Escola Politécnica da Universidade de São Paulo

PTC5725 – Introdução aos Métodos Espectrais

Relatório: Primeira Lista de Exercícios

Renan de Luca Avila

São Paulo, 2 de outubro de 2025

1 Introdução e Organização

Este relatório apresenta as soluções para a primeira tarefa proposta na disciplina *Introdução aos Métodos Espectrais (PTC5725)*. As questões abordam a interpolação de Lagrange e seu papel na aproximação numérica, além da demonstração da *unicidade* do polinômio interpolador, derivação da matriz de diferenciação e estudo de interpolação com pesos baricêntricos. Durante a elaboração do relatório três referências principais foram utilizadas: [1] para fórmulas e teoria, e [2] para inspiração em relação à códigos e aplicações. Para a derivação dos pesos baricêntricos, a referência foi o documento do *edisciplinas* [3]. A escolha de linguagem para programação foi Python.

2 Base Teórica

Dados nós $x_0, \ldots, x_n$ distintos e valores $f(x_0), \ldots, f(x_n)$, o interpolador de Lagrange é

$$P_n(x) = \sum_{j=0}^{n} f(x_j)\,\ell_j(x), \qquad \ell_j(x) = \prod_{\substack{0\le k\le n \\ k\neq j}} \frac{x - x_k}{x_j - x_k}, \tag{1}$$

com $P_n(x_j) = f(x_j)$.

3 Exercício 1 — Interpolação de Lagrange

3.1 Enunciado

Escreva um código para obter os valores de $f(x_k)$ com a interpolação de Lagrange feita para os $(n+1)$ pontos $f(x_j)$.

3.2 Código-fonte (Lagrange, código isolado)

```python
import numpy as np

def lagrange_interpolation(x_nodes, y_nodes, x_eval):
    x_nodes = np.array(x_nodes, dtype=float)
    y_nodes = np.array(y_nodes, dtype=float)
    x_eval = np.array(x_eval, dtype=float)
    n = len(x_nodes)
    y_eval = np.zeros_like(x_eval, dtype=float)
    for j in range(n):
        lj = np.ones_like(x_eval, dtype=float)
        for m in range(n):
            if m != j:
                lj *= (x_eval - x_nodes[m]) / (x_nodes[j] - x_nodes[m])
        y_eval += y_nodes[j] * lj
    return y_eval
```

Listing 1: Interpolação de Lagrange (função de avaliação)

3.3 Conexão com a fórmula teórica

O laço externo soma $f(x_j)\,\ell_j(x)$, enquanto o laço interno constrói $\ell_j(x) = \prod_{m\neq j} \frac{x-x_m}{x_j-x_m}$. Assim, a implementação corresponde a $P_n(x) = \sum_j f(x_j)\ell_j(x)$.

3.4 Extensão: funções-teste, figuras e comparação com Chebyshev

Usamos nós equidistantes $x_k = -1+\frac{2k}{11}$, $k = 0,\ldots,11$, e a malha de avaliação $x_i = -1+\frac{2i}{120}$, $i = 1,\ldots,120$. As funções-teste são: e^{-x}, $\sin(\pi x)$, $\cos(\pi x)$ e $32x^6 - 48x^4 + 18x^2 - 1$. As Fig. 1–4 ilustram as aproximações. Também comparamos nós equidistantes (Fig. 5) e nós de Chebyshev-Lobatto (Fig. 6) para a função de Runge.

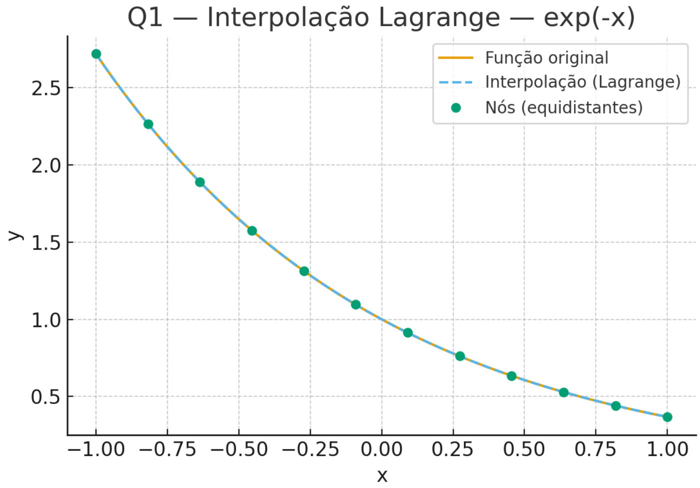

Figura 1: Ex.1 — Interpolação para e^{-x} com 12 nós equidistantes.

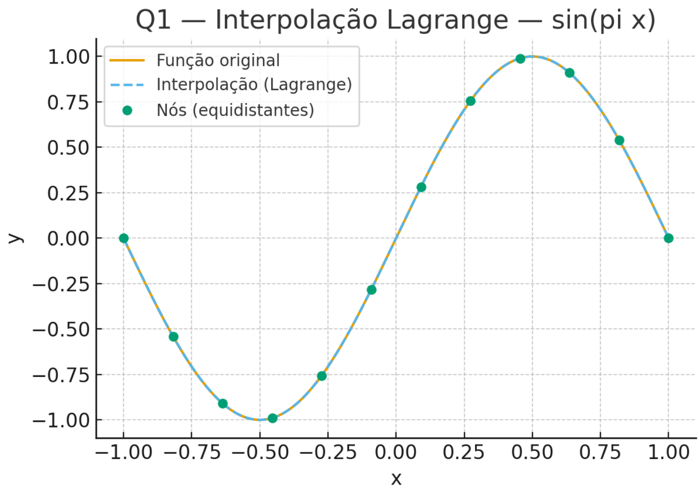

Figura 2: Ex.1 — Interpolação para $\sin(\pi x)$ com 12 nós equidistantes.

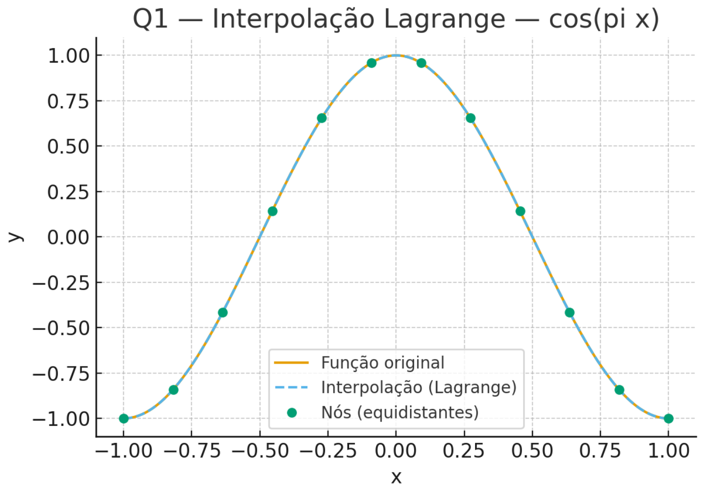

Figura 3: Ex.1 — Interpolação para $\cos(\pi x)$ com 12 nós equidistantes.

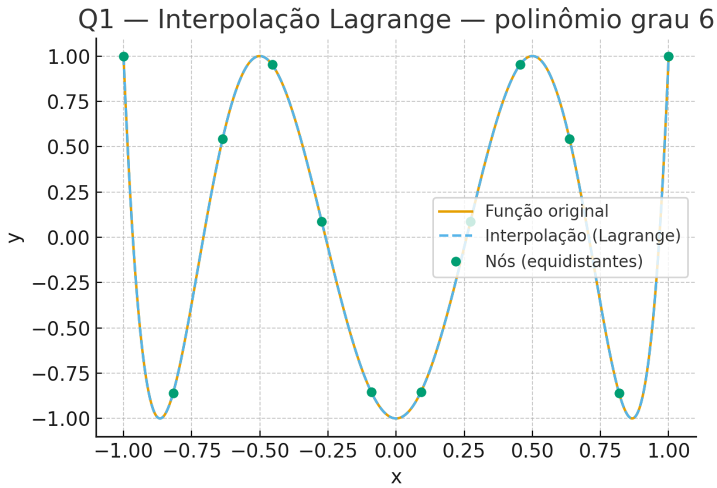

Figura 4: Ex.1 — Interpolação para $32x^6 - 48x^4 + 18x^2 - 1$ com 12 nós equidistantes.

4 Execício 2 - Unicidade do Polinômio Interpolador

4.1 Enunciado

Mostrar que o polinômio interpolante de Lagrange para $n+1$ pontos é o *único* polinômio de grau $\leq n$ que passa por todos esses pontos.

4.2 Existência

Dados nós distintos $x_0, \ldots, x_n$ e valores $f(x_0), \ldots, f(x_n)$, o polinômio de Lagrange

$$P_n(x) \;=\; \sum_{j=0}^{n} f(x_j)\,\ell_j(x), \qquad \ell_j(x) \;=\; \prod_{\substack{0\leq k\leq n\\ k\neq j}} \frac{x - x_k}{x_j - x_k},$$

satisfaz, por construção,

$$P_n(x_j) \;=\; f(x_j), \qquad j = 0, \ldots, n,$$

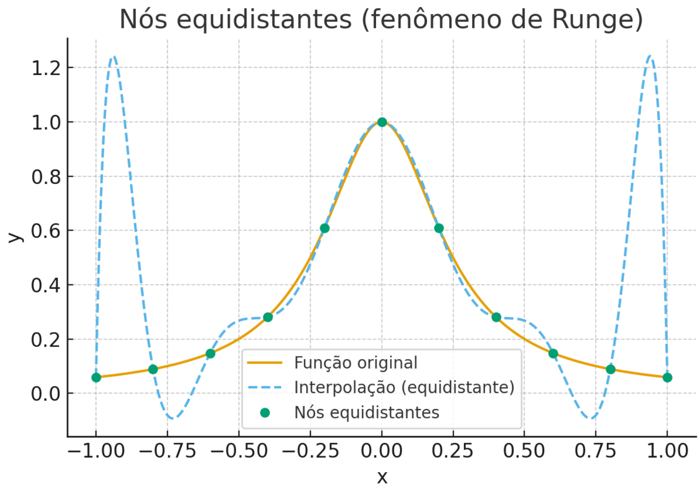

Figura 5: Ex.1 — Nós equidistantes (fenômeno de Runge) para $f(x) = \frac{1}{1+16x^2}$.

