UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

MÉTODOS NUMÉRICOS EM ENGENHARIA

GABRIEL CAICEDO POTOSÍ

RA:272605

LISTA 2

Campinas

2024

Resumo

A integração numérica é o estudo de como o valor numérico de uma integral pode ser encontrado. A importância da integração numérica reside em sua capacidade de fornecer soluções aproximadas para problemas de cálculo que não têm solução analítica direta. Isso é crucial em numerosos campos da ciência, engenharia e matemática aplicada.

O objetivo deste trabalho é implementar um código em C++ para realizar integração numérica utilizando os seguintes métodos: 1) Método do trapézio; 2) Simpson 1/3; 3) Simpson 3/8; e 4) Gauss-Legendre com 1 a 4 pontos de integração.

Para cada um desses métodos, calcula-se a taxa de convergência dividindo os intervalos de integração e compara-se com a taxa teórica. Essa comparação permite avaliar a precisão de cada método na aproximação da integral de cada função.

Além disso, gráfica-se a taxa de convergência para o método de Gauss-Legendre com 4 pontos de integração, e identificam-se os segmentos onde a integração é exata, proporcionando assim uma melhor compreensão deste método em particular.

Os resultados obtidos são analisados para compreender o desempenho de cada método de integração numérica para as funções fornecidas.

1 Introdução

A integração numérica é o estudo de como o valor numérico de uma integral pode ser encontrado, ou seja, é usada para aproximar o valor de uma integral definida quando não é possível obter uma solução analítica (Davis & Rabinowitz, 2007; Smyth, 1998). O presente trabalho foca na implementação e análise de vários métodos de integração numérica utilizando C++ como ferramenta de programação.

O objetivo principal deste estudo é avaliar a precisão de diferentes métodos de integração numérica, tais como o método do trapézio, Simpson 1/3, Simpson 3/8 e Gauss-Legendre com 1 até 4 pontos de integração.

Além de implementar os métodos mencionados, calcula-se a taxa de convergência para cada um deles, comparando os resultados obtidos com as taxas teóricas. Assim, esta análise permite avaliar a eficiência dos métodos na aproximação da integral.

O código implementado em C++ também permite o uso do método de Gauss-Legendre com 4 pontos de integração e a graficação da taxa de convergência em função do intervalo de integração. Consequentemente, identificam-se os trechos onde a integração é exata, proporcionando uma melhor compreensão da precisão e estabilidade deste método em específico.

Os resultados obtidos são analisados com o objetivo de fornecer uma visão clara e completa sobre o desempenho de cada método de integração numérica em diversos contextos e situações de convergência. Estas análises e conclusões serão fundamentais para entender a aplicabilidade e limitações dos métodos estudados, assim como para orientar seu uso eficiente em problemas práticos de cálculo e engenharia.

1.1 Método do Trapézio

A regra do trapézio é um método de integração que permite calcular o valor aproximado de uma integral definida. O princípio se baseia em aproximar o valor da integral de $f(x)$ pelo da função linear, que passa através dos seguintes pontos: *1)* $(a, f(a))$ e *2)* $(b, f(b))$. A integral dessa função $f(x)$ é aproximadamente igual à área sob a função linear gerada a partir dos pontos 1 e 2.

A seguinte equação mostra como calcular a integral de uma função $f(x)$empregando o método do trapézio.

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2} \tag{1}$$

1.2 Método de Simpson

As regras de Simpson são aproximações para integrais definidas; entre essas regras está a regra de Simpson 1/3. Se essa regra for aplicada a n subdivisões iguais no intervalo de integração $[a, b]$, obtém-se a regra composta de Simpson 1/3. Então, os pontos dentro do intervalo de integração recebem pesos alternados 4/3 e 2/3.

1.2.1 Método de Simpson 1/3

O método de integração numérica de Simpson 1/3 baseia-se na interpolação quadrática (avaliada para $n = 2$) e está definido pela equação 5.

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)] \tag{2}$$

Se no intervalo de integração $[a, b]$ estiver sendo integrada uma função relativamente suave, então este método obtém uma solução aproximada adequada à solução exata.

1.2.2 Método de Simpson 3/8

O método de Simpson 3/8, conhecido como a segunda regra de Simpson, baseia-se em uma interpolação cúbica. Esta regra é mostrada na seguinte equação:

$$\int_a^b f(x)dx \approx \frac{b-a}{8}[f(a) + 3 + 3f(\frac{2a+b}{3}) + 3f(\frac{a+2b}{3}) + f(b)] \tag{3}$$

1.3 Método de Gauss-Legendre

O método de integração de Gauss-Legendre permite aproximar o valor da integral de uma função $f(x)$ no intervalo $[a, b]$. A solução aproximada da integral definida utilizando o método de Gauss-Legendre é calculada usando a equação 4.

