Trabalho Prático de Cálculo Numérico

Lucas Marins Ramalho de Lima

Jonatas Dias Machado Costa

Resolução de sistemas não-lineares para o cálculo de pontos de integração da quadratura de Gauss-Legendre

Universidade Federal de Juiz de Fora

Juiz de Fora - Brasil 2022/2023

INTRODUÇÃO

O método de integração numérica conhecido como Quadratura de Gauss-Legendre, ou simplesmente, Quadratura de Gauss busca obter uma aproximação para a integral de uma função em determinado intervalo $[a, b]$ através da somatória dos fatores dos pontos de integração utilizados para a aproximação com pesos calculados pelo método. Essa aproximação é descrita da seguinte forma:

$$\int_a^b f(t)\,dx \approx I = w_0 \times f(t_0) + \cdots + w_n \times f(t_n) \tag{1}$$

Para o cálculo dos pesos w_i e dos pontos de integração t_i, é necessário começar o método com um chute inicial de N valores para ambos. No presente estudo, o chute inicial busca uma distribuição que cubra de maneira eficiente o intervalo $[a, b]$ e é calculado conforme descrito a seguir:

$$w_i = \frac{(b-a)\times i}{2N} \qquad \text{para i = 1,2,...,N/2} \tag{2}$$

$$w_i = w_{N/2-1} \qquad \text{para i > N/2} \tag{3}$$

$$t_i = \frac{a + i \times w_i}{2} \qquad \text{para i = 1,2,...,N/2} \tag{4}$$

$$t_i = (a+b) - t_{N/2-1} \qquad \text{para i > N/2} \tag{5}$$

Os valores de w e t são calculados iterativamente a partir da solução de um sistema de equações não lineares generalizado pