ESCOLA DE DOUTORAMENTO
INTERNACIONAL DA USC

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio
Preto

Lourenço
de Lima Peixoto

Tese de doutoramento

UM ALGORITMO EFICIENTE PARA APROXIMAÇÃO PELO MÉTODO DOS MÍNIMOS QUADRADOS

Santiago de Compostela, 2021
São José do Rio Preto, 2021

Programa de Doutoramento en Matemáticas

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Lourenço de Lima Peixoto

UM ALGORITMO EFICIENTE PARA APROXIMAÇÃO PELO MÉTODO DOS MÍNIMOS QUADRADOS

São José do Rio Preto
2021

Lourenço de Lima Peixoto

UM ALGORITMO EFICIENTE PARA APROXIMAÇÃO PELO MÉTODO DOS MÍNIMOS QUADRADOS

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto, em cotutela com a Universidade de Santiago de Compostela.

Orientador: Prof. Dr. Dimitar Kolev Dimitrov
Co-orientador: Prof. Dr. Juan José Nieto Roig

Financiadora: CAPES

São José do Rio Preto
2021

Peixoto, Lourenço de Lima.
Um algoritmo eficiente para aproximação pelo método dos mínimos quadrados / Lourenço de Lima Peixoto. -- São José do Rio Preto, 2021
85 p. : il, tabs.

Orientador: Dimitar Kolev Dimitrov
Co-orientador: Juan José Nieto Roig
Tese (doutorado com dupla titulação) – Universidade Estadual Paulista (Unesp), Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto e Universidade de Santiago de Compostela

1. Matemática. 2. Teoria da aproximação. 3. Mínimos quadrados – Método. 4. Quadratura gaussiana. I. Título.

CDU – 518.7

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Lourenço de Lima Peixoto

UM ALGORITMO EFICIENTE PARA APROXIMAÇÃO PELO MÉTODO DOS MÍNIMOS QUADRADOS

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto, em cotutela com a Universidade de Santiago de Compostela.

Financiadora: CAPES

Comissão Examinadora

Prof. Dr. Dimitar Kolev Dimitrov
Departamento de Matemática - UNESP - Câmpus de São José do Rio Preto
Orientador

Prof. Dr. Juan José Nieto Roig
Departamento de Matemáticas - Universidade de Santiago de Compostela
Co-orientador

Prof. Dr. Antonio Carlos Gardel Leitão
Departamento de Matemática - Universidade Federal de Santa Catarina

Prof. Dr. Carlos Roberto Valêncio
DCCE - UNESP - Câmpus de São José do Rio Preto

Prof. Dr. Roberto Andreani
Departamento de Matemática Aplicada - Universidade Estadual de Campinas

São José do Rio Preto
5 de maio de 2021

AGRADECIMENTOS

Agradeço a todos aqueles que estiveram comigo durante essa longa jornada!

A todos os professores do IBILCE, especialmente ao professor Dimitar pelo incentivo e pela dedicação de sempre. Também agradeço aos professores Ranga e Cleonice e aos demais colegas do Grupo de Pesquisa em Polinômios Ortogonais. Agradeço também aos professores Nieto e Iván pela minha estância na Galícia.

Agradeço a todos amigos de Rio Preto e de Santiago de Compostela, especialmente à Laura, Ana Lívia, Eliton, Frank, Rodrigo, Venktesh, Sebastián, Isaac, Juan, Jorge, Javier e Jesús.

A todos meus familiares, em especial à minha mãe, à minha irmã, à Heleninha e ao Lázaro.

Ao Instituto Federal de Minas Gerais *Campus* Congonhas pelo afastamento das minhas atividades docentes, para realização do doutorado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"Nossa missão central é
calcular quantidades tipicamente incalculáveis,
de um ponto de vista analítico,
e realizá-las na velocidade da luz."
Lloyd N. Trefethen [1]

RESUMO

Esta tese tem o propósito de apresentar um novo método eficiente para a aproximação clássica pelos mínimos quadrados para grande quantidade de dados, desenvolvido e implementado em softwares que funcionam com suporte à precisão dupla. O método é baseado na expansão de Fourier da solução com respeito à base ortogonal composta pelos polinômios de Gram e no cálculo dos coeficientes de Fourier via uma fórmula de quadratura Gaussiana. Todas as características importantes são analisadas e discutidas em detalhes profundos. Comparações extensas mostram que geralmente o novo método é mais estável e rápido do que os demais disponíveis na literatura.

Palavras-chave: Problema de aproximação por mínimos quadrados, Melhor aproximação, Base polinomial de Gram, Fórmula de quadratura gaussiana.

ABSTRACT

The purpose of this thesis is to describe a new efficient method for the classical least squares approximation of a large quantity of data, developed and implemented in a software that works in double precision. The method is based on the Fourier expansion of the solution in terms of the orthogonal basis of Gram polynomials and on the calculation of the corresponding Fourier coefficients via a Gaussian type quadrature formula. All the important features are analysed and discussed in deep details. Extensive comparisons show that in general the new method is more precise and faster than the ones available in the literature.

Keywords: Least squares approximation problem, Best approximation, Gram polynomials basis, Gaussian quadrature formula.

Lista de Figuras

Lista de Tabelas

Sumário

Introdução

O principal objetivo da presente tese é apresentar um algoritmo eficiente e rápido para o cálculo preciso do polinômio algébrico, que melhor aproxima grandes quantidades de dados pelo método mais utilizado nas ciências, chamado método dos mínimos quadrados.

A formulação matemática do problema é bem conhecida: dados N pontos distintos x_j da reta real e N números reais y_j, $j = 1, \ldots, N$, em que os últimos podem ser considerados valores de uma função f, definida nos pontos x_j, isto é $y_j = f(x_j)$, deve-se encontrar um polinômio algébrico real p de grau, no máximo n, que minimiza a quantidade

$$\sum_{j=1}^{N} \left[y_j - p(x_j) \right]^2 ,$$

entre todos os polinômios de grau n. Em português, esse problema de aproximação é conhecido como "método dos mínimos quadrados". Com o propósito de evitar ambiguidade, será referenciado ao problema como o "problema clássico dos mínimos quadrados" (PCMQ), principalmente pelo fato de existirem vários métodos para sua solução. Uma vez que o problema possui uma interpretação natural para a aproximação em espaços de Hilbert, sua solução existe, é única e é fornecida pela "projeção".

A versão mais utilizada em problemas de diversas ciências ocorre quando os pontos são equidistantes e a quantidade N de dados mostra-se relativamente muito maior do que o grau n do polinômio que resolve o problema, e é essa a versão do PCMQ que é estudada na tese. Uma vez que quaisquer pontos equidistantes podem ser levados ao intervalo $[-1, 1]$ por meio de uma transformação afim, fixamos

$$x_j(N) = -1 + \frac{2j-1}{N}, \quad j = 1, \ldots, N, \tag{1}$$

e definimos o produto interno

$$\langle f_1, f_2 \rangle = \frac{1}{N} \sum_{j=1}^{N} f_1(x_j(N))\, f_2(x_j(N)), \tag{2}$$

que, por sua vez, define a norma induzida

$$\|f\|^2 = \frac{1}{N} \sum_{j=1}^{N} f^2(x_j(N)). \tag{3}$$

Denotando por $\mathbb{P}_n$ o espaço linear dos polinômios algébricos, cujos graus não excedem n, o objetivo do PCMQ é determinar o polinômio $p_n(x; f) \in \mathbb{P}_n$ que satisfaz

$$\|f - p_n(\cdot\,; f)\|_2^2 = \min_{p \in \mathbb{P}_n} \|f - p\|_2^2 = \min_{p \in \mathbb{P}_n} \sum_{j=1}^{N} \left(y_j - p(x_j) \right)^2 , \tag{4}$$

no senso da norma ℓ^2 usual, isto é,

$$\|u\|_2^2 = \sum_{j=1}^{N} u^2(x_j(N)).$$

Pela teoria da aproximação em espaços de Hilbert, a única solução $p_n(\cdot\,; f)$ é dada pela projeção ortogonal, isto é, pelo polinômio que satisfaz

$$\langle f - p_n(\cdot\,; f),\, p\rangle \;=\; 0$$

para todo $p \in \mathbb{P}_n$. Essa condição implica que, se $e_0(x), \ldots, e_n(x)$ é qualquer base de $\mathbb{P}_n$, então

$$p_n(x; f) = a_0\, e_0(x) + \cdots + a_n\, e_n(x),$$

onde o vetor dos coeficientes $\mathbf{a} = [a_0, \ldots, a_n]^{\mathrm{T}}$ é a única solução do sistema linear

$$\mathbf{G}(e_0, \ldots, e_n)\ \mathbf{a} = \mathbf{h},$$

onde $\mathbf{G}(e_0, \ldots, e_n) = (\langle e_i, e_k\rangle)$ é a matriz de Gram dos vetores $e_0, \ldots, e_n$ e $\mathbf{h}$ é o vetor-coluna com entradas $\langle f, e_k\rangle$, $k = 0, \ldots, n$.

Naturalmente todos os métodos conhecidos para a solução do PCMQ são baseados na solução do último sistema linear. Entretanto, tais métodos possuem várias deficiências, algumas oriundas do fato de que a matriz de Gram não é bem condicionada, sobretudo quando a base coincide com as potências da variável, isto é, quando $e_k(x) = x^k$.

Contudo, se a base $e_0(x), \ldots, e_n(x)$ é ortonormal com respeito ao produto interno, a matriz de Gram se reduz a uma múltipla da matriz identidade, e, por isso, a solução do sistema é

$$p_n(x; f) = \langle f, e_0\rangle\, e_0(x) + \ldots + \langle f, e_n\rangle\, e_n(x).$$

Mas os polinômios $G_k(\cdot\,; N)$, $k = 0, \ldots, N-1$, ortonormais com respeito ao produto interno acima, são conhecidos como polinômios discretos de Chebyshev ou, até mais comumente, como polinômios de Gram, representados pela função hipergeométrica da forma

$$\begin{aligned} G_k(x; N) &= (-1)^k \sqrt{\frac{(2k+1)(N-k)_k}{(N+1)_k}}\, {}_3F_2\left(\begin{matrix} -k, k+1, (1-N-Nx)/2 \\ 1, 1-N \end{matrix}\;\middle|\; 1\right) \\ &= (-1)^k \sqrt{\frac{(2k+1)(N-k)_k}{(N+1)_k}} \sum_{\nu=0}^{k} \frac{(-k)_\nu\, (k+1)_\nu\, ((1-N-Nx)/2)_\nu}{(\nu!)^2\, (1-N)_\nu}, \end{aligned} \tag{5}$$

onde o símbolo de Pochhammer é definido por $(x)_\nu = x(x+1)\cdot\ldots\cdot(x+\nu-1)$, $\nu \geq 1$, e $(x)_0 := 1$. Com essas notações, a solução do PCMQ é escrita explicitamente na forma

$$p_n(x; f) = \sum_{k=0}^{n} a_k\, G_k(x; N), \tag{6}$$

onde os coeficientes de Fourier a_k são definidos por

$$a_k = \langle f, G_k(\cdot\,; N)\rangle = \frac{1}{N} \sum_{j=1}^{N} f(x_j(N))\ G_k(x_j(N); N). \tag{7}$$

Portanto, a principal dificuldade para construir $p_n(x; f)$ explicitamente é calcular as somas (7) quando o número N dos seus termos é excessivo. Uma forma viável para obter

os coeficientes de Fourier (7) é usar uma "fórmula de quadratura" do tipo Gauss para somas da forma

$$\frac{1}{N}\sum_{j=1}^{N} F(x_j) \approx \sum_{k=1}^{m} W_{m,k}\, F(g_{m,k}(N)), \qquad m \ll N, \tag{8}$$

cujos nós $g_{m,k}(N)$ coincidem com os zeros do polinômio de Gram $G_m(\cdot\,; N)$. Observa-se o papel fundamental e "duplo" dos polinômios de Gram na solução do PCMQ.

Apesar da ideia de calcular a solução do PCMQ por meio da expansão (6), bem como o uso da fórmula de Gauss (8) para o cálculo dos coeficientes de Fourier, serem muito naturais, eles foram sugeridos e desenvolvidos bem recentemente, nos artigos [2, 3]. A razão por não ter existido um algoritmo para a aproximação pelo método dos mínimos quadrados, baseado nessas ideias, antes de [2,3], é que, para que ele funcionasse, seriam necessárias análises rigorosas de vários outros métodos correlacionados. Houve, por exemplo, um grande desafio para desenvolver um algoritmo verdadeiramente eficiente para a construção de (8). Além disso, a difícil tarefa de calcular aproximadamente os zeros de $G_m(\cdot\,; N)$ exigiu resultados teóricos sobre estimativas precisas para esses zeros, obtidas em [2] e a convergência do chamado método de Weierstrass-Dochev-Durand-Kerner (WDDK), que foi provada em [3].

Todas as ideias sugeridas e desenvolvidas em [3] funcionam com eficiência razoável na prática. Entretanto, a ênfase em [3] é principalmente na análise dos métodos e algoritmos propostos do ponto de vista teórico. Os cálculos em [3] são feitos no software Mathematica com precisão variável. Nesta tese são oferecidos vários melhoramentos do algoritmo desenvolvido em [3], é construído e desenvolvido um algoritmo praticamente novo, que pode ser facilmente implementado em várias linguagens de programação, pois trabalhamos com dupla precisão. Nossos algoritmos estão implementados em MATLAB.

Outra tarefa importante cumprida é a comparação rigorosa de todos os métodos específicos empregados, bem como o algoritmo como um todo, com relação aos mais eficientes utilizados na literatura até então. Como resultado dessa análise comparativa, afirma-se com toda a certeza que são utilizados os métodos mais eficientes em todas as etapas. O resultado final é um algoritmo para a aproximação pelo método dos mínimos quadrados que, na grande maioria dos casos, é mais rápido e com desempenho superior aos demais disponíveis.

Os resultados do algoritmo e a análise comparativa já foram publicados no artigo:

Dimitar K. Dimitrov e Lourenço L. Peixoto, An efficient algorithm for the classical least squares approximation, *SIAM Journal on Scientific Computing*, 42 (2020), A3233-A3249.

Vale mencionar que a revista é uma das mais conceituadas sobre métodos e algoritmos da Análise Numérica e Computação. Um reconhecimento especial foi o fato de a revista ter publicado nosso artigo apenas uma semana após as correções dos chamados "galley-proofs".

Todos os códigos do algoritmo estão disponíveis na página:
`https://clsap.dcce.ibilce.unesp.br`.

Para exemplificar o funcionamento extraordinário do algoritmo, é fornecido o gráfico que mostra o polinômio de grau $n = 12$ (em azul) que resolve o PCMQ, para $N = 10^8$ (cem milhões) de dados que são perturbações aleatórias de uma função suave. Mais precisamente,

$$f(x_j) = e^{x_j} \operatorname{sen} 12x_j + \delta_{10^8}(j),$$

onde $x_j = -1 + (2j-1)/N$ e δ_N retorna N números pseudoaleatórios pela distribuição normal, em MATLAB.

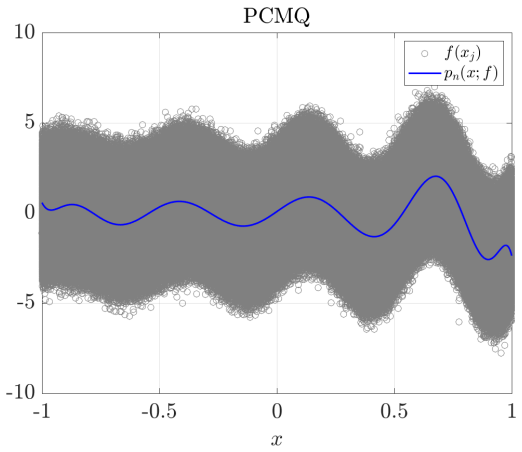

Figura 1: Aproximação por mínimos quadrados para $N = 10^8$ dados de $f(x_j) = e^{x_j} \operatorname{sen} 12\, x_j + \delta_{10^8}(j)$.

Vale mencionar que nenhum algoritmo para a solução do problema clássico dos mínimos quadrados, conhecido na literatura antes do nosso artigo [4], funciona com esse exemplo em nossa máquina[1]. A vantagem do nosso novo algoritmo o faz altamente utilizável, não somente nas diversas áreas da ciência e tecnologia, em que já é tradicionalmente empregado, mas também como ingrediente essencial em problemas com características de "big data" no quesito de volume de dados. Nesse sentido, sobre o último assunto, recomendamos com entusiasmo as palestras e os textos do Professor Emmanuel Candès, professor de matemática e estatística da Universidade de Stanford e um dos mais célebres especialistas na área.

O Capítulo 1 da presente tese apresenta os problemas de aproximação em espaços vetoriais com produtos internos gerais. Nele também é introduzido o PCMQ. O Capítulo 2 trata dos polinômios de Gram e suas propriedades. A quadratura que é utilizada consta do Capítulo 3, com todos os detalhes numéricos para um cálculo em excelente precisão. São mostrados os métodos mais populares para resolver o PCMQ no Capítulo 4. Nos Capítulos 5 e 6, são introduzidos nosso algoritmo e realizadas diversas comparações para exibir sua eficiência. As conclusões estão no Capítulo 7.

[1]Todos os testes numéricos foram realizados em MATLAB R2017b (Student License) com um MacBook Pro 2012 2.3 GHz Intel Core i7.

1 Aproximações em espaços de Hilbert

O conceito de espaço vetorial com produto interno que é tratado neste trabalho surgiu por volta de 1912 com as contribuições de David Hilbert (1862-1943). O objetivo deste capítulo é introduzir os problemas de aproximação nestes espaços dando ênfase ao problema dos mínimos quadrados. Para principais definições, propriedades e teoremas, recomenda-se a obra de Deutsch [5].

1.1 Preliminares

Seja $\mathbb{H}$ um espaço de Hilbert com produto interno

$$\langle \cdot , \cdot \rangle \ : \ \mathbb{H} \times \mathbb{H} \longrightarrow \mathbb{R}.$$

Se u é um elemento de $\mathbb{H}$ então a norma induzida é

$$\|u\| := \sqrt{\langle u, u \rangle}.$$

Tal espaço é estritamente normado, pois a igualdade $\|u+v\| = \|u\| + \|v\|$ é satisfeita se, e somente se, u e v são linearmente dependentes. Com a definição da norma, uma métrica (ou distância) entre dois elementos u e v de $\mathbb{H}$ fica definida por

$$d(u, v) := \|u - v\|.$$

Agora é discutida a melhor aproximação em um espaço de Hilbert. Para tal, sejam $\{\varphi_i\}_{i=0}^n$ um subconjunto de vetores linearmente independentes de $\mathbb{H}$ e $\Phi_n = \text{span}(\{\varphi_i\}_{i=0}^n)$ o subespaço gerado por todas combinações lineares dos elementos φ_i, $i = 0, \ldots, n$, ou seja, $\Phi_n := \sum_{i=0}^{n} a_i \varphi_i$ com $a_i \in \mathbb{R}$, $i = 0, 1 \ldots, n$.

Definição 1.1. Um elemento p de Φ_n é chamado de *melhor aproximação para* $u \in \mathbb{H}$ se

$$d(u, p) = \inf_{\varphi \in \Phi_n} \|u - \varphi\| := \varepsilon(u, \Phi_n).$$

Além disto, o número não negativo $\varepsilon(u, \Phi_n)$ é o erro em tal aproximação e também define a *distância entre* u *e* Φ_n.

A existência do elemento de melhor aproximação está garantida, pois $\mathbb{H}$ é estritamente normado, o que implica que Φ_n é um subconjunto convexo e fechado [5, Cap. 3]. Além disto, tal elemento é único.

Se os elementos distintos $u, v \in \mathbb{H}$ são tais que $\langle u, v \rangle = 0$, então dizemos que u é *ortogonal* a v.

Teorema 1.2. *Em um espaço de Hilbert $\mathbb{H}$, o elemento $p \in \Phi_n$ é o elemento de melhor aproximação para $u \in \mathbb{H}$ se, e somente se, $\langle u - p, \varphi \rangle = 0$ para todo $\varphi \in \Phi_n$.*

Demonstração. Veja [5, p. 50]. □

Agora obtemos uma expressão para o elemento de melhor aproximação por meio da condição de ortogonalidade caracterizada no teorema anterior. Como $p \in \Phi_n = \text{span}(\{\varphi_i\}_{i=0}^n)$, então

$$p = a_0\varphi_0 + a_1\varphi_1 + \ldots + a_n\varphi_n. \tag{1.1}$$

As condições necessárias e suficientes, $\langle u - p, \varphi \rangle = 0$ para todo $\varphi \in \Phi_n$, aplicadas aos vetores φ_k, $k = 0, \ldots, n$, isto é, $\langle u - p, \varphi_k \rangle = 0$, $k = 0, \ldots, n$, implicam nas chamadas *equações normais*

$$\sum_{i=0}^{n} a_i \langle \varphi_i, \varphi_j \rangle = \langle u, \varphi_i \rangle, \quad j = 0, 1, \ldots, n$$

que são representadas matricialmente na forma

$$\begin{bmatrix} \langle \varphi_0, \varphi_0 \rangle & \langle \varphi_1, \varphi_0 \rangle & \ldots & \langle \varphi_n, \varphi_0 \rangle \\ \langle \varphi_0, \varphi_1 \rangle & \langle \varphi_1, \varphi_1 \rangle & \ldots & \langle \varphi_n, \varphi_1 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \varphi_0, \varphi_n \rangle & \langle \varphi_1, \varphi_n \rangle & \ldots & \langle \varphi_n, \varphi_n \rangle \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \langle u, \varphi_0 \rangle \\ \langle u, \varphi_1 \rangle \\ \vdots \\ \langle u, \varphi_n \rangle \end{bmatrix}, \tag{1.2}$$

cuja matriz é conhecida por *matriz de Gram* denotada por $\mathbf{G}(\varphi_0, \varphi_1, \ldots, \varphi_n)$. Então a existência e a unicidade do polinômio de melhor aproximação (1.1) é óbvia pelo fato de que o determinante de uma matriz de Gram é sempre positivo para vetores linearmente independentes.

Evidentemente os valores a_i, $i = 0, 1 \ldots, n$ são prontamente obtidos se o conjunto $\{\varphi_i\}_{i=0}^n$ é *ortogonal*, isto é, se $\langle \varphi_i, \varphi_j \rangle = 0$ para $i \neq j$. Neste caso a solução é dada pelos chamados *coeficientes de Fourier*

$$a_k = \frac{\langle u, \varphi_k \rangle}{\langle \varphi_k, \varphi_k \rangle}, \quad k = 0, \ldots, n. \tag{1.3}$$

Portanto, os coeficientes acima compõem uma fórmula para a melhor aproximação p em (1.1). Além disto, do ponto de vista numérico, a ortogonalidade do conjunto $\{\varphi_i\}_{i=0}^n$ guia a resultados mais estáveis.

Teorema 1.3. *O determinante da matriz de Gram satisfaz às seguintes desigualdades*

$$0 \;\leq\; \det \mathbf{G}(y_0, y_1, \ldots, y_n) \;\leq\; \|y_0\|^2 \|y_1\|^2 \ldots \|y_n\|^2.$$

A igualdade inferior é atingida se, e somente se, o conjunto $\{y_i\}_{i=0}^n$ é linearmente dependente. A igualdade superior é atingida se, e somente se, tal conjunto é ortogonal.

Demonstração. Ver [5, p. 65 exerc. 7]. □

As desigualdades do teorema anterior alertam para o malcondionamento que as equações normais estão sujeitas caso o determinante seja quase singular. Golub e van Loan [6, p. 239] mostram como pequenas perturbações na matriz de Gram $\mathbf{G}(\varphi_0, \varphi_1, \ldots, \varphi_n)$ ou no vetor $[\langle u, \varphi_0 \rangle, \langle u, \varphi_1 \rangle \ldots, \langle u, \varphi_n \rangle]^{\mathrm{T}}$ podem levar a grandes erros sobre o vetor $[a_0, a_1, \ldots, a_n]^{\mathrm{T}}$ caso o determinante seja próximo de zero. Portanto, é desejável que o determinante seja tão grande quanto possível, isto é, que o conjunto $\{\varphi_i\}_{i=0}^n$ seja ortogonal.

Estas últimas observações tornam os coeficientes de Fourier ainda mais atraentes, pois além de fornecer diretamente a solução para a melhor aproximação p, eles são menos sensíveis às pequenas perturbações nos dados que compõem o sistema das equações normais.

1.2 Identidade de Parseval

Note que se o conjunto $\{\varphi_k\}_{k=0}^n$ é *ortonormal*, isto é, se $\langle \varphi_i, \varphi_j \rangle = \delta_{ij}$, onde δ_{ij} é o delta de Kronecker, temos

$$\begin{aligned} \varepsilon^2(u, \Phi_n) &= \langle u - p, u - p \rangle = \langle u, u \rangle - \langle p, u \rangle, \\ &= \langle u, u \rangle - \sum_{k=0}^{n} a_k \langle \varphi_k, u \rangle, \\ &= \|u\|^2 - \sum_{k=0}^{n} a_k^2. \end{aligned}$$

Como $\varepsilon^2(u, \Phi_n) \geq 0$, então

$$\sum_{k=0}^{n} a_k^2 \leq \|u\|^2. \tag{1.4}$$

A importante relação anterior é conhecida por *desigualdade de Bessel*. Se a dimensão do espaço de Hilbert $\mathbb{H}$ for $n+1$, ou seja, se $\{\varphi_k\}_{k=0}^n$ forma uma base ortonormal para $\mathbb{H}$, temos obviamente $u \in \Phi_n = \mathbb{H}$. Logo, o erro $\varepsilon(u, \Phi_n)$ é nulo e a igualdade em (1.4) é atingida. Sob tais condições obtemos a *Identidade de Parseval*,

$$\sum_{k=0}^{n} a_k^2 = \|u\|^2. \tag{1.5}$$

O caso mais geral para a validade da identidade acima ocorre quando Φ_n é denso em $\mathbb{H}$ com $n \leq \infty$. Ver [5, p. 59].

1.3 Problema clássico dos mínimos quadrados

Historicamente, ainda no século XIX, Adrien-Marie Legendre (1752-1833) e Carl F. Gauss (1777-1855) foram os primeiros a resolver problemas concretos sobre aproximação de dados por meio do PCMQ. Em 1805, Legendre publicou o procedimento algébrico do método em *Nouvelles méthodes pour la détermination des orbites des comètes*. Porém, em 1809, Gauss afirmou em *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections* que já conhecia o princípio do método e o usava desde 1795 para analisar dados estatísticos e realizar cálculos astronômicos. Devido ao sucesso da previsão de Gauss para a localização precisa do asteróide Ceres em 1801, o método dos mínimos quadrados tornou-se ferramenta básica para análise de dados astronômicos e geodésicos. Hoje em dia o método está disponível na maioria dos pacotes computacionais que desenham linhas e curvas suaves ou que servem para ajustar dados experimentais perturbados.

Sejam N um número natural, x_j, $j = 1, \ldots, N$, pontos distintos e f uma função definida nestes pontos. O problema clássico dos mínimos quadrados consiste em obter um polinômio algébrico p de grau, no máximo, n, em geral, muito menor do que N, que minimiza a soma dos quadrados dos desvios $f(x_j) - p(x_j)$. Mais precisamente, se

$$\mathbb{P}_n := \left\{ p(x) = \sum_{k=0}^{n} \alpha_k x^k \ : \ \alpha_k \in \mathbb{R}, \ k = 0, \ldots, n \right\}$$

é o espaço de todos os polinômios algébricos reais cujo grau não excede n, então busca-se determinar o polinômio $p \in \mathbb{P}_n$ tal que a soma residual de quadrados,

$$D(f;p) := \sum_{j=1}^{N} \left(f(x_j) - p(x_j)\right)^2 \tag{1.6}$$

seja mínima. Tal soma é o parâmetro que mede a discrepância entre os dados $f(x_j)$, $j = 1, \ldots, n$, e o ajuste realizado na aproximação.

Este problema possui uma interpretação natural em termos de aproximação em um espaço de Hilbert. De fato, sejam os N pontos $x_1, \ldots, x_N$ e o espaço euclidiano de vetores de dimensão N os quais interpretaremos como valores de funções f definidas nos pontos $x_1, \ldots, x_N$, isto é, vetores da forma

$$\mathbf{f} = (f(x_1), \ldots, f(x_N)) \in \mathbb{R}^N.$$

Com tal notação, tomamos o produto interno usual entre dois vetores $\mathbf{f}_1$ e $\mathbf{f}_2$ do espaço $\mathbb{R}^N$:

$$\langle \mathbf{f}_1, \mathbf{f}_2 \rangle_2 = \sum_{j=1}^{N} f_1(x_j) f_2(x_j). \tag{1.7}$$

Logo a norma fica dada por

$$\|\mathbf{f}\|_2 = \left(\sum_{j=1}^{N} [f(x_j)]^2 \right)^{1/2}$$

que, por sua vez, produz a distância

$$\mathbf{d}(\mathbf{f}_1, \mathbf{f}_2) = \left[\sum_{j=1}^{N} (f_1(x_j) - f_2(x_j))^2 \right]^{1/2}.$$

Comparando a distância anterior com (1.6), vemos que para $\mathbf{f} = (f(x_1), \ldots, f(x_N))$ e $\mathbf{p} = (p(x_1), \ldots, p(x_N))$, onde $p \in \mathbb{P}_n$,

$$D(f;p) \;=\; \mathbf{d}^2(\mathbf{f}, \mathbf{p}).$$

Seja $p_n(\cdot\,; f) \in \mathbb{P}_n$ o polinômio que minimiza $D(f;\cdot)$. Segundo a notação da Definição 1.1 de melhor aproximação,

$$\sqrt{D(f;p_n)} \;= \min_{p \in \mathbb{P}_n} \sqrt{D(f;p)} \;=\; \inf_{p \in \mathbb{P}_n} \|\mathbf{f} - \mathbf{p}\|_2 \;=\; \varepsilon(f, \mathbb{P}_n).$$

Logo o problema dos mínimos quadrados pode ser interpretado como um problema de aproximação de uma função $f \in \mathbb{H}$ por polinômios do subespaço $\mathbb{P}_n \subset \mathbb{H}$, onde $\mathbb{H}$ é o espaço das funções definidas em $x_1, \ldots, x_N$. Daí, a solução do problema é dada por

$$p_n(\cdot\,; f) = a_0\varphi_0 + a_1\varphi_1 + \ldots + a_n\varphi_n,$$

onde $\{\varphi_k\}_{k=0}^n$ é uma base para $\mathbb{P}_n$ e os coeficientes $a_0, \ldots, a_n$ são a solução do sistema (1.2).

Além disto, $a_0, \ldots, a_n$ são prontamente obtidos por meio dos coeficientes de Fourier (1.3) sempre que $\{\varphi_k\}_{k=0}^n$ forma um conjunto ortonormal de polinômios.

Mais geralmente, a soma residual de quadrados ponderada

$$\sum_{j=1}^{N} w_j \left(f(x_j) - p(x_j)\right)^2, \quad w_j > 0, \tag{1.8}$$

define um problema de mínimos quadrados ponderados. Os valores w_j são chamados de "pesos". A escolha dos pesos w_j caracteriza uma sequência de polinômios ortogonais, como será mostrado adiante.

2 Polinômios Ortogonais

Como mostrado no capítulo anterior, as funções mais adequadas para o conjunto $\{\varphi_i\}_{i=0}^n$ são as ortogonais. Uma vez que os problemas de aproximação deste trabalho buscam aproximar funções de um espaço de Hilbert por meio de polinômios, então trabalharemos com polinômios ortogonais. As obras clássicas de Gabor Szegő [7] e Theodore S. Chihara [8] são nossas principais referências.

O objetivo deste capítulo é apresentar os polinômios de Gram, uma classe de polinômios ortogonais de medida discreta. Inicialmente apresentamos algumas definições e teoremas gerais de polinômios ortogonais. Em seguida são caracterizados os polinômios de medida discreta e são apresentados os polinômios de Gram. O estudo é sempre realizado em espaços vetoriais polinomiais.

2.1 Preliminares

Seja $\mathrm{d}\lambda$ uma medida de Borel para a qual $\lambda(x)$ é uma função não decrescente sobre com limites finitos quando $x \to \pm\infty$. Sejam os momentos

$$\mu_r = \mu_r(\mathrm{d}\lambda) := \int_{\mathbb{R}} x^r \,\mathrm{d}\lambda(x), \qquad r = 0, 1, \ldots, \quad \text{com} \quad \mu_0 > 0, \tag{2.1}$$

finitos onde a integral é tomada no senso de Riemann-Stieltjes. Em tais condições, λ é chamada de *função distribuição*. Seja $\mathbb{P}$ o espaço vetorial de todos polinômios algébricos com coeficientes reais. Para quaisquer u e $v \in \mathbb{P}$, definimos o produto interno por

$$\langle u, v\rangle := \int_{\mathbb{R}} u(x)v(x) \,\mathrm{d}\lambda(x). \tag{2.2}$$

Como os momentos μ_r existem, então o produto interno acima fica bem definido.

Definimos por *pontos de aumento* da distribuição λ, os pontos x_0 nos quais $\lambda(x_0+\varepsilon) > \lambda(x_0-\varepsilon)$ para todo $\varepsilon > 0$. A positividade do produto interno depende da função λ. De fato, a relação

$$\|u\|^2 = \int_{\mathbb{R}} [u(x)]^2 \,\mathrm{d}\lambda(x) = 0$$

so é possível se, e somente se, u é nulo em todos os pontos de aumento de λ. Logo o produto interno é definido positivo em $\mathbb{P}$ sempre que λ possui infinitos pontos de aumento. Entretanto, caso λ tenha apenas um número finito de pontos de aumento, digamos $x_1, x_2, \ldots, x_N$, então a positividade do produto interno é válida apenas em $\mathbb{P}_k$ para $k \leq N-1$. De fato, se $k \geq N$, poderíamos tomar $u(x) = (x-x_1)(x-x_2)\ldots(x-x_N)$ não nulo com $\|u\| = 0$ em $\mathbb{P}_k$, o que seria contraditório.

O conjunto dos pontos de aumento da distribuição λ é chamado de *suporte* da medida $\mathrm{d}\lambda$ e sua envoltória convexa é o *intervalo suporte* de $\mathrm{d}\lambda$. Estas últimas definições e as observações acima são expressas no seguinte lema.

Lema 2.1 (Szegő [7, §2.2]). *O espaço vetorial $\mathbb{P}_{N-1}$ (ou $\mathbb{P}$) será um espaço de Hilbert com o produto interno (2.2) se o suporte da medida $\mathrm{d}\lambda$ possuir N (ou infinitos) pontos de aumento.*

Evidentemente, todas propriedades e definições apresentadas no capítulo anterior permanecem válidas para os espaços vetoriais do lema acima. Um polinômio é dito *mônico* se seu coeficiente líder for igual a 1.

Definição 2.2. Os polinômios mônicos ϕ_k de grau k são chamados de *polinômios ortogonais mônicos com relação à medida* $\mathrm{d}\lambda$ e são representados por $\phi_k(x) = \phi_k(x; \mathrm{d}\lambda)$, se $\langle \phi_k, \phi_l \rangle = 0$ para $k \neq l$ e $\|\phi_k\| > 0$. A normalização $\tilde{\phi}_k = \phi_k / \|\phi_k\|$ guia aos *polinômios ortonormais* que satisfazem $\langle \tilde{\phi}_k, \tilde{\phi}_l \rangle = \delta_{kl}$.

Como os polinômios da definição acima possuem graus distintos, então o conjunto $\{\phi_0, \phi_1, \ldots, \phi_k\}$ forma uma base para o espaço vetorial $\mathbb{P}_k$ com $k \leq \infty$, sendo $\mathbb{P}_\infty := \mathbb{P}$. A quantidade de polinômios ortogonais para uma dada medida é identificada pelos dois teoremas a seguir.

Teorema 2.3 (Szegő [7, §2.2]). *Se o produto interno (2.2) é definido positivo em $\mathbb{P}$, então existe uma única sequência infinita de polinômios ortogonais mônicos $\{\phi_k\}_{k=0}^{\infty}$.*

Demonstração. Os polinômios ϕ_k podem ser gerados pelo processo de ortogonalização de Gram-Schmidt com a sequência de potências $e_k(x) = x^k$, $k = 0, 1, \ldots$. Assim, basta tomar $\phi_0 = 1$ e, para $k = 1, 2, \ldots$, podemos calcular os demais polinômios recursivamente:

$$\phi_k = e_k - \sum_{l=0}^{k-1} c_l \phi_l, \qquad c_l = \frac{\langle e_k, \phi_l \rangle}{\langle \phi_l, \phi_l \rangle}. \tag{2.3}$$

Por hipótese, $\langle \phi_l, \phi_l \rangle > 0$ para $l = 1, 2 \ldots$, então cada polinômio ϕ_k está bem definido e é unicamente determinado. Além disso, por construção, ϕ_k é ortogonal a todos polinômios ϕ_l para $l < k$. □

Teorema 2.4 (Szegő [7, §2.2]). *Se o produto interno (2.2) é definido positivo em $\mathbb{P}_{N-1}$ mas não em $\mathbb{P}_d$ para $d \geq N$, então existem apenas N polinômios ortogonais $\{\phi_k\}_{k=0}^{N-1}$ unicamente determinados.*

Demonstração. O procedimento de Gram-Schmidt (2.3) pode ser aplicado enquanto os denominadores $\langle \phi_l, \phi_l \rangle$ de c_l são positivos, isto é, para $k \leq N$. O último polinômio encontrado ϕ_N é ortogonal a todos ϕ_j, $j \leq N-1$, sendo todos estes últimos ortogonais entre si e de norma positiva. Porém, ϕ_N possui norma zero. De fato, por hipótese existe um polinômio mônico $\varphi \in \mathbb{P}_N$ tal que $\|\varphi\| = 0$. Como $\varphi - \phi_N$ possui grau $N-1$ e $\{\phi_k\}_{k=0}^{N-1}$ é uma base para $\mathbb{P}_{N-1}$, então

$$\varphi = \phi_N + \sum_{j=0}^{N-1} a_j \phi_j,$$

para determinados coeficientes a_j. Logo,

$$0 = \|\varphi\|^2 = \|\phi_N\|^2 + \sum_{j=0}^{N-1} a_j^2 \|\phi_j\|^2.$$

Daí, $\|\phi_N\| = 0$. Portanto, ϕ_N não pode pertencer à sequência dos polinômios ortogonais. □

O importante teorema a seguir trata da localização dos zeros dos polinômios ortogonais.

Teorema 2.5 (Szegő [7, p. 44])**.** *Todos os zeros de* $\phi_k(\,\cdot\,) = \phi_k(\,\cdot\,;\mathrm{d}\lambda)$*,* $k \geq 1$*, são reais, simples e contidos no interior do intervalo suporte* $[a, b]$ *de* $\mathrm{d}\lambda$*.*

Demonstração. Uma vez que $\int_{\mathbb{R}} \phi_k(x)\,\mathrm{d}\lambda(x) = 0$ para $k \geq 1$, então existe ao menos um ponto no interior do intervalo suporte $[a, b]$ no qual ϕ_k muda de sinal. Sejam $z_1, z_2, \ldots, z_k$, $k \leq n$, todos tais pontos. Suponhamos que valha a desigualdade estrita $k < n$. Logo, pela ortogonalidade, teremos

$$\int_{\mathbb{R}} \phi_n(x) \prod_{j=1}^{k} (x - z_j)\,\mathrm{d}\lambda(x) = 0.$$

Por outro lado, a igualdade acima é impossível porque o integrando possui sinal constante. Portanto, $k = n$. □

O algoritmo de Clenshaw [9] fornece um eficiente método para calcular combinações lineares de funções que satisfazem uma relação de recorrência de três termos. É bem conhecido [7, Teo. 3.2.1] que os polinômios ortonormais satisfazem a uma relação de três termos da forma:

$$\sqrt{\beta_{n+1}}\,\tilde{\phi}_{n+1} \;=\; (x - \alpha_n)\,\tilde{\phi}_n \;-\; \sqrt{\beta_n}\,\tilde{\phi}_{n-1}, \quad n = 0, 1, \ldots, \tag{2.4}$$

para $\tilde{\phi}_{-1} := 0$ com $\beta_n > 0$ para todo n. O resultado a seguir apresenta o algoritmo de Clenshaw para o caso dos polinômios ortonormais.

Teorema 2.6 (Algoritmo de Clenshaw [9])**.** *Os seguintes passos retornam a seguinte combinação linear dos polinômios ortonormais*

$$r(x) = d_0\tilde{\phi}_0(x) + d_1\tilde{\phi}_1(x) + \ldots + d_n\tilde{\phi}_n(x), \quad d_i \in \mathbb{R}, \quad i = 0, 1 \ldots, n.$$

Inicialização: $\sqrt{\beta_n}u_n = d_n, \quad u_{n+1} = 0.$

Continuação: *Se* $n > 0$*, para* $k = n-1, n-2, \ldots, 0$*, faça:*

$$\sqrt{\beta_k}u_k \;=\; (x - \alpha_k)u_{k+1} - \sqrt{\beta_{k+1}}u_{k+2} + d_k.$$

Resultado: $r(x) \;=\; \sqrt{\beta_0}u_0\tilde{\phi}_0.$

Demonstração. Sejam a matriz quadrada de ordem $n + 1$,

$$C = \begin{bmatrix} \sqrt{\beta_0} & & & & \\ \alpha_0 - x & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & \alpha_1 - x & \sqrt{\beta_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\beta_{n-1}} & \alpha_{n-1} - x & \sqrt{\beta_n} \end{bmatrix},$$

e os vetores $\tilde{\phi} = [\tilde{\phi}_0(x), \tilde{\phi}_1(x), \ldots, \tilde{\phi}_n(x)]^{\mathrm{T}}$, $d = [d_0, d_1, \ldots, d_n]^{\mathrm{T}}$ e $u = [u_0, u_1, \ldots, u_n]^{\mathrm{T}}$. A relação de recorrência (2.4) pode ser escrita matricialmente na forma $C\tilde{\phi} = \rho$, onde $\rho = [\sqrt{\beta_0}\tilde{\phi}_0,\, 0, \ldots,\, 0]^{\mathrm{T}}$. Logo, os passos da "Inicialização" e da "Continuação" podem ser escritos na forma $C^{\mathrm{T}}u = d$. A matriz C é invertível, pois $\det(C) = \sqrt{\beta_0\beta_1 \ldots \beta_n} \neq 0$. Com isto, $r = d^{\mathrm{T}}\phi = d^{\mathrm{T}}C^{-1}\rho = ((C^{\mathrm{T}})^{-1}d)^{\mathrm{T}}\rho = u^{\mathrm{T}}\rho$. □

2.2 Polinômios ortogonais de medida discreta

Até aqui podemos perceber que a escolha da medida $\mathrm{d}\lambda$, ou melhor, da função distribuição λ, define a sequência dos polinômios ortogonais. Quando λ é contínua temos que $\mathrm{d}\lambda = w(x)\,\mathrm{d}x$, onde w é a *função peso* que é não negativa e integrável em $\mathbb{R}$. Os polinômios de Jacobi, Hermite e Laguerre são os exemplos mais conhecidos deste caso devido às suas diversas aplicações. Ver [7, 8]. Contudo, uma vez que a continuidade da distribuição λ não é exigida, a mesma poderá admitir saltos de descontinuidade em alguns pontos. Mais especificamente, chamamos de *medida discreta* aquela cujo suporte consiste de um conjunto enumerável de pontos x_j nos quais λ possui salto positivo w_j. O produto interno nestes casos torna-se um interessante somatório:

$$\int_{\mathbb{R}} u(x)v(x)\,\mathrm{d}\lambda(x) = \sum_{j=1}^{N} w_j u(x_j)v(x_j), \qquad N \leq \infty.$$

Nestes casos os polinômios obtidos são chamados de *polinômios ortogonais de medida discreta* ou, simplesmente, de *polinômios discretos*, como serão denominados neste trabalho. Os tipos mais gerais encontrados na literatura podem ser classificados em polinômios de Hahn, Krawtchouk, Charlier e Meixner. Para mais detalhes, ver [10,11]. Os dois primeiros têm $N < \infty$; e os dois últimos, $N = \infty$.

Especificamente, os polinômios de Hahn $Q_n(\,\cdot\,;\alpha,\beta,M)$ de grau $n \leq M < \infty$ e parâmetros $\alpha, \beta > -1$ ou $\alpha, \beta < -M$ são expressos pela seguinte fórmula hipergeométrica

$$Q_n(s;\alpha,\beta,M) = {}_3F_2\left(\begin{matrix} -n, n+\alpha+\beta+1, -s \\ \alpha+1, -M \end{matrix}\,\middle|\, 1\right), \quad n = 0, 1, \ldots, M, \tag{2.5}$$

de acordo com a notação de [12, Cap. 2]. Segundo Szegő [7, p. 33], Chebyshev [13] investigou o caso notável dos polinômios cuja distribuição λ possui saltos $w_j = 1$ nos pontos $s_j = j - 1$, $j = 1, 2, \ldots, N$. Eles são conhecidos por polinômios discretos de Chebyshev $t_n(\cdot\,; N)$ com a seguinte fórmula do tipo Rodrigues

$$t_n(s;N) = n!\,\Delta^n\left\{\binom{s}{n}\binom{s-N}{n}\right\}, \quad n = 0, 1, \ldots, N-1, \tag{2.6}$$

onde $\Delta f(s) = f(s+1) - f(s)$ é o operador de diferença finita ascendente. Nós verificamos que, exceto por uma constante, eles são um caso particular dos polinômios de Hahn por meio da seguinte relação

$$t_n(s;N) = (-1)^n (N-n)_n\, Q_n(s;0,0,N-1),$$

na qual o símbolo de Pochhammer $(x)_n$ denota o fatorial deslocado $(x)_n := x(x+1)\ldots(x+n-1)$, $n \geq 1$, e $(x)_0 := 1$. Chebyshev mostrou em [13, pp. 547, 552] que o produto interno neste caso é

$$\langle t_m(s;N), t_n(s;N)\rangle = \sum_{j=1}^{N} t_m(s_j;N)t_n(s_j;N) = \frac{N(N-n)_n(N+1)_n}{2n+1}\,\delta_{mn}, \tag{2.7}$$

$n, m = 0, 1, \ldots, N-1$. Chebyshev [14, 15] também considerou o caso mais geral no qual os pontos s_j são substituídos por um conjunto de N pontos distintos, como mostraremos na próxima seção.

2.3 Polinômios de Gram

Transformações afins podem mapear os tais pontos $s_j = j - 1$, $j = 1, 2, \ldots, N$, para outros pontos igualmente espaçados num intervalo real. Exceto por fatores constantes, tais polinômios correspondentes são comumente chamados de polinômios de Gram conforme a terminologia de [16–22]. Eles recebem este nome devido ao importante trabalho *On series expansions determined by the methods of least squares* do matemático dinamarquês Jørgen Pedersen Gram. Seu trabalho conferiu-lhe o grau de Doutor em Ciência[1] em 1879 e foi publicado mais tarde em [23]. Particularmente, a transformação afim de variáveis dada por $x = -1 + (2s + 1)/N$ leva os pontos s_j para os pontos

$$x_j = x_j(N) := -1 + \frac{2j - 1}{N}, \quad j = 1, 2, \ldots, N, \tag{2.8}$$

igualmente espaçados no intervalo $[-1 + 1/N,\ 1 - 1/N]$.

Denominamos por *polinômios de Gram* $G_n(\cdot\,; N)$, de grau $n \leq N - 1$, aqueles cuja distribuição λ tem salto $w_j = 1/N$ nos pontos x_j, $j = 1, 2, \ldots, N$. Além disso, por convenção, vamos considerá-los ortonormais. Logo estamos tratando do seguinte produto interno

$$\langle G_m(x; N), G_n(x; N) \rangle = \sum_{j=1}^{N} \frac{1}{N}\, G_m(x_j; N) G_n(x_j; N) = \delta_{mn}, \tag{2.9}$$

$m, n = 0, 1, \ldots, N - 1$. Evidentemente,

$$G_n(x; N) = \frac{\sqrt{2n + 1}}{\sqrt{(N - n)_n (N + 1)_n}}\, t_n(s(x); N)$$

e também

$$G_n(x; N) = (-1)^n \frac{\sqrt{(2n + 1)(N - n)_n}}{\sqrt{(N + 1)_n}}\, Q_n(s(x); 0, 0, N - 1). \tag{2.10}$$

Os polinômios ortonormais de Gram obedecem à seguinte relação de recorrência de três termos,

$$G_n(x; N) = 2\alpha_{n-1}\, x\, G_{n-1}(x; N) - \frac{\alpha_{n-1}}{\alpha_{n-2}}\, G_{n-2}(x; N), \quad n = 2, 3, \ldots, N - 1, \tag{2.11}$$

onde $G_0(x; N) = 1$ e $G_1(x; N) = 2\alpha_0\, x$ com

$$\alpha_{n-1} = \frac{N}{n} \left(\frac{n^2 - \frac{1}{4}}{N^2 - n^2} \right)^{\frac{1}{2}}, \quad n = 1, 2, \ldots, N - 2. \tag{2.12}$$

Tal resultado acima é facilmente obtido por meio da relação de recorrência dos polinômios de Hahn, ver [10, Cap. 9]. A Figura 2.1 apresenta os gráficos dos polinômios $G_n(\cdot\,; 10^2)$ no intervalo $[-1, 1]$ para $n = 1, 2, \ldots, 5$.

Pelos gráficos na Figura 2.1, vemos que os polinômios atingem seu máximo, em módulo, para $x = \pm 1$. De fato, Barnard *et al.* [19] demonstraram que a norma $\|G_n(\cdot\,; N)\|_\infty$ é atingida em tais pontos, onde $\|f\|_\infty := \sup\{\, |f(x)| \,:\, x \in [-1, 1]\}$. Pesquisando o comportamento de $\|G_n(\cdot\,; N)\|_\infty$ com relação ao crescimento de x, n e N eles verificaram

[1] Este diploma, equivalente ao D.Sc. britânico de hoje, está em um nível superior ao de um doutorado. Fonte: `https://mathshistory.st-andrews.ac.uk/Biographies/Gram/`.