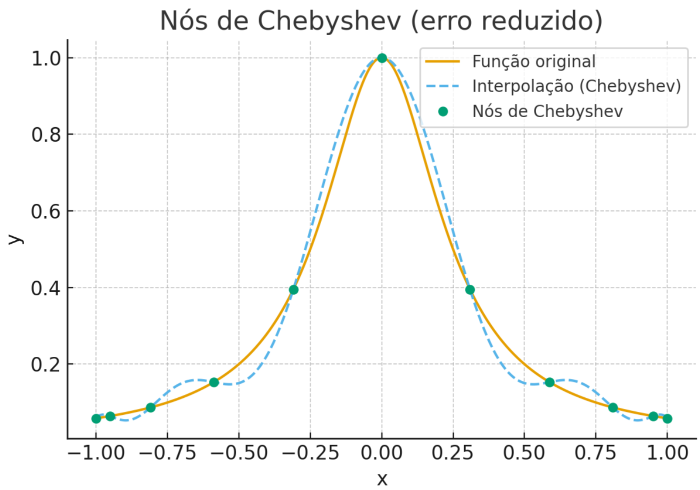

Figura 6: Ex.1 — Nós de Chebyshev-Lobatto (erro reduzido) para $f(x) = \frac{1}{1+16x^2}$.

logo *existe* pelo menos um polinômio de grau $\leq n$ que interpola os dados.

4.3 Unicidade: Prova 1 (contagem de raízes)

Suponha que existam P e Q, ambos de grau $\leq n$, tais que

$$P(x_j) \;=\; Q(x_j) \;=\; f(x_j), \qquad j = 0, \dots, n.$$

Considere a diferença

$$R(x) \;=\; P(x) - Q(x).$$

Então R é um polinômio de grau $\leq n$ com

$$R(x_j) \;=\; 0, \qquad j = 0, \dots, n,$$

isto é, R possui ao menos $n+1$ raízes *distintas*. Mas um polinômio não nulo de grau $\leq n$ não pode ter mais do que n raízes distintas. Portanto, necessariamente $R \equiv 0$ e, assim, $P \equiv Q$. Logo, o interpolador é *único*. $\square$

4.4 Unicidade: Prova 2 (matriz de Vandermonde)

Outra via é escrever o problema como um sistema linear para os coeficientes de $P(x)$. Suponha

$$P(x) \;=\; a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

Impor $P(x_j) = f(x_j)$ para $j = 0, \ldots, n$ gera

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{V(x_0,\ldots,x_n)} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \;=\; \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

A matriz V é a *Vandermonde*. Para nós *distintos*, seu determinante é

$$\det V \;=\; \prod_{0 \le i < j \le n} (x_j - x_i) \;\neq\; 0,$$

portanto V é invertível e o sistema tem *solução única* para $(a_0, \ldots, a_n)^\top$. Consequentemente, o polinômio interpolador é único. $\square$

4.5 Toy case (verificação manual com $n = 2$)

Considere três nós distintos no intervalo $[-1, 1]$:

$$x_0 = -1, \qquad x_1 = 0, \qquad x_2 = 1,$$

e valores genéricos

$$f(x_0) = a, \qquad f(x_1) = b, \qquad f(x_2) = c.$$

Buscamos um polinômio de grau ≤ 2,

$$P(x) \;=\; \alpha x^2 + \beta x + \gamma,$$

tal que

$$\begin{aligned} P(-1) = \alpha(-1)^2 + \beta(-1) + \gamma \;&=\; \alpha - \beta + \gamma \;=\; a, \\ P(0) = \gamma \;&=\; b, \\ P(1) = \alpha(1)^2 + \beta(1) + \gamma \;&=\; \alpha + \beta + \gamma \;=\; c. \end{aligned}$$

Da segunda equação, obtemos imediatamente

$$\gamma \;=\; b.$$

Somando a primeira e a terceira equações:

$$(\alpha - \beta + \gamma) + (\alpha + \beta + \gamma) \;=\; a + c \;\Longrightarrow\; 2\alpha + 2\gamma \;=\; a + c \;\Longrightarrow\; \alpha \;=\; \frac{a + c - 2b}{2}.$$

Subtraindo a primeira da terceira:

$$(\alpha + \beta + \gamma) - (\alpha - \beta + \gamma) \;=\; c - a \;\Longrightarrow\; 2\beta \;=\; c - a \;\Longrightarrow\; \beta \;=\; \frac{c - a}{2}.$$