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} W_i f(Xi) \tag{4}$$

Wi e Xi são calculados utilizando as equações 5 e 6, respectivamente.

$$W_i{=}w_i(\frac{b-a}{2}) \tag{5}$$

$$X_i = (\frac{x_i+1}{2})(b-a) + a \tag{6}$$

Onde:

- wi são os pesos no intervalo $[-1, 1]$.
- xi são os pontos no intervalo $[-1, 1]$.

Os pesos (wi) e os pontos (xi) usados neste método de integração são mostrados na tabela 1.

Tabela 1: Pontos e pesos do método de Gauss-Legendre.

Núm. de pontos (n)	x_i	w_i
1	0	2
2	-0.57735026	1
	0.57735026	1
3	-0.77459666	0.555555555
	0	0.888888888
	0.77459666	0.555555555
4	-0.861136311	0.347854845
	-0.339981043	0.652145154
	0.339981043	0.652145154
	0.861136311	0.347854845

1.4 Erro

O erro é definido como o valor absoluto da diferença entre a solução exata e a solução aproximada. A equação 7 mostra o erro para o método de Gauss-Legendre.

$$e = \mid \int_a^b f(x)dx - \sum_{i=1}^{n} W_i f(Xi) \mid \tag{7}$$

1.4.1 Norma do erro

A norma do erro é calculada utilizando a equação 8.

$$\| e \| = Ch^p \tag{8}$$

Onde:

- $\|e\|$ é a norma do erro.
- C é uma constante que depende dos dados do problema.

• h é o tamanho do elemento.

• p é a taxa de convergência.

Para este trabalho, h é determinado uttilizando a equação 9.

$$h = \frac{(b-a)}{n_{el}} \tag{9}$$

Onde, n_{el} representa o número de elementos, ou o número de intervalos nos quais o domínio é dividido.

O número de elementos é calculado utilizando a equação 10.

$$n_{el} = 2^{nref} \tag{10}$$

Onde, n_{ref} é o número de refinamientos no domínio.

1.4.2 Erro em função do número de intervalos

O erro em função do número de intervalos está definido pela equação 11.

$$\| e \| \approx \sum_{i=1}^{n} \| e_i \| \tag{11}$$

1.5 Taxa de convergência

Para determinar a taxa de convergência, é necessário aplicar o logaritmo à equação 8, como mostrado na equação seguinte:

$$Log(\| e \|) = Log(C) + pLog(h) \tag{12}$$

Para o caso de estudo, a taxa de convergência p é calculada em função dos seguintes erros:

• e_i calculado em função de h_i.

• e_{i+1} calculado em função de h_{i+1}.

Empregando a equação 12 para e_i e e_{i+1}, obtemos as seguintes equações:

$$Log \| e_i \| = Log(C) + pLog(h_i) \tag{13}$$

$$Log \| e_{i+1} \| = Log(C) + pLog(h_{i+1}) \tag{14}$$

Procede-se a subtração da equação (13) menos a equação (14), com o objetivo de eliminar o termo constante C e isolar o expoente p, e assim obter a equação 15.

$$p = \frac{Log(\parallel e_{i+1} \parallel / \parallel e_i \parallel)}{Log(h_{i+1}/h_i)} \tag{15}$$

2 Implementação de Métodos de Integração Numérica em C++

A figura 1 mostra a implementação do método "ValExact" (Linha 25), que pertence à classe "NumericalIntegration". Este método recebe como argumentos x1 e x2 (linha 25), que correspondem aos limites de integração. Utiliza "funcionExact", que estabelece a solução da integral indefinida e calcula "val1" e "val2" (linhas 26 e 27), representando os valores obtidos ao avaliar a solução da integral indefinida em x1 e x2, respectivamente. A subtração de "val2" menos "val1" (linha 28) permite encontrar a solução exata da integral, que é retornada na linha 32.

```
23 //EXACT INTEGRATION
24 //Implementation of definite integral
25 double NumericalIntegration::ValExact(double x1, double x2){
26     double val1 = funcionExact(x1);
27     double val2 = funcionExact(x2);
28     double area = val2 - val1;
29     if(area <0.0){
30         std::cout<<"Area Negativa"<<std::endl;
31     }
32     return area;
33     }
```

Figura 1: Implementação da solução exata em C++.

