PICME

OBMEP - CNPq

Iniciação Científica
Interpolação Polinomial

Gustavo Rodrigues da Silva

Última Atualização: 12 de março de 2021

Sumário

Capítulo 1

Dados Gerais

Projeto de Iniciação Científica.

Período: fevereiro de 2020 até fevereiro de 2021

Aluno: Gustavo Rodrigues da Silva
E-mail: gustavor10silva@gmail.com
Instituição: Universidade Federal do Paraná - UFPR
Curso: Bacharelado em Matemática
Ano de início no PICME: 2019
CPF: 11660525918
Data de nascimento: 01/07/2000
Link do Lattes: http://lattes.cnpq.br/2868517316417618

Orientador: Elías Alfredo Gudiño Rojas
E-mail: egudino@ufpr.br
Instituição: Universidade Federal do Paraná - UFPR
CPF: 01415575916
Data de nascimento: 02/07/1980
Link do Lattes: http://lattes.cnpq.br/4595354952800890

Título do projeto: Estudo sobre interpolação e integração numérica

Palavras-chave: Polinômios de Lagrange, Newton-Cottes, Splines

Capítulo 2

Introdução

Este material possui os relatos dos estudos realizados na Iniciação Científica orientada pelo Prof° Dr. Elías Alfredo Gudiño Rojas, durante o período de fevereiro de 2020 até fevereiro de 2021, com o tema inicial de Interpolação Polinomial, que posteriormente estendeu-se sobre outros tópicos relacionados.

No início da Iniciação Científica, foram trabalhados conceitos iniciais acerca da interpolação polinomial, como a existência e a unicidade do polinômio interpolador e a obtenção desse polinômio por meio da resolução de um sistema linear. Em seguida, foram estudados métodos mais eficazes para fazer a interpolação, como o Método de Lagrange e a interpolação de Newton - com as diferenças divididas, dada também uma atenção especial ao caso onde os nós da interpolação são igualmente espaçados.

Depois disso, foi estudado o exemplo de Runge, que mostra de uma maneira interessante que o aumento de nós na interpolação nem sempre aumenta a precisão da aproximação e sugere um estudo mais detalhado sobre o erro da interpolação. O erro foi justamente o próximo assunto estudado na Iniciação Científica.

Posteriormente, iniciou-se um trabalho com a integração do polinômio interpolador, o que é usado para fazer a aproximação de integrais. Isso foi feito estudando as fórmulas de Newton-Cotes e as fórmulas de quadratura de Gauss. Todos os métodos estudados foram implementados usando o MATLAB ou a linguagem Julia.

Como cada fórmula de quadratura de Gauss trabalha com a integração em um intervalo já delimitado, foi estudada a mudança de variável, para permitir a adequação de diferentes integrais nesses limites de integração pré-estabelecidos. Feito isso, iniciou-se o estudo das ordens de convergência dos métodos estudados.

Por fim, foram estudados os mesmos métodos, mas com o uso da subdivisão do intervalo de integração em intervalos menores, de mesmo tamanho, onde foi observado que isso melhora a aproximação das integrais sem aumentar o grau dos polinômios envolvidos, o que seria mais caro computacionalmente. O último tópico estudado foi a aproximação da ordem de convergência desses métodos com a subdivisão do intervalo de integração.

A seguir, este material apresenta os relatos detalhados mês a mês dos estudos realizados, onde cada mês corresponderá a um capítulo. Os meses que não estão presentes neste material são aqueles nos quais não foram feitos relatórios.

Capítulo 3

Abril de 2020

No mês de abril, começamos os estudos partindo dos conceitos básicos e definições por trás da interpolação polinomial. De início, o estudo foi voltado ao entendimento do que é uma interpolação, suas aplicações e uma visão geral sobre os conceitos básicos em análise de erros. Em seguida, foi estudada a interpolação polinomial feita por meio da resolução de um sistema linear. Visto que o sistema linear obtido no processo de interpolação envolve uma Matriz de Vandermonde, cujo determinante é sempre diferente de zero, provamos a existência e a unicidade de uma solução para o sistema linear envolvido. Consequentemente, provamos a existência e a unicidade do polinômio interpolador, o que forma a base do estudo da interpolação polinomial.

Feito isso, iniciou-se o estudo de formas mais eficazes e rápidas para fazer a interpolação polinomial. Primeiramente, estudamos a interpolação polinomial pelo Método de Lagrange, com o qual podemos fazer uma demonstração diferente para a existência e a unicidade do polinômio interpolador. Em seguida, estudamos a interpolação polinomial de Newton, utilizando as diferenças divididas. Além disso, foi estudada a aplicação da interpolação de Newton quando os nós da interpolação são igualmente espaçados.

Após o estudo desses três métodos de interpolação polinomial, o estudo voltou-se à análise do erro cometido nas estimativas. Foi estudada a forma de obtenção da expressão que estima o erro cometido e, com base nela, podemos encontrar um limitante superior para o erro. Dessa forma, dada uma interpolação polinomial, podemos garantir que o erro cometido em cada estimativa será sempre menor ou igual do que o valor dado pelo limitante superior.

Por fim, foi iniciado o estudo do Exemplo de Runge, que aborda a aproximação dos valores da função $f(x) = \dfrac{1}{1+25x^2}$ no intervalo [-1,1], utilizando, inicialmente, 11 nós igualmente espaçados e aumentando o número de nós a fim de observar o que acontece. Para tal, é necessário um conhecimento básico de programação em MATLAB, uma vez que os cálculos são muito extensos. Portanto, no atual momento, o estudo está voltado ao aprendizado necessário para fazer as interpolações por meio do computador.

Com a interrupção das atividades presenciais por conta do coronavírus, as atividades estão sendo recomendadas pelo meu Orientador por e-mail, meio pelo qual ele também esclarece minhas dúvidas, além de recomendar materiais, se necessário.

Capítulo 4

Maio de 2020

No início do mês de maio, os estudos estavam voltados ao comportamento dos polinômios interpoladores no Exemplo de Runge. Esse exemplo aborda a aproximação, por meio da interpolação polinomial, dos valores da função $f(x) = \dfrac{1}{1+25x^2}$ no intervalo $[-1, 1]$. O intuito do estudo desse exemplo é começar a interpolar com 11 nós igualmente espaçados e aumentar o número de nós para, em seguida, comparar os polinômios interpoladores e verificar o que acontece.

Tendo em vista que as interpolações do exemplo dado envolvem cálculos extensos e que o mesmo demanda plotagens dos gráficos, são necessários conhecimentos básicos de programação no MATLAB para o seu estudo. A conclusão dessa etapa inicial possibilitou o início da escrita das funções que efetuariam a interpolação polinomial por meio do Método de Lagrange. Logo em seguida, foram plotados os gráficos dos polinômios interpoladores da função $f(x) = \dfrac{1}{1+25x^2}$ no intervalo $[-1, 1]$. O resultado obtido foi o seguinte:

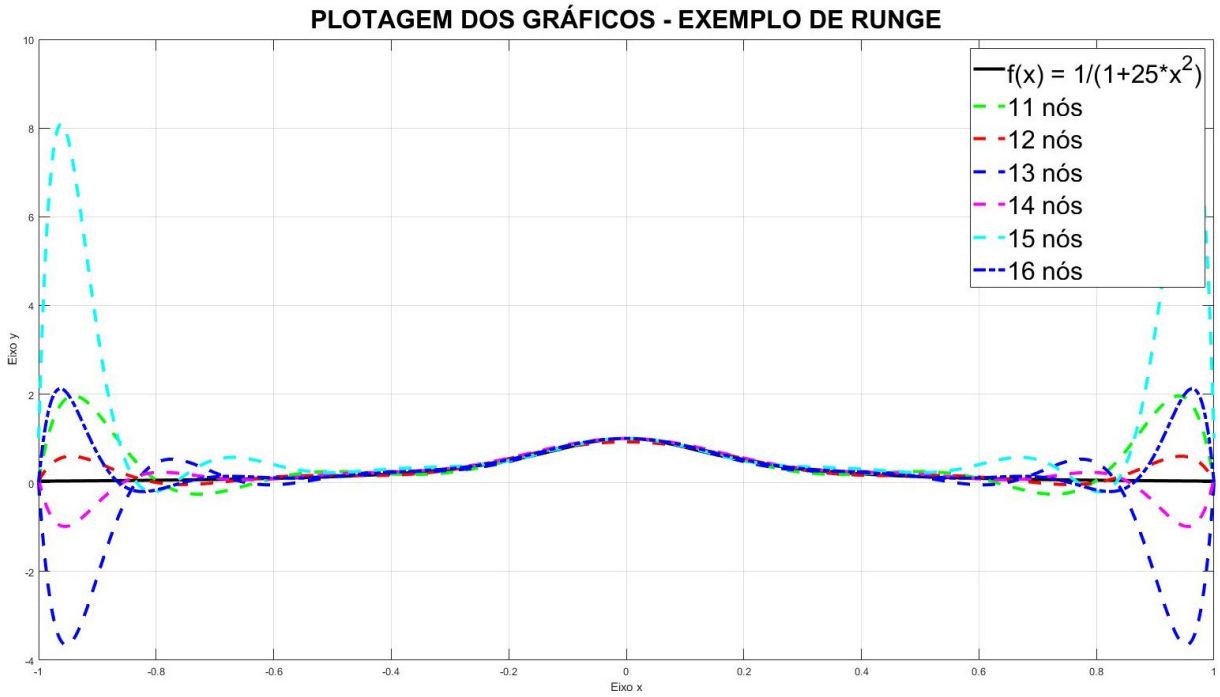

Figura 4.1: Comparação dos polinômios interpoladores no Exemplo de Runge.

Note que as comparações feitas mostram que, à medida que usamos mais nós para fazer a interpolação da função dada, a qualidade da interpolação piora, ou seja, o erro cometido na aproximação é maior. Isso ocorre por conta da expressão que define o erro cometido na interpolação, que será explicada a seguir.

Se $p_n(x)$ é o polinômio de grau n que interpola $f(x)$ com os nós $x_0, ..., x_n$, então o erro $E_n(x)$ cometido

na interpolação é dado por:

$$E_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(\zeta)}{(n+1)!} \cdot W_{n+1}(x),$$

onde $W_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$ e x é a abcissa do ponto que queremos conhecer. Note que $E_n(x)$ depende da derivada de ordem $n+1$ na função interpolada, e $n+1$ é o número de nós utilizados. A função do Exemplo de Runge é tal que, à medida que n aumenta, suas derivadas de ordem $n+1$ aumentam rapidamente e, com isso, o erro na interpolação também aumenta muito.

Concluído esse estudo do Exemplo de Runge e da relação do erro cometido com as derivadas de ordem $n+1$ da função interpolada, teve início o estudo da forma de obtenção dos nós ideais para uma interpolação. Para tal, é necessário o conhecimento sobre os Polinômios de Chebyshev. A obtenção desses polinômios é feita embasada na seguinte fórmula de recorrência:

$$T_{n+1}(x) = 2x \cdot T_n(x) - T_{n-1}(x), \text{ onde } T_n(x) = cos(n \cdot cos^{-1}(x)).$$

Se estamos interpolando uma função com $n+1$ nós, para que esses sejam os ideais, eles devem ser as raízes do Polinômio de Chebyshev de grau $n+1$. Particularmente, para uma interpolação com os nós $x_0, ..., x_n$, todos escolhidos no intervalo $[-1, 1]$, obtemos os nós ideais da seguinte forma:

$$x_k = cos\left(\frac{(2k+1)\pi}{2(n+1)}\right), \text{ com } k = 0, 1, ..., n.$$

Com isso, dado o número de nós, se queremos interpolar apenas no intervalo $[-1, 1]$, podemos encontrar facilmente os nós ideais para a interpolação. Por outro lado, para encontrarmos o mesmo número de nós ideais no intervalo $[a, b]$, prosseguimos da seguinte maneira:

$$x_k = \left(\frac{a+b}{2}\right) + \left(\frac{b-a}{2}\right) \cdot cos\left(\frac{(2k+1)\pi}{2(n+1)}\right), \text{ com } k = 0, 1, ..., n.$$

Sabendo disso, o próximo passo do estudo foi interpolar novamente a função $f(x) = \frac{1}{1+25x^2}$ no intervalo [-1,1], começando com 11 nós e aumentando até chegar em 16. Porém, dessa vez as interpolações foram feitas com os nós ideais. O resultado obtido foi o seguinte:

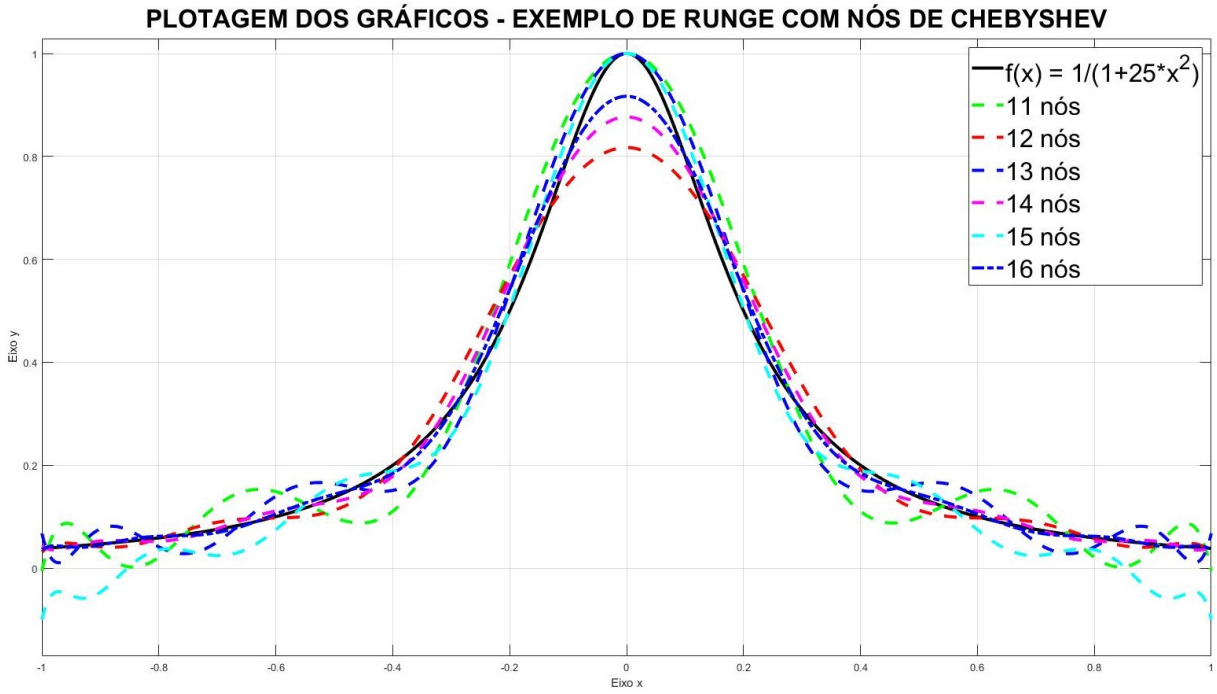

Figura 4.2: Polinômios interpoladores no Exemplo de Runge usando os nós ideais.

Note que, conforme a Figura 4.1, os polinômios interpoladores obtidos com os nós igualmente espaçados apresentam um enorme erro em relação à função interpolada, principalmente nas extremidades. Por outro lado, de acordo com a Figura 4.2, ao usar os nós ideais, o erro cometido foi muito menor e, neste caso, os polinômios interpoladores mostraram-se eficientes para aproximar os valores da função em questão.

Por último, o Elías Gudiño - meu orientador - e eu, tivemos uma conversa por chamada de vídeo para que eu sanasse algumas dúvidas com relação ao MATLAB e ao comportamento das derivadas da função do Exemplo de Runge.

Capítulo 5

Junho de 2020

5.1 Introdução

No mês de junho, foi dado início aos estudos da interpolação polinomial na aproximação de integrais, bem como o estudo das Fórmulas de Newton-Cottes mais utilizadas e o uso do erro para determinar o número de casas decimais de precisão que queremos na aproximação.

5.2 Integrando o Polinômio Interpolador

Em geral, quando uma função $f(x)$ é de difícil integração ou, se conhecemos alguns pares ordenados $(x_0, f(x_0)), ..., (x_n, f(x_n))$ mas não conhecemos a expressão de $f(x)$ em termos de x, uma ótima alternativa é encontrar um polinômio interpolador de $f(x)$ e integrar esse polinômio. A vantagem é que o cálculo da integral de um polinômio sempre é imediato e pode fornecer uma ótima aproximação para a integral da função original.

Sabendo disso, seja $p_n(x) = f(x_0) \cdot l_0(x) + ... + f(x_n) \cdot l_n(x)$ o polinômio de Lagrange de grau n, que interpola $f(x)$ com os nós $x_0, ..., x_n$. Seja $E_n(x)$ o erro cometido nessa interpolação. Tendo isso em vista, se queremos integrar $f(x)$ num intervalo $[a, b]$, temos:

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^b \left[p_n(x) + E_n(x)\right] dx \\ &= \sum_{i=0}^{n} a_i \cdot f(x_i) + \int_a^b E_n(x)dx, \text{ com } a_i = \int_a^b l_i(x)dx\end{aligned}$$

Com isso, conseguimos aproximar a integral de $f(x)$ em $[a, b]$ integrando seu polinômio interpolador nesse mesmo intervalo. Por outro lado, aproximar uma integral dessa maneira é muito trabalhoso. Note que, para cada integral a ser aproximada, precisaríamos obter todos os polinômios de Lagrange $l_i(x)$, $i = 0, ..., n$ e suas respectivas integrais. A fim de facilitar a aproximação das integrais, utilizamos as Fórmulas de Newton-Cottes, que foram estudadas na sequência.

5.3 Fórmulas de Newton-Cottes

As Fórmulas de Newton-Cottes aplicam-se quando os nós são igualmente espaçados. Se queremos aproximar a integral de $f(x)$ em $[a, b]$ e, para tal, interpolamos $f(x)$ com os nós $x_0, ..., x_n$, dizemos que a fórmula obtida nesse caso é do tipo **fechado** se $x_0 = a$ e $x_n = b$ e é do tipo **aberto** quando $x_0, x_n \in (a, b)$. Os estudos dessas fórmulas, de forma geral, foram voltados às fórmulas do tipo fechado.

Suponha que queremos aproximar a seguinte integral:

$$\int_a^b f(x)dx.$$

Foram estudadas a fundo três Fórmulas de Newton-Cottes para aproximar essa integral. A primeira com os pontos $x_0 = a$ e $x_1 = b$, ou seja, com um espaço ($n = 1$); a segunda com os nós $x_0 = a, x_1$ e $x_2 = b$ -

dois espaços ($n = 2$) - e a terceira com $x_0 = a, x_1, x_2$ e $x_3 = b$ ($n = 3$). Em todos esses casos, trabalhamos com nós igualmente espaçados. A forma de obtenção dessas fórmulas e as conclusões às quais cheguei com o seu estudo serão apresentadas a seguir.

5.3.1 Regra do Trapézio

Seja $p_1(x)$ um polinômio interpolador de $f(x)$ com os nós x_0 e x_1 (veja que, nesse caso, temos $n = 1$). Seja h o espaçamento desses nós. Se estamos trabalhando com nós igualmente espaçados, então, nesse caso, temos $h = b - a = x_1 - x_0$. Sabendo disso, para obter a Fórmula de Newton-Cottes para $n = 1$, basta integrar $p_1(x)$ para obtermos:

$$\int_{x_0}^{x_1} f(x)dx \cong h \cdot \frac{[f(x_0) + f(x_1)]}{2}.$$

A fórmula acima recebe o nome de Regra do Trapézio. Note que, para aplicá-la uma vez, precisamos de um espaço ou dois nós. Se queremos integrar $f(x)$ num intervalo $[a, b]$ dividido em mais espaços, basta dividir esse intervalo em vários sub-intervalos da forma $[x_i, x_{i+1}]$ e aplicar a Regra do Trapézio sucessivas vezes. Fazendo isso, obtemos a generalização dessa regra, que é dada por:

$$\int_{x_0}^{x_n} f(x)dx \cong \frac{h}{2} \cdot \Big[f(x_0) + 2\left(f(x_1) + ... + f(x_{n-1})\right) + f(x_n))\Big]$$

5.3.2 Regra $\frac{1}{3}$ de Simpson

Suponha que dividimos o intervalo $[a, b]$ em dois sub-intervalos, obtendo os nós $x_0 = a$, x_1 e $x_2 = b$, igualmente espaçados a uma distância h. Então, $h = \dfrac{b-a}{2} = \dfrac{x_2 - x_0}{2}$. Se queremos aproximar a integral de $f(x)$ em $[a, b]$, podemos integrar o polinômio que interpola $f(x)$ com os nós x_0, x_1 e x_2, obtendo:

$$\int_{x_0}^{x_2} f(x)dx \cong \frac{h}{3} \cdot \left[f(x_0) + 4f(x_1) + f(x_2)\right].$$

Essa fórmula é conhecida como Regra $\frac{1}{3}$ de Simpson. Veja que, para aplicar essa regra uma vez, necessitamos de 2 espaços (ou 3 nós). Então, para aplicarmos essa regra num intervalo $[a, b]$ dividido em mais sub-intervalos, basta dividir $[a, b]$ num número par $2n$ de sub-intervalos igualmente espaçados e aplicar sucessivas vezes essa regra. Assim, obtemos:

$$\int_{x_0}^{x_{2n}} f(x)dx \cong \frac{h}{3} \cdot \left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + ... + 4f(x_{2n-1}) + f(x_{2n})\right].$$

A fórmula acima é a generalização da Regra $\frac{1}{3}$ de Simpson.

5.3.3 Regra $\frac{3}{8}$ de Simpson

Por último, se vamos aproximar a integral de $f(x)$ num intervalo $[a, b]$ dividido em 3 espaços iguais, com os nós $x_0 = a$, x_1, x_2 e x_3. Basta integrarmos o polinômio que interpola $f(x)$ com esses nós para obtermos:

$$\int_{x_0}^{x_3} f(x)dx \cong \frac{3h}{8} \cdot \Big[f(x_0) + 3\left(f(x_1) + f(x_2)\right) + f(x_3)\Big].$$

Chamamos a fórmula acima de Regra $\frac{3}{8}$ de Simpson. Veja que, para aplicar essa regra uma vez, precisamos de três intervalos, ou seja, 4 nós. Com isso, para aplicar essa regra de forma generalizada, dividimos o intervalo de integração $[a, b]$ em $3n$ sub-intervalos e aplicamos essa regra sucessivas vezes, como nas generalizações anteriores. Fazendo isso, obtemos a Regra $\frac{3}{8}$ de Simpson Generalizada:

$$\begin{aligned}\int_{x_0}^{x_{3n}} f(x)dx \cong \frac{3h}{8} \cdot [&f(x_0) + 3\left(f(x_1) + f(x_2)\right) + 2f(x_3) + 3\left(f(x_4) + f(x_5)\right) + 2f(x_6) \\ &+ \cdots + \\ &2f(x_{3n-3}) + 3\left(f(x_{3n-2}) + f(x_{3n-1})\right) + f(x_{3n})]\end{aligned}$$

5.4 Erro nas Fórmulas de Newton-Cottes

O estudo do erro na aproximação de integrais foi voltado para as três fórmulas de Newton-Cottes estudadas. Veremos que o valor numérico do erro nem sempre pode ser obtido, porém, é possível determinar um limitante superior para o erro. Sabendo disso, o intuito do estudo do erro não é subtraí-lo da aproximação obtida a fim de chegar no resultado exato da integral, mas utilizar o limitante superior para impor quantas casas decimais precisas queremos e, com base nisso, determinar o menor número de espaços com os quais conseguimos essa precisão.

