Universidade Federal de Uberlândia
Instituto de Matemática e Estatística

Bacharelado em Matemática

ESTUDO E IMPLEMENTAÇÃO DE TÉCNICAS DE INTERPOLAÇÃO POLINOMIAL

Heitor Wellington de Lima Pereira

Uberlândia-MG

2024

Heitor Wellington de Lima Pereira

ESTUDO E IMPLEMENTAÇÃO DE TÉCNICAS DE INTERPOLAÇÃO POLINOMIAL

Trabalho de conclusão de curso apresentado à Coordenação do Curso de Matemática, da Universidade Federal de Uberlândia, como requisito parcial para obtenção do grau de Bacharel em Matemática.

Orientador: Prof. Dr. Rafael Alves Figueiredo, UFU/MG

Uberlândia-MG

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Instituto de Matemática e Estatística
Av. João Naves de Àvila, 2121, Bloco 1F - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: +55 (34) 3239-4158/4156/4126 - www.famat.ufu.br - famat@ufu.br

ATA DE DEFESA - GRADUAÇÃO

Curso de Graduação em:	Bacharelado em Matemática				
Defesa de:	Trabalho de Conclusão de Curso 2 (FAMAT 31804)				
Data:	25/04/2024	Hora de início:	14:00	Hora de encerramento:	15:17
Matrícula do Discente:	11811MAT030				
Nome do Discente:	Heitor Wellington de Lima Pereira				
Título do Trabalho:	Estudo e Implementação de Técnicas de Interpolação Polinomial.				
A carga horária curricular foi cumprida integralmente?	(X) Sim ( ) Não				

Reuniu-se na sala 1F119 do IME, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Matemática, assim composta: Professores: Rosana Sueli Da Motta Jafelice-IME/UFU; Josuel Kruppa Rogenski-IME/UFU; Rafael Alves Figueiredo-IME/UFU, orientador do candidato.

Iniciando os trabalhos, o(a) presidente da mesa, Dr Rafael Alves Figueiredo, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

(X) Aprovado Nota [100] (Somente números inteiros)

OU

( ) Aprovado sem nota.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Rafael Alves Figueiredo**, **Professor(a) do Magistério Superior**, em 25/04/2024, às 16:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Josuel Kruppa Rogenski**, **Professor(a) do Magistério Superior**, em 25/04/2024, às 16:28, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rosana Sueli da Motta Jafelice**, **Professor(a) do Magistério Superior**, em 25/04/2024, às 19:06, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5366909** e o código CRC **1BE54069**.

Referência: Processo nº 23117.028936/2024-79 SEI nº 5366909

Agradecimentos

Nesta etapa final, gostaria de expressar minha profunda gratidão a todas as pessoas e instituições que me motivaram e contribuíram para a realização deste trabalho. Sem o apoio, incentivo e orientação de vocês, este TCC não teria sido possível.

Em primeiro lugar, agradeço aos meus pais, Jorge e Rita, alicerces inabaláveis de amor, compreensão e apoio incondicional. Desde os primeiros passos, semearam em mim a paixão pelo conhecimento e perseverança diante dos desafios. A vocês, dedico esta conquista com imensa gratidão. Vocês são minha base sólida, fundamentais para que eu pudesse alcançar meus objetivos nessa jornada acadêmica.

Ao meu orientador, Prof. Rafael, agradeço por sua paciência, conhecimento e orientação ao longo deste projeto. Suas sugestões e correções foram essenciais ao longo do desenvolvimento deste trabalho. Obrigado pela disposição e assiduidade nas reuniões durante tantos meses.

Aos meus amigos, Bruno, Marcello, Vitor e Wanderson, que compartilharam risadas, desafios e tantos momentos de estudo, muito obrigado por celebrarem cada conquista com a mesma intensidade que eu. Vocês tornaram essa jornada mais leve e significativa desde o início da graduação, e são amizades que levo da faculdade para o resto da vida. Agradeço também aos meus amigos, Ladislau e Viana, pelas palavras de admoestação, pelos conselhos profissionais e por sempre acreditarem em mim. A vocês dedico meu reconhecimento por laços de amizade que transcendem o tempo e a distância. Muito obrigado pelo companheirismo e sabedoria adquirida nos momentos em que estivemos juntos.

Em especial, agradeço ao IMPA/CNPq pela realização e manutenção do PICME, programa pelo qual me foi concedido uma bolsa de iniciação científica durante os primeiros anos na universidade. Na verdade, muito além da bolsa remunerada, obrigado pela experiência que enriqueceu minha formação e a oportunidade de trilhar uma trajetória marcante desde a época do PIC Jr./OBMEP. Este, com certeza, foi o grande diferencial que despertou minha paixão pela matemática e me inspirou a buscar novos desafios.

Igualmente, agradeço aos meus antigos orientadores de IC no PICME, Profª. Catiana e Prof. Jean, pelas valiosas discussões e pela oportunidade de aprendizado. A paciência e orientação dedicada de vocês sempre foram excepcionais e ajudaram muito a amadurecer meu pensamento matemático. Agradeço ainda ao Prof. Luis Renato, coordenador do PICME na época, pela oportunidade de permanência no programa, como também por sua dedicação e profissionalismo sério. Sua paixão pela matemática inspira não só a mim, mas muitos alunos, a enxergarem a matemática sob diferentes perspectivas com muito entusiasmo.

A Samuel, meu irmão, e a Katyely, minha namorada, agradeço pelo amor, carinho, apoio, compreensão e incentivo nos momentos desafiadores. Além de tudo, a presença de vocês me proporcionam momentos de pura felicidade.

Por fim, agradeço a colaboração e camaradagem dos meus colegas de curso, que juntos experimentamos desafios e trocas de conhecimento. Ao corpo docente extremamente qualificado desta unidade acadêmica, deixo também meu sincero agradecimento. Tenho orgulho de ter sido aluno de diversos professores ao longo desta caminhada. Me perdoem se em algum momento fui um aluno medíocre ou não entreguei a dedicação que vocês mereciam. Este trabalho é dedicado a todos vocês. Muito obrigado por fazerem parte da minha trajetória acadêmica e profissional.

Resumo

Este trabalho apresenta um estudo abrangente sobre técnicas de interpolação polinomial, explorando desde os métodos clássicos até abordagens mais avançadas, como a interpolação por partes. O objetivo principal é fornecer uma base sólida para a compreensão e aplicação das técnicas discutidas, de forma a investigar a validade e viabilidade dessas técnicas na aproximação de funções e na representação de conjuntos de dados, analisando suas vantagens, limitações e aplicações práticas. Para isso, foram explorados métodos como interpolação de Lagrange, forma de Newton e *splines* naturais, com implementação computacional em MATLAB/GNU-Octave. A análise detalhada das técnicas de interpolação polinomial revelou sua importância tanto teórica quanto prática. Além disso, revelaram-se fenômenos oscilatórios que prejudicam o ajuste de uma função, e apresentou-se uma solução para minimizar esse problema. Ao final, é abordado um problema prático que reforça a aplicabilidade da teoria estudada ao longo do trabalho.

Palavras-chave: Ajuste de curvas; aproximação de funções; fenômeno de Runge; interpolação polinomial; splines cúbicos.

Abstract

This works presents a comprehensive study on polynomial interpolation techniques, exploring from classical methods to more advanced approaches, such as piecewise interpolation. The main objective is to provide a solid foundation for understanding and applying the discussed techniques, in order to investigate their validity and feasibility in function approximation and data set representation, analyzing their advantages, limitations, and practical applications. For this purpose, methods such as Lagrange interpolation, Newton's form and natural splines were explored, with computational implementation in MATLAB/Octave. The detailed analysis of polynomial interpolation techniques revealed their importance both theoretically and practically. Additionally, oscillatory phenomena that affect the fitting of a function were identified, and a solution was presented to minimize this problem. Finally, a practical example is addressed that reinforces the applicability of the theory studied throughout the work.

Keywords: Cubic splines; curve fitting; function approximation; polynomial interpolation; Runge's phenomenon.

Lista de Figuras

Lista de Tabelas

Sumário

1. INTRODUÇÃO

A interpolação polinomial é uma técnica fundamental na análise numérica e em diversas áreas da matemática aplicada. Quando estudamos funções, frequentemente nos deparamos com situações em que a obtenção de uma fórmula analítica precisa é complexa ou impossível. Nesse contexto, a aproximação por polinômios assume um papel crucial, fornecendo ferramentas para representar funções de forma prática e eficiente. Em síntese, essa técnica consiste em encontrar um polinômio que passe por um conjunto finito de pontos conhecidos, permitindo a aproximação de uma função desconhecida (ou até mesmo conhecida) com base em alguns valores disponíveis. Isso é especialmente relevante na resolução de problemas práticos em diversas áreas, como engenharia, física, economia e ciência da computação, onde é necessário modelar fenômenos complexos com métodos matemáticos acessíveis e eficazes.

A escolha do tema ocorreu ao despertar interesse no assunto de ajuste de curvas, após ter finalizado a disciplina de Cálculo Numérico durante a graduação, tendo como professor da matéria o presente orientador deste trabalho. Na verdade, a intenção inicial era apresentar um estudo sobre aproximações por mínimos quadrados discretos, mas acabamos evoluindo para a interpolação polinomial. Sobre isso, é importante ressaltar que a Teoria da Aproximação geralmente envolve dois tipos de problemas. Um deles surge quando uma função é dada de forma explícita, mas queremos encontrar um tipo mais simples de função, tal como um polinômio, que possa ser utilizado para estimar valores aproximados da função dada. O outro tipo de problema está relacionado ao ajuste de pontos dados e a determinação da melhor função em certa classe para representar os dados. Como veremos, os polinômios interpoladores de Lagrange (ou de maneira mais geral, os polinômios osculadores) resolvem bem problemas do primeiro tipo e, em certas circunstâncias, os do segundo tipo também. Porém, a técnica de aproximação por mínimos quadrados oferece uma melhor discussão sobre problemas do segundo tipo, pois dispõe de outras abordagens e ultrapassa certas limitações que a interpolação polinomial possui.

Os polinômios interpoladores também estão relacionados com os polinômios de Taylor. Parafraseando Burden [1], os polinômios de Taylor podem ser descritos como um dos blocos de construção fundamentais da análise numérica. Mais especificamente, podemos defini-los (baseado em Elon [2]) do seguinte modo:

Definição. *Seja $f : I \to \mathbb{R}$ definida no intervalo I e n vezes diferenciável em $x_0 \in I$. O* ***enésimo polinômio de Taylor*** *da função f em torno de $x = x_0$ é o polinômio*

$$p(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \ .$$

Seguindo essa definição, o polinômio de Taylor de grau n em torno do ponto x_0 é uma excelente aproximação de uma função f que seja $n+1$ vezes diferenciável em uma pequena vizinhança de x_0 (isso vem de um resultado conhecido como Teorema de Taylor que, apesar de não enunciarmo-lo, pode ser facilmente encontrado em qualquer bibliografia sobre análise real). Em outras palavras, os polinômios de Taylor coincidem tão bem quanto possível com uma certa função em um ponto específico, mas eles concentram sua precisão somente perto daquele ponto. Nesse sentido, a aproximação polinomial de Taylor se restringe às situações nas quais são necessárias aproximações apenas em pontos próximos de x_0, pois não é raro encontrarmos péssimas aproximações fornecidas por tais polinômios à medida que nos afastamos de x_0. Consequentemente, para propósitos computacionais, faz-se necessário utilizar métodos que incluam informações em vários pontos. Dessa forma, a interpolação polinomial surge como solução para esse problema. Ainda segundo Burden [1], a utilização dos polinômios de Taylor em análise numérica não serve para propósitos de aproximação, mas sim para dedução de outras técnicas numéricas e estimativas de erro.

Um resultado fundamental, que podemos chamá-lo de pedra angular da aproximação polinomial, é o Teorema da aproximação de Weierstrass. Apenas o enunciaremos abaixo, mas sua demonstração pode ser encontrada em Santos [3].

Teorema 1.1 (**Teorema da aproximação de Weierstrass**). *Seja $f : [a, b] \to \mathbb{R}$ uma função contínua. Para todo $\epsilon > 0$, existe um polinômio $p(t)$ tal que $\mid f(t) - p(t) \mid < \epsilon$, para todo $t \in [a, b]$.*

Assim, esse importante teorema estabelece que qualquer função real contínua definida em um intervalo compacto pode ser aproximada uniformemente por uma sequência de polinômios. Em outras palavras, por mais complexa que seja a função, é possível encontrar um polinômio que se aproxima dela com um grau de precisão arbitrário. Esta propriedade torna os polinômios extremamente úteis para diversas aplicações, tanto na matemática aplicada quanto na própria análise real. Através deles podemos, por exemplo, estimar rapidamente a derivada ou integral de funções bem complicadas, uma vez que a derivada e integral indefinidas de um polinômio são fáceis de determinar.

Neste trabalho, exploraremos as técnicas de interpolação polinomial, aprofundando-se, principalmente, em seus fundamentos teóricos e implementações computacionais. Abordaremos diferentes métodos de interpolação, como o método de Lagrange, o método de Newton e a interpolação por partes, e analisaremos suas vantagens e desvantagens. O objetivo deste estudo é fornecer uma base sólida para a compreensão e aplicação de técnicas de interpolação polinomial em diversas áreas do conhecimento.

O trabalho está disposto da seguinte maneira: no capítulo 2 está contida a teoria necessária para o entendimento e realização da interpolação polinomial, utilizando um conjunto de nós. No capítulo 3, é explorada a fraqueza inerente da oscilação, quando a interpolação é feita a partir de uma grande quantidade de nós, e mostrada uma maneira de minimizar esse problema através dos pontos de Chebyshev. O capítulo 4 é destinado a apresentar a forma mais comumente usada de interpolação, caracterizada pela economia computacional e extrema precisão,

geralmente supondo informações adicionais sobre derivadas. No capítulo 5, abordamos um interessante exemplo prático, num contexto geográfico local, aplicando a teoria estudada ao longo do trabalho. As conclusões são expostas no capítulo 6. Nos apêndices estão contidos resultados e algoritmos que desenvolvemos ao longo do projeto.

Adicionalmente, deixamos no Apêndice A.4 alguns resultados bastante conhecidos, que de certa forma relacionam-se entre si, aos quais faremos menção diversas vezes ao longo do trabalho. Suas demonstrações podem ser encontradas na maioria dos textos elementares sobre cálculo/análise real.

2. APROXIMAÇÃO POLINOMIAL

Iniciamos este capítulo com alguns conceitos fundamentais, colocados a seguir.

Definição. *Diz-se que o conjunto de funções* $\{\phi_0, \ldots, \phi_n\}$ *é* ***linearmente independente*** *em* $[a, b]$ *se*

$$c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = 0 \ , \ \forall x \in [a, b],$$

implica que $c_0 = c_1 = \cdots = c_n = 0$. *Caso contrário, diz-se que o conjunto de funções é linearmente dependente.*

Teorema 2.1. *Se* $\phi_j(x)$ *for um polinômio de grau* j, *para cada* $j = 0, 1, \ldots, n$, *então* $\{\phi_0, \ldots, \phi_n\}$ *é linearmente independente em qualquer intervalo* $[a, b]$.

Demonstração:

Suponha que $c_0, \ldots, c_n$ são números reais para os quais

$$p(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = 0, \ \forall x \in [a, b].$$

Isso implica que $p(x)$ é o polinômio identicamente nulo em todo $[a, b]$. Logo, os coeficientes c_k de todas as potências de x são zero. O resultado segue.

□

Teorema 2.2. *Seja* $\prod_n$ *o conjunto de todos os polinômios de grau no máximo* n. *Se* $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ *for um conjunto de polinômios linearmente independentes em* $\prod_n$, *então qualquer polinômio em* $\prod_n$ *pode ser escrito de forma única como uma combinação linear de* $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$.

Demonstração:

Temos que $\prod_n$ é um espaço de dimensão $n+1$, afinal $\{1, x, x^2, \ldots, x^n\}$ forma uma base com $n+1$ elementos de $\prod_n$.

Da álgebra (Boldrini [4], 1980), sabemos que qualquer conjunto de elementos linearmente independentes de um espaço de dimensão finita pode ser completado de modo a formar uma base do espaço. Assim, o conjunto $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ de $n+1$ polinômios linearmente independentes forma uma base de $\prod_n$. De fato, se não formasse uma base, poderíamos completar o conjunto até formá-la e dessa forma teríamos uma base com mais do que $n+1$ elementos num espaço de dimensão $n+1$, o que é um absurdo.

Seja então $p(x) \in \prod_n$. Como $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ forma uma base de $\prod_n$, segue que $p(x)$ é escrito de maneira única como combinação linear de $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$, ou seja,

$$p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x). \qquad \square$$

A Interpolação Polinomial é uma técnica de aproximações de funções. Essa técnica tem por base aproximar uma função $f(x)$ em um intervalo $[a, b]$ por um polinômio $p_n(x)$ de grau no máximo n.

Para a construção do polinômio $p_n(x)$ em geral definimos

$$p_n(x) = \sum_{j=0}^{n} c_j \phi_j(x) \ , \tag{2.1}$$

em que c_j são coeficientes que devem ser determinados e ϕ_j são funções de base L.I. (linearmente independentes) pré-estabelecidas, polinômios de grau no máximo n.

Os métodos de interpolação polinomial podem ser aplicados para fornecer uma aproximação de uma função que ajuste os pontos conhecidos (dados discretos) , isto é, quando têm-se conhecimento de um conjunto de dados tabelados $\{(x_i, y_i) \ ; \ i = 0, 1, \ldots, n\}$ e desejamos encontrar uma função $p_n(x)$ que ajusta os valores discretos respeitando

$$p_n(x_i) = y_i, \qquad i = 0, 1, \ldots, n.$$

Também podem ser usados quando é necessário lidar com uma função "complexa", em que deseja-se calcular uma integral ou até mesmo sua derivada num intervalo. Desta forma, pode-se determinar igualmente uma função que aproxima $f(x)$, por exemplo, $p_n(x)$, que também satisfaça

$$p(x_i) = f(x_i), \qquad i = 0, 1, \ldots, n.$$

Nesse último caso, abrimos mão de um pouco de precisão em benefício da simplificação dos cálculos.

De todo modo, para que a condição $p_n(x_i) = y_i$ seja satisfeita para todo $i = 0, 1, \ldots, n$, deve-se ter o seguinte sistema linear:

$$\begin{cases} p_n(x_0) = y_0 \\ p_n(x_1) = y_1 \\ p_n(x_2) = y_2 \\ \quad \vdots \quad \vdots \\ p_n(x_n) = y_n \end{cases} \Rightarrow \begin{cases} c_0\phi_0(x_0) + c_1\phi_1(x_0) + c_2\phi_2(x_0) + \cdots + c_n\phi_n(x_0) = y_0 \\ c_0\phi_0(x_1) + c_1\phi_1(x_1) + c_2\phi_2(x_1) + \cdots + c_n\phi_n(x_1) = y_1 \\ c_0\phi_0(x_2) + c_1\phi_1(x_2) + c_2\phi_2(x_2) + \cdots + c_n\phi_n(x_2) = y_2 \\ \qquad\qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad = \ \vdots \\ c_0\phi_0(x_n) + c_1\phi_1(x_n) + c_2\phi_2(x_n) + \cdots + c_n\phi_n(x_n) = y_n \end{cases}$$

Na forma matricial o sistema pode ser escrito como:

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.2}$$

Como as funções de base $\phi_j(x)$ e os pontos (x_i, y_i) são conhecidos, resolve-se o sistema linear e encontra-se os coeficientes c_j do polinômio interpolador.

Dependendo da escolha da base $\phi_j(x)$, têm-se um tipo de sistema linear a ser resolvido. Por exemplo, a construção dos polinômios nominais, que veremos a seguir, utiliza como base a base canônica do espaço vetorial dos polinômios de grau no máximo n, obtendo a matriz de Vandermonde como matriz dos coeficientes do sistema linear. No entanto, adiantamos que independente da técnica de interpolação utilizada para interpolar uma função $f(x)$ em $n+1$ pontos distintos, o polinômio interpolador no final é sempre o mesmo.

2.1 POLINÔMIO NOMINAL

Um polinômio nominal é uma função polinomial que representa valores conhecidos em pontos específicos, mas não necessariamente corresponde a uma função analítica global. Os polinômios nominais são construídos para fornecer uma aproximação geral da função entre os pontos conhecidos.

Definição. *Seja $f(x)$ uma função, com lei de formação conhecida ou não, com valores conhecidos em $(n+1)$ pontos distintos x_i, com $i = 0, 1, 2, \ldots, n$, sendo $x_i < x_{i+1}$, para $i = 0, 1, 2, \ldots, n-1$. O polinômio, chamado* ***Polinômio Nominal****, que interpola $f(x)$ nesses $n+1$ pontos distintos é o polinômio p_n de grau no máximo n definido em $\mathbb{R}$ tal que $p_n(x_i) = f(x_i)$, com $i = 0, 1, 2, \ldots, n$, sendo:*

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

De acordo com essa definição, ao realizar diretamente a interpolação polinomial via Polinômio Nominal ($p_n(x) = \sum_{j=0}^{n} a_j x^j$) estaremos escolhendo as funções de base da forma $\phi_j(x) = x^j$, e os coeficientes $c_j = a_j$ serão fornecidos ao resolver o sistema linear associado.

Assim sendo, para obter um polinômio de grau no máximo n que interpola uma dada função $f(x)$, com lei de formação conhecida ou com valores conhecidos em determinados pontos, é necessário ter conhecimento de valores de $f(x)$ em $n+1$ pontos distintos de tal forma que pela relação $p_n(x_i) = f(x_i)$, com $i = 0, 1, 2, \ldots, n$, possamos montar um sistema linear com $n+1$ equações e $n+1$ incógnitas. E, como foi mencionado, os coeficientes do polinômio interpolador nesse caso são as incógnitas a_i do sistema linear (em forma matricial):

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} \tag{2.3}$$

Repare que o erro nos nós de interpolação será igual a zero ($| \, p_n(x_i) - f(x_i) \, | = 0$).

Mais adiante veremos que este polinômio interpolador existe e é único. Por agora, vamos visualizar um exemplo dessa técnica de aproximação polinominal.

EXEMPLO

Considere a integral:

$$\int_0^{0.9} e^{sen(2\sqrt{x})}dx$$

Observe que esta é uma integral muito difícil de resolver analiticamente. Digamos que o objetivo seja encontrar uma aproximação para essa integral, utilizando o método numérico que acabamos de descrever. Vamos então aproximar, a título de exemplo, o integrando por um polinômio de grau 3. Daí, ao invés de integrar a função original, o que deverá ser integrado é o polinômio interpolador com a finalidade de obter uma aproximação para a integral.