A Figura 2 exibe a implementação do método dos trapézios, onde se cria o método “IntegrateTrapecio” na linha 50, o qual possui como argumento os limites de integração e calcula a integral (linha 52) utilizando a equação 1. A linha 56 na figura mostra a implementação desse método em função do número de refinamentos, onde se calcula:

1. O número de elementos (linha 58) empregando a equação 10.
2. h utilizando a equação 9 (linha 60).

3. Os pontos “point1” e “point2”, que são calculados em função de h, para posteriormente serem avaliados.

4. A “área” usando o método “IntegrateTrapecio” (linha 50).

```
49 //TRAPEZIUM METHOD
50 double NumericalIntegration::IntegrateTrapecio(double x1, double x2){
51     //x1 and x2 = limits of integration
52     double area = 0.5*(x2-x1)*(funcion(x1) + funcion(x2));
53     return area;
54 }
55 //Implementation according to the number of refinements
56 std::vector<double> NumericalIntegration::IntegrateTrapecio(double a, double b, int ref){
57 
58     int nel = pow(2, ref);
59     std::vector<double> vectorAreas(nel,0);
60     double h = (b-a)/nel;
61     for (int i=0; i< nel; i++) {
62         double point1 = a + i*h;
63         double point2 = a + (i+1)*h;
64         double area = IntegrateTrapecio(point1, point2);
65         vectorAreas[i]=area;
66     }
67     fsol =vectorAreas;
68     return vectorAreas;
69 }
```

Figura 2: Implementação do método do trapézio em C++.

A Figura 3 ilustra a implementação do método de Simpson 1/3, na qual se estabelece o método “IntegrateSimpson1” (linha 72). Este método recebe como argumentos os limites de integração e calcula a área utilizando a equação 2. A linha 78 da figura detalha a aplicação deste método em função do número de refinamentos.

```cpp
// SIMPSON METHOD 1/3
double NumericalIntegration::IntegrateSimpson1(double x1,double x2){
    double area = ((x2-x1)/6)*(funcion(x1)+(4*funcion((x1+x2)/2))+funcion(x2));
    return area;
}

//Implementation according to the number of refinements
std::vector<double> NumericalIntegration::IntegrateSimpson1(double a, double b, int ref){

    int nel = pow(2, ref);
    std::vector<double> vectorAreas(nel,0);
    double h = (b-a)/nel;
    for (int i=0; i< nel; i++) {
        double point1 = a + i*h;
        double point2 = a + (i+1)*h;
        double area = IntegrateSimpson1(point1, point2);
        vectorAreas[i]=area;
    }
    fsol =vectorAreas;
    return vectorAreas;
}
```

Figura 3: Implementação do método de Simpson 1/3 em C++.

A Figura 4 apresenta a implementação do método de Simpson 3/8, na qual o método "IntegrateSimpson3" é estabelecido (linha 94). Este método recebe como argumentos os limites de integração, x1 e x2, e calcula a área (a integral) utilizando a equação 3. A linha 100 da figura mostra como este método é aplicado em função do número de refinamentos.

```cpp
//SIMPSON METHOD 3/8
double NumericalIntegration::IntegrateSimpson2(double x1, double x2){
    double area = ((x2-x1)/8)*(funcion(x1)+(3*funcion((2*x1+x2)/3))+(3*funcion((x1+(2*x2))/3))+funcion(x2));
    return area;
}

//Implementation according to the number of refinements
std::vector<double> NumericalIntegration::IntegrateSimpson2(double a, double b, int ref){
    int nel = pow(2, ref);
    std::vector<double> vectorAreas(nel,0);
    double h = (b-a)/nel;
    for (int i=0; i< nel; i++) {
        double point1 = a + i*h;
        double point2 = a + (i+1)*h;
        double area = IntegrateSimpson2(point1, point2);
        vectorAreas[i]=area;
    }
    fsol =vectorAreas;
    return vectorAreas;
}
```

Figura 4: Implementação do método Simpson 3/8 em C++.