Logo,

$$\boxed{\alpha \;=\; \frac{a+c-2b}{2}, \qquad \beta \;=\; \frac{c-a}{2}, \qquad \gamma \;=\; b}$$

são *determinados univocamente* pelos dados (a,b,c). Assim, existe um *único* $P(x) = \alpha x^2 + \beta x + \gamma$ que interpola os três pontos. Como verificação rápida, se tomarmos, por exemplo,

$$(a,b,c) \;=\; \big(0,\,1,\,0\big),$$

então

$$\alpha \;=\; \frac{0+0-2\cdot 1}{2} \;=\; -1, \qquad \beta \;=\; \frac{0-0}{2} \;=\; 0, \qquad \gamma \;=\; 1,$$

e obtemos

$$P(x) \;=\; 1-x^2,$$

o que de fato satisfaz $P(-1)=0$, $P(0)=1$, $P(1)=0$ de forma *única*.

5 Exercício 3 — Matriz de Diferenciação Generalizada

5.1 Enunciado

Demonstre a expressão da matriz de diferenciação generalizada e escreva um código para obtê-la.

Estrutura:

- Parte teórica: apresente a dedução detalhada dos elementos da matriz de diferenciação D em grade arbitrária, incluindo as expressões para D_{ij} $(i \neq j)$ e D_{ii}, e a forma equivalente com pesos baricêntricos.
- Parte prática: implemente, em `Python` (ou `Julia`), um código que:
  1. construa D a partir de um vetor de nós x (grade arbitrária);
  2. aplique D^k a valores $f(x)$ para aproximar derivadas de ordem k;
  3. valide a implementação com funções-teste no intervalo $[-1,1]$.

5.2 Dedução de D

Seja $P_n(x)$ o interpolador de Lagrange. Então

$$P_n'(x_i) = \sum_{j=0}^{n} f(x_j)\,\ell_j'(x_i) \;\Rightarrow\; f'(x_i) \approx \sum_{j=0}^{n} D_{ij} f(x_j), \quad D_{ij} = \ell_j'(x_i). \tag{2}$$

Para $i \neq j$, escreva $\ell_j(x) = (x-x_i)\Phi_{ij}(x)$, onde

$$\Phi_{ij}(x) = \frac{1}{x_j - x_i} \prod_{k\neq j,i} \frac{x-x_k}{x_j-x_k}. \tag{3}$$

Logo,

$$\ell_j'(x_i) = \Phi_{ij}(x_i) = \frac{1}{x_j - x_i} \prod_{k\neq j,i} \frac{x_i-x_k}{x_j-x_k}. \tag{4}$$

Com $a_j = \prod_{k\neq j}(x_j - x_k)$ e $a_i = \prod_{k\neq i}(x_i - x_k)$,

$$\prod_{k\neq j,i} \frac{x_i - x_k}{x_j - x_k} = \frac{a_i}{(x_i - x_j)\, a_j}, \tag{5}$$

de modo que

$$\boxed{D_{ij} = \frac{a_i}{a_j} \frac{1}{x_i - x_j} \quad (i \neq j)}, \qquad \boxed{D_{ii} = -\sum_{j\neq i} D_{ij}}. \tag{6}$$

Os elementos da diagonal devem ser o oposto da soma dos demais elementos da linha. Isso garante, por exemplo, que $D\mathbf{1} = \mathbf{0}$ (a derivada de uma função constante é zero), o que é um teste de sanidade importante.

5.3 Código-fonte (matriz D e diferenciação)

```
import numpy as np

def barycentric_weights(x):
    x = np.asarray(x, dtype=float)
    n = x.size
    w = np.ones(n, dtype=float)
    for j in range(n):
        diff = x[j] - x
        diff[j] = 1.0
        w[j] = 1.0 / np.prod(diff)
    return w

def diff_matrix(x):
    x = np.asarray(x, dtype=float)
    n = x.size
    w = barycentric_weights(x)
    D = np.zeros((n, n), dtype=float)
    for i in range(n):
        for j in range(n):
            if i != j:
                D[i, j] = (w[j] / w[i]) / (x[i] - x[j])
        D[i, i] = -np.sum(D[i, :]) + D[i, i]
    return D

def differentiate(x, fvals, k=1):
    x = np.asarray(x, dtype=float)
    fvals = np.asarray(fvals, dtype=float)
    D = diff_matrix(x)
    M = np.eye(len(x))
    for _ in range(k):
        M = M @ D
    return M @ fvals
```

Listing 2: Matriz de diferenciação e aplicação a f

5.4 Por que usar pesos baricêntricos em D?

Os pesos baricêntricos $w_j = 1/\prod_{m\neq j}(x_j - x_m)$ surgem naturalmente na derivada das bases de Lagrange e permitem reescrever a matriz de diferenciação como

$$D_{ij} = \frac{w_j}{w_i}\,\frac{1}{x_i - x_j} \quad (i \neq j), \qquad D_{ii} = -\sum_{j\neq i} D_{ij}.$$

Esta forma é *equivalente* à expressão com $a_j = \prod_{m\neq j}(x_j - x_m)$, porém é numericamente mais estável (evita produtos de muitos fatores) e facilita implementações vetorizadas. Os mesmos pesos também são usados na *avaliação baricêntrica* do interpolante, o que não mistura temas, mas reaproveita um mesmo objeto matemático do interpolador de Lagrange em dois contextos: naturalmente na diferenciação e de forma intencional na avaliação.