Comece definindo $f(x) = e^{sen(2\sqrt{x})}$. Para obter os pontos de interpolação vamos dividir o intervalo de integração em 3 subintervalos igualmente espaçados, obtendo os seguintes nós de interpolação $x_0 = 0$, $x_1 = 0.3$, $x_2 = 0.6$ e $x_3 = 0.9$. Trabalharemos com 4 casas decimais nas operações aritméticas. Obtemos então os seguintes valores de f nos pontos de interpolação:

$$\begin{aligned} f(0) &= e^{sen(2\sqrt{0})} = 1.0000 \\ f(0.3) &= e^{sen(2\sqrt{0.3})} = 2.4330 \\ f(0.6) &= e^{sen(2\sqrt{0.6})} = 2.7176 \\ f(0.9) &= e^{sen(2\sqrt{0.9})} = 2.5783 \end{aligned}$$

A partir dessas informações, o seguinte sistema linear é obtido:

$$\begin{bmatrix} 1 & 0 & 0^2 & 0^3 \\ 1 & 0.3 & 0.3^2 & 0.3^3 \\ 1 & 0.6 & 0.6^2 & 0.6^3 \\ 1 & 0.9 & 0.9^2 & 0.9^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(0) \\ f(0.3) \\ f(0.6) \\ f(0.9) \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 1.0000 & 0.3000 & 0.0900 & 0.0270 \\ 1.0000 & 0.6000 & 0.3600 & 0.2160 \\ 1.0000 & 0.9000 & 0.8100 & 0.7290 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1.0000 \\ 2.4330 \\ 2.7176 \\ 2.5783 \end{bmatrix}.$$

Utilizando o Método de Eliminação de Gauss com Pivoteamento Parcial (MEGPP) ou qualquer outro método de resolução de sistemas lineares, obtemos facilmente a solução:

$$a_0 = 1.0000, a_1 = 7.4972, a_2 = -10.4100, a_3 = 4.4759.$$

Observacão: *Uma solução mais precisa diferirá em algumas casas decimais dos valores acima, uma vez que no código que utilizamos para a resolução do sistema linear foi colocado um truncamento de apenas* 4 *casas decimais ao longo de todos os cálculos.*

Agora então, podemos integrar o polinômio $p_3(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ que aproxima a função $f(x)$:

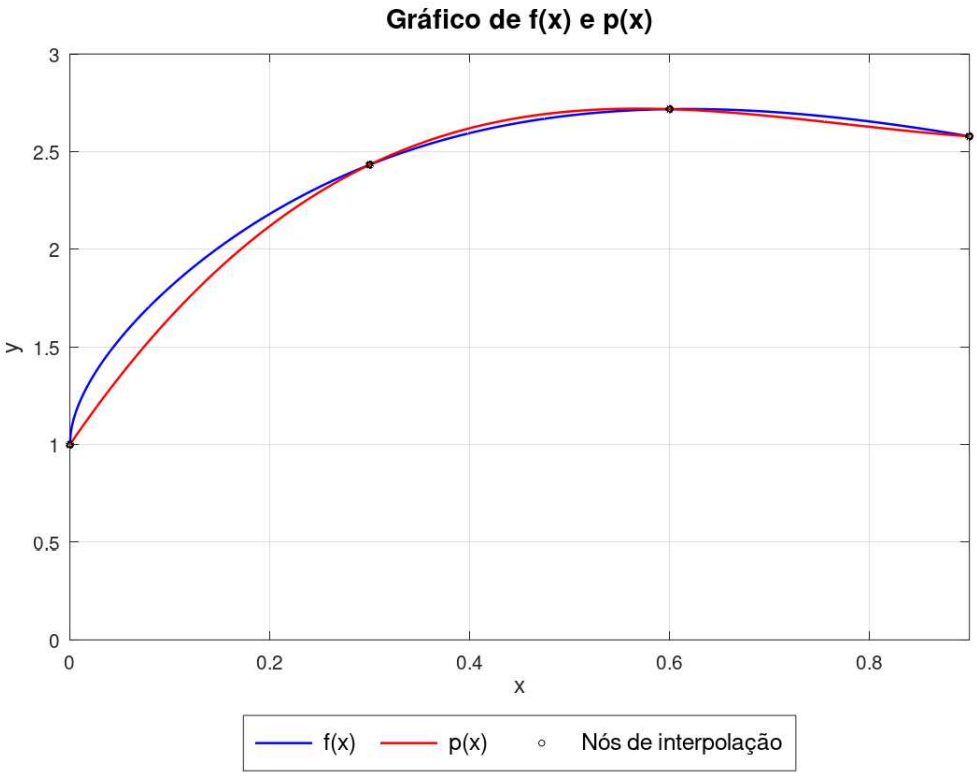
$$\int_0^{0.9} p_3(x)dx = \int_0^{0.9} 1 + 7.4972x - 10.41x^2 + 4.4759x^3dx = 2.1409.$$

Assim, obtemos finalmente que

$$\int_0^{0.9} e^{sen(2\sqrt{x})}dx \approx 2.1409$$

O valor real dessa integral é 2.1721. Perceba que, diante das condições dadas, o valor encontrado de 2.1409 é uma boa aproximação. Na Figura 2.1 é possível visualizar a comparação entre a função $f(x)$ (curva em azul) e polinômio interpolador $p_3(x)$ (curva em vermelho), dentro do intervalo de integração.

Figura 2.1: Aproximação via polinômio nominal - Comparação entre a função $f(x)$ e $p_3(x)$.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

2.2 INTERPOLAÇÃO DE LAGRANGE

De acordo com Burden [1], a fórmula de interpolação que recebeu o nome de Joseph Louis Lagrange (1736-1813) provavelmente já era conhecida por Isaac Newton por volta de 1675, mas só foi publicada pela primeira vez por Edward Waring (1736-1798) em 1779. Lagrange escreveu extensivamente sobre interpolação e seu trabalho sobre isto influenciou significativamente matemáticos posteriores. Esse resultado foi publicado por ele em 1795.

A Interpolação de Lagrange é uma técnica que nos permite obter a interpolação de n+1 pontos distintos, $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, por um polinômio de grau no máximo n (exatamente igual ao polinômio nominal), porém sem a necessidade de resolver um sistema linear.

Começamos definindo as funções de base da interpolação, $\phi_j(x)$, como os polinômios $L_j(x)$ de Lagrange, para $j = 0, 1, \ldots, n$:

$$L_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)(x - x_1) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0)(x_j - x_1) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)} \ . \tag{2.4}$$

Repare que a matriz das funções de base na Equação (2.2) será uma matriz identidade, e os coeficientes c_j serão os próprios $y_j = f(x_j)$.

A forma de Lagrange para o polinômio que interpola uma função $f(x)$ em $n + 1$ pontos distintos $x_0, x_1, x_2, \ldots, x_n$ é dada por

$$p_n(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2) + \cdots + L_n(x)f(x_n) \ . \tag{2.5}$$

Os polinômios $L_j(x)$, para $j = 0, 1, 2, \ldots, n$, funcionam como se fossem pesos no momento de avaliar $p_n(x)$ em um determinado x. Pode-se notar que

$$L_j(x_i) = \begin{cases} 0, \ i \neq j \\ 1, \ i = j \end{cases} \ . \tag{2.6}$$

Dessa forma, temos a garantia que $p_n(x_i) = f(x_i)$ em cada um dos pontos de interpolação.

EXEMPLO

Suponha que queremos obter, a partir da tabela abaixo, uma estimativa para $f(0.25)$ usando um polinômio interpolador de grau 2.

Tabela 2.1: Conjunto de dados para estimação do polinômio interpolador na forma de Lagrange.

x_i	0	0.1	0.2	0.3	0.4	0.5
$f(x_i)$	0	0.1350	0.3644	0.7379	1.3280	2.2408

Inicialmente observe que, como desejamos obter um polinômio interpolador de grau 2, devemos escolher três pontos consecutivos na vizinhança de $x = 0.25$ para tentar minimizar o erro. Como os pontos são igualmente espaçados, temos então duas possibilidade para tomar nosso conjunto de pontos, sendo que, independente da escolha o erro da aproximação será da mesma ordem de grandeza:

I)

x_i	0.1	0.2	0.3
$f(x_i)$	0.1350	0.3644	0.7379

II)

x_i	0.2	0.3	0.4
$f(x_i)$	0.3644	0.7379	1.3280

Vamos considerar os dados do segundo conjunto (II):

x_i	0.2	0.3	0.4
$f(x_i)$	0.3644	0.7379	1.3280

Calculando os polinômios $L_0(x)$, $L_1(x)$ e $L_2(x)$, obtemos

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-0.3)(x-0.4)}{(0.2-0.3)(0.2-0.4)} = \frac{x^2-0.7x+0.12}{0.02},$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-0.2)(x-0.4)}{(0.3-0.2)(0.3-0.4)} = \frac{-x^2+0.6x-0.08}{0.01},$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0.2)(x-0.3)}{(0.4-0.2)(0.4-0.3)} = \frac{x^2-0.5x+0.06}{0.02}.$$

Portanto, nosso polinômio interpolador de grau 2 na forma de Lagrange é dado por:

$$p_2(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2)$$

$$\Rightarrow p_2(x) = \left[\frac{x^2-0.7x+0.12}{0.02}\right]0.3644 + \left[\frac{-x^2+0.6x-0.08}{0.01}\right]0.7379 + \left[\frac{x^2-0.5x+0.06}{0.02}\right]1.3280.$$

Reescrevendo esse polinômio na forma geral, segue que

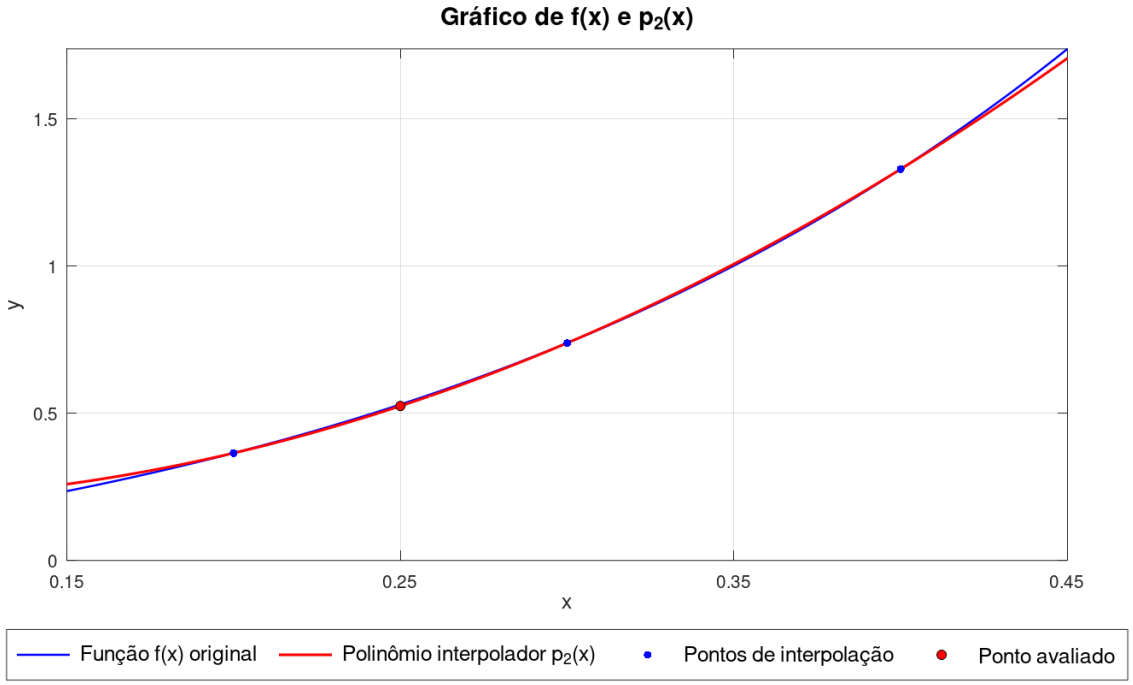
$$p_2(x) = 10.83x^2 - 1.68x + 0.2672\ .$$

Logo, nossa estimativa para $f(0.25)$ é igual a $p_2(0.25) = 0.5241$.

Os dados da Tabela 2.1 para esse problema foram obtidos a partir de $f(x) = xe^{3x}$. Com isso, a solução exata é $f(0.25) = 0.5293$ (arredondando para quatro casas decimais). Calculando $|\ f(0.25) - p_2(0.25)\ | = 0.0051$, vemos que o erro relativo foi por volta de 0.98% e que conseguimos obter uma aproximação com dois dígitos significativos de precisão. Se tivéssemos optado pelos dados do primeiro conjunto (I), a estimativa para $f(0.25)$ seria 0.5331.

Observe na figura 2.2 a comparação de $f(x)$ com a interpolação que realizamos.

Figura 2.2: Comparação entre $f(x)$ e o polinômio interpolador de Lagrange, $p_2(x)$.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

No capítulo 3 da obra de Moler [5], é detalhado como implementar em MATLAB (com um exemplo prático) uma função de interpolação baseada na forma de Lagrange, tomando como base a seguinte instrução:

```
function v = polyinterp(x, y, u)
    n = length(x);
    v = zeros(size(u));
    for k = 1:n
        w = ones(size(u));
        for j = [1:k-1, k+1:n]
            w = (u - x(j)) ./ (x(k) - x(j)) .* w;
        end
        v = v + w * y(k);
    end
end
```

No âmbito de nossas aplicações, por enquanto, não é necessário implementar nenhuma função adicional com a finalidade de realizar a interpolação polinomial, pois como veremos mais adiante, o MATLAB/GNU-Octave já dispõe de funções nativas para isso.

2.3 Interpolação de Newton

Como foi visto, a grande vantagem da forma de Lagrange em relação ao polinômio nominal é que não existe a necessidade de resolver um sistema linear para chegarmos aos coeficientes do polinômio interpolador. Contudo, o método de Lagrange para determinação do polinômio interpolador de uma função $f(x)$ sobre um conjunto de $n+1$ pontos ainda possui um inconveniente: sempre que se deseja passar de um polinômio de grau n construído sobre $n+1$ pontos para um polinômio de grau $n+k$ construído sobre $n+k+1$ pontos, todo trabalho tem que ser praticamente refeito.

Nesse sentido, como veremos, a interpolação de Newton nos concede justamente essa interessante possibilidade. Isto é, a partir de um polinômio de grau n, conseguimos passar para um polinômio, digamos de grau $n+1$, aproveitando a maioria dos cálculos já feitos do polinômio de grau n. O instrumento por trás dessa técnica é conhecido por **diferenças divididas**.

Suponha que $p_n(x)$ seja o enésimo polinômio interpolador que coincida com a função f nos pontos distintos $x_0, x_1, \ldots, x_n$ (mesmo que não tenhamos noção da expressão deste polinômio, sabemos que ele existe). Muito embora esse polinômio seja único, em determinadas situações a representação desse polinômio por formas algébricas alternativas é bastante útil. Considere a expressão de $p_n(x)$ na seguinte forma:

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\ldots(x - x_{n-1}) \ , \qquad (2.7)$$

para constantes $a_0, a_1, \ldots, a_n$ adequadas. Repare que essa expressão de $p_n(x)$ é válida e está de pleno acordo com a forma da Equação (2.1), onde $c_j = a_j$, $\phi_0(x) = 1$ e $\phi_j(x) = \prod_{i=0}^{j-1}(x - x_i)$, $j = 1, \ldots, n$. Não é difícil ver que as funções $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ definidas dessa maneira são linearmente independentes.

Observe que se dispuséssemos as funções $\phi_j(x)$ do modo como definimos acima em um sistema equivalente ao da Equação (2.2), veríamos que a matriz correspondente a essas funções de base constituem uma matriz triangular inferior, em que todos os elementos da diagonal principal são diferentes de zero. Logo, o determinante nesse caso também seria diferente de zero. Com isso, a solução desse sistema existe e é única. Isso significa que conseguiríamos resolver esse sistema linear facilmente, bastando para isso realizar sucessivas substituições (a fim de encontramos os coeficientes $c_j = a_j$). No entanto, em vez de fazermos isso, procederemos de maneira análoga ao apresentado por Burden [1] sobre essa técnica. O intuito é evidenciar a utilidade das chamadas diferenças divididas, que determina a maneira como realiza-se a interpolação na prática.

Para determinar a primeira dessas constantes, a_0, observe que se calcularmos o valor do polinômio $p_n(x)$ da Equação (2.7) no ponto $x = x_0$, restará apenas o termo a_0. Isto é,

$$a_0 = p_n(x_0) = f(x_0).$$

Da mesma forma, quando $p_n(x)$ é calculado em x_1, os únicos termos não-nulos que sobram são os termos constante e linear:

$$\begin{gathered} a_0 + a_1(x_1 - x_0) = f(x_0) + a_1(x_1 - x_0) = p_n(x_1) = f(x_1) \\ \Rightarrow a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}\ . \end{gathered}$$

Continuando, ao calcular $p_n(x)$ em $x = x_2$, temos que

$$\begin{gathered} a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\ = f(x_0) + \left(\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right)(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = p_n(x_2) = f(x_2) \\ \Rightarrow a_2 = \frac{f(x_2) - \left(\frac{f(x_1) - f_(x_0)}{x_1 - x_0}\right)(x_2 - x_0) - f(x_0)}{(x_2 - x_0)(x_2 - x_1)} \\ \Rightarrow a_2 = \frac{f(x_2) - (f(x_1) - f(x_0))\left(\frac{x_2 - x_0}{x_1 - x_0}\right) - f(x_0)}{(x_2 - x_0)(x_2 - x_1)}. \end{gathered}$$

Manipulando o termo $\dfrac{x_2 - x_0}{x_1 - x_0} = \dfrac{x_2 - x_1 + x_1 - x_0}{x_1 - x_0} = \dfrac{x_2 - x_1}{x_1 - x_0} + 1$, obtemos:

$$a_2 = \frac{f(x_2) - (f(x_1) - f(x_0))\left(\frac{x_2 - x_1}{x_1 - x_0} + 1\right) - f(x_0)}{(x_2 - x_0)(x_2 - x_1)}$$

$$\Rightarrow a_2 = \frac{f(x_2) - (f(x_1) - f(x_0))\left(\frac{x_2 - x_1}{x_1 - x_0}\right) - f(x_1) + \cancel{f(x_0)} - \cancel{f(x_0)}}{(x_2 - x_0)(x_2 - x_1)}$$

$$\Rightarrow a_2 = \frac{f(x_2) - f(x_1) - (f(x_1) - f(x_0))\left(\frac{x_2 - x_1}{x_1 - x_0}\right)}{(x_2 - x_0)(x_2 - x_1)}.$$

E finalmente, passando $(x_2 - x_1)$ para o numerador, chegamos na seguinte expressão

$$a_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}. \tag{2.8}$$

Torna-se conveniente apresentar agora a seguinte definição.

Definição (Diferenças divididas)**.** *Sejam $n+1$ pontos distintos, $x_0, x_1, \ldots, x_n$, num intervalo $[a, b]$, e sejam $f(x_0), f(x_1), \ldots, f(x_n)$ os respectivos $n+1$ valores de uma função $y = f(x)$ sobre $x = x_i$, $i = 0, 1, \ldots, n$. Definem-se indutivamente:*
As diferenças divididas de ordem zero:

$$f[x_i] = f(x_i),\ i = 0, 1, \ldots, n;$$

As primeiras diferenças divididas (diferenças divididas de ordem um) de f em relação a x_i e x_{i+1}:

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i},\ i = 0, \ldots, n-1;$$

E as k-ésimas diferenças divididas (diferenças divididas de ordem k) com relação a $x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k}$:

$$f[x_i, x_{i+1}, \ldots, x_{i+k-1}, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i},\ i = 0, \ldots, n-k.$$

O processo termina com a única enésima diferença dividida:

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0}.$$

Veja que, a partir dessa definição, e com base no que estávamos fazendo, podemos reescrever $a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$ como $a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f[x_0, x_1]$. Da mesma forma que a_0 pode ser expresso como $a_0 = f(x_0) = f[x_0]$. Assim, como se pode esperar, retomando o processo a partir de $a_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = f[x_0, x_2]$ na Equação (2.8), as constantes procuradas são:

$$a_j = f[x_0, x_1, \ldots, x_j]\ ,$$

para cada $j = 0, 1, \ldots, n$. Assim, $p_n(x)$ pode ser escrito na forma chamada **diferença dividida de Newton** como

$$p_n(x) = f[x_0] + \sum_{j=1}^{n} f[x_0, x_1, \ldots, x_j](x - x_0) \cdots (x - x_{j-1})\ . \tag{2.9}$$

Uma maneira didática de gerar diferenças divididas é esboçada na Tabela 2.2.

Tabela 2.2: Interpolação de Newton - Esquema prático do cálculo das diferenças divididas.

x	ORDEM 0	ORDEM 1	ORDEM 2	$\cdots$	ORDEM N
x_0	$\boxed{f[x_0]}\searrow$				
x_1	$f[x_1]\rightarrow$	$\boxed{f[x_0,x_1]=\frac{f[x_1]-f[x_0]}{x_1-x_0}}\searrow$			
x_2	$f[x_2]\rightarrow$	$f[x_1,x_2]=\frac{f[x_2]-f[x_1]}{x_2-x_1}\rightarrow$	$\boxed{f[x_0,x_1,x_2]}\searrow$		
x_3	$f[x_3]\rightarrow$	$f[x_2,x_3]=\frac{f[x_3]-f[x_2]}{x_3-x_2}\rightarrow$	$f[x_1,x_2,x_3]\rightarrow$		
x_4	$f[x_4]\rightarrow$	$f[x_3,x_4]=\frac{f[x_4]-f[x_3]}{x_4-x_3}\rightarrow$	$f[x_2,x_3,x_4]\rightarrow$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
x_n	$f[x_n]\rightarrow$	$f[x_{n-1},x_n]=\frac{f[x_n]-f[x_{n-1}]}{x_n-x_{n-1}}\rightarrow$	$f[x_{n-2},x_{n-1},x_n]\rightarrow$	$\cdots$	$\boxed{f[x_0,x_1,\ldots,x_n]}$

Repare que, assim como na interpolação de Lagrange, não precisamos resolver nenhum sistema linear para determinar os coeficientes do polinômio interpolador pela forma de Newton, bastando apenas calcular as diferenças divididas relacionadas.

Observe também que em nenhum momento falamos que para calcular determinada diferença dividida é necessário haver ordenação dos pontos x_i relacionados a ela. Com efeito, segundo Burden [1], pode-se provar que o valor de $f[x_0, x_1, \ldots, x_j]$ independe da ordem dos pontos $x_0, x_1, \ldots, x_j$. Formalizamos esse resultado através de uma proposição no Apêndice A.1. Além disso, se quisermos acrescentar novas informações, digamos um novo ponto a fim de elevar o grau do polinômio interpolador, podemos aproveitar boa parte dos cálculos já feitos e calcular somente os termos necessários para gerar o novo polinômio interpolador (a partir de novas diferenças divididas).

EXEMPLO

Considere a tabela abaixo e obtenha uma aproximação para $f(1.3)$ utilizando um polinômio de grau 3. Vamos trabalhar com 4 casas decimais nas operações aritméticas.

Tabela 2.3: Interpolação de Newton - Dados tabelados.

x	0.80	0.81	1.08	1.12	1.25	1.28	1.41	1.46	1.53	1.68
$f(x)$	2.5754	2.5739	2.3470	2.2801	2.0079	1.9353	1.5999	1.4714	1.3044	1.0562

Resolução: Nesse caso, para determinar um polinômio de grau 3, precisamos de quatro informações, mas essa tabela nos fornece muito mais do que isso. Devemos então considerar 4 pontos consecutivos da tabela acima de tal forma que:

- O valor de x, para o qual se quer estimar $f(x)$, deve estar entre os 4 pontos consecutivos;
- A distância entre os extremos desses 4 pontos consecutivos seja a menor possível para minimizar o erro na estimativa.

Na verdade, existem outros critérios mais rigorosos que podem ser aplicados a fim de escolhermos pontos para determinado tratamento numérico. No entanto, para nosso problema proposto, esses critérios que foram colocados são suficientes para caracterizar uma correta seleção de dados. Considere então os seguintes pontos escolhidos convenientemente com os seus respectivos valores em f:

x	1.25	1.28	1.41	1.46
$f(x)$	2.0079	1.9353	1.5999	1.4714

Montando a tabela de diferenças divididas, da mesma maneira que ilustramos na Tabela 2.2, obtemos

x	ORDEM 0	ORDEM 1	ORDEM 2	ORDEM 3
1.25	$\boxed{2.0079}$			
1.28	1.9353	$\boxed{-2.4200}$		
1.41	1.5999	-2.5800	$\boxed{-1.0000}$	
1.46	1.4714	-2.5700	0.0556	$\boxed{5.0267}$

Assim, o polinômio interpolador via fórmula de Newton sobre esses quatro pontos tabelados é dado por

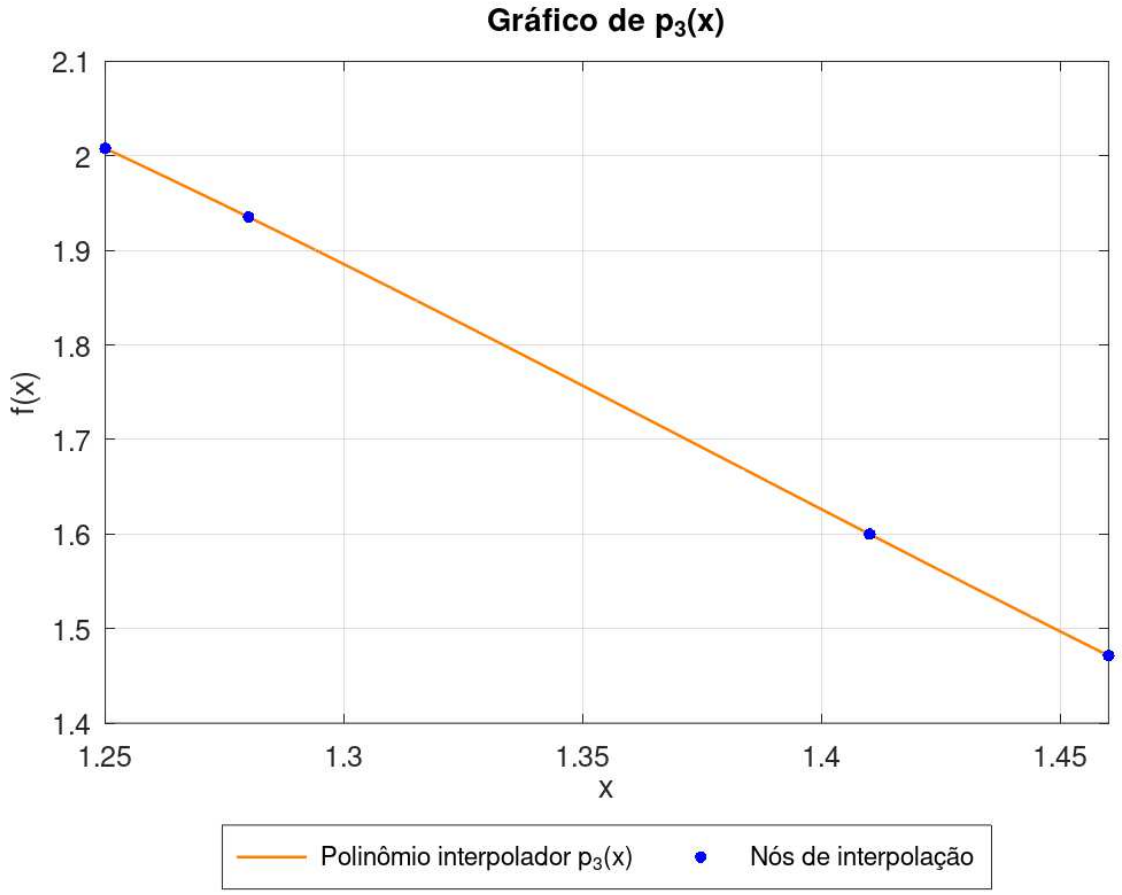
$$p_3(x) = 2.0079 - 2.42(x - 1.25) - (x - 1.25)(x - 1.28) + 5.0267(x - 1.25)(x - 1.28)(x - 1.41) \ ,$$

sendo a aproximação para $f(1.3)$ dada por

$$f(1.3) \approx p_3(1.3) = 2.0079 - 2.42(0.05) - (0.05)(0.02) + 5.0267(0.05)(0.02)(-0.11)$$
$$\Rightarrow f(1.3) \approx 1.8853.$$

Veja que o processo para obtenção do polinômio interpolador utilizando este método é bastante fácil e simples. Na Figura 2.3, exibimos o polinômio $p_3(x)$ encontrado passando pelos quatro nós de interpolação. No intervalo que tomamos para a interpolação, seu gráfico se assemelha muito com o de uma reta.

Figura 2.3: Interpolação de Newton - Polinômio interpolador de grau 3.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

2.4 EXISTÊNCIA E UNICIDADE DO POLINÔMIO INTERPOLADOR

Na Seção 2.1 chegamos a comentar sobre a existência e unicidade do polinômio interpolador, mas não provamos essa asseveração. Para esta finalidade, enunciamos e demonstramos o teorema abaixo, que foi extraído e adaptado de REAMAT [6].

Teorema 2.3 (Polinômio Nominal)**.** *Seja* $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ *um conjunto de* $n+1$ *pares ordenados de números reais tais que* $x_i \neq x_j$*, se* $i \neq j$*, então existe um único polinômio* $p_n(x)$ *de grau no máximo* n *que passa por todos os pontos dados, isto é,* $p_n(x_i) = y_i$*,* $i = 0, \ldots, n$*.*

Demonstração:

O problema de encontrar os coeficientes $a_0, a_1, \ldots, a_n$ do polinômio

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \sum_{k=0}^{n} a_k x^k$$

tal que $p_n(x_i) = y_i$ é equivalente a resolver o sistema linear com $n+1$ equações e $n+1$ incógnitas dado em forma matricial por:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.10}$$

Essa matriz à esquerda na Equação (2.10) é uma matriz de Vandermonde[1] de ordem $n+1$, cujo determinante é dado pelo produtório duplo

$$\prod_{0 \leq i < j \leq n} (x_j - x_i) \ .$$

Como $x_j \neq x_i$ sempre que $i < j$, então esse determinante é diferente de zero. Segue que a matriz envolvida é inversível e, portanto, o sistema é possível e determinado (possui solução única). Logo, existe um único polinômio $p_n(x)$ de grau no máximo n que passa por todos os pontos dados.

□

Acrescentamos também no Apêndice A.3 proposições equivalentes a esse teorema para os casos das interpolações de Lagrange e de Newton. No entanto, note que uma vez estabelecido o Teorema 2.3, tratar a existência e unicidade do polinômio interpolador para outras técnicas que assumam as mesmas condições desse teorema é um tanto quanto redundante. Isso porque se obtivermos, por qualquer que seja a técnica utilizada, um polinômio de grau no máximo n que interpole os $n+1$ pontos distintos dados, então o Teorema 2.3 nos diz que necessariamente esse polinômio é igual ao Polinômio Nominal, a menos de possíveis pequenas diferenças decimais em virtude de aproximações e propagação de erros característicos de cada método.

Ademais, para realizar em MATLAB/Octave a interpolação polinomial de um conjunto de dados podemos calcular os coeficientes do polinômio interpolador usando a função *polyfit* e depois avaliar o polinômio nos pontos de interesse usando *polyval*. A função *polyfit*, na verdade, encontra o melhor polinômio aproximador, baseado no método dos mínimos quadrados. Porém, se estamos utilizando ela para encontrar o polinômio de grau n que melhor aproxime $n+1$ pontos, esse polinômio coincidirá com o polinômio interpolador, pois o erro nos nós de interpolação é sempre zero. Também, se preferível, é possível resolver o sistema linear associado a interpolação utilizando indiretamente a função *vander*. Supletivamente, deixamos no Apêndice B.1 dois exemplos de como implementar, nos moldes do que tratamos nesse capítulo, funções auxiliares de interpolação via forma de Lagrange e de Newton.

2.5 ANÁLISE DO ERRO

Consoante Burden [1], o seguinte teorema é um resultado muito importante ao tratar de interpolação polinomial, uma vez que os polinômios de Lagrange desempenham papel signifi-

[1] Matemático francês Alexandre-Theóphile Vandermonde (1735-1796).

cativo na formulação de diversos métodos numéricos para diferenciação e integração. Por meio dele é possível calcular o resto, ou melhor, um limitante para o erro envolvido na aproximação de uma função por um polinômio interpolador.

Teorema 2.4 (Teorema do Erro)**.** *Sejam $x_0, x_1, \ldots, x_n$ números distintos no intervalo $[a, b]$ e $f \in C^{n+1}$ uma função definida em $[a, b]$. Então para cada $x \in [a, b]$, existe um número $\xi(x) \in (a, b)$ entre $\min\{x, x_0, x_1, \ldots, x_n\}$ e $\max\{x, x_0, x_1, \ldots, x_n\}$, com:*

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n) \ , \tag{2.11}$$

onde $p_n(x)$ é o polinômio interpolador dado na Equação (2.5).

Demonstração:

Observe que se $x = x_k$, para $k = 0, 1, \ldots, n$, então $f(x_k) = p_n(x_k)$, e qualquer que seja a escolha de $\xi(x_k)$ em (a, b) vale a igualdade da Equação (2.11).

Seja então $x \neq x_k$, para todo $k = 0, 1, \ldots, n$. Para $t \in [a, b]$, defina a função g por:

$$\begin{aligned} g(t) &= f(t) - p_n(t) - [f(x) - p_n(x)]\frac{(t - x_0)(t - x_1)\cdots(t - x_n)}{(x - x_0)(x - x_1)\cdots(x - x_n)} \\ &= f(t) - p_n(t) - [f(x) - p_n(x)]\prod_{i=0}^{n}\frac{(t - x_i)}{(x - x_i)} \ . \end{aligned}$$

Como $f \in C^{n+1}$ em $[a, b]$ e $p_n \in C^\infty$ em $[a, b]$, segue que $g \in C^{n+1}$ em $[a, b]$. Para $t = x_k$, temos

$$g(x_k) = f(x_k) - p_n(x_k) - [f(x) - p_n(x)]\prod_{i=0}^{n}\frac{(x_k - x_i)}{(x - x_i)} = 0 - [f(x) - p_n(x)] \cdot 0 = 0 \ .$$

Além disso,

$$g(x) = f(x) - p_n(x) - [f(x) - p_n(x)]\prod_{i=0}^{n}\frac{(x - x_i)}{(x - x_i)} = f(x) - p_n(x) - [f(x) - p_n(x)] = 0 \ .$$

Logo, $g \in C^{n+1}$ em $[a, b]$ e g vale zero nos $n + 2$ números distintos $x, x_0, x_1, \ldots, x_n$. Pelo teorema generalizado de Rolle, existe um número ξ em (a, b) para o qual $g^{(n+1)}(\xi) = 0$. Então,

$$0 = g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p_n^{(n+1)}(\xi) - [f(x) - p_n(x)]\frac{d^{n+1}}{dt^{n+1}}\left[\prod_{i=0}^{n}\frac{(t - x_i)}{(x - x_i)}\right]_{t=\xi} . \tag{2.12}$$

Como $p_n(x)$ é um polinômio de grau no máximo n, a (n+1)-ésima derivada $p_n^{(n+1)}(x)$ é identicamente nula. Além disso, $\prod_{i=0}^{n}\left[\frac{(t-x_i)}{(x-x_i)}\right]$ é um polinômio de grau $(n + 1)$. Então temos

$$\prod_{i=0}^{n}\frac{(t - x_i)}{(x - x_i)} = \left[\frac{1}{\prod_{i=0}^{n}(x - x_i)}\right] t^{n+1} + \textit{(termos de grau inferior em } t\textit{)},$$

e

$$\frac{d^{n+1}}{dt^{n+1}} \prod_{i=0}^{n} \frac{(t-x_i)}{(x-x_i)} = \frac{(n+1)!}{\prod_{i=0}^{n}(x-x_i)} \ .$$

A Equação (2.12) agora se transforma em

$$0 = f^{(n+1)}(\xi) - 0 - [f(x) - p_n(x)] \frac{(n+1)!}{\prod_{i=0}^{n}(x-x_i)} \ ,$$

e, ao isolar $f(x)$, obtemos finalmente

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n}(x-x_i) \ .$$

□

Apesar de garantida a existência do ponto $\xi(x)$ por este teorema, na prática não conseguimos determiná-lo. No entanto, a partir da Equação (2.11) no referido teorema, conseguimos calcular o limitante do erro para saber o erro máximo que é possível cometer em determinada interpolação. Neide Franco [7] trata isso como corolário do último teorema, e assim fazemos também:

Corolário (Limitante do Erro)**.** *Seja* $E_n(x) = f(x) - p_n(x)$*. Se* $f \in C^{n+1}$ *é uma função definida em* $[a, b]$*, então:*

$$| \ E_n(x) \ | = | \ f(x) - p_n(x) \ | \leq \ \frac{| \ \prod_{i=0}^{n}(x-x_i) \ |}{(n+1)!} M_{n+1} \ , \tag{2.13}$$

sendo $M_{n+1} = \max_{a \leq x \leq b} | \ f^{(n+1)}(x) \ |$*. Chamamos*

$$\mathcal{L}_n(x) = \frac{| \ \prod_{i=0}^{n}(x-x_i) \ |}{(n+1)!} M_{n+1}$$

de ***limitante do erro****.*

Note que a expressão 2.13 está em conformidade com o fato de que nos nós de interpolação o erro sempre é zero. Isto é, $| \ E_n(x_i) \ | = 0$, para todo $i = 0, 1, \ldots, n$.

Ainda, de acordo com Quarteroni [8], quando estivermos diante de uma distribuição uniformemente ordenada de nós, isto é, quando dados x_0 e $h > 0$ tivermos $x_{i+1} = x_i + h$, com $i = 0, \ldots, n-1$, é possível mostrar que $\forall x \in (x_0, x_n)$ temos

$$| \ E_n(x) \ | \leq \ \frac{h^{n+1}}{4(n+1)} M_{n+1} \ , \tag{2.14}$$

onde $M_{n+1} = \max_{a \leq x \leq b} | \ f^{(n+1)}(x) \ |$. Isso simplifica a Equação (2.13).

EXEMPLO

Considere a seguinte tabela de valores para a função $f(x) = e^{-x^2}$ nos pontos $x_0 = 0$, $x_1 = 0.3$, $x_2 = 0.7$ e $x_3 = 1.0$. Vamos obter uma aproximação para $f(0.5)$ utilizando um polinômio de grau 3. Trabalharemos com 5 casas decimais.

Tabela 2.4: Nós de interpolação para fins de análise do erro.

x_i	0	0.3	0.7	1
$f(x_i)$	1	0.91393	0.61263	0.36788

Podemos utilizar a forma de Lagrange para obtenção do polinômio. Calculando as funções $L_k(x)$, temos:

$$L_0(x) = \frac{(x-0.3)(x-0.7)(x-1)}{(0-0.3)(0-0.7)(0-1)} = \frac{x^3-2x^2+1.21x-0.21}{-0.21}$$

$$L_1(x) = \frac{(x-0)(x-0.7)(x-1)}{(0.3-0)(0.3-0.7)(0.3-1)} = \frac{x^3-1.7x^2+0.7x}{0.084}$$

$$L_2(x) = \frac{(x-0)(x-0.3)(x-1)}{(0.7-0)(0.7-0.3)(0.7-1)} = \frac{x^3-1.3x^2+0.3x}{-0.084}$$

$$L_3(x) = \frac{(x-0)(x-0.3)(x-0.7)}{(1-0)(1-0.3)(1-0.7)} = \frac{x^3-x^2+0.21x}{0.21}.$$

Assim, o polinômio interpolador de grau 3 é dado por

$$p_3(x) = \left[\frac{x^3-2x^2+1.21x-0.21}{-0.21}\right](1) + \left[\frac{x^3-1.7x^2+0.7x}{0.084}\right](0.91393)$$
$$+ \left[\frac{x^3-1.3x^2+0.3x}{-0.084}\right](0.61263) + \left[\frac{x^3-x^2+0.21x}{0.21}\right](0.36788).$$

Rearranjando os termos:

$$p_3(x) = 0.57682x^3 - 1.24303x^2 + 0.03409x + 1 \ .$$

Avaliando $p_3(0.5) = 0.57682(0.5)^3 - 1.24303(0.5)^2 + 0.03409(0.5) + 1 = 0.77839$ encontramos nossa aproximação desejada. O valor real de $f(0.5)$ é $f(0.5) = e^{-(0.5)^2} = 0.77880$. O erro que cometemos nesse caso foi

$$| \ E(0.5) \ | = | \ f(0.5) - p_3(0.5) \ | = | \ 0.77880 - 0.77839 \ | = 0.00041 \ .$$

Vamos mostrar que esse erro cometido na interpolação para avaliar $f(0.5)$ usando $p_3(0.5)$ não ultrapassa o limitante do erro. Com efeito, temos que

$$\mathcal{L}_3(0.5) = \frac{| \ \prod_{i=0}^{3}(x-x_i) \ |}{4!} M_4 \ .$$

Para determinar $M_4 = \max\limits_{0 \le x \le 1} | \ f^{(4)}(x) \ |$, precisamos calcular a quarta derivada de $f(x) = e^{-x^2}$:

- $f^{(0)}(x) = e^{-x^2}$
- $f^{(1)}(x) = e^{-x^2}(-2x)$
- $f^{(2)}(x) = e^{-x^2}(-2x)(-2x) + e^{-x^2}(-2) = e^{-x^2}(4x^2 - 2)$
- $f^{(3)}(x) = e^{-x^2}(-2x)(4x^2 - 2) + e^{-x^2}(8x) = e^{-x^2}(-8x^3 + 12x)$
- $f^{(4)}(x) = e^{-x^2}(-2x)(-8x^3 + 12x) + e^{-x^2}(-24x^2 + 12) = e^{-x^2}(16x^4 - 48x^2 + 12)$

Para sabermos o valor máximo que $\mid f^{(4)}(x) \mid$ assume no intervalo $[0, 1]$ ainda precisamos avaliá-la em seus pontos críticos e nos extremos do intervalo. Para encontrar os pontos críticos de $f^{(4)}(x)$ derivamos mais uma vez:

$$\frac{df^{(4)}}{dx} = f^{(5)}(x) = e^{-x^2}(-2x)(16x^4 - 48x^2 + 12) + e^{-x^2}(64x^3 - 96x) = e^{-x^2}(-32x^5 + 160x^3 - 120x)$$

e igualamos a zero:

$$\begin{aligned} & e^{-x^2}(-32x^5 + 160x^3 - 120x) = 0 \\ \iff & -32x^5 + 160x^3 - 120x = 0 \end{aligned}$$

Para encontrar as raízes desse polinômio à esquerda da igualdade pode-se utilizar qualquer método numérico de sua preferência (método de Newton, método da bissecção, entre outros) ou até mesmo abaixando o grau do polinômio e resolvendo analiticamente (uma pequena tarefa enfadonha). Aqui temos que as duas raízes desse polinômio no intervalo $[0, 1]$ são $x' = 0$ e $x'' = 0.95857$. Avaliando, portanto, $\mid f^{(4)}(x) \mid$ nesses dois pontos críticos e no extremo $x = 1$, temos:

i) $\mid f^{(4)}(0) \mid = \mid e^0(16 \cdot 0 - 48 \cdot 0 + 12) \mid = \mid 12 \mid = 12$

ii) $\mid f^{(4)}(0.95857) \mid = \mid e^{-0.95857^2}[16(0.95857)^4 - 48(0.95857)^2 + 12] \mid = \mid -7.41948 \mid = 7.41948$

iii) $\mid f^{(4)}(1) \mid = \mid e^0(16 \cdot 1 - 48 \cdot 1 + 12) \mid = \mid -7.35759 \mid = 7.35759$

Dessa forma, temos que $\max\limits_{0 \leq x \leq 1} \mid f^{(4)}(x) \mid = 12$. Exibimos na Figura 2.4 o comportamento de $f^{(4)}(x)$ no intervalo $[0, 1]$.

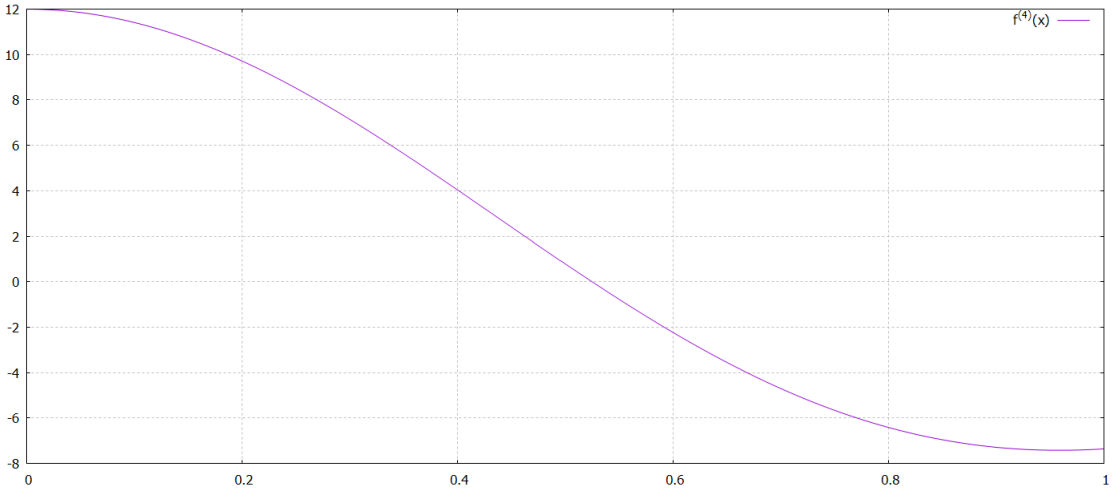
Enfim, o limitante do erro é dado por

$$\begin{aligned} \mathcal{L}_3(0.5) &= \frac{\mid (0.5 - 0)(0.5 - 0.3)(0.5 - 0.7)(0.5 - 1) \mid}{4 \cdot 3 \cdot 2 \cdot 1} \cdot (12) \\ &= 0.00042 \cdot 12 \\ &= 0.00504 \ . \end{aligned}$$

Assim, verificamos que $\mid E(0.5) \mid = 0.00041 < 0.00504 = \mathcal{L}_3(0.5)$. Observe ainda que o erro relativo nessa aproximação foi de $\frac{0.00041}{0.7788} \approx 0.00053$, que corresponde a 0.053%. Na Figura 2.5 é possível ver a sobreposição das funções $f(x)$ e $p_3(x)$ no intervalo $[0, 1]$. Note que o polinômio interpolador ajusta tão perfeitamente a função que praticamente não é possível distinguir o traço

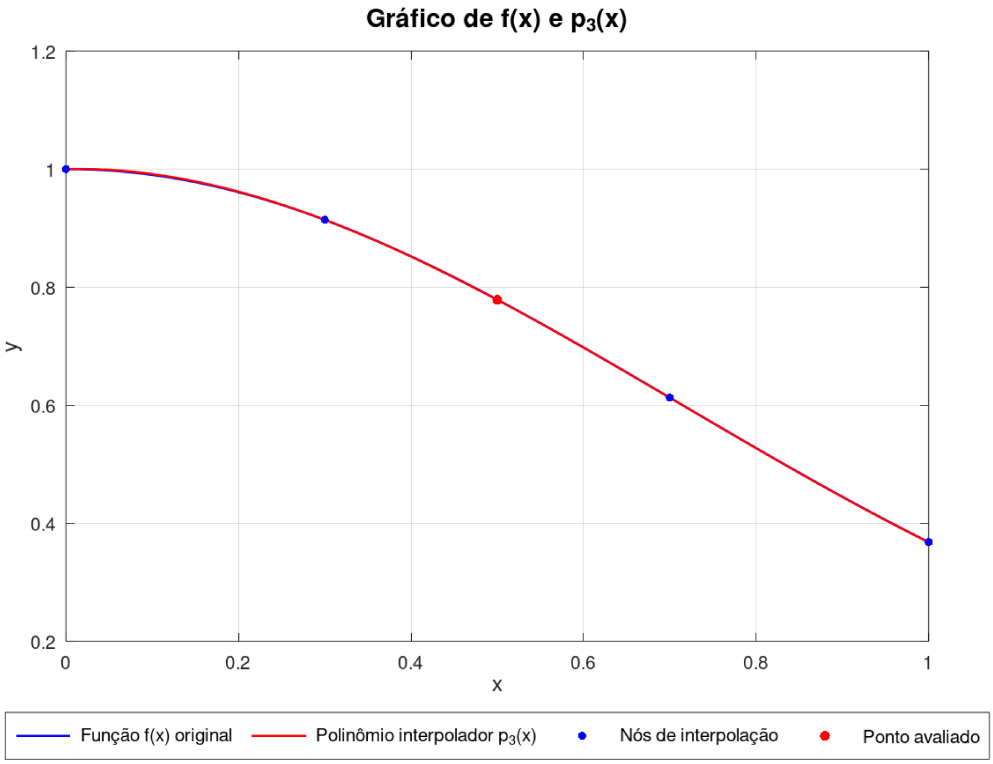
das duas funções nesse intervalo. Na Figura 2.6 demos apenas um "zoom" em torno do ponto avaliado para percebermos a sutil diferença entre a função dada e o polinômio interpolador.

Figura 2.4: Análise do erro - Gráfico de $f^{(4)}(x)$.



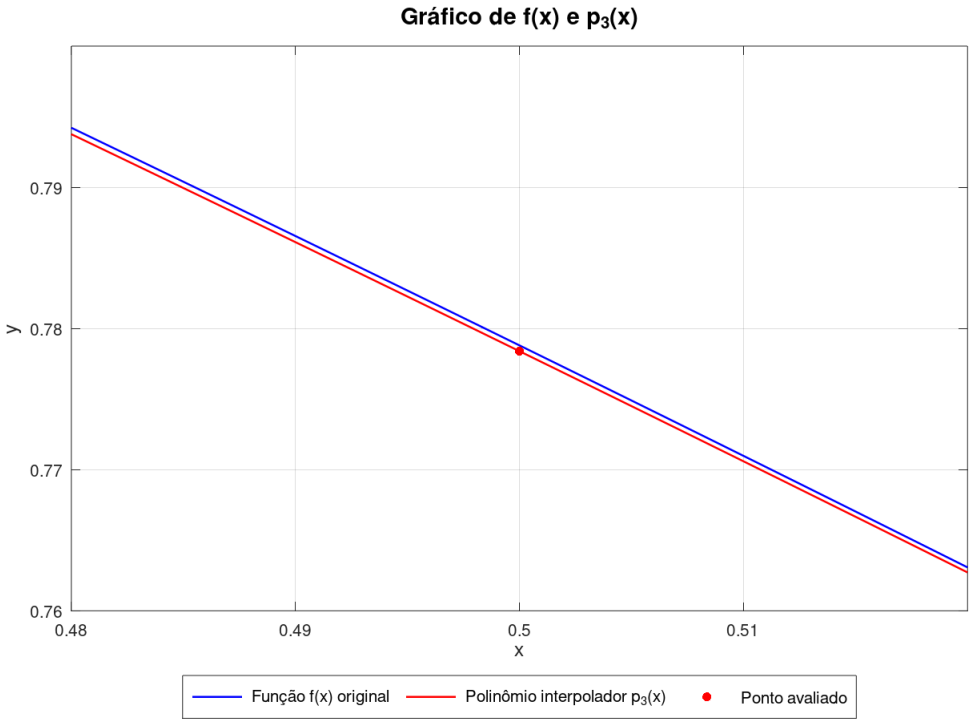
Fonte: De autoria própria (elaborado através do gnuplot v5.4).

Figura 2.5: Análise do erro - A função dada e sua aproximação por $p_3(x)$.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

Figura 2.6: Análise do erro - Ampliação em torno do ponto avaliado.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

No próximo capítulo, veremos uma situação em que a interpolação polinomial experimenta fenômenos oscilatórios que dificultam o processo de encontrar um bom polinômio aproximador, mostrando que a ideia intuitiva de aumentar o número de nós de interpolação para obter uma melhor aproximação é equivocada.

3. POLINÔMIOS DE CHEBYSHEV

Ocorre a falsa impressão de que se agregarmos mais valores de x e $f(x)$ em um intervalo $[a, b]$ para realizar uma interpolação polinomial, iremos obter um polinômio cada vez mais próximo da função original. Infelizmente, em geral, isso não é verdade. Não podemos deduzir nem mesmo da Equação (2.14) que o erro tende para 0 quando $n \to \infty$, apesar de $\frac{h^{n+1}}{4(n+1)}$ tender para 0. Na verdade, existem funções f para as quais o limite do erro pode ser infinito, isto é:

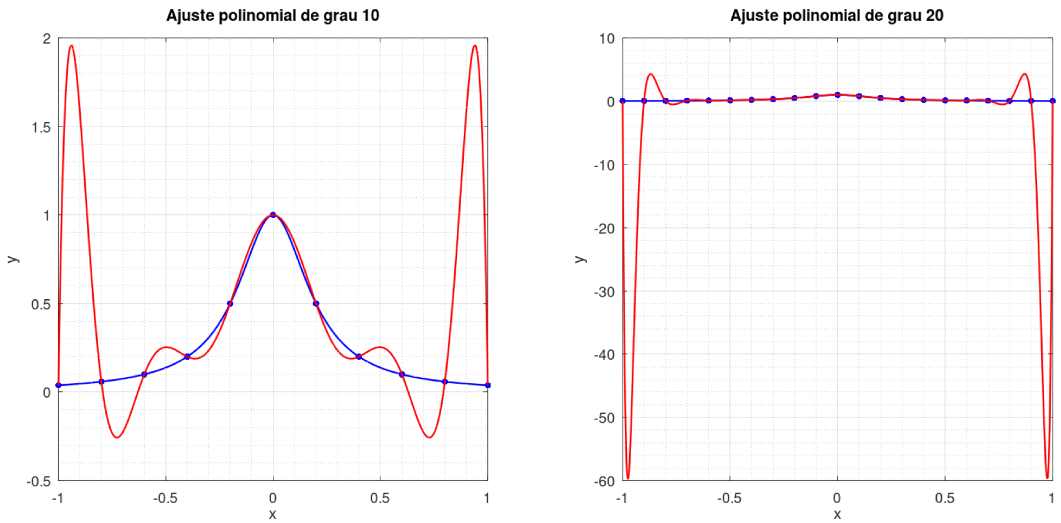
$$\lim_{n\to\infty} \mathcal{L}_n(x) = \infty \ .$$

Essa situação é conhecida como **Fenômeno de Runge**. Para ilustrar isso, considere o problema de interpolar a função

$$f(x) = \frac{1}{1 + 25x^2}$$

no intervalo $[-1, 1]$ utilizando 11 pontos igualmente espaçados. Nesse caso, temos $x_i = -1 + ih$, com $i = 0, 1, 2, \ldots, 10$ e $h = 0.2$. Na figura 3.1, do lado esquerdo, mostramos essa interpolação com 11 pontos. A função $f(x)$ é a curva na cor azul, enquanto o polinômio interpolador é mostrado na cor vermelha. Perceba como nos extremos do intervalo o ajuste polinomial diverge muito da função $f(x)$. Do lado direito, realizamos a interpolação polinomial com a mesma função no intervalo $[-1, 1]$, mas dessa vez adicionando mais 10 pontos, totalizando 21 pontos uniformemente espaçados. Nesse caso, a amplitude do erro aumenta significativamente.

Figura 3.1: Polinômios de Chebyshev - Fenômeno de Runge.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

Introduziremos a seguir os polinômios de Chebyshev[1] e mostraremos que eles satisfazem propriedades de ortogonalidade. Os polinômios de Chebyshev podem ser utilizados, como veremos posteriormente, para atenuar o problema do fenômeno de Runge que foi descrito.

Definição. *Uma função integrável w é chamada* ***função peso*** *em um intervalo I se $w(x) \geq 0$, para todo $x \in I$, porém $w(x) \not\equiv 0$ em qualquer subintervalo de I.*

A definição acima segue a mesma dada por Burden [1]. Segundo ele, a finalidade de uma função peso é atribuir graus variados de importância a aproximações em certas partes do intervalo.

Tanto a função peso quanto a noção de ortogonalidade, que veremos a seguir, na verdade, nascem em um outro contexto (de mínimos quadrados) que não detalharemos aqui, ao tentar minimizar o erro da aproximação de funções por polinômios (uma abordagem diferente da interpolação). Outrossim, a título de curiosidade, o problema da aproximação por mínimos quadrados é muito simplificado quando as funções $\phi_0, \phi_1, \ldots, \phi_n$ são escolhidads de modo a satisfazerem as condições de *ortogonalidade*.

Definição (Ortogonalidade)**.** *Diz-se que $\{\phi_0, \phi_1, \ldots, \phi_n\}$ é um* ***conjunto de funções ortogonais*** *no intervalo $[a, b]$, com relação à função peso w, se*

$$\int_a^b w(x)\phi_k(x)\phi_j(x) = \begin{cases} 0, & j \neq k \\ \alpha_j > 0, & j = k \end{cases}$$

Se, além disso, $\alpha_j = 1$, $\forall j = 0, 1, \ldots, n$, o conjunto é dito ***ortonormal****.*

A quem interessar, podem ser encontrados também em Burden [1] alguns teoremas decorrentes dessa definição e um interessante estudo (vale alertar que existem alguns erros mesmo na edição mais recente até este trabalho) sobre conjuntos ortogonais, em particular *polinômios de Legendre*, cujas raízes são utilizadas como nós na quadratura de Gauss. Em certo sentido, as funções ortogonais são perpendiculares umas às outras, afinal a palavra "ortogonal" significa "com ângulo reto".

Nos concentremos agora numa maneira recursiva de determinar os **Polinômios de Chebyshev**.

Para $x \in [-1, 1]$, defina

$$T_n(x) = cos[n\ arccos\ x],\ \forall n \geq 0. \tag{3.1}$$

Esta é a definição geral do enésimo polinômio de Chebyshev. Na prática, é difícil de enxergar $T_n(x)$ definido dessa maneira como um polinômio em x. Mas, observe que $T_0(x) = cos\ 0 = 1$ e $T_1(x) = cos(arccos\ x) = x$.

Para $n \geq 1$, se introduzirmos a substituição $\theta = arccos\ x$, mudamos a Equação (3.1) para:

[1] Pafnuty Lvovich Chebyshev (1821-1894) realizou notáveis contribuições em diversas áreas da matemática, abrangendo matemática aplicada, probabilidade e teoria dos números. Chebyshev foi o primeiro a perceber as importantes consequências dos estudos de polinômios ortogonais. Ele desenvolveu os polinômios de Chebyshev para explorar a aproximação por mínimos quadrados e questões de probabilidade, aplicando seus resultados em problemas de interpolação, quadratura aproximada e outras áreas.

$$T_n(\theta(x)) \equiv T_n(\theta) = cos(n\theta),\ \theta \in [0, \pi]\ .$$

Veja que

$$T_{n+1}(\theta) = cos((n+1)\theta) = cos\ \theta\ cos(n\theta) - sen\ \theta\ sen(n\theta)$$

e

$$T_{n-1}(\theta) = cos((n-1)\theta) = cos\ \theta\ cos(n\theta) + sen\ \theta\ sen(n\theta).$$

Somando essas duas equações, obtemos

$$T_{n+1}(\theta) = 2\ cos\ \theta\ cos(n\theta) - T_{n-1}(\theta)\ .$$

Retornando à variável $x = cos\ \theta;\ n \geq 1$:

$$T_{n+1}(x) = 2x\ cos(n\ arccos\ x) - T_{n-1}(x)$$

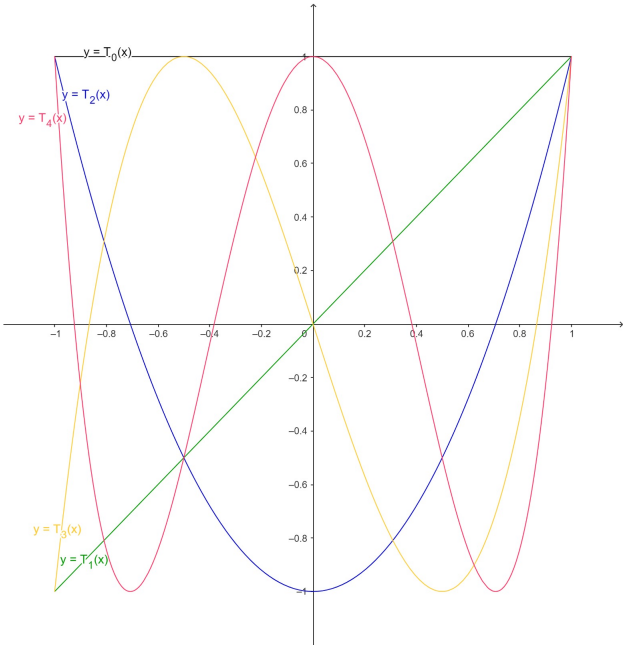
$$\Rightarrow \boxed{T_{n+1}(x) = 2x\ T_n(x) - T_{n-1}(x)}\ ,\ \forall n \geq 1. \tag{3.2}$$

Estamos diante de uma relação de recorrência. Assim, temos:

$$\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_2(x) &= 2x\ T_1(x) - T_0(x) = 2x^2 - 1 \\
T_3(x) &= 2x\ T_2(x) - T_1(x) = 4x^3 - 3x \\
T_4(x) &= 2x\ T_3(x) - T_2(x) = 8x^4 - 8x^2 + 1 \\
&\vdots
\end{aligned}$$

A relação de recorrência também implica que, quando $n \geq 1$, $T_n(x)$ é um polinômio de grau n, com coeficiente líder 2^{n-1}. Exibimos os gráficos de T_0, T_1, T_2, T_3 e T_4 na Figura 3.2.

Figura 3.2: Polinômios de Chebyshev - Os cinco primeiros polinômios de Chebyshev.



Fonte: De autoria própria (elaborado através do GeoGebra Classic v6.0).

Proposição 3.1. *Os polinômios de Chebyshev $\{T_n(x)\}$ são ortogonais em $[-1,1]$, com relação à função peso $w(x) = (1-x^2)^{-1/2}$.*

Demonstração:

Considere

$$\int_{-1}^{1} \frac{T_n(x)\ T_m(x)}{\sqrt{1-x^2}}dx = \int_{-1}^{1} \frac{cos(n\ arccos\ x)\ cos(m\ arccos\ x)}{\sqrt{1-x^2}}dx\ .$$

Ao reintroduzir $\theta = arccos\ x$, obtemos

$$d\theta = -\frac{1}{\sqrt{1-x^2}}dx$$

e

$$\int_{-1}^{1} \frac{T_n(x)\ T_m(x)}{\sqrt{1-x^2}}dx = -\int_{\pi}^{0} cos(n\theta)\ cos(m\theta)\ d\theta = \int_{0}^{\pi} cos(n\theta)\ cos(m\theta)\ d\theta\ .$$

Suponha que $n \neq m$. Pela fórmula de adição do cosseno, temos que

$$\begin{aligned} cos((n+m)\theta) = cos(n\theta + m\theta) &= cos(n\theta)\ cos(m\theta) - sen(n\theta)\ sen(m\theta) \\ \Rightarrow cos(n\theta)\ cos(m\theta) &= cos((n+m)\theta) + sen(n\theta)\ sen(m\theta) \quad \textbf{(I)} \end{aligned}$$

Por outro lado:

$$\begin{aligned} cos((n-m)\theta) = cos(n\theta - m\theta) &= cos(n\theta)\ cos(m\theta) + sen(n\theta)\ sen(m\theta) \\ \Rightarrow cos(n\theta)\ cos(m\theta) &= cos((n-m)\theta) - sen(n\theta)\ sen(m\theta) \quad \textbf{(II)} \end{aligned}$$

Somando **(I)** e **(II)**, obtemos

$$\begin{aligned} 2\ cos(n\theta)\ cos(m\theta) &= cos((n+m)\theta) + cos((n-m)\theta) \\ \Rightarrow cos(n\theta)\ cos(m\theta) &= \frac{1}{2}[cos((n+m)\theta) + cos((n-m)\theta)]\ . \end{aligned}$$

Daí,

$$\begin{aligned} \int_{-1}^{1} \frac{T_n(x)\ T_m(x)}{\sqrt{1-x^2}}dx &= \frac{1}{2}\int_{0}^{\pi} cos((n+m)\theta)\ d\theta + \frac{1}{2}\int_{0}^{\pi} cos((n-m)\theta)\ d\theta \\ &= \left[\frac{1}{2(n+m)}sen((n+m)\theta) + \frac{1}{2(n-m)}sen((n-m)\theta)\right]_{0}^{\pi} = 0\ . \end{aligned}$$

Analogamente, suponha que $n = m$. Então

$$\int_{-1}^{1} \frac{[T_n(x)]^2}{\sqrt{1-x^2}}dx = \int_{-1}^{1} \frac{cos^2(n\ arccos\ x)}{\sqrt{1-x^2}}dx = -\int_{\pi}^{0} cos^2(n\theta)\ d\theta = \int_{0}^{\pi} cos^2(n\theta)\ d\theta\ .$$

Usando a fórmula de arco metade:

$$\int_{0}^{\pi} cos^2(n\theta)\ d\theta = \frac{1}{2}\int_{0}^{\pi} (1 + cos(2n\theta))\ d\theta = \frac{1}{2}\int_{0}^{\pi} 1\ d\theta + \frac{1}{2}\int_{0}^{\pi} cos(2n\theta)\ d\theta\ .$$

Se $n = 0$, então

$$\int_{-1}^{1} \frac{[T_n(x)]^2}{\sqrt{1-x^2}} dx = \frac{1}{2}\int_0^{\pi} 1 \; d\theta + \frac{1}{2}\int_0^{\pi} 1 \; d\theta = \left[\frac{\theta}{2} + \frac{\theta}{2}\right]_0^{\pi} = \pi > 0 \; .$$

Se $n \geq 1$, então

$$\int_{-1}^{1} \frac{[T_n(x)]^2}{\sqrt{1-x^2}} dx = \frac{1}{2}\int_0^{\pi} 1 \; d\theta + \frac{1}{2}\int_0^{\pi} cos(2n\theta) \; d\theta = \left[\frac{\theta}{2} + \frac{1}{4n} sen(2n\theta)\right]_0^{\pi} = \frac{\pi}{2} > 0 \; .$$

□

É justamente pela propriedade de serem ortogonais, demonstrada nessa proposição, que os polinômios de Chebyshev podem ser utilizados para calcular efetivamente um polinômio que aproxime por mínimos quadrados uma função f qualquer no intervalo $[-1, 1]$ com relação ao peso $w(x) = (1 - x^2)^{-1/2}$. Sem se aprofundar nessa discussão, podemos mencionar ainda que eles fornecem uma maneira de reduzir o grau de um polinômio de aproximação com o mínimo de perda de precisão. No entanto, o principal motivo de trazermos essa abordagem sobre polinômios de Chebyshev é que através deles conseguimos também obter um ótimo posicionamento de pontos interpoladores para minimizar o erro na interpolação de Lagrange, sobretudo, amenizando o fenômeno de Runge observado no começo deste capítulo. É sobre isso que trata a seção seguinte.

3.1 PONTOS DE CHEBYSHEV

Damos sequência ao nosso estudo, fundamentado em Burden [1], enunciando dois resultados cruciais e, em seguida, explicamos como eles respondem à questão da minimização do erro na interpolação polinomial.

Teorema 3.1. *O polinômio de Chebyshev $T_n(x)$, de grau $n \geq 1$, tem n zeros (raízes simples) em $[-1, 1]$, a saber, nos pontos*

$$\overline{x}_k = cos\left(\frac{2k-1}{2n}\pi\right), \; para \; cada \; k = 1, 2, \ldots, n.$$

Além disso, $T_n(x)$ assume seus extremos absolutos em

$$\overline{x}'_k = cos\left(\frac{k\pi}{n}\right) \; com \; T_n(\overline{x}'_k) = (-1)^k, \; para \; cada \; k = 0, 1, \ldots, n.$$

Demonstração:

Uma vez que os $\overline{x}_k$ são todos distintos e $T_n(x)$ é um polinômio de grau n, para mostrar a primeira parte do teorema é suficiente então que tenhamos $T_n(\overline{x}_k) = 0$, $\forall k = 1, 2, \ldots, n$. De fato, se $\overline{x}_k = cos\left(\frac{2k-1}{2n}\pi\right)$, $k = 1, 2, \ldots, n$, então

$$T_n(\overline{x}_k) = cos(n \; arccos \; \overline{x}_k) = cos\left(n \; arccos\left(cos\left(\frac{2k-1}{2n}\pi\right)\right)\right) = cos\left(\frac{2k-1}{2}\pi\right) = 0 \; .$$

Para a segunda parte, observe primeiro que

$$T_n'(x) = \frac{d}{dx}[cos(n\ arccos\ x)] = \frac{n\ sen(n\ arccos\ x)}{\sqrt{1-x^2}}$$

e que, quando $k = 1, 2, \ldots, n-1$, temos

$$T_n'(\overline{x}_k') = \frac{n\ sen\left(n\ arccos\left(cos\left(\frac{k\pi}{n}\right)\right)\right)}{\sqrt{1-\left[cos\left(\frac{k\pi}{n}\right)\right]^2}} = \frac{n\ sen(k\pi)}{sen\left(\frac{k\pi}{n}\right)} = 0\ .$$

Como $T_n(x)$ é um polinômio de grau n, então sua derivada $T_n'(x)$ é um polinômio de grau $(n-1)$ e, portanto, todos os zeros de $T_n'(x)$ devem ocorrer nesses $n-1$ pontos distintos. As únicas outras possibilidades para valores extremos de $T_n(x)$ ocorrem nas extremidades do intervalo $[-1, 1]$; ou seja, em $\overline{x}_0' = 1$ e em $\overline{x}_n' = -1$.

Por fim, para cada $k = 0, 1, \ldots, n$, temos

$$T_n'(\overline{x}_k') = cos\left(n\ arccos\left(cos\left(\frac{k\pi}{n}\right)\right)\right) = cos(k\pi) = (-1)^k\ .$$

Ocorre, então, um máximo em cada valor par de k e um mínimo em cada valor ímpar. □

Obtemos os **polinômios mônicos de Chebyshev** (polinômios com coeficiente líder 1) $\tilde{T}_n(x)$ dividindo os polinômios de Chebyshev $T_n(x)$ por 2^{n-1}, quando $n \geq 1$. Isto é,

$$\tilde{T}_0(x) = 1;\ \tilde{T}_n(x) = \frac{1}{2^{n-1}}\ T_n(x),\ \forall n \geq 1. \tag{3.3}$$

Assim, note que $\tilde{T}_0(x) = T_0(x)$ e $\tilde{T}_1(x) = T_1(x)$. Pela fórmula acima e pela relação de recorrência da Equação (3.2), temos que

$$\begin{aligned}\tilde{T}_2(x) &= \frac{1}{2}\ T_2(x)\\ &= \frac{1}{2}(2x\ T_1(x) - T_0(x))\\ &= x\ T_1(x) - \frac{1}{2}\ T_0(x)\\ &= x\ \tilde{T}_1(x) - \frac{1}{2}\ \tilde{T}_0(x)\end{aligned}$$

e

$$\begin{aligned}\tilde{T}_{n+1}(x) &= \frac{1}{2^n}\ T_{n+1}(x)\\ &= \frac{1}{2^n}(2x\ T_n(x) - T_{n-1}(x))\\ &= \frac{1}{2^{n-1}}\ x\ T_n(x) - \frac{1}{2^n}\ T_{n-1}(x)\end{aligned}$$

$$\Rightarrow \boxed{\tilde{T}_{n+1}(x) = x\ \tilde{T}_n(x) - \frac{1}{4}\ \tilde{T}_{n-1}(x)}\ ,\ \forall n \geq 2. \tag{3.4}$$

Como $\tilde{T}_n(x)$ é apenas múltiplo por constante de $T_n(x)$, então, pelo Teorema 3.1, os zeros de $\tilde{T}_n(x)$, $n \geq 1$, também ocorrem em

$$\overline{x}_k = cos\left(\frac{2k-1}{2n}\pi\right), \text{ para cada } k = 1, 2, \ldots, n,$$

e os valores extremos de $\tilde{T}_n(x)$, $n \geq 1$, ocorrem em

$$\overline{x}'_k = cos\left(\frac{k\pi}{n}\right);\ \tilde{T}_n(\overline{x}'_k) = \frac{(-1)^k}{2^{n-1}},\ \forall k = 0, 1, \ldots, n. \tag{3.5}$$

Denotemos por $\tilde{\prod}_n$ o conjunto de todos os polinômios mônicos de grau n. A relação expressa na Equação (3.5) conduz a uma importante propriedade de minimização, enunciada através do próximo teorema, que distingue $\tilde{T}_n(x)$ de outros polinômios em $\tilde{\prod}_n$.

Teorema 3.2. *Os polinômios da forma $\tilde{T}_n(x)$, quando $n \geq 1$, têm a propriedade que*

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\ \tilde{T}_n(x)\ | \leq \max_{x\in[-1,1]} |\ P_n(x)\ |\ ,\ \forall P_n(x) \in \tilde{\prod}_n\ .$$

Além disso, a igualdade ocorre se, e somente se, $P_n \equiv \tilde{T}_n$.

Demonstração:

Suponha por absurdo que $\exists\ P_n(x) \in \tilde{\prod}_n$, $P_n \not\equiv \tilde{T}_n$, tal que

$$\max_{x\in[-1,1]} |\ P_n(x)\ | < \frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\ \tilde{T}_n(x)\ |\ .$$

Seja $Q = \tilde{T}_n - P_n \not\equiv 0$. Como $\tilde{T}_n(x)$ e $P_n(x)$ são ambos polinômios mônicos de grau n, segue que $Q(x)$ é um polinômio de grau no máximo $(n-1)$. Além disso, nos $n+1$ pontos extremos $\overline{x}'_k$ de $\tilde{T}_n(x)$, temos

$$Q(\overline{x}'_k) = \tilde{T}_n(\overline{x}'_k) - P_n(\overline{x}'_k) = \frac{(-1)^k}{2^{n-1}} - P_n(\overline{x}'_k)\ .$$

Uma vez que

$$|\ P_n(\overline{x}'_k)\ | \leq \max_{x\in[-1,1]} |\ P_n(x)\ | < \frac{1}{2^{n-1}};\ k = 0, 1, \ldots, n,$$

então

$$Q(\overline{x}'_k) < 0, \text{ quando } k \text{ é ímpar, e } Q(\overline{x}'_k) > 0, \text{ quando } k \text{ é par.}$$

Como Q é contínuo, pelo teorema do valor intermediário, isso implica que o polinômio $Q(x)$ tem pelo menos uma raiz entre $\overline{x}'_j$ e $\overline{x}'_{j+1}$, para cada $j = 0, 1, \cdots, n-1$. Ou seja, Q tem pelo menos n raízes no intervalo $[-1, 1]$. Mas isso é um absurdo, pois $Q(x)$ é um polinômio não identicamente nulo cujo grau é menor que n. O mesmo raciocínio se aplica se supormos que vale a igualdade $\max_{x\in[-1,1]} |\ P_n(x)\ | = \frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\ \tilde{T}_n(x)\ |$.