A Figura 5 ilustra a implementação do método de Gauss-Legendre, na qual o método "Cal_points_weights" é estabelecido dentro da classe “NumericalIntegration” (linha 72). Este método é projetado para configurar o número de pontos e pesos de acordo com o caso específico.

```
114 //GAUSS METHOD
115 std::vector<std::pair<double, double> > NumericalIntegration::Calc_points_weights(int num_points) {
116     // Table of points and weights for Gaussian squaring
117 
118 
119     std::vector<std::pair<double, double> > points_weights;
120     // Defining points and weights for Gaussian quadrature with num_points points
121     switch (num_points) {
122         case 1:
123                 points_weights.resize(1);
124                 points_weights[0] = std::make_pair(0.0, 2.0);
125 
126                 break;
127             case 2:
128                 points_weights.resize(2);
129                 points_weights[0] = std::make_pair(-0.5773502691896257, 1.0);
130                 points_weights[1] = std::make_pair(0.5773502691896257, 1.0);
131                 break;
132             case 3:
133                 points_weights.resize(3);
134                 points_weights[0] = std::make_pair(-0.7745966692414834, 0.5555555555555556);
135                 points_weights[1] = std::make_pair(0.0, 0.8888888888888888);
136                 points_weights[2] = std::make_pair(0.7745966692414834, 0.5555555555555556);
137                 break;
138             case 4:
139                 points_weights.resize(4);
140                 points_weights[0] = std::make_pair(-0.8611363115940526, 0.3478548451374539);
141                 points_weights[1] = std::make_pair(-0.3399810435848563, 0.6521451548625461);
142                 points_weights[2] = std::make_pair(0.3399810435848563, 0.6521451548625461);
143                 points_weights[3] = std::make_pair(0.8611363115940526, 0.3478548451374539);
144                 break;
145             case 5:
```

Figura 5: Implementação do método de Gauss em C++.

A Figura 6 apresenta a continuação do método "Cal_points_weights", que retorna os pontos e os pesos (linha 158) de acordo com o caso específico, ou seja, o número de pontos configurado pelo usuário.

```
145             case 5:
146                 points_weights.resize(5);
147                 points_weights[0] = std::make_pair(-0.9061798459386640, 0.2369268850561891);
148                 points_weights[1] = std::make_pair(-0.5384693101056831, 0.4786286704993665);
149                 points_weights[2] = std::make_pair(0.0, 0.5688888888888889);
150                 points_weights[3] = std::make_pair(0.5384693101056831, 0.4786286704993665);
151                 points_weights[4] = std::make_pair(0.9061798459386640, 0.2369268850561891);
152                 break;
153             default:
154                 std::cerr << "Número de puntos no soportado" << std::endl;
155                 return points_weights;
156     }
157 
158     return points_weights;
159 }
```

Figura 6: Implementação do método de Gauss em C++.

A Figura 7 exibe a implementação do método "IntegrateGauss" (linha 162), que recebe como argumentos os limites de integração, x1 e x2, e calcula a integral utilizando a equação

4. A linha 179 da figura demonstra como este método é aplicado em função do número de refinamentos.

```
161 //Implemetation based on the number of points
162 double NumericalIntegration::IntegrateGauss(double x1, double x2, int np){
163 
164     std::vector<std::pair<double, double> > pointsw =Calc_points_weights(np);
165     double area = 0;
166     for (int i = 0; i< np; i++) {
167         double xi =pointsw[i].first;
168         double wi =pointsw[i].second;
169         double xi_t = 0.5*(xi+1.0)*(x2 - x1) + x1;
170         double wi_t = 0.5*(wi)*(x2 - x1);
171         area += funcion(xi_t)*wi_t;
172     }
173     return area;
174 }
175 
176 //Implementation according to the number of refinements
177 
178 //(a, b = Integration limits); (ref = Domain refinements); (np = Number of points)
179 std::vector<double> NumericalIntegration::IntegrateGauss(double a, double b, int ref, int np){
180 
181     int nel = pow(2, ref);
182     std::vector<double> vectorAreas(nel,0);
183     double h = (b-a)/nel;
184     for (int i=0; i< nel; i++) {
185         double point1 = a + i*h;
186         double point2 = a + (i+1)*h;
187         double area = IntegrateGauss(point1, point2, np);
188         vectorAreas[i]=area;
189     }
190     fsol =vectorAreas;
191     return vectorAreas;
192 }
```

Figura 7: Implementação do método de Gauss em função do número de refinamentos em C++.

3 Avaliação dos métodos de integração

Os métodos númericos implementados na seção 2 são avaliados empregando as funções mostradas na tabela 2.

Tabela 2: Funções a serem avaliadas utilizando métodos numéricos.

Função	Definição
1	$f(x) = e^{-\frac{(x-1)^2}{\xi}}; \quad x \in R \mid 0 \leq x \leq 2, \xi \to 0$
2	$f(x) = xSin(\frac{1}{x}); \quad x \in R \mid 1/100 \leq x \leq 1/10$
3	$f(x) = \begin{cases} 2x+5 & 0 \leq x < 1/\pi \\ \frac{-5\pi^2(x^2-2x-5)+10\pi x+2x}{1+5\pi^2} & 1/\pi \leq x < 2/\pi \\ 2Sin(2x) + \frac{4+20\pi^2+25\pi^3}{\pi+5\pi^3} - 2Sin(\frac{4}{\pi}) & 2/\pi \leq x \leq 8/\pi \end{cases}$

Os resultados obtidos utilizando os métodos implementados para a integração da função 1, considerando $\xi = 0.1$, são apresentados na tabela 3.