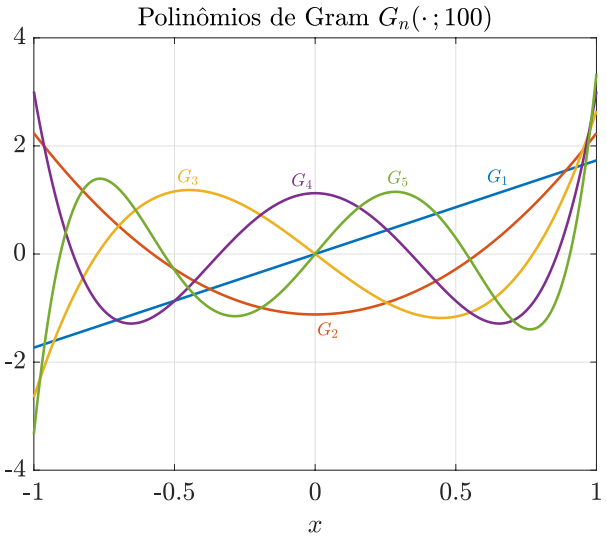

Figura 2.1: Polinômios de Gram com $N = 100$ de graus 1 até 5.

que os polinômios de Gram oscilam com grande amplitude nas proximidades de tais pontos quando n está muito próximo de N. Zaremba [24] observou a importância da razão $n/\sqrt{N}$ e Barnard *et al.* [19] notaram que para valores grandes de N e n, a norma $\|G_n(\cdot\,;N)\|_\infty$ varia pouco sempre que $n/\sqrt{N} \leq 5/2$. Estes resultados contribuirão diretamente para o nosso algoritmo, como mostraremos ao longo deste trabalho.

Sabemos que existem equações de diferenças finitas relacionando os polinômios discretos [10, Cap. 9] que poderiam aproximar as derivadas destes polinômios. Infelizmente, nossos testes numéricos comprovaram que as mesmas não são suficientemente precisas para os fins deste trabalho. Todavia ao derivar a relação (2.11), encontramos a seguinte fórmula que atende aos nossos propósitos,

$$G'_n(x;N) = 2\alpha_{n-1}\left[G_{n-1}(x;N) + xG'_{n-1}(x;N)\right] - \frac{\alpha_{n-1}}{\alpha_{n-2}}G'_{n-2}(x;N), \tag{2.13}$$

para $n \geq 2$, com $G'_1(x;N) = 2\alpha_0$ e $G'_0(x;N) = 0$. De fato, os nossos testes numéricos comprovam que as fórmulas (2.11) e (2.13) que avaliam o polinômio e a derivada são precisas o suficiente para cálculos em precisão dupla[2]. Confira um exemplo na Figura 2.2 para $n = 60$. O valor exato foi considerado aquele obtido pelo mesmo algoritmo com precisão quádrupla[3]. Exibimos apenas sobre o intervalo $[0, 1]$ devido à simetria dos polinômios.

Ressaltamos que em [3, Algoritmo 1] foi apresentado um algoritmo baseado no método de Horner para avaliar o polinômio e a derivada de Gram que se mostra muito eficiente para softwares que suportam cálculos simbólicos. Infelizmente, para softwares com cálculos numéricos em precisão dupla, este algoritmo apresenta um grande erro de arredondamento à medida que x se aproxima dos extremos ± 1. Por este motivo, preferimos o uso das fórmulas mencionadas aqui.

Para checarmos também a ortonormalidade dos polinômios via fórmula (2.11), seja a

[2]Equivalente à 16 dígitos, aproximadamente.
[3]Equivalente à 34 dígitos, aproximadamente.

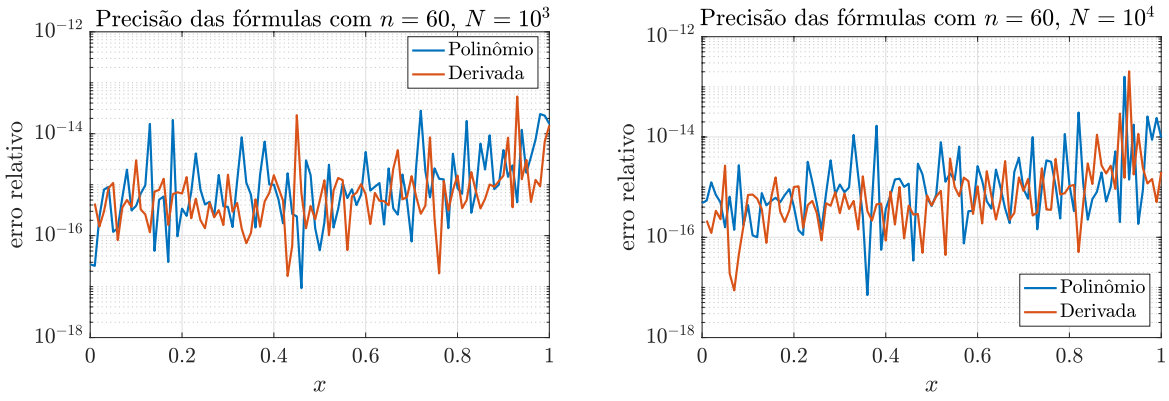

Figura 2.2: Erro relativo das fórmulas para avaliar o polinômio e a derivada.

matriz retangular

$$\mathbf{V} = \begin{bmatrix} G_0(x_1;N) & \dots & G_n(x_1;N) \\ \vdots & \ddots & \vdots \\ G_0(x_N;N) & \dots & G_n(x_N;N) \end{bmatrix},$$

de dimensão $N \times (n+1)$. Agora seja a matriz resultante $I - \frac{1}{N}\mathbf{V}^{\mathrm{T}}\mathbf{V}$, onde I é a matriz identidade de ordem $n+1$. Evidentemente um resultado muito próximo da matriz nula comprova a precisão de (2.11). De fato, a Tabela 2.1 mostra que a tal matriz resultante pode ser considerada nula em precisão dupla.

Tabela 2.1: Maior valor em módulo dos elementos da matriz $I - \frac{1}{N}\mathbf{V}^{\mathrm{T}}\mathbf{V}$ via (2.11).

	$N = 10^3$	$N = 10^4$	$N = 10^6$
$n = 10$	4.4409e-16	1.3323e-15	5.5511e-15
$n = 60$	1.3087e-15	1.9984e-15	5.7732e-15
$n = 100$	8.2808e-15	4.4409e-15	8.1046e-15

Seja a expansão do polinômio de Gram de grau n na base monomial,

$$G_n(x;N) = b_{n,0} + b_{n,1}x + \ldots + b_{n,n}x^n. \tag{2.14}$$

Apresentamos o seguinte resultado que é útil para o cálculo de todos os coeficientes $b_{l,k}$, $k = 0, 1, \ldots, l$, de todos os polinômios de Gram de grau $l \leq n$, a saber:

Teorema 2.7. *Os polinômios de Gram obedecem à seguinte relação:*

$$G_n(x;N) = \alpha_{n-1}\left[2^n \prod_{j=0}^{n-2} \alpha_j\, x^n + \sum_{j=1}^{\lfloor n/2 \rfloor} \left(2b_{n-1,n-1-2j} - \frac{1}{\alpha_{n-2}} b_{n-2,n-2j}\right) x^{n-2j}\right], \tag{2.15}$$

onde $n \geq 2$, $G_1(x;N) = 2\alpha_0 x$, $G_0(x;N) = 1$.

Demonstração. A derivada de ordem j de $G_n(\cdot\,;N)$ avaliada em $x = 0$ é dada por

$$G_n^{(j)}(0;N) = 2j\alpha_{n-1}G_{n-1}^{(j-1)}(0;N) - \frac{\alpha_{n-1}}{\alpha_{n-2}}G_{n-2}^{(j)}(0;N), \qquad j \geq 2. \tag{2.16}$$

Vamos obter os coeficientes dos polinômios recursivamente. Os coeficientes de $G_0(\cdot\,;N)$ e $G_1(\cdot\,;N)$ são $b_{0,0}=1$ e $b_{1,1}=2\alpha_0$. Os coeficientes de $G_2(\cdot\,;N)$ são obtidos ao observar (2.16) com $j=2$. Logo,

$$b_{2,2}=\frac{1}{2}G_2^{(2)}(0;N)=2\alpha_1 G_1^{(1)}(0;N)=4\alpha_1\alpha_0$$

e

$$b_{2,0}=G_2(0;N)=-\frac{\alpha_1}{\alpha_0}.$$

Novamente observamos a equação (2.16) com $j=3$ e obtemos

$$b_{3,3}=\frac{1}{6}G_3^{(3)}(0;N)=\alpha_2 G_2^{(2)}(0;N)=8\alpha_2\alpha_1\alpha_0$$

e

$$b_{3,1}=G_3^{(1)}(0;N)=2\alpha_2 G_2(0;N)-\frac{\alpha_2}{\alpha_1}G_1^{(1)}(0;N)=2\alpha_2 b_{2,0}-\frac{\alpha_2}{\alpha_1}b_{1,1}.$$

Analogamente usamos equação (2.16) com $j=4,5,\ldots,n$ para obter os coeficientes dos próximos polinômios de grau j em função dos coeficientes dos polinômios de grau $j-1$ e $j-2$. Em seguida o resultado é facilmente provado por indução. □

Na verdade, por meio da relação de recorrência, o teorema anterior apresenta um método iterativo para obter todos os coeficientes dos polinômios. Mostramos na Tabela 2.2 que a fórmula (2.15) permite calcular os coeficientes de Gram praticamente na precisão da máquina. Comparamos os resultados com os valores analíticos obtidos por meio própria relação de recorrência.

Tabela 2.2: Erro relativo máximo entre todos os coeficientes não nulos dos polinômios $G_k(\cdot\,;N)$, para $k=0,1,\ldots,50$.

$N=10^3$	$N=10^4$	$N=10^6$	$N=10^7$
5.5663e-16	1.2345e-15	9.2871e-16	6.2390e-16

Claramente o coeficiente líder de $G_n(\cdot\,;N)$ é

$$b_{n,n}=2^n\prod_{j=0}^{n-1}\alpha_j. \tag{2.17}$$

Observamos o comportamento dos coeficientes dos polinômios de Gram e propomos as seguintes conjecturas relacionadas a eles.

Conjectura 2.8. *Se $n\geq 4$ é par e $b_{n,k}$, $k=0,\ldots,n$, representa o maior coeficiente em módulo de $G_n(\cdot\,;N)$ com $|b_{n,k}|>|b_{n,k-2}|>|b_{n,k+2}|$, então a seguinte sequência é estritamente decrescente*

$$\{|b_{n,k}|,|b_{n,k-2}|,|b_{n,k+2}|,|b_{n,k-4}|,|b_{n,k+4}|,\ldots,|b_{n,2k-n}|,|b_{n,n}|,|b_{n,2k-n-2}|,|b_{2k-n-4}|,\ldots,|b_{n,2}|,|b_{n,0}|\}.$$

Conjectura 2.9. *Se $n\geq 1$ e $\bar{b}_n$ representa o maior coeficiente, em módulo, de $G_n(\cdot\,;N)$, então*

$$\left|\bar{b}_n\right|>\left|\bar{b}_{n-1}\right|.$$

Por meio de (2.5) e (2.10), obtemos

$$G_n(x;N) = (-1)^n \frac{\sqrt{(2n+1)(N-n)_n}}{\sqrt{(N+1)_n}} \sum_{k=0}^{n} \frac{(-n)_k (n+1)_k ((1-N-Nx)/2)_k}{(k!)^2 (1-N)_k}. \quad (2.18)$$

Agora vejamos os polinômios de Legendre P_n que são um caso particular dos polinômios de Jacobi. Os polinômios P_n são ortogonais com relação à medida $\mathrm{d}\lambda(x) = \mathrm{d}x$ no intervalo real $[-1, 1]$ e podem ser dados por [7, p. 62],

$$P_n(x) \;=\; {}_2F_1\left(\begin{matrix} -n, n+1 \\ 1 \end{matrix} \;\middle|\; \frac{1-x}{2}\right) \;=\; \sum_{k=0}^{n} \frac{(-n)_k (n+1)_k}{(k!)^2} \left(\frac{1-x}{2}\right)^k.$$

Temos que

$$\lim_{N \longrightarrow \infty} G_n(x;N) \;=\; (-1)^n \sqrt{2n+1} \sum_{k=0}^{n} \frac{(-n)_k (n+1)_k}{(k!)^2} \left(\frac{1+x}{2}\right)^k.$$

Mas pela relação de recorrência (2.11), podemos perceber que os polinômios de Gram são funções pares ou ímpares, isto é, $G_n(x;N) = (-1)^n G_n(-x;N)$. Daí obtemos o interessante limite

$$\lim_{N \longrightarrow \infty} G_n(x;N) = \sqrt{2n+1}\, P_n(x). \quad (2.19)$$

Portanto, os zeros de $G_n(\cdot\,;N)$ convergem para os zeros do polinômio de Legendre de grau n quando $N \longrightarrow \infty$. O Teorema 2.5 comprova que $G_n(\cdot\,;N)$ possui n zeros distintos que estão contidos no intervalo aberto $(-1+1/N,\ \ 1-1/N)$ bem como os de Legendre estão em $(-1, 1)$. Computacionalmente, em precisão dupla, os zeros de ambos polinômios tornam-se iguais a partir de $N \geq 10^{10}$. Nos próximos capítulos abordaremos os detalhes numéricos envolvidos nestes cálculos.

Exceto por um fator constante, a função distribuição de Gram converge para a distribuição de Legendre quando $N \longrightarrow \infty$. De fato, a medida de Gram pode ser dada por

$$\mathrm{d}\lambda(x) = \sum_{j=1}^{N} \frac{1}{N} \delta(x - x_j)\, \mathrm{d}x \quad (2.20)$$

com δ representando o delta de Dirac para o qual a medida $\delta(A) := \chi_A(0)$ possui a propriedade $\int_{\mathbb{R}} \delta(x)\, \mathrm{d}x = 1$, sendo χ_A a função característica de um conjunto mensurável A, ver [25, p. 100] e [26, p. 72]. Deste modo a distribuição λ correspondente aos polinômios de Gram é dada por

$$\lambda(x) = \begin{cases} k \text{ constante real, se } x < x_1, \\ \frac{1}{N} \left\lfloor \frac{Nx+N+1}{2} \right\rfloor + k, \text{ se } x_1 \leq x \leq x_N, \\ 1 + k, \text{ se } x_N \leq x. \end{cases} \quad (2.21)$$

Veja a Figura 2.3 onde usamos $k = 0$. Notemos que a distribuição de Legendre pode ser expressa por $\lambda(x) = x + 1$ no intervalo $[-1, 1]$. Então, exceto por um fator constante igual a $1/2$, a distribuição de Gram converge para aquela de Legendre em $[-1, 1]$. O teorema a seguir sintetiza estas ideias na topologia fraca-estrela.

Teorema 2.10. *Se* $\mathrm{d}\lambda$ *representa a medida de Gram em (2.20), então, na topologia fraca-estrela do espaço de Hilbert das funções definidas em* $[-1, 1]$,

$$\int_{\mathbb{R}} \mathrm{d}\lambda(x) \xrightarrow{\;w^*\;} \frac{1}{2} \int_{-1}^{1} \mathrm{d}x,$$

quando $N \longrightarrow \infty$.

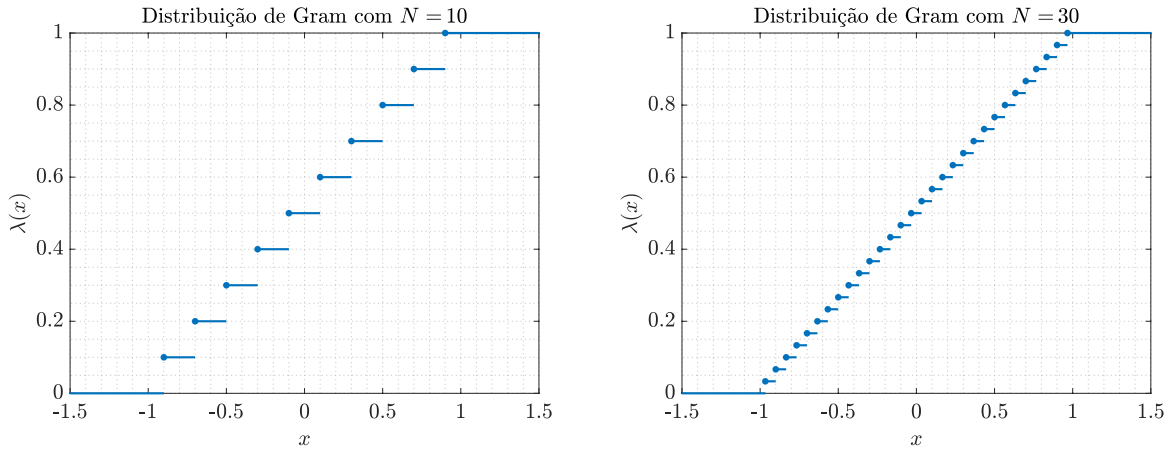

Figura 2.3: Esboços dos gráficos de distribuições λ de Gram com $N = 10$ e 30.

Demonstração. Seja a distribuição dada por $\Lambda(x) = x + k$, k constante real, com intervalo suporte $[-1, 1]$. Tomemos a partição deste intervalo como sendo o conjunto de pontos $\bar{x}_{j-1} \leq x_j \leq \bar{x}_j$, onde $\bar{x}_0 = -1$ e

$$\bar{x}_j = -1 + \frac{2j}{N}, \quad j = 1, 2, \ldots, N.$$

Para toda função f definida em $[-1, 1]$, o limite da soma de Riemann-Stieltjes implicará em

$$\begin{aligned}
\lim_{N\to\infty} \int_{\mathbb{R}} f(x)\,\mathrm{d}\lambda(x) &= \lim_{N\to\infty} \sum_{j=1}^{N} \frac{1}{N} f(x_j) = \lim_{N\to\infty} \sum_{j=1}^{N} \frac{\bar{x}_j - \bar{x}_{j-1}}{2} f(x_j) \\
&= \frac{1}{2} \lim_{N\to\infty} \sum_{j=1}^{N} f(x_j)[\Lambda(\bar{x}_j) - \Lambda(\bar{x}_{j-1})] \\
&= \frac{1}{2} \int_{-1}^{1} f(x)\,\mathrm{d}\Lambda(x) \\
&= \frac{1}{2} \int_{-1}^{1} f(x)\,\mathrm{d}x.
\end{aligned}$$

□

Corolário 2.11. *O limite quando $N \longrightarrow \infty$ do produto interno $\langle \cdot, \cdot \rangle$ com medida de Gram é a metade do produto interno com a medida de Legendre.*

Demonstração. Basta ver que $f = u \cdot v$ com $u, v \in \mathbb{P}$, é uma função definida em $[-1, 1]$ e utilizar o teorema anterior. □

Teorema 2.12. *Os termos da fórmula de recorrência dos polinômios ortonormais de Gram (2.11) convergem para os respectivos termos da relação de recorrência dos polinômios $\sqrt{2}\,\tilde{P}_n$, $n = 0, 1 \ldots$, onde $\tilde{P}_n(x) = \sqrt{2n+1} P_n(x)/\sqrt{2}$ é o polinômio ortonormal de Legendre. Além disto, o limite*

$$\lim_{N\to\infty} G_n(x; N) = \sqrt{2n+1}\, P_n(x)$$

é uniforme em $[-1, 1]$.

Demonstração. Basta observar o colorário anterior e o resultado de Gautschi [27, Teo. 3.1]. □

3 Quadratura de Gauss

Uma regra ou fórmula de quadratura consiste em aproximar numericamente uma integral da forma

$$\int_{\mathbb{R}} f(x)\,\mathrm{d}\lambda(x)$$

por meio de uma combinação linear de avaliações da função f. Combinações não-lineares ocorrem ocasionalmente.

A fórmula de quadratura que apresentaremos neste capítulo tem sua origem nos trabalhos de Gauss no início do século XIX. Em sua célebre obra [28] o autor utilizou frações contínuas para obter a regra que aproxima integrais com medida de Legendre $\mathrm{d}\lambda(x) = \mathrm{d}x$. Nela está apresentada a aproximação

$$\int_{10^5}^{2\cdot 10^5} \frac{1}{\log x}\,\mathrm{d}x \;\approx\; 8406.24312,$$

por meio da regra que hoje leva o nome de Gauss-Legendre. O resultado acima tem todos seus 9 dígitos corretos. Gauss calculou-o comparando sucessivas aplicações da sua regra usando um número cada vez maior de avaliações da função integrando, de um a sete. Isto é garantido pela convergência do método, como explicaremos adiante. Ressaltamos que as mesmas comparações aplicadas às regras de Newton-Cotes ou de Clenshaw-Curtis [29, §2.5], sob as mesmas condições de arredondamento, forneceriam apenas 5 dígitos corretos. Como explicaremos neste capítulo, a supremacia das fórmulas de Gauss são devidas ao seu grau máximo de precisão.

Christoffel [30] generalizou o resultado de Gauss para outras medidas contínuas incluindo a de Jacobi $\mathrm{d}\lambda(x) = (1-x)^\alpha(1+x)^\beta\,\mathrm{d}x$, $\alpha, \beta > -1$ cujos casos particulares são os clássicos de Chebyshev de 1ª e 2ª espécies ($\alpha = \beta = \mp 1/2$), de Gegenbauer ($\alpha = \beta \neq 1/2$), além de Legendre ($\alpha = \beta = 0$).

Neste capítulo demonstraremos os principais resultados para fornecer a fórmula geral para uma quadratura gaussiana. Nossas principais referências são [7, 31, 32]. Em seguida nos dedicaremos àquela com medida discreta de Gram e discutiremos os métodos numéricos para obtenção dos nós e pesos desta quadratura.

3.1 Preliminares

Uma *regra de quadratura* com m nós distintos para uma medida $\mathrm{d}\lambda$ é uma fórmula do seguinte tipo

$$\int_{\mathbb{R}} f(x)\,\mathrm{d}\lambda(x) = \sum_{k=1}^{m} \mathrm{W}_{m,k} f(\mathrm{z}_{m,k}) + \mathrm{R}_m(f), \tag{3.1}$$

onde a combinação linear de avaliações de f fornece a aproximação da integral e R_m é o erro cometido na aproximação. Os números $\mathrm{z}_{m,k}$ são chamados de *nós* e os coeficientes $\mathrm{W}_{m,k}$ são os *pesos* da quadratura.

Definição 3.1 (Grau de precisão)**.** Dizemos que a regra de quadratura (3.1) possui *grau de precisão* d se $\mathrm{R}_m(u) = 0$ para todo $u \in \mathbb{P}_d$ e $\mathrm{R}_m(u) \neq 0$ para algum $u \in \mathbb{P}_{d+1}$.

Dizemos que a regra de quadratura (3.1) é *interpolatória* se ela for obtida por interpolação, ou seja, se ela for obtida por meio da integração do polinômio interpolador de Lagrange nos nós $\mathrm{z}_{m,k}$, $k = 1, \ldots, m$ com medida $\mathrm{d}\lambda$. Krylov [31, Cap. 6] mostra que para uma regra de quadratura com m nós ser interpolatória é necessário e suficiente que a mesma possua grau de precisão, no mínimo, $m - 1$. Vejamos como as quadraturas de Gauss possuem grau máximo de precisão.

Teorema 3.2 (Krylov [31, p. 101])**.** *Dado um inteiro j com $0 \leq j \leq m$, a regra (3.1) possui grau de precisão $d \geq m - 1 + j$ se, e somente se, as condições são satisfeitas:*

(i) *a fórmula (3.1) é interpolatória;*

(ii) *se $j \geq 1$, então $\displaystyle\int_{\mathbb{R}} \pi_m(x)u(x)\,\mathrm{d}\lambda(x) = 0$ para o polinômio nodal*

$$\pi_m(x) := \prod_{k=1}^{m}(x - \mathrm{z}_{m,k}) \tag{3.2}$$

e todo $u \in \mathbb{P}_{j-1}$.

Demonstração. O caso $j = 0$ é evidente. Consideremos $j \geq 1$. É imediato que a condição (i) é necessária. Também é necessário que (ii) seja válida uma vez que $\pi_m u \in \mathbb{P}_{m-1+j}$ e, consequentemente,

$$\int_{\mathbb{R}} \pi_m(x)u(x)\,\mathrm{d}\lambda(x) = \sum_{k=1}^{m} \mathrm{W}_{m,k}\pi_m(\mathrm{z}_{m,k})u(\mathrm{z}_{m,k}) = 0,$$

pois $\pi_n(\mathrm{z}_{m,k}) = 0$.

Agora provaremos que ambas condições são suficientes. Para tal, basta provar que $\mathrm{R}_m(u) = 0$ para qualquer $u \in \mathbb{P}_{m-1+j}$. Notemos que a divisão euclidiana de tal polinômio u pelo polinômio nodal π_m implica na igualdade

$$u = q\pi_m + r, \quad q \in \mathbb{P}_{j-1},\ r \in \mathbb{P}_{m-1}.$$

Logo,

$$\int_{\mathbb{R}} u(x)\,\mathrm{d}\lambda(x) = \int_{\mathbb{R}} q(x)\pi_m(x)\,\mathrm{d}\lambda(x) + \int_{\mathbb{R}} r(x)\,\mathrm{d}\lambda(x).$$

A primeira integral no lado direito da equação acima anula-se por (ii), pois $q \in \mathbb{P}_{j-1}$. Por (i), a segunda integral é igual a $\displaystyle\sum_{k=1}^{m} \mathrm{W}_{m,k}r(\mathrm{z}_{m,k})$, uma vez que $r \in \mathbb{P}_{m-1}$. Porém,

$$r(\mathrm{z}_{m,k}) = u(\mathrm{z}_{m,k}) - q(\mathrm{z}_{m,k})\underbrace{\pi_m(\mathrm{z}_{m,k})}_{=0} = u(\mathrm{z}_{m,k}).$$

Daí,

$$\int_{\mathbb{R}} u(x)\,\mathrm{d}\lambda(x) = \sum_{k=1}^{m} \mathrm{W}_{m,k}u(\mathrm{z}_{m,k}),$$

ou seja, $\mathrm{R}_m(u) = 0$. □

Perceba que a condição (ii) implica na ortogonalidade de π_m a todos polinômios $u \in \mathbb{P}_{j-1}$. Além disto, note que o valor $j = m$ é ótimo. De fato, $j = m + 1$ implicaria que π_m seria ortogonal aos polinômios de grau até m, incluindo ele mesmo, o que é incabível. Logo a condição (ii) com $j = m$ implica que $\pi_m(x) = \phi_m(x; \mathrm{d}\lambda)$, ou seja, os nós da quadratura $\mathrm{z}_{m,k}$ são os zeros do polinômio ortogonal de grau m e, neste caso, o grau de precisão é $d \geq 2m - 1$. A regra de quadratura ótima (3.1) com $j = m$ é chamada de *quadratura de Gauss* com relação à medida $\mathrm{d}\lambda$.