Sejam os nós $x_0, \cdots, x_n$ igualmente espaçados, com $x_0 = a$ e $x_n = b$. Seja h o espaço entre os nós. São dadas duas fórmulas para a obtenção do erro:

- Se $[a, b]$ foi dividido numa quantia **ímpar** de sub-intervalos e $f^{(n+1)}(x)$ existe e é contínua em $[a, b]$, então o erro é dado por:

$$R(f) = \frac{h^{n+2} \cdot f^{(n+1)}(\zeta)}{(n+1!)} \cdot \int_0^n \left[\prod_{i=0}^{n} (u - i) \right] du, \text{para algum } \zeta \in [a, b]. \tag{5.1}$$

- Se $[a, b]$ foi dividido numa quantia **par** de sub-intervalos e $f^{(n+2)}(x)$ existe e é contínua em $[a, b]$, então o erro é dado por:

$$R(f) = \frac{h^{n+3} \cdot f^{(n+2)}(\zeta)}{(n+2!)} \cdot \int_0^n \left[\left(u - \frac{n}{2} \right) \cdot \prod_{i=0}^{n} (u - i) \right] du, \text{para algum } \zeta \in [a, b]. \tag{5.2}$$

Em ambos os casos, o argumento ζ sempre existe e é único, mas nunca é determinado. A única informação que temos é que $\zeta \in [a, b]$. Sendo assim, para estabelecermos o limitante superior para o erro no caso ímpar, por exemplo, assumimos ζ tal que:

$$f^{(n+1)}(\zeta) \leq \max_{a \leq t \leq b} \left| f^{(n+1)}(t) \right|.$$

Sabendo disso, se queremos aproximar a integral de $f(x)$ em $[a, b]$ com uma precisão de duas casas decimais, por exemplo, basta impor $R(f) < 0,5 \cdot 10^{-2}$ que encontraremos o maior espaçamento h com o qual obtemos a precisão desejada. Conhecendo o maior h possível e sabendo que $n = \dfrac{b-a}{h}$, determinamos o menor valor possível de n, que é o número de pontos a serem utilizados na aproximação.

Utilizando as fórmulas 5.1 e 5.2, podemos substituir n por 1, 2 e 3 para obter o erro cometido ao aproximar uma integral com as regras do Trapézio, $\frac{1}{3}$ de Simpson e $\frac{3}{8}$ de Simpson, respectivamente. Além disso, se estamos usando a Regra $\frac{1}{3}$ de Simpson Generalizada, por exemplo, num intervalo dividido em $2k$ espaços, sabemos que estamos aplicando k vezes a Regra $\frac{1}{3}$ de Simpson e, com isso, o erro cometido será k vezes maior. Esse mesmo raciocínio aplica-se à todas as outras Fórmulas de Newton-Cottes.

Sabendo disso, ao utilizarmos as generalizações das três fórmulas estudadas, temos:

- Erro na Regra do Trapézio Generalizada:

$$R(f) = -\frac{(b-a)}{12} \cdot h^2 \cdot f''(\zeta), \text{ com } x_0 \leq \zeta \leq x_n \tag{5.3}$$

- Erro na Regra $\frac{1}{3}$ de Simpson Generalizada:

$$R(f) = -\frac{(b-a)}{180} \cdot h^4 \cdot f^{(4)}(\zeta), \text{ com } x_0 \leq \zeta \leq x_n \tag{5.4}$$

- Erro na Regra $\frac{3}{8}$ de Simpson Generalizada:

$$R(f) = -\frac{(b-a)}{80} \cdot h^4 \cdot f^{(4)}(\zeta), \text{ com } x_0 \leq \zeta \leq x_n \tag{5.5}$$

A respeito disso, é muito interessante notarmos que, na expressão do erro na Regra do Trapézio Generalizada, temos h^2 no numerador e, nas generalizações das duas Regras de Simpson, temos h^4 no numerador. Isso faz com que, à medida que $h \to 0$, o erro nas Regras de Simpson se aproxima de zero mais rapidamente do que na Regra do Trapézio. Com isso, entre as três fórmulas estudadas, as Regras de Simpson mostram-se mais eficientes na aproximação de integrais.

Capítulo 6

Julho de 2020

6.1 Introdução

Inicialmente, no mês de julho, foi estudada a parte de implementação das Fórmulas de Newton-Cotes para a aproximação de integrais. Em seguida, o estudos voltaram-se aos polinômios ortogonais. Essa segunda etapa começou com o estudo de alguns conceitos básicos, mas muito importantes. Depois disso, foi visto o Processo de Ortogonalização de Gram-Schmidt, os principais polinômios ortogonais e suas propriedades mais importantes. Todo esse estudo serviu de base para o início do aprendizado das Fórmulas de Quadratura de Gauss, que mostram-se muito eficientes para a aproximação de integrais e compuseram o último assunto visto no mês de julho.

6.2 Implementação das Fórmulas de Newton-Cotes

Usando integrais de cálculo simples, a intenção dessa parte do estudo era comparar o erro cometido ao aproximar integrais com as regras do Trapézio, de $\frac{1}{3}$ de Simpson e de $\frac{3}{8}$ de Simpson. Por exemplo, se queremos aproximar $\int_0^{1.2} cos(x)dx$ com os métodos estudados, podemos fazer da forma que segue. Primeiramente, dividimos o intervalo $[0, 1.2]$ em 12 espaços iguais, então temos o espaçamento $h = 0.1$ entre os pontos. Em seguida, calculamos $cos(x)$ em cada um desses pontos e temos o tabelamento:

x	0	0.1	0.2	0.3	0.4	0.5	0.6
$cos(x)$	1.0000	0.9950	0.9800	0.9553	0.9210	0.8775	0.8253
x	0.7	0.8	0.9	1.0	1.1	1.2	
$cos(x)$	0.7648	0.6967	0.6216	0.5403	0.4535	0.3623	

Agora, usando o tabelamento acima e as três regras estudadas, aproximamos a integral em questão com as três regras e obtemos os resultados:

- **Usando a Regra do Trapézio:** lembremos que a fórmula para a aproximação nesse caso é a seguinte:

$$\int_{x_0}^{x_n} f(x)dx \cong \frac{h}{2} \cdot \Big[f(x_0) + 2\left(f(x_1) + \ldots + f(x_{n-1})\right) + f(x_n)\Big]$$

  Substituindo os valores do problema, temos que:

$$\int_0^{1.2} cos(x)dx \cong \frac{0.1}{2} \cdot \Big[cos(x_0) + 2\left(cos(x_1) + \ldots + cos(x_{11})\right) + cos(x_{12})\Big]$$

  Substituindo $cos(x_i)$, $i = 0, 1, \ldots, 12$ com base no tabelamento construído, obtemos a aproximação:

$$\int_0^{1.2} cos(x)dx \cong 0.931215$$

- **Usando a Regra $\frac{1}{3}$ de Simpson:** nesse caso, temos a fórmula:

$$\int_{x_0}^{x_{2n}} f(x)dx \cong \frac{h}{3} \cdot \left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 4f(x_{2n-1}) + f(x_{2n})\right].$$

Substituindo os valores do problema, temos:

$$\begin{aligned}\int_0^{1.2} cos(x)dx &\cong \frac{0.1}{3} \cdot \left[cos(x_0) + 4cos(x_1) + 2cos(x_2) + 4cos(x_3) + \ldots + 4cos(x_{11}) + cos(x_{12})\right] \\ &\cong 0.93199\end{aligned}$$

- **Usando a Regra $\frac{3}{8}$ de Simpson:** nesse último caso, usamos a fórmula:

$$\begin{aligned}\int_{x_0}^{x_{3n}} f(x)dx \cong \frac{3h}{8} \cdot [&f(x_0) + 3\left(f(x_1) + f(x_2)\right) + 2f(x_3) + 3\left(f(x_4) + f(x_5)\right) + 2f(x_6) \\ &+ \ldots + \\ &2f(x_{3n-3}) + 3\left(f(x_{3n-2}) + f(x_{3n-1})\right) + f(x_{3n})]\end{aligned}$$

Aplicando os dados do problema, obtemos a aproximação:

$$\begin{aligned}\int_0^{1.2} cos(x)dx \cong \frac{3 \cdot 0.1}{8} \cdot [&cos(x_0) + 3\left(cos(x_1) + cos(x_2)\right) + 2cos(x_3) + 3\left(cos(x_4) + cos(x_5)\right) \\ &+ \ldots + \\ &2cos(x_9) + 3\left(cos(x_{10}) + cos(x_{11})\right) + f(x_{12})]\end{aligned}$$

Substituindo os valores de acordo com o tabelamento, temos que:

$$\int_0^{1.2} cos(x)dx \cong 0.93199125$$

Nas três aproximações, os cálculos foram feitos usando 4 casas decimais. Depois de feitas as aproximações, resta comparar os resultados. Sabemos que:

$$\int_0^{1.2} cos(x)dx = [sen(x)]_0^{1.2} = sen(1.2) = 0.9320$$

Com isso, temos os erros:

	Erro cometido
Regra do Trapézio	$824{,}0859 \cdot 10^{-6}$
Regra 1/3 de Simpson	$10 \cdot 10^{-6}$
Regra 3/8 de Simpson	$8{,}75 \cdot 10^{-6}$

Veja que, como esperado, a Regra do Trapézio teve o maior erro. Por outro lado, o erro cometido com as regras de Simpson foi pequeno e muito próximo. Isso ocorre porque essas duas possuem a mesma ordem de convergência, conforme foi apresentado no relatório do mês anterior.

6.3 Estudo de Conceitos Básicos

Antes de começar o estudo dos polinômios ortogonais, conforme está sugerido no livro [3], foram estudados alguns conceitos básicos, mas de grande importância para o entendimento dos assuntos seguintes. Nessa etapa do estudo, foram vistos brevemente os conceitos de produto escalar, ortogonalidade, base ortogonal e mudança de base.

Seja E um espaço vetorial real e $x, y \in E$. O produto escalar (ou produto interno) de x por y, simbolizado por (x, y) é qualquer função definida em $E \times E$ com valores em $\mathbb{R}$ satisfazendo uma série de propriedades. Dizemos que x e y são ortogonais se $(x, y) = 0$.

Se definimos o produto escalar com o uso de uma função peso $w(x)$, simbolizamos por $(x, y)_w$ o produto interno de x por y. Nesse caso, x e y são w-ortogonais se $(x, y)_w = 0$.

Seja B uma base de E. Dizemos que B é uma base ortogonal de E se seus elementos são dois a dois ortogonais.

6.4 Polinômios Ortogonais

6.4.1 Introdução aos Polinômios Ortogonais

O estudo dos polinômios ortogonais é a base necessária para o entendimento das Fórmulas de Quadratura de Gauss. Sejam $\phi_0(x), \phi_1(x), \phi_2(x), \ldots$ uma família de polinômios de grau $0, 1, 2, \ldots$. Se:

$$\begin{cases} (\phi_i(x), \phi_j(x))_w = 0 \text{ para } i \neq j \text{ e} \\ (\phi_i(x), \phi_i(x))_w \neq 0 \text{ para } \phi_i \neq 0 \end{cases} \tag{6.1}$$

então $\phi_0(x), \phi_1(x), \ldots$ dizem-se w-ortogonais. Nesse caso, estamos trabalhando com o produto escalar:

$$(f, g)_w = \int_a^b w(x) f(x) g(x) dx, \text{ onde } w(x) \text{ é a função peso.}$$

6.4.2 Processo de Ortogonalização de Gram - Schmidt

Uma maneira de obtermos os polinômios $\phi_i(x)$, $i = 0, 1, 2 \ldots$, citados na seção anterior, é utilizando o Processo de Ortogonalização de Gram-Schmidt. Vale lembrar que esse processo não é o mais eficiente, pois para a obtenção de $\phi_k(x)$, devemos conhecer $\phi_{k-1}(x)$. Ou seja, a obtenção dos polinômios $\phi_i(x)$, $i = 0, 1, 2, \ldots$ ocorre na ordem crescente de i. Porém, a lógica simples por trás desse processo é um ponto positivo.

Seja $F = \{f_1, f_2, \ldots, f_n\}$ uma base do espaço vetorial E. Com base nela, vamos construir uma base ortogonal $e_1, e_2, \ldots, e_n$ para E. Primeiro, tomamos:

$$e_1 = f_1.$$

Agora, definimos e_2 como combinação linear de f_2 e e_1:

$$e_2 = f_2 + \alpha_1 e_1,$$

onde α_1 é escolhido de forma que e_1 e e_2 sejam ortogonais, ou seja, $(e_1, e_2) = 0$.

Da mesma forma, e_3 será combinação linear de f_3, e_2 e e_1, como segue:

$$e_3 = f_3 + \alpha_2 e_2 + \alpha_1 e_1,$$

onde α_2 e α_1 são tais que e_3 é ortogonal aos demais, ou seja, $(e_3, e_1) = 0$ e $(e_3, e_2) = 0$.

Por fim, supondo que já conhecemos $e_1, e_2, \ldots, e_{k-1}$, todos dois a dois ortogonais, temos que e_k será combinação linear de f_k e todos os e_i já calculados, ou seja:

$$e_k = f_k + \alpha_{k-1} e_{k-1} + \alpha_{k-2} e_{k-2} + \ldots + \alpha_1 e_1,$$

onde os α_i, $i = 1, 2, \ldots, k-1$ são tais que e_k é ortogonal a e_j, $j = 0, 1, \ldots, k-1$.

Feito isso, basta sabermos determinar as constantes α_i, $i = 1, 2, \ldots, k-1$. Seja o produto escalar (e_k, e_j), com $1 \leq j < k$. Pela construção dos polinômios e_i, $i = 1, 2, \ldots, k$, devemos ter $e_k \perp e_j$, ou seja:

$$\begin{aligned} (e_k, e_j) = 0 &\Rightarrow (f_k + \alpha_{k-1} e_{k-1} + \alpha_{k-2} e_{k-2} + \ldots + \alpha_j e_j + \ldots + \alpha_1 e_1, e_j) = 0 \\ &\Rightarrow (f_k, e_j) + \alpha_{k-1}(e_{k-1}, e_j) + \alpha_{k-2}(e_{k-2}, e_j) + \ldots + \alpha_j (e_j, e_j) + \ldots + \alpha_1 (e_1, e_j) = 0 \end{aligned}$$

Veja que, de acordo com a definição 6.1 dos polinômios ortogonais, temos que $(e_i, e_j) = 0$ para $i \neq j$ e $(e_j, e_j) \neq 0$ para $e_j \neq 0$. Então temos:

$$\begin{aligned} &\Rightarrow (f_k, e_j) + \cancel{\alpha_{k-1}(e_{k-1}, e_j)} + \cancel{\alpha_{k-2}(e_{k-2}, e_j)} + \ldots + \alpha_j (e_j, e_j) + \ldots + \cancel{\alpha_1 (e_1, e_j)} = 0 \\ &\Rightarrow (f_k, e_j) + \alpha_j (e_j, e_j) = 0 \\ &\Rightarrow \alpha_j = -\frac{(f_k, e_j)}{(e_j, e_j)} \end{aligned}$$

Com isso, conseguimos determinar todas as constantes α_i ao longo do processo de ortogonalização.

6.4.3 Principais Polinômios Ortogonais

É lógico que a sequência de polinômios $\phi_0(x), \phi_1(x), \ldots$ depende do produto escalar adotado. Sabemos que o produto escalar $(f, g)_w$ é dado por:

$$(f, g)_w = \int_a^b w(x)f(x)g(x)dx.$$

Veja que a definição do produto escalar depende da função peso $w(x)$ e do intervalo de integração. Escolhendo os produtos escalares a seguir, temos os principais polinômios ortogonais:

- **Polinômios de Legendre:** obtidos com o produto escalar:

$$(f, g)_w = \int_{-1}^1 f(x)g(x)dx, \text{ com } w(x) = 1, \ a = -1 \text{ e } b = 1.$$

- **Polinômios de Tchebyshev:** obtidos com o produto escalar:

$$(f, g)_w = \int_{-1}^1 \frac{1}{\sqrt{1-x^2}} f(x)g(x)dx, \text{ com } w(x) = \frac{1}{\sqrt{1-x^2}}, \ a = -1 \text{ e } b = 1.$$

- **Polinômios de Laguerre:** nesse caso, o produto escalar adotado é o seguinte:

$$(f, g)_w = \int_0^\infty e^{-x} f(x)g(x)dx, \text{ isto é, } w(x) = e^{-x}, \ a = 0 \text{ e } b = \infty.$$

- **Polinômios de Hermite:** obtidos com o produto escalar:

$$(f, g)_w = \int_{-\infty}^\infty e^{-x^2} f(x)g(x)dx, \text{ com } w(x) = e^{-x^2}, \ a = -\infty \text{ e } b = \infty.$$

6.4.4 Propriedades dos Polinômios Ortogonais

Finalizados os entendimentos sobre a definição, o processo de ortogonalização e os principais polinômios ortogonais, o próximo passo do estudo foi estudar as propriedades mais importantes desses polinômios. São quatro propriedades principais.

Sejam $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ polinômios ortogonais não-nulos, segundo um produto escalar qualquer. Basicamente, as propriedades são:

Propriedade 6.1. Qualquer polinômio de grau menor ou igual a n pode ser escrito como combinação linear de $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$.

Propriedade 6.2. $\phi_n(x)$ é ortogonal a qualquer polinômio de grau menor do que n.

Propriedade 6.3. Se o produto escalar adotado for:

$$(f, g)_w = \int_a^b w(x)f(x)g(x)dx, \text{ com } w(x) \geq 0 \text{ e contínua em } [a, b],$$

então $\phi_n(x)$ possui n raízes distintas reais em $[a, b]$.

Propriedade 6.4. Sejam $x_0, x_1, \ldots, x_n$ as raízes de $\phi_{n+1}(x)$. Se $f(x)$ é um polinômio de grau menor ou igual a $2n + 1$, então:

$$\int_a^b w(x)f(x)dx = \sum_{k=0}^n f(x_k) \int_a^b w(x)l_k(x)dx,$$

onde $l_k(x)$ são os polinômios de Lagrange, obtidos na interpolação de $f(x)$ com os nós $x_0, \ldots, x_n$.

A demonstração das duas primeiras propriedades decorre diretamente do fato de que $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ é uma base para o espaço vetorial dos polinômios de grau menor ou igual a n.

Por outro lado, as duas últimas têm uma demonstração mais extensa. Para provar a terceira propriedade, dividimos o processo em três partes:

- Primeiro, supomos que $\phi_n(x)$ não possui raízes em $[a, b]$, o que leva-nos a um absurdo.
- Depois, provamos que as raízes de $\phi_n(x)$ em $[a, b]$ são simples.
- Por último, provamos que os n zeros de $\phi_n(x)$ estão em $[a, b]$.

Agora, para provar a quarta propriedade, escrevemos ϕ_{n+1} em função de suas raízes. Sabemos que, dado o polinômio interpolador $P_n(x)$ de $f(x)$ com os nós $x_0, x_1, \ldots, x_n$, temos que:

$$f(x) - P_n(x) = R_n(x), \text{ onde } R_n(x) \text{ é o erro cometido.}$$

Usando as propriedades anteriores e a expressão que define o erro, podemos escrever $R_n(x)$ em função de $\phi_{n+1}(x)$, obtendo:

$$f(x) - P_n(x) = b_0 \cdot \phi_{n+1}(x) \cdot q(x), \tag{6.2}$$

onde b_0 é uma constante. Em seguida, mostramos que $q(x)$ é um polinômio de grau menor do que $n+1$. Com essa informação, considerando o produto escalar com a função peso $w(x)$ e usando a Propriedade 2, temos que:

$$\phi_{n+1}(x) \perp q(x) \Rightarrow \left(\phi_{n+1}, q(x)\right)_w = 0.$$

Por último, integrando ambos os lados da igualdade 6.2 com a função peso $w(x)$, temos:

$$\begin{aligned}
&\int_a^b w(x)[f(x) - P_n(x)]dx = \int_a^b \left[w(x) \cdot b_0 \cdot \phi_{n+1}(x) \cdot q(x)\right] dx \\
\Rightarrow &\int_a^b w(x)f(x)dx - \int_a^b w(x)P_n(x)dx = b_0 \cdot \cancel{\int_a^b \left[w(x) \cdot \phi_{n+1}(x) \cdot q(x)\right] dx} = 0 \\
\Rightarrow &\int_a^b w(x)f(x)dx = \int_a^b w(x)P_n(x)dx
\end{aligned}$$

Como $P_n(x)$ é o polinômio interpolador de $f(x)$ com os nós $x_0, \ldots, x_n$, se interpolarmos pelo Método de Lagrange, temos:

$$P_n(x) = \sum_{k=0}^{n} f(x_k)l_k(x).$$

Substituindo, temos:

$$\int_a^b w(x)f(x)dx = \sum_{k=0}^{n} \left[f(x_k) \int_a^b w(x)l_k(x)dx\right]. \tag{6.3}$$

6.5 Fórmulas de Quadratura de Gauss

A parte principal no que refere-se às Fórmulas de Quadratura de Gauss é a obtenção da igualdade 6.3, na demonstração da proposição 4. Depois de estudada a obtenção dessa igualdade, o estudo voltou-se à sua aplicação.

Levando em conta a proposição 4, se queremos determinar $\int_a^b w(x)f(x)dx$, com $gr(f) = k$, fazemos $k \leq 2n+1 \Rightarrow \dfrac{k-1}{2} \leq n$. É óbvio que, quanto menor o valor de n, menos contas teremos a fazer, então fazemos $n = \dfrac{k-1}{2}$, dessa forma teremos o menor valor de n tal que o valor da integral calculado na fórmula de quadratura é exato.

Agora, devemos encontrar $\phi_{n+1}(x)$ com o produto escalar adequado, ou seja, no intervalo $[a, b]$ e com a função peso $w(x)$. Sendo assim, usamos o produto escalar:

$$(f, g)_w = \int_a^b w(x)f(x)g(x)dx.$$

Feito isso, encontremos os zeros de $\phi_{n+1}(x)$ em $[a, b]$. Lembremos que, pela propriedade 3, a função $\phi_{n+1}(x)$ possui $n+1$ raízes reais em $[a, b]$. Sejam $x_0, \ldots, x_n$ essas raízes de ϕ_{n+1} em $[a, b]$.

Agora, resta calcular $f(x_k)$, $k = 0, 1 \ldots, n$ e usar a fórmula:

$$\int_a^b w(x)f(x)dx = \sum_{k=0}^{n} \left[f(x_k) \int_a^b w(x) l_k(x) dx \right],$$

onde $l_k(x)$, $k = 0, 1 \ldots, n$ são os polinômios de Lagrange obtidos na interpolação de $f(x)$ com os nós $x_0, \ldots, x_n$ (raízes de $\phi_{n+1}(x)$).

Veja que, por fim, conhecemos $f(x_k)$, $l_k(x)$ e $w(x)$. Portanto, podemos usar a fórmula em questão para obter o **valor exato** da integral que queríamos calcular.

Conforme vimos na subseção 6.4.3, sobre os principais polinômios ortogonais, sabemos que a determinação dos polinômios ortogonais $\phi_0(x), \phi_1(x), \ldots, \phi_{n+1}(x)$ depende do produto escalar adotado que, por sua vez, depende da escolha do intervalo de integração $[a, b]$ e da função peso $w(x)$.

Sendo assim, se queremos aproximar uma integral cujo intervalo de integração e a função peso coincidem com os adotados em algum produto escalar dos principais polinômios ortogonais, basta calcularmos os valores de $f(x_0), \ldots, f(x_n)$, pois nesse caso os valores de x_k e A_k já encontram-se tabelados, onde $A_k = \int_a^b w(x) l_k(x) dx$

Como exemplo de aplicação das Fórmulas de Quadratura de Gauss, temos o seguinte exercício, retirado do livro [3]:

Exemplo 6.1. Calcular $\int_{-1}^{1}(x^3 + x^2 + x + 1)dx$ por quadratura de Gauss e diretamente e comparar os resultados.