Por outro lado, se $P_n \equiv \tilde{T}_n$, é óbvio que $\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\ \tilde{T}_n(x)\ | = \max_{x\in[-1,1]} |\ P_n(x)\ |$. □

Consoante Burden [1], esse teorema pode ser utilizado para responder à questão sobre onde posicionar os nós de interpolação para minimizar o erro na interpolação de Lagrange. Aplicando o Teorema 2.4 ao intervalo $[-1, 1]$ temos que se $x_0, \ldots, x_n$ forem números distintos nesse intervalo e $f \in C^{n+1}$, então para cada $x \in [-1, 1]$, existe um número $\xi(x) \in (-1, 1)$ com

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n) \ ,$$

em que $p_n(x)$ é o polinômio interpolador de Lagrange. Apesar de não haver controle sobre $\xi(x)$, podemos efetivamente encontrar $x_0, \ldots, x_n$ que minimize o valor de

$$\mid (x - x_0)(x - x_1)\cdots(x - x_n) \mid$$

no intervalo $[-1, 1]$. Para isso, perceba que $(x - x_0)(x - x_1)\cdots(x - x_n)$ é um polinômio mônico de grau $(n + 1)$. Vimos anteriormente que o mínimo é obtido quando

$$(x - x_0)(x - x_1)\cdots(x - x_n) = \tilde{T}_{n+1}(x) \ .$$

Algebricamente isso é o mesmo que dizer que cada x_k é uma raiz simples de $\tilde{T}_{n+1}(x)$ (que pelo Teorema 3.1 são todas distintas). Ou seja, para limitar $\mid (x - x_0)(x - x_1)\cdots(x - x_n) \mid$ no menor valor possível, basta tomar cada x_k como o (k+1)-ésimo zero de $\tilde{T}_{n+1}(x)$, $k = 0, 1, \ldots, n$. Assim, escolhemos x_k como sendo

$$x_k = \overline{x}_{k+1} = cos\left(\frac{2k+1}{2(n+1)}\pi\right). \tag{3.6}$$

Pelo Teorema 3.2, isso também implica que

$$\frac{1}{2^n} = \max_{x\in[-1,1]} \mid (x - \overline{x}_1)(x - \overline{x}_2)\cdots(x - \overline{x}_{n+1}) \mid \leq \max_{x\in[-1,1]} \mid (x - x_0)(x - x_1)\cdots(x - x_n) \mid ,$$

para qualquer escolha de $x_0, x_1, \ldots, x_n$ no intervalo $[-1, 1]$.

Em determinada situação, podemos nos referir ao conjunto de todos x_k da Equação (3.6) como os **pontos de Chebyshev**.

Corolário. *Se $p_n(x)$ for o polinômio interpolador de grau no máximo n com nós nas raízes de $T_{n+1}(x)$, então*

$$\max_{x\in[-1,1]} \mid f(x) - p_n(x) \mid \leq \frac{1}{2^n(n+1)!} \max_{x\in[-1,1]} \mid f^{(n+1)}(x) \mid ,$$

para cada $f \in C^{n+1}$ definida em $[-1, 1]$.

Demonstração:

Basta ver que

$$\max_{x\in[-1,1]} \mid f(x) - p_n(x) \mid \leq \max_{x\in[-1,1]} \left(\frac{\mid \prod_{i=0}^{n}(x - \overline{x}_{i+1}) \mid}{(n+1)!} \cdot \max_{x\in[-1,1]} \mid f^{(n+1)}(x) \mid \right)$$

$$\leq \max_{x\in[-1,1]} \left(\frac{\mid \prod_{i=0}^{n}(x - \overline{x}_{i+1}) \mid}{(n+1)!} \right) \max_{x\in[-1,1]} \mid f^{(n+1)}(x) \mid$$

$$= \frac{1}{2^n(n+1)!} \max_{x\in[-1,1]} \mid f^{(n+1)}(x) \mid .$$

□

Ainda de acordo com Burden [1], a técnica acima de escolher pontos que minimizem o erro da interpolação pode ser estendida a um intervalo fechado geral $[a, b]$ utilizando uma mudança de variáveis

$$\tilde{x} = \frac{1}{2}[(b-a)\overline{x} + a + b]$$

para transformar os números $\overline{x}_{k+1}$ no intervalo $[-1, 1]$ em um número correspondente $\tilde{x}_{k+1}$ no intervalo arbitrário $[a, b]$, para $k = 0, 1, \ldots, n$.

EXEMPLO

Vamos considerar o problema introduzido no início do capítulo. Queremos obter um polinômio interpolador de grau 10 que aproxime $f(x) = \dfrac{1}{1+25x^2}$ no intervalo $[-1, 1]$ com o menor erro possível.

De acordo com o que vimos anteriormente, para alcançar tal objetivo, devemos posicionar os 11 pontos para interpolação nas raízes de T_{11}. Isto é,

$$x_k = cos\left(\frac{2k+1}{22}\pi\right), k = 0, 1, \ldots, 10 \ .$$

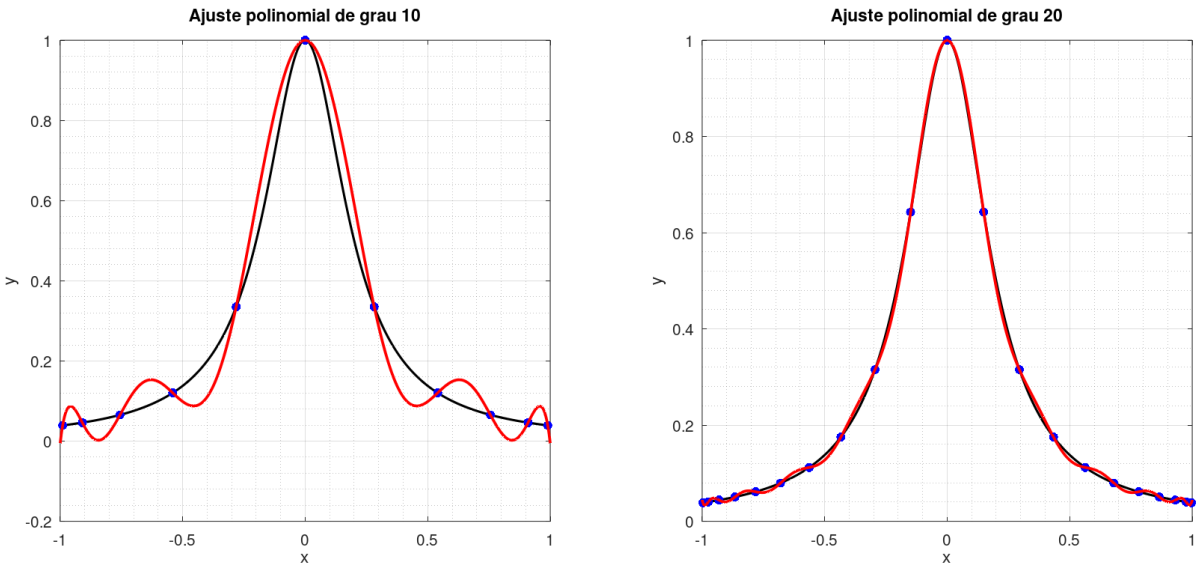
Assim, temos que os nós de interpolação devem ser $(x_k, f(x_k))$, $k = 0, 1, \ldots, 10$. Realizando a interpolação de Lagrange com esses dados, encontramos o seguinte polinômio interpolador (coeficientes com 6 casas decimais de precisão):

$$p_{10}(x) = -46.632917x^{10} + 130.105839x^8 - 133.444756x^6 + 61.443019x^4 - 12.476512x^2 + 1 \ .$$

Observacão: *Se, até aqui, ainda houverem dúvidas de como obter computacionalmente esse polinômio, deixamos como modelo no Apêndice B.2 o algoritmo que utilizamos nesse exemplo.*

Exibimos na Figura 3.3, do lado esquerdo, o polinômio interpolador obtido (em vermelho) e a função original $f(x)$ (em preto). Do lado direito, temos uma interpolação realizada da mesma maneira, porém de grau 20 (utilizando 21 pontos de Chebyshev no intervalo). Compare esse resultado com a Figura 3.1.

Figura 3.3: Pontos de Chebyshev - Interpolação de Chebyshev.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

4. Interpolação por Partes

Nos capítulos anteriores tratamos da interpolação polinomial em intervalos fechados e mostramos que nem sempre obtemos uma aproximação melhor se introduzirmos mais nós na interpolação. Na verdade, polinômios de alto grau podem oscilar erraticamente se tomarmos conjuntos arbitrários de pontos. Evidenciou-se isso no capítulo anterior, através do fenômeno de Runge, onde foi apresentado uma solução para minimizar esse problema ao tomar a distribuição de pontos de Chebyshev. No entanto, a interpolação de Chebyshev (tomamos a liberdade de chamá-la assim) apenas permite uma boa aproximação de funções regulares f quando a sua expressão é conhecida. Se f não for regular ou se f for conhecida somente pelos seus valores num dado conjunto de pontos (que não coincidem com os nós de Chebyshev), pode-se recorrer a outra técnica de interpolação, que chamamos de **aproximação polinomial por partes** ou interpolação composta.

Essa abordagem alternativa consiste em dividir o intervalo em uma coleção de subintervalos e construir um polinômio aproximador diferente (em geral) em cada subintervalo.

4.1 Interpolação linear por partes

De acordo com Burden [1], a aproximação polinomial por partes mais simples é a **interpolação linear por partes**, que consiste em unir um conjunto de pontos dados

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$$

por uma série de retas. Mais precisamente, dada uma distribuição ordenada de nós $x_0 < x_1 < \cdots < x_n$, denotamos o intervalo $[x_i, x_{i+1}]$ por I_i. Aproxima-se f por uma função contínua que, em cada intervalo, está definida pelo segmento de reta que une os dois pontos $(x_i, f(x_i))$ e $(x_{i+1}, f(x_{i+1}))$. A função linear que interpola esses dois extremos no intervalo I_i é dada por

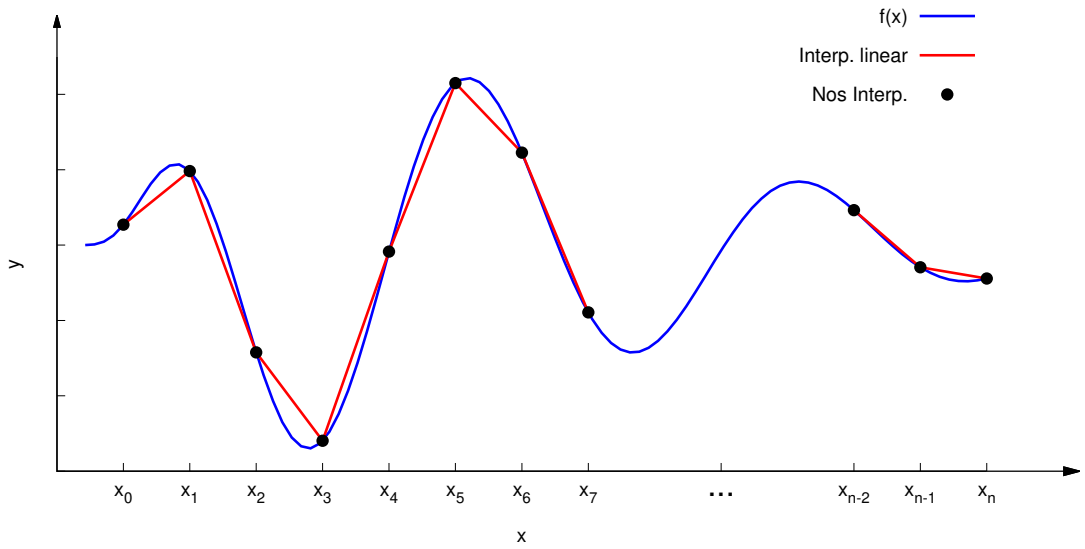
$$P_i(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \ , \ \forall x \in I_i. \tag{4.1}$$

Assim, o resultado da interpolação linear é a seguinte função contínua definida por partes no intervalo $[x_0, x_n]$:

$$s(x) = P_i(x) \ , \ x \in [x_i, x_{i+1}]. \tag{4.2}$$

Não trazemos exemplo de aplicação desse método, por se tratar de um processo muito simples (não há nada de extraordinário em ligar vários pontos por segmentos de reta). Apesar disso, ilustramos de maneira genérica esse procedimento por meio da Figura 4.1.

Figura 4.1: Interpolação linear por partes - Esquema ilustrativo.



Fonte: De autoria própria (elaborado através do gnuplot v5.2).

Parafraseando Burden [1], uma desvantagem da interpolação utilizando funções lineares é a eventual falta de diferenciabilidade nos extremos dos subintervalos, o que, em um contexto geométrico, significa que a função interpoladora não é suave. Muitas vezes, torna-se evidente a partir das condições dadas que a suavidade da função é essencial, assim, a função aproximadora precisa ser continuamente diferenciável.

Para realizar computacionalmente a interpolação linear por partes, pode-se utilizar a seguinte instrução em MATLAB, encontrada em Moler [5]:

```
function v = piecelin(x,y,u)
%PIECELIN Interpolacao linear por partes.
% v = piecelin(x,y,u) encontra o ajuste linear L(x)
% com L(x(j)) = y(j) e retorna v(k) = L(u(k)).

% Primeira diferenca dividida
    delta = diff(y)./diff(x);

% Encontra o subintervalo de indice k tal que x(k) <= u < x(k+1)
    n = length(x);
    k = ones(size(u));
    for j = 2:n-1
        k(x(j) <= u) = j;
    end

    % Avaliar interpolacao
    s = u - x(k);
    v = y(k) + s.*delta(k);
```

Nesse código, a entrada u é o vetor de pontos onde a interpolação deve ser avaliada e k é, na verdade, um outro vetor de índices.

O código calcula as primeiras diferenças divididas (delta) correspondentes a y em relação a x. Isso é feito utilizando a função *diff*, que calcula a diferença entre elementos consecutivos de um vetor. Em seguida, o código determina em qual subintervalo u está localizado, ou seja, qual subintervalo contém u. Ele faz isso iterando sobre x e verificando em qual subintervalo cada valor de u está contido. Por fim, o código usa a fórmula da interpolação linear por partes para calcular os valores de interesse v. Ele calcula v usando a equação da reta que passa pelos pontos $(x(k), y(k))$ e $(x(k+1), y(k+1))$ dentro de cada subintervalo, onde k é o índice do subintervalo.

Podemos ainda, simplesmente, usar a função *interp1* (MATLAB/Octave), cuja sintaxe pode ser, por exemplo, *interp1(x,y,z)* ou *interp1(x,y,z,linear)*; nesse caso, a função calcula os valores em pontos arbitrários (guardados no vetor z, que pode ter dimensão arbitrária) da função linear por partes que interpola os valores $y(i)$ nos nós $x(i)$, para $i = 1, \ldots, n+1$. Essa maneira é muito mais simples e direta do que implementar uma função própria (como a instrução apresentada) para encontrar a interpolação linear por partes.

4.2 Interpolação de Hermite

Suponha que sejam dados $n+1$ números distintos $x_0, x_1, \ldots, x_n$ em $[a, b]$ e números inteiros não negativos $m_0, m_1, \ldots, m_n$. Seja $m = max\{m_0, m_1, \ldots, m_n\}$. Chamamos de *polinômio osculador* ao polinômio de menor grau que aproxima uma função $f \in C^m$ (definida em $[a, b]$) em x_i, para cada $i = 0, \ldots, n$, tal que coincide com a função f e todas suas respectivas derivadas de grau menor ou igual a m_i em x_i. O grau desse polinômio osculador é de, no máximo,

$$M = n + \sum_{i=0}^{n} m_i$$

pois o número de condições a serem satisfeitas dessa forma é $\sum_{i=0}^{n} m_i + (n+1)$ e, como sabemos, um polinômio de grau M tem $M+1$ coeficientes que podem ser usados para satisfazer tais condições. De forma mais clara, estabelecemos formalmente sua definição, assim como em Burden [1], da maneira a seguir.

Definição. *Sejam $x_0, x_1, \ldots, x_n$ os $n+1$ números distintos em $[a, b]$ e para $i = 0, 1, \ldots, n$, seja m_i um número inteiro não negativo. Seja $f \in C^m$, definida em $[a, b]$, onde $m = \max_{0 \le i \le n} m_i$. O* ***polinômio osculador*** *que aproxima f é o polinômio $P(x)$ de menor grau tal que*

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k} \text{ , para cada } i = 0, 1, \ldots, n \text{ e } k = 0, 1, \ldots, m_i.$$

Observe que, quando $n = 0$, o polinômio osculador que aproxima f é o m_0-ésimo polinômio de Taylor de f em x_0, pois:

$$\frac{d^k}{dx^k}\left(f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \cdots + \frac{f^{(m_0)}(x_0)}{m_0!}(x - x_0)^{m_0}\right) = \frac{d^k f(x_0)}{dx^k} \text{ ,}$$

para cada $k = 0, 1, \ldots, m_i$.

Também, quando $m_i = 0$ para todo i, o polinômio osculador é o enésimo polinômio interpolador de Lagrange de f por $x_0, x_1, \ldots, x_n$, pois essencialmente temos a mesma condição (de nós) que dá origem à interpolação polinomial que vimos no Capítulo 2:

$$P(x_i) = f(x_i);\ i = 0, 1, \ldots, n.$$

Por esses motivos, dizemos que os polinômios osculadores generalizam tanto os polinômios de Taylor quanto os polinômios de Lagrange. A palavra "ósculo", do latim *osculum*, significa "beijo"; quando usada no contexto de curvas, indica que ela "apenas toca e tem a mesma forma". A interpolação de Hermite possui essa propriedade osculadora. Ela ajusta uma curva dada, e sua derivada força a curva de interpolação a "beijar" a curva dada.

Quando temos $m_i = 1$, para todo $i = 0, 1, \cdots, n$, chamamos o polinômio osculador de **polinômio de Hermite**. Esses polinômios coincidem com a função f nos nós $x_0, x_1, \ldots, x_n$, assim como os de Lagrange. Mas, além disso, como suas primeiras derivadas coincidem com as de f, eles têm a mesma "forma" da função nos nós $(x_i, f(x_i))$ (as *retas tangentes* do polinômio e da função são as mesmas nesses pontos).

Teorema 4.1 (Polinômios de Hermite)**.** *Seja $f \in C^1$ definida em $[a, b]$. Se $x_0, \ldots, x_n \in [a, b]$ são distintos, o único polinômio de grau mínimo que coincide com f, e sua derivada com f', em $x_0, \ldots, x_n$ é o polinômio de Hermite de grau no máximo $2n + 1$ dado por*

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_{n,j}(x)\ ,$$

onde, para $L_{n,j}(x)$ denotando o j-ésimo coeficiente polinomial de Lagrange de grau n, temos

$$H_{n,j}(x) = [1 - 2(x - x_j) L'_{n,j}(x_j)] L^2_{n,j}(x) \text{ e } \hat{H}_{n,j}(x) = (x - x_j) L^2_{n,j}(x).$$

Além disso, se $f \in C^{2n+2}$ em todo $[a, b]$, então

$$f(x) = H_{2n+1}(x) + \frac{(x - x_0)^2 \cdots (x - x_n)^2}{(2n + 2)!} f^{(2n+2)(\xi(x))}\ , \tag{4.3}$$

para algum $\xi(x)$, geralmente desconhecido, no intervalo (a, b).

Demonstração:

Antes de tudo, vamos supor, sem perda de generalidade, que $a \leq x_0 < x_1 < \cdots < x_n \leq b$. Se assim não fosse, bastaria uma reordenação de números.

Da Equação (2.6), sabemos que

$$L_{n,j}(x_i) = L_j(x_i) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

Por isso, quando $i \neq j$,

$$H_{n,j}(x_i) = 0 \text{ e } \hat{H}_{n,j}(x_i) = 0\ ,$$

ao passo que, para cada i,

$$H_{n,i}(x_i) = [1 - 2(x_i - x_i)L'_{n,i}(x_i)] \cdot 1 = 1 \text{ e } \hat{H}_{n,i}(x_i) = (x_i - x_i) \cdot 1^2 = 0 \ .$$

Como consequência, temos

$$H_{2n+1}(x_i) = \sum_{\substack{j=0 \\ j\neq i}}^{n} f(x_j) \cdot 0 + f(x_i) \cdot 1 + \sum_{\substack{j=0 \\ j\neq i}}^{n} f'(x_j) \cdot 0 + f'(x_i) \cdot 0 = f(x_i) \ .$$

Logo, H_{2n+1} coincide com f em $x_0, x_1, \ldots, x_n$.

Para demonstrar a coincidência de H'_{2n+1} com f' nos nós, observe primeiramente que $L_{n,j}(x)$ é um fator de $H'_{n,j}(x)$, e então $H'_{n,j}(x_i) = 0$ quando $i \neq j$. Além disso, quando $i = j$ temos $L_{n,i}(x_i) = 1$, então

$$\begin{aligned} H'_{n,i}(x_i) &= -2L'_{n,i}(x_i) \cdot L^2_{n,i}(x_i) + [1 - 2(x_i - x_i)L'_{n,i}(x_i)]2L_{n,i}(x_i)L'_{n,i}(x_i) \\ &= -2L'_{n,i}(x_i) + 2L'_{n,i}(x_i) = 0 \ . \end{aligned}$$

Assim, $H'_{n,j}(x_i) = 0$ quaisquer que sejam i e j.

Por outro lado,

$$\begin{aligned} \hat{H}'_{n,j}(x_i) &= L^2_{n,j}(x_i) + (x_i - x_j)2L_{n,j}(x_i)L'_{n,j}(x_i) \\ &= L_{n,j}(x_i)[L_{n,j} + 2(x_i - x_j)] \ . \end{aligned}$$

Logo, $\hat{H}'_{n,j}(x_i) = 0$ se $i \neq j$ e $\hat{H}'_{n,i}(x_i) = 1$. Combinando esses fatos, temos

$$H'_{2n+1}(x_i) = \sum_{j=0}^{n} f(x_j) \cdot 0 + \sum_{\substack{j=0 \\ j\neq i}}^{n} f'(x_j) \cdot 0 + f'(x_i) \cdot 1 = f'(x_i) \ .$$

Portanto, H_{2n+1} coincide com f e H'_{2n+1} com f' em $x_0, x_1, \ldots, x_n$.

Para provar a unicidade desse polinômio, suponha que exista outro polinômio $P(x)$ de grau no máximo $2n + 1$ que também verifique $P(x_i) = f(x_i)$ e $P'(x_i) = f'(x_i)$, para $i = 0, \ldots, n$. Considere $D(x) = H_{2n+1}(x) - P(x)$. Temos, então, que o grau de $D(x)$ também deve ser menor ou igual a $2n + 1$.

Veja que

$$D(x_i) = H_{2n+1}(x_i) - P(x_i) = f(x_i) - f(x_i) = 0 \ , \forall i = 0, 1, \ldots, n.$$

Pelo teorema de Rolle (ou também pelo teorema do valor médio), isso implica que em cada subintervalo aberto (x_i, x_{i+1}), $i = 0, \ldots, n-1$, existe um número c_i tal que $D'(c_i) = 0$. Ou seja $D'(x)$ possui pelo menos uma raiz em cada um desses subintervalos. Porém, temos também que

$$D'(x_i) = H'_{2n+1}(x_i) - P'(x_i) = f'(x_i) - f'(x_i) = 0 \ .$$

Então, o polinômio $D'(x)$ tem pelo menos $n + (n + 1) = 2n + 1$ raízes no intervalo $[x_0, x_n]$. Mas isso é um absurdo, uma vez que o grau de $D'(x)$ deve ser no máximo $2n$. Segue então que $D(x)$ só pode ser o polinômio nulo. Logo,

$$D(x) = 0 \Rightarrow H_{2n+1}(x) = P(x) \ .$$

Por fim, para a fórmula do erro, se $x = x_k$, $k = 0, 1, \ldots, n$, então $H_{2n+1}(x_k) = f(x_k)$, e qualquer que seja a escolha de $\xi(x)$ em (a, b) vale a igualdade da Equação (4.3).