Tabela 3: Métodos de Trapezio e Gauss-Legendre: Erro e Taxa de Convergência da Função 1.

Função 1				
nref	M. Trapézios		M. Gauss-Legendre	
	$\| \boldsymbol{e} \|$	$\boldsymbol{p}$	$\| \boldsymbol{e} \|$	$\boldsymbol{p}$
0	0,5604	- - -	0,14951	- - -
1	0,43955	0,35044	0,002158	6,1144
2	0,021613	4,3461	0,000022882	6,5593
3	7,1549E-06	11,561	2,1864E-07	6,7096
4	2,3928E-06	1,5803	2,1899E-07	-0,0022996
5	7,9735E-07	1,5854	2,1899E-07	-5,1002E-06
6	3,6596E-07	1,1235	2,1899E-07	-1,8286E-08
7	2,5588E-07	0,51621	2,1899E-07	7,3142E-10
8	2,2822E-07	0,16505	2,1899E-07	-5,8514E-09

O erro e a taxa de convergência da Função 1 utilizando os métodos de Simpson são apresentados na tabela 4.

Tabela 4: Erro e taxa de convergência para a Função 1 com métodos de Simpson.

Função 1				
nref	**M. Simpson 1/3**		**M. Simpson 3/8**	
	$\| e \|$	p	$\| e \|$	p
0	0,77287	- - -	0,066683	- - -
1	0,1177	2,7151	0,054781	0,28363
2	0,0072138	4,0282	0,0027059	4,3395
3	8,0539E-07	13,129	4,8966E-07	12,432
4	2,6555E-07	1,6007	2,3996E-07	1,029
5	2,2216E-07	0,25734	2,204E-07	0,12265
6	2,1919E-07	0,019441	2,1908E-07	0,0087071
7	2,19E-07	0,0012525	2,1899E-07	0,00055728
8	2,1899E-07	0,000078772	2,1899E-07	0,00003502

A tabela a seguir mostra o erro e a taxa de convergência utilizando o método do trapézio e o método de Gauss-Legendre para a função 2.

Tabela 5: Métodos de Trapezio e Gauss-Legendre: Erro e Taxa de Convergência da Função 2.

Função2				
nref	**M. Trapézios**		**M. Gauss-Legendre**	
	$\| e \|$	p	$\| e \|$	p
0	0,0035749	- - -	0,0013856	- - -
1	0,0037694	-0,076459	0,002168	-0,64578
2	0,0021984	0,77787	0,0029882	-0,46295
3	0,003204	-0,54339	0,003602	-0,2695
4	0,0033113	-0,047557	0,0035379	0,025907
5	0,0038118	-0,20305	0,0039842	-0,1714
6	0,0040362	-0,082535	0,0041166	-0,04716
7	0,0041386	-0,03615	0,0041788	-0,021642
8	0,0041992	-0,020957	0,0042299	-0,017522

A tabela 6 apresenta o erro e a taxa de convergência da função 2.

Tabela 6: Erro e taxa de convergência para a Função 2 com métodos de Simpson.

Função 2				
nref	**M. Simpson 1/3**		**M. Simpson 3/8**	
	$\| e \|$	p	$\| e \|$	p
0	0,0038343	- - -	0,0023877	- - -
1	0,0018848	1,0245	0,0012638	0,91785
2	0,0036025	-0,93457	0,0029692	-1,2323
3	0,0038274	-0,087377	0,0034307	-0,20845
4	0,0036618	0,063787	0,0035666	-0,056009
5	0,0040417	-0,1424	0,003987	-0,16079
6	0,0041495	-0,037971	0,004116	-0,04592
7	0,0041845	-0,012102	0,004181	-0,022601
8	0,0042304	-0,015762	0,0042301	-0,016861

A tabela seguinte mostra o erro e a taxa de convergência utilizando o método do trapézio e o método de Gauss-Legendre para a integração da função 3.

Tabela 7: Métodos de Trapezio e Gauss-Legendre: Erro e Taxa de Convergência da Função 2.