Solução. Calculando a integral diretamente, temos:

$$\begin{aligned}
\int_{-1}^{1}(x^3 + x^2 + x + 1)dx &= \left[\frac{x^4}{4} + \frac{x^3}{3} + \frac{x^2}{2} + x \right]_{-1}^{1} \\
&= \left(\frac{1}{4} + \frac{1}{3} + \frac{1}{2} + 1 \right) - \left(\frac{1}{4} - \frac{1}{3} + \frac{1}{2} - 1 \right) \\
&= \frac{1}{3} + \frac{1}{3} + 1 + 1 \\
&= \frac{2}{3} + 2 \\
&= 0.666666666 + 2 \\
&= 2.666666666
\end{aligned}$$

Agora, calcularemos a integral usando a fórmula de quadratura estudada. Veja que o intervalo de integração é $[-1, 1]$ e a função peso é $w(x) = 1$, ou seja, devemos escolher os polinômios de Legendre para usar na fórmula de quadratura pois, esses polinômios, conforme vimos na subseção 6.4.3, são obtidos com o produto escalar:

$$(f, g)_w = \int_{-1}^{1} f(x)g(x)dx, \text{ com } w(x) = 1,\ a = -1 \text{ e } b = 1.$$

Agora, usando a fórmula de quadratura e chamando $A_k = \int_{-1}^{1} w(x) l_k(x) dx$, temos:

$$\int_{-1}^{1}(x^3 + x^2 + x + 1)dx = \sum_{k=0}^{1} f(x_k)A_k.$$

Os valores de x_k e A_k encontram-se na tabela de valores da fórmula de quadratura de Gauss-Legendre e temos:

$$x_0 = -0,5773502691,\ x_1 = 0,5773502691,\ A_0 = 1 \text{ e } A_1 = 1.$$

Substituindo isso, temos:

$$\int_{-1}^{1}(x^3 + x^2 + x + 1)dx = f(x_0)A_0 + f(x_1)A_1 = f(-0,5773502691) + f(0,5773502691).$$

Sabendo que $f(x) = x^3 + x^2 + x + 1$, temos $f(-0,5773502691) = 0.563532974$ e $f(0.5773502691) = 2.1031336919701$. Substituindo na fórmula:

$$\int_{-1}^{1} (x^3 + x^2 + x + 1)dx = 0.563532974 + 2.1031336919701 = 2.666666666.$$

□

O estudo mais detalhado sobre as fórmulas de Gauss-Legendre, Gauss-Tchebyshev, Gauss-Laguerre e Gauss-Hermite, bem como o erro em cada uma dessas fórmulas, constitui o próximo alvo do estudo.

Capítulo 7

Setembro de 2020

7.1 Introdução

O primeiro estudo do mês de setembro foi a implementação das fórmulas de aproximação de integrais estudadas: fórmulas de Newton-Cottes e fórmulas de quadratura de Gauss.

Ao longo do relatório, serão relembradas brevemente as fórmulas utilizadas em cada caso para fazer a aproximação de integrais. Em seguida, para cada caso, serão utilizadas funções no MATLAB para fazer sua implementação.

Ao fim do relatório, serão feitas algumas comparações dos resultados obtidos por meio de algumas tabelas.

Observação: o relatório, em seu formato atual, não está finalizado. Pode-se ver, no título das seções 3.6 e 3.7 a observação "em andamento".

7.2 Implementação das Fórmulas de Aproximação de Integrais

Primeiro, foram feitas funções no MATLAB para fazer as aproximações com a Regra do Trapézio, as regras de $\frac{1}{3}$ e de $\frac{3}{8}$ de Simpson e as fórmulas de quadratura de Gauss-Legendre, (...). Com cada método de aproximação, foram estudadas as aproximações de algumas integrais. Por último, foi feita a comparação dos resultados obtidos por meio de uma tabela.

A seguir, apresentarei cada fórmula juntamente com o código da sua função no MATLAB.

7.3 Regra do Trapézio

A generalização dessa regra e o código da sua função no MATLAB são os seguintes:

$$\int_{x_0}^{x_n} f(x)dx \cong \frac{h}{2} \cdot \Big[f(x_0) + 2\left(f(x_1) + ... + f(x_{n-1})\right) + f(x_n))\Big]$$

Figura 7.1: Função da Regra do Trapézio Generalizada

```
1   %REGRA DO TRAPÉZIO GENERALIZADA
2   function integral = trapezio(a,b,n)
3   % a = início do intervalo de integração
4   % b = fim do intervalo de integração
5   % n = quantia de nós
6
7   format long
8   %encontrando as ordenadas
9   x = linspace(a,b,n);
10  for i=1:n
11      %inserir a função aqui:
12      f(i) = x(i)^3 + x(i)^2 + x(i) + 1;
13  end
14
15  %integral = (h/2)*[ f(x_0)+ 2(f(x_1)+f(x_2)+...+f(x_{n-1})) +f(x_n) ]
16
17  %somatorio
18  somatorio = 0;
19  for k=2:n-1
20      somatorio = somatorio + f(k);
21  end
22
23  %espaçamento
24  h = (b-a)/(n-1);
25
26  %resultado final
27  integral = (h/2)*(f(1)+ 2*somatorio + f(n));
```

A função "trapezio(a,b,n)"necessita que declaremos a função a ser integrada dentro do seu código. Por exemplo, na figura 9.6, na linha 12, vemos que a função a ser integrada é $f(x) = x^3 + x^2 + x + 1$. Depois disso, para usá-la, o intervalo de integração é $[a, b]$ e n é o número de nós que usaremos na aproximação. Sabendo disso, basta substituir os valores de a, b e n como veremos a seguir.

Para aproximar a integral dessa $f(x)$ no intervalo $[-1, 1]$ com 13 nós, fazemos:

Figura 7.2: Aproximação com 13 nós.

```
>> trapezio(-1,1,13)

ans =

   2.675925925925926

fx >>
```

Podemos melhorar a precisão aumentando o número de nós para 21, por exemplo, obtendo:

Figura 7.3: Aproximação com 21 nós.

```
>> trapezio(-1,1,21)

ans =

   2.670000000000000

fx >>
```

7.4 Regra $\frac{1}{3}$ de Simpson

A generalização dessa regra e o código da sua função no MATLAB são, respectivamente:

$$\int_{x_0}^{x_{2n}} f(x)dx \cong \frac{h}{3} \cdot \left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + ... + 4f(x_{2n-1}) + f(x_{2n})\right].$$

Figura 7.4: Função da Regra $\frac{1}{3}$ de Simpson Generalizada.

```
%REGRA 1/3 DE SIMPSON GENERALIZADA
function integral = um_terco_simpson(a,b,n)
% a = início do intervalo de integração
% b = fim do intervalo de integração
% n = quantia de nós
% (n deve ser ÍMPAR para termos uma quantia PAR de sub-intervalos)

format long
%encontrando as ordenadas
x = linspace(a,b,n);
for i=1:n
    %inserir a função aqui:
    f(i) = x(i)^3 + x(i)^2 + x(i) + 1;
end

%integral = (3h/8)*[ f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3)
%            +...+
%           4f(x_{2n-1}) + f(x_{2n}) ]

%somatorio dos termos centrais
somatorio = 0;
for i=2:n-1
   if rem(i,2)== 0
       somatorio = somatorio + 4*f(i);
   end
   if rem(i,2)== 1
       somatorio = somatorio + 2*f(i);
   end
end

%espaçamento
h = (b-a)/(n-1);

%resultado final
integral = (h/3)*(f(1) + somatorio + f(n));
```

Como no caso anterior, devemos declarar a função a ser integrada dentro do código da função no MATLAB. Nesse caso, para aproximar a integral de $f(x) = x^3 + x^2 + x + 1$, no intervalo $[-1, 1]$, com 13 nós, fazemos:

Figura 7.5: Aproximação com a Regra $\frac{1}{3}$ de Simpson.

```
>> um_terco_simpson(-1,1,13)

ans =

   2.666666666666667

fx >>
```

7.5 Regra $\frac{3}{8}$ de Simpson

A generalização dessa regra e a sua função no MATLAB, respectivamente, são:

$$\int_{x_0}^{x_3n} f(x)dx \cong \frac{3h}{8} \cdot [f(x_0) + 3\left(f(x_1) + f(x_2)\right) + 2f(x_3) + 3\left(f(x_4) + f(x_5)\right) + 2f(x_6)$$
$$+ \cdots +$$
$$2f(x_{3n-3}) + 3\left(f(x_{3n-2}) + f(x_{3n-1})\right) + f(x_{3n})]$$

Figura 7.6: Função da Regra $\frac{3}{8}$ de Simpson Generalizada.

```
%REGRA 3/8 DE SIMPSON GENERALIZADA
function integral = tres_oitavos_simpson(a,b,n)
% a = início do intervalo de integração
% b = fim do intervalo de integração
% n = quantia de nós
% (n=3k+1, k\in\Z, para termos uma quantia múltipla de 3 de sub-intervalos)

format long
%encontrando as ordenadas
x = linspace(a,b,n);
for i=1:n
    %inserir a função aqui:
    f(i) = x(i)^3 + x(i)^2 + x(i) + 1;
end

%integral = (3h/8)*[ f(x_0) + 3(f(x_1) + f(x_2)) + 2f(x_3)
%            +...+
%            2f(x_{3n-3}) + 3(f(x_{3n-2}) + f(x_{3n-1})) + f(x_{3n})]

%somatorio dos termos centrais
somatorio = 0;
for i=2:n-1
   if rem(i,3)== 2 | rem(i,3)== 0
       somatorio = somatorio + 3*f(i);
   end
   if rem(i,3)== 1
       somatorio = somatorio + 2*f(i);
   end
end

%espaçamento
h = (b-a)/(n-1);

%resultado final
integral = ((h*3)/8)*(f(1) + somatorio + f(n));
```

Com essa função, para aproximarmos a integral de $f(x) = x^3 + x^2 + x + 1$, no intervalo $[-1, 1]$, com 13 nós, fazemos:

Figura 7.7: Aproximação com a Regra $\frac{3}{8}$ de Simpson.

```
>> tres_oitavos_simpson(-1,1,13)

ans =

   2.666666666666667

fx >>
```

7.6 Fórmula de Quadratura de Gauss-Legendre

Usamos os polinômios de Legendre para aproximar uma integral do tipo:

$$\int_{-1}^{1} f(x)dx, \text{ com a função peso } w(x) = 1,\ a = -1 \text{ e } b = 1.$$

Basicamente, na prática, para usar a fórmula de quadratura de Gauss-Legendre na aproximação de uma função $f(x)$, fazemos:

$$\int_{-1}^{1} f(x)dx \cong \sum_{k=0}^{n} A_k f(x_k),$$

onde n é o grau dos polinômios de Legendre a serem utilizados na aproximação. Os valores de A_k e x_k são tabelados da seguinte forma:

Figura 7.8: Tabela de A_k e x_k para a fórmula de Gauss-Legendre.

TABELA 1 $\int_{-1}^{1} f(x)dx$

x_i	A_i
N = 2	
0.5773502691	(1)0.1000000000
N = 3	
0.7745966692	0.5555555555
0.0000000000	0.8888888888
N = 4	
0.8611363115	0.3478548451
0.3399810435	0.6521451548
N = 5	
0.9061798459	0.2369268850
0.5384693101	0.4786286704
0.0000000000	0.5688888888
N = 6	
0.9324695142	0.1713244923
0.6612093864	0.3607615730
0.2386191860	0.4679139345
N = 7	
0.9491079123	0.1294849661
0.7415311855	0.2797053914
0.4058451513	0.3818300505
0.0000000000	0.4179591836
N = 8	
0.9602898564	0.1012285362
0.7966664774	0.2223810344
0.5255324099	0.3137066458
0.1834346424	0.3626837833

Fonte: (FRANCO, 2006, p. 375).

Observação: em todas as tabelas, temos $N = n + 1$.

Sendo assim, o código desta fórmula de quadratura no MATLAB é o seguinte:

Figura 7.9: Função da Fórmula de Quadratura de Gauss-Legendre.

```
1   %FÓRMULA DE GAUSS-LEGENDRE
2   %função peso w(x)=1
3   %intervalo de integração: [-1,1]
4   function integral = legendre(n)
5   %n é o grau do polinômio que usaremos
6   %para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato
7
8   %tabela 1 (gauss-legendre)
9   %x_i
10  xi = [-0.5773502691, 0.5773502691, 0, 0,                          %N=2
11        -0.7745966692, 0.7745966692, 0, 0,                          %N=3
12        -0.8611363115, 0.8611363115, -0.3399810435, 0.3399810435 %N=4
13        ];
14
15  %A_i
16  ai = [1, 1, 0, 0,                                                 %N=2
17        0.5555555555, 0.5555555555, 0.8888888888, 0,                %N=3
18        0.3478548451, 0.3478548451, 0.6521451548, 0.6521451548 %N=4
19       ];
20
21  format long
22  N = n+1;
23
24  %calculando f(x_i)
25  for i=1:N
26      %inserir a função aqui:
27      f(i) = xi(N-1,i)^3 + xi(N-1,i)^2 + xi(N-1,i) + 1;
28  end
29
30  %resultado final
31  integral = 0
32  for k=1:N
33      integral = integral + f(k)*ai(N-1,k)
34  end
```

Note que alguns valores tabelados de A_k e x_k foram inseridos nas matrizes xi (linha 10) e ai (linha 16). Dessa forma, a integral pode ser aproximada usando os valores da tabela 7.8.

Para usar essa função, inserimos a função a ser integrada diretamente no seu código e declaramos o valor de n, que é o grau do polinômio de Legendre a ser utilizado na aproximação.

Conforme está explicado nas linhas 5 e 6 do código, se vamos integrar um polinômio $f(x)$, fazemos $2n + 1 = gr(f)$ e descobrimos o valor de n tal que o cálculo da integral é exato. Se $f(x)$ não é um polinômio, fixamos um n qualquer e teremos uma aproximação.

Para aproximar a integral de $f(x) = x^3 + x^2 + x + 1$ no intervalo $[-1, 1]$, vemos que $f(x)$ é um polinômio de grau 3, então $2n + 1 = 3 \Rightarrow n = 1$. Aproximando, temos:

Figura 7.10: Aproximação com a Fórmula de Quadratura de Gauss-Legendre.

```
>> legendre(1)

ans =

   2.666666666459685

fx >>
```

7.7 Fórmula de Quadratura de Gauss-Tchebyshev

Para usar essa fórmula - ou, de modo geral - para usar qualquer fórmula de quadratura de Gauss, o processo será o mesmo: analisamos a função peso envolvida e o intervalo de integração, escolhemos a fórmula que coincide com ambos e aplicamos. Caso o intervalo de integração não coincida com o do produto escalar envolvido nos polinômios que usaremos na aproximação, basta fazermos uma mudança de variável.

Sendo assim, a única diferença na aplicação de cada fórmula são os valores tabelados de A_k e x_k. Para a fórmula de Gauss-Tchebyshev, temos a tabela:

Figura 7.11: Tabela de A_k e x_k para a fórmula de Gauss-Tchebyshev.

TABELA 2 $\int_{-1}^{1} (1-x^2)^a f(x)dx$

x_i	A_i
$a = -1/2$	
N = 2	
0.7071067811	(1)0.1570796326
N = 3	
0.8660254037	(1)0.1047197551
0.0000000000	(1)0.1047197551
N = 4	
0.9238795325	0.7853981633
0.3826834323	0.7853981633
N = 5	
0.9510565162	0.6283185307
0.5877852522	0.6283185307
0.0000000000	0.6283185307
N = 6	
0.9659258262	0.5235987755
0.7071067811	0.5235987755
0.2588190451	0.5235987755
N = 7	
0.9749279121	0.4487989505
0.7818314824	0.4487989505
0.4338837391	0.4487989505
0.0000000000	0.4487989505
N = 8	
0.9807852804	0.3926990816
0.8314696123	0.3926990816
0.5555702330	0.3926990816
0.1950903220	0.3926990812

Fonte: (FRANCO, 2006, p. 376).

Inserindo alguns valores dessa tabela no código, a função do MATLAB para essa fórmula é a seguinte:

Figura 7.12: Função da Fórmula de Quadratura de Gauss-Tchebyshev.

```
1   %FÓRMULA DE GAUSS-TCHEBYSHEV
2   %função peso w(x)=1/(sqrt(1-x^2))
3   %intervalo de integração: [-1,1]
4   function integral = tchebyshev(n)
5   %n é o grau do polinômio que usaremos
6   %para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato
7
8   %tabela 2 (gauss-tchebyshev)
9   %x_i
10  xi = [-0.7071067811 , 0.7071067811, 0, 0,                          %N=2
11        -0.8660254037 , 0.8660254037, 0, 0,                          %N=3
12        -0.9238795325 , 0.9238795325 , -0.3826834323 , 0.3826834323 %N=4
13        ];
14
15  %A_i
16  ai = [1.570796326, 1.570796326, 0, 0,                         %N=2
17        1.047197551, 1.047197551, 1.047197551, 0,               %N=3
18        0.7853981633, 0.7853981633, 0.7853981633, 0.7853981633 %N=4
19       ];
20
21  format long
22  N = n+1;
23
24  %calculando f(x_i)
25  for i=1:N
26      %inserir a função aqui:
27      f(i) = (sin(xi(N-1,i)))/(sqrt(1-xi(N-1,i)^2));
28  end
29
30  %resultado final
31  integral = 0;
32  for k=1:N
33      integral = integral + f(k)*ai(N-1,k);
34  end
```

Veja, na linha 27, que a função inserida é $f(x) = \dfrac{sen(x)}{\sqrt{1-x^2}}$ e queremos aproximar sua integral no intervalo $[-1, 1]$. Como essa função não é um polinômio, podemos fixar um n qualquer para aproximar. Fixando $n = 1$ obtemos a aproximação:

Figura 7.13: Aproximação com a Fórmula de Quadratura de Gauss-Tchebyshev.

```
>> tchebyshev(1)

ans =

     0

fx >>
```

7.8 Fórmula de Quadratura de Gauss-Laguerre

Usamos essa fórmula para fazer uma aproximação do tipo:

$$\int_0^{\infty} e^{-x} f(x)dx \cong \sum_{k=0}^{n} A_k f(x_k),$$

onde n é o grau dos polinômios a serem utilizados na aproximação. Ou seja, essa fórmula é indicada quando o intervalo de integração é $[0, \infty]$ e a função peso envolvida é $w(x) = e^{-x}$.

Para essa fórmula, temos um tabelamento diferente dos demais. Neste caso, os valores de x_k são todos positivos e já estão todos especificados. O tabelamento de A_k e x_k é o seguinte:

Figura 7.14: Tabela de A_k e x_k para a fórmula de Gauss-Laguerre.

TABELA 3 $\int_0^{\infty} e^{-x} f(x)dx$

x_i	A_i
N = 2	
0.5857864376	0.8535533905
(1)0.3414213562	0.1464466094
N = 3	
0.4157745567	0.7110930099
(1)0.2294280360	0.2785177335
(1)0.6289945082	(-1)0.1038925650
N = 4	
0.3225476896	0.6031541043
(1)0.1745761101	0.3574186924
(1)0.4536620296	(-1)0.3888790851
(1)0.9395070912	(-3)0.5392947055
N = 5	
0.2635603197	0.5217556105
(1)0.1413403059	0.3986668110
(1)0.3596425771	(-1)0.7594244968
(1)0.7085810005	(-2)0.3611758679
(2)0.1264080084	(-4)0.2336997238
N = 6	
0.2228466041	0.4589646739
(1)0.1188932101	0.4170008307
(1)0.2992736326	0.1133733820
(1)0.5775143569	(-1)0.1039919745
(1)0.9837467418	(-3)0.2610172028
(2)0.1598297398	(-6)0.8985479064

Fonte: (FRANCO, 2006, p. 377).

Usando isso, o código da função dessa fórmula no MATLAB ficou muito parecido com os anteriores. Porém, com os valores da tabela 7.17, como veremos:

Figura 7.15: Função da Fórmula de Quadratura de Gauss-Laguerre.

```
%FÓRMULA DE GAUSS-LAGUERRE
%função peso w(x)=e^(-x)
%intervalo de integração: [0, infinito]
function integral = laguerre(n)
%n é o grau do polinômio que usaremos
%para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato

%tabela 3 (gauss-laguerre)
%x_i
xi = [0.5857864376, 3.414213562, 0, 0,                    %N=2
      0.4157745567, 2.29428036, 6.289945082, 0];          %N=3

%A_i
ai = [0.8535533905, 0.1464466094, 0,0,                    %N=2
      0.7110930099, 0.2785177335, 0.01038925650, 0];      %N=3

format long
N = n+1;

%calculando f(x_i)
for i=1:N
    %inserir a função aqui:
    %função peso (exp(-xi(N-1,i)))*
    f(i) = cos(xi(N-1,i));
end

%resultado final
integral = 0;
for k=1:N
    integral = integral + f(k)*ai(N-1,k);
end
```

No código acima, a função inserida foi $f(x) = cos(x)$. Queremos aproximar a integral desta função no intervalo $[0, \infty]$ com a função peso $w(x) = e^{-x}$. O resultado obtido é o seguinte:

Figura 7.16: Aproximação com a Fórmula de Quadratura de Gauss-Laguerre.

```
>> laguerre(2)

ans =

   0.476520838769488

fx >>
```

7.9 Fórmula de Quadratura de Gauss-Hermite

Usamos essa fórmula para fazer uma aproximação do tipo:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \cong \sum_{k=0}^{n} A_k f(x_k),$$

onde n é o grau dos polinômios a serem utilizados na aproximação. Ou seja, essa fórmula é indicada quando o intervalo de integração é $[-\infty, \infty]$ e a função peso envolvida é $w(x) = e^{-x^2}$.

Para essa fórmula, temos o seguinte tabelamento de A_k e x_k:

Figura 7.17: Tabela de A_k e x_k para a fórmula de Gauss-Hermite.

TABELA 4 $\int_{-\infty}^{\infty} e^{-x^2} f(x)dx$

x_i	A_i
N = 2	
0.7071067811	0.8862269254
N = 3	
(1)0.1224744871	0.2954089751
0.0000000000	(1)0.1181635900
N = 4	
(1)0.1650680123	(-1)0.8131283544
0.5246476323	0.8049140900
N = 5	
(1)0.2020182870	(-1)0.1995324205
0.9585724646	0.3936193231
0.0000000000	0.6453087204
N = 6	
(1)0.2350604973	(-2)0.4530009905
(1)0.1335849074	0.1570673203
0.4360774119	0.7246295952
N = 7	
(1)0.2651961356	(-3)0.9717812450
(1)0.1673551628	(-1)0.5451558281
0.8162878828	0.4256072526
0.0000000000	0.8102646175
N = 8	
(1)0.2930637420	(-3)0.1996040722
(1)0.1981656756	(-1)0.1707798300
(1)0.1157193712	0.2078023258
0.3811869902	0.6611470125

Fonte: (FRANCO, 2006, p. 378).

Inserindo alguns valores da tabela e a função que queremos aproximar, ficamos com o seguinte código da função no MATLAB:

Figura 7.18: Função da Fórmula de Quadratura de Gauss-Hermite.

```
%FÓRMULA DE GAUSS-HERMITE
%função peso w(x)=exp(-x^2)
%intervalo de integração: [-infinito, infinito]
function integral = hermite(n)
%n é o grau do polinômio que usaremos
%para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato

%tabela 4 (gauss-hermite)
%x_i
xi = [-0.7071067811, 0.7071067811 , 0, 0,                        %N=2
      -1.224744871, 1.224744871, 0, 0,                           %N=3
      -1.650680123, 1.650680123, -0.5246476323, 0.5246476323     %N=4
      ];

%A_i
ai = [0.8862269254, 0.8862269254, 0, 0,                          %N=2
      0.2954089751 , 0.2954089751, 1.181635900, 0,               %N=3
      0.08131283544, 0.08131283544, 0.8049140900, 0.8049140900 %N=4
     ];

format long
N = n+1;

%calculando f(x_i)
for i=1:N
    %inserir a função aqui:
    %função peso (exp(-xi(N-1,i)^2))*
    f(i) = (xi(N-1,i)^2)/2;
end

%resultado final
integral = 0;
for k=1:N
    integral = integral + f(k)*ai(N-1,k);
end
```

Veja que a função inserida foi $f(x) = \dfrac{x^2}{2}$. Ou seja, estamos aproximando a integral:

$$\int_{-\infty}^{\infty} e^{-x^2} \cdot \frac{x^2}{2} dx.$$

O resultado obtido foi o seguinte:

Figura 7.19: Aproximação com a Fórmula de Quadratura de Gauss-Hermite.

```
>> hermite(1)

ans =

   0.443113462591529

fx >>
```

7.10 Complicações dos Primeiros Estudos

Nos primeiros estudos feitos com a implementação das fórmulas de quadratura, as funções das fórmulas de Gauss-Laguerre e de Gauss-Hermite não ofereceram uma aproximação significativa para as integrais. Depois de algum tempo revisando os códigos e comparando com o cálculo que deveria ser feito pelas funções no MATLAB, percebi onde estava o erro.

Ao inserir a função no código dessas funções no MATLAB, eu havia inserido também a função peso envolvida em cada caso, o que é um erro. Ao inserir uma função $f(x)$ no código para fazermos a aproximação, devemos desconsiderar a função peso, uma vez que ela já está levada em conta nos valores tabelados de A_k e x_k.

Ao perceber isso, notei também que, no código da função de Gauss-Tchebyshev, eu havia cometido o mesmo erro, o que estava resultando em erros consideráveis nas aproximações.

Com relação à função da fórmula de Gauss-Legendre, esse erro não ocorreu porque a função peso envolvida é $w(x) = 1$.

7.11 Comparação das Aproximações

Além das aproximações detalhadas anteriormente, com cada função trabalhada com as fórmulas de quadratura foram feitas aproximações com as fórmulas de Newton-Cottes para a comparação dos resultados, como veremos a seguir.

7.11.1 Aproximação da Integral $\int_{-1}^{1}(x^3 + x^2 + x + 1)dx$

Sabemos que a integral em questão tem o seguinte valor numérico:

$$\begin{aligned}
\int_{-1}^{1}(x^3 + x^2 + x + 1)dx &= \left[\frac{x^4}{4} + \frac{x^3}{3} + \frac{x^2}{2} + x\right]_{-1}^{1} \\
&= \left(\frac{1^4}{4} + \frac{1^3}{3} + \frac{1^2}{2} + 1\right) - \left(\frac{(-1)^4}{4} + \frac{(-1)^3}{3} + \frac{(-1)^2}{2} - 1\right) \\
&= \left(\frac{\cancel{1}}{\cancel{4}} + \frac{1}{3} + \frac{\cancel{1}}{\cancel{2}} + 1\right) - \left(\frac{\cancel{1}}{\cancel{4}} - \frac{1}{3} + \frac{\cancel{1}}{\cancel{2}} - 1\right) \\
&= \frac{1}{3} + 1 + \frac{1}{3} + 1 \\
&= \frac{2}{3} + 2 \\
&= \frac{2}{3} + \frac{6}{3} \\
&= \frac{8}{3} \\
&= 2,666666666666666...
\end{aligned}$$

Com base nisso, usando 15 casas decimais para calcular o erro (que será escrito em módulo) em cada caso, temos os resultados:

Método Utilizado	Resultado	Erro
Regra do Trapézio	2.675925925925926	0.009259259259260
Regra 1/3 de Simpson	2.666666666666667	8.881784197001252e-16
Regra 3/8 de Simpson	2.666666666666667	8.881784197001252e-16
Fórmula de Gauss-Legendre	2.666666666459685	2.069810989269172e-10

Um fato interessante observado é que as regras de $\frac{1}{3}$ e $\frac{3}{8}$ de Simpson obtiveram o mesmo erro - o que deve-se ao fato de possuírem a mesma ordem de convergência. Além disso, essas regras tiveram um erro menor do que o uso da fórmula de quadratura de Gauss-Legendre com $n = 1$, que não deveria ter erro. Eu não sei explicar por que isso ocorreu.

7.11.2 Aproximação da Integral $\int_{-\pi/4}^{\pi/4} \left(\frac{sen(x)}{\sqrt{1-x^2}} \right) dx$

Método Utilizado	Resultado	Erro
Regra do Trapézio	2.906557081673860e-17	2.906557081673860e-17
Regra 1/3 de Simpson	-1.937704721115907e-17	1.937704721115907e-17
Regra 3/8 de Simpson	0	0
Fórmula de Gauss-Tchebyshev	0	0

Nesta aproximação, a Regra do Trapézio forneceu a pior aproximação, a Regra $\frac{1}{3}$ de Simpson obteve um erro pequeno e as demais forneceram o resultado exato. Nas três fórmulas de Newton-Cottes foram usados 13 nós na interpolação.

Na fórmula de Gauss-Tchebyshev, poderíamos usar o fato de que $2n + 1 = gr(f)$, onde n é o grau do polinômio de Tchebyshev que usaremos e $gr(f)$ é o grau do polinômio cuja integral será aproximada. Porém, neste caso temos $f(x) = sen(x)$, que não é um polinômio, então podemos simplesmente fixar $n = 2$ e fazer a aproximação, como foi feito.

7.12 Dúvidas atuais

Na subseção 7.11.2, coloquei o intervalo de integração $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ para não ter problemas com as descontinuidades da função em $x = -1$ e $x = 1$. Feito isso, obtive a seguinte integral para fazer a aproximação:

$$\int_{-\pi/4}^{\pi/4} \left(\frac{sen(x)}{\sqrt{1-x^2}} \right) dx \tag{7.1}$$

Veja que temos a função peso $w(x) = \frac{1}{\sqrt{1-x^2}}$, portanto devemos fazer a aproximação com a fórmula de Gauss-Tchebyshev. Porém, para usar essa fórmula, necessitamos que o intervalo de integração seja $[-1, 1]$, mas esse não é o caso. Sendo assim, faz-se necessária uma mudança de variável, conforme explicado em (FRANCO, 2006).

Porém, ao fazer a mudança de variável, a função peso envolvida deixa de ser $w(x) = \frac{1}{\sqrt{1-x^2}}$. Com isso, não sei o que fazer para usar a fórmula de Gauss-Tchebyshev. Em outras palavras, não sei aproximar a integral 7.1 usando fórmulas de quadratura.

7.13 Próximos estudos

Ao enviar este relatório ao Elías Gudiño, obtive algumas correções que podem otimizar os códigos no MATLAB com funções simbólicas, sugestões de implementação de aproximações com polinômios de grau diferente para obter melhores aproximações e uma explicação para a dúvida da seção 7.12.

Os próximos estudos serão:

1. Essas correções apontadas pelo Elías;
2. Continuar com as implementações em intervalos gerais;
3. Aproximar as ordens de convergência numérica de cada método e comparar com as estimativas.

Esses estudos constituem a parte principal das atividades a serem realizadas no mês de outubro.

Capítulo 8

Outubro de 2020

8.1 Introdução

Conforme explicado no fim do relatório do mês anterior, os principais estudos do mês de outubro são:

1. Efetuar algumas correções nos estudos de setembro;

2. Continuar com as implementações em intervalos gerais;

3. Aproximar as ordens de convergência numérica de cada método e comparar com as estimativas.

Cada uma dessas etapas será abordada em um capítulo a seguir.

8.2 Algumas correções

8.2.1 Otimizando os códigos

A primeira correção efetuada foi no código das fórmulas de Newton-Cottes no MATLAB. De início, estava sendo usado um laço "for"para preencher um vetor com as ordenadas da função a ser aproximada. Por otimização, essa parte do código foi substituída por uma função simbólica, conforme veremos a seguir:

Figura 8.1: Função da Regra do Trapézio com o comando "for".

```
%REGRA DO TRAPÉZIO GENERALIZADA
function integral = trapezio(a,b,n)
% a = início do intervalo de integração
% b = fim do intervalo de integração
% n = quantia de nós

format long
%encontrando as ordenadas
x = linspace(a,b,n);
for i=1:n
    %inserir a função aqui:
    f(i) = x(i)^3 + x(i)^2 + x(i) + 1;
end

%integral = (h/2)*[ f(x_0)+ 2(f(x_1)+f(x_2)+...+f(x_{n-1})) +f(x_n) ]

%somatorio
somatorio = 0;
for k=2:n-1
    somatorio = somatorio + f(k);
end

%espaçamento
h = (b-a)/(n-1);

%resultado final
integral = (h/2)*(f(1)+ 2*somatorio + f(n));
```

Figura 8.2: Função da Regra do Trapézio com a função simbólica.

```
%REGRA DO TRAPÉZIO GENERALIZADA
function integral = trapezio(a,b,n)
% a = início do intervalo de integração
% b = fim do intervalo de integração
% n = quantia de nós

format long
%encontrando as ordenadas
x = linspace(a,b,n);

%inserir a função aqui:
F = @(x) x.^3+x.^2+x+1
f = F(x)

%integral = (h/2)*[ f(x_0)+ 2(f(x_1)+f(x_2)+...+f(x_{n-1})) +f(x_n) ]

%somatorio
somatorio = 0;
for k=2:n-1
    somatorio = somatorio + f(k);
end

%espaçamento
h = (b-a)/(n-1);

%resultado final
integral = (h/2)*(f(1)+ 2*somatorio + f(n));
```

A mesma alteração foi feita nas funções das regras de $\frac{1}{3}$ e de $\frac{3}{8}$ de Simpson, otimizando o código com

uma função simbólica.

8.2.2 Melhorando a aproximação

Nos estudos do mês de setembro, a integral $\int_{-1}^{1}(x^3 + x^2 + x + 1)dx$ foi aproximada com o uso de diferentes métodos. Ao utilizar o método de Gauss-Legendre, o erro foi considerável. Isso ocorreu porque eu não estava utilizando os polinômios de Legendre com o grau adequado. Vale lembrar que o grau n dos polinômios de Legendre que oferecem o resultado exato da integral da função f é dado por:

$$2n - 1 = gr(f)$$

Como a função envolvida tem grau 3, temos $2n - 1 = 3 \Rightarrow n = 2$. Eu estava fazendo a aproximação com $n = 1$ e obtendo o resultado 2.666666666459685. Ao fazer a aproximação com $n = 2$, o resultado obtido foi 2.666666666328594.

O resultado exato da integral é 2.666..., porém, ao utilizar $n = 2$, o erro - que deveria ser zero, com exceção dos arredondamentos do MATLAB - aumentou, em vez de diminuir.

Após revisar o código da função no MATLAB diversas vezes e refazer os cálculos no papel, concluí que o aumento do erro ao utilizar $n = 2$ ocorreu por conta do número de casas decimais que estavam sendo utilizadas nos valores tabelados de A_i e x_i.

Tanto para $n = 1$ ($N = 2$), quanto para $n = 2$ ($N = 3$) eu estava utilizando os valores tabelados de x_i e A_i com 10 casas decimais. Porém, na tabela de $n = 1$, os valores $A_1 = A_2 = 1$, que estavam sendo utilizados, são exatos. Por outro lado, dentre os valores tabelados para $n = 2$, eu não estava utilizando nenhum valor exato, somente os números com 10 casas decimais, conforme fornecido em (FRANCO, 2006, p. 375). Isso estava causando um erro maior ao utilizar $n = 2$, o que não deveria ocorrer, afinal este valor deveria fornecer o resultado exato da integral em questão.

Este problema foi resolvido substituindo todos os valores tabelados com 10 casas decimais por seus respectivos valores exatos, como segue no *antes* e *depois*:

Figura 8.3: Função da fórmula de quadratura de Gauss-Legendre (ANTES).

```
%FÓRMULA DE GAUSS-LEGENDRE
%função peso w(x)=1
%intervalo de integração: [-1,1]
function integral = legendre(n)
%n é o grau do polinômio que usaremos
%para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato

%tabela 1 (gauss-legendre)
%x_i
xi = [-0.5773502691, 0.5773502691, 0, 0,                          %N=2
      -0.7745966692, 0.7745966692, 0, 0,                          %N=3
      -0.8611363115, 0.8611363115, -0.3399810435, 0.3399810435 %N=4
      ];

%A_i
ai = [1, 1, 0, 0,                                                 %N=2
      0.5555555555, 0.5555555555, 0.8888888888, 0,                %N=3
      0.3478548451, 0.3478548451, 0.6521451548, 0.6521451548 %N=4
     ];

format long
N = n+1;

%calculando f(x_i)
for i=1:N
    %inserir a função aqui:
    f(i) = xi(N-1,i)^3 + xi(N-1,i)^2 + xi(N-1,i) + 1;
end

%resultado final
integral = 0
for k=1:N
    integral = integral + f(k)*ai(N-1,k)
end
```

Conforme explicado, usando essa função, para $n = 1$ o resultado foi 2.666666666459685 e, com $n = 2$, o resultado obtido foi 2.666666666328594.

Observação: para as fórmulas de quadratura, não implementei as funções simbólicas no lugar do laço "for"porque preciso resgatar os valores de x_i das tabelas. Considerei mais fácil fazer isso com o "for".

Figura 8.4: Função da fórmula de quadratura de Gauss-Legendre (DEPOIS).

```
%FÓRMULA DE GAUSS-LEGENDRE
%função peso w(x)=1
%intervalo de integração: [-1,1]
function integral = legendre(n)
%n é o grau do polinômio que usaremos
%para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato

%tabela 1 (gauss-legendre)
%x_i
xi = [-1/sqrt(3), 1/sqrt(3), 0, 0,                                    %N=2
      -sqrt(15)/5, sqrt(15)/5, 0, 0,                                  %N=3
      -0.8611363115, 0.8611363115, -0.3399810435, 0.3399810435 %N=4
      ];

%A_i
ai = [1, 1, 0, 0,                                                   %N=2
      5/9, 5/9, 8/9, 0,                                             %N=3
      0.3478548451, 0.3478548451, 0.6521451548, 0.6521451548 %N=4
     ];
format long
N = n+1;

%calculando f(x_i)
for i=1:N
    %inserir a função aqui:
    f(i) = xi(N-1,i)^3 + xi(N-1,i)^2 + xi(N-1,i) + 1;
end

%resultado final
integral = 0;
for k=1:N
    integral = integral + f(k)*ai(N-1,k);
end
```

Feitas essas alterações, tanto para $n = 1$, quanto para $n = 2$, o resultado obtido foi 2.666666666666667. Ou seja, conseguimos 14 casas decimais de precisão, com o arredondamento na 15° casa, um resultado muito mais preciso do que o anterior, tendo em vista que o resultado exato da integral é $2,6666\ldots$.

8.3 Uso das fórmulas de quadratura de Gauss com mudança de variável

A minha principal dúvida nos estudos do mês de setembro surgiu com a tentativa de aproximação da seguinte integral:

$$\int_{-\pi/4}^{\pi/4} \left(\frac{sen(x)}{\sqrt{1-x^2}} \right) dx.$$

A minha intenção inicial era utilizar a fórmula de quadratura de Gauss-Tchebyshev por conta da função peso $w(x) = \dfrac{1}{\sqrt{1-x^2}}$. Para isso, deve ser feita uma mudança de variável para que o intervalo de integração passe a ser $[-1, 1]$. Eu não sabia como fazer a mudança de variável e prosseguir com o uso dos valores tabelados de x_k e A_k. Após estudar as notas enviadas Pelo Elías Gudiño e uma lista de vídeos sobre o assunto, consegui - finalmente - entender como deve-se fazer a mudança de variável e prosseguir. Porém, eu resolvi o problema utilizando a fórmula de quadratura de Gauss-Legendre, considerando a função peso $w(x) = 1$, ou seja, a função cuja integral será aproximada é $f(x) = \dfrac{sen(x)}{\sqrt{1-x^2}}$. Segue a explicação.

Queremos que o intervalo $[a, b]$ passe a ser $[-1, 1]$. Então, estabelecemos x como uma função linear de uma variável t, como segue:

$$x = c_1 \cdot t + c_0$$

Quando $t = -1$, queremos $x = a$; quando $t = 1$, queremos $x = b$, ou seja:

$$\begin{cases} a = c_0 - c_1 \\ b = c_0 + c_1 \end{cases}$$

Das igualdades acima, obtemos $c_1 = \dfrac{b-a}{2}$ e $c_0 = \dfrac{b+a}{2}$. Com isso, a mudança de variável procurada é:

$$x = c_1 \cdot t + c_0 \Rightarrow x = \left(\frac{b-a}{2}\right) \cdot t + \left(\frac{b+a}{2}\right) \tag{8.1}$$

Nesse caso, como $[a, b] = \left[-\dfrac{\pi}{4}, \dfrac{\pi}{4}\right]$, temos $a = -\dfrac{\pi}{4}$ e $b = \dfrac{\pi}{4}$. Substituindo em 8.1, encontramos a relação:

$$x = \frac{\pi \cdot t}{4}$$

Derivando ambos os lados em relação a t, obtemos que $dx = \dfrac{\pi \cdot dt}{4}$ e, portanto:

$$\int_{-\pi/4}^{\pi/4} \left(\frac{sen(x)}{\sqrt{1-x^2}}\right) dx = \frac{\pi}{4} \cdot \int_{-1}^{1} \left(\frac{sen\left(\dfrac{\pi \cdot t}{4}\right)}{\sqrt{1 - \left(\dfrac{\pi \cdot t}{4}\right)^2}} \right) dt = \frac{\pi}{4} \cdot \left[A_1 \cdot f(x_1) + A_2 \cdot f(x_2) + A_3 \cdot f(x_3)\right] \tag{8.2}$$

Recordemos o tabelamento para a fórmula de Gauss-Legendre:

Figura 8.5: Tabela da fórmula de quadratura de Gauss-Legendre.

TABELA 1 $\int_{-1}^{1} f(x)dx$

x_i	A_i
N = 2	
0.5773502691	(1)0.1000000000
N = 3	
0.7745966692	0.5555555555
0.0000000000	0.8888888888
N = 4	
0.8611363115	0.3478548451
0.3399810435	0.6521451548
N = 5	
0.9061798459	0.2369268850
0.5384693101	0.4786286704
0.0000000000	0.5688888888
N = 6	
0.9324695142	0.1713244923
0.6612093864	0.3607615730
0.2386191860	0.4679139345
N = 7	
0.9491079123	0.1294849661
0.7415311855	0.2797053914
0.4058451513	0.3818300505
0.0000000000	0.4179591836
N = 8	
0.9602898564	0.1012285362
0.7966664774	0.2223810344
0.5255324099	0.3137066458
0.1834346424	0.3626837833

Fonte: (FRANCO, 2006, p. 375).

Usaremos $N = 3$ para fazermos a aproximação. Os valores de x_i da tabela serão chamados de t_i neste relatório. Então temos a tabela:

i	t_i	x_i	$f(x_i)$	A_i	$A_i \cdot f(x_i)$
1	-0.774597	-0.608367	-0.720121	0.555556	-0.400068
2	0	0	0	0.888889	0
3	0.774597	-0.608367	0.720121	0.555556	0.400068

As colunas t_i e A_i foram preenchidas de acordo com a tabela 8.5. A coluna x_i é preenchida fazendo $x_i = \dfrac{\pi \cdot t_i}{4}$. A coluna $f(x_i)$ é preenchida usando $f(x_i) = \dfrac{sen(x_i)}{\sqrt{1 - x_i^2}}$.

Portanto, da equação 8.2, temos:

$$\begin{aligned}
\int_{-\pi/4}^{\pi/4} \left(\frac{sen(x)}{\sqrt{1-x^2}} \right) dx &\cong \frac{\pi}{4} \cdot \left[A_1 \cdot f(x_1) + A_2 \cdot f(x_2) + A_3 \cdot f(x_3) \right] \\
&\cong \frac{\pi}{4} \cdot (\cancel{-0.400068} + 0 + \cancel{0.400068}) \\
&\cong \frac{\pi}{4} \cdot 0 \\
&\cong 0.
\end{aligned}$$

Veja que, de fato, esse é o resultado da integral, pois a função $f(x)$ envolvida é ímpar e os extremos do intervalo de integração são opostos.

8.4 Como aproximar a ordem de convergência

Em conversa via Microsoft Teams com o Elías, ele me explicou como fazer a aproximação das ordens de convergência dos métodos estudados. Ao fazer uma aproximação, o erro é $e \cong c \cdot h^P$, onde c é uma constante, h é o espaçamento entre os nós e P é a ordem de convergência. Ao fazermos duas aproximações com uma quantidade diferente de nós, obtemos duas relações do tipo:

$$e_1 \cong c \cdot h_1^P \text{ e } e_2 \cong c \cdot h_2^P$$

Fazendo e_1/e_2, temos:

$$\begin{aligned}\frac{e_1}{e_2} &\cong \frac{c * h_1^P}{c * h_2^P} \\ &\cong \frac{h_1^P}{h_2^P} \\ &\cong \left(\frac{h_1}{h_2}\right)^P\end{aligned}$$

Aplicando o logaritmo de ambos os lados, temos:

$$\begin{aligned}log\left(\frac{e_1}{e_2}\right) &\cong log\left(\left(\frac{h_1}{h_2}\right)^P\right) \\ \Rightarrow log\left(\frac{e_1}{e_2}\right) &\cong P \cdot log\left(\frac{h_1}{h_2}\right) \\ \Rightarrow P &\cong \frac{log\left(\frac{e_1}{e_2}\right)}{log\left(\frac{h_1}{h_2}\right)}\end{aligned}$$

8.5 Aproximando a ordem de convergência na prática

Nesta parte do estudo, o objetivo era escolher uma função $f(x)$ qualquer cuja aproximação não seja exata, para que tenhamos erros nas aproximações. Para isso, foi necessário escolher uma função que não é um polinômio. Depois disso, ao aproximar $f(x)$ duas vezes com o mesmo método, usando um número de nós diferente em cada aproximação, temos os valores de e_1, e_2, h_1 e h_2 e podemos aproximar a ordem de convergência P.

A função usada para isso foi $f(x) = e^x \cdot cos(x)$, cuja integral deveria ser aproximada no intervalo $[-1, 1]$. Integrando $f(x)$ em $[-1, 1]$ por partes, temos que

$$\int_{-1}^{1} e^x \cdot cos(x)dx = \left[\frac{e^x \cdot (sen(x) + cos(x))}{2}\right]_{-1}^{1} \cong 1.93342149.$$

Depois, aproximei a integral em questão usando a fórmula de quadratura de Gauss-Legendre com $n_1 = 1$, obtendo o resultado 1.96297276 e o erro $e_1 \cong 0.29551264$; e com $n_2 = 2$, obtendo o resultado 1.93339046, com o erro $e_2 \cong 0.000031027$. Os valores de h_1 e h_2 são obtidos fazendo:

$$\begin{aligned}h_1 &= \frac{1 - (-1)}{n_1} = \frac{2}{1} = 2 \\ h_2 &= \frac{1 - (-1)}{n_2} = \frac{2}{2} = 1\end{aligned}$$

Assim, fazemos $P \cong \dfrac{log\left(\frac{e_1}{e_2}\right)}{log\left(\frac{h_1}{h_2}\right)}$ e obtemos $P \cong 9.89$.

Para conferir o resultado obtido, eu fiz novamente esse cálculo usando $n_3 = 3$ e $n_4 = 4$. Dessa vez, cheguei ao resultado $P \cong -2.25$. Veja que houve uma divergência entre os dois valores encontrados, o que indica algum erro cometido.

Ao informar o Elías sobre esse erro, recebi um material sobre o assunto. O próximo passo do estudo é estudar esse material sobre a aproximação das ordens de convergência.