Agora vejamos um limite máximo para o grau de precisão da fórmula (3.1).

Teorema 3.3 (Krylov [31, p. 102])**.** *Se a medida* $\mathrm{d}\lambda$ *é positiva, então o grau de precisão da fórmula quadratura* (3.1) *é menor do que* $2m$*.*

Demonstração. Uma vez que a medida $\mathrm{d}\lambda$ é positiva, temos

$$\int_{\mathbb{R}} [\pi_m(x)]^2 \, \mathrm{d}\lambda(x) \; > \; 0,$$

pois $\pi_m(x) = \phi_m(x; \mathrm{d}\lambda)$ no caso ótimo. Mas o somatório $\displaystyle\sum_{k=1}^{m} \mathrm{W}_{m,k} \pi_n^2(\mathrm{z}_{m,k})$ é nulo porque $\pi_m(\mathrm{z}_{m,k}) = 0$ para $k = 1, \ldots, m$. Consequentemente, a fórmula (3.1) não poderia ser exata para um polinômio de grau $2m$. □

Portanto, a quadratura de Gauss tem a seguinte forma

$$\int_{\mathbb{R}} f(x) \, \mathrm{d}\lambda(x) = \sum_{k=1}^{m} W_{m,k} f(z_{m,k}) + R_m(f), \tag{3.3}$$

onde $z_{m,k}$, $k = 1, \ldots, m$, são os zeros de $\phi_m(\cdot\,; \mathrm{d}\lambda)$ com erro $R_m(\mathbb{P}_{2m-1}) = 0$. Logo, o grau de precisão da quadratura de Gauss é o máximo possível e igual a $2m - 1$. Como a regra é interpolatória, então os pesos são

$$W_{m,k} = \int_{\mathbb{R}} l_k(x) \, \mathrm{d}\lambda(x), \quad l_k(x) := \frac{\pi_m(x)}{(x - z_{m,k}) \pi_m'(z_{m,k})}, \quad k = 1, \ldots, m. \tag{3.4}$$

Os pesos são positivos porque

$$0 \; < \; \int_{\mathbb{R}} [l_i(x)]^2 \, \mathrm{d}\lambda(x) = \sum_{k=1}^{m} W_{m,k} [l_i(z_{m,k})]^2 \; = \; W_{m,i}, \quad i = 1, \ldots, m.$$

Note que

$$\mu_0(\mathrm{d}\lambda) \; = \; \int_{\mathbb{R}} \mathrm{d}\lambda(x) \; = \; \lim_{x \to \infty} \lambda(x) - \lim_{x \to -\infty} \lambda(x) \; > \; 0.$$

Por outro lado, $\displaystyle\int_{\mathbb{R}} \mathrm{d}\lambda(x) = \sum_{k=1}^{m} W_{m,k}$, pois o integrando é a função constante igual a 1. Logo, para todo m,

$$\sum_{k=1}^{m} W_{m,k} = \mu_0(\mathrm{d}\lambda) \; > \; 0.$$

A desigualdade anterior garante a convergência da quadratura de Gauss [31, Seção 12.3] para funções f contínuas quando o intervalo suporte da medida for finito $[a, b]$, isto é,

$$\lim_{m \to \infty} \sum_{k=1}^{m} W_{m,k} f(z_{m,k}) \; = \; \int_a^b f(x) \, \mathrm{d}\lambda(x).$$

Logo, em tais condições, a quadratura com medida de Gram é convergente. O próximo resultado trata da expressão para o erro.

Teorema 3.4 (Krylov [31, p. 104])**.** *Se f é de classe C^{2m} sobre o intervalo suporte $[a,b]$ de $\mathrm{d}\lambda$, então o erro $R_m(f)$ da fórmula de Gauss* (3.3) *é dado por*

$$R_m(f) = \frac{f^{(2m)}(\xi)}{(2m)!} \int_{\mathbb{R}} [\phi_m(x;\mathrm{d}\lambda)]^2 \,\mathrm{d}\lambda(x), \quad \xi \in (a,b). \tag{3.5}$$

Demonstração. Seja $h_{2m-1}(\cdot\,;f)$ o conhecido polinômio interpolador de Hermite (ver [16, p. 382]) de grau $2m-1$ tal que

$$h_{2m-1}(z_{m,k};f) = f(z_{m,k}), \quad h'_{2m-1}(z_{m,k};f) = f'(z_{m,k}), \quad k = 1,\ldots,m.$$

É conhecido que

$$f(x) = h_{2m-1}(x;f) + r_{2m-1}(x;f), \tag{3.6}$$

com

$$r_{2m-1}(x;f) = \frac{f^{(2m)}(\xi(x))}{(2m)!}\,[\phi_m(x;\mathrm{d}\lambda)]^2, \qquad \xi(x) \in (a,b), \tag{3.7}$$

pois os nós $z_{m,k}$ estão contidos em (a,b) conforme o Teorema 2.5. Integrando (3.6) e utilizando a fórmula (3.3),

$$\sum_{k=1}^{m} W_{m,k} f(z_{m,k}) + R_m(f) = \int_{\mathbb{R}} h_{2m-1}(x;f)\,\mathrm{d}\lambda(x) + \int_{\mathbb{R}} r_{2m-1}(x;f)\,\mathrm{d}\lambda(x).$$

Mas a primeira integral no lado direito da equação acima é igual ao somatório no lado esquerdo, pois $f(z_{m,k}) = h_{2m-1}(z_{m,k};f)$. Então o resultado fica provado ao usar (3.7) e o Teorema do valor médio para integrais. □

3.2 Quadratura de Gauss do tipo Gram

A conhecida quadratura de Gauss-Legendre aproxima integrais com medida $\mathrm{d}\lambda(x) = \mathrm{d}x$ sobre o intervalo $[-1,1]$, ou seja, integrais da forma

$$\int_{-1}^{1} f(x)\,\mathrm{d}x$$

com erro

$$R_m(f) \;=\; \frac{2^{2m+1}\,(m!)^4}{((2m)!)^2\,(2m+1)!}\, f^{(2m)}(\xi), \qquad \xi \in (-1,\,1).$$

Por meio da fórmula de Stirling,

$$\|R_m(f)\|_\infty \;=\; \mathcal{O}\left(\frac{\sqrt{m}}{2^{4m}}\left(\frac{e}{2m+1}\right)^{2m+1}\right)\|f^{(2m)}\|_\infty,$$

onde $\|f\|_\infty := \sup\{\,|f(x)| \,:\, x \in [-1,1]\}$. A relação anterior mostra como o erro da quadratura de Gauss-Legende decresce rapidamente para zero quando $m \longrightarrow \infty$. Consequentemente, é razoável esperar que uma maneira eficiente de aproximar um somatório da forma

$$\frac{1}{N}\sum_{j=1}^{N} f(x_j(N))$$

pode ser dado ao escolhermos $\mathrm{d}\lambda$ sendo exatamente a medida de Gram para a quadratura de Gauss, já que esta é a "versão discreta" da quadratura de Gauss-Legendre. Esta ideia foi originalmente concebida por Area, Dimitrov, Godoy e Paschoa [2].

Diferentemente das quadraturas de medidas contínuas que levam o nome composto com o nome da própria medida (por exemplo, Gauss-Legendre quando $\mathrm{d}\lambda(x) = \mathrm{d}x$ em $[-1, 1]$, Gauss-Laguerre quando $\mathrm{d}\lambda(x) = e^{-x}\,\mathrm{d}x$ em $[0, \infty]$ etc), as quadraturas de medidas discretas não possuem uma nomenclatura consagrada na literatura. Neste trabalho, chamamos de *quadratura de Gauss com medida de Gram* ou, simplesmente, de *quadratura de Gauss do tipo Gram* àquela com tal medida.

3.2.1 Convergência e fórmula de erro

A quadratura de Gauss do tipo Gram é convergente para toda função contínua, pois o intervalo suporte da medida é finito. Além disto, se f é de classe C^{2m}, sempre existe um valor de m suficientemente grande para o qual o lado direito da desigualdade

$$|R_m(f)| \;\leq\; \frac{\|f^{(2m)}\|_\infty}{(2m)!}\;\|\phi_m(x;\mathrm{d}\lambda)\|^2$$

seja tão pequeno quanto se queira em $[-1+1/N,\;\; 1-1/N]$. Todavia um somatório como

$$\frac{1}{N}\sum_{j=1}^{N} f(x_j(N)) \tag{3.8}$$

sequer exige que tal função f seja definida em todo intervalo de integração de Gram. Aqui vale destacar que os resultados das quadraturas de Gauss ainda são significativos mesmo quando a classe de f é, no máximo, d, com $d < 2m$. Além disto, quanto menor a diferença $2m - d$, mais rápida será a convergência. De fato, Davis e Rabinowitz [29, p. 100] mostraram como a rapidez da convergência de Gauss-Legendre aumenta (para $m = 2, 3, 4, 10, 16, 32, 48$) à medida que a função f torna-se cada vez mais suave: desde uma função descontínua até uma função de classe C^1 que não é de classe C^2. Em nossas pesquisas anteriores [33, p. 137], verificamos que as regras de Newton-Cotes (cujo grau de precisão é, no máximo, m) apresentam desempenho análogo nestas condições. Contudo, mesmo quando $d < 2m$, as quadraturas de Gauss ainda fornecem resultados mais exatos devido ao grau máximo de precisão.

Em todo caso, assumindo que os dados $f(x_j(N))$ são "bem comportados", então podemos considerar empiricamente que a tal função f seja suave o suficiente para ser estendida à classe C^{2m}.

Buscamos agora um limite superior para $\|f^{(2m)}\|_\infty/(2m)!$. O Teorema do valor médio para diferenças divididas estabelece que $\|f^{(2m)}\|_\infty/(2m)!$ é igual à diferença dividida em $2m+1$ pontos no intervalo de integração. Logo, poderíamos combinar $2m+1$ pontos dentre os N pontos x_j disponíveis para realizar as diferenças divididas. Em seguida, tomaríamos a maior diferença dividida para ser o limite superior de $\|f^{(2m)}\|_\infty/(2m)!$. Apesar de existirem $N!\,/\,[(2m+1)!(N-2m-1)!]$ combinações possíveis, algumas poucas comparações poderiam ser o bastante.

Contudo, ainda que não haja *a priori* um controle de $\|f^{(2m)}\|_\infty/(2m)!$ quando m cresce, o valor de $\|\phi_m(x;\mathrm{d}\lambda)\|^2$ decresce fortemente para zero tanto quanto desejável independentemente de f. De fato, como ϕ_m representa o polinômio mônico de Gram de

grau m, então, por (2.17),

$$\|\phi_m(x;\mathrm{d}\lambda)\|^2 \;=\; \frac{2^{-2m}}{\prod_{j=0}^{m-1}\alpha_j^2} \;=\; 2^{-2m}\prod_{j=1}^{m}\frac{j^2}{N^2}\left(\frac{N^2-j^2}{j^2-\frac{1}{4}}\right) \;<\; 2^{-2m}\prod_{j=1}^{m}\frac{j^2}{j^2-\frac{1}{4}} = \prod_{j=1}^{n}\frac{j^2}{4j^2-1}.$$

Mas, $\displaystyle\prod_{j=1}^{m} 4j^2-1 = \left[\prod_{j=1}^{m}(2j-1)\right]\cdot\left[\prod_{j=1}^{m}(2j+1)\right]$. Pelos valores fracionais em [34, (6.1.8) e (6.1.12)] temos, após algumas simplificações,

$$\|\phi_m(x;\mathrm{d}\lambda)\|^2 \;<\; \frac{\pi}{2^{2m+1}\left(m+\frac{1}{2}\right)}\frac{\Gamma^2(m+1)}{\Gamma^2\left(m+\frac{1}{2}\right)}.$$

Mas a Desigualdade de Gautschi [35, (5.6.4)] implica que

$$\frac{\Gamma(m+1)}{\Gamma\left(m+\frac{1}{2}\right)} \;<\; (m+1)^{\frac{1}{2}}.$$

Logo, $\|\phi_m(x;\mathrm{d}\lambda)\|^2 \;<\; \dfrac{\pi}{2^{2m+1}}\left(\dfrac{m+1}{m+\frac{1}{2}}\right) \;<\; \dfrac{\pi}{2^{2m+1}}\,\dfrac{4}{3}$, para todo $m\geq 1$. Daí,

$$\|\phi_m(x;\mathrm{d}\lambda)\|^2 \;<\; \frac{2\pi}{3}\left(\frac{1}{4}\right)^m.$$

A desigualdade anterior é melhor do que aquela obtida em [3, p. 2219]. Portanto, para todo $\varepsilon>0$, a condição $\|\phi_m(x;\mathrm{d}\lambda)\|^2<\varepsilon$ é satisfeita sempre que tomamos

$$m > \log_4\frac{2\pi}{3\varepsilon}.$$

Finalmente apresentamos a fórmula de erro da quadratura gaussiana do tipo Gram:

$$R_m(f) = \frac{f^{(2m)}(\xi)}{4^m(2m)!}\prod_{j=1}^{m}\frac{j^2}{N^2}\left(\frac{N^2-j^2}{j^2-\frac{1}{4}}\right), \qquad \xi\in\left(-1+\tfrac{1}{N},\;1-\tfrac{1}{N}\right).$$

3.2.2 Cálculo dos zeros

Sejam os zeros

$$g_{m,k}(N) := g_{m,k}, \qquad k=1,2,\ldots,m,$$

do polinômio de Gram $G_m(\cdot\,;N)$ arranjados em ordem crescente,

$$-1+\frac{1}{N} \;<\; g_{m,1} \;<\; g_{m,2} \;<\ldots<\; g_{m,m} \;<\; 1-\frac{1}{N}.$$

Como já explicamos, os nós da quadratura são os zeros acima. Contudo, o método mais elegante e tradicional para cálculo dos zeros e também dos pesos das quadraturas de Gauss baseia-se no problema de cálculo de autovalores e autovetores de uma matriz simétrica e tridiagonal chamada matriz de Jacobi. Seus elementos são formados pelo termos da relação de recorrência dos polinômios ortonormais. Os autovalores desta matriz são os zeros $z_{m,k}$. Já os pesos podem ser obtidos em função dos autovetores. Tal caracterização foi observada por Wilf [36, p. 80 exerc. 9]. O celebrado algoritmo de Golub e Welsch [37] calcula os autovalores e autovetores da matriz de Jacobi e o código em FORTRAN

[38] implementa tal algoritmo com um custo computacional $\mathcal{O}(m^2)$ menor do que em MATLAB com complexidade $\mathcal{O}(m^3)$, conforme [39]. O código em FORTRAN [38] pode ser adaptado facilmente para a quadratura de Gram. Contudo, é amplamente conhecido que o algoritmo de Golub e Welsch (GW) não é o mais preciso para cálculos dos zeros e pesos das quadraturas de Gauss, conferir, por exemplo, [39–42]. Em geral, as outras técnicas mais precisas baseiam-se no cálculo dos zeros do polinômio ortogonal e os pesos são obtidos por fórmulas conhecidas.

Com boas aproximações iniciais, o método de Weierstrass-Dochev-Durand-Kerner (WDDK) pode calcular os zeros do polinômio de Gram com convergência quadrática, conforme [3]. Explicitamente, o método WDDK aproxima os zeros do polinômio $G_m(\cdot\,; N)$ por meio da iteração

$$g_{m,k}^{(\nu+1)} = g_{m,k}^{(\nu)} - \frac{G_m(g_{m,k}^{(\nu)}; N)}{b_{m,m} \prod\limits_{\substack{j=1 \\ j\neq k}}^{m} \left(g_{m,k}^{(\nu)} - g_{m,j}^{(\nu)}\right)}, \qquad \nu = 0, 1, \ldots, \tag{3.9}$$

onde $g_{m,k}^{(0)}$ são as aproximações iniciais e $b_{m,m}$ é o coeficiente líder de $G_m(\cdot\,; N)$. Este método funciona como uma modificação do popular método de Newton-Raphson para o cálculo simultâneo de todos os zeros do polinômio, com isso, a implementação do algoritmo é facilmente vetorizada. De fato, veja que as iterações do método de Newton-Raphson seriam

$$g_{m,k}^{(\nu+1)} = g_{m,k}^{(\nu)} - \frac{G_m(g_{m,k}^{(\nu)}; N)}{G_m'(g_{m,k}^{(\nu)}; N)}, \qquad \nu = 0, 1, \ldots.$$

Mas $G_m'(g_{m,k}; N) = b_{m,m} \prod_{\substack{j=1 \\ j\neq k}}^{m} (g_{m,k} - g_{m,j})$. Logo, se $g_{m,i}^{(\nu)} \approx g_{m,i}$, para todo $i = 1, 2, \ldots, m$, então o denominador em (3.9) seria aproximadamente o denominador do método de Newton-Raphson.

Agora discutiremos um pouco sobre aproximações iniciais para os zeros de Gram. Começaremos com alguns importantes resultados obtidos em [2] e [3]. Sejam os zeros do polinômio de Legendre P_m arranjados em ordem crescente,

$$-1 < \; x_{m,1} \; < \; x_{m,2} \; < \ldots < \; x_{m,m} \; < \; 1.$$

O seguinte resultado apresenta desigualdades para os zeros positivos de Gram em função dos zeros de Legendre.

Teorema 3.5 (Area, Dimitrov, Godoy e Paschoa [2])**.** *As desigualdades*

$$\sqrt{x_{m,k}^2 - \frac{m^2\zeta_1}{N^2}} \;\leq\; g_{m,k}(N) \;\leq\; \sqrt{x_{m,k}^2 + \frac{m^2\zeta_2}{N^2}},$$

são válidas para todo $m \leq N-1$, com $k = \lfloor (m-1)/2 \rfloor + 1, \ldots, m$, onde $\zeta_1 \approx 1.00518$ e $\zeta_2 \approx 0.0161824$ ou 0.0144593 se m for par ou ímpar, respectivamente.

Demonstração. Ver [2, pp. 9-15]. □

Devido à simetria dos zeros, é possível obter desigualdades para os zeros negativos, por reflexão. O teorema anterior evidencia a proximidade dos zeros de ambos polinômios à medida que N aumenta. Por meio deste último resultado, os mesmos autores também introduziram as seguintes desigualdades que aqui adaptamos para os polinômios ortonormais.

Teorema 3.6 (Area, Dimitrov, Godoy e Paschoa [3])**.** *Para todo $m \geq 1$, os polinômios de Gram satisfazem às seguintes desigualdades*

$$|G_m(x_{m,k}; N)| \;\leq\; \zeta_1\, b_{m,m}\, \frac{m^2}{N^2}, \qquad k = 1, 2, \ldots, m.$$

Demonstração. Se m é ímpar então $g_{m,(m+1)/2} = x_{m,(m+1)/2} = 0$ e a desigualdade é trivial. Note que

$$G_m(x; N) = b_{m,m}(x^2 - g_{m,1}^2)\ldots(x^2 - g_{m,\lfloor m/2\rfloor}^2)x^\tau,$$

onde $\tau = 0$ ou 1, conforme m for par ou ímpar, respectivamente. Como os zeros de Gram e de Legendre são, em módulo, menores do que 1, então o quadrado deles pertence ao intervalo $(0, 1)$. Logo, para todo $k = 1, 2, \ldots, \lfloor m/2\rfloor$,

$$|G_m(x_{m,k}; N)| \;=\; b_{m,m}|x_{m,k}^2 - g_{m,k}^2| \prod_{\substack{j=1\\ j\neq k}}^{\lfloor m/2\rfloor} |x_{m,k}^2 - g_{m,j}^2||x_{m,k}^\tau| \;\leq\; b_{m,m}|x_{m,k}^2 - g_{m,k}^2|.$$

O Teorema 3.5 completa a prova ao utilizarmos a simetria dos zeros destes polinômios. □

O resultado anterior evidencia como os zeros de Legendre podem ser excelentes aproximações iniciais para os zeros de Gram, principalmente à medida que N cresce. De fato, ζ_1/N^2 converge rapidamente para zero e, pelo coeficiente líder do polinômio de Legendre [8, p. 39], é óbvio que

$$\lim_{N\longrightarrow\infty} m^2 b_{m,m} = \frac{\Gamma(2m+1)}{2^m\Gamma^2(m)}\sqrt{2m+1}.$$

Para determinados valores de m e N, as aproximações iniciais $g_{m,k}^{(0)} = x_{m,k}$ garantem a convergência de WDDK, ver [3, Teo. 3.1]. Contudo, para evitar o cálculo explícito de $x_{m,k}$, sugerimos o limite superior das desigualdades de Förster e Petras [43, Teo. 1],

$$x_{m,m-k+1} \leq \cos\left\{\theta_{m,k} + \frac{1}{8(m+\frac{1}{2})^2}\left[1 - \frac{6 + \frac{1}{4}(9 - 2\cos^2\theta_{m,k})}{12(m+\frac{1}{2})^2\,\mathrm{sen}^2\,\theta_{m,k}}\right]\mathrm{cotg}\,\theta_{m,k}\right\}, \qquad (3.10)$$

onde $\theta_{m,k} := (k - \frac{1}{4})/(m + \frac{1}{2})\,\pi$, com $k = 1, \ldots, \lfloor (m+1)/2\rfloor$ para serem as aproximações iniciais para os zeros não negativos. As aproximações para os zeros negativos são obtidas por reflexão. Destacamos que, dentro do nosso conhecimento, (3.10) é uma das melhores desigualdades elementares (que não dependem de zeros de outras funções) para os zeros não negativos de Legendre. Além disto, pelos testes numéricos que realizamos, estabelecemos a seguinte afirmação.

Conjectura 3.7. *Os zeros não negativos dos polinômios de Gram $G_m(\cdot\,; N)$ e de Legendre satisfazem às seguintes desigualdades*

$$g_{m,m-k+1}(N) \;\leq\; x_{m,m-k+1}, \qquad k = 1, \ldots, \lfloor (m+1)/2\rfloor.$$

Os zeros estão mais próximos para $k = \lfloor (m+1)/2\rfloor$, sendo a igualdade atingida se m for ímpar.

A conjectura anterior implica que os zeros positivos de Gram convergem por baixo para os zeros de Legendre, isto é $g_{m,m-k+1}(N) \nearrow x_{m,m-k+1}$ à medida que $N \longrightarrow \infty$, para

cada $k = 1, 2, \ldots, \lfloor (m+1)/2 \rfloor$. Na verdade, a conjectura implica que os zeros estão mais próximos do que já demonstrado em [2, Eq. (4.13)].

Realizamos experimentos para testar a convergência do método WDDK com a aproximação em (3.10) com diversos valores de m e N. Poucas iterações foram requeridas para obter o zero na exatidão máxima do computador `eps` $\approx$ `2.2204e-16` em ponto flutuante. Para checarmos os resultados, consideramos que os valores exatos eram aqueles obtidos por GW em precisão quádrupla. Já o método de Newton-Raphson com as mesmas aproximações iniciais não convergiu para alguns zeros. Por exemplo, para $N = 10^3$ com $m = 92, 93, 98$ e 99 algumas aproximações iniciais para os maiores zeros convergiram erroneamente. A Figura 3.1 mostra que houve perturbação para x próximo de 1 levando à falha do método. Pelo gráfico vemos que $g^{(0)}_{93,998} \longrightarrow g_{93,997}$ e $g^{(0)}_{93,999} \longrightarrow g_{93,1000}$.

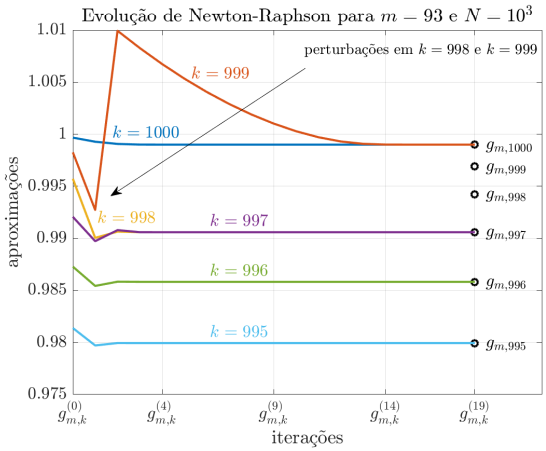

Figura 3.1: Perturbações no método de Newton.

Neste caso em que o método de Newton-Raphson falhou, temos a razão $m/\sqrt{N} \approx 2.9$. Aqui é relembrado o apontamento de Barnard *et al.* [19] a respeito dos grandes valores de $\|G_m(\cdot\,; N)\|_\infty$ quando $m/\sqrt{N} > 2.5$ e que ocorrem próximo aos extremos do intervalo $[-1, 1]$, justamente onde Newton-Raphson ficou perturbado. Pelos nossos testes,

$$\|G'_m(\cdot\,; N)\|_\infty \gg \|G_m(\cdot\,; N)\|_\infty.$$

Como tal método utiliza a derivada de $G_m(\cdot\,; N)$, então é normal que os cálculos fiquem mais sensíveis nesta região. Cremos ser este o motivo da sua falha porque o método WDDK (que não usa a derivada) convergiu normalmente neste caso. Estas observações reforçaram ainda mais nossa predileção por WDDK.

Agora vamos mostrar como o método WDDK é mais preciso do que o tradicional método GW. Para isto seja a medição para o erro absoluto máximo dado por

$$\max_{k=1,\ldots,m} \left| v_k - v_k^{\text{true}} \right|$$

onde v_k^{true} representa o valor exato do vetor $(v_1, v_2, \ldots, v_m)$. A Figura 3.2 compara a exatidão dos dois métodos. A linha preta que aparece na figura à esquerda delimita o valor de `eps`/2. Os valores que não aparecem nos gráficos, correspondem aos valores exatos em precisão dupla. Na Figura 3.2 (à esq.), o método WDDK calcula todos os zeros de $G_{60,10^3}$ com precisão máxima e os valores são sempre melhores ou iguais aos de GW. Na Figura 3.2 (à dir.), vemos o erro absoluto máximo para vários valores de

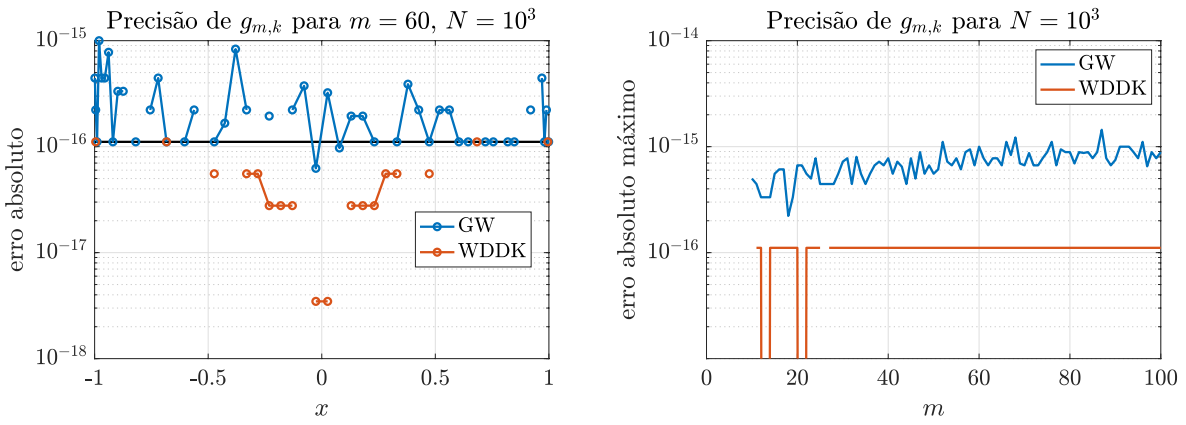

Figura 3.2: Comparação da precisão para cálculo dos zeros por meio de GW e WDDK.

$m = 10, 11, \ldots, 100$. Nela percebemos a exatidão máxima constante de WDDK enquanto que, GW sofre um pequeno aumento no erro à medida que m cresce. Perceba que, apesar da razão $m/\sqrt{N}$ ultrapassar 2.5 levemente nos valores finais $m = 80, 81, \ldots, 100$, o método WDDK foi estável e convergente. Vários outros testes semelhantes foram realizados para diferentes valores de m e N. Em todos eles WDDK foi mais preciso do que GW.

Ao observamos a convergência dos zeros de $G_m(\cdot\,; N)$ para os zeros de Legendre de grau m quando $N \longrightarrow \infty$, descobrimos empiricamente que, para $N \geq 10^{10}$, os dois zeros podem ser considerados iguais em dupla precisão, como mostrado na Figura 3.3. A partir

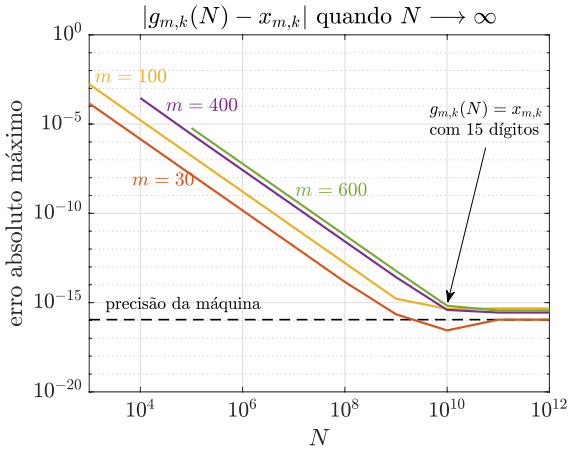

Figura 3.3: Convergência dos zeros de Gram para os de Legendre.

de $N = 10^{10}$ a diferença absoluta entre os zeros alcança a precisão da máquina. Usamos aqui os zeros de Legendre $x_{m,k}$ calculados pelo código `legpts` do software Chebfun [44] que reúne técnicas modernas e muito precisas para os zeros deste polinômio [39, 45]. Pelo gráfico, vê-se claramente que

$$g_{m,k}(N) \;\approx\; x_{m,k} \;+\; \mathcal{O}(N^{-2}m^2),$$

em total consonância com o Teorema 3.6 para os polinômios mônicos $G_m(\cdot\,; N)/b_{m,m}$.

3.2.3 Cálculo dos pesos

Como mostrado os pesos $W_{m,k}$, $k = 1, \ldots, m$, podem ser obtidos pelas fórmulas (3.4). Contudo, existem fórmulas mais adequadas a fim de evitar as integrações. Tais fórmulas podem ser obtidas por meio da Identidade de Christoffel-Darboux [7, p. 43 Teo. 3.2.2] com o uso da relação de recorrência de três termos, ver [16, Eqs. (8.4.17) e (8.4.18)]. No caso de Gram, temos

$$W_{m,k} = \frac{2\alpha_{m-1}}{G_{m-1}(g_{m,k}(N); N) G'_m(g_{m,k}(N); N)}, \tag{3.11}$$

Também temos a curiosa relação que independe do cálculo da derivada do polinômio de Gram:

$$W_{m,k} = \left[\sum_{j=0}^{m-1} \left(G_j(g_{m,k}(N); N) \right)^2 \right]^{-1}. \tag{3.12}$$

Apesar de parecer menos atraente do ponto de vista numérico devido ao somatório dos quadrados, a fórmula (3.12) é mais estável do que (3.11), como mostraremos adiante. Foi Lether [46, p. 51] quem observou este fato para a quadratura de Gauss-Legendre com os valores de $m = 10$, 30, 96 e 256. Há uma terceira possibilidade para a fórmula dos pesos [16, Eq. (8.4.16)]. Entretanto esta última possui precisão equivalente à de (3.11) com a sobrecarga da avaliação de um polinômio de grau ainda maior.

Uma vez que os zeros $g_{m,k}$ são bem calculados pelo método WDDK, podemos comparar as fórmulas (3.11) e (3.12) com os valores obtidos pelo tradicional método GW. Para isto, seja a medição do erro a seguir

$$\text{erro relativo máximo} \;=\; \max_{k=1,\ldots,m} \left| \frac{v_k - v_k^{\text{true}}}{v_k^{\text{true}}} \right|,$$

para um dado vetor $(v_1, v_2 \ldots, v_m)$. As comparações estão na Figura 3.4. Usamos os resultados de GW em precisão quádrupla para serem os valores exatos. A fórmula com derivada (3.11) apresenta um grande erro nas proximidades dos extremos do intervalo $[-1+1/N,\ 1-1/N]$ muito provavelmente devido à grande oscilação de $\|G'(\cdot\,; N)\|_\infty$, como já mencionado na subseção anterior. Contudo, a fórmula sem derivada (3.12) apresenta melhor exatidão quando comparada à GW.

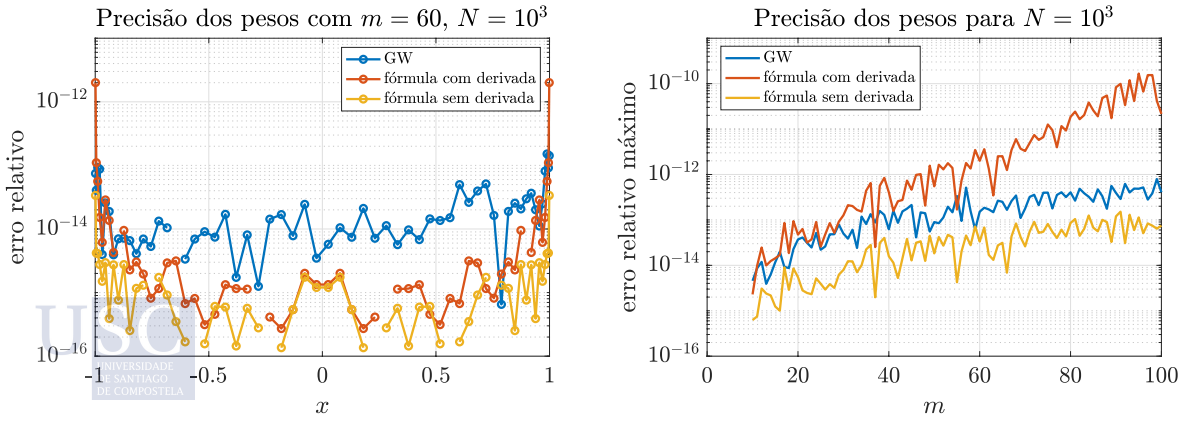

Figura 3.4: Comparação da precisão para cálculo dos pesos.

Demais testes com vários valores de m e N comprovaram a supremacia da fórmula (3.12) sobre as outras duas.

3.2.4 Cálculo da quadratura

Os métodos para cálculos dos zeros e pesos já está estabelecido pelo que foi mostrado nas subseções anteriores. Então, *a priori*, para realizar a quadratura bastaria efetuarmos

$$I_m(f) := \sum_{k=1}^{m} W_{m,k} f(g_{m,k}(N)).$$

Contudo, é comum que desconheçamos a função f geradora dos dados $f(x_j)$, $j = 1, \ldots, N$. Logo não poderíamos avaliar $f(g_{m,k}(N))$ diretamente nestes casos. Para contornar este problema, recuperamos os dados $f(g_{m,k}(N))$ com base nas informações parciais $f(x_j)$. Especificamente o Lema de Smolyak [47, p. 75 Lema 5.7] garante que, dentre os melhores métodos de recuperação, existe um que realiza uma combinação linear das informações. Os splines cúbicos naturais são obtidos por meio de tais combinações e em [3] eles foram usados para recuperar as informações $f(g_{m,k})$. Realizamos diversas comparações entre as precisões dos splines naturais bem como de interpolações lineares para os dados perturbados $f(x_j) + \delta(j)$, onde $\delta(j)$ representa a perturbação no dado $f(x_j)$. A interpolação linear é calculada por meio dos pontos cujas abscissas estão mais próximas de $g_{m,k}$. A competitividade de ambos métodos é grande e, em diversos casos, as aproximações lineares podem fornecer os melhores resultados a um custo computacional muito mais baixo, como mostra a tabela a seguir.

Tabela 3.1: Complexidade para recuperar m dados.

método	adições	multiplicações	divisões
splines ($s \geq 4$ pontos)	$(14s+4)m$	$5s+3$	$(3s+3)m$
interpolação linear	$4m$	$2m$	m

Vejamos agora a precisão da quadratura para os dados perturbados

$$f(x_j) = e^{x_j} + 10^{-2}\delta_{10^4}(j), \quad j = 1, 2, \ldots, 10^4,$$

onde $\delta_N :=$ `randn(1,N)` é um vetor com N números pseudo aleatórios normalmente distribuídos do MATLAB[1]. Aproximadamente, a perturbação relativa acima é tal que

$$10^{-6} \;\leq\; \frac{10^{-2}\delta_{10^4}(j)}{e^{x_j}} \;\leq\; 10^{-2}.$$

Mostramos na Tabela 3.2 o erro relativo da quadratura com $m = 30$ zeros quando usamos diferentes métodos para recuperar os dados $f(g_{30,k}(10^4))$, $k = 1, \ldots, 30$.

Tabela 3.2: Erro relativo na quadratura usando diferentes métodos para recuperar $f(g_{30,k})$.

Método	Erro
interpolação linear	8.9709e-04
splines (4 pts.)	9.6824e-04
splines (6 pts.)	1.0386e-03
splines (8 pts.)	1.0550e-03
splines (10 pts.)	1.0587e-03

[1] `https://www.mathworks.com/help/matlab/ref/randn.html`

Devido ao baixo custo computacional e eficiência para recuperar os dados, optamos por utilizar as interpolações lineares. Ilustramos na Figura 3.5 a convergência da quadratura para aproximar somas dos dados suaves

$$f(x_j) = x_j e^{x_j}, \quad j = 1, \ldots, 10^4$$

e também a soma dos dados perturbados

$$f(x_j) = x_j e^{x_j} + 10^{-4}\delta_{10^4}(j), \quad j = 1, \ldots, 10^4,$$

com perturbação relativa variando entre 10^{-8} e 10^{0}.

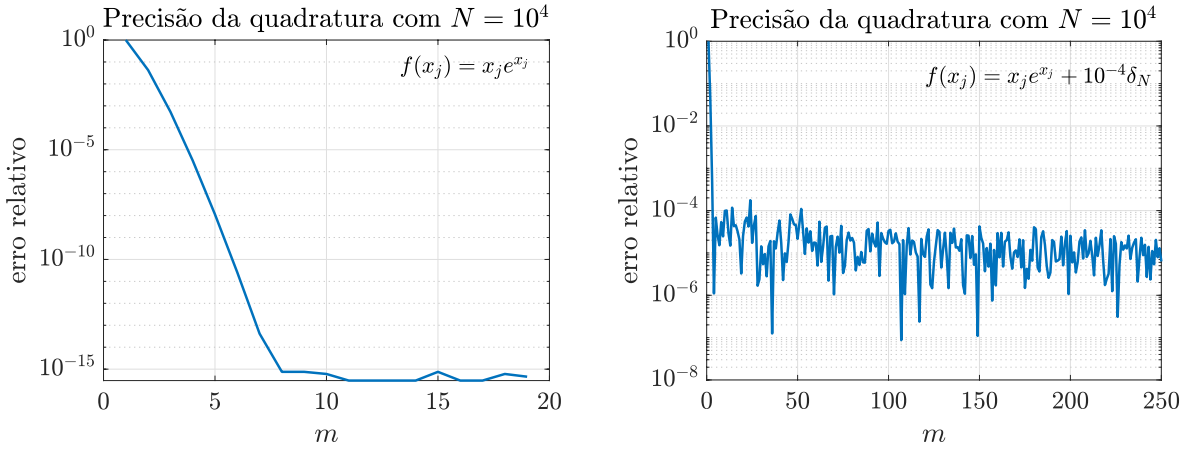

Figura 3.5: Precisão da quadratura para diferentes funções.

Evidentemente, com a função suave, a convergência é rápida e atinge a precisão da máquina com $m = 8$ zeros. No caso da função perturbada, a convergência também é rápida mas estabiliza-se em torno de 10^{-5} com um leve aumento na precisão à medida que m cresce até $2.5\sqrt{N}$. Isto era esperado, pois a função perturbada evidentemente não é de classe C^{2m} e a aproximação fica condicionada à aproximação linear e também ao ajuste da suavidade dos dados (como explicamos na Subseção 3.2.1).

4 Métodos conhecidos para solução do problema de mínimos quadrados

Neste capítulo apresentaremos os métodos mais populares e difundidos para o problema de mínimos quadrados. O primeiro deles trata simplesmente de resolver o sistema das equações normais que guia à solução. Os outros baseiam-se na decomposição de uma matriz do tipo Vandermonde. Discutiremos também as complexidades e algumas particularidades de cada método. A obra de Åke Björck [22] apresenta um tratado geral sobre o tema. Ao final exibimos um código em MATLAB que implementa um dos métodos.

Para o conjunto de dados $(x_j, f(x_j))$ para $j = 1, \ldots, N$, o problema de mínimos quadrados também pode ser formulado da seguinte maneira. Inicialmente seja o polinômio $p(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \ldots + a_n\varphi_n(x)$ formado por uma base $\{\varphi_k\}_{k=0}^n$ de $\mathbb{P}_n$ e o sistema sobredeterminado $\mathbf{Va} = \mathbf{f}$,

$$\begin{bmatrix} \varphi_0(x_1) & \varphi_1(x_1) & \ldots & \varphi_n(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \ldots & \varphi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_N) & \varphi_1(x_N) & \ldots & \varphi_n(x_N) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix}, \tag{4.1}$$

no qual $\mathbf{V} := \mathbf{V}(\varphi_0, \ldots, \varphi_n; x_1, \ldots, x_N)$ é uma matriz do tipo Vandermonde[1] de ordem $N \times (n+1)$, o vetor $\mathbf{a} \in \mathbb{R}^{n+1}$ contém os coeficientes de p e o vetor $\mathbf{f} := \mathbf{f}(f; x_1, \ldots, x_N) \in \mathbb{R}^N$ contém os dados $f(x_j)$, $j = 1, \ldots, N$. Perceba que o problema de mínimos quadrados consiste em encontrar o vetor $\mathbf{a}$ tal que $\mathbf{f}$ seja a melhor aproximação para $\mathbf{Va}$. Em outras palavras, desejamos minimizar o resíduo

$$\|\mathbf{f} - \mathbf{Va}\|_2$$

no senso da norma ℓ^2 usual.

4.1 Método das equações normais

Teorema 4.1 (Björck [22, p. 5])**.** *O vetor* $\mathbf{a}$ *é a solução do seguinte sistema linear de ordem* $n+1$*,*

$$\mathbf{V}^{\mathrm{T}}\mathbf{Va} = \mathbf{V}^{\mathrm{T}}\mathbf{f}. \tag{4.2}$$

Demonstração. Pela caracterização da melhor aproximação (Teorema 1.2), todo vetor da forma $\mathbf{Vv}'$ com $\mathbf{v}' \in \mathbb{R}^{n+1}$ é ortogonal à $\mathbf{f} - \mathbf{Va}$. Em particular, é ortogonal à cada vetor coluna da matriz $\mathbf{V}$. Matricialmente falando, devemos ter $\mathbf{V}^{\mathrm{T}}(\mathbf{f} - \mathbf{Va}) = \mathbf{0}$. □

[1] A matriz do tipo Vandermonde em (4.1) é a matriz de Vandermonde usual onde trocamos os monômios x^k por φ_k, para todo $k = 0, \ldots, n$.

O sistema de ordem $n+1$ em (4.2) representa exatamente as equações normais em (1.2) com $\mathbf{f} = u$ e $\mathbf{V}^{\mathrm{T}}\mathbf{V}$ igual à matriz de Gram

$$\mathbf{G}(\varphi_0, \ldots, \varphi_n) = \begin{bmatrix} \langle \varphi_0, \varphi_0 \rangle_2 & \langle \varphi_1, \varphi_0 \rangle_2 & \cdots & \langle \varphi_n, \varphi_0 \rangle_2 \\ \langle \varphi_0, \varphi_1 \rangle_2 & \langle \varphi_1, \varphi_1 \rangle_2 & \cdots & \langle \varphi_n, \varphi_1 \rangle_2 \\ \vdots & \vdots & \ddots & \vdots \\ \langle \varphi_0, \varphi_n \rangle_2 & \langle \varphi_1, \varphi_n \rangle_2 & \cdots & \langle \varphi_n, \varphi_n \rangle_2 \end{bmatrix},$$

onde $\langle \cdot\,, \cdot \rangle_2$ representa o produto interno da norma ℓ^2 para os vetores coluna da matriz $\mathbf{V}$.

Na literatura, o método para resolução do problema de mínimos quadrados via sistema (4.2) é normalmente chamado de *método das equações normais*, ver [22, Seção 2.2]. Por meio da Desigualdade de Cauchy-Schwarz é fácil verificar que para todo vetor $v \in \mathbb{R}^{n+1}$, temos $v^{\mathrm{T}}\mathbf{G}v > 0$. Logo a matriz de Gram é definida positiva. Consequentemente ela admite a decomposição de Cholesky para a solução do sistema ao custo de $\mathcal{O}(n^3)$ operações. Contudo, apesar da aparente simplicidade deste método, uma base monomial $\varphi_k = x^k$, $k = 0, \ldots, n$, pode deixar a matriz de Gram extremamente malcondicionada, sensibilizando todos os métodos que utilizam sua decomposição. Sobretudo, além da matriz de Vandermonde ocupar grande espaço de memória computacional, a composição da matriz de Gram tem um custo de $\mathcal{O}(n^2 N)$ operações e a composição do vetor $\mathbf{V}^{\mathrm{T}}\mathbf{f}$ tem um custo de $\mathcal{O}(nN)$ operações. Por outro lado, Demanet e Townsend [48] mostraram que, se as bases $\{\varphi_k\}_{k=0}^n$ são os polinômios de Legendre $\{P_k\}_{k=0}^n$ ou de Chebyshev $\{T_k\}_{k=0}^n$, o sistema das equações normais fica bem condicionado para pontos x_j, $j = 1, \ldots, N$, igualmente espaçados. Explicitamente, para $n \leq \frac{1}{2}\sqrt{N}$,

$$\kappa_2(\mathbf{G}(P_0, \ldots, P_n)) \leq \sqrt{5(2n+1)}$$

e

$$\kappa_2(\mathbf{G}(T_0, \ldots, T_n)) \leq \sqrt{375(2n+1)/2},$$

onde $\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2$ é o número de condição da matriz A segundo a norma espectral. Além disto, a base de Chebyshev pode reduzir o custo da composição da matriz de Gram para $\mathcal{O}(n^3)$ operações [48, Apêndice A]. Contudo, apesar da melhor estabilidade do método nestes casos, o seu custo geral ainda dependeria de N para compor o vetor $\mathbf{V}^{\mathrm{T}}\mathbf{f}$.

4.2 Método da decomposição em valores singulares

Teorema 4.2 (Björck [22, p. 15])**.** *O vetor* $\mathbf{a}$ *é dado por*

$$\mathbf{a} = \sum_{k=1}^{n+1} \frac{(u^{(k)})^{\mathrm{T}}\mathbf{f}}{\sigma_k} v^{(k)},$$

onde $u^{(k)}$, $k = 1, \ldots, N$, *são as colunas da matriz ortogonal* $U \in \mathbb{R}^{N \times N}$, $v^{(k)}$, $k = 1, \ldots, n+1$, *são as colunas da matriz ortogonal* $V \in \mathbb{R}^{(n+1)\times(n+1)}$ *e* $\sigma_1, \sigma_2, \ldots, \sigma_{n+1}$ *são os valores singulares*[2] *de* $\mathbf{V}$ *tais que* $\mathbf{V} = U\Sigma V^{\mathrm{T}}$ *com*

$$\Sigma = \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{N \times (n+1)}$$

e $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_{n+1})$.

[2] O número real não negativo σ é valor singular de $\mathbf{V}$ se existirem u e v unitários tais que $\mathbf{V}v = \sigma u$ e $\mathbf{V}^{\mathrm{T}}u = \sigma v$.

Demonstração. Sejam a decomposição em valores singulares $\mathbf{V} = U\Sigma V^{\mathrm{T}}$ (ver [6, Teo. 2.5.2 p. 70] e [22, Teo. 1.2.1 p. 9]) e os vetores

$$\mathbf{z} = V^{\mathrm{T}}\mathbf{a} \in \mathbb{R}^{n+1} \qquad \text{e} \qquad \mathbf{y} = U^{\mathrm{T}}\mathbf{f} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \in \mathbb{R}^N, \tag{4.3}$$

com $y_1 \in \mathbb{R}^{n+1}$. Logo,

$$\begin{aligned} \|\mathbf{f} - \mathbf{V}\mathbf{a}\|_2 &= \|U^{\mathrm{T}}(\mathbf{f} - \mathbf{V}VV^{\mathrm{T}}\mathbf{a})\|_2 = \|U^{\mathrm{T}}\mathbf{f} - U^{\mathrm{T}}\mathbf{V}VV^{\mathrm{T}}\mathbf{a}\|_2 = \|U^{\mathrm{T}}\mathbf{f} - \Sigma V^{\mathrm{T}}\mathbf{a}\|_2 \\ &= \left\| \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} \Sigma_1\mathbf{z} \\ 0 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} y_1 - \Sigma_1\mathbf{z} \\ y_2 \end{pmatrix} \right\|_2. \end{aligned}$$

Daí o erro $\|\mathbf{f} - \mathbf{V}\mathbf{a}\|_2$ é mínimo quando $\mathbf{z} = \Sigma_1^{-1}y_1$. Consequentemente, por (4.3), $\mathbf{a} = V\Sigma_1^{-1}y_1$. □

Apesar deste método ser independente da matriz de Gram, ele requer grande espaço de memória computacional porque cada vetor coluna da matriz U tem N elementos. Com isso, seu tempo computacional e sua complexidade para os cálculos pode superar às requeridas pelo método das equações normais (incluindo a decomposição de Cholesky para a solução deste último).