Função 3				
nref	**M. Trapézios**		**M. Gauss-Legendre**	
	$\| e \|$	p	$\| e \|$	p
0	2,2109	- - -	0,018279	- - -
1	0,071106	4,9585	0,00055852	5,0325
2	0,010245	2,795	0,000041554	3,7485
3	0,0026488	1,9516	0,000041554	-0,000011148
4	0,00069674	1,9266	0,000041554	-4,1752E-08
5	0,00020556	1,7611	0,000041554	-1,8502E-10
6	0,000082569	1,3159	0,000041554	0
7	0,000051809	0,6724	0,000041554	1,8502E-10
8	0,000044118	0,23183	0,000041554	-3,0836E-10

Os resultados obtidos utilizando os métodos de Simpson para a integração da função 3 são mostrados na tabela8.

Tabela 8: Erro e taxa de convergência para a Função 3 com métodos de Simpson.

Função 3				
nref	**M. Simpson 1/3**		**M. Simpson 3/8**	
	$\| e \|$	p	$\| e \|$	p
0	0,64215	- - -	0,27257	- - -
1	0,010041	5,9989	0,00027756	9,9396
2	0,00011657	6,4286	0,000074715	1,8933
3	0,000046072	1,3392	0,00004356	0,77841
4	0,000041834	0,13922	0,000041679	0,063684
5	0,000041572	0,0090766	0,000041562	0,0040406
6	0,000041555	0,00056784	0,000041555	0,00025238
7	0,000041554	0,000035476	0,000041554	0,000015767
8	0,000041554	2,2171E-06	0,000041554	9,8539E-07

3.1 Convergência do Método de Gauss-Legendre

A seguir estão os resultados obtidos utilizando o método de Gauss-Legendre com 4 pontos para as funções mostradas na tabela 2

A figura 8 mostra a taxa de convergência da função 1, onde se evidencia que, nos últimos 5 trechos, a integração utilizando este método é exata.

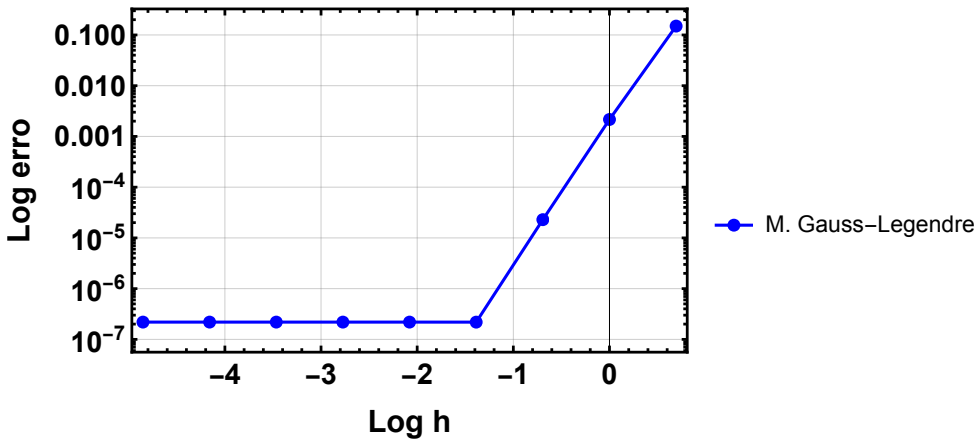

Figura 8: Taxa de Convergência para a Função 1.

A seguinte figura permite observar que, para a função 2, a aproximação obtida utilizando este método não é a solução exata, isso se deve às características da função e à sua natureza oscilatória.

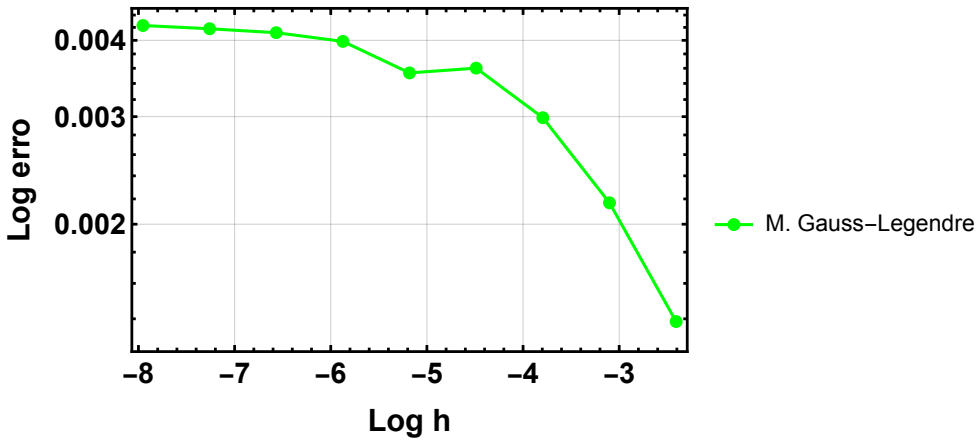

Figura 9: Erro númerico para a funçao 1

A figura 10 mostra a taxa de convergência da função 3. Pode-se observar que nos últimos 6 trechos a solução obtida utilizando este método é a solução exata.

