2º Projeto Prático

Universidade Federal da Bahia - UFBA
Departamento de Ciência da Computação - DCC
MAT174 - Cálculo Numérico

06 de Junho de 2019

1 Objetivos

- Desenvolver técnicas de programação
- Aplicar os conceitos de **Interpolação** vistos em sala de aula

2 Data de Entrega

- **02/07/2019**

3 Proposta

- Desenvolvimento de duas técnicas de Interpolação para auxiliar na restauração de imagens
  - Método de Newton com uso de Diferenças Divididas
  - Método de Gregory-Newton com uso de Diferenças Finitas

4 Itens do projeto

- Código-fonte do projeto
  - Organizado em módulos (funções separadas)
    - Diferença Dividida
    - Diferença Finita
    - Restauração da imagem
    - Função main
- Relatório escrito

4.1 Implementação

O programa deverá fazer o cálculo de novos pontos da imagem a partir de uma tabela de pontos (pixels), predefinida pelo usuário.

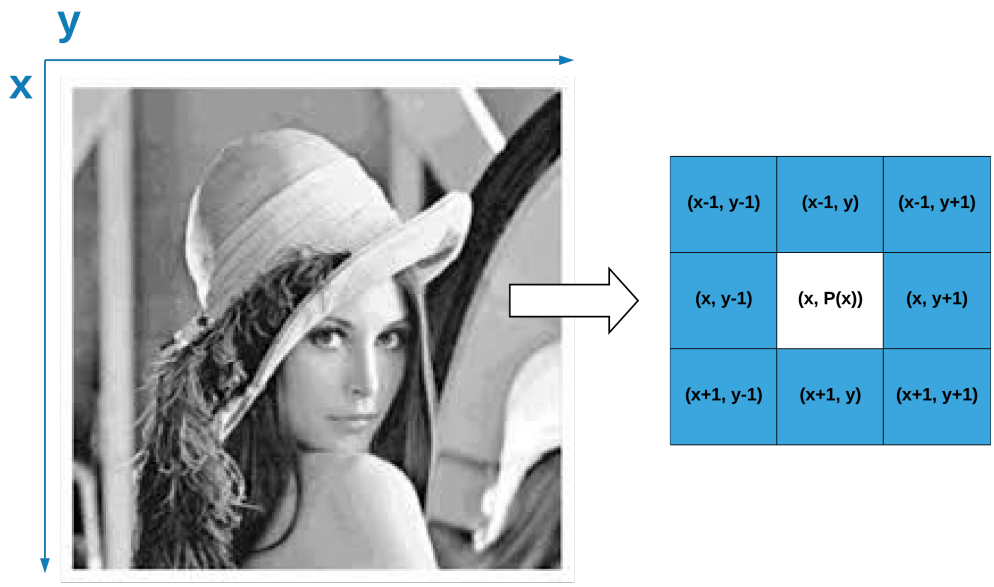

Figure 1: Amostra de pixels na imagem

1. Deverá ser implementada um menu, onde o usuário irá escolher qual o tipo de método quer executar no momento
   (a) Método de Newton (Diferenças Divididas)
   (b) Método de Gregory-Newton (Diferenças Finitas)

2. Depois da escolha do método, o usuário deverá inserir a quantidade de pontos que serão utilizados na interpolação
   (a) O usuário deverá digitar um número inteiro PAR
3. E o **grau do polinômio**
   - Em seguida, deverá inserir todos os valores dos nós da interpolação
   - E o valor de x para calcular $P_n(x)$

4. Por último, deverá inserir o caminho da imagem que será utilizada na interpolação

5. O programa deverá IMPRIMIR, para cada método de interpolação, a tabela adequada:

(a) Tabela de diferenças divididas

(b) Tabela de diferenças finitas

6. O programa deverá imprimir na tela o valor do novo ponto: $(x, P_n(x))$

7. Com o auxílio das funções pré-implementadas, o programa deverá salvar a imagem reconstruída com os novos pontos calculados em **formato .PNG**

   - A cor do pixel na posição $(x, P_n(x))$ será igual a média ponderada entre os índices dos pontos (pixels) e suas respectivas cores

4.2 Relatório

- Elaboração do relatório do projeto contendo os seguintes itens (MÁXIMO de 2 páginas)

  1. Apresentação das imagens e pontos escolhidos para reconstrução
  2. Apresentação dos resultados finais (novas imagens)

5 Detalhes importantes

- O trabalho poderá ser densenvolvido **individualmente**, em **dupla** ou **trio**
- Não serão aceitos pseudo-códigos
- Não serão aceitos códigos em outras linguagens
- Não serão aceitos trabalhos incompletos:
  - Somente o relatório ou somente o código
- **Projetos COPIADOS terão nota 0**
- Não serão aceitos entrega do código final dentro do .pdf
  - O aluno deverá enviar os arquivos para compilação do projeto [.c , .cpp, .h, .hpp]

6 Avaliação

- O projeto será avaliado em dois pontos:
  - Funcionamento do código (9.0);
    - Permite entrada/saída de dados (0.5);