Capítulo 9

Novembro de 2020

9.1 Introdução

No mês de novembro, a principal atividade realizada foi a aproximação das ordens de convergência numérica de cada método para comparar com as estimativas teóricas. Para tal, como recomendação do Elías Gudiño, meu orientador, foi estudado o texto [4]: Numerical Convergence Rates, escrito por Olof Runborg.

Ao longo deste relatório, descreverei brevemente os procedimentos usados para fazer a aproximação da ordem de convergência, levando em conta se conhecemos - ou não - o valor exato a ser aproximado. Em seguida, apresentarei as aproximações feitas com a fórmula de quadratura de Gauss-Legendre e a Regra do Trapézio e os resultados obtidos em cada caso.

Por fim, é descrita a discussão final dos resultados e os estudos a serem feitos no mês de dezembro.

9.2 Aproximação da ordem de convergência

Considere uma aproximação $\tilde{a}_h$ de um valor exato a, onde o parâmetro usado é h. Seja h tal que, à medida que h diminui, o erro $|a - \tilde{a}_h|$ também diminui. Nessas condições, podemos escrever

$$|a - \tilde{a}_h| \leq C \cdot h^p, \tag{9.1}$$

onde C é uma constante e p é a ordem de convergência do método numérico utilizado. Outra igualdade utilizada é a seguinte, cuja dedução não foi estudada:

$$a - \tilde{a}_h = Dh^p + O(h^{p+1}) \tag{9.2}$$

onde D é uma constante e $O(h^{p+1})$ é o erro na aproximação da ordem de convergência.

Note que, à medida que h diminui, se p for grande, então o erro diminuirá rapidamente e as aproximações tenderão rapidamente para o valor exato. Sendo assim, encontrar a ordem de convergência é útil para testar a eficiência de um determinado método numérico.

Os estudos acerca desse tema ocorreram basicamente em dois casos: quando o valor exato que estamos aproximando é conhecido e quando não é. Em cada caso, temos procedimentos diferentes para encontrar a ordem de convergência, como veremos adiante.

9.2.1 Quando o valor exato é conhecido

Este é o caso mais simples. Se conhecemos o valor exato a que estamos aproximando, podemos tomar dois caminhos principais para aproximar a ordem de convergência:

- Plotar $|a - \tilde{a}_h|$ como função de h na escala logarítmica nos dois eixos e calcular a inclinação da reta;
- Dividir h por 2 e considerar a razão dos erros $a - \tilde{a}_h$ e $a - \tilde{a}_{h/2}$. Da equação 9.2, temos que $log_2\left(\frac{a - \tilde{a}_h}{a - \tilde{a}_{h/2}}\right) = p + O(h)$, onde $O(h)$ é o erro em função de h. Na prática, desconsideramos $O(h)$.

9.2.2 Quando o valor exato é desconhecido

Neste caso, também temos duas opções para aproximar a ordem de convergência:

- Tomar uma solução numérica k com h muito pequeno e usá-lo como referência. Ou seja, assumir $k = a$ como o valor exato para calcular $|\tilde{a}_h - k|$ e prosseguir para o caso onde o valor exato é conhecido;
- Outro jeito comum de aproximar a ordem de convergência quando desconhecemos o valor exato é dividindo h pela metade sucessivamente, obtendo $\frac{h}{2}$ e $\frac{h}{4}$ e, em seguida, calculando $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = p + O(h)$, onde $O(h)$ é o erro em função de h. Essa igualdade é consequência da equação 9.2. Na prática, desconsideramos o valor de $O(h)$. Assim, obtemos uma aproximação para p.

9.3 Aproximando a ordem de convergência na prática

Para aplicar os procedimentos apresentados na seção anterior, tomei como exercício aproximar a integral $\int_{-1}^{1} e^x cos(x) dx$ usando dois métodos diferentes: a fórmula de quadratura de Gauss-Legendre e a Regra do Trapézio. Em seguida, foram feitas tentativas de aproximação para as ordens de convergência dos métodos utilizados. Nessa parte do estudo, os métodos foram implementados na linguagem de programação Julia.

Note, primeiramente, que essa integral pode ser calculada analiticamente, como segue:

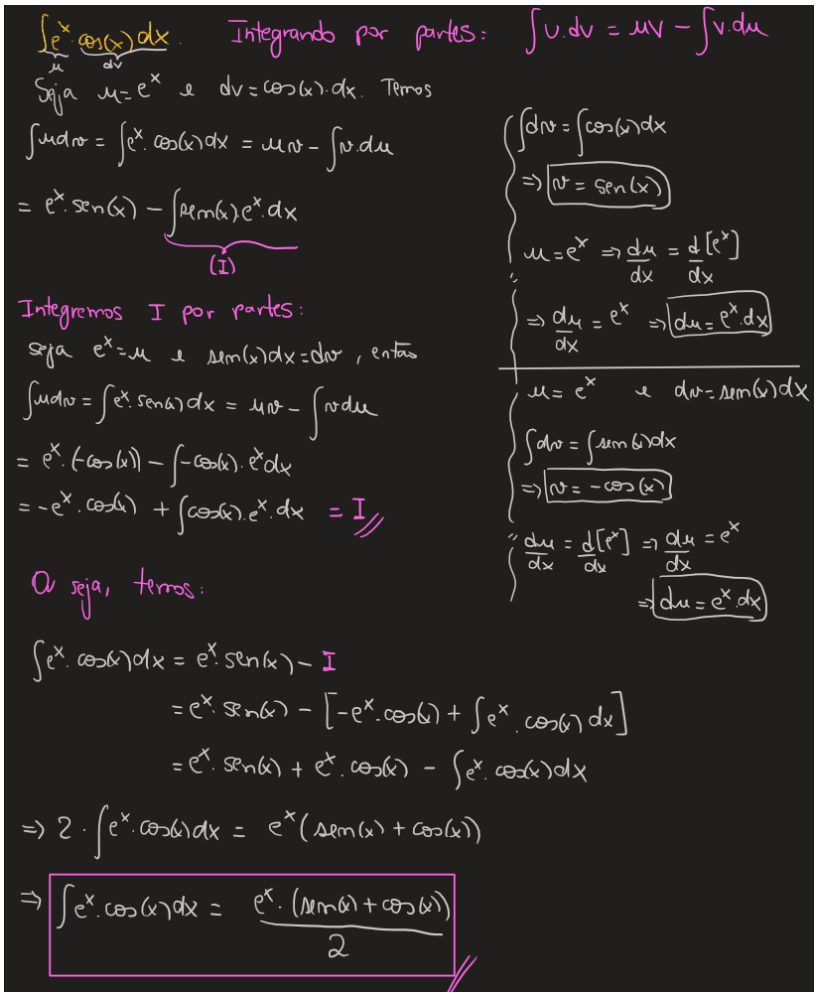

Figura 9.1: Integração por partes de $\int_{-1}^{1} e^x cos(x) dx$.

Com isso, podemos obter $\int_{-1}^{1} e^x cos(x) dx = 1.9334214962$, truncando o resultado com 10 casas decimais de precisão.

9.3.1 Aproximando com a fórmula de quadratura de Gauss-Legendre

Inicialmente, foi feita a aproximação da integral em questão com a fórmula de quadratura de Gauss-Legendre. O código utilizado, em Julia, desse método, foi seguinte:

```
function legendre(F, g=7)
    # FÓRMULA DE GAUSS-LEGENDRE
    # função peso w(x)=1
    # intervalo de integração: [-1,1]
    # n é o grau do polinômio que usaremos
    # para integrar um polinômio f(x), fazemos 2n+1=gr(f) => resultado exato

    # tabela 1 (gauss-legendre)
    # valores tabelados de x_i
    xi = [-1/sqrt(3) 1/sqrt(3) 0 0 0 0 0 0;                                                                    #N=2
          -sqrt(15)/5 sqrt(15)/5 0 0 0 0 0 0;                                                                  #N=3
          -0.8611363115 0.8611363115 -0.3399810435 0.3399810435 0 0 0 0;                                       #N=4
          -0.9061798459 0.9061798459 -0.5384693101 0.5384693101 0 0 0 0;                                       #N=5
          -0.9324695142 0.9324695142 -0.6612093864 0.6612093864 -0.2386191860 0.2386191860 0 0;                #N=6
          -0.9491079123 0.9491079123 -0.7415311855 0.7415311855 -0.4058451513 0.4058451513 0 0;                #N=7
          -0.9602898564 0.9602898564 -0.7966664774 0.7966664774 -0.5255324099 0.5255324099 -0.1834346424 0.1834346424] #N=8
    # valores tabelados de A_i
    ai = [1 1 0 0 0 0 0 0;                                                                                   #N=2
          5/9 5/9 8/9 0 0 0 0 0;                                                                             #N=3
          0.3478548451 0.3478548451 0.6521451548 0.6521451548 0 0 0 0;                                       #N=4
          0.2369268850 0.2369268850 0.4786286704 0.4786286704 0.5688888888 0 0 0;                            #N=5
          0.1713244923 0.1713244923 0.3607615730 0.3607615730 0.4679139345 0.4679139345 0 0;                 #N=6
          0.1294849661 0.1294849661 0.2797053914 0.2797053914 0.3818300505 0.3818300505 0.4179591836 0;      #N=7
          0.1012285362 0.1012285362 0.2223810344 0.2223810344 0.3137066458 0.3137066458 0.3626837833 0.3626837833] #N=8
    resultado = []
    for n in 1:g
        N = n+1
        # calculando f(x_i)
        f = []
        for i=1:N
            # inserir a função aqui:
            # F(x) = big(exp(x))*big(cos(x))
            push!(f, F(xi[N-1,i]))
        end

        # resultado final
        integral = 0
        for k=1:N
            integral = integral + f[k]*ai[N-1,k]
        end
        push!(resultado, integral)
    end
    return resultado
end
```

Figura 9.2: Função em Julia da fórmula de Gauss-Legendre.

Para chamar a função, fazemos $legendre(F, g)$, onde F é a função cuja integral queremos aproximar no intervalo $[-1, 1]$ e g é o grau dos polinômios de Legendre que queremos utilizar. Note que, se não dermos entrada num valor para g, ele será 7 automaticamente. Sendo assim, temos:

```
F(x) = big(exp(x))*big(cos(x))
legendre(F)
```

```
7-element Array{Any,1}:
 1.962972760754352415982527741361710432165994827112534637973760131418599073072073
 1.933390469264297553827662822627783526892652283092323355313139043982598299158828
 1.933416894001862490538171269856099849831646090549359255486881415826896942133 04
 1.933421496893523414284129878157528071966346117399708343077246364935648798453 06
 1.933421495932823806461406948450065430869969896374866673089787723963212806973 18
 1.933421495852347990669078320395849925124677289421036893580554208268746084805658
 1.933421495650374293269962300421844514706103866347149412979742715223463792531678
```

Figura 9.3: Resultado das aproximações com Gauss-Legendre.

Marcadas em verde estão as casas decimais corretas. Note que obtemos ótimas aproximações, com 8 e 9 casas decimais de precisão.

Feito isso, o próximo passo foi plotar alguns gráficos para visualizar melhor as aproximações, o erro de cada aproximação e a taxa de decréscimo do erro. Para isso, criei a função $graficos()$:

```
function graficos()
    # insira a função aqui
    f(x) = big(exp(x))*big(cos(x))
    # insira a função do método aqui
    γ = legendre(f)
    layout = grid(2, 2)
    p = plot(layout=layout, leg=false)
    plot!(p[1], γ, lw=1, l=:dash, marker='.', ms=3)
    valor_exato = (big(ℯ^2)*big(sin(1)) + big(ℯ^2)*big(cos(1)) + big(sin(1)) - big(cos(1)))/big((2*ℯ))
    a = abs.(γ .- valor_exato)
    plot!(p[2], a, lw=1, l=:dash, marker='.', ms=3)
    plot!(p[3], a, lw=1, l=:dash, marker='.', ms=3, yaxis=:log)
    b = []
    for i in 2:length(γ)
        push!(b, abs(γ[i] - valor_exato)/(abs(γ[i-1] - valor_exato)))
    end
    plot!(p[4], b, lw=1, l=:dash, marker='.', ms=3)
end
```

Figura 9.4: Função para plotar os gráficos das aproximações.

Essa função não recebe entrada alguma, a função cuja integral será aproximada e o método a ser utilizado estão inseridos nela, nas primeiras linhas.

No primeiro gráfico, temos os valores das aproximações; no segundo, temos o erro de cada aproximação; no terceiro, também temos o erro, mas com a escala *log* no eixo y; no último gráfico, temos a taxa de decréscimo do erro.

Plotando isso, temos:

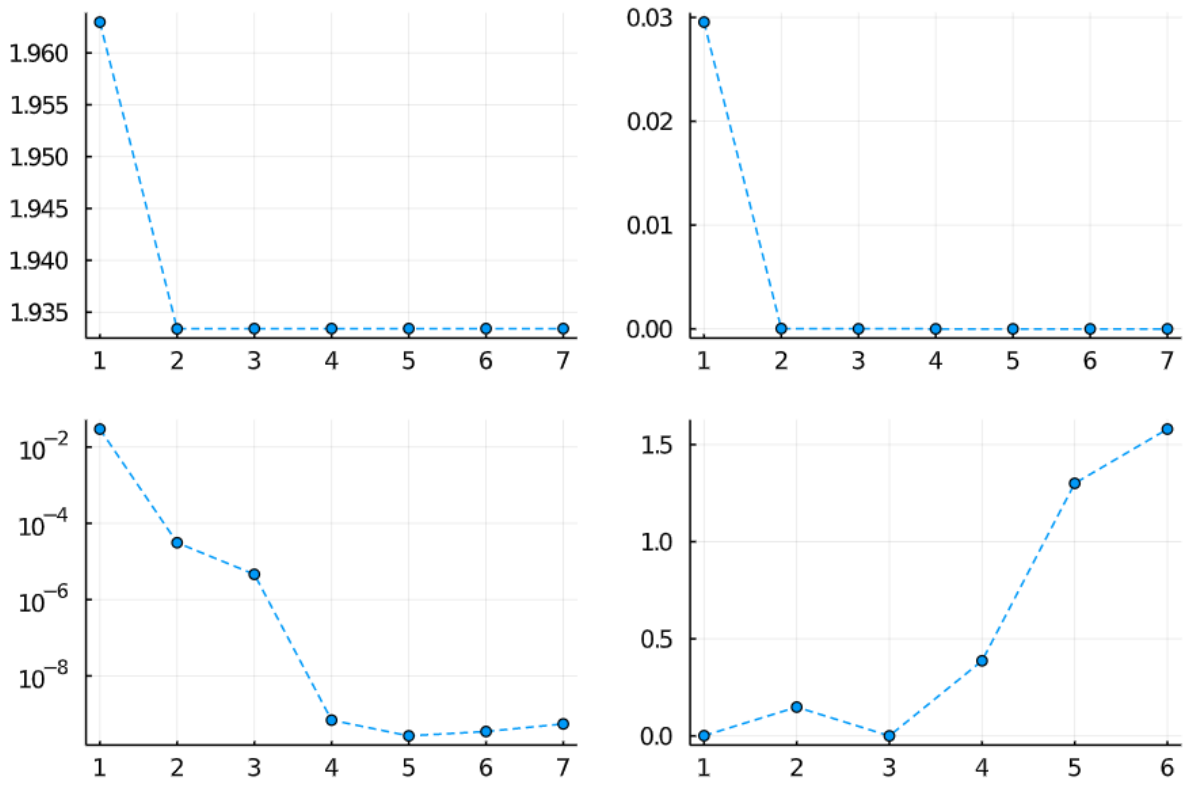

Figura 9.5: Gráficos das aproximações com Gauss-Legendre.

Note que, de acordo com o último gráfico, a taxa de decréscimo do erro varia muito. Isso se tornou um empecilho para que eu encontrasse a ordem de convergência desse método. Além disso, na fórmula de quadratura de Gauss-Legendre, não temos um parâmetro explícito h tal que, à medida que h diminui, o erro também diminui. Em vista disso, eu não soube prosseguir para aproximar a ordem de convergência neste caso.

Feito isso, resolvi fazer o mesmo com a Regra do Trapézio e tentar aproximar sua ordem de convergência.

9.3.2 Aproximando com a Regra do Trapézio

Todo o procedimento explicado na seção anterior foi repetido com a Regra do Trapézio. Primeiro, foi criada a função para aplicar a Regra do Trapézio:

```
function trapezio(f,a,b,n=129)
    # f é a função
    # a = início do intervalo de integração
    # b = fim do intervalo de integração
    # n = quantia de nós
    # integral = (h/2)*[f(x_1) + 2(f(x_2) + f(x_3) + ... + f(x_{n-1})) + f(x_n)]
    resultado = []

    if n ≤ 1
        error("n deve ser > 1")
    end

    for i in 2:n
        # 1º - Calcular o espaçamento h e os valores x_1, ..., x_n
        x = LinRange(a,b,i)
        h = big((b-a)/(i-1))

        # 2º Calcular o somatório f(x_2) + ... + f(x_{n-1})
        S = 0
        for j in 2:i-1
            S = S + big(f(x[j]))
        end

        # 3º Calcular a integral e colocar no vetor resultado
        integral = (h/2)*(big(f(x[1])) + 2*S + big(f(x[i])))
        push!(resultado, integral)
    end
    return resultado
end
```

Figura 9.6: Função da Regra do Trapézio.

Veja que a função fará uma aproximação para cada número de nós i de 2 até n. Caso não dermos entrada no valor de n, o número de nós utilizados será automaticamente 129. Aproximando $\int_{-1}^{1} e^x cos(x)dx$ com a Regra do Trapézio usando a função acima, temos:

```
f(x)=big(exp(x))*big(cos(x))
trapezio(f,-1,1)
```

```
128-element Array{Any,1}:
 1.667460050262298173305157468786320259260146809614810302834983501973908914806088
 1.833730025131149086652578734393160129630073404807405151417491750986954457403044
 1.886409946505406765631265078143082704471422482783120142983804480111963270082751
 1.906449895965494548884343266597916054520122410551236832098134267587941081956 16
 1.916004009413438766621933886859037572878425059190363017375925424121201460690855
 1.921267421105700222257455657929294983850056944719064061826752141422516098393686
 1.924466044211944943720556356329731169062475574685430602864374904374070869944 24
 1.926552104234469216226791470897678847845982219787476481199093767369967533653607
 1.927986858093729682783825578055940548155457449989276342502392533385992044124921
 1.929015402226933404486230042973871731963149025437092517158000391652541173358418
 1.929777628270960754900790546933737632869035796563404233673517910791461466171646
 1.930358057996057455127373505633423763851992415763697728451671300248512628548812
 1.930810183787190384399227595867150570867834601193651841861508971206109729947972
 ⋮
 1.933389184376881026545977237963541370220137283159756395458618871846761161148752
 1.933389729699109374915075102894134728469018316218300564032270426007685755942507
 1.933390261332066658666191540791575947953940258782906426627500242836211890998873
 1.933390779730134717650629148421469064748608054864275490724079097634976213262877
 1.933391285328997934069229153190536850412000795451839404038024089057355841420886
 1.933391778546559327121757446434667473031945388233591226298643727143815836316113
 1.933392259783804444264624208048457858425402776631922836078847077138518428458228
 1.933392729425615717044659926566717569536810307532672961855343712252376936581569
 1.933393187841544046997508141449293269086482472323043296871839882602170979059989
 1.933393635386533537185413914106159304297478224865183440350452808596826103884445
 1.933394072401611017393509800706416665503275805820392695328887840689881035221878
 1.933394499214534030418524956697424611670573665830556905302006587628907375783771
```

Figura 9.7: Resultados das aproximações com a Regra do Trapézio.

Veja que, usando 129 nós, obtemos apenas 3 casas decimais de precisão, o que é péssimo em comparação com as aproximações feitas com Gauss-Legendre, onde, com um polinômio de Legendre de grau acima de 3, já conseguimos 8 casas decimais de precisão.

Feito isso, foram plotados os gráficos dessas aproximações feitas:

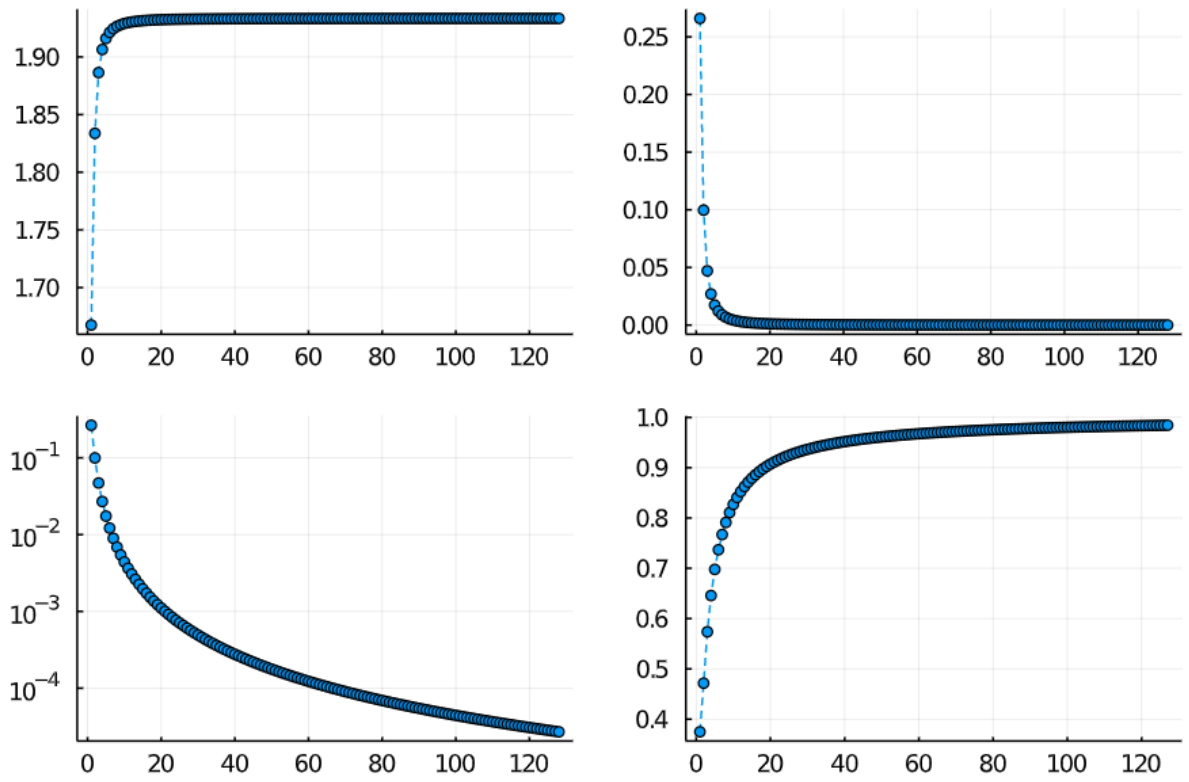

Figura 9.8: Gráficos das aproximações com a Regra do Trapézio.

No primeiro gráfico, vemos o valor das aproximações. Podemos notar que, dos 20 nós em diante, as aproximações se mantém com um valor muito próximo, isso indica que não é interessante estender tanto o número de nós com esse método, pois não são obtidos resultados interessantes à medida que aumentamos muito a quantia de nós.

No segundo gráfico, vemos o valor do erro em cada aproximação e, no terceiro, o erro com a escala *log* no eixo y.

No último gráfico, vemos a taxa de decréscimo do erro. Note que o valor da taxa de decréscimo se mantém muito próximo de 1, isso mostra que o erro diminui muito pouco à medida que acrescentamos mais nós.

Depois de tiradas essas conclusões, o próximo passo foi tentar aproximar a ordem de convergência desse método. Para fazer isso, prosseguimos da segunda maneira apresentada na seção 9.2.1, quando o valor exato é conhecido.