5.5 Validação experimental e figuras

As Fig. 7–10 mostram comparação da derivada exata e numérica, e dos erros absolutos, para $f(x) = e^x \sin(5x)$ em duas grades.

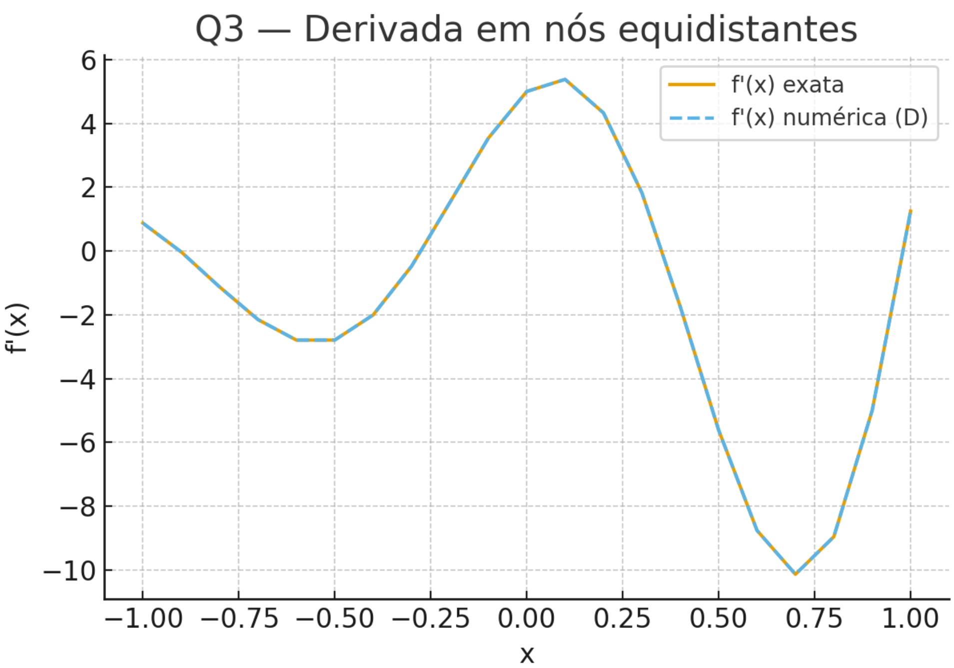

Figura 7: Ex.3 — Derivada em nós equidistantes.

6 Exercício 4 — Interpolação Baricêntrica

6.1 Enunciado

Analise o artigo sobre interpolação baricêntrica e escreva um código para efetuá-la. Dado um conjunto de nós x_k com valores $y_k = f(x_k)$, implemente uma rotina que avalie o interpolante baricêntrico em pontos de consulta x_j.

Estrutura:

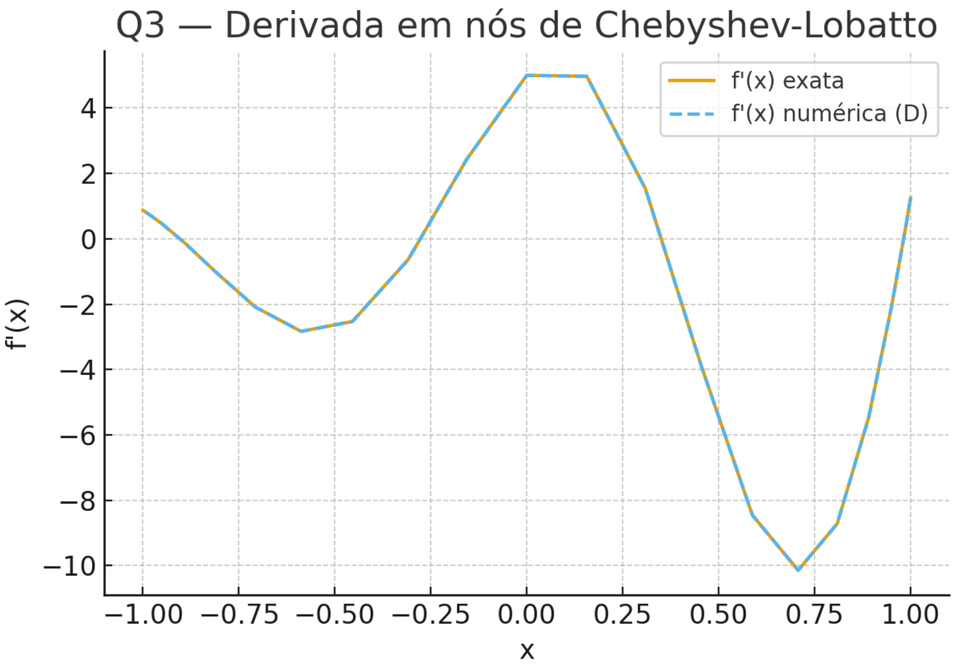

Figura 8: Ex.3 — Derivada em nós de Chebyshev-Lobatto.