4.3 Método QR

Teorema 4.3 (Björck [22, p. 20])**.** *O vetor* $\mathbf{a}$ *é a solução do sistema triangular inferior* $R\mathbf{a} = D^{-1}Q^{\mathrm{T}}\mathbf{f}$ *no qual* $\mathbf{V} = QR$*, onde* $Q \in \mathbb{R}^{N\times(n+1)}$ *é uma matriz cujas colunas são vetores ortogonais,* $R \in \mathbb{R}^{(n+1)\times(n+1)}$ *é uma matriz triangular superior unitária e* $D = Q^{\mathrm{T}}Q \in \mathbb{R}^{(n+1)\times(n+1)}$ *é uma matriz diagonal.*

Demonstração. Como $\mathbf{V}$ possui $n+1$ colunas linearmente independentes, então ela admite a decomposição $\mathbf{V} = QR$ nas condições acima, ver [18, Teo. 5.7.3 p. 201]. Como a matriz de Gram tem determinante não nulo, então temos garantida a existência de $(\mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}$. Logo, podemos escrever

$$\mathbf{a} = (\mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{f}.$$

Se D é a matriz diagonal de ordem $n+1$ tal que $Q^{\mathrm{T}}Q = D$, então, substituindo $\mathbf{V}$ pela sua decomposição QR na equação acima e usando a existência de R^{-1}, temos

$$\begin{aligned} \mathbf{a} &= ((QR)^{\mathrm{T}}QR)^{-1}(QR)^{\mathrm{T}}\mathbf{f}, \\ &= (R^{\mathrm{T}}Q^{\mathrm{T}}QR)^{-1}R^{\mathrm{T}}Q^{\mathrm{T}}\mathbf{f}, \\ &= (R^{\mathrm{T}}DR)^{-1}R^{\mathrm{T}}Q^{\mathrm{T}}\mathbf{f}, \\ &= (R^{-1}D^{-1}R^{-\mathrm{T}})R^{\mathrm{T}}Q^{\mathrm{T}}\mathbf{f}, \\ &= R^{-1}D^{-1}Q^{\mathrm{T}}\mathbf{f}, \\ R\mathbf{a} &= D^{-1}Q^{\mathrm{T}}\mathbf{f}. \end{aligned}$$

□

Assim, a solução é prontamente obtida por meio da substituição retroativa no sistema triangular $R\mathbf{a} = y$, onde $y = D^{-1}Q^{\mathrm{T}}\mathbf{f}$. Existem diversos algoritmos [22, §2.4] para calcular os fatores Q e R que utilizam, por exemplo, transformações de Householder e rotações de Givens. Há também o método de Gram-Schmidt modificado [18, p. 201] para obtenção destes fatores. Contudo, assim como nos casos anteriores, persistirão os cálculos excessivos, pois o fator Q possui N linhas.

4.4 Código

Para testar e comprovar a eficiência do nosso algoritmo em precisão dupla, deveremos compará-lo a outros métodos já consagrados. O poderoso comando da barra invertida \ em MATLAB[3] obtém a solução **a** por meio da decomposição QR e é usado no código a seguir que chamaremos de STANDARD. Este foi sugerido gentilmente por nossos revisores[4] em nosso trabalho [4] e é um pouco mais eficiente do que o comando `polyfit()`, pois este último realiza algumas chamadas de verificação de parâmetros e calcula os fatores Q e R explicitamente antes de usar \. Os parâmetros de entrada são os pontos x_j no vetor coluna `x`, os dados $f(x_j)$ no vetor coluna `f` e o grau do polinômio em `n`. Os coeficientes são retornados no vetor `a` para a base monomial. A matriz do tipo Vandermonde é montada e armazenada em `V` sendo $\varphi_k = x^k$, $k = 0, \ldots, n$.

```
function a = standard(x,f,n)
N = numel(f);
V = zeros(N, n+1);
V(:,1) = ones(N,1,class(x));
for j = 2:n+1
    V(:,j) = x.*V(:,j-1);
end
a = V\f;
end
```


[3]Para mais detalhes, `https://www.mathworks.com/help/matlab/ref/mldivide.html`.

[4]Propusemos inicialmente os demais métodos apresentados neste capítulo para realizar as comparações, mas os revisores indicaram exclusivamente o código STANDARD para esta finalidade.

5 Um novo algoritmo para solução do problema clássico dos mínimos quadrados

Seja o caso específico quando os pontos x_j, $j = 1, \ldots, N$, são igualmente espaçados. Sem perda de generalidade, sejam x_j os pontos de aumento da distribuição de Gram e multipliquemos as equações do sistema (4.2) por $1/N$. Se usamos a base $\{\varphi_k\}_{k=0}^{n}$ sendo os polinômios ortonormais de Gram, então a solução do problema é dada por

$$a_k = \frac{1}{N} \sum_{j=1}^{N} f(x_j) G_k(x_j; N), \qquad k = 0, \ldots, n, \tag{5.1}$$

que são os coeficientes de Fourier. Obviamente qualquer decomposição de matrizes seria totalmente desnecessária e, para todo $n = 0, \ldots, N-1$,

$$\kappa_2(\mathbf{G}(G_0, \ldots, G_n)) = 1,$$

o que garante uma estabilidade perfeita ao resultado. Por outro lado, cada somatório em (5.1) para $k = 0, \ldots, n$, (que representa um elemento do vetor $\mathbf{V}^{\mathrm{T}}\mathbf{f}$) requer todos os dados da matriz do tipo Vandermonde e novamente muito espaço de memória é requerido para N grande. Logo o custo computacional para efetuar os somatórios (ou montar $\mathbf{V}^{\mathrm{T}}\mathbf{f}$) é $\mathcal{O}(nN)$.

Até aqui as ideias são conhecidas e podem ser encontradas na literatura. Há uma implementação que pode ser encontrada em Gautschi [32, p. 219].

A principal ideia do nosso trabalho e que foi originalmente apresentada em [2] e [3] é aproximar os coeficientes (5.1) pela quadratura do tipo Gram e, com isso, criar um algoritmo cuja complexidade seja independente da quantidade N para dados "bem comportados", tornando-o muito mais rápido do que os demais conhecidos na atualidade.

A seguir apresentamos o algoritmo e damos um limite para o erro cometido na aproximação em função dos polinômios de Gram. Os códigos também são apresentados.

5.1 Algoritmo

Utilizando a quadratura de Gauss do tipo Gram com os zeros calculados pelo método WDDK e os pesos por meio da fórmula (3.12) estamos em condições de obter a solução na base dos polinômios de Gram,

$$p_n(x; f) = a_0 G_0(x; N) + a_1 G_1(x; N) + \ldots + a_n G_n(x; N), \tag{5.2}$$

onde, para cada $k = 0, 1, \ldots, n$,

$$a_k \;\approx\; \sum_{j=1}^{m} W_{m,j}\, f(g_{m,j}(N))\, G_k(g_{m,j}(N); N) \;=\; I_m(f \cdot G_k(\cdot\,; N)), \qquad m \ll N. \tag{5.3}$$

Caso os coeficientes sejam obtidos dentro da precisão requerida, então a complexidade do número de operações do nosso algoritmo é $\mathcal{O}(1)$ em relação ao número N, significando que o o custo computacional não se altera à medida que N aumenta. Evidentemente, além das multiplicações em a_k, $k = 0, \ldots, n$, existe um custo que depende de n e m para obter: os zeros $g_{m,j}$, os pesos $W_{m,j}$ e as avaliações de f e G_k em $g_{m,j}$ para $j = 1, \ldots, m$. Os testes que realizaremos adiante colocarão à prova o custo total do nosso algoritmo em relação ao algoritmo STANDARD para diversos tipos de dados $\mathbf{f}$.

Caso a função f seja conhecida explicitamente, então podemos avaliá-la diretamente em $g_{m,j}(N)$. Caso contrário, iremos recuperar os dados $f(g_{m,j}(N))$ via interpolação linear. Finalmente basta realizar os somatórios em (5.3).

Para uma boa precisão nos somatórios (5.3) observamos o erro relativo

$$r_{m_0,m_1} := \frac{|I_{m_0}(f^2) - I_{m_1}(f^2)|}{I_{m_1}(f^2)}, \quad m_0 < m_1,$$

com $m_1 = \min\{100, \lfloor 2.5\sqrt{N} \rfloor\}$ e $m_0 = m_1 - 5$ para $I_{m_1}(f^2) > 1$. Caso $I_{m_1}(f^2) \leq 1$, observamos o erro absoluto.

Se a função f é fornecida explicitamente e

$$r_{m_0,m_1} \leq 10^{-15},$$

então adotamos $m = m_1$ em (5.3). Caso a desigualdade anterior não seja satisfeita, aumentamos os valores de m_1 e m_0 até que $r_{m_1,m_0} \leq 10^{-15}$. Daí adotamos $m = m_1$ para calcular (5.3). Nestes casos, para n fixo, o algoritmo tem complexidade $\mathcal{O}(1)$ em relação à N, pois o valor requerido m_1 dependerá exclusivamente da suavidade do integrando f^2.

Caso a função não seja fornecida explicitamente, isto é, quando temos apenas os dados $f(x_j)$, $j = 1, \ldots, N$, iremos adotar a desigualdade

$$r_{m_0,m_1} \leq 5 \cdot 10^{-5}, \tag{5.4}$$

como um critério para identificar se os dados estão demasiadamente perturbados. Se (5.4) é satisfeita, consideramos os dados razoavelmente pouco perturbados e adotamos $m = m_1$ em (5.3). Mas se a desigualdade não é satisfeita, isso implica que, neste caso excepcional, os dados provavelmente estão muito perturbados. Consequentemente o cálculo dos valores $f(g_{m,j})$ em (5.3), por aproximação linear, não será preciso. Logo, nesta situação peculiar, a quadratura não é utilizada. Portanto, lamentavelmente, aqui os coeficientes de Fourier a_k são obtidos através das suas próprias somas (5.1). Desta forma mantemos a idéia geral da expansão do polinômio $p_n(\cdot\,; f)$ mas, infelizmente, somos obrigados a deixar de utilizar outra idéia principal que é uso das fórmulas de quadratura para os coeficientes de Fourier. Por isto, como consequência, nestes casos, a complexidade do algoritmo será $\mathcal{O}(N)$ para n fixo, tornando-o bem mais lento. Contudo, ele ainda é mais rápido do que os métodos tradicionais, em geral. No último capítulo fornecemos exemplos de dados que obedecem nosso critério (5.4) e outros com dados realmente muito perturbados. Observamos que quando (5.4) é satisfeita — e logo a fórmula de quadratura é utilizada — o nosso algoritmo é muito mais rápido.

A primeira e indiscutível vantagem do nosso algoritmo é o pouco uso de memória requerida em relação aos outros métodos que necessitam montar matrizes e vetores com muitos elementos a depender de N.

Agora discutiremos uma estratégia que adotamos a fim de obter uma melhor exatidão para coeficientes de p_n. Quando algum dos coeficientes de Fourier for matematicamente nulo (isto ocorre, por exemplo, quando a função é muito bem aproximada por um polinômio), a quadratura obterá valores em torno da precisão da máquina — aproximadamente 10^{-16} —, o que é um fato perfeitamente esperado. Para zerarmos definitivamente estes valores, lembremos primeiramente que, por (1.8), aproximamos f no espaço de Hilbert $\mathbb{P}_{N-1}$ cuja base é $\{G_k(\cdot\,;N)\}_{k=0}^{N-1}$ com produto interno de Gram. Daí, a Identidade de Parseval implica que

$$\|f\|^2 = \sum_{k=0}^{N-1} a_k^2.$$

Mas, em virtude de (5.2),

$$\|p_n(\cdot\,;f)\|^2 = \sum_{k=0}^{n} a_k^2.$$

Além disto, p_n é a projeção ortogonal de f sobre $\mathbb{P}_{N-1}$. Logo, pela desigualdade triangular, temos $\|f\| - \|p_n(\cdot\,;f)\| \leq \|f - p_n(\cdot\,;f)\|$. Essas observações nos motivam a zerar os coeficientes de Fourier a_k tais que

$$\frac{|a_k|}{\|f\|} < \epsilon,$$

para um ϵ relativamente pequeno. Empiricamente elegemos $\epsilon = 5(r_m + 2\texttt{eps})$. Esta estratégia mostrou-se eficaz e nos guiará a resultados excelentes para obter os coeficientes de p_n na base monomial, como explicaremos a seguir.

O fluxograma geral do nosso algoritmo está na Figura 5.1. Adicionalmente, nosso algoritmo pode avaliar o polinômio p_n por meio do algoritmo de Clenshaw (Teorema 2.6). Ele também pode fornecer a solução p_n na base monomial, cuja avaliação pode ser dada pelo método de Horner [16, p. 28] que, como verificado, pode ser mais rápido do que o algoritmo de Clenshaw quando $N \geq 2 \cdot 10^4$.

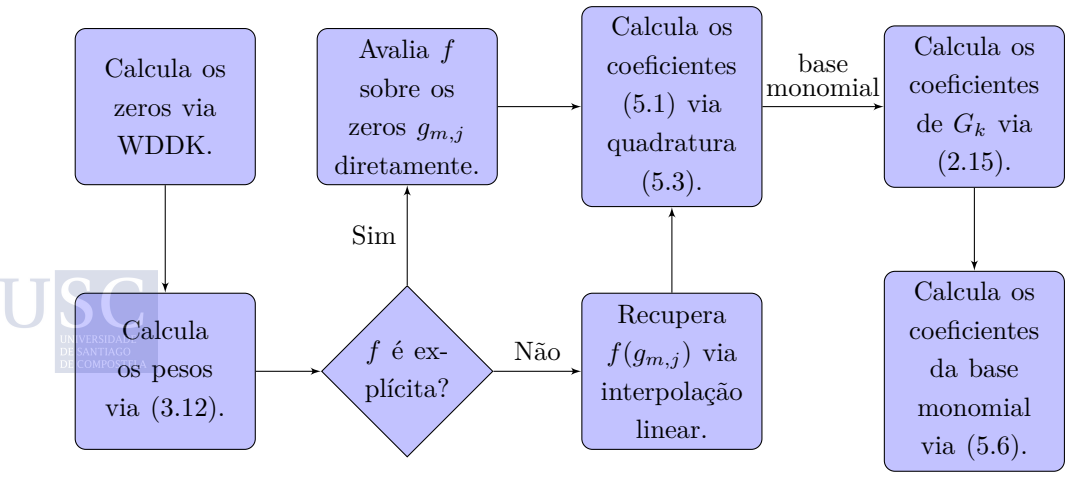

Figura 5.1: Novo algoritmo para solução do problema clássico dos mínimos quadrados.

5.1.1 Base monomial

O algoritmo STANDARD, bem como os demais métodos apresentados no capítulo anterior, fornecem os coeficientes de p_n na base monomial,

$$p_n(x; f) = c_{n,0} + c_{n,1}x + \ldots + c_{n,n}x^n. \tag{5.5}$$

Mas é fácil obtê-la observando a expansão de Maclaurin dos polinômios de Gram, isto é, sua expansão na base monomial,

$$G_k(x; N) = b_{k,0} + b_{k,1}x + \ldots + b_{k,k}x^k,$$

cujos termos podem ser calculados em precisão máxima via Teorema 2.7. Deste modo,

$$p_n(x; f) = \sum_{k=0}^{n} a_k \sum_{j=0}^{k} b_{k,j}\, x^j = \sum_{j=0}^{n} \left(\sum_{k=j}^{n} a_k b_{k,j} \right) x^j,$$

daí,

$$c_{n,j} = \sum_{k=j}^{n} a_k b_{k,j}, \qquad j = 0, \ldots, n. \tag{5.6}$$

Perceba que a estratégia de impor a condição $a_k \equiv 0$ para os coeficientes muito pequenos é de fundamental importância. Pois, caso contrário, os coeficientes $c_{n,j}$ nas equações (5.6) podem sofrer grandes erros de arredondamento na multiplicação de a_k por $b_{k,j}$. Por exemplo, $b_{30,30}$ é da ordem de 10^8. Logo, nesta situação nosso algoritmo produziria uma multiplicação da ordem de 10^{-8} enquanto esperaríamos um resultado próximo de zero em dupla precisão. Na verdade, como assinala Trefethen [49, p. 16], embora seja muito confortável e familiar, a base monomial nunca deveria ser usada para trabalhar com funções sobre um intervalo real. Assim, *a priori*, não seria recomendável realizar a passagem para esta base. Contudo, como já explicamos, nós zeramos tais coeficientes por meio da Identidade de Parseval e removemos os possíveis erros de arredondamento.

Uma vez que obtemos os coeficientes de p_n na base monomial, podemos avaliá-lo pelo conhecido algoritmo de Horner.

5.2 Estimativa de erro

O importante teorema devido a Brass [50], permite-nos estabelecer estimativas para o erro $f - p_n(\cdot\,; f)$.

Teorema 5.1 (Brass [50])**.** *Seja $\varphi_0, \varphi_1, \ldots$, uma sequência (finita ou infinita) de polinômios ortonormais com relação à medida $\mathrm{d}\lambda$ cujo suporte está contido em $[-1, 1]$ e que possui as seguintes propriedades:*

(i) $\displaystyle\int_{-1}^{1} f(x)\,\mathrm{d}\lambda(x) = \int_{-1}^{1} f(-x)\,\mathrm{d}\lambda(x)$, *para toda f contínua;*

(ii) $\|\varphi_k\|_\infty = \varphi_k(1)$, *para $k = 0, 1, \ldots, n+1$, onde $\|\varphi\|_\infty := \sup\{\, |\varphi(x)| \,:\, x \in [-1, 1]\}$.*

Se $f^{(n+1)}$ existe, então

$$\|f - p_n(\cdot\,; f)\|_\infty \leq \frac{\|\varphi_{n+1}\|_\infty}{\left\|\varphi_{n+1}^{(n+1)}\right\|_\infty} \left\|f^{(n+1)}\right\|_\infty.$$

Demonstração. Se ω_n denota o coeficiente líder de φ_n então, pela Identidade de Christoffel-Darboux [7, p. 43 Teo. 3.2.2],

$$R(f) := f - p_n(\cdot\,; f) = \frac{\omega_n}{\omega_{n+1}} \int_{-1}^{1} \frac{f(x) - f(t)}{x - t} [\varphi_{n+1}(x)\varphi_n(t) - \varphi_n(x)\varphi_{n+1}(t)]\, \mathrm{d}\lambda(t),$$

para x fixo. Definido a função $g(t) = \dfrac{f(x) - f(t)}{x - t}$, então

$$\|g^{(s)}\|_\infty \leq \frac{\|f^{(s+1)}\|_\infty}{s+1}, \qquad s = 0, 1, \ldots, n.$$

Logo,

$$\|R(f)\|_{s+1} \leq \frac{\omega_n}{\omega_{n+1}} \frac{1}{s+1} \|\varphi_{n+1}(x)C_n - \varphi_n(x)C_{n+1}\|_s, \quad C_k(g) := \int_{-1}^{1} g(t)\varphi_k(t)\, \mathrm{d}\lambda(t),$$

onde $\|A\|_s := \sup_{\|f^{(s)} \leq 1\|} \|A(f)\|$ é uma norma para funcionais A. Pela condição (i) é possível concluir que

$$\|R(f)\|_{s+1} \leq \frac{\omega_n}{\omega_{n+1}} \frac{1}{s+1} \|\varphi_{n+1}(1)C_n - \varphi_n(1)C_{n+1}\|_s,$$

$s = 0, 1, \ldots, n$. Utilizando o fato de que o polinômio $\varphi_{n+1}(1)\varphi_n - \varphi_n(1)\varphi_{n+1}$ tem 1 como zero e outros n zeros no intervalo $[-1, 1]$ (Szegő [7, p. 46]) e também considerando o polinômio interpolador para g com respeito aos tais outros zeros, prova-se que

$$\varphi_{n+1}(1)C_n(g) - \varphi_n(1)C_{n+1}(g) = \frac{g^{(n)}(\xi_1)}{n!} \frac{\varphi_{n+1}(1)}{\omega_n}.$$

Logo,

$$\|\varphi_{n+1}(1)C_n - \varphi_n(1)C_{n+1}\|_n = \frac{\varphi_{n+1}(1)}{n!\,\omega_n}$$

e com a última igualdade é possível concluir a prova. □

Teorema 5.2. *Se $f^{(n+1)}$ existe, então o resíduo $f - p_n(\cdot\,; f)$ para os dados $f(x_j)$, $j = 1, \ldots, N$, é tal que:*

$$\|f - p_n(\cdot\,; f)\|_\infty \leq \frac{\sqrt{\pi}\, G_{n+1}(1; N)}{2^{n+1}\sqrt{2n+3}\ \Gamma(n + \frac{3}{2})} \left\|f^{(n+1)}\right\|_\infty, \tag{5.7}$$

onde $\|f\|_\infty := \sup\{\, |f(x)| \,:\, x \in [-1, 1]\}$. Além disto,

$$\lim_{N \longrightarrow \infty} \|f - p_n(\cdot\,; f)\|_\infty \leq \frac{\sqrt{\pi}}{2^{n+1}\,\Gamma(n + \frac{3}{2})} \left\|f^{(n+1)}\right\|_\infty.$$

Demonstração. Por [19], temos $\|G_k(\cdot\,; N)\|_\infty = G_k(1; N)$ para todo $k = 0, 1, \ldots, N-1$. Também temos que

$$G_{n+1}^{(n+1)}(x; N) = (n+1)!\, 2^{n+1} \prod_{j=0}^{n} \alpha_j.$$

Expandindo o produtório acima e utilizando um procedimento análogo ao realizado na Subseção 3.2.1, obtemos $1/G_{n+1}^{(n+1)}(x; N) < \sqrt{\pi}/\left(2^{n+1}\sqrt{2n+3}\,\Gamma(n + \frac{3}{2})\right)$. A primeira desigualdade segue imediatamente por meio do Teorema 5.1. Por (2.19),

$$G_{n+1}(1; N) \longrightarrow \sqrt{2n+3},$$

quando $N \longrightarrow \infty$, pois $P_n(1) = 1$ para todo $n = 0, 1 \ldots$. Isto prova a segunda desigualdade. □

5.3 Código

A seguir apresentamos a implementação do nosso algoritmo em MATLAB. O código principal é `clsap`. Os parâmetros de entrada são o vetor `xj` (contendo os pontos x_j), o vetor `f` (contendo os dados $f(x_j)$) e o grau do polinômio é fornecido em `n`. O parâmetro de saída `c` retorna os coeficientes na base de Gram e também na base monomial, caso desejado.

Opcionalmente, o código também retorna: as avaliações do polinômio sobre os N dados em `p`, a soma residual de quadrados em `rss` e também o conhecido coeficiente de determinação para medir a qualidade do ajuste em `r2`.

Os subcódigos `rec` e `gram` são chamados dentro de `clsap` e servem para avaliar os polinômios de Gram e calcular os zeros e pesos da quadratura, respectivamente.

Todos códigos estão disponíveis em: `https://clsap.dcce.ibilce.unesp.br`.