Note que h é o espaçamento dos nós na Regra do Trapézio. Sabemos que h é tal que, à medida que h diminui, o erro também diminui, então h será o parâmetro que usaremos nessa aproximação da ordem de convergência.

Sendo assim, dividimos h por dois sucessivamente, determinamos quantos nós são necessários para obter os espaçamentos obtidos e prosseguimos para aproximar a ordem de convergência. Para tal, temos a seguinte função:

```
function ordem_trapezio()
    γ = trapezio(f,-1,1)
    valor_exato = (big(e^2)*big(sin(1)) + big(e^2)*big(cos(1)) + big(sin(1)) - big(cos(1)))/big((2*e))
    b = []
    v = [2,3,5,9,17,33,65,129]
    for i in 1 : length(v) - 1
        d_1 = γ[v[i]-1] - valor_exato
        d_2 = γ[v[i+1]-1] - valor_exato
        push!(b, log2(d_1/d_2))
    end
    return b
end
```

Figura 9.9: Função para aproximar a ordem de convergência da Regra do Trapézio.

Chamando essa função, obtemos:

```
ordem_trapezio()
```

```
7-element Array{Any,1}:
 1.4156751384845514714241386920780154486442364669383722192415136454992121045104 55
 1.8860289628837468906468051646239662857120472677912230690203624662292913434089 82
 1.9731868078985546633851439250852777214497583422709317954330770457918325781605 25
 1.9933978365347287971774490130020719266287291368974682510411243725749251475489 82
 1.9983557233682135410648751200662079921010253941427957953641553496852096751153 43
 1.9995893214810144879644607641447411613010961483584902066903193319146001180470 13
 1.9998973547637337578577773958327283803131369826061679670071814843078203698360 36
```

Figura 9.10: Aproximações para a ordem de convergência da Regra do Trapézio.

A última aproximação para a ordem de convergência foi feita calculando $log_2\left(\frac{a-\tilde{a}_{h/64}}{a-\tilde{a}_{h/128}}\right)$, onde $a = \int_{-1}^{1} e^x cos(x)dx = 1.9334214962$.

Veja que as aproximações indicam que a ordem de convergência da Regra do Trapézio é 2, o que é verdade, pois esse método é quadrático. Sendo assim, aproximamos com sucesso a ordem de convergência do método utilizado.

9.4 Discussão dos resultados obtidos e próximos estudos

Terminados os estudos listados acima, os resultados obtidos foram apresentados ao meu orientador - Elías Gudiño - em uma conversa via Microsoft Teams. Fui orientado a usar melhor a escala *log* no eixo y para melhorar a visualização de alguns gráficos, como poderia ser feito no primeiro gráfico da figura 4.5 para a melhor visualização da diferença entre as aproximações.

Nessa conversa, Elías indicou duas novas atividades:

- Comparar o erro das aproximações feitas com as fórmulas de quadratura de Gauss, usando alguma integral;
- Aproximar as ordens de convergência das demais fórmulas de Newton-Cottes: a Regra $\frac{1}{3}$ de Simpson e a Regra $\frac{3}{8}$ de Simpson, usando alguma integral. Uma sugestão para essa integral foi $\int_{-1}^{1} e^{x^2} dx$, que não possui solução analítica, somente numérica.

Essas duas novas atividades constituem o início dos estudos a serem feitos no mês de dezembro.

Capítulo 10

Dezembro de 2020

10.1 Introdução

No mês de dezembro, foram realizadas, essencialmente, duas atividades indicadas pelo orientador Elías Gudiño:

- Comparar o erro das aproximações feitas com as fórmulas de quadratura de Gauss de modo repetido - dividindo o intervalo de integração em subintervalos menores;
- Aproximar as ordens de convergência das demais fórmulas de Newton-Cottes: a Regra $\frac{1}{3}$ de Simpson e a Regra $\frac{3}{8}$ de Simpson, usando alguma integral. Uma sugestão para essa integral foi $\int_{-1}^{1} e^{x^2} dx$, que não possui primitiva envolvendo funções elementares.

Neste relatório, cada atividade será descrita em um capítulo separado.

10.2 Comparando o erro com as fórmulas de quadratura de Gauss repetidas

No relatório das atividades do mês de novembro, foi apresentado o cálculo analítico da integral $\int_{-1}^{1} e^x cos(x) dx$. Com ele, obtemos que $\int_{-1}^{1} e^x cos(x) dx = 1.9334214962$, truncando o resultado com 10 casas decimais de precisão.

A imagem abaixo mostra as aproximações para essa integral feitas com a fórmula de quadratura de Gauss Legendre. O primeiro resultado foi obtido usando essa fórmula de quadratura com 1 ponto, depois com 2 pontos e assim sucessivamente, de modo que a última aproximação foi obtida com a fórmula de quadratura de Gauss-Legendre de 7 pontos.

Os dígitos marcados em verde estão corretos.

```
F(x) = big(exp(x))*big(cos(x))
legendre(F)
```

```
7-element Array{Any,1}:
 1.962972760754352415982527741361710432165994827112534637973760131418599073072073
 1.933390469264297553827662822627783526892652283092323355313139043982598299158828
 1.933416894001862490538171269856099849831646090549359255486881415826896942133 04
 1.933421496893523414284129878157528071966346117399708343077246364935648798453 06
 1.933421495932823806461406948450065430869969896374866673089787723963212806973 18
 1.933421495852347990669078320395849925124677289421036893580554208268746084805658
 1.933421495650374293269962300421844514706103866347149412979742715223463792531678
```

Figura 10.1: Aproximações feitas com a Fórmula de quadratura de Gauss-Legendre

Nesta parte do estudo, o intuito é mostrar como o erro varia ao dividirmos o intervalo de integração repetidas vezes. Sabemos que a precisão da aproximação melhora à medida que usamos mais pontos,

como vemos na imagem acima. Usar um número grande de pontos, porém, pode tornar-se caro computacionalmente. Em vista disso, uma alternativa é manter sempre o mesmo número de pontos, mas dividir o intervalo de integração em subintervalos menores e aproximar a integral em cada subintervalo separadamente, como faremos a seguir.

10.2.1 Aproximando $\int_{-1}^{1} e^x cos(x)dx$ com a fórmula de quadratura de Gauss-Legendre de 3 pontos repetida

Com 2 subintervalos

Primeiro, usamos somente o intervalo original $[-1, 1]$ e fazemos uma aproximação. Fazendo isso, obtemos o resultado 1.9334168940 e o erro 4.6021988511e-06.

Em seguida, dividimos esse intervalo em dois e aproximamos a integral da função em cada um separadamente. Ou seja, fazemos

$$\int_{-1}^{1} e^x cos(x)dx = \int_{-1}^{0} e^x cos(x)dx + \int_{0}^{1} e^x cos(x)dx \tag{10.1}$$

e aproximaremos as duas integrais da direita usando a fórmula de quadratura de Gauss-Legendre de 3 pontos.

Vamos aproximar, primeiro, a integral $\int_{-1}^{0} e^x cos(x)dx$.

Para fazer isso, precisamos fazer a mudança de variável:

$$x = \frac{(b-a)\cdot t + (b+a)}{2}$$

Neste caso, temos $[a, b] = [-1, 0]$ e, com isso, $x = \dfrac{t-1}{2}$. Derivando em ambos os lados em relação a t, temos $dx = \dfrac{dt}{2}$. Portanto, queremos aproximar a integral:

$$\int_{-1}^{0} e^x cos(x)dx = \frac{1}{2} \cdot \int_{-1}^{1} e^{\frac{t-1}{2}} cos\left(\frac{t-1}{2}\right) dt$$

Calculando isso numericamente usando a fórmula de quadratura de Gauss-Legendre com 3 pontos, temos:

$$\frac{1}{2} \cdot \int_{-1}^{1} e^{\frac{t-1}{2}} cos\left(\frac{t-1}{2}\right) dt \cong 0.5553968777\ldots$$

Agora, aproximaremos a integral $\int_{0}^{1} e^x cos(x)dx$. Fazendo a mesma mudança de variável, mas desta vez com $[a, b] = [0, 1]$, obtemos $x = \dfrac{t+1}{2}$. Derivando de ambos os lados em relação a t, temos $dx = \dfrac{dt}{2}$. Então, queremos aproximar a integral:

$$\int_{0}^{1} e^x cos(x)dx = \frac{1}{2} \cdot \int_{-1}^{1} e^{\frac{t+1}{2}} cos\left(\frac{t+1}{2}\right) dt$$

Aproximando isso com a mesma fórmula de quadratura de 3 pontos:

$$\frac{1}{2} \cdot \int_{-1}^{1} e^{\frac{t+1}{2}} cos\left(\frac{t+1}{2}\right) dt \cong 1.3780246004\ldots$$

Por último, efetuando a soma 10.1, temos:

$$\begin{aligned}\int_{-1}^{1} e^x cos(x)dx &\cong 0.5553968777 + 1.3780246004 \\ &\cong 1.9334214782\ldots\end{aligned}$$

Disso, conseguimos o erro 1.7931950565e-08.

Com 4 subintervalos

Desta vez, dividiremos o intervalo [-1,1] em quatro subintervalos menores. Com isso, teremos que aproximar quatro integrais e somar os quatro resultados. Nas quatro integrais, faremos um procedimento idêntico ao mostrado na subseção anterior, fazendo a mudança de variável $x = \frac{(b-a)\cdot t + (b+a)}{2}$. A partir disso, temos:

$$\begin{aligned}\int_{-1}^{1} e^x cos(x)dx &= \int_{-1}^{-0.5} e^x cos(x)dx + \int_{-0.5}^{0} e^x cos(x)dx + \int_{0}^{0.5} e^x cos(x)dx + \int_{0.5}^{1} e^x cos(x)dx \\ &= \frac{1}{4}\cdot\int_{-1}^{1} e^{\frac{0.5t-1.5}{2}} cos\left(\frac{0.5t-1.5}{2}\right)dt + \frac{1}{4}\cdot\int_{-1}^{1} e^{\frac{0.5t-0.5}{2}} cos\left(\frac{0.5t-0.5}{2}\right)dt \\ &+ \frac{1}{4}\cdot\int_{-1}^{1} e^{\frac{0.5t+0.5}{2}} cos\left(\frac{0.5t+0.5}{2}\right)dt + \frac{1}{4}\cdot\int_{-1}^{1} e^{\frac{0.5t+1.5}{2}} cos\left(\frac{0.5t+1.5}{2}\right)dt \\ &\cong 1.9334214959\end{aligned}$$

Com essa aproximação, temos o erro 2.5785578127e-10, que é menor que os dois erros anteriores, obtidos com um e dois intervalos.

10.2.2 Interpretação dos resultados

Agora, para melhorar a visualização dos resultados obtidos, temos um gráfico com os erros das três aproximações feitas:

Figura 10.2: Erros das aproximações feitas.

Note que, ao aproximar a integral $\int_{-1}^{1} e^x cos(x)dx$ considerando apenas o intervalo [-1,1], o erro foi 4.60×10^{-6}. Depois de dividir o intervalo original em dois subintervalos [-1,0] e [0,1], o erro caiu para 1.79×10^{-8}. Por último, dividindo o intervalo inicial em quatro partes iguais, o erro foi menor ainda: 2.57×10^{-10}.

Vale lembrar que todas as aproximações foram feitas usando a fórmula de quadratura de Gauss-Legendre de 3 pontos. Ou seja, a melhora nas aproximações ocorreu exclusivamente com as divisões repetidas do intervalo de integração.

10.3 Aproximando a ordem de convergência das fórmulas de Newton-Cottes

A aproximação da ordem de convergência da Regra do Trapézio já foi feita nos estudos do mês de novembro. Sendo assim, neste capítulo, trabalharemos com a Regra $\frac{1}{3}$ de Simpson e a Regra $\frac{3}{8}$ de Simpson. Para fazer isso, usaremos a integral $\int_{-1}^{1} e^{x^2} dx$ e vamos prosseguir da seguinte maneira para aproximar a ordem de convergência:

Seja $\tilde{a}_h$ uma aproximação da integral feita com o parâmetro h. Vamos dividir h pela metade sucessivamente, obtendo $\frac{h}{2}$ e $\frac{h}{4}$ e, em seguida, calcular $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = p + O(h)$, onde $O(h)$ é ordem de h. Na prática, desconsideramos o valor de $O(h)$. Assim, obtemos uma aproximação para p.

À medida que reduzimos o parâmetro h, obtemos aproximações melhores para a integral em questão e, consequentemente, melhoramos a aproximação para a ordem de convergência do método utilizado.

Faremos isso com a Regra $\frac{1}{3}$ de Simpson e com a Regra $\frac{3}{8}$ de Simpson. Em ambos os casos, utilizaremos a integral $\int_{-1}^{1} e^{x^2} dx$, que não possui primitiva envolvendo funções elementares.

10.3.1 Ordem de convergência da Regra $\frac{1}{3}$ de Simpson

Primeiro, foi feita uma função para aproximar integrais com a Regra $\frac{1}{3}$ de Simpson, como segue:

```
function simpson13simples(f,a,b,n=129)
    # n deve ser ÍMPAR maior do que 2 para termos uma quantia PAR de sub-intervalos
    if n % 2 == 0 || n < 3
        error("n deve ser um ímpar ≥ 3")
    end
    # integral = (h/3)*[ f(x_1) + 4f(x_2) + 2f(x_3) + 4f(x_4) +...+ 4f(x_{2k}) + f(x_{2k+1})]
    # encontramos os pontos x_1, ..., x_n e calculamos as ordenadas nesses pontos
    x = LinRange(a,b,n)
    F = f.(x)

    # calculamos a soma central S = 4f(x_2) + 2f(x_3) + 4f(x_4) +...+ 4f(x_{2k})
    S = 0
    for i=2:n-1
        if i%2 == 0
            S = S + 4*F[i]
        else
            S = S + 2*F[i]
        end
    end

    # calculamos o espaçamento h e o resultado da aproximação
    h = (b-a)/(n-1)
    integral = (h/3)*(F[1] + S + F[n])
    return integral
end
```

Figura 10.3: Função da Regra $\frac{1}{3}$ de Simpson.

Agora, sabendo que nesta regra o número de pontos utilizados deve ser ímpar para termos uma quantia par de sub-intervalos, o menor número possível de pontos que podemos utilizar é 3, pois com somente um ponto não conseguimos formar intervalos. Sendo assim, para um número de pontos $n = 3$, temos o maior espaçamento possível entre os pontos, que chamaremos de h. Agora, para obter $\frac{h}{2}$ precisamos que $n = 5$. De forma geral, se queremos um espaçamento $\frac{h}{k}$, tomamos um número de pontos $n = 2k + 1$. Fazendo isso, obtemos os valores do vetor v, presente na função a seguir:

A função acima toma as aproximações feitas com h, $\frac{h}{2}$ e $\frac{h}{4}$ e calcula $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right)$.

Em seguida, toma as aproximações feitas com $\frac{h}{2}$, $\frac{h}{4}$ e $\frac{h}{8}$ e calcula $log_2\left(\frac{\tilde{a}_{h/2} - \tilde{a}_{h/4}}{\tilde{a}_{h/4} - \tilde{a}_{h/8}}\right)$, e assim sucessivamente, até que o último espaçamento seja $\frac{h}{128}$.

```
function ordem_simpson13()
    v = [2,3,5,9,17,33,65,129,257]
    w = []
    aproximacoes = []
    for k in v
        push!(w, simpson13simples(x->exp(x.^2), -1,1,k))
    end
    for i in 1 : length(w) - 2
        push!(aproximacoes, log2((w[i] - w[i+1])/(w[i+1] - w[i+2])))
    end
    return aproximacoes
end
```

Figura 10.4: Função para aproximar a ordem de convergência da Regra $\frac{1}{3}$ de Simpson.

Os resultados obtidos são as aproximações para a ordem de convergência da Regra $\frac{1}{3}$ de Simpson. À medida que diminuímos o valor do espaçamento obtemos melhores aproximações, como vemos:

```
ordem_simpson13()
```

```
7-element Array{Any,1}:
 1.3030843385673432
 3.013015731132045
 3.605716321443637
 3.8793151480629713
 3.9678036555599348
 3.991807580666992
 3.9979426292869875
```

Figura 10.5: Aproximações para a ordem de convergência da Regra $\frac{1}{3}$ de Simpson.

Como podemos ver, os resultados se aproximam cada vez mais de 4. Ou seja, concluímos que a ordem de convergência dessa regra é 4.

10.3.2 Ordem de convergência da Regra $\frac{3}{8}$ de Simpson

Os mesmos procedimentos apresentados na seção anterior foram feitos com a Regra $\frac{3}{8}$ de Simpson. Primeiro, foi feita a função para aproximar integrais usando essa regra:

```
function simpson38(f,a,b,n=130)
    # n deve ser da forma n = 3k+1, com k natural, para termos uma quantia 3k sub-intervalos
    if n % 3 != 1 || n < 4
        error("n deve ser da forma n = 3k+1 e ≥ 4")
    end
    # integral = (3h/8)*[ f(x_1) + 3(f(x_2) + f(x_3)) + 2f(x_4)
    #            +...+
    #            2f(x_{3k-2}) + 3(f(x_{3k-1}) + f(x_{3k})) + f(x_{3k+1})]
    # encontramos os pontos x_1, ..., x_n e calculamos as ordenadas nesses pontos
    x = LinRange(a,b,n)
    F = f.(x)

    # calculamos a soma central
    # S = 3(f(x_2) + f(x_3)) + 2f(x_4) +...+ 2f(x_{3k-2}) + 3(f(x_{3k-1}) + f(x_{3k}))
    S = 0
    for i=2:n-1
        if i % 3 == 1
            S = S + 2*F[i]
        else
            S = S + 3*F[i]
        end
    end

    # calculamos o espaçamento h e o resultado da aproximação
    h = (b-a)/(n-1)
    integral = (3*h/8)*(F[1] + S + F[n])
    return integral
end
```

Figura 10.6: Função da Regra $\frac{3}{8}$ de Simpson.

Feito isso, o próximo passo foi encontrar as aproximações feitas com os espaçamentos h, $\frac{h}{2}$, $\frac{h}{4}$, e assim por diante. Para isso, note que o menor número de pontos que podemos usar neste método é 4, pois queremos uma quantia $n = 3k + 1$ de pontos, com k natural, para termos uma quantia de $3k$ subintervalos. Sendo assim, para $n = 4$ temos o espaçamento h, que é o maior possível. Agora, para termos o espaçamento $\frac{h}{2}$, precisamos de $n = 7$. De modo geral, para termos o espaçamento $\frac{h}{k}$, precisamos de um número de pontos $n = 3k + 1$.

Com isso, temos os valores do vetor v, presente na seguinte função:

```
function ordem_simpson38()
    v = [4,7,13,25,49,97,193,385]
    w = []
    aproximacoes = []
    for k in v
        push!(w, simpson38(x->exp(x.^2), -1,1,k))
    end
    for i in 1 : length(w) - 2
        push!(aproximacoes, log2((w[i] - w[i+1])/(w[i+1] - w[i+2])))
    end
    return aproximacoes
end
```

Figura 10.7: Função para aproximar a ordem de convergência da Regra $\frac{3}{8}$ de Simpson.

Assim como apresentado na seção anterior, com os valores do vetor v são calculadas as aproximações para ordem de convergência do método em questão. Fazendo isso, temos os resultados:

```
ordem_simpson38()
```

```
6-element Array{Any,1}:
 3.1328991118435914
 3.6515756377497492
 3.8929243407997873
 3.9713953167701614
 3.9927187758704368
 3.998171437041729
```

Figura 10.8: Aproximações para a ordem de convergência da Regra $\frac{3}{8}$ de Simpson.

Note que as aproximações tendem ao valor 4. Ou seja, podemos concluir que a ordem de convergência da Regra $\frac{3}{8}$ de Simpson também é 4.

Note que isso já era o esperado, visto que já foi estudado (e apresentado em um relatório anterior) que as duas regras de Simpson possuem a mesma ordem de convergência. Nesses estudos anteriores, vimos que a ordem de convergência da Regra do Trapézio é menor do que a das regras de Simpson. Nestes estudos mais recentes, comprovamos isso fazendo aproximações para as ordens de convergência desses três métodos, onde vimos que a Regra do Trapézio possui ordem de convergência 2 e as regras de Simpson possuem ordem 4.

Capítulo 11

Janeiro de 2021

11.1 Introdução

No mês anterior - dezembro de 2020 - foi observada a melhora na aproximação ao usar as fórmulas de quadratura de Gauss de modo repetido, dividindo o intervalo de integração em subintervalos menores. Além disso, foram aproximadas as ordens de convergência das regras $\frac{1}{3}$ e $\frac{3}{8}$ de Simpson usando a aproximação da integral $\int_{-1}^{1} e^x cos(x)dx$.

Como continuação, no mês de janeiro, ainda utilizando a integral $\int_{-1}^{1} e^x cos(x)dx$, foi calculada a ordem de convergência da fórmula de quadratura de Gauss-Legendre repetida: usando a subdivisão em intervalos.

Isso foi feito com as fórmulas de 2, 3, 4 e 5 pontos. Com isso, foi possível observar como a ordem de convergência dessa fórmula repetida varia à medida que aumentamos a quantidade de pontos, conforme veremos.