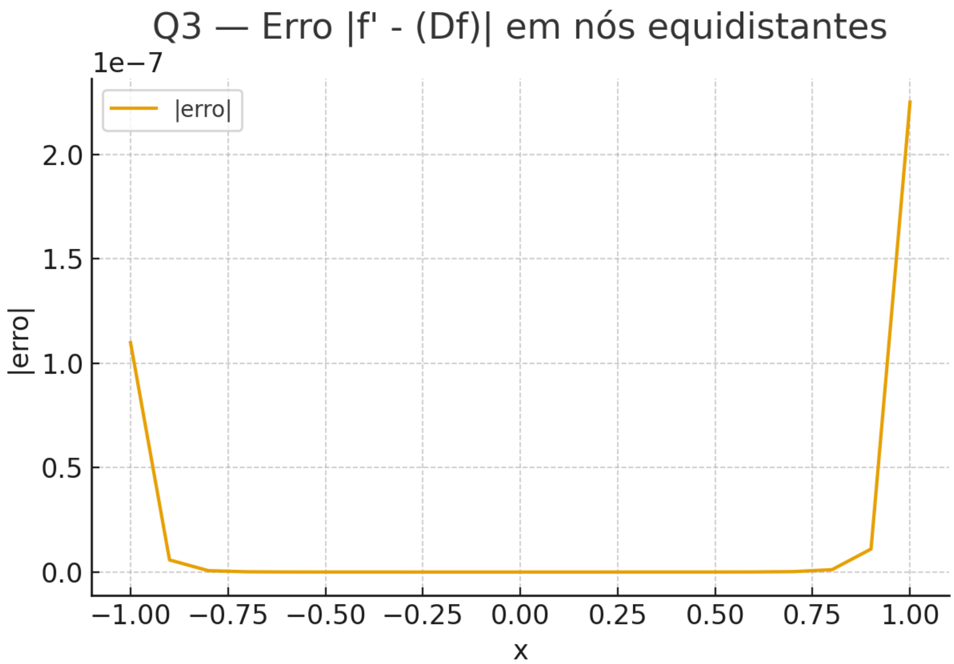

Figura 9: Ex.3 — Erro $|f' - (Df)|$ em grade equidistante.

- Parte teórica: resuma a fórmula baricêntrica

$$p(x) = \frac{\displaystyle\sum_{k=0}^{n} \frac{w_k}{x - x_k} y_k}{\displaystyle\sum_{k=0}^{n} \frac{w_k}{x - x_k}}, \qquad w_k \propto \frac{1}{\prod_{m \neq k}(x_k - x_m)},$$

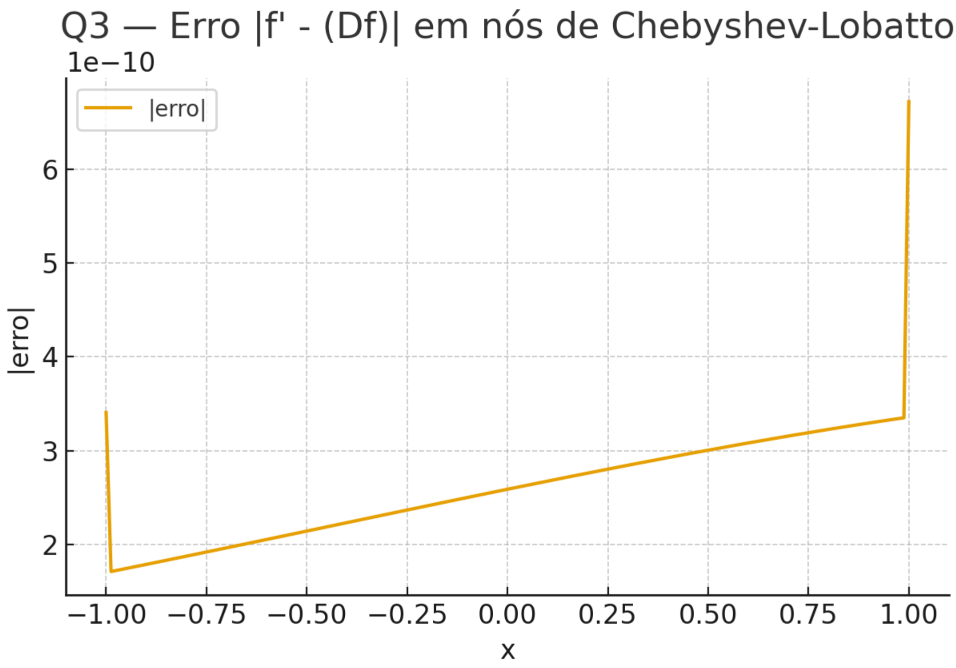

Figura 10: Ex.3 — Erro $|f' - (Df)|$ em grade Chebyshev-Lobatto.

explicando a ideia dos *pesos* w_k e o tratamento especial do caso $x = x_k$ (retornar exatamente y_k).