Seja então $x \neq x_k$, para todo $k = 0, 1, \ldots, n$. Para $t \in [a, b]$, defina a função g por:

$$\begin{aligned} g(t) &= f(t) - H_{2n+1}(t) - [f(x) - H_{2n+1}(x)]\frac{(t-x_0)^2 \cdots (t-x_n)^2}{(x-x_0)^2 \cdots (x-x_n)^2} \\ &= f(t) - H_{2n+1}(t) - [f(x) - H_{2n+1}(x)] \prod_{i=0}^{n} \frac{(t-x_i)^2}{(x-x_i)^2} \ . \end{aligned}$$

Por hipótese, $f \in C^{2n+2}$ em $[a, b]$ e $H_{2n+1} \in C^\infty$ em $[a, b]$. Então $g \in C^{2n+2}$ em $[a, b]$. Para $t = x_k$, temos

$$g(x_k) = f(x_k) - H_{2n+1}(x_k) - [f(x) - H_{2n+1}(x)] \prod_{i=0}^{n} \frac{(x_k - x_i)^2}{(x - x_i)^2} = 0 - [f(x) - H_{2n+1}(x)] \cdot 0 = 0 \ .$$

Também,

$$\begin{aligned} g(x) &= f(x) - H_{2n+1}(x) - [f(x) - H_{2n+1}(x)] \prod_{i=0}^{n} \frac{(x - x_i)^2}{(x - x_i)^2} \\ &= f(x) - H_{2n+1}(x) - [f(x) - H_{2n+1}(x)] = 0 \ . \end{aligned}$$

Assim, $g \in C^{2n+2}$ e g vale zero em $n+2$ números distintos $x, x_0, x_1, \ldots, x_n$ em $[a, b]$. Aliando isso ao teorema de Rolle (argumento análogo ao que fizemos para a unicidade) e observando que

$$\begin{aligned} g'(x_k) &= f'(x_k) - H'_{2n+1}(x_n) - [f(x) - H_{2n+1}(x)] \cdot \frac{d}{dt}\left[\prod_{i=0}^{n} \frac{(t-x_i)^2}{(x-x_i)^2}\right]_{t=x_k} \\ &= 0 - [f(x) - H_{2n+1}(x)] \cdot \frac{1}{\prod_{i=0}^{n}(x-x_i)^2} \cdot \frac{d}{dt}\left[(t-x_k)^2 \prod_{\substack{i=0 \\ i\neq k}}^{n} (t-x_i)^2\right]_{t=x_k} \\ &= -\frac{[f(x) - H_{2n+1}(x)]}{\prod_{i=0}^{n}(x-x_i)^2} \cdot \left[\cancelto{0}{2(t-x_k)} \cdot \prod_{\substack{i=0 \\ i\neq k}}^{n} (t-x_i)^2 + \cancelto{0}{(t-x_k)^2} \cdot \frac{d}{dt}\left(\prod_{\substack{i=0 \\ i\neq k}}^{n} (t-x_i)^2\right)\right]_{t=x_k} \\ &= -\frac{[f(x) - H_{2n+1}(x)]}{\prod_{i=0}^{n}(x-x_i)^2} \cdot 0 = 0 \ , \end{aligned}$$

concluímos que o polinômio $g'(t)$ possui $(n+1) + (n+1) = 2n+2$ raízes distintas no intervalo $[a, b]$. Pelo teorema generalizado de Rolle, existe um número ξ em (a, b) para o qual $(g')^{(2n+1)}(\xi) = g^{(2n+2)}(\xi) = 0$. Então,

$$0 = g^{(2n+2)}(\xi) = f^{(2n+2)}(\xi) - H_{2n+1}^{(2n+2)}(\xi) - [f(x) - H_{2n+1}(x)] \frac{d^{2n+2}}{dt^{2n+2}}\left[\prod_{i=0}^{n} \frac{(t-x_i)^2}{(x-x_i)^2}\right]_{t=\xi} \ . \quad (4.4)$$

Como $H_{2n+2}(x)$ é um polinômio de grau no máximo $2n+1$, a $(2n+2)$-ésima derivada $H_{2n+1}^{(2n+2)}(x)$ é identicamente nula. Além disso, $\prod_{i=0}^{n}\left[\frac{(t-x_i)^2}{(x-x_i)^2}\right]$ é um polinômio de grau $(2n+2)$. Então, temos

$$\prod_{i=0}^{n} \frac{(t-x_i)^2}{(x-x_i)^2} = \left[\frac{1}{\prod_{i=0}^{n}(x-x_i)^2}\right] t^{2n+2} + \textit{(termos de grau inferior em } t\textit{)},$$

e

$$\frac{d^{n+2}}{dt^{n+2}} \prod_{i=0}^{n} \frac{(t-x_i)^2}{(x-x_i)^2} = \frac{(2n+2)!}{\prod_{i=0}^{n}(x-x_i)^2} \ .$$

A Equação (4.4) agora se transforma em

$$0 = f^{(n+2)}(\xi) - 0 - [f(x) - H_{2n+1}(x)] \frac{(2n+2)!}{\prod_{i=0}^{n}(x-x_i)^2} \ ,$$

e, ao isolar $f(x)$, obtemos finalmente

$$f(x) = H_{2n+1}(x) + f^{(2n+2)}(\xi) \frac{\prod_{i=0}^{n}(x-x_i)^2}{(2n+2)!} \ .$$

□

Esse teorema fornece uma descrição fantástica de polinômio osculador. A fim de obter-se o polinômio de Hermite H_{2n+1}, só precisamos ter em mãos os dados $(x_0, f(x_0), f'(x_0))$, $(x_1, f(x_1), f'(x_1)), \ldots, (x_n, f(x_n), f'(x_n))$ para calcular as funções L_j de Lagrange (Equação (2.4)) e suas derivadas, com $j = 0, 1, \ldots, n$. Em seguida, os usamos para determinar cada $H_{n,j}$ e $\hat{H}_{n,j}$, na forma que vimos nesse último teorema. Por último, somamos os termos $f(x_j)H_{n,j}(x) + f'(x_j)\hat{H}_{n,j}(x)$, $j = 0, 1, \ldots, n$, e encontramos finalmente o polinômio H_{2n+1}, podendo avaliá-lo nos pontos em que tivermos interesse. Perceba que todos esses cálculos - principalmente a parte de determinar as funções L_j e suas derivadas - pode tornar o processo tedioso (e bastante demorado) até mesmo para valores pequenos de n. Em Burden [1], p. 151, é dado um exemplo com $n = 2$ que ilustra bem isso.

Para contornar esse problema, Burden [1] sugere um método alternativo para gerar polinômios de Hermite, que se baseia na fórmula de interpolação por diferenças divididas de Newton. Esse método usa a ligação entre a enésima diferença dividida e a enésima derivada de f, orientado da maneira que segue.

Suponha que os números distintos $x_0, x_1, \ldots, x_n$ sejam fornecidos juntamente com os valores de f e f' nesses números. Defina uma nova sequência $z_0, z_1, \ldots, z_{2n+1}$ por

$$z_{2i} = z_{2i+1} = x_i \ , \text{ para cada } i = 0, 1, \ldots, n,$$

e construa uma tabela de diferenças divididas no formato da Tabela 2.2, utilizando $z_0, z_1, \ldots, z_{2n+1}$.

Como $z_{2i} = z_{2i+1} = x_i$ para cada i, não podemos definir $f[z_{2i}, z_{2i+1}]$ pela fórmula de diferença dividida. Entretanto, podemos presumir, com base no teorema exposto no Apêndice A.2 - apesar de não satisfazer rigorosamente a hipótese do teorema -, que uma substituição razoável nessa situação seria $f[z_{2i}, z_{2i+1}] = f'(z_{2i}) = f'(x_i)$. Podemos utilizar, então, os valores

$$f'(x_0), f'(x_1), \ldots, f'(x_n)$$

no lugar das primeiras diferenças divididas não definidas

$$f[z_0, z_1], f[z_2, z_3], \ldots, f[z_{2n}, z_{2n+1}] \ .$$

As demais diferenças divididas são calculadas normalmente e, então, todas as diferenças divididas adequadas são empregadas na fórmula de interpolação de Newton (Equação (2.9)). A Tabela 4.1 ilustra as três primeiras colunas de diferenças divididas se estabelecêssemos, por exemplo, o polinômio de Hermite $H_5(x)$, para x_0, x_1 e x_2 dados. As diferenças divididas restantes são geradas da mesma maneira que na Tabela 2.2.

O polinômio de Hermite é dado então por

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \ldots, z_k](x - z_0)(x - z_1) \cdots (x - z_{k-1}) \ . \tag{4.5}$$

Observacão: Não provamos essa última parte, porém uma demonstração disso pode ser encontrada em Powell [9], p. 54-57.

Tabela 4.1: Interpolação de Hermite - Adequação das diferenças divididas.

z	ORDEM 0	ORDEM 1	ORDEM 2
$z_0 = x_0$	$f[z_0] = f(x_0)$		
$z_1 = x_0$	$f[z_1] = f(x_0)$	$f[z_0, z_1] = f'(x_0)$	
$z_2 = x_1$	$f[z_2] = f(x_1)$	$f[z_1, z_2] = \frac{f[z_2] - f[z_1]}{z_2 - z_1}$	$f[z_0, z_1, z_2] = \frac{f[z_1, z_2] - f[z_0, z_1]}{z_2 - z_0}$
$z_3 = x_1$	$f[z_3] = f(x_1)$	$f[z_2, z_3] = f'(x_1)$	$f[z_1, z_2, z_3] = \frac{f[z_2, z_3] - f[z_1, z_2]}{z_3 - z_1}$
$z_4 = x_2$	$f[z_4] = f(x_2)$	$f[z_3, z_4] = \frac{f[z_4] - f[z_3]}{z_4 - z_3}$	$f[z_2, z_3, z_4] = \frac{f[z_3, z_4] - f[z_2, z_3]}{z_4 - z_2}$
$z_5 = x_2$	$f[z_5] = f(x_2)$	$f[z_4, z_5] = f'(x_2)$	$f[z_3, z_4, z_5] = \frac{f[z_4, z_5] - f[z_3, z_4]}{z_5 - z_3}$

Deixamos no Apêndice B.3 o algoritmo construído aplicando-se esse método. Usá-lo-emos no exemplo a seguir.

EXEMPLO

Construa o polinômio de Hermite que interpole os dados da tabela abaixo. Em seguida, obtenha uma aproximação para $f(0.25)$.

Tabela 4.2: Interpolação de Hermite - Dados tabelados.

x	$f(x)$	$f'(x)$
-1	0.86199480	0.15536240
-0.5	0.95802009	0.23269654
0	1.0986123	0.33333333
0.5	1.2943767	0.45186776

Solução: Utilizando o algoritmo implementado para a interpolação de Hermite (Apêndice B.3), encontramos as seguintes diferenças divididas:

$$Q_0 = 0.86199480$$
$$Q_1 = 0.15536240$$
$$Q_2 = 0.07337636$$
$$Q_3 = 0.01583112$$
$$Q_4 = -0.00014728$$
$$Q_5 = -0.00089244$$
$$Q_6 = -0.00007672$$
$$Q_7 = 0.00006864.$$

Com isso, conseguimos construir o nosso polinômio:

$$H_7(x) = 0.86199480 + 0.15536240(x+1) + 0.07337636(x+1)^2 + 0.01583112(x+1)^2(x+0.5)$$
$$-0.00014728(x+1)^2(x+0.5)^2 - 0.00089244(x+1)^2(x+0.5)^2(x-0)$$
$$-0.00007672(x+1)^2(x+0.5)^2(x-0)^2 + 0.00006864(x+1)^2(x+0.5)^2(x-0)^2(x-0.5)$$

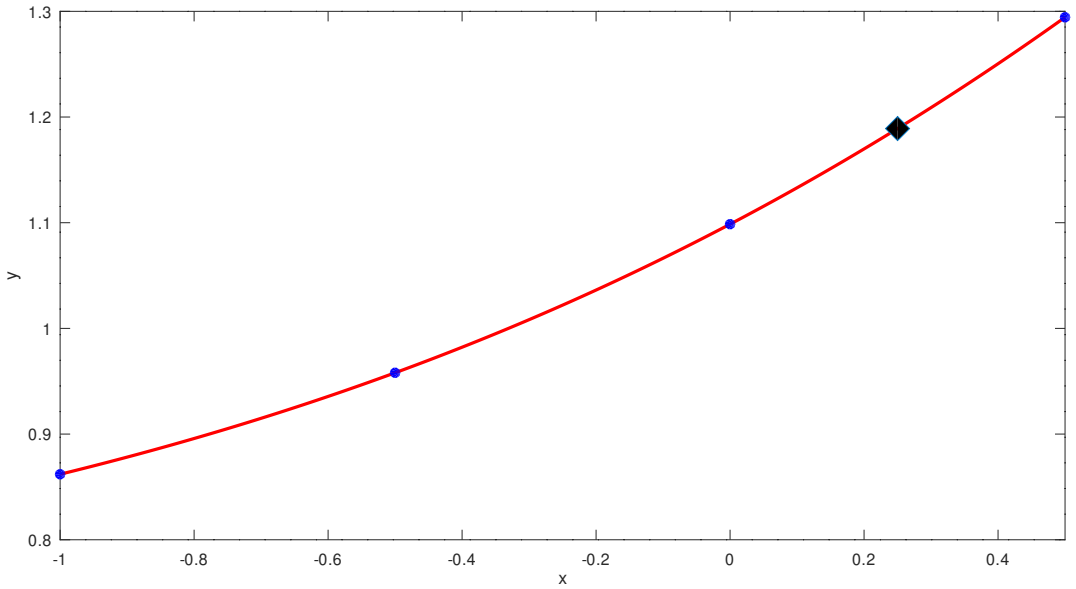
$$\Rightarrow H_7(x) = 0.00006864x^7 + 0.00009488x^6 - 0.00100248x^5 - 0.00308252x^4 + 0.01233945x^3 +$$
$$0.11110908x^2 + 0.33333333x + 1.0986123$$

Dessa forma, a aproximação para $f(0.25)$ é dada por

$$H_7(0.25) = 1.18906976\ .$$

Representamos essa interpolação no gráfico abaixo.

Figura 4.2: Interpolação de Hermite - Polinômio $H_7(x)$ e estimativa para $f(0.25)$.



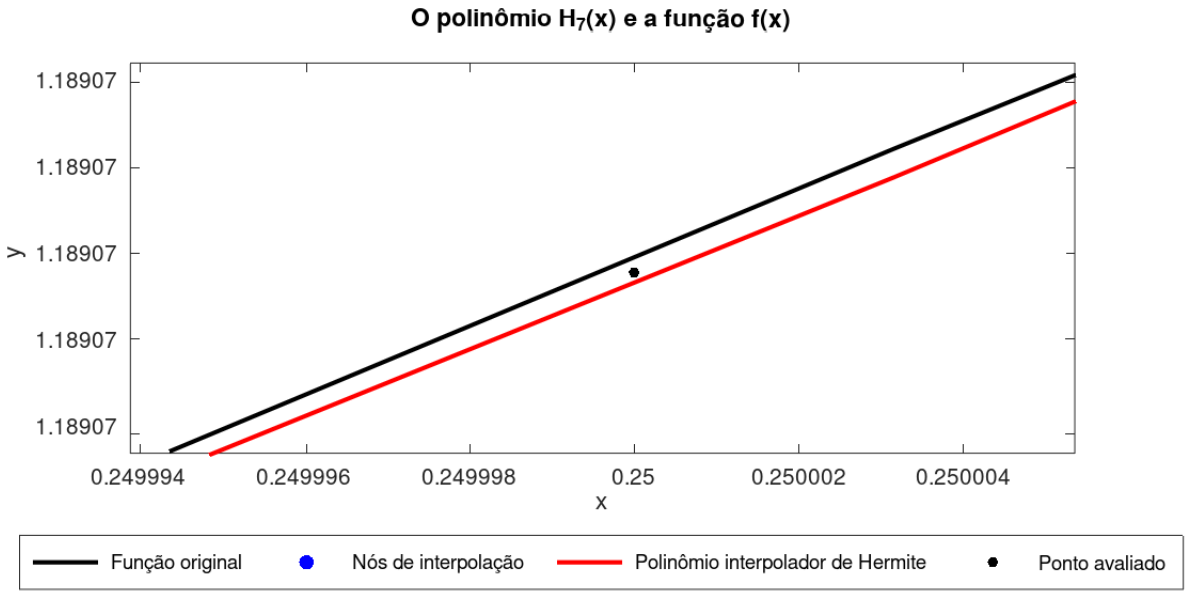
Fonte: De autoria própria (elaborado através do GNU Octave 9.1.0).

Os dados da Tabela 4.2 foram gerados a partir da função $f(x) = ln(e^x + 2)$. O erro absoluto da aproximação que obtivemos é

$$E_{abs} = \mid H_7(0.25) - ln(e^{0.25} + 2) \mid = \mid 1.18906976 - 1.18906993 \mid = 0.00000017\ .$$

O polinômio $H_7(x)$ passa tão próximo de $f(x)$ que se plotarmos um gráfico com essas duas curvas, sobrepostas no intervalo $[-1, 0.5]$, seria impossível distingui-las visualmente. Na Figura 4.3 exibimos uma excessiva ampliação em volta do ponto que avaliamos (note o "erro" na plotagem causado pela ordem de precisão exigida na escala).

Figura 4.3: Interpolação de Hermite - Comparação entre $H_7(x)$ e $f(x)$.



Fonte: De autoria própria (elaborado através do GNU Octave 8.3.0).

A razão de tratarmos a interpolação de Hermite no contexto de interpolação por partes é que ela surge como uma alternativa ao problema descrito na sessão anterior, sobre a eventualidade da interpolação por funções lineares não ser "suave". Podemos utilizar um polinômio por partes do tipo Hermite para substituir a aproximação linear nos subintervalos. Por exemplo, se os valores de f e de f' forem conhecidos em cada um dos pontos $x_0 < x_1 < \cdots < x_n$, um polinômio cúbico de Hermite pode ser usado em cada um dos subintervalos $[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n]$ para obter uma função que tenha uma derivada contínua no intervalo $[x_0, x_n]$. Denotemos por h_k o comprimento do k-ésimo subintervalo:

$$h_k = x_{k+1} - x_k \ .$$

Considere a seguinte função no intervalo $x_k \leq x \leq x_{k+1}$, espressa em termos das variáveis locais $s = x - x_k$ e $h = h_k$:

$$P(x) = \frac{3hs^2 - 2s^3}{h^3} f(x_{k+1}) + \frac{h^3 - 3hs^2 + 2s^3}{h^3} f(x_k) + \frac{s^2(s-h)}{h^2} f'(x_{k+1}) + \frac{s(s-h)^2}{h^2} f'(x_k) \ .$$

Este é um polinômio cúbico em s, e portanto em x, que satisfaz quatro condições de interpolação (ou informações), duas nos valores da função e duas nos valores da derivada:

$$P(x_k) = f(x_k) \ , \ P(x_{k+1}) = f(x_{k+1}),$$
$$P'(x_k) = f'(x_k) \ , \ P'(x_{k+1}) = f'(x_{k+1}) \ .$$

Nesse caso, $P(x)$ é exatamente o polinômio $H_3(x)$ definido em $[x_k, x_{k+1}]$. Para verificar isso, basta comparar a fórmula do polinômio de Hermite com essa expressão de P, fazendo as substituições $x - x_k = s$ e $x - x_{k+1} = s - h$, quando necessário. Sendo assim, determinar o polinômio cúbico de Hermite adequado em determinado intervalo é simplesmente uma questão de calcular $H_3(x)$ para aquele intervalo. Como as funções de Lagrange L_k necessárias para a determinação de H_3 são de grau um, isto pode ser feita com certa facilidade.

Contudo, se não conhecermos os valores das derivadas, precisamos definir essas inclinações de alguma forma (existem várias maneiras de fazer isso em Análise Numérica). A função 'pchip', no MATLAB, realiza a interpolação por partes via polinômios cúbicos de Hermite; ela pode ser usada quando sabemos apenas os valores da função nos pontos, na forma *pchip(x,y)*, como também fornecendo informações adicionais sobre a derivada da função nesses pontos, na forma *pchip(x,y,dydx)*. Para Octave, pode-se utilizar *interp1(x,y,xq,pchip)* para avaliar a função interpoladora diretamente nos pontos xq. Por ora, não abordaremos nenhum exemplo aplicando essa técnica de Hermite por partes, mas no Capítulo 5 utilizaremos essa interpolação para compará-la visualmente com outras.

Na verdade, muitas das mais eficientes técnicas de interpolação baseiam-se nos polinômios cúbicos por partes. Na seção a seguir veremos uma aproximação desse tipo, que não exije nenhuma informação sobre a derivada, exceto possivelmente nas extremidades do intervalo sobre o qual a função esteja sendo aproximada.

Por fim, ressaltamos, caso não tenha ficado suficientemente claro, que se a função $f(x)$ a ser aproximada for "suave" em $[x_0, x_n]$, então a interpolação de Hermite (seja por partes ou não) também a será, inclusive nos nós $x_1, x_2, \ldots, x_{n-1}$.

4.3 INTERPOLAÇÃO POR SPLINE CÚBICO

A ideia empregada na interpolação linear por partes pode ser estendida para polinômios de grau superior. A escolha de polinômios de grau superior implica numa maior liberdade na construção da interpolação, pois há um número maior de coeficientes. Quarteroni [8] dispõe sobre uma transição de interpolações por partes quadráticas para cúbicas, e em REAMAT [6] é apresentada uma definição que generaliza splines de ordem m. Nós abordaremos diretamente os splines cúbicos.

A **interpolação por spline cúbico**, como o nome sugere, usa polimônios cúbicos entre cada par de nós sucessivos e é a aproximação polinomial mais comumente empregada. No procedimento de splines cúbicos existe flexibilidade suficiente para assegurar que a função interpoladora seja não somente continuamente diferenciável no intervalo, mas também tenha uma segunda derivada contínua. Entretanto, diferentemente da interpolação de Hermite, a construção do spline cúbico não supõe que as derivadas da função interpoladora coincidam com aquelas da função que está sendo aproximada, mesmo nos nós. Trazemos uma definição de spline cúbico baseada na que é apresentada por Burden [1].

Definição. *Seja f uma função definida em $[a, b]$ e, em particular, num conjunto de nós $a =$*

$x_0 < x_1 < \cdots < x_n = b$. Um ***spline cúbico interpolador*** *S para f é uma função que satisfaz as seguintes condições:*

(a) *$S(x)$ é um polinômio cúbico, denotado por $S_j(x)$, em $[x_j, x_{j+1}]$, para cada $j = 0, 1, \ldots, n-1$;*
(b) *$S_j(x_j) = f(x_j)$ e $S_j(x_{j+1}) = f(x_{j+1})$, para cada $j = 0, 1, \ldots, n-1$;*
(c) *$S_{j+1}(x_{j+1}) = S_j(x_{j+1})$, para cada $j = 0, 1, \ldots, n-2$; (Implicada por* ***(b)****)*
(d) *$S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$, para cada $j = 0, 1, \ldots, n-2$;*
(e) *$S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$, para cada $j = 0, 1, \ldots, n-2$.*

Além disso, um dos seguintes conjuntos de condições de contorno também deve ser satisfeito:
(i) *$S''(x_0) = S''(x_n) = 0$ (condições de contorno* ***livres*** *ou* ***naturais****);*
(ii) *$S'(x_0) = f'(x_0)$ e $S'(x_n) = f'(x_n)$ (condições de contorno* ***fixadas****).*

Quando as condições de contorno livres ocorrem, o spline é chamado **spline natural**[1], e seu gráfico se assemelha à forma que uma haste longa flexível assumiria se fosse forçada a passar pelos pontos dados $\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$.

Neste trabalho tratamos apenas dos splines cúbicos naturais, mas vale dizer que, geralmente, condições de contorno fixadas levam a melhores aproximações, pois incluem mais informações sobre a função. Nesse caso, é necessário ter os valores da derivada nas extremidades (ou pelo menos uma aproximação precisa desses valores).

Apesar da definição de splines parecer complicada, construir um spline cúbico natural é, na verdade, relativamente simples.

Seja $[a, b]$ um intervalo dividido em n subintervalos, onde $a = x_0 < x_1 < \cdots < x_n = b$ são os pontos sobre os quais queremos construir o spline cúbico interpolador para uma dada função f. Para construir o spline, as condições da definição devem ser aplicadas aos polinômios cúbicos

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3 \ ,$$

para cada $j = 0, 1, \ldots, n-1$. Existem $4n$ constantes a serem determinadas, portanto são necessárias $4n$ condições. $2n$ condições vêm do fato de que os splines devem coincidir com os dados nos nós (condição **(b)** da definição). Isto é,

$$\begin{gathered} f(x_0) = S_0(x_0) \ , \ f(x_1) = S_0(x_1) \ ; \\ f(x_1) = S_1(x_1) \ , \ f(x_2) = S_1(x_2) \ ; \\ \vdots \\ f(x_{n-1}) = S_{n-1}(x_{n-1}) \ , \ f(x_n) = S_{n-1}(x_n) \ . \end{gathered}$$

Outras $2(n-1) = 2n-2$ condições são dadas a partir das condições **(d)** e **(e)**:

$$\begin{gathered} S'_0(x_1) = S'_1(x_1) \ , \ S''_0(x_1) = S''_1(x_1) \ ; \\ S'_1(x_2) = S'_2(x_2) \ , \ S''_1(x_2) = S''_2(x_2) \ ; \\ \vdots \\ S'_{n-2}(x_{n-1}) = S'_{n-1}(x_{n-1}) \ , \ S''_{n-2}(x_{n-1}) = S''_{n-1}(x_{n-1}) \ . \end{gathered}$$

[1] Um *spline* natural não tem condições impostas para a direção em suas extremidades, então a curva toma a forma de uma linha reta quando extrapola os dois pontos extremos de interpolação. O nome "spline natural" vem da interpretação de que um spline livre é como se fosse a forma natural que uma haste flexível assume se forçada a passar pelos pontos de interpolação especificados sem restrições adicionais.

