimentos
  - Exemplos ilustrativos dos resultados