- Parte prática (código):
  1. implemente o cálculo de pesos baricêntricos genéricos a partir de x_k;
  2. implemente uma função de avaliação baricêntrica $p(x)$ que:
     - use os pesos w_k e trate de forma robusta $x \approx x_k$;
     - aceite pesos fechados para nós *Chebyshev-Lobatto* (quando aplicável).
  3. valide com funções de teste em $[-1, 1]$ (por exemplo, e^{-x}, $\sin(\pi x)$, $\cos(\pi x)$ e o polinômio dado em aula);
  4. compare qualitativamente/quantitativamente com a avaliação "tradicional" de Lagrange (quando disponível), destacando estabilidade e custo.

6.2 Fórmula e pesos

A forma baricêntrica é

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j)}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}, \tag{7}$$

com $w_j \propto 1/\prod_{m \neq j}(x_j - x_m)$; para Chebyshev-Lobatto $x_j = \cos(j\pi/n)$, pode-se usar $w_0 = w_n = \frac{1}{2}(-1)^j$ e $w_j = (-1)^j$ (interiores).

6.3 Código-fonte (baricêntrica)

```python
import numpy as np

def bary_weights_generic(x):
    x = np.asarray(x, dtype=float)
    n = x.size
    w = np.ones(n, dtype=float)
    for j in range(n):
        diff = x[j] - x
        diff[j] = 1.0
        w[j] = 1.0 / np.prod(diff)
    return w

def bary_weights_cheb2(n):
    w = np.ones(n+1, dtype=float)
    for j in range(n+1):
        w[j] = (-1.0)**j
    w[0] *= 0.5
    w[-1] *= 0.5
    return w

def bary_eval(x, y, xq, w=None):
    x = np.asarray(x, dtype=float)
    y = np.asarray(y, dtype=float)
    xq = np.asarray(xq, dtype=float)

    if w is None:
        w = bary_weights_generic(x)
    else:
        w = np.asarray(w, dtype=float)

    yq = np.empty_like(xq, dtype=float)
    for idx, z in enumerate(xq):
        diffs = z - x
        mask = np.isclose(diffs, 0.0, atol=0.0, rtol=0.0)
        if mask.any():
            yq[idx] = y[mask.argmax()]
        else:
            numer = np.sum((w / diffs) * y)
            denom = np.sum(w / diffs)
            yq[idx] = numer / denom
    return yq
```

Listing 3: Pesos e avaliação baricêntrica

6.4 Experimento e figura

Para $f(x) = \frac{1}{1+16x^2}$ com $n = 10$ e nós de Chebyshev-Lobatto, a Fig. 11 ilustra a aproximação baricêntrica.

6.5 Intuição e vantagens da interpolação baricêntrica

A *interpolação baricêntrica* é uma forma de *avaliar* o polinômio interpolador de Lagrange que evita construir explicitamente os polinômios-base $\ell_j(x)$ ou resolver sistemas de Vandermonde. Em vez de trabalhar com produtos longos e numericamente frágeis, usamos

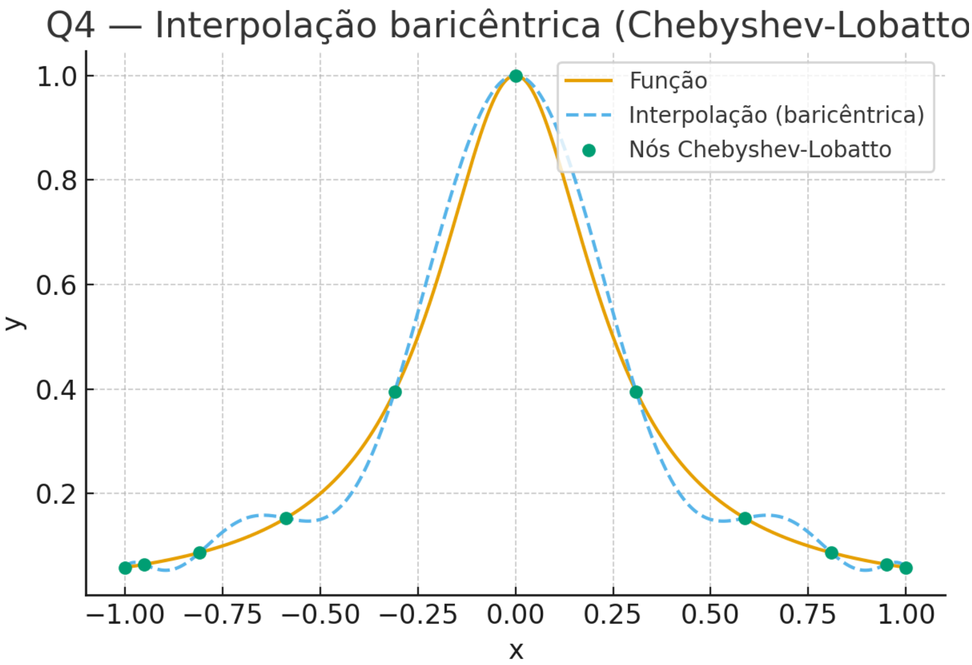

Figura 11: Ex.4 — Interpolação baricêntrica em nós de Chebyshev-Lobatto.