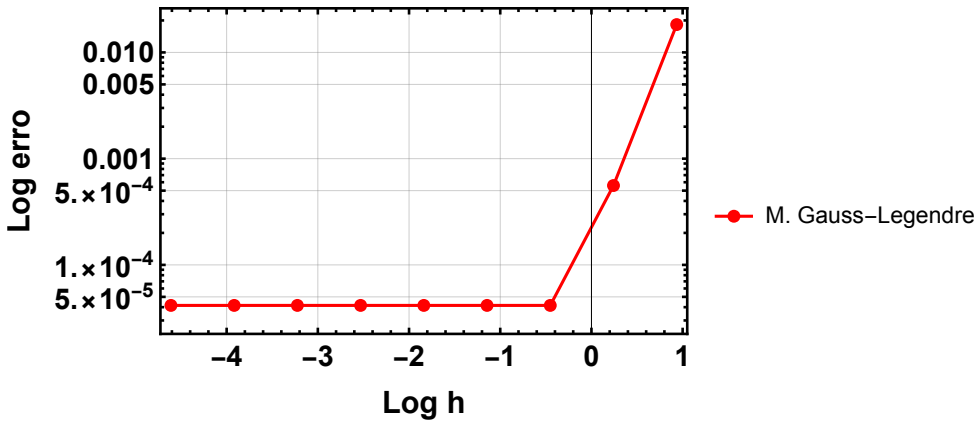

Figura 10: Taxa de Convergência para a Função 1.

3.2 Análise dos Resultados

A figura 11 mostra uma comparação dos métodos de integração numérica aplicados à função 1. Pode-se observar que o método de Gauss-Legendre supera os demais métodos, requerendo apenas 3 refinamentos para convergir. No entanto, é importante considerar que todos os

métodos tendem a convergir até alcançar um ponto onde um maior número de refinamentos não proporciona melhorias significativas na precisão do método.

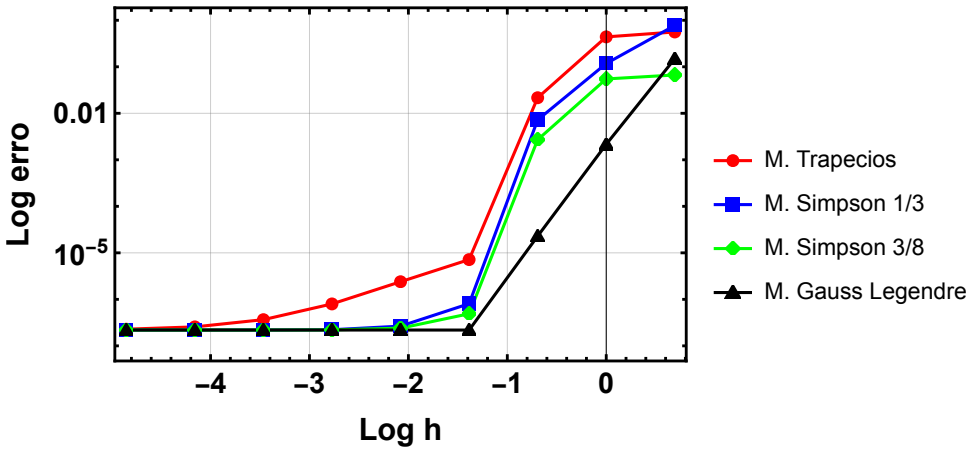

Figura 11: Erro númerico para a funçao 1.

A figura 12 mostra mudanças significativas no erro para os métodos devido à natureza oscilatória da função quando $x \to 0$, como é observado para o método de Simpson 3/8. Isso também pode ser consequência da instabilidade do método ao calcular o erro usando esta função. Além disso, observa-se que o método de Gauss-Legendre apresenta mais estabilidade em função do número de intervalos.