5.3.1 clsap.m

```
function [c,p,rss,r2] = clsap(xj, f, n, varargin)
%%  CLSAP   Classical least squares approximation problem
%
%   [C] = CLSAP(XJ, F, N) computes the coefficients from the polynomial of
%   least squares of degree N in Gram basis given the equidistant data
%   F(XJ) in the interval [-1+1/NUP, 1-1/NUP], where NUP is a large number,
%   NUP>>N.
%
%   The polynomial is:
%      C(1)*G_N(x) + C(2)*G_{N-1}(x) +...+ C(N)*G_1(x) + C(N+1)*G_0(x),
%      where G_K is the orthonormal Gram polynomial of degree K and
%      parameter NUP.
%
%   CLSAP(XJ, F, N, EXPLICIT_F) allows the user furnishes the string
%   EXPLICIT_F with the function F of a variable x. It is highly
%   recommended when F is explicitly known.
%
%   [C] = CLSAP(XJ, F, N, 'MON') also computes the coefficients from the
%   polynomial in monomial basis in the matrix C, where the first row
%   contains the coefficients in Gram basis, and the second one contains
%   the coefficients in monomial basis, both in descending order of degree.
%
%   CLSAP(XJ, F, N, EXPLICIT_F, 'MON') or CLSAP(XJ, F, N, 'MON', EXPLICIT_F)
%   produces:
%          CLSAP(XJ, F, N, EXPLICIT_F) and CLSAP(XJ, F, N, 'MON').
%
%   [C,P] = CLSAP(XJ, F, N) returns also a vector P with the polynomial
%   evaluated at the points XJ.
%
%   [C,P,RSS] = CLSAP(XJ, F, N) returns also the residual sum of squares
%   RSS.
%
```

```
%   [C,P,RSS,R2] = CLSAP(XJ, F, N) returns also the coefficient of
%   determination R2.
%
%
%   Calls GRAM, REC.
%
%
%   Copyright 2021 by Dimitar K. Dimitrov and Lourenco L. Peixoto
%   All rights reserved.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CLSAP implemented by Lourenco L. Peixoto, 2021 - see [1-3].
%
%   Basic references:
%
%       [1] I. Area, D. K. Dimitrov, E. Godoy, and V. Paschoa, "Approximate
%   calculation of sums I: Bounds for the zeros of Gram polynomials", SIAM
%   J. Numer. Anal., 2014.
%
%       [2] I. Area, D. K. Dimitrov, E. Godoy, and V. Paschoa, "Approximate
%   calculation of sums II: Gaussian type quadrature", SIAM J. Numer.
%   Anal., 2016.
%
%       [3] D. K. Dimitrov and L. L. Peixoto, "An efficient algorithm for
%   the classical least squares approximation", SIAM J. Sci. Comput., 2020.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% standard parameters:
base = 'gram';
explicit_f = [];
flag = [];

if (nargin > 3)
    if (nargin > 4)
        if ~strcmpi(varargin{1},'mon')
            % clsap(xj, f, n, 'explicit_f', 'mon')
            base = 'mon';
            explicit_f = varargin{1};
        else
            % clsap(xj, f, n, 'mon', 'explicit_f')
            base = 'mon';
            explicit_f = varargin{2};
        end
    else
        if ~strcmpi(varargin{1},'mon')
            % clsap(xj, f, n, 'explicit_f')
            explicit_f = varargin{1};
        else
```

```
            % clsap(xj, f, n, 'mon')
            base = 'mon';
        end
    end
end

% number of data:
nup = length(xj);

% check the mesh-points:
if ( xj(1)~=-1+1/nup || xj(end)~=1-1/nup )
    error('The mesh-points must be in interval [-1+1/NUP, 1-1/NUP], NUP>>N.')
end

m_max = floor(2.5*sqrt(nup));
m0 = m_max;

% number of nodes for the quadrature:
m = min(100,m0);

% accuracy required for quadrature:
if ( isempty(explicit_f) )
    % noisy data:
    rm_max = 5e-5;
else
    % smooth data:
    flag = 1;
    rm_max = 1e-15;
end


% check the accuracy:
% computes F in the zeros of the Gram polynomial of degree M-5:
[g,w] = gram(m-5,nup);
[fg] = eval_f(f, g, nup, explicit_f);

 % quadrature to sum(f^2)/NUP with M-5 points:
q0 = w*transpose(fg.^2);

% computes F in the zeros of the Gram polynomial of degree M:
[g,w] = gram(m,nup);
[fg] = eval_f(f, g, nup, explicit_f);

 % quadrature to sum(f^2)/NUP with M points:
q1 = w*transpose(fg.^2);

% accuracy of quadrature to sum(f^2)/nup with M nodes:
if (q1 > 1)
```

```
    rm = abs(q1-q0)/q1;
else
    rm = abs(q1-q0);
end

% computes the Fourier coefficients:
if (rm <= rm_max) % computes by quadrature:

    % evaluates the Gram polynomials of degree 0,...,N with parameter NUP at
    % the zeros of the Gram polynomial of degree m:
    [~,MG] = rec(n,nup,transpose(g)); % MG is a (N+1)-by-M matrix

    % the vector with the inner products <F,G_k>, k=0,...,N, ie, the vector
    % with the Fourier coefficients:
    a = w.*fg*transpose(MG);

elseif (flag == 1) % increase M when the explicit function is furnished:
    % step:
    step = 10*floor(log10(nup));

    % increase M:
    m = m+step;

    while ( rm > rm_max && m <= m_max )
        q0 = q1;

        % check the accuracy:
        % computes F in the zeros of the Gram polynomial of degree M:
        [g,w] = gram(m,nup);
        [fg] = eval_f(f, g, nup, explicit_f);

        % quadrature to sum(f^2)/NUP with M points:
        q1 = w*transpose(fg.^2);

        % accuracy of quadrature to sum(f^2)/NUP with M nodes:
        if (q1 > 1)
            rm = abs(q1-q0)/q1;
        else
            rm = abs(q1-q0);
        end

        % increase M:
        m = m+step;
    end
    if (m > m_max)
        warning('clsap: Data very noisy. Precision in quadrature is %f.', rm)
    end
```

```
    % uses the quadrature:

    % evaluates the Gram polynomials of degree 0,...,N with parameter NUP at
    % the zeros of the Gram polynomial of degree M:
    [~,MG] = rec(n,nup,transpose(g)); % MG is a (N+1)-by-M matrix

    % the vector with the inner products <F,G_k>, k=0,...,N, ie, the vector
    % with the Fourier coefficients:
    a = w.*fg*transpose(MG);

else % data are very noisy (computes by summation):
    cte = sqrt(3*nup^2/(nup^2-1));

    % parity of NUP:
    nup_p = mod(nup,2);

    % only non negative mesh-points:
    z = xj(floor(nup/2)+1:nup);

    % polynomial of degree 0:
    g1 = ones(1,length(z));

    % polynomial of degree 1:
    g = cte*z;

    % reflect the values:
    if (nup_p) % odd nup
        g1x = [flip(g1), g1(2:end)];
        gx =  [-flip(g), g(2:end)];
    else % even nup
        g1x = [flip(g1), g1];
        gx = [-flip(g), g];
    end

    f = transpose(f);

    % prealocating:
    a = zeros(1,n+1);

    % Fourier coefficients a_0 and a_1:
    a(1) = g1x*f;
    a(2) = gx*f;

    k=1:n;
    alpha = sqrt(nup^2*(k.^2-.25)./(k.^2.*(nup^2-k.^2)));
    for k=2:n
        % initialise for polynomial:
        g2 = g1; g1 = g;
```

```
        % polynomial of degree k:
        g = 2*alpha(k)*z.*g1 - alpha(k)/alpha(k-1).*g2;

        % reflect the values:
        if (nup_p) % odd nup
            gx =  [(-1)^k*flip(g), g(2:end)];
        else  % even nup
            gx = [(-1)^k*flip(g), g];
        end

        % Fourier coefficient a_k:
        a(k+1) = gx*f;
    end

    % Fourier coefficients:
    a = a/nup;

    q1 = (transpose(f)*f)/nup;
    rm = 1e-15;
end


epsilon = (5*(rm + 2*eps))^2;

% find the null Fourier coefficients:
if (rm <= rm_max)
    lima = a.^2/q1;
    v = find(lima<=epsilon);
    a(v) = zeros(1,numel(v));
end

% coefficients in descending order of degree:
% c(1)*G_n(x) + c(2)*G_{n-1}(x) +...+ c(n)*G_1(x) + c(n+1)*G_0(x):
c = flip(a);

% monomial basis:
if strcmp(base,'mon')
    cm = monomial(n,nup,a); % monomial basis
    c = [c; cm]; % first row contains Gram basis, the second one contains
    % the monomial ones. Both in descending order of degree.
end
```



```
% Optionally evaluate the polynomial and compute RSS and R2:
if nargout>1 % polynomial at mesh-points:
    v = find(a~=0,1,'last'); % use the precise degree
    if nup >= 2e4 % Horner's method is faster
        [m1,~] = size(c);
```

```
        if m1 == 1
            cm = monomial(n,nup,a); % monomial basis
            c = [c; cm];
        end
        p = polyval(c(2,:),xj); % Horner's method
    else
        p = clenshaw(v-1,nup,a(1:v),xj); % Clenshaw's algorithm
    end
    if nargout>2 % the RSS:
        rss = sum((f-p).^2);
        if nargout>3 % the coefficient of determination R2:
            r2 = 1 - rss / ( sum(f.^2) - (sum(f))^2/nup );
        end
    end
end


end


function [c] = monomial(n,nup,a)
% Computes the coefficients in monomial basis.

% the matrix with the coefficients from all polynomials of degree 0,...,N:
B = coefmatrix_full(n,nup);

% the coefficients in monomial basis:
c = sum(transpose(a).*B,1);

c = flip(c);

end

function [fg] = eval_f(f, g, nup, explicit_f)
%     EVAL_F evaluate F in the points G. It uses linear interpolant when
%     the function is unknown.

% evaluate f(g):
if ischar(explicit_f)
    % eval_f(f, g, nup, explicit_f)
    x=g;
    fg = transpose(eval(explicit_f));
else
    % eval_f(f, g, nup, [])
    fg = linear_interp(f,g,nup);
end
```

```
end


function [fg] = linear_interp(f,g,nup)
% approximate the data F(G) using linear interpolant between the mesh
% points, where x_t < G < x_{t+1}, t=0,...,NUP-2.

% identify every zero G between the mesh points:
t = transpose(floor(.5*nup*(g+1-1/nup)));

x0 = -1+(1+2*t)/nup; x1 = x0+2/nup; % abscissas
y0 = f(t+1); y1 = f(t+2);           % F(x0), F(x1)
a1 = (y1-y0)./(x1-x0);              % linear coefficient
a0 = y0-a1.*x0;                     % independent coefficient
fg = a0+a1.*transpose(g);           % linear interpolant
end




function B = coefmatrix_full(max,nup)
% Compute all coefficients from Gram polynomials of degree = 0,...,MAX

% B is the square matrix of order MAX+1 with the coefficients of the
% orthonormal Gram polynomials G_0, G_1, ..., G_MAX with parameter NUP.
%
%       | b_{0,0}     0         0       0   ...    0|
%       | b_{1,0}  b_{1,1}      0       0   ...    0|
%       | b_{2,0}  b_{2,1}   b_{2,2}    0   ...    0|
%   B = |    .                                     .|
%       |    .                                     .|
%       |    .                                     .|
%       | b_{n,0}  b_{n,1}   b_{n,2}   ...   b_{n,n}|
%

B = zeros(max+1);

B(1,1) = 1;

if (max>0)
    k=1:max;
    alp = sqrt(nup^2*(k.^2-.25)./(k.^2.*(nup^2-k.^2))); % alp(k)=alpha_{k-1},
    % where alpha is a term in the REC of the orthonormal polynomials.

    B(2,2) = 2*alp(1);

    if (max>1)
        B(3,3) = 2*alp(2)*B(2,2); B(3,1) = -alp(2)/alp(1);
```

```
        for i=4:max+1
            im1 = i-1;
            im2 = im1-1;
            B(i,i) = 2*alp(im1)*B(im1,im1);
            for j=i-2:-2:2
                B(i,j) = 2*alp(im1)*B(im1,j-1) - alp(im1)/alp(im2)*B(im2,j);
            end
            B(i,1) = -alp(im1)/alp(im2)*B(im2,1);
        end
    end
end

end
```

```
function s = clenshaw(n,nup,d,x)
% Clenshaw's algorithm to the Gram orthonormal polynomial.

% S = CLENSHAW(N,NUP,D,X) returns the summation:
%
%   D(1)*G_0(X) + D(2)*G_1(X) + ... + D(n+1)*G_n(X),
%
% where G obeys the relation: G_k - a_k*G_{k-1} - b_k*G_{k-2} = 0, b_k~=0,
% k=2,...,N.
%
% G_k is the orthonormal Gram polynomial of degree k and parameter NUP.
%
%
% See (Deuflhard, 1976) and (Clenshaw, 1955).

% computes alpha_0,...,alpha_{n-1}:
k=1:n;
alp = sqrt(nup^2*(k.^2-.25)./(k.^2.*(nup^2-k.^2)));

nt = length(x);
G0 = ones(1,nt);

if ( n > 1 )
    % initialise:
    u(n,1:nt) = d(n+1);
    u(n-1,:) =  2*alp(n)*x.*u(n,:) + d(n);

    for k=n-1:-1:2
        u(k-1,:) = 2*alp(k)*x.*u(k,:) - alp(k+1)/alp(k) *u(k+1,:) + d(k);
    end

    G1 = 2*alp(1)*x;
```

```
    s = (d(1) - alp(2)/alp(1) * u(2,:)).*G0 + u(1,:).*G1;

elseif ( n > 0 ) % n=1

    G1 = 2*alp(1)*x;
    s = d(1)*G0 + d(2).*G1;

else % n=0
    s = d(1)*G0;

end

end
```

5.3.2 gram.m

```
function [x,w] = gram(m,nup)
%%  GRAM    Nodes and weights of Gaussian type quadrature for sums.
%
%   [X] = GRAM(M, NUP) returns the zeros X of the orthonormal Gram
%   polynomial of degree M with parameter NUP.
%
%   [X, W] = GRAM(M, NUP) returns both X and a row vector W of
%   weights for the Gaussian type quadrature.
%
%   Calls REC.
%
%   Copyright 2021 by Dimitar K. Dimitrov and Lourenco L. Peixoto
%   All rights reserved.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   GRAM implemented by Lourenco L. Peixoto, 2021 - see [1-3].
%
%   Basic references:
%
%       [1] I. Area, D. K. Dimitrov, E. Godoy, and V. Paschoa, "Approximate
%   calculation of sums I: Bounds for the zeros of Gram polynomials", SIAM
%   J. Numer. Anal., 2014.
%
%       [2] I. Area, D. K. Dimitrov, E. Godoy, and V. Paschoa, "Approximate
%   calculation of sums II: Gaussian type quadrature", SIAM J. Numer.
%   Anal., 2016.
%
%       [3] D. K. Dimitrov and L. L. Peixoto, "An efficient algorithm for
%   the classical least squares approximation", SIAM J. Sci. Comput., 2020.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Check inputs:
if ( m > floor(2.5*sqrt(nup)) )
    warning('gram: M = %i. M must be less than floor(2.5*sqrt(NUP)).', m)
end

% Deal with trivial cases:
if (m<=3)
    if (m<1)
        error('gram:inputs:m', 'First input should be a positive integer.');
    end
    if (m==1)
        x = 0;
        w = 1;
        return
    elseif (m==2)
        x = sqrt((nup^2-1)/(3*nup^2));
        x = [-x x]';
        w = [.5 .5];
        return
    else % m=3
        x = sqrt((3*nup^2-7)/(5*nup^2));
        x = [-x 0 x]';
        w = 4/3*(nup^2-4)/(3*nup^2-7);
        w = [.5*(1-w) w .5*(1-w)];
        return
    end
end



%% Compute the zeros with WDDK method

[x1] = transpose(forster_petras(m)); % initial approximations

m_odd = (mod(m,2));
if m_odd % m is odd number
    x1(floor(m/2)+1) = 0;
end

dt = inf;
count = 0;
maxit = 50;
denom = zeros(1,m);

% alpha_0,...,alpha_{M-1}:
k=1:m;
alpha = sqrt(nup^2*(k.^2-.25)./(k.^2.*(nup^2-k.^2)));
```

```
% compute the leading coefficient of the polynomial of degree M:
coef_lead = 2^m*prod(alpha);

% initialise:
x = x1;
lim = 1-1/nup; % interval limit

while (norm(dt,inf) > eps && count <= maxit)
    g = rec(m,nup,x,alpha);  % evaluate the polynomial via recurrence formula

    % WDDK:
    for k = 1:m
        d = coef_lead;
        for j = 1:m
            if (k~=j)
                d = d*(x(k)-x(j)); % denominator
                if d==0 % underflow:
                    error('Underflow in WDDK.')
                end
            end
        end
        denom(k) = d;
    end
    dt = g./denom;
    x = x - dt;

    if (x(1) < -lim || x(end) > lim)
        error('Nodes are out of range (-1+1/NUP, 1-1/NUP).')
    end

    if (m_odd) % m is an odd number
        x(floor(m/2)+1) = 0;
    end

    count = count + 1;
end

if (count == maxit+1 && norm(dt,inf) > eps)
     warning('gram: %i iterations in WDDK method. ', maxit)
end
```



```
%% Compute the weights

% only non negative nodes:
z = x(floor(m/2)+1:m);
```

```
cte = sqrt(3*nup^2/(nup^2-1));
g1 = ones(1,length(z)); g = cte*z;

w = g1.^2 + g.^2;
for k=2:m-1
    % initialise for polynomial:
    g2 = g1; g1 = g;
    % polynomial of degree k:
    g = 2*alpha(k)*z.*g1 - alpha(k)/alpha(k-1).*g2;
    w = w + g.^2;
end
w = 1./w;

% Reflect for symmetric values:
if mod(m,2)
    w = [flip(w), w(2:end)];
else
    w = [flip(w), w];
end

x = transpose(x);

end
```

```
%% Compute initial approximations for the zeros of Gram pol. of degree M
function [x] = forster_petras(m)
% X = FORSTER_PETRAS(M) returns the approximations in Forster and Petras
% (1993, Theorem 1) for the zeros of Legendre polynomial.
if m==1
    x=0;
    s=1;
else
    if ( mod(m,2) )
        s = 1; % odd m
    else
        s = 0; % even m
    end
    k = ((m+s)/2:-1:1).';

    theta = (k-.25)/(m+.5)*pi;
    x0 = cos(theta);
    cos2 = x0.*x0;
    % Sharp initial guess (Forster and Petras, 1993):
    x = cos( theta + .25/(2*(m+.5)^2)*(1-(6+.25*(9-2*cos2))./(12*(m+.5)^2*...
    (1-cos2))).*cot(theta) );
end
```

```
x = [-x(end:-1:1+s) ; x]; % reflect negative zeros

end
```

5.3.3 rec.m

```
function [g,MG] = rec(n,nup,x)
%%  REC Recurrence relation for orthonormal Gram polynomials
%
%   [G] = REC(N,NUP,X) returns the orthonormal Gram polynomial of degree N
%   and parameter NUP evaluated at vector X.
%
%   [G,MG] = REC(N,NUP,X) also returns the (N+1)-by-size(X) matrix MG with
%   the Gram polynomials of degrees 0,1,...,N evaluated on X. Its first row
%   contains the polynomial of degree 0, the second row contains the
%   polynomial of degree 1 etc. Its first column are the polynomials on
%   X_1, the second column are the polynomials on X_2 etc.
%
%   REC(N,NUP,X,ALPHA) allows the user furnishes the vector with the
%   parameters of the recurrence relation:
%       ALPHA = alpha_0,...,alpha_{n-1}
%
%

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   REC implemented by Lourenco L. Peixoto, 2021 - see [1-3].
%
%   Basic references:
%
%       [1] I. Area, D. K. Dimitrov, E. Godoy, and V. Paschoa, "Approximate
%   calculation of sums I: Bounds for the zeros of Gram polynomials", SIAM
%   J. Numer. Anal., 2014.
%
%       [2] I. Area, D. K. Dimitrov, E. Godoy, and V. Paschoa, "Approximate
%   calculation of sums II: Gaussian type quadrature", SIAM J. Numer.
%   Anal., 2016.
%
%       [3] D. K. Dimitrov and L. L. Peixoto, "An efficient algorithm for
%   the classical least squares approximation", SIAM J. Sci. Comput., 2020.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


comp = length(x);

if n==0
    g = ones(1,comp);
    if (nargout == 2)
```

```
        MG = g;
    end
elseif n==1
    cte = sqrt(3*nup^2/(nup^2-1));
    g = cte*x;
    if (nargout == 2)
        MG = [ones(1,comp); g];
    end
else % n>=2
    cte = sqrt(3*nup^2/(nup^2-1));
    g1 = ones(1,comp); g = cte*x;

    if ( nargin == 4 )
        alpha = varargin{1};
    else
        k=1:n;
        alpha = sqrt(nup^2*(k.^2-.25)./(k.^2.*(nup^2-k.^2)));
    end

    if (nargout == 1)
        for k=2:n
            g2 = g1; g1 = g; % initialise for polynomial
            g = 2*alpha(k)*x.*g1 - alpha(k)/alpha(k-1).*g2; % polynomial
        end
    else % stores all the polynomials of degree 0,...,n:
        MG = ones(n+1,comp); % preallocating
        MG(2,:) = g; % polynomial of degree 1
        for k=2:n
            % initialise for polynomial:
            g2 = g1; g1 = g;
            % polynomial of degree k:
            g = 2*alpha(k)*x.*g1 - alpha(k)/alpha(k-1).*g2;
            % polynomial of degree k:
            MG(k+1,:) = g;
        end
    end
end

end
```

6 Comparações

Neste capítulo iremos confrontar a eficiência do nosso algoritmo que chamaremos de NEW. Vamos compará-lo ao STANDARD com relação à precisão dos seus resultados e complexidades computacionais. Iremos usar os resultados do nosso algoritmo em ambas bases: monomial e de Gram. Para avaliações de p_n, usaremos o tradicional método de Horner (quando a base for monomial) e o algoritmo de Clenshaw (quando a mesma for de Gram).

6.1 Dados polinomiais

Começamos comparando quando os dados são obtidos por meio de um polinômio de grau 3,

$$f(x_j) = x_j^3 - \pi x_j^2 - 1, \qquad j = 1, 2, \ldots, N,$$

quando $N = 10^5$. Neste caso, a expansão de f na base de Gram é

$$f(x) = a_0\, G_0(x; 10^5) + a_1\, G_1(x; 10^5) + a_2\, G_2(x; 10^5) + a_3\, G_3(x; 10^5),$$

onde

$$\begin{aligned} a_0 &= -\left(1 + \frac{3333333333\,\pi}{10000000000}\right), & a_1 &= \frac{29999999993\,\sqrt{3333333333}}{5000000000000000}, \\ a_2 &= -\frac{\pi\,\sqrt{13888888881944444445}}{12500000000}, & a_3 &= \frac{57\,\sqrt{109923932484720573417095 37}}{1250000000000000}. \end{aligned}$$

Vejamos os resultados na Figura 6.1. O nosso algoritmo NEW sempre calcula os coeficientes na precisão da máquina num tempo muito mais rápido do que o do STANDARD. Além disto, o algoritmo STANDARD possui um erro crescente a partir de $n = 7$ até $n = 31$ quando ele detecta uma deficiência no posto da matriz do tipo Vandermonde. A partir daí ele mantém a precisão do método em torno de 10^{-5}. Em todo caso, os resultados de STANDARD são muito aquém de NEW.

Já na Figura 6.2 nós comparamos os algoritmos quando N e n aumentam simultaneamente com $n = \lfloor\sqrt{N}/10\rfloor$. NEW é sempre mais preciso. Além disto, é mais rápido em ambas as bases para $N > 10^4$. STANDARD apresenta deficiências de posto para $N \geq 10^5$. Na Figura 6.3 realizamos comparações quando n é mantido fixo e N aumenta. NEW calcula os coeficientes com complexidade de operações $\mathcal{O}(1)$ na precisão da máquina. A complexidade de STANDARD é $\mathcal{O}(N)$ devido ao fator Q da decomposição QR.

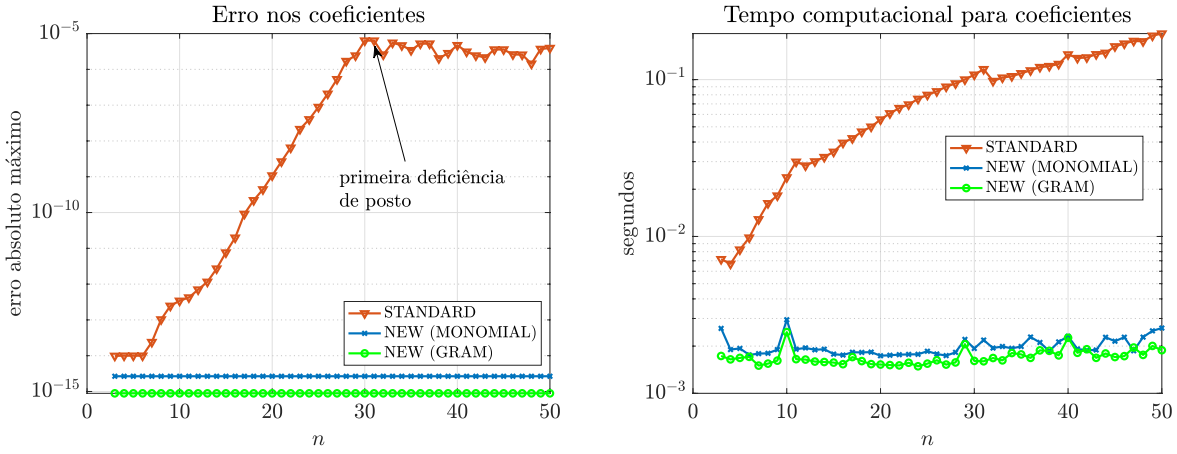

Figura 6.1: Erro absoluto máximo nos coeficientes e tempo computacional quando $f(x_j) = x_j^3 - \pi x_j^2 - 1$ com $N = 10^5$.

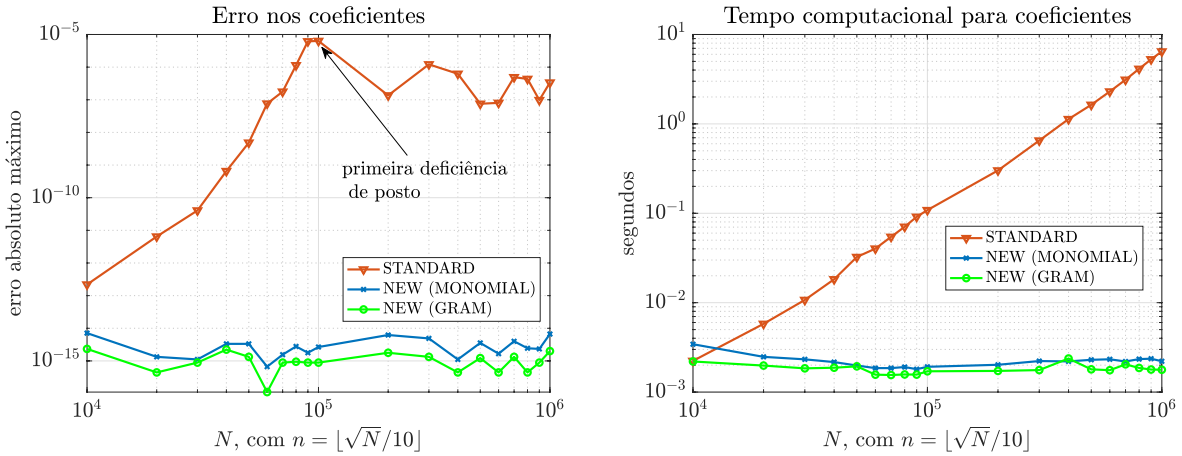

Figura 6.2: Erro absoluto máximo nos coeficientes e tempo computacional quando $f(x_j) = x_j^3 - \pi x_j^2 - 1$ com $n = \lfloor \sqrt{N}/10 \rfloor$.