As duas últimas condições decorrem das condições de contorno livres:

$$S_0''(x_0) = 0 \text{ e } S_{n-1}''(x_n) = 0 \ .$$

A solução desse sistema de equações nos dá o spline

$$S(x) = \begin{cases} a_0 + b_0(x - x_0) + c_0(x - x_0)^2 + d_0(x - x_0)^3 \ , \ x \in [x_0, x_1] \\ a_1 + b_1(x - x_1) + c_0(x - x_1)^2 + d_0(x - x_1)^3 \ , \ x \in [x_1, x_2] \\ \vdots \\ a_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3 \ , \ x \in [x_{n-1}, x_n] \end{cases} \tag{4.6}$$

Vale a pena conferir a seção sobre "Interpolação cúbica segmentada - spline" do capítulo 6 de REAMAT [6], onde é explorada essa construção de maneira mais rigorosa e elegante (contempla inclusive outras variações de splines cúbicos). O teorema enunciado a seguir garante a unicidade do spline natural. Não o provaremos, mas uma demonstração para ele pode ser encontrada em Burden [1], p. 162-164.

Teorema 4.2. *Se f for definida em $a = x_0 < x_1 < \cdots < x_n = b$, então f tem um spline S interpolador natural único nos nós $x_0, x_1, \ldots, x_n$, isto é, um spline interpolador que satisfaz as condições de contorno $S''(a) = 0$ e $S''(b) = 0$.*

Segundo Quarteroni [8], pode-se utilizar a interpolação polinomial para aproximar dados e funções em várias dimensões. Em particular, a interpolação por partes, baseada em funções lineares ou em funções spline, adapta-se bem sempre que o domínio Ω pode se subdividir em polígonos em 2D (triângulos e quadriláteros) e em poliedros em 3D (tetraedros ou prismas).

A interpolação por spline cúbico com as condições de contorno $S''(x_0) = S''(x_n) = 0$ pode ser obtida utilizando o algoritmo que disponibilizamos no Apêndice B.3. Se preferível, pode ser utilizada a função 'spline' no MATLAB, junto com a função 'ppval', para construir e avaliar a interpolação por splines. Também, no GNU Octave, a função 'interp1' realiza a interpolação polinomial por splines cúbicos se você usá-la na forma *interp1(x,y,xq,spline)*. No entanto, esteja ciente que essas funções 'spline' do MATLAB/Octave distorcem um pouco o spline cúbico natural próximo das extremidades, pois elas acrescentam duas condições adicionais, a saber:

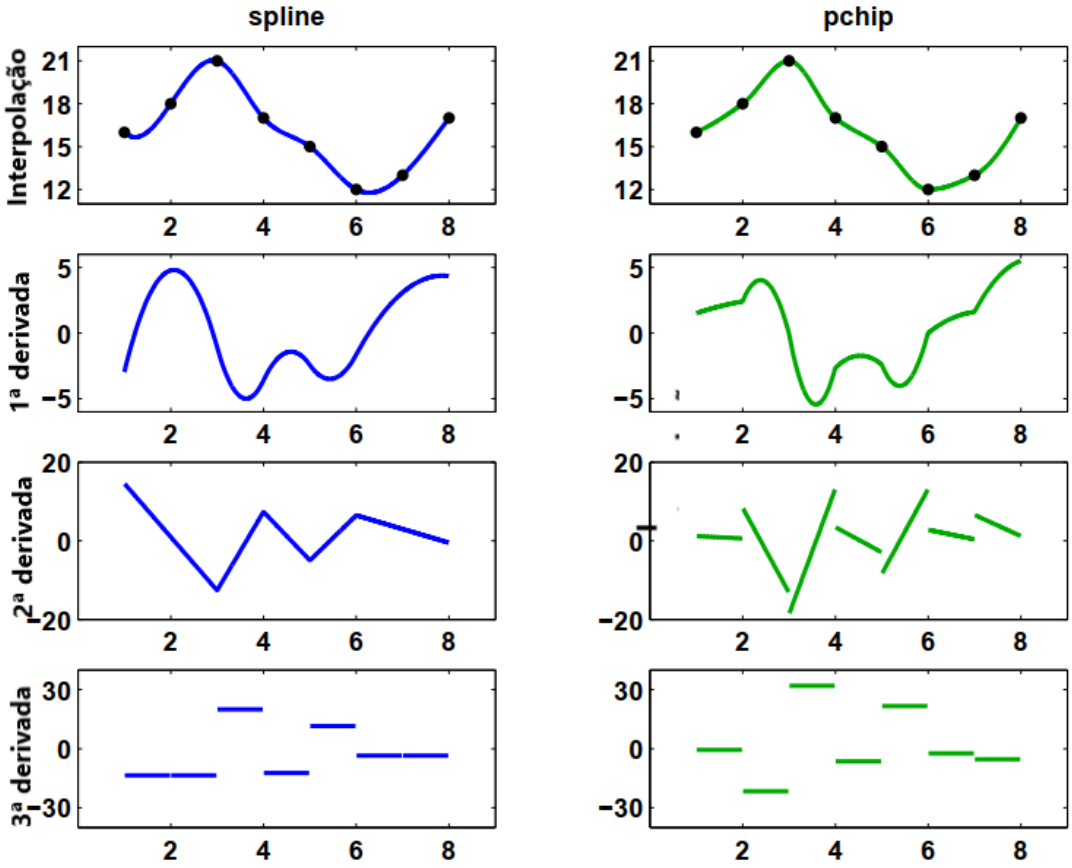
$$\lim_{x \to x_1} S_0'''(x) = S_1'''(x_1) \text{ e } \lim_{x \to x_{n-1}} S_{n-2}'''(x) = S_{n-1}'''(x_{n-1}) \ .$$

Isso obriga a terceira derivada de S a ser contínua em x_1 e x_{n-1}. Quando isso acontece, é comum chamar o spline cúbico de spline *not-a-knot*.

A interpolação por spline cúbico é muito semelhante à interpolação por partes via polinômios cúbicos de Hermite, afinal ambas as interpolações baseiam-se em polinômios cúbicos em cada subintervalo. Sobre isso, Moler [5] comenta que a diferença entre essas interpolações é quase imperceptível, mas se observarmos o comportamento de suas derivadas conseguimos entender a diferença entre as duas. Veja a Figura 4.4. A primeira derivada da função spline, $S'(x)$, é suave, enquanto a primeira derivada da função pchip, $P'(x)$, é apenas contínua. A segunda

derivada do spline, $S''(x)$, é contínua, enquanto a segunda derivada da pchip, $P''(x)$, admite "saltos" nos nós. A terceira derivada das duas são constantes, dentro de cada subintervalo de interpolação, mas observe como a $S'''(x)$ mantém os mesmos valores durante os dois primeiros e os dois últimos subintervalos, evidenciando o fenômeno que comentamos anteriormente do spline *not-a-knot*.

Figura 4.4: Interpolação por spline cúbico - As funções *spline* e *pchip*, e suas derivadas.



Fonte: Extraído de Moler [5], 2016.

EXEMPLO

Considere a seguinte tabela de valores de x e $y = f(x)$. Vamos obter um spline cúbico natural que interpole os dados.

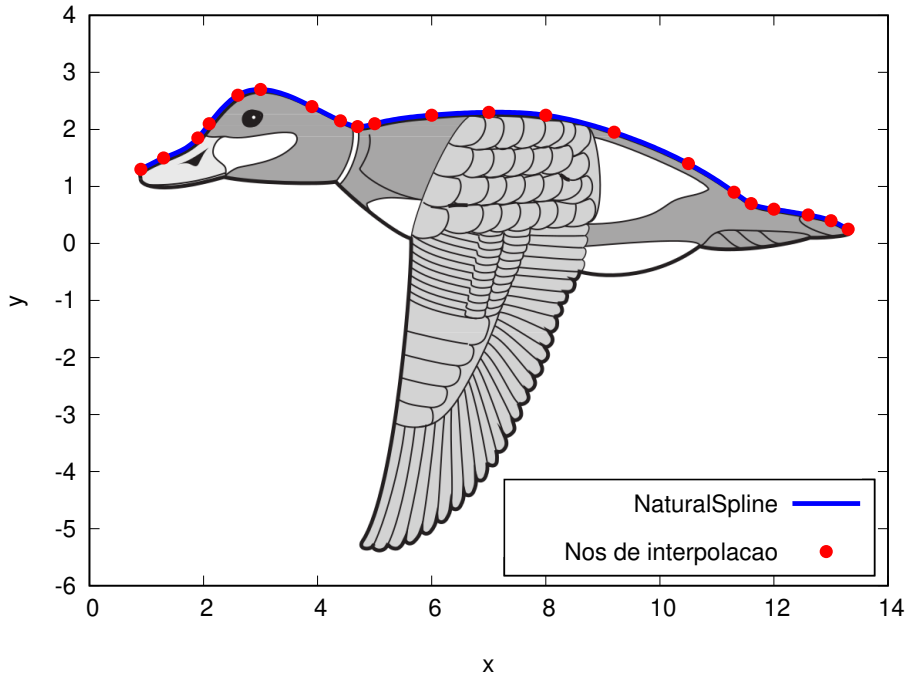
Tabela 4.3: Interpolação por spline cúbico - Dados para interpolação.

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

Temos 21 pontos (x_j, y_j), $j = 0, 1, \ldots, 20$, a serem interpolados. Logo, nosso spline deve ser dividido em 20 partes. Utilizando o algoritmo implementado para interpolação por spline cúbico (Apêndice B.3), conseguimos gerar os coeficientes que determinam o spline (Equação (4.6)), dispostos na tabela a seguir.

j	x_j	a_j	b_j	c_j	d_j
0	0.900000	1.300000	0.539624	0.000000	−0.247649
1	1.300000	1.500000	0.420752	−0.297179	0.946912
2	1.900000	1.850000	1.086803	1.407263	−2.956382
3	2.100000	2.100000	1.294942	−0.366567	−0.446635
4	2.600000	2.600000	0.593399	−1.036519	0.445051
5	3.000000	2.700000	−0.022191	−0.502457	0.174160
6	3.900000	2.400000	−0.503406	−0.032226	0.078076
7	4.400000	2.150000	−0.477075	0.084888	1.314171
8	4.700000	2.050000	−0.071316	1.267642	−1.581219
9	5.000000	2.100000	0.262340	−0.155455	0.043115
10	6.000000	2.250000	0.080776	−0.026109	−0.004666
11	7.000000	2.300000	0.014558	−0.040108	−0.024450
12	8.000000	2.250000	−0.139008	−0.113458	0.017471
13	9.200000	1.950000	−0.335834	−0.050564	−0.012728
14	10.500000	1.400000	−0.531830	−0.100202	−0.020325
15	11.300000	0.900000	−0.731178	−0.148983	1.213405
16	11.600000	0.700000	−0.492949	0.943082	−0.839275
17	12.000000	0.600000	−0.141335	−0.064048	0.036382
18	12.600000	0.500000	−0.178900	0.001440	−0.447971
19	13.000000	0.400000	−0.392775	−0.536126	0.595695

Figura 4.5: Interpolação por spline cúbico - Ajuste dos pontos.



Fonte: De autoria própria (elaborado através do GNU Octave 9.1.0).

O resultado dessa interpolação é mostrado na Figura 4.5. Os dados da Tabela 4.3 foram, na verdade, retirados de um exemplo em Burden [1], p. 172, em que os pontos são selecionados a partir da figura de um pato em pleno voo. O spline que obtivemos é a curva que aproxima o perfil superior do pato.

Adicionalmente, indicamos para leitura o e-book de PPGMAT [10], criado pelos estudantes (na época) da disciplina de Análise Numérica do Mestrado em Matemática da UFU, juntamente com a Professora Doutora Rosana Sueli da Motta Jafelice. Esse trabalho contém algumas aplicações bastante criativas com splines cúbicos e polinômios interpoladores de Lagrange.

5. Análise Comparativa em um Contexto Prático

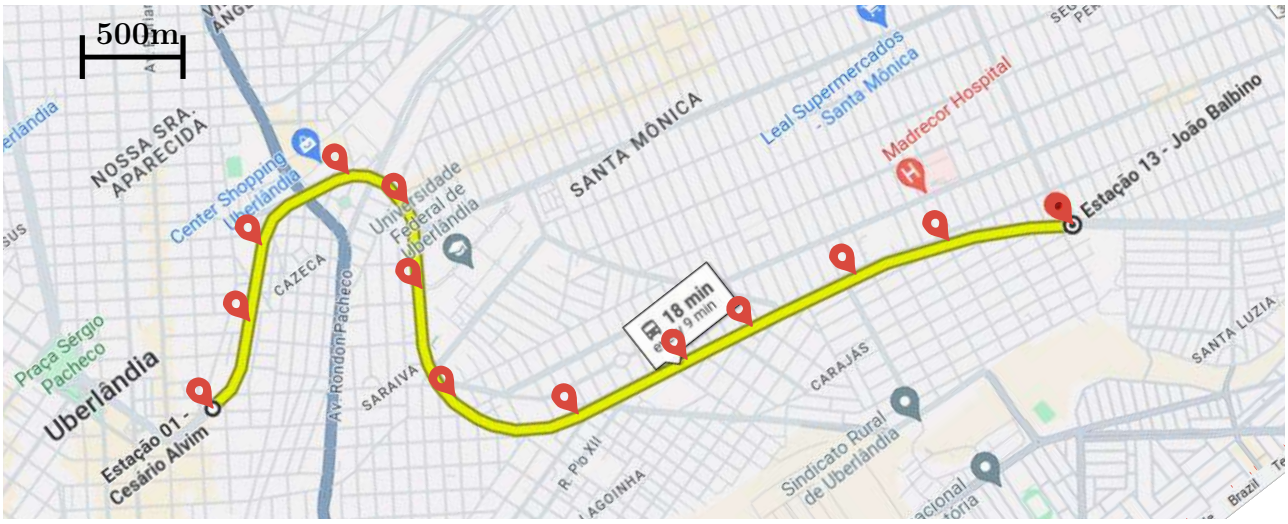
O objetivo deste pequeno capítulo é mostrar como podemos aplicar a interpolação polinomial em uma situação prática (dentre inúmeras outras possibilidades reais), e comparar os métodos entre si, a fim de identificar aquele que melhor se apresenta como solução do problema.

Imagine que queremos obter uma curva que represente o mapeamento de uma rua/avenida (ou até mesmo de uma estrada), dispondo somente da marcação de alguns pontos ao longo da via. Essa é a situação que abordaremos aqui.

Inserindo-se concretamente nesse contexto, tomamos uma das principais avenidas de Uberlândia, que possui uma forma muito peculiar no seu trajeto. Coincidentemente, esta avenida é endereço da Universidade Federal de Uberlândia. Estamos falando da Av. João Naves de Ávila.

Em primeiro lugar, para considerarmos a avenida como traço de uma curva, assumiremos que ela não possui largura, apenas comprimento. Além disso, os pontos que convenientemente escolhemos para interpolação foram as estações de ônibus, tidas como pontos de referência e usualmente frequentadas por boa parte da população de Uberlândia - Figura 5.1. Sendo assim, restringimos nosso problema apenas ao trecho compreendido entre a Estação 01 e a Estação 13 da avenida, ao invés da avenida inteira. Vamos supor que essas estações estejam dispostas sobre o plano cartesiano, e adotaremos a Estação 01 como origem (ponto $(0, 0)$).

Figura 5.1: Descrição do problema - Mapa das estações de ônibus da Av. João Naves de Ávila.



Fonte: Google Maps.

Utilizando o GNU Octave e outros softwares de edição de imagem, foi possível selecionar as coordenadas (em metros) de cada estação da Figura 5.1, organizadas na Tabela 5.1. Cada ponto (x_i, y_i) representa a respectiva Estação i, estes são os nós de interpolação.

Tabela 5.1: Dados do problema - Coordenadas dos pontos de interpolação (m).

i	x_i	y_i
0	0.0000	0.0000
1	179.4872	436.7089
2	237.1795	816.4557
3	679.4872	1145.5696
4	961.5385	1000.0000
5	1025.6410	594.9367
6	1185.8974	56.9620
7	1826.9231	−44.3038
8	2371.7949	227.8481
9	2705.1282	392.4051
10	3224.3590	639.2405
11	3692.3077	803.7975
12	4307.6923	905.0633

Apesar das coordenadas dos pontos estarem apresentadas com 4 casas decimais, as reais medidas podem diferir (e muito) dessa precisão. Esses valores estão relacionados com o tratamento computacional da imagem que tomamos.

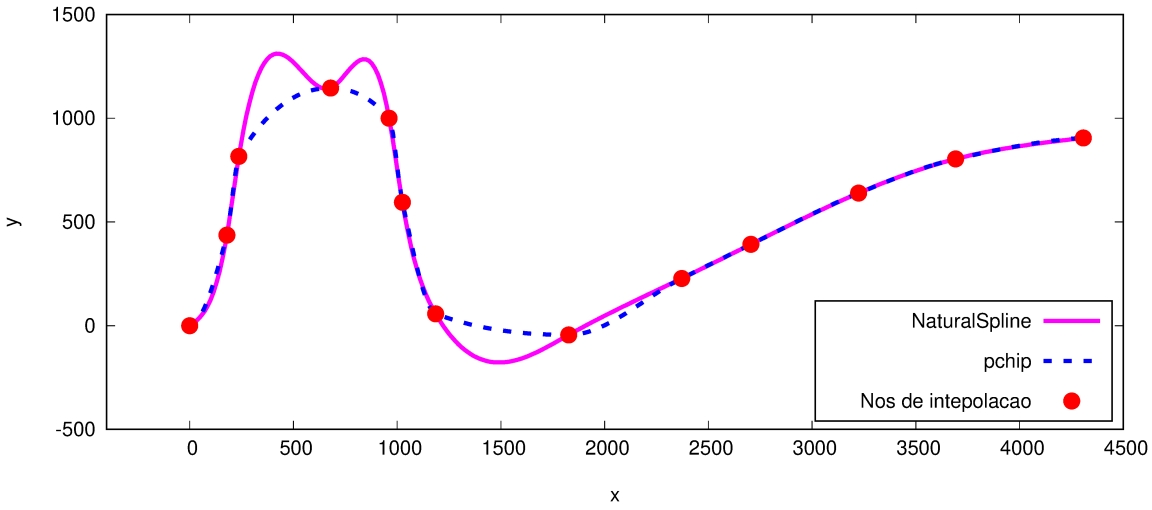
A princípio, as interpolações que poderíamos aplicar nesse conjunto de dados seriam: a interpolação de Lagrange, a interpolação de Hermite e a interpolação por spline cúbico. No entanto, a interpolação de Lagrange apresentou oscilações exageradamente grandes perto dos pontos onde seriam as últimas estações (mal era possível visualizar o traço da avenida na escala do gráfico do polinômio gerado), e por isso descartamo-la. Possivelmente, isso seria evitado se tivéssemos coletado as informações nos pontos de Chebyshev por meio de uma mudança de variáveis que mencionamos no final da Seção 3.1, minimizando o fenômeno de Runge. Quanto à interpolação de Hermite, teríamos que estimar numericamente as derivadas nos pontos para poder aplicar essa técnica de maneira geral no intervalo todo. Por isso, optou-se por substituí-la pela interpolação por partes via polinômios cúbicos de Hermite ("pchip").

Utilizando, então, as funções *pchip* e *NaturalSpline* (Código B.5), conseguimos ajustar os dados através das referidas interpolações. O resultado é mostrado na Figura 5.2. Comparando os dois ajustes, na Figura 5.3, é possível notar que a interpolação por partes via polinômios cúbicos de Hermite mostrou-se mais bem comportada. A interpolação por splines cúbicos não adequou-se tão bem quanto esperado quando comparado com o exemplo apresentado na seção de splines cúbicos do capítulo anterior (note que aquele ajuste ficou perfeito). A razão disso se deve ao fato de não termos selecionados uma quantidade maior de pontos em regiões próximas das grandes deformidades da avenida (regiões onde baixas curvaturas são encontradas). Nesse

caso, se usássemos pontos mais "juntos" onde ficam as curvas mais acentuadas da avenida, o ajuste via spline teria ficado com melhor precisão.

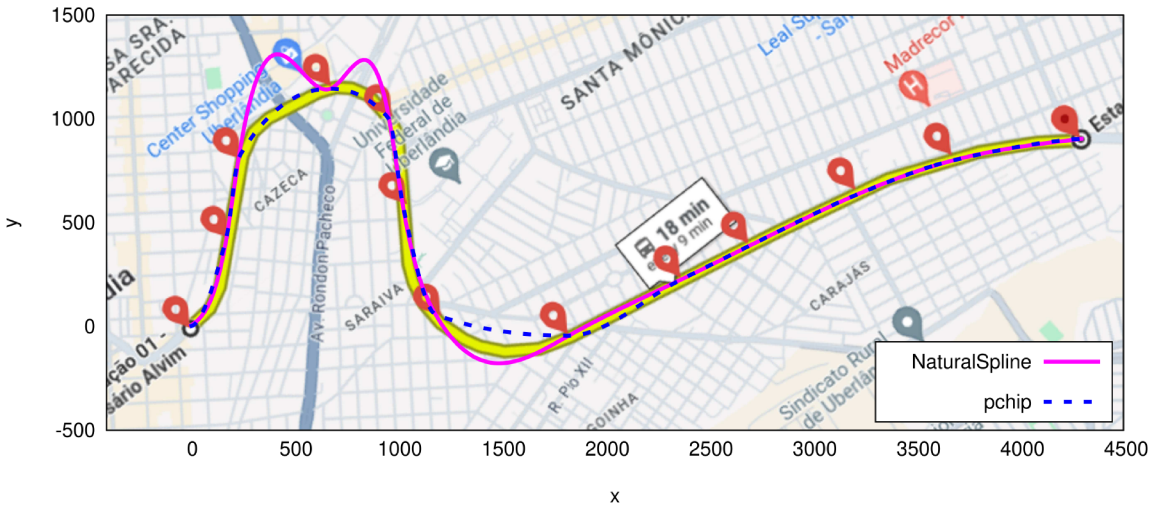
Um exercício extra, a quem estiver disposto, seria obter as expressões analíticas de cada parte desses ajustes (vimos como obter os coeficientes) e calcular, por exemplo, a extensão (comprimento de arco) da avenida utilizando integral de linha. Nesse caso, é possível fazer isso diretamente com integração numérica, adicionando apenas algumas linhas de comando no código que utilizamos para realizar as interpolações.

Figura 5.2: Aplicação - Resultado das interpolações por *pchip* e splines naturais.



Fonte: De autoria própria (elaborado através do gnuplot v5.4).

Figura 5.3: Análise comparativa - Adequação dos ajustes na Av. João Naves de Ávila.



Fonte: De autoria própria (elaborado através do gnuplot v5.4).

6. CONCLUSÃO

Neste trabalho exploramos a interpolação polinomial como uma ferramenta fundamental para a aproximação de funções em contextos teóricos e práticos. Através do estudo de métodos clássicos, como a interpolação de Lagrange e a interpolação de Newton, e abordagens mais avançadas, como a interpolação por Splines, pudemos compreender a complexidade e a versatilidade dessas técnicas na representação de dados e na aproximação de funções. Também foram implementadas algumas das técnicas, utilizando ferramentas computacionais como MATLAB e GNU Octave. Os ajustes obtidos nos exemplos demonstram a viabilidade das técnicas de interpolação polinomial para aproximar funções com diferentes graus de precisão, para uma variedade de conjuntos de dados, destacando as vantagens e desvantagens de cada método.

A análise detalhada das diferentes abordagens de interpolação polinomial, juntamente com a implementação dessas técnicas, permitiu uma compreensão mais aprofundada de como esses métodos podem ser aplicados em contextos reais, como fizemos no exemplo prático apresentado no Capítulo 5.

Além do exemplo que trouxemos ao final do trabalho, existem muitas outras aplicações geográficas que podem ser feitas com uso da interpolação polinomial, como análise de impactos ambientais em regiões florestais com focos de incêndio ou em vazamentos de óleo no mar, delimitação de fronteiras territoriais, entre outros. Além disso, destacamos a importância dessas técnicas em áreas como análise numérica, processamento de sinais e computação gráfica, onde a precisão na representação de funções é crucial para o sucesso das aplicações. Na verdade, a aplicação da interpolação polinomial em problemas reais abre um leque de possibilidades em outras áreas do conhecimento. Na engenharia, por exemplo, pode ser utilizada para estimar o comportamento de materiais sob diferentes condições. Na física, permite modelar trajetórias de corpos em movimento. Já na economia, auxilia na previsão de tendências de mercado.