11.2 Aproximando $\int_{-1}^{1} e^x cos(x)dx$ com a Fórmula de Gauss-Legendre repetida

Usando 2, 3, 4 e 5 pontos, foram feitas aproximações para a integral em questão. Em todos esses casos, a integral foi calculada usando o intervalo de integração inteiro [-1,1], depois dividindo-o em 2, 4 e 8 subintervalos menores de igual tamanho. Para integrar em cada subintervalo, foi necessário tomar o intervalo novo [a,b] e fazer a mudança de variável:

$$x = \frac{(b-a)\cdot t + (b+a)}{2}$$

Depois de feitas as aproximações, os resultados obtidos são os que seguem. Note que o erro decresce à medida que aumentamos o número de pontos. Além disso, em cada caso, com as quantidades de subintervalos testadas, pode-se notar que o erro diminui consideravelmente sempre que dobramos o número de subintervalos:

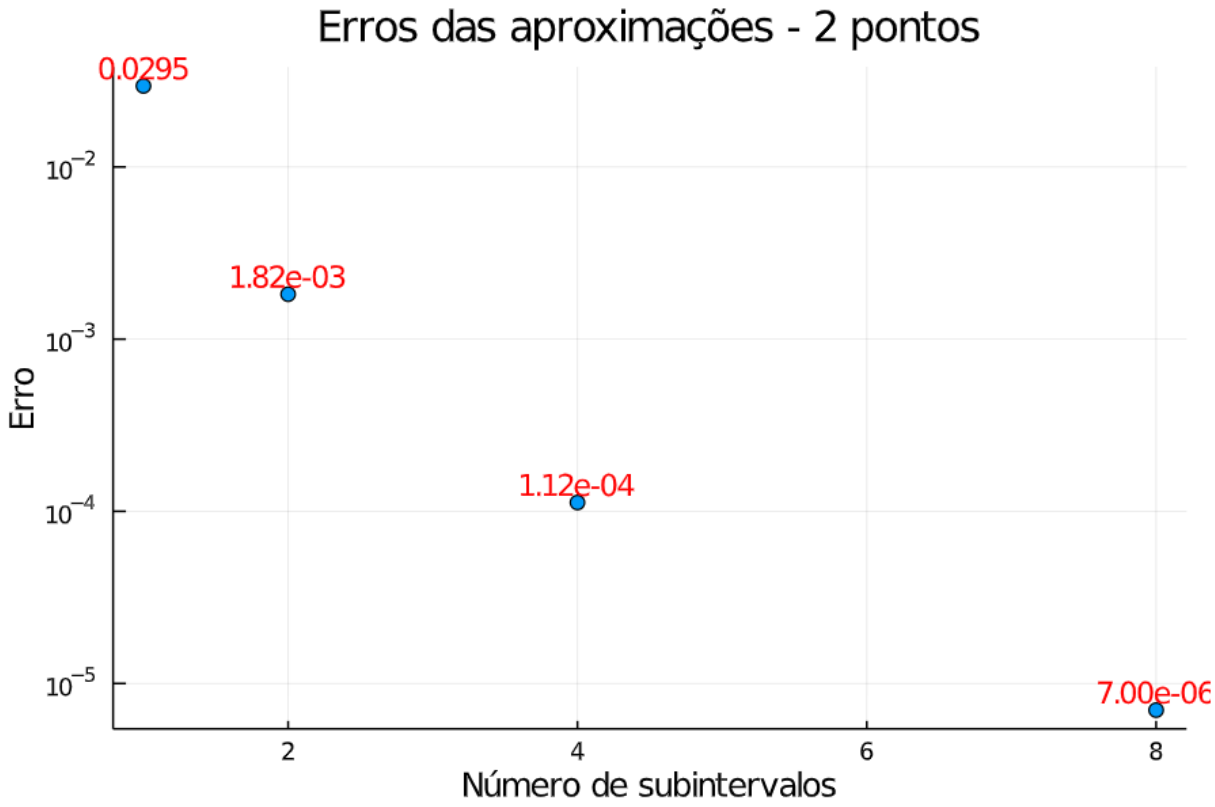

Figura 11.1: Erros com a fórmula de Gauss-Legendre de 2 pontos repetida

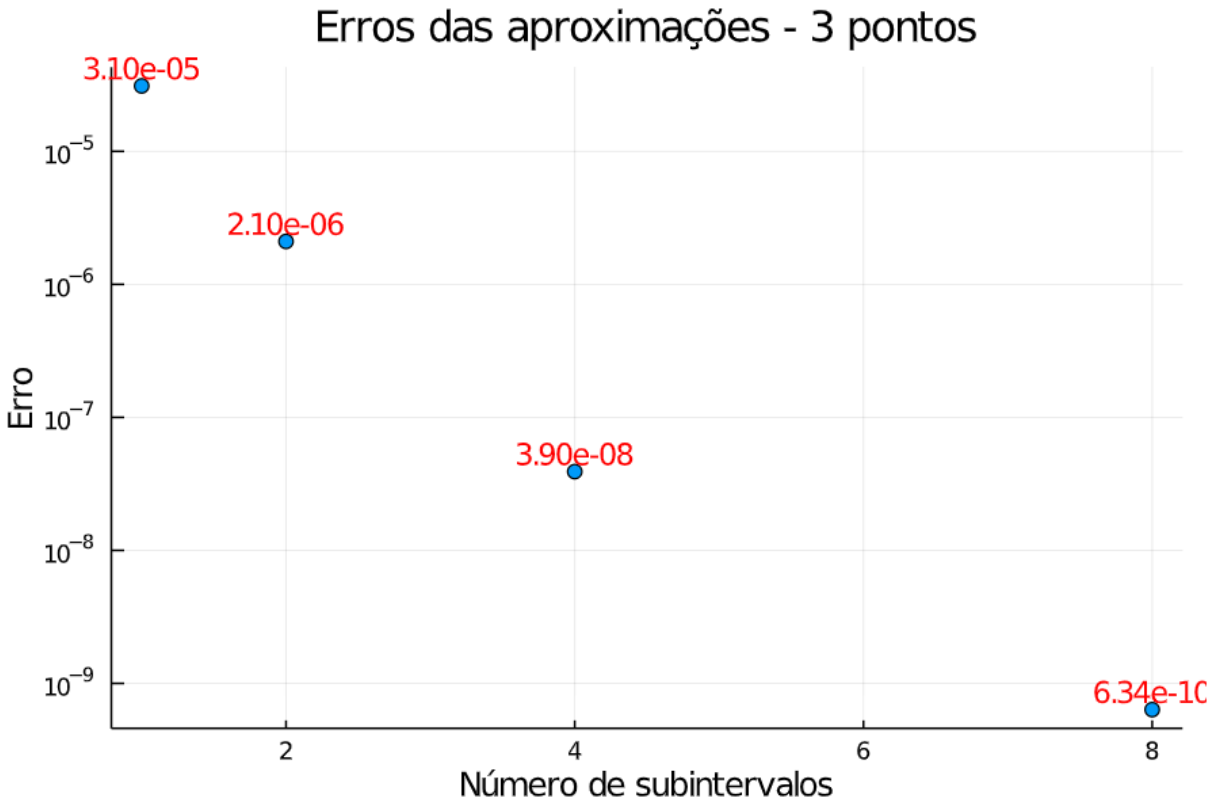

Figura 11.2: Erros com a fórmula de Gauss-Legendre de 3 pontos repetida

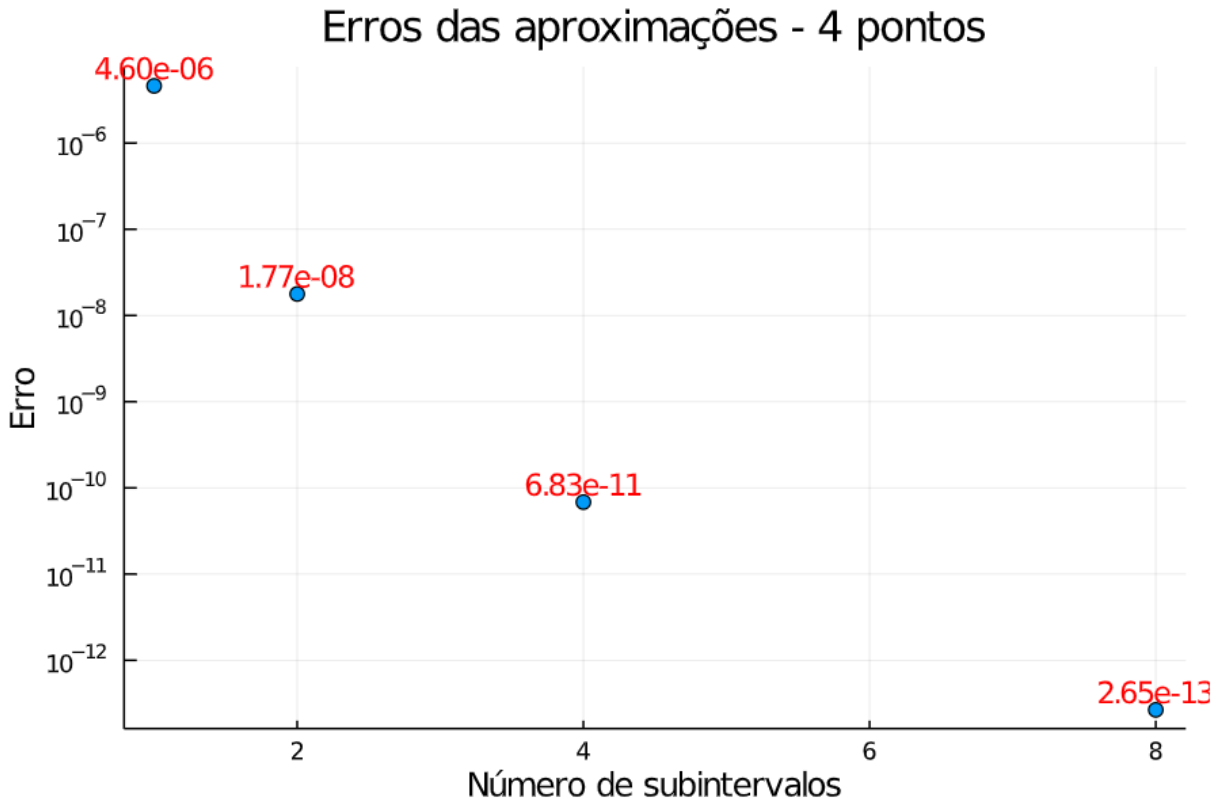

Figura 11.3: Erros com a fórmula de Gauss-Legendre de 4 pontos repetida

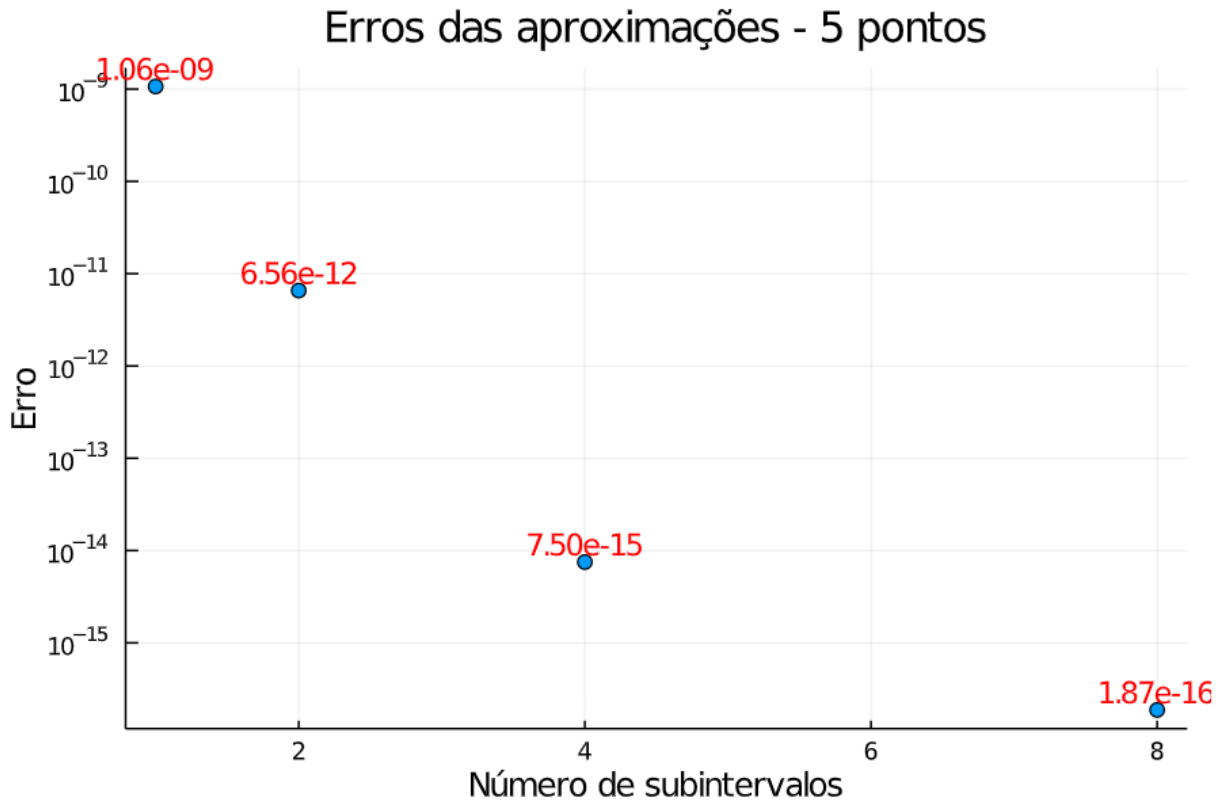

Figura 11.4: Erros com a fórmula de Gauss-Legendre de 5 pontos repetida

11.3 Aproximando as ordens de convergência

Com os resultados das aproximações feitas, prosseguimos para o cálculo da ordem de convergência destes métodos.

Para aproximar a ordem de convergência de cada método, faremos um procedimento semelhante ao já feito anteriormente - nos estudos de dezembro:

Seja $\tilde{a}_h$ uma aproximação da integral feita com o parâmetro h. Vamos dividir h pela metade sucessivamente, obtendo $\frac{h}{2}$ e $\frac{h}{4}$ e, em seguida, calcular $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = p + O(h)$, onde $O(h)$ é ordem de h. Na prática, desconsideramos o valor de $O(h)$. Assim, obtemos uma aproximação para p.

Nos estudos anteriores, esse mesmo procedimento foi usado para aproximar a ordem de convergência das fórmulas de Newton Cottes. Nesses estudos, o parâmetro h envolvido era o espaçamento entre os pontos utilizados na aproximação. Por outro lado, para aproximar a ordem de convergência da fórmula de Gauss-Legendre repetida, o parâmetro h será o tamanho dos subintervalos de integração.

Note que, inicialmente, utilizando somente o intervalo [-1,1], temos o tamanho $h = 2$ do intervalo. Usando dois intervalos, temos o tamanho $\frac{h}{2} = 1$. Usando quatro intervalos, temos $\frac{h}{4} = 0,5$ e, por último, com 8 subintervalos, temos $\frac{h}{8} = 0,25$. O que é importante nessas relações é que temos as aproximações feitas com h, $\frac{h}{2}$, $\frac{h}{4}$ e $\frac{h}{8}$, que correspondem às aproximações com 1, 2, 4 e 8 intervalos, respectivamente. Isso é suficiente para fazermos uma aproximação razoável da ordem de convergência do método em questão utilizando o procedimento citado anteriormente.

Em todos os casos, faremos duas aproximações para a ordem de convergência do método em questão. A primeira será usando h, $\frac{h}{2}$ e $\frac{h}{4}$. A segunda será feita usando $\frac{h}{2}$, $\frac{h}{4}$ e $\frac{h}{8}$, conforme segue.

- **Fórmula de Gauss-Legendre repetida com 2 pontos:**

  Calculando, temos que $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = log_2\left(\frac{\tilde{a}_{h/2} - \tilde{a}_{h/4}}{\tilde{a}_{h/4} - \tilde{a}_{h/8}}\right) = 4.01$, o que nos indica que a ordem de convergência da fórmula de Gauss-Legendre de 2 pontos repetida é 4.

- **Fórmula de Gauss-Legendre repetida com 3 pontos:**

  Calculando, temos que $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = 3.80$ e $log_2\left(\frac{\tilde{a}_{h/2} - \tilde{a}_{h/4}}{\tilde{a}_{h/4} - \tilde{a}_{h/8}}\right) = 5.74$. Com isso, não foi possível determinar precisamente a ordem de convergência deste método com o procedimento usado.

- **Fórmula de Gauss-Legendre repetida com 4 pontos:**

  Calculando, temos que $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = 8.01$ e $log_2\left(\frac{\tilde{a}_{h/2} - \tilde{a}_{h/4}}{\tilde{a}_{h/4} - \tilde{a}_{h/8}}\right) = 8.08$. Com isso, podemos inferir que a ordem de convergência da fórmula de Gauss-Legendre de 4 pontos repetida é 8.

- **Fórmula de Gauss-Legendre repetida com 5 pontos:**

  Calculando, temos que $log_2\left(\frac{\tilde{a}_h - \tilde{a}_{h/2}}{\tilde{a}_{h/2} - \tilde{a}_{h/4}}\right) = 7.33$ e $log_2\left(\frac{\tilde{a}_{h/2} - \tilde{a}_{h/4}}{\tilde{a}_{h/4} - \tilde{a}_{h/8}}\right) = 9.73$. Com isso, não foi possível obter a ordem de convergência deste método de forma precisa.

Como vimos, não foi obtida precisamente a ordem de convergência dos métodos com 3 e 5 pontos. Por isso, foi aproximada novamente a ordem dos métodos, mas dessa vez usando o valor exato da integral (truncado com 10 casas decimais), que é:

$$\int_{-1}^{1} e^x cos(x)dx = \frac{e^x(sen(x) + cos(x))}{2} = 1.9334214962$$

Usando o valor exato da integral, prosseguiremos da seguinte forma para aproximar a ordem de convergência:

Seja a o valor exato e $\tilde{a}_h$ uma aproximação com o parâmetro h. Dividimos h por 2 e consideramos a razão dos erros $a - \tilde{a}_h$ e $a - \tilde{a}_{h/2}$. Com isso, temos que $log_2\left(\frac{a - \tilde{a}_h}{a - \tilde{a}_{h/2}}\right) = p + O(h)$, onde $O(h)$ é o erro em função de h. Na prática, desconsideramos $O(h)$.

Para cada método, foi usado esse procedimento três vezes, usando h e $\frac{h}{2}$ na primeira aproximação; $\frac{h}{2}$ e $\frac{h}{4}$ na segunda e, na última aproximação, $\frac{h}{4}$ e $\frac{h}{8}$.

Os resultado obtidos foram os seguintes:

- **Fórmula de Gauss-Legendre repetida com 2 pontos:**

  Foram obtidas as aproximações 4.01, 4.01 e 4.00, confirmando o que já havíamos inferido: que a ordem de convergência neste caso é 4.

- **Fórmula de Gauss-Legendre repetida com 3 pontos:**

  Foram obtidas as aproximações 3.88, 5.74 e 5.94. Os resultados não indicam muito precisamente a ordem, mas a melhor aproximação (5.94), calculada com $\frac{h}{4}$ e $\frac{h}{8}$, sugere que - neste caso - a ordem de convergência é 6. Adotemos isso como apenas uma sugestão, já que as aproximações não foram tão precisas como no caso de 2 pontos.

- **Fórmula de Gauss-Legendre repetida com 4 pontos:**

  As aproximações obtidas foram 8.01, 8.02 e 8.00. Isso confirma que a ordem de convergência deste método é 8.

- **Fórmula de Gauss-Legendre repetida com 5 pontos:**

  Neste caso só foram obtidas 2 aproximações por conta de um erro na aproximação com $\frac{h}{4}$ e $\frac{h}{8}$. As aproximações foram 7.34 e 9.77, que foi a melhor obtida. Com isso, uma sugestão é que a ordem de convergência, neste caso, é 10.

Depois de obtidas todas as aproximações, o comportamento notado foi que as fórmulas de quadratura de Gauss-Legendre repetidas de 2, 3, 4 e 5 pontos possuem ordem de convergência 4, 6, 8 e 10, respectivamente. Ou seja, quando aumentamos um ponto na fórmula, a ordem de convergência aumenta em 2.

Ressalto que essa conclusão dependeu de alguns chutes para as ordens de convergência, principalmente para 3 e 5 pontos, onde as aproximações não apontaram precisamente para um único valor.

11.4 Próximos estudos

Para os próximos estudos, a sugestão do meu orientador - Elías Gudiño - foi considerar outros problemas no estudo, além da integral $\int_{-1}^{1} e^x cos(x) dx$. Portanto, no próximo mês, ainda serão estudadas as ordens de convergência, mas utilizando mais problemas.

Capítulo 12

Fevereiro de 2021

12.1 Introdução

No mês anterior, usando a integral $\int_{-1}^{1} e^x cos(x)dx$, foi calculada a ordem de convergência da fórmula de quadratura de Gauss-Legendre repetida: usando a subdivisão em intervalos.

Ao fazer isso com as fórmulas de 2, 3, 4 e 5 pontos, foi observado que, quando aumentamos 1 ponto na fórmula, a ordem de convergência do método aumenta em duas unidades. Apesar disso, em alguns casos, a aproximação da ordem de convergência não foi precisa.

Com isso, os estudos do mês de fevereiro contemplam:

- Aproximar as ordens de convergência dos métodos estudados no mês anterior de uma forma diferente, para obter mais precisão;
- Considerar outras integrais para repetir o estudo sobre as fórmulas repetidas e suas ordens de convergência.

12.2 Aproximando as ordens de convergência com gráficos

Seja $\tilde{a}_h$ uma aproximação da integral feita com o parâmetro h, onde a é o valor exato da integral. Neste caso, a aproximação da ordem de convergência será feita da seguinte forma:

- Será plotado, no eixo x, os valores de $log(h)$, para diferentes valores do parâmetro h. No eixo y, será plotado: $log(|a - \tilde{a}_h|)$, o erro - na escala logarítimica - da aproximação com o parâmetro h.
- Será encontrada a reta que melhor se ajusta ao conjunto de pontos, minimizando a SQE (Soma dos Quadrados dos Erros).
- O coeficiente angular dessa reta será uma aproximação para a ordem de convergência do método em questão.

Observação: Como estamos trabalhando com a aplicação da fórmula de quadratura de Gauss-Legendre repetida, dividindo o intervalo de integração principal $[a, b]$ em k subintervalos, o parâmetro h será o espaçamento entre os pontos. Ou seja: $h = \dfrac{b-a}{k}$.

Observação: Vale ressaltar que minimizamos a soma dos quadrados dos erros porque a tarefa de minimizar os erros é mais difícil, visto que o erro é dado em módulo, o que torna as operações mais complicadas. Por outro lado, ao elevar o erro ao quadrado, eliminamos o módulo e estamos trabalhando com uma função cujas operações de integração e diferenciação são muito mais simples.