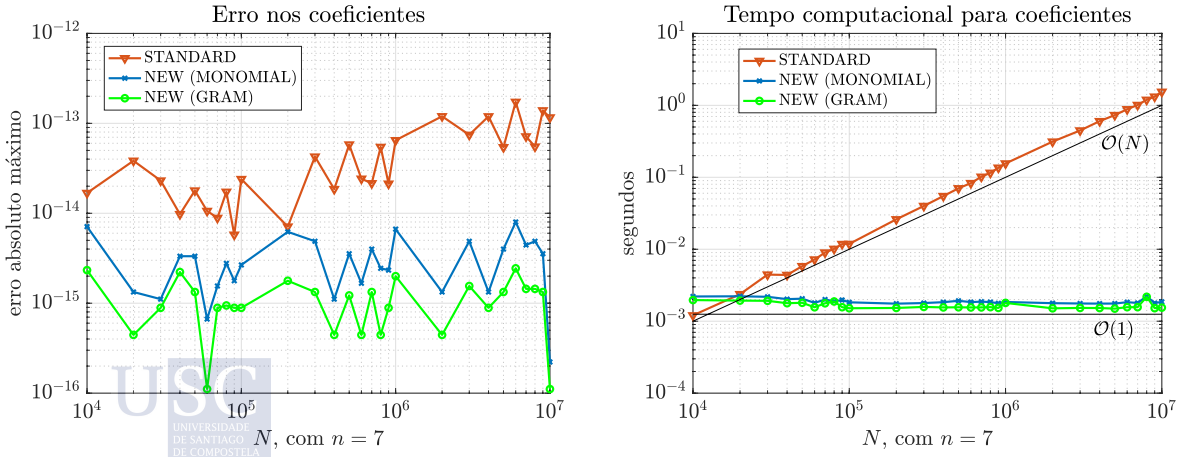

Figura 6.3: Erro absoluto máximo nos coeficientes e tempo computacional quando $f(x_j) = x_j^3 - \pi x_j^2 - 1$ e $n = 7$.

6.2 Dados suaves

Nesta seção comparamos os algoritmos para dados de funções suaves, isto é, de classe C^∞, e que podem ser avaliadas diretamente sobre os pontos da quadratura. Como neste caso não dispomos dos coeficientes analíticos de p_n, comparamos a eficiência dos algoritmos ao utilizar a soma residual de quadrados em (1.6),

$$D(p_n) := \sum_{j=1}^{N} \left(f(x_j) - p_n(x_j; f)\right)^2 .$$

Fundamentalmente, o método dos mínimos quadrados busca justamente minimizar a soma acima. Utilizamos

$$f(x_j) = \operatorname{sen}(15x_j), \quad j = 1, \ldots, N,$$

com $N = 5 \cdot 10^4$. Vejamos as comparações na Figura 6.4. Todos métodos tem essencial-

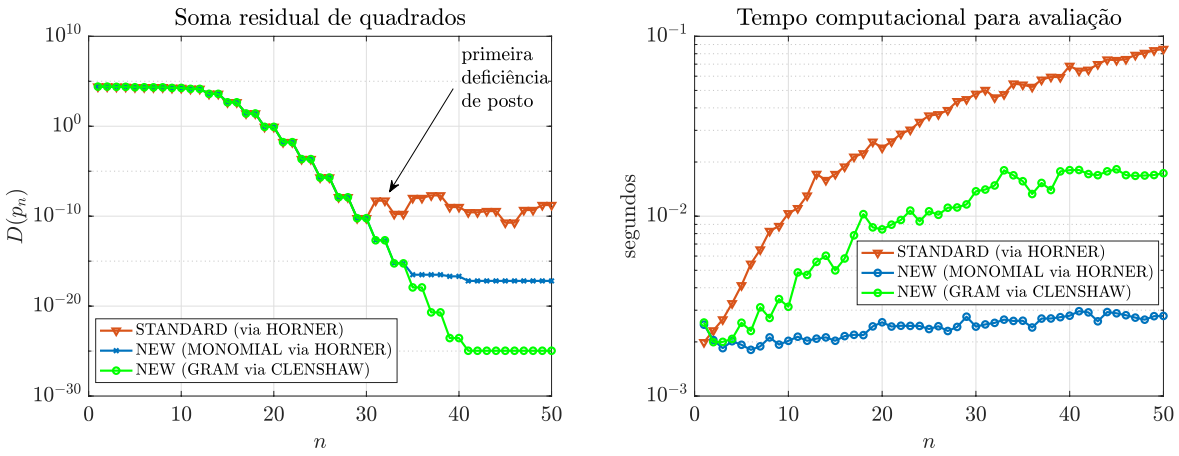

Figura 6.4: Soma residual de quadrados para a função $f(x_j) = \operatorname{sen}(15x_j)$ com $N = 5 \cdot 10^4$ e tempo computacional para avaliação.

mente a mesma precisão para todos valores de n até 31 quando STANDARD começa a apresentar deficiência. Verificamos um pequeno erro de arredondamento nos coeficientes na base monomial para $35 \leq n \leq 40$ mas para $n \geq 41$ ele torna-se desprezível. Em todo caso, NEW é mais preciso para $n \geq 32$ e alcança um D em torno de 10^{-25} na base de Gram. Nosso algoritmo é mais rápido em ambas bases para avaliar p_n para $n > 1$. Também notamos que o algoritmo de Horner se mostrou mais rápido do que o de Clenshaw. O tempo computacional é o tempo gasto para obter os coeficientes de p_n acrescido do tempo gasto para obter as avaliações de p_n em cada um dos N pontos x_j.

6.3 Dados perturbados

Este é o caso mais interessante, pois em geral desconhecemos a função geradora dos dados e aqui nosso algoritmo deverá recuperar os dados de f sobre os pontos da quadratura. Tomemos

$$f(x_j) = e^{x_j^3} + 10^{-7}\delta_N(j), \quad j = 1, \ldots, N,$$

para $N = 3 \cdot 10^4$. A perturbação relativa varia entre 10^{-14} e 10^{-7}. Vejamos os resultados na Figura 6.5. Todos métodos têm essencialmente a mesma precisão. STANDARD apresenta

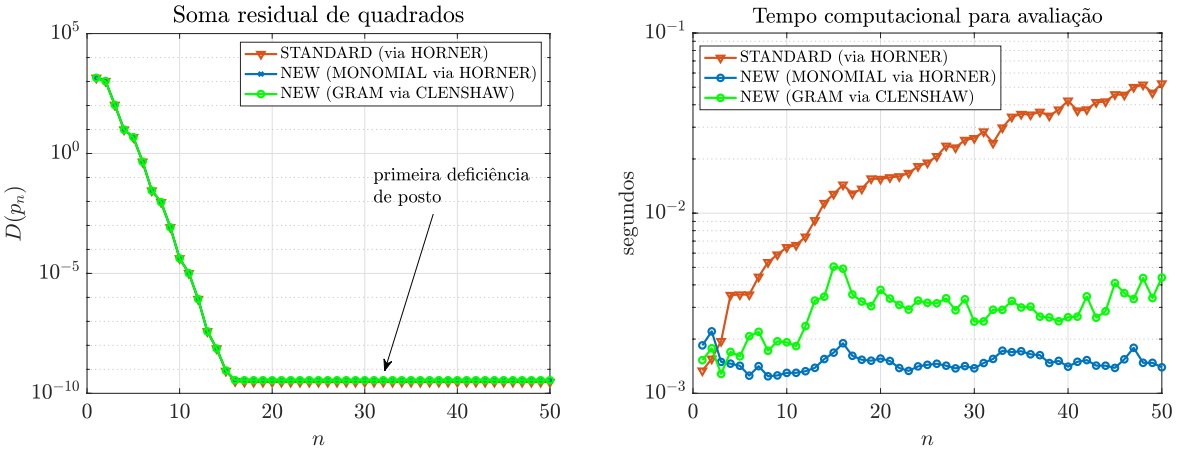

Figura 6.5: Soma residual de quadrados para a função $f(x_j) = e^{x_j^3} + 10^{-7}\delta_N(j)$ com $N = 3 \cdot 10^4$ e tempo computacional.

deficiência de posto a partir de $n = 32$ sem prejuízos na sua precisão. Contudo, nosso algoritmo é mais rápido em ambas bases para $n > 2$. Veja que tomamos um valor modesto para N. As diferenças com relação ao tempo computacional podem ser muito maiores quando tomamos $N = 10^6$ ou 10^7, por exemplo.

O algoritmo de Clenshaw, embora mais elegante e, às vezes, ligeiramente mais preciso, pode requerer muito espaço de memória para valores grandes de n e N (porque ele utiliza N vetores de n elementos). Caso este algoritmo exceda os limites da máquina, indicamos obter a base monomial para usar o método de Horner que pode ser utilizado com o comando `polyval` do MATLAB.

As Tabelas 6.1 e 6.2 comparam D e o tempo computacional para diversos valores de n e N com os dados perturbados

$$f(x_j) = \cos(20x_j) + 10^{-8}\delta_N(j), \quad j = 1, \ldots, N.$$

A perturbação absoluta varia, aproximadamente, entre $10^{-9}N^{-1}$ e 10^{-8}, para cada N. Os símbolos "ST" (STANDARD) e "NEW" indicam qual dos algoritmos foi mais eficiente. O símbolo "·" mostra que ambos foram equivalentes. Além disto, o símbolo "NEW*" mostra que NEW resolve o problema e STANDARD estoura a capacidade de memória do MATLAB impossibilitando-o de retornar resultado.

	$N = 10^3$	$N = 10^4$	$N = 10^5$	$N = 10^6$	$N = 10^7$	$N = 10^8$
$n = 5$	·	·	·	·	·	·
$n = 10$	·	·	·	·	·	NEW*
$n = 20$	·	·	·	·	·	NEW*
$n = 30$	ST	·	·	NEW	·	NEW*
$n = 40$	ST	ST	NEW	NEW	NEW	NEW*
$n = 50$	ST	ST	NEW	NEW	NEW	NEW*

Tabela 6.1: Comparação de D.

	$N = 10^3$	$N = 10^4$	$N = 10^5$	$N = 10^6$	$N = 10^7$	$N = 10^8$
$n = 5$	ST	ST	NEW	NEW	NEW	NEW
$n = 10$	ST	·	NEW	NEW	NEW	NEW*
$n = 20$	ST	NEW	NEW	NEW	NEW	NEW*
$n = 30$	ST	NEW	NEW	NEW	NEW	NEW*
$n = 40$	ST	NEW	NEW	NEW	NEW	NEW*
$n = 50$	ST	NEW	NEW	NEW	NEW	NEW*

Tabela 6.2: Comparação do tempo de avaliação.

6.4 Resolvendo o PCMQ com quantidade colossal de dados

Agora resolvemos o problema para uma quantidade enorme de dados e exibiremos o gráfico de p_n. Nos seguintes casos, o algoritmo STANDARD não pode retornar resultado algum devido ao excesso de memória requerida. Tomamos dados suaves

$$f(x_j) = e^{x_j} \operatorname{sen} 12\, x_j, \quad j = 1, \ldots, N,$$

e muito perturbados

$$f(x_j) = e^{x_j} \operatorname{sen} 12 x_j + \delta_N(j), \quad j = 1, \ldots, N,$$

para $N = 10^8$. A perturbação absoluta varia, aproximadamente, entre 10^{-9} e 8. Em ambos casos calculamos $p_{12}(\cdot\,; f)$ pelo método de Horner. Confira a Figura 6.6 com os ajustes do polinômio p_{12} aos dados suaves (à esq.) e perturbados (à dir.). Nota-se que a perturbação sobre os dados é muito forte.

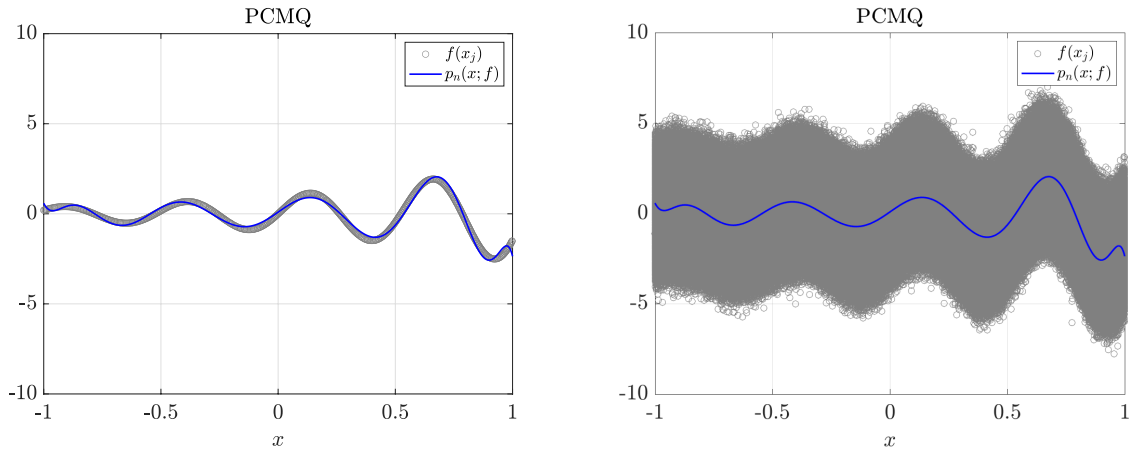

Figura 6.6: Soluções para PCMQ com grau 12 com $N = 10^8$ para dados suaves e perturbados.

Agora vejamos a Figura 6.7 para a aproximação por $p_{15}(\cdot\,; f)$ para os dados suaves (à esq.)

$$f(x_j) = e^{x_j^2 \operatorname{sen} 11 x_j}, \quad j = 1, \ldots, N,$$

e também com forte perturbação (à dir.),

$$f(x_j) = e^{x_j^2 \operatorname{sen} 11 x_j} + \delta_N(j), \quad j = 1, \ldots, N,$$

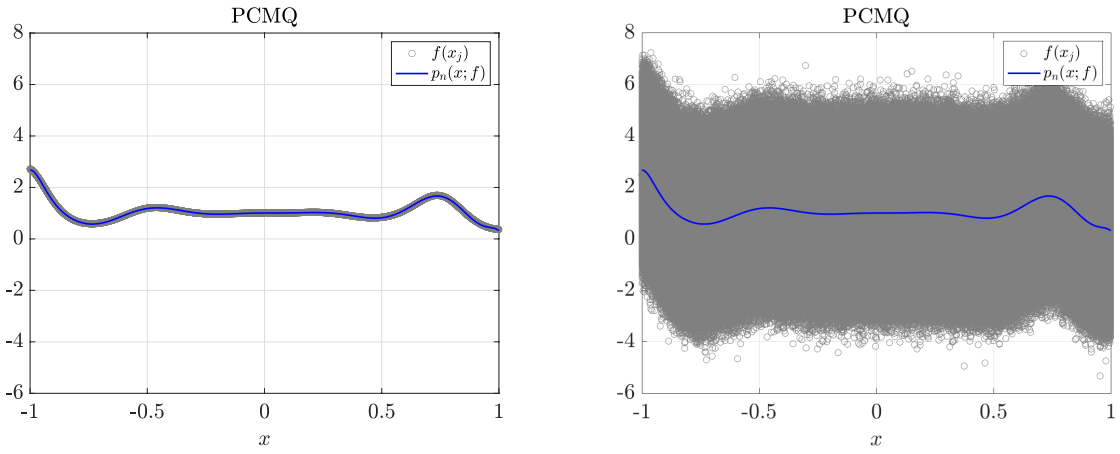

Figura 6.7: Soluções para PCMQ com grau 15 com $N = 10^8$ para dados suaves e perturbados.

com $N = 10^8$. É notável o ajuste do polinômio de grau 15 aos dados suaves deste caso.

Ressaltamos que o método das equações normais e da decomposição em valores singulares apresentados no Capítulo 4 também foram incapazes de retornar resultados devido ao excesso de memória requerida para os exemplos desta seção.

7 Conclusões e futuras pesquisas

Nosso trabalho obteve com êxito um novo algoritmo eficiente para resolver o problema clássico dos mínimos quadrados com grande quantidade de dados. Em geral, o novo algoritmo é mais estável e mais rápido do que os demais conhecidos na atualidade. Buscamos, sobretudo, um algoritmo em dupla precisão que fosse facilmente vetorizado para ser implementado em diversas linguagens de programação.

Todos os testes numéricos foram realizados em MATLAB R2017b (Student License) com um MacBook Pro 2012 2.3 GHz Intel Core i7.

7.1 Problemas em aberto

Como o desenvolvimento e a pesquisa sobre os polinômios de Gram, deixamos as seguintes Conjecturas 2.8, 2.9 e 3.7.

7.2 Trabalhos futuros

Atualmente estamos buscando aplicar o PCMQ para resolver um problema relacionado à recuperação dos níveis de energia de uma função partição da mecânica estatística. Em breve pretendemos trabalhar na criação de um algoritmo análogo para o problema dos mínimos quadrados ponderados [22, §4.4] utilizando as medidas discretas de Hahn e Krawtchouk. Também desejamos trabalhar com algoritmos para aproximações em espaços $L^2[a, b]$, ver [5, p. 28 probl. 3].

Referências

[1] TREFETHEN, L. N. The definition of numerical analysis. *SIAM News*, v. 25, 1992.

[2] AREA, I. et al. Approximate calculation of sums I: Bounds for the zeros of Gram polynomials. *SIAM Journal on Numerical Analysis*, v. 52, n. 4, p. 1867–1886, 2014.

[3] AREA, I. et al. Approximate calculation of sums II: Gaussian type quadrature. *SIAM Journal on Numerical Analysis*, v. 54, n. 4, p. 2210–2227, 2016.

[4] DIMITROV, D. K.; PEIXOTO, L. L. An efficient algorithm for the classical least squares approximation. *SIAM Journal on Scientific Computing*, v. 42, n. 5, p. A3233–A3249, 2020.

[5] DEUTSCH, F. *Best Approximation in Inner Product Spaces*. New York: Springer-Verlag, 2001.

[6] GOLUB, G. H.; LOAN, C. F. van. *Matrix Computations*. 3. ed. London: The Johns Hopkins University Press, 1996.

[7] SZEGŐ, G. *Orthogonal Polynomials*. 4th. ed. [S.l.]: American Mathematical Society, 1975.

[8] CHIHARA, T. S. *An Introduction to Orthogonal Polynomials*. New York: Gordon and Breach, 1978.

[9] CLENSHAW, C. W. A note on the summation of Chebyshev series. *Math. Comp.*, v. 9, p. 118–120, 1955.

[10] KOEKOEK, R.; LESKY, P. A.; SWARTTOUW, R. F. *Hypergeometric Orthogonal Polynomials and Their q-Analogues*. Berlin: Springer Berlin Heidelberg, 2010.

[11] BAIK, J. et al. *Discrete orthogonal polynomials: asymptotics and applications*. Princeton: Princeton University Press, 2007.

[12] ANDREWS, G. E.; ASKEY, R.; ROY, R. *Special Functions*. Cambridge: Cambridge University Press, 1999.

[13] CHEBYSHEV, P. L. Sur l'interpolation. *Zapiski Akademii Nauk (Oeuvres, Vol. 1, pp. 539-560)*, v. 4, 1864.

[14] CHEBYSHEV, P. L. Sur les fractions continues. *Journal de Mathématiques (Oeuvres, Vol. 1, pp. 201-230)*, v. 3, p. 289–323, 1858.

[15] CHEBYSHEV, P. L. Sur l'interpolation par la méthode des moindres carrés. *Mémoires de l'Académie Impériale des Sciences de St. Pétersbourg (Oeuvres, Vol. 1, pp. 471-498)*, v. 1, p. 1–24, 1859.

[16] HILDEBRAND, F. B. *Introduction to Numerical Analysis*. [S.l.]: Dover Publ., 1987.

[17] ISAACSON, E.; KELLER, H. B. *Analysis of Numerical Methods*. New York: Dover Publications, 1994.

[18] BJÖRCK, Å.; DAHLQUIST, G. *Numerical Methods*. Mineola: Dover Publications, 2003.

[19] BARNARD, R. W. et al. Gram polynomials and the Kummer function. *J. Approx. Theory*, v. 9, p. 128–143, 1998.

[20] RIVLIN, T. J. *An Introduction to the Approximation of Functions*. New York: Dover Publications, 1981. (Blaisdell book in numerical analysis and computer science).

[21] RALSTON, A.; RABINOWITZ, P. *A First Course in Numerical Analysis*. New York: McGraw-Hill, 1978.

[22] BJÖRCK, Å. *Numerical Methods for Least Squares Problems*. Philadelphia: SIAM, 1996.

[23] GRAM, J. P. Ueber die entwickelung reeller functionen in reihen mittelst der methode der kleinsten quadrate. *Journal für die reine und angewandte Mathematik*, v. 1883, n. 94, p. 41–73, 1883.

[24] ZAREMBA, S. K. Some properties of polynomials orthogonal over the set $< 1, 2, \ldots, n >$. *Annali Matem.*, v. 105, p. 333–345, 1975.

[25] DIEUDONNÉ, J. *Treatise on Analysis*. New York: Academic Press, 1976. II.

[26] BENEDETTO, J. J. *Harmonic Analysis and Applications*. Boca Raton: CRC Press, 1996.

[27] GAUTSCHI, W. Construction of Gauss-Christoffel quadrature formulas. *Math. Comp.*, v. 22, n. 102, p. 251–270, 1968.

[28] GAUSS, C. F. Methodus nova integralium valores per approximationem inveniendi. *Commentationes Societatis Regiae Scientarium Gottingensis Recentiores*, p. 1–40, 1814.

[29] DAVIS, P. J.; RABINOWITZ, P. *Methods of Numerical Integration*. Orlando: Academic Press, 1984.

[30] CHRISTOFFEL, E. B. Sur une classe particulière de fonctions entières et de fractions continues. *Annali di Matematica*, v. 8, p. 1–10, 1877.

[31] KRYLOV, V. I. *Approximate Calculation of Integrals*. New York: Macmillan Co., 1962.

[32] GAUTSCHI, W. *Orthogonal polynomials: computation and approximation*. New York: Oxford university press, 2004.

[33] PEIXOTO, L. L. *Quadratura de Gauss iterativa com base nos polinômios ortogonais clássicos*. Dissertação (Mestrado) — CEFET-MG, Belo Horizonte, Dez. 2008.

[34] ABRAMOWITZ, M.; STEGUN, I. A. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. [S.l.]: Dover Publications, 1964.

[35] OLVER, F. W. J. et al. *NIST Handbook of Mathematical Functions*. New York: Cambridge University Press, 2010.

[36] WILF, H. S. *Mathematics for the Physical Sciences*. New York: Wiley, 1962.

[37] GOLUB, G. H.; WELSCH, J. H. Calculation of Gauss quadrature rules. *Mathematics of Computation*, v. 23, n. 106, p. 221–230+s1–s10, 1969.

[38] GROSSE, E. `gaussq.f` *GO quadrature library*. 1983. `http://netlib.org/go/gaussq.f`.

[39] HALE, N.; TOWNSEND, A. Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights. *SIAM Journal on Scientific Computing*, v. 35, n. 2, p. A652 – A671, 2013.

[40] GLASER, A.; ROKHLIN, V.; XIANGTAO, L. A fast algorithm for the calculation of the roots of special functions. *SIAM Journal on Scientific Computing*, v. 29, n. 4, p. 1420 – 1438, 2007.

[41] YAKIMIW, E. Accurate computation of weights in classical Gauss-Christoffel quadrature rules. *Journal of Computational Physics*, v. 129, n. 2, p. 406 – 430, 1996.

[42] SWARZTRAUBER, P. N. On computing the points and weights for Gauss–Legendre quadrature. *SIAM Journal on Scientific Computing*, v. 24, n. 3, p. 945–954, 2003.

[43] FÖRSTER, K. J.; PETRAS, K. Inequalities for the zeros of ultraspherical polynomials and Bessel functions. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, v. 73, n. 9, p. 232–236, 1993.

[44] DRISCOLL, T. A.; HALE, N.; TREFETHEN, L. N. (Ed.). *Chebfun Guide*. Oxford: Pafnuty Publications, 2014.

[45] BOGAERT, I. Iteration-free computation of Gauss-Legendre quadrature nodes and weights. *SIAM Journal on Scientific Computing*, v. 36, n. 3, p. A1008–A1026, 2014.

[46] LETHER, F. G. On the construction of Gauss-Legendre quadrature rules. *Journal of Computational and Applied Mathematics*, v. 4, n. 1, p. 47–52, 1978.

[47] SMOLYAK, S. A. *On the optimal recovery of functions and functionals of them*. Dissertação (Mestrado) — Moscow State University, Moscow, 1965.

[48] DEMANET, L.; TOWNSEND, A. Stable extrapolation of analytic functions. *Found. Comput. Math.*, v. 19, n. 2, p. 297–331, 2019.

[49] TREFETHEN, L. N. *Approximation Theory and Approximation Practice*. Oxford: Society for Industrial and Applied Mathematics, 2019.

[50] BRASS, H. Error estimates for least squares approximation by polynomials. *Journal of Approximation Theory*, v. 41, n. 4, p. 345–349, 1984.