    - Implementação correta + dados de saída corretos (Diferenças Dividas + Polinômio Interpolador) (4.0);
    - Implementação correta + dados de saída corretos (Diferenças Finitas + Polinômio Interpolador) (3.5);
    - Cálculo das novas cores (1.0);
  - Organização do relatório final (1.0);
    - Descrição detalhada contendo todos os itens solicitados (1.0);

- O projeto deverá ser enviado via Moodle (no espaço reservado) em um arquivo compactado (.zip, .tar, .rar) contendo:
  - O relatório final em .pdf
  - Os arquivos .c ou .cpp (e, se houver, .h ou .hpp)

- **Os alunos deverão enviar o trabalho até às 23:55hrs da data prevista para entrega**

7 Mais detalhes sobre o proejto:

Como discutido em sala, o objetivo do projeto é fazer a alteração das cores dos *pixels* de uma imagem através da média das cores dos *pixels vizinhos*.

Desse modo, a primeira etapa do projeto se resume a definição de qual a posição do *pixel* que será alterado. Para isso, deverão ser utilizados os conceitos aprendidos no contexto da Interpolação (Diferenças Divididas e Finitas).

Assim, o usuário deverá fazer o carregamento da imagem (através da função **loadPNG**, definida na classe Imagem). Após o carregamento, é possível saber o tamanho da imagem (largura e altura) e com isso, será possível saber a quantidade de *pixels* que a imagem possui.

Vamos avaliar a imagem abaixo (Figura 2)

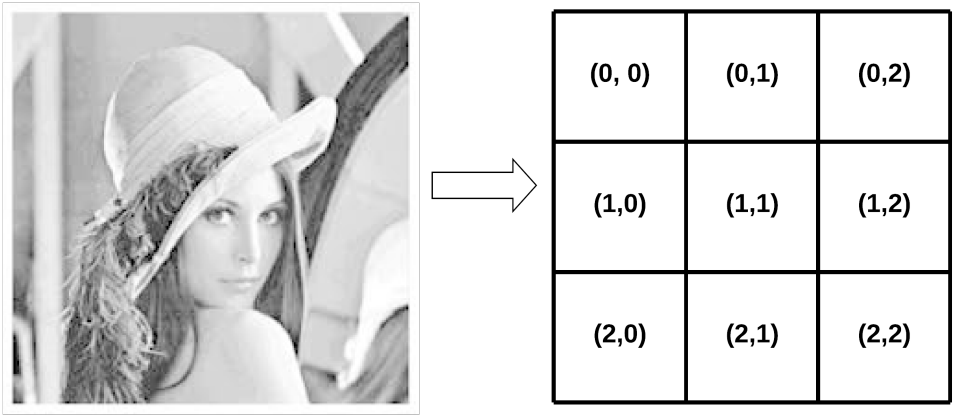

Figure 2: Amostra de pixels na imagem

Vamos supor que a imagem tenha tamanho 3×3 (largura e altura). Desse modo, as posições dos *pixels* estão descritas na matriz ao lado da imagem da Lena. O usuário irá inserir os nós da interpolação, que corresponderão as coordenadas (x, y) de alguns *pixels*, por exemplo:

- $(0, 0)$, $(1, 1)$

Depois, é solicitado ao usuário, que digite o valor de x, sendo esse, diferente das abcissas que foram digitadas nos nós da interpolação (no exemplo anterior, poderia ser digitado $x = 2$. Assim, o programa (através da interpolação) irá calcular o valor de $f(x)$ (y), correspondente. Esse **ponto**, corresponderá a uma nova posição de *pixel* da imagem.

Vamos supor então, que a interpolação devolva $f(x) = 2$. Assim, o ponto final será $(x, y) = (2, 2)$. Lembrando que, nessa etapa, deverá ser impresso na tela as

tabelas de diferenças divididas e finitas (como solicitado no início do PDF).

Por fim, agora com a interpolação finalizada, deverá ser calculada a nova cor do *pixel* na posição $(2, 2)$, através da média ponderada das cores dos nós da interpolação previamente digitados pelo usuário. As cores dos *pixels* poderão ser acessadas através da função **at(x,y)**, também implementada na classe Imagem. Desse modo, teremos:

$$\text{CorFinal} = (1 \times [\text{Cor do pixel na posição (0,0)}] + 2 \times [\text{Cor do pixel na posição (1,1)}])/3$$

onde 1 e 2, são os índices da tabela de nós (digitando na ordem). E o valor do denominador corresponderá a soma de todos os índices, que nesse caso será 1 + 2 = 3.

Ainda sobre o exemplo, vamos supor que a cor do *pixel* em $(0, 0) = 200$ e a cor do *pixel* em $(1, 1) = 100$, então a equação acima ficará:

$$CorFinal = \frac{[1\times100]+[2\times200]}{3} = 133.33$$

como a imagem é formada por valores inteiros, o número será automáticamente arredondado para 133, quando o **operador ()** for chamada para inserir o novo valor do *pixel* na posição (teste(2,2) = 133).