Fazendo isso com as fórmulas de quadratura de Gauss-Legendre repetidas de 2, 3, 4 e 5 pontos, obtemos os seguinte resultados:

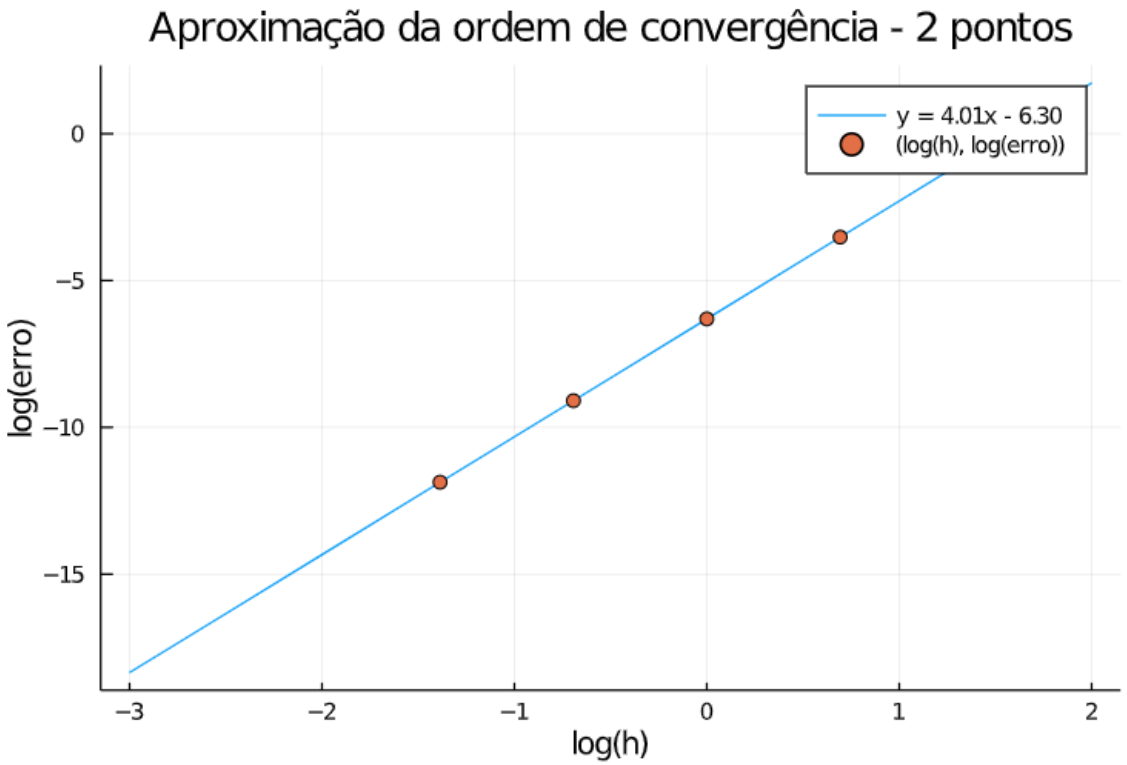

Figura 12.1: Aproximação da ordem de convergência - 2 pontos

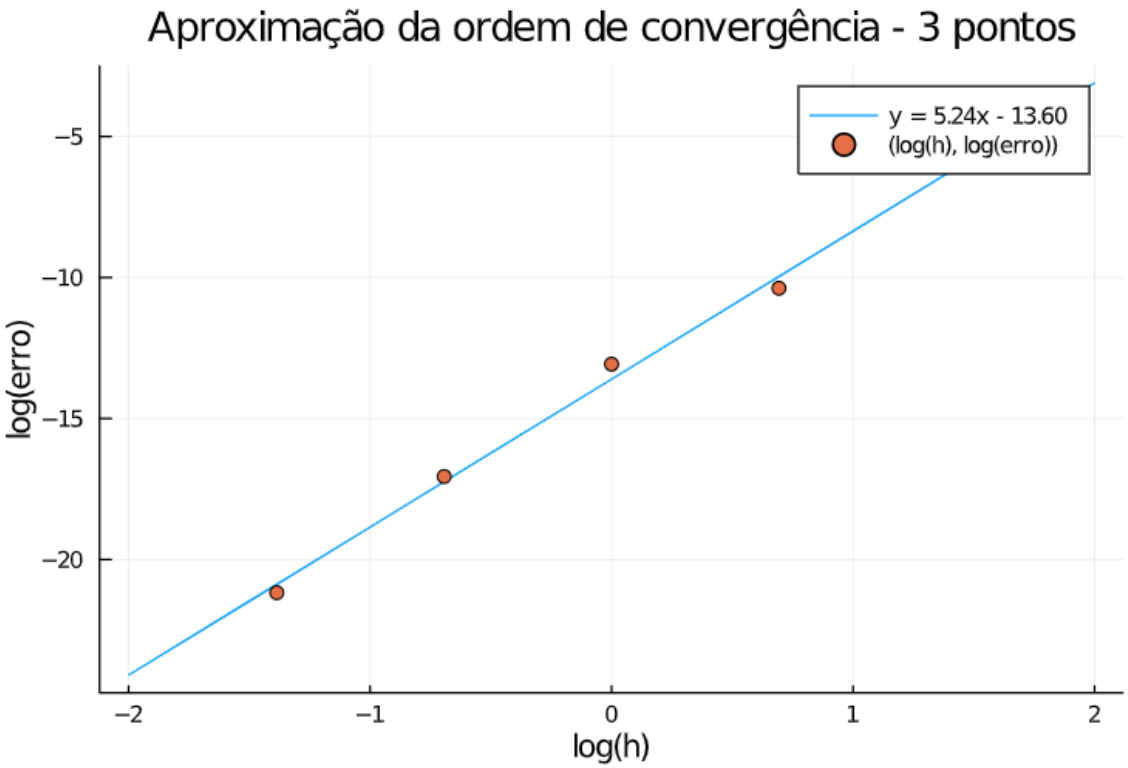

Figura 12.2: Aproximação da ordem de convergência - 3 pontos

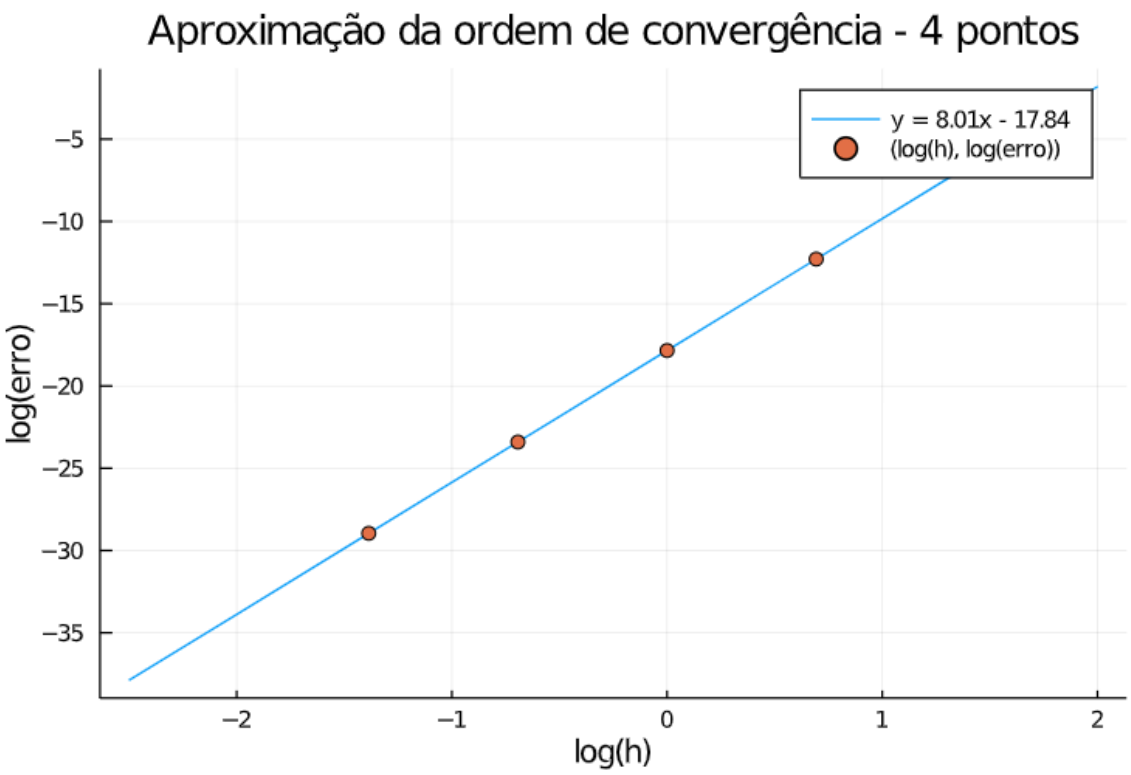

Figura 12.3: Aproximação da ordem de convergência - 4 pontos

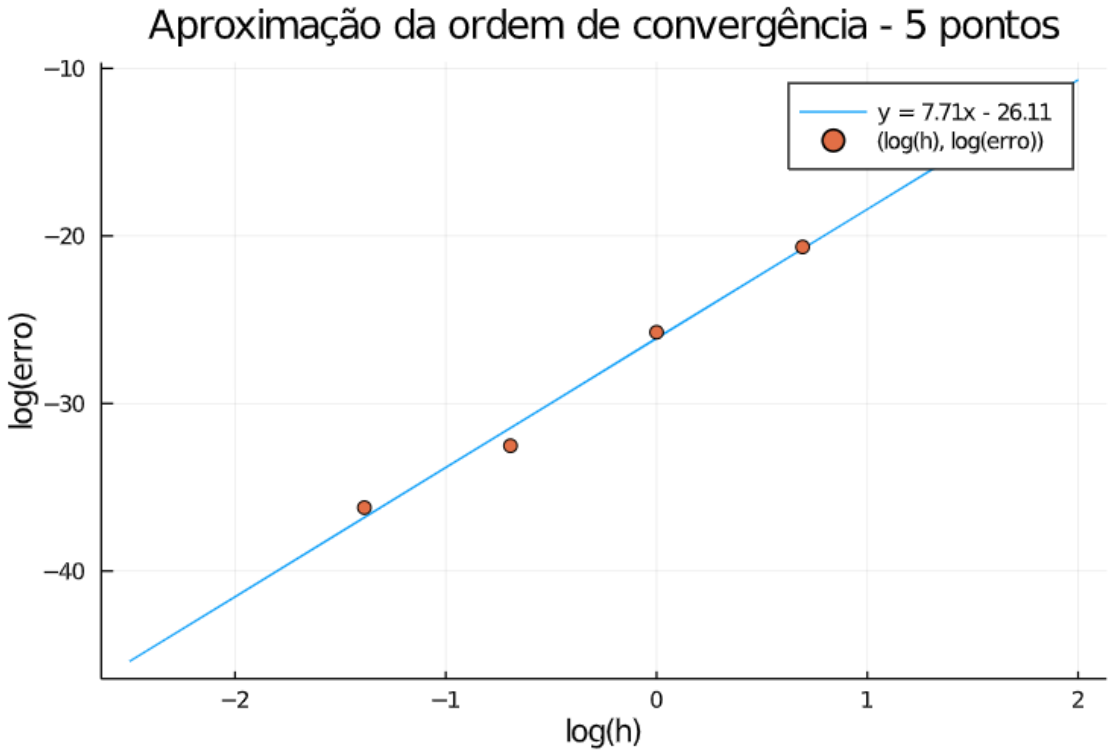

Figura 12.4: Aproximação da ordem de convergência - 5 pontos

Com base nos gráficos obtidos e na equação da reta que fita os pontos em cada caso - presentes nas respectivas legendas - podemos inferir as seguintes aproximações para as ordens de convergência dos métodos de quadratura de Gauss-Legendre repetidos (com a subdivisão do intervalo de integração em intervalos menores):

- Com 2 pontos: ≈ 4.01;

- Com 3 pontos: ≈ 5.24;
- Com 4 pontos: ≈ 8.01;
- Com 5 pontos: ≈ 7.71.

Esses valores - conforme já explicado - são os coeficientes angulares (ou inclinações) das retas que minimizam a soma dos quadrados dos erros ao fitar o conjunto de pontos.

Comparando com as aproximações feitas no mês anterior, a ordem de convergência para os métodos de 2 e 4 pontos foram precisas, pois as ordens de convergência desses métodos são 4 e 8, respectivamente, como obtido. Por outro lado, as aproximações para as ordens de convergência dos métodos com 3 e 5 pontos não foram boas e precisas, tanto é que apresentam divergência até mesmo com as aproximações já feitas no mês anterior. Além disso, nota-se que, com 5 pontos, a aproximação da ordem de convergência foi menor do que com 4 pontos, o que vai no sentido oposto ao esperado.

12.3 Trabalhando com outra integral: $\int_0^{\pi/2} \sqrt{1 - sen^2(x)}dx$

Nesta parte do estudo, será trabalhada a integral $\int_0^{\pi/2} \sqrt{1 - sen^2(x)}dx$ da seguinte forma: sua integral será aproximada usando as fórmulas de quadratura de Gauss-Legendre repetidas com 2, 3, 4 e 5 pontos, fazendo a subdivisão do intervalo principal em vários subintervalos menores, de mesmo tamanho, e integrando separadamente em cada intervalo menor.

Em seguida, serão feitas as aproximações das ordens de convergência dos métodos estudados de duas formas diferentes: algebricamente e calculando a inclinação da reta que minimiza a soma dos quadrados dos erros ao fitar o conjunto de pontos obtidos com as aproximações. Todo o estudo será detalhado nas seções seguintes.

12.3.1 Aproximando com as fórmulas de Gauss-Legendre repetidas

Esta parte não será detalhada porque o procedimento é idêntico ao já mostrado nos relatórios anteriores. De forma geral, o que será feito é o seguinte: a integral em questão será aproximada com as fórmulas de Gauss-Legendre de 2, 3, 4 e 5 pontos, fazendo a divisão do intervalo original em intervalos menores.

Para fazer essa divisão, o intervalo de integração será dividido e será feita uma mudança de variável, de forma que cada integral menor esteja com os limites de integração -1 e 1, para ser possível usar Gauss-Legendre.

Isso foi feito. Em geral, as aproximações foram boas para o valor exato da integral: 1. A maior parte dos erros foi da ordem de 10^{-5}.

12.3.2 Aproximando as ordens de convergência

Para aproximar as ordens de convergência, foi feito como nos estudos abordados no início deste relatório: foram plotados os erros no eixo y e os valores do espaçamento h no eixo x. Ambos os eixos na escala log. Em seguida, os pontos foram fitados por uma reta e a inclinação desta reta é a aproximação para a ordem de convergência.

Os resultados não foram convincentes. Com a fórmula de 2 pontos, dividindo o intervalo de integração em 1 até 6 subintervalos, a aproximação foi 7.08.

Com 3 pontos, a inclinação da reta foi 11.70; com 4 pontos, 17.86 e, com 5 pontos, 19.68.

Como ilustração, seguem as imagens dos gráficos obtidos:

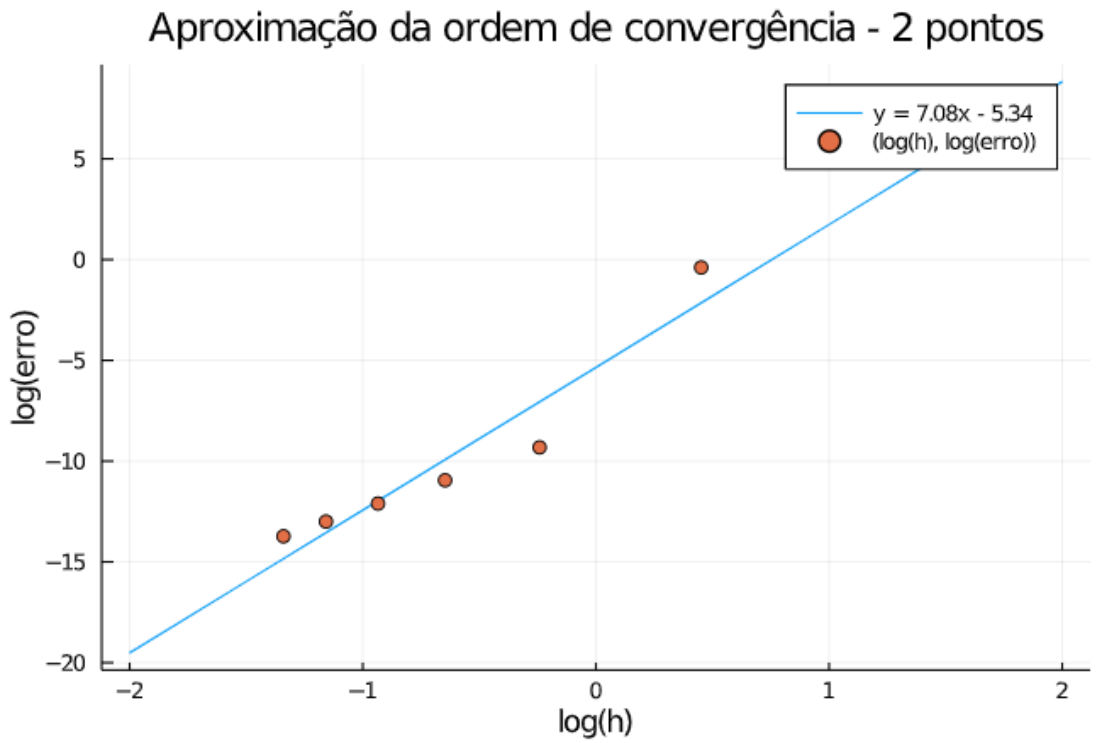

Figura 12.5: Aproximação para a ordem de convergência - 2 pontos.

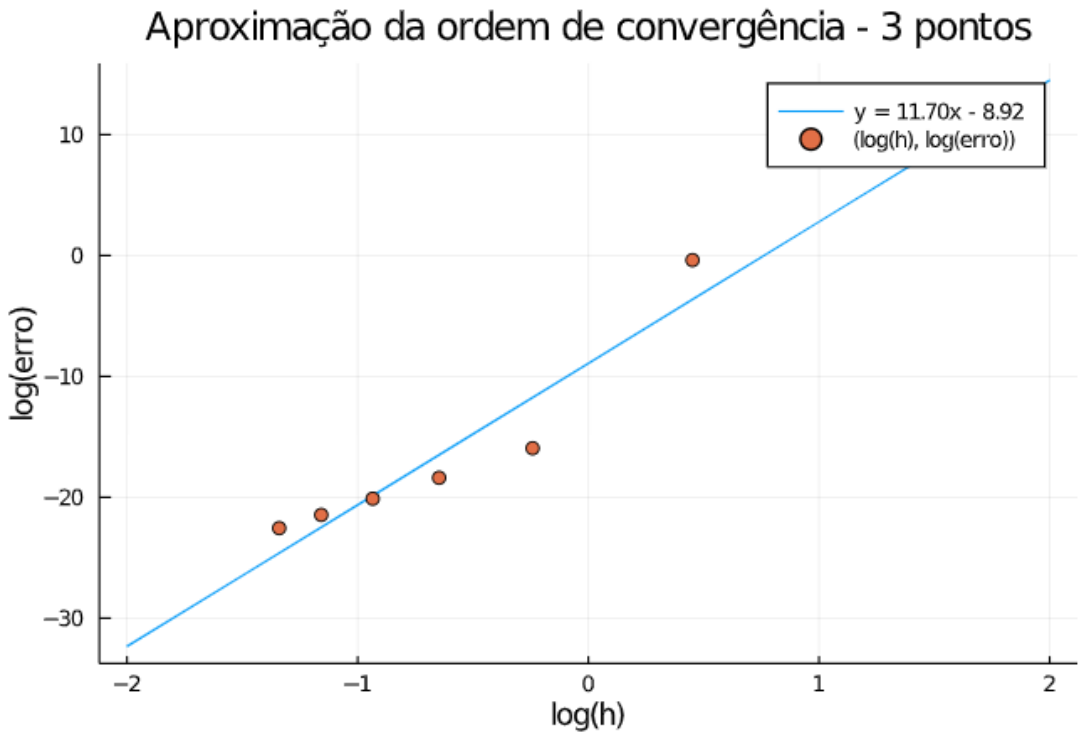

Figura 12.6: Aproximação para a ordem de convergência - 3 pontos.

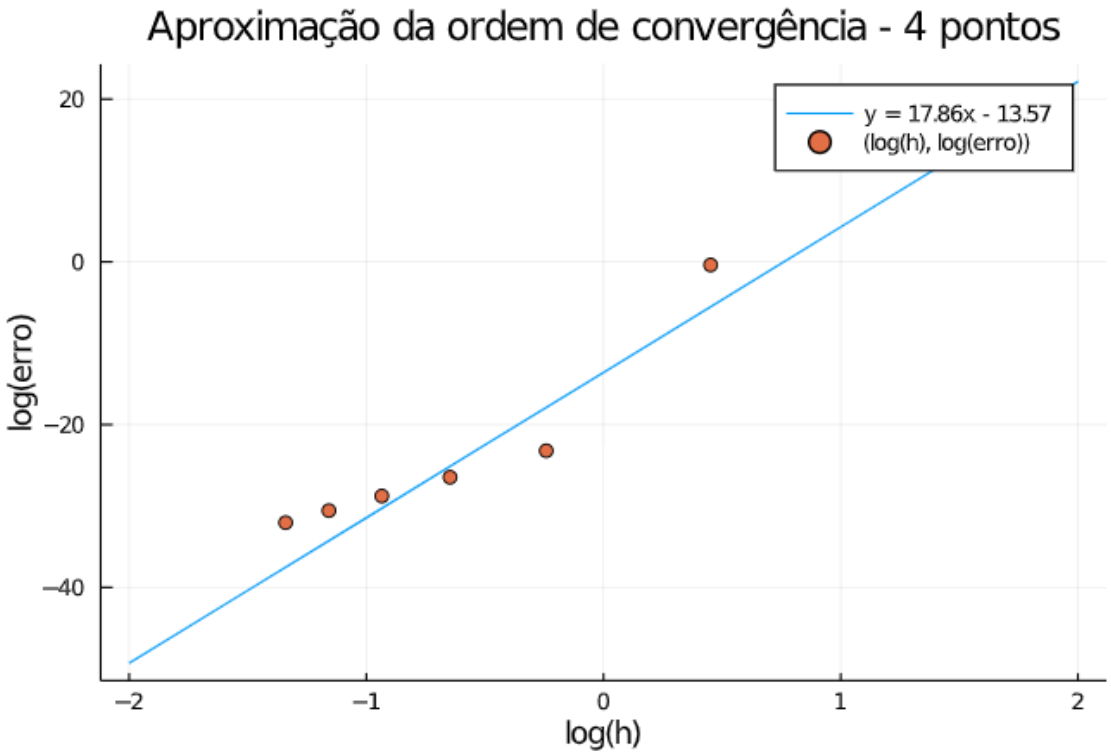

Figura 12.7: Aproximação para a ordem de convergência - 4 pontos.

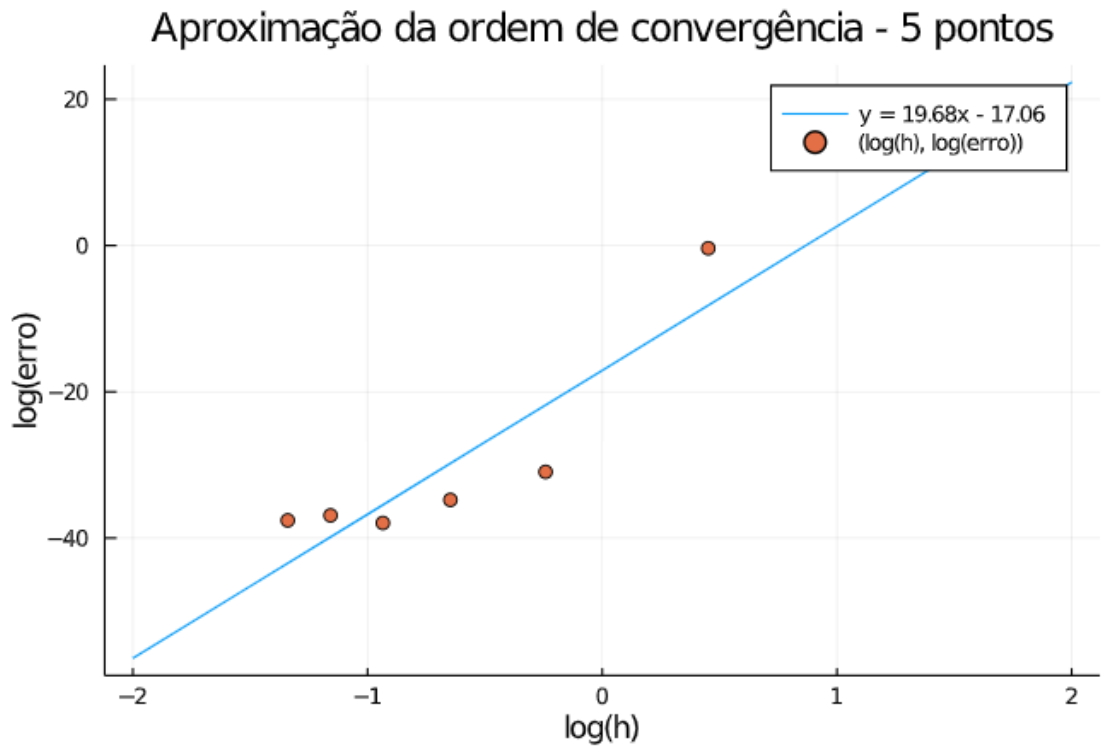

Figura 12.8: Aproximação para a ordem de convergência - 5 pontos.

Percebe-se que as retas não fitam bem os conjuntos de dados por conta do último ponto. Não foram encontrados erros nos valores que definem o último ponto, mas é visível que ele não se comporta bem quando comparado ao restante do conjunto de pontos.

Talvez um conjunto maior de pontos fornecesse uma aproximação melhor para a ordem de convergência dos métodos. Porém, no caso das fórmulas repetidas usando Gauss-Legendre, isso torna-se inviável porque, para dividir um intervalo de integração em n subintervalos, são necessárias n mudanças de

variável, para que - em cada subintervalo - a integral seja calculada com os limites de integração -1 e 1, para ser possível usar Gauss-Legendre.

12.4 Próximos estudos

Para os próximos estudos, uma intenção é conferir se os resultados obtidos para as aproximações das ordens de convergência estão corretos.

Também é um alvo de estudo futuro entender por que calcular o coeficiente da reta que fita os dados da forma $(h, erro)$ é o mesmo que usar a fórmula $log_2\left(\frac{a-\tilde{a}_h}{a-\tilde{a}_{h/2}}\right) \cong P$ para aproximar a ordem de convergência P reduzindo o parâmetro h pela metade sucessivamente.

Capítulo 13

Referências

[1] Quarteroni, Sacco and Saleri, Numerical Mathematics, Springer, 2007.

[2] M. A. G. Ruggiero e V. L. R. Lopes, Cálculo Numérico Aspectos Teóricos e Computacionais, 2ª. ed., Pearson, 1996.

[3] N. M. Bertoldi Franco, Cálculo Numérico, USP.

[4] Olof Runborg, Numerical Convergence Rates. KTH Engineering Sciences, 2012.