Entretanto, é importante reconhecer que este estudo possui algumas limitações. O número de pontos de interpolação e a escolha do método podem influenciar na precisão dos resultados. Poderíamos explorar mais técnicas para otimizar a seleção de pontos, como a que mostramos no Capítulo 3, e desenvolver métodos híbridos que combinem as vantagens das diferentes técnicas. Por exemplo, existe uma conexão entre as técnicas de interpolação polinomial estudadas neste trabalho e as Séries de Fourier. Enquanto as técnicas de interpolação polinomial são capazes de aproximar funções num intervalo através de polinômios, as séries de Fourier oferecem uma abordagem global para representar funções periódicas através da combinação de senos e cossenos, excelente para descrever fenômenos oscilatórios e entender o comportamento de sistemas físicos

e naturais. Assim, as séries de Fourier complementam e enriquecem o estudo da interpolação polinomial, fornecendo ferramentas adicionais para análise e resolução de problemas.

Além disso, o exemplo que abordamos no Capítulo 5 despertou o interesse de utilizar as técnicas de interpolação polinomial em conjunto com Curvas Parametrizadas. Curvas parametrizadas, representadas por funções vetoriais, são essenciais para descrever trajetórias no espaço. Através delas podemos representar curvas gerais usando um parâmetro para expressar ambas as coordenadas x e y. No exemplo do Capítulo 5, tivemos que rotacionar a imagem da Av. João Naves de Ávila para que ela atingisse a forma característica de uma função real no plano, mas poderíamos ter aplicado a interpolação polinomial diretamente no mapa original (com norte fixado), utilizando curvas parametrizadas.

Assim, para trabalhos futuros, sugerimos explorar ainda mais a interpolação polinomial através da combinação dos conceitos mencionados acima, de forma que essa abordagem possa abrir novas possibilidades de modelagem e representação de situações complexas em áreas como design gráfico, animação computadorizada e gráficos 3D para jogos, simulação de sistemas dinâmicos, design industrial e robótica, entre outras.

Por fim, esperamos que as sugestões apresentadas aqui inspirem estudantes, pesquisadores e profissionais a explorar ainda mais esse campo fascinante da matemática aplicada, da mesma forma com que este trabalho contribuiu para expandir nosso entendimento sobre as técnicas de interpolação polinomial e seu papel na aproximação de funções.

Referências Bibliográficas

[1] BURDEN, R. L.; FAIRES, J. D.; BURDEN, A. M. *Análise Numérica. Tradução da 10ª edição norte-americana: All Tasks e Helena Maria Ávila de Castro*. 3. ed. São Paulo: Cengage, 2017. ISBN 978-85-221-2340-7.

[2] LIMA, E. L. *Análise Real: Funções de Uma Variável, vol. 1*. 12. ed. Rio de Janeiro: IMPA, 2017. ISBN 978-85-244-0048-3.

[3] SANTOS, R. J. *Teorema da Aproximação de Weierstrass*. ICEx-UFMG, 2010. Acesso em: 07 abr. 2024. Disponível em: <https://regijs.github.io/eqdif/teoaproximweierstrass.pdf>.

[4] BOLDRINI, J. L. et al. *Álgebra Linear*. 3. ed. São Paulo: Harper & Row do Brasil, 1980. ISBN 85-2940-202-2.

[5] MOLER, C. B. *Numerical Computing with MATLAB. Version 2.0*. [S.l.]: The MathWorks Inc, 2016. <https://doi.org/10.1137/1.9780898717952.fm>.

[6] REAMAT (org). *Cálculo Numérico - Um Livro Colaborativo*. IME-UFRGS, 2020. Acesso em: 31 jan. 2024. Disponível em: <https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/main.html>.

[7] FRANCO, N. B. *Cálculo numérico*. São Paulo: Pearson Prentice Hall, 2006. ISBN 85-7605-087-0.

[8] QUARTERONI, A.; SALERI, F. *Cálculo Científico com MATLAB e Octave. Tradução da obra italiana: Adélia Sequeira*. Milano, Italia: Springer, 2007. <https://doi.org/10.1007/978-88-470-0718-5>.

[9] POWELL, M. J. D. *Approximation theory and methods*. Cambridge: Cambridge University Press, 1981. <https://doi.org/10.1017/CBO9781139171502>.

[10] PPGMAT (org). *Análise Numérica: Splines cúbicos*. FAMAT-UFU, 2023. Acesso em: 29 mar. 2024. Disponível em: <https://heyzine.com/flip-book/53cd1a668b.html>.

A. Resultados Complementares

A.1 Independência da ordenação de pontos (enésima diferença dividida)

Proposição A.1. *Sejam $n+1$ números distintos, $x_0, x_1, \ldots, x_n$, e f uma função cujos valores são dados por esses números. Seja também $f[x_0, x_1, \ldots, x_n]$ a enésima diferença dividida com relação a $x_0, x_1, \ldots, x_n$. Se $i_0, i_1, \ldots, i_n$ é uma reordenação dos números inteiros $0, 1, \ldots, n$, então $f[x_{i_0}, x_{i_1}, \ldots, x_{i_n}] = f[x_0, x_1, \ldots, x_n]$.*

Demonstração:

Dados $x_0, x_1, \ldots, x_n$ e $f(x_0), f(x_1), \ldots, f(x_n)$. Se expandirmos o polinômio interpolador na forma de Newton $p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\ldots(x - x_{n-1})$, veremos que o coeficiente dominante (o coeficiente que acompanha x^n) é exclusivamente o a_n, enquanto os coeficientes das outras potências podem ser combinações lineares diversas. Isso significa que independente da ordenação dos pontos $x_0, x_1, \ldots, x_n$, ao realizarmos a interpolação pelo método de Newton, o coeficiente a_n que acompanha $\phi_n(x) = \prod_{k=0}^{n-1}(x - x_k)$ sempre será o mesmo, caso contrário, pelo que acabamos de observar, obteríamos polinômios diferentes, o que é impossível.

Por outro lado, sabemos que o coeficiente a_n da interpolação é exatamente $a_n = f[x_0, x_1, \ldots, x_n]$. Daí, se $i_0, i_1, \ldots, i_n$ é uma reordenação dos números inteiros $0, 1, \ldots, n$ e

$$p_n(x) = \overline{a_0} + \overline{a_1}(x - x_{i_0}) + \overline{a_2}(x - x_{i_0})(x - x_{i_1}) + \cdots + \overline{a_n}(x - x_{i_0})\ldots(x - x_{i_{n-1}})$$

a interpolação de Newton referente a $x_{i_0}, x_{i_1}, \ldots, x_n$ e $f(x_{i_0}), f(x_{i_1}), \ldots, f(x_{i_n})$, então $f[x_{i_0}, x_{i_1}, \ldots, x_{i_n}] = \overline{a_n} = a_n = f[x_0, x_1, \ldots, x_n]$.

□

A.2 Estimação da enésima derivada através da enésima diferença dividida

Teorema A.1. *Sejam $f \in C^n$, definida em $[a, b]$, e $a = x_0, x_1, \ldots, x_n = b$ números distintos em $[a, b]$. Então, existe um número ξ em (a, b) tal que*

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!} \ .$$

Demonstração:

Defina

$$g(x) = f(x) - p_n(x),$$

onde $p_n(x)$ é o polinômio interpolador de $x_0, x_1, \ldots, x_n$ (equação 2.9).

Como $f(x_i) = p_n(x_i)$, para cada $i = 0, 1, \ldots, n$, a função g possui $n+1$ zeros distintos em $[a, b]$. O teorema generalizado de Rolle implica que existe um número ξ em (a, b) com $g^{(n)}(\xi) = 0$. Então

$$0 = g^{(n)}(\xi) = f^{(n)}(\xi) - p_n^{(n)}(\xi) \ .$$

Como $p_n(x)$ é um polinômio de grau n cujo coeficiente dominante é $f[x_0, x_1, \ldots, x_n]$, segue que

$$p_n^{(n)}(x) = n! \ f[x_0, x_1, \ldots, x_n] \ ,$$

para todos os valores de x. Como consequência,

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!} \ .$$

□

A.3 Existência e unicidade do polinômio interpolador (outras formas)

Proposição A.2 (Forma de Lagrange)**.** *Seja* $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ *um conjunto de* $n+1$ *pares ordenados de números reais tais que* $x_i \neq x_j$*, se* $i \neq j$*, então existe um único polinômio* $p_n(x)$ *de grau no máximo* n *que passa por todos os pontos dados, isto é,* $p_n(x_i) = y_i$*,* $i = 0, \ldots, n$*.*

Demonstração:

Partindo de $x_0, x_1, \ldots, x_n$, ao definir as funções linearmente independentes $\phi_j(x) = L_j(x) = \dfrac{\prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i)}{\prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i)}$, $j = 0, 1, \ldots, n$; o problema de encontrar os coeficientes a_j tais que $p_n(x_i) = \sum_{j=0}^{n} a_j \phi_j(x_i) = y_i$ é o mesmo que resolver o sistema linear:

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} .$$

Mas, como $\phi_j(x_i) = L_j(x_i) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$, então na verdade a matriz das funções de base torna-se uma matriz identidade:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Assim, esse sistema linear possui solução única, sendo os coeficientes $a_0, a_1, \ldots, a_n$ os próprios $y_0, y_1, \ldots, y_n$, respectivamente. Logo, o polinômio $p_n(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2) + \cdots + L_n(x)f(x_n)$ interpola devidamente os dados e é único.

□

Proposição A.3 (Forma de Newton)**.** *Seja* $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ *um conjunto de* $n+1$ *pares ordenados de números reais tais que* $x_i \neq x_j$*, se* $i \neq j$*, então existe um único polinômio* $p_n(x)$ *de grau no máximo* n *que passa por todos os pontos dados, isto é,* $p_n(x_i) = y_i$*,* $i = 0, \ldots, n$*.*

Demonstração:

Considere o polinômio $p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\ldots(x - x_{n-1})$. Afirmamos que, para constantes $a_0, a_1, \ldots, a_n$ adequadas, esse polinômio $p_n(x)$ interpola os dados e é único.

De fato, procedendo de maneira análoga ao que fizemos na Proposição A.2, temos que $\phi_0(x) = 1$ e $\phi_j(x) = \prod_{i=0}^{j-1}(x - x_i)$, $j = 1, \ldots, n$. Não é difícil ver que as funções $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ definidas dessa maneira são linearmente independentes. Além disso, perceba que $\phi_j(x_i) = 0$ sempre que $j > i$. Assim, ao esquematizar a resolução do sistema linear para interpolação dos dados, temos em forma matricial:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & \phi_1(x_1) & 0 & \cdots & 0 \\ 1 & \phi_1(x_2) & \phi_2(x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Desse modo, a matriz correspondente às funções de base constituem uma matriz triangular inferior, em que todos os elementos da diagonal principal são diferentes de zero. Logo, o determinante nesse caso é diferente de zero. Ou seja, a solução desse sistema existe e é única. O resultado segue. Além disso, conseguimos resolver esse sistema linear facilmente, bastando para isso realizar sucessivas substituições a partir da primeira linha.

□

A.4 OUTROS RESULTADOS

Teorema A.2 (Teorema de Rolle)**.** *Seja $f : [a, b] \to \mathbb{R}$ uma função contínua em seu domínio e diferenciável em (a, b). Se $f(a) = f(b)$, então existe um número c em (a, b) tal que $f'(c) = 0$.*

Teorema A.3 (Teorema de Rolle generalizado)**.** *Seja $f : [a, b] \to \mathbb{R}$ uma função contínua em seu domínio e n vezes diferenciável em (a, b). Se $f(x) = 0$ em $n + 1$ números distintos $a \leq x_0 < x_1 < \cdots < x_n \leq b$, então existe um número c em (x_0, x_n), e portanto em (a, b), tal que $f^{(n)}(c) = 0$.*

Teorema A.4 (Teorema do valor médio)**.** *Se $f : [a, b] \to \mathbb{R}$ for uma função contínua em seu domínio e diferenciável em (a, b), então existe um número c em (a, b) tal que*

$$f'(c) = \frac{f(b) - f(a)}{b - a} \ .$$

Teorema A.5 (Teorema do valor intermediário)**.** *Seja $f : [a, b] \to \mathbb{R}$ uma função contínua. Se d é um número entre $f(a)$ e $f(b)$, então existe um número $c \in (a, b)$ tal que $f(c) = d$.*

B. Algoritmos Implementados

B.1 Interpolações clássicas

Interpolação de Lagrange (ilustrativo)

```
% EXEMPLO GNU Octave

% Funcao para calcular os polinomios de Lagrange
function L = lagrange_polynomial(x, k, u)
    n = length(x);
    L = ones(size(u));
    for j = 1:n
        if j ~= k
            L = L .* (u - x(j)) / (x(k) - x(j));
        end
    end
end

% Pontos de interpolacao
x = [1, 2, 3, 4];
y = [4, 3, 2, 1];

% Pontos onde a funcao sera avaliada
u = linspace(1, 4, 100);

% Interpolacao de Lagrange
v = zeros(size(u));
for k = 1:length(x)
    v = v + y(k) * lagrange_polynomial(x, k, u);
end

% Plotar os resultados
plot(x, y, 'o', u, v);
title('Interpolacao de Lagrange');
xlabel('x');
```

```
ylabel('y');
```

Código B.1: Exemplo de interpolação via forma de Lagrange.

INTERPOLAÇÃO DE NEWTON (ILUSTRATIVO)

```
% EXEMPLO GNU Octave

% Funcao para calcular as diferencas divididas
function div_diff = divided_differences(x, y)
    n = length(x);
    div_diff = zeros(n, n);
    div_diff(:,1) = y(:);
    for j = 2:n
        for i = 1:n-j+1
            div_diff(i,j) = (div_diff(i+1,j-1) - div_diff(i,j-1)) /
            (x(i+j-1) - x(i));
        end
    end
end

% Funcao para calcular o polinomio interpolador de Newton
function p = newton_interpolation(x, y, u)
    n = length(x);
    div_diff = divided_differences(x, y);
    p = div_diff(1,1);
    for j = 2:n
        term = 1;
        for i = 1:j-1
            term = term .* (u - x(i));
        end
        p = p + div_diff(1,j) * term;
    end
end

% Pontos de interpolacao
x = [1, 2, 3, 4];
y = [4, 3, 2, 1];

% Pontos onde a funcao sera avaliada
u = linspace(1, 4, 100);

% Interpolacao de Newton
```

```
v = newton_interpolation(x, y, u);

% Plotar os resultados
plot(x, y, 'o', u, v);
title('Interpolacao de Newton');
xlabel('x');
ylabel('y');
```

Código B.2: Exemplo de interpolação via forma de Newton.

B.2 Minimização do erro

Pontos de Chebyshev (modelo)

```
% EXEMPLO GNU Octave

% Definir a funcao f(x)
f = @(x) 1./(1+25*x.^2);

% Gerar os pontos para interpolacao de Chebyshev
k = 0:10;
x = cos(((2*k + 1) / 22) * pi);
y = f(x);

% Interpolacao de Lagrange usando polyfit
degree = 10; % Grau do polinomio
coefficients = polyfit(x, y, degree);

% Imprimir os valores dos coeficientes
printf('Coeficientes do polinomio interpolador:\n');
printf('%f\n', coefficients);

% Avaliar o polinomio interpolador nos pontos x
P = polyval(coefficients, x);

% Gerar pontos para o plot suave da funcao
x_smooth = linspace(-1, 1, 1000);
f_smooth = f(x_smooth);

% Plotar grafico
figure('Position', [100, 100, 800, 750]); % Definir tamanho da figura
plot(x_smooth, f_smooth, 'k', 'LineWidth', 1.5, 'DisplayName',
'Funcao f(x) original'); % Plotar funcao f
```

```
hold on;
% Definir o tamanho dos pontos de dispersao
scatter(x, y, 35, 'b', 'filled', 'DisplayName', 'Nos_de_Chebyshev');
% Plotar polinomio interpolador
plot(x_smooth, polyval(coefficients, x_smooth), 'r', 'LineWidth', 1.8,
'DisplayName', 'Polinomio_interpolador_p_{10}(x)');

% Adicionar grade
grid on;
grid minor;

% Titulo e rotulos dos eixos
title('Ajuste_polinomial_de_grau_10');
xlabel('x');
ylabel('y');
```

Código B.3: Exemplo de interpolação utilizando os pontos de Chebyshev como nós.

B.3 OUTRAS INTERPOLAÇÕES

INTERPOLAÇÃO DE HERMITE

```
%-------------------------------------------------------------------
% Universidade Federal de Uberlandia
% Faculdade de Matematica
% Data: 05/10/23
% Autores: Heitor Pereira e Rafael Figueiredo
% Interpolacao Polinomial de Hermite via diferencas divididas
% Algoritmo baseado em:
% Burden et. al, Analise Numerica, 10a Ed. (2017)
% Pagina 155, Algoritmo 3.3
%-------------------------------------------------------------------
% IMPLEMENTACAO GNU Octave
%-------------------------------------------------------------------
clear all; % Limpa a memoria
clc;       % Limpa a tela
%-------------------------------------------------------------------

% Interpolacao de Hermite via Diferencas Divididas

function H = HermiteFit(x, f, dfdx)

  % x: vetor com n+1 pontos distintos no intervalo [a,b]
```

```
22    % f: vetor dos valores da funcao f(x)
23    % dfdx: vetor dos valores da derivada da funcao f em x
24    % Q: vetor das diferencas para aproximar polinomio de Hermite
25
26    n = length(x)-1; % x numero de pontos da interpolacao
27
28    % Passo 1
29    for i = 1:n+1
30
31      % Passo 2
32      z(2*i-1)    = x(i);
33      z(2*i)      = x(i);
34      Q(2*i-1,1) = f(i);
35      Q(2*i,1)    = f(i);
36      Q(2*i,2)    = dfdx(i);
37
38      % Passo 3
39      if ( i == 1 )
40        continue;
41      endif
42
43      Q(2*i-1,2) = (Q(2*i-1,1)-Q(2*i-2,1))/(z(2*i-1)-z(2*i-2));
44    endfor
45
46    % Passo 4
47    for i = 3:2*n+2
48      for j = 3:i
49        Q(i,j) = (Q(i,j-1)-Q(i-1,j-1))/(z(i)-z(i-j+1));
50      endfor
51    endfor
52
53      Q
54      for i = 1:2*n+2
55        H(i,1) = Q(i,i);
56      endfor
57  endfunction
58  %-----------------------------------------------------------------------
59
60  function y = HermiteVal(H,x,a)
61
62    % y: valor do polinomio de Hermite avaliado no ponto a
63    % H: Estimativa das diferencas divididas da interpolacao de Hermite
64    % x: vetor dos pontos distintos de interpolacao
```

```
  % a: valor a ser avaliado no polinomio de Hermite

 n = length(H); % Numero de coeficientes de Hermite
 k = length(x);

  for i = 1:k
      z(2*i-1)    = x(i);
      z(2*i)      = x(i);
  endfor

  Hermite = sprintf("@(x)"); % String do polinomio de Hermite

  for i = 1:n
   Hermite = sprintf("%s+H(%i)", Hermite, i);
    for j = 2:i
      Hermite = sprintf("%s.*(x-z(%i-1))", Hermite, j);
    endfor
  endfor

  Hermite = str2func(Hermite); % Converte string para funcao
  y = Hermite(a);
endfunction
%-----------------------------------------------------------------------

% Exemplo 1, Burden, p. 151
x = [1.3; 1.6; 1.9]
f = [0.620086; 0.4554022; 0.2818186]
dfdx = [-0.5220232; -0.5698959; -0.5811571]

H = HermiteFit(x, f, dfdx)
HermiteVal(H,x,1.5)

% Construcao do grafico
xx = linspace(1.2,2,200); % Vetor com 200 pontos espacados em [1.2,2]
yy = HermiteVal(H,x,xx);  % Avalia o Polinomio de Hermite no vetor xx

figure; % Abre a figura
plot(x,f, ".r", "MarkerSize", 25); % plot dos pontos de interpolacao
hold on; % Mantem o grafico aberto
plot(xx,yy, "LineWidth", 3); % plot do ajuste via polinomio de Hermite
hold off;
```

Código B.4: Interpolação de Hermite usando diferenças divididas (com um exemplo).

Interpolação por spline cúbico

```
%-------------------------------------------------------------------------
% Universidade Federal de Uberlandia
% Faculdade de Matematica
% Data: 16/11/23
% Autores: Heitor Pereira e Rafael Figueiredo
% Interpolacao Polinomial: Splines Naturais
% Algoritmo baseado em:
% Burden et. al, Analise Numerica, 10a Ed. (2017)
% Pagina 164, Algoritmo 3.4
%-------------------------------------------------------------------------
clear all; % Limpa a memoria
clc;       % Limpa a tela
%-------------------------------------------------------------------------

function s = NaturalSpline(x,a,pts)

  % s: valores avaliados na Spline cubica natural nos pontos pts
  % (x_i,a_i=f(x_i)): pontos utilizados na interpolacao
  % pts: pontos a serem avaliados no ajuste via Spline

  % S(x) = S_j(x) = a_j + b_j*(x-x_j) + c_j*(x-x_j)^2 + d_j*(x-x_j)^3,
  %para x_j <= x <= x_{j+1}
  % Satisfazendo S"(x_0)=S"(x_n) = 0

  n = length(x)-1; % Numero de pontos da interpolacao menos 1

  % Passo 1
  for i = 1:n
    h(i) = x(i+1)-x(i);
  endfor

  % Passo 2
  for i = 2:n
    alpha(i) = 3*(a(i+1)-a(i))/h(i) - 3*(a(i)-a(i-1))/h(i-1);
  endfor

  % Passos 3,4,5 e parte do 6 resolvem um sistema linear tridiagonal,
  %usando
  % o metodo descrito no algoritmo 6.7, pg. 468.
  % Fatoracao de Crout para sistemas lineares tridiagonais

```

```
42    % Passo 3
43    l(1)  = 1;
44    mu(1) = 0;
45    z(1)  = 0;
46
47    % Passo 4
48    for i = 2:n
49      l(i)  = 2*(x(i+1)-x(i-1))-h(i-1)*mu(i-1);
50      mu(i) = h(i)/l(i);
51      z(i)  = (alpha(i)-h(i-1)*z(i-1))/l(i);
52    endfor
53
54    % Passo 5
55    l(n+1) = 1;
56    z(n+1) = 0;
57    c(n+1) = 0;
58
59    % Passo 6
60    for j = n:-1:1
61      c(j) = z(j) - mu(j)*c(j+1);
62      b(j) = (a(j+1)-a(j))/h(j) - h(j)*(c(j+1)+2*c(j))/3;
63      d(j) = (c(j+1)-c(j))/(3*h(j));
64    endfor
65
66    % Passo 7
67    for j = 1:n
68      printf("j=%i,\ta_%i=%f,b_%i=%f,c_%i=%f,d_%i=%f\n",j-1,j-1,a(j),j
          -1,b(j),j-1,c(j),j-1,d(j));
69    endfor
70
71    % Avaliar os pontos pts no ajuste da spline
72    for i = 1:length(pts)
73
74          j = 1;
75          while ( pts(i) - x(j+1) > 0 )
76            j++;
77            if ( j == n ) break; endif
78          endwhile
79
80          s(i,1) = a(j) + (pts(i)-x(j))*b(j) + (pts(i)-x(j))^2 * c(j) +
              (pts(i)-x(j))^3 * d(j);
81    endfor
82  endfunction
```

```
83 %------------------------------------------------------------------------
84
85 %Exemplo: Tabela 3.18, pg. 172
86 x =
       [0.9;1.3;1.9;2.1;2.6;3.0;3.9;4.4;4.7;5;6;7;8;9.2;10.5;11.3;11.6;12;
87 12.6;13;13.3];
88 y =
       [1.3;1.5;1.85;2.1;2.6;2.7;2.4;2.15;2.05;2.1;2.25;2.3;2.25;1.95;1.4;
89 0.9;0.7;0.6;0.5;0.4;0.25];
90
91 figure;
92 plot(x,y,".r","MarkerSize", 20);
93 hold on;
94 xx = linspace(min(x),max(x),100);
95 plot(xx,NaturalSpline(x,y,xx),"LineWidth",2);
96 legend({'(x,y)', 'NaturalSpline'},'Location','southwest')
```

Código B.5: Interpolação via spline cúbico natural (com um exemplo).