uma *fórmula racional* que combina os dados com *pesos* pré-computados:

$$p(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j)}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_j}}, \qquad \text{com} \quad w_j \propto \frac{1}{\prod_{m \neq j}(x_j - x_m)}.$$

Apesar da aparência *racional*, $p(x)$ é exatamente o mesmo polinômio interpolador de grau $\leq n$: a razão cancela as singularidades e recupera um polinômio. Os pesos w_j podem ser escalonados por qualquer constante comum sem afetar $p(x)$ (a constante cancela no quociente). Um *detalhe prático* importante: se x coincide com algum nó x_k, então definimos $p(x_k) = f(x_k)$ diretamente (o que a implementação deve tratar para evitar divisão por zero).

Por que ela é necessária? (Motivação prática) A forma ingênua de avaliar o interpolador

$$P_n(x) = \sum_{j=0}^{n} f(x_j)\, \ell_j(x), \qquad \ell_j(x) = \prod_{m \neq j} \frac{x - x_m}{x_j - x_m},$$

é computacionalmente cara e *instável* para n moderado/grande:

- **Produtos longos e cancelamento catastrófico:** os fatores $(x - x_m)$ e $(x_j - x_m)$ podem ser muito pequenos/grandes, induzindo *under/overflow* e perda de significância numérica ao formar $\ell_j(x)$.
- **Custo por ponto de avaliação:** reconstituir todas as $\ell_j(x)$ em cada x custa tipicamente $\mathcal{O}(n^2)$ por ponto avaliado. Para muitos pontos de consulta, isso escala mal.

- **Vandermonde mal condicionado:** tentar obter coeficientes do polinômio via sistema $Va = f$ (com V de Vandermonde) amplifica erros de arredondamento e ruído em $f(x_j)$ conforme n cresce ou os nós são desfavoráveis.

A *baricêntrica* contorna esses problemas ao:

1. **Separar pré-cálculo e avaliação:** computamos os pesos w_j *uma única vez* (custos $\mathcal{O}(n^2)$ no caso genérico; $\mathcal{O}(n)$ em nós especiais como Chebyshev-Lobatto com fórmulas fechadas) e depois avaliamos $p(x)$ em $\mathcal{O}(n)$ *por ponto.*

2. **Melhor estabilidade numérica:** evita formar explicitamente os produtos de $\ell_j(x)$; as razões $\frac{w_j}{x-x_j}$ tendem a ser mais bem comportadas numericamente.

3. **Tratamento robusto em $x \approx x_j$:** a forma quociente produz uma *avaliação contínua* que, com a regra $p(x_j) = f(x_j)$, lida elegantemente com pontos próximos aos nós.

Vantagens em relação à forma tradicional

- **Eficiência:** após pré-computar w_j, a avaliação em vários x é linear no número de nós; ideal para *muitas* consultas.

- **Estabilidade:** reduz erros de arredondamento e cancelamentos por evitar produtos de muitos termos; frequentemente é o método recomendado em prática computacional moderna (*de fato, é o padrão de referência*).

- **Integração com nós de Chebyshev(-Lobatto):** com nós bem escolhidos, como Chebyshev, há fórmulas fechadas para w_j e controle melhor da *constante de Lebesgue*, mitigando oscilações em bordas e melhorando a acurácia global.

O que ela *não* resolve sozinha A interpolação baricêntrica *não* elimina o *fenômeno de Runge* se os *nós* forem ruins (por exemplo, equidistantes de alta ordem). Ela melhora a *avaliação* do interpolador, mas a *escolha dos nós* continua crítica. Em conjunto com nós de Chebyshev(-Lobatto), porém, a prática mostra ganhos substanciais em estabilidade e erro.

Resumo prático Use a fórmula baricêntrica quando:

1. Você precisa avaliar o interpolador *muitas vezes* (custo total menor).

2. Você busca *estabilidade numérica* sem montar Vandermonde nem bases de Lagrange explícitas.

3. Você pode (ou quer) usar *nós de Chebyshev(-Lobatto)* para reduzir oscilações e obter pesos w_j por fórmulas fechadas.

Reconhecimento de uso de LLM

Este relatório contou com o apoio de uma ferramenta de LLM para redação, LaTeX, código e figuras. Todo o conteúdo foi revisado pelo autor.

Referências

[1] L. N. Trefethen, *Spectral Methods in MATLAB*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000.

[2] S. Ameh, "A quick visual guide to lagrange interpolation in numerical methods." https://medium.com/@amehsunday178/a-quick-visual-guide-to-lagrange-interpolation-in-numerical-methods-1fcc47ca218b, 2023. Medium article.

[3] M. Floater, "Weights in the barycentric lagrange formula." https://edisciplinas.usp.br/pluginfile.php/9227613/mod_resource/content/1/Weight_Bary_Cheb.pdf, 2014. Arquivo do edisciplinas.