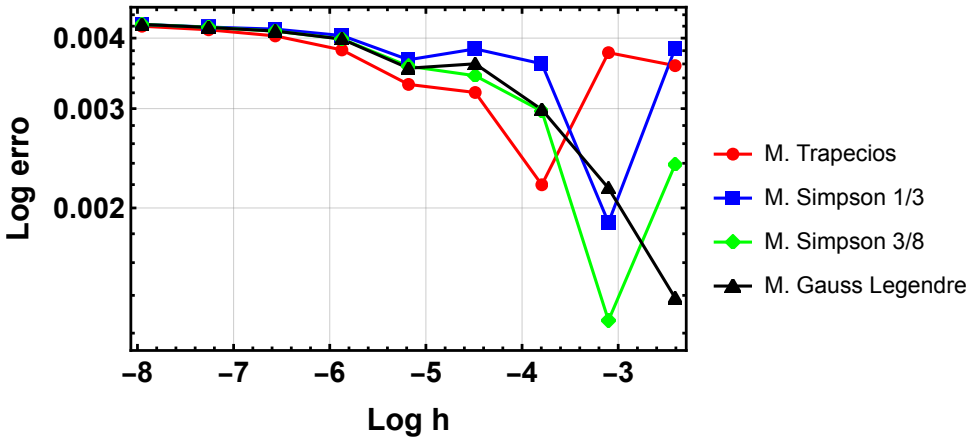

Figura 12: Erro numérico da funçao 2.

A figura 13 fornece uma comparação dos métodos de integração para a função 3. Observa-se que o Método dos Trapézios apresenta uma redução do erro proporcional ao número de refinamentos, no entanto, é o método menos eficiente para esta função, pois requer um maior número de refinamentos para obter uma aproximação com erro baixo. Por outro lado, os métodos de Simpson mostram uma diminuição mais acentuada do erro em função do número de refinamentos em comparação com o método dos trapézios. Também pode-se observar que o Método de Gauss-Legendre se destaca entre os demais métodos por requerer apenas 3 refinamentos para obter um erro baixo, o que indica sua rápida convergência para esta função. Deve-se considerar que o método de Gauss obteve uma rápida convergência devido ao uso de 4 pontos.

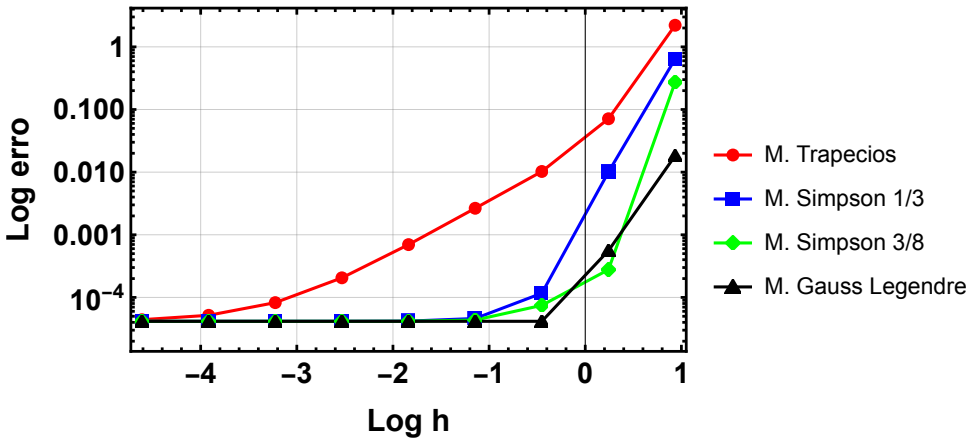

Figura 13: Erro númerico para a funçao 3.

4 Conclusões

1. Os resultados indicam que, à medida que o tamanho dos intervalos diminui, o erro tende a diminuir, o que é consistente com a expectativa de obter maior precisão em domínios mais refinados. O método de Gauss-Legendre mostrou maior eficiência em relação ao comportamento das funções integradas, mantendo um erro baixo em função do número de refinamentos. Isso indica que, para funções com comportamentos oscilatórios, este método é recomendado devido à sua maior estabilidade em comparação com os outros métodos de integração numérica implementados.

2. A expectativa de obter maior eficiência dos métodos de Simpson em comparação com o método dos trapézios foi confirmada. Isso é evidenciado pela obtenção de um erro menor em função do número de refinamentos utilizados.

3. Para a seleção do método de integração adequado, não apenas a precisão esperada deve ser considerada, mas também as características da função a ser integrada. Uma avaliação detalhada da natureza da função utilizada pode fornecer resultados confiáveis, especialmente no contexto de funções com comportamentos oscilatórios.

5 Referências Bibliográficas

- Davis, P. J., & Rabinowitz, P. (2007). Methods of numerical integration. Courier Corporation.
- Smyth, G. K. (1998). Numerical integration. Encyclopedia of Biostatistics, 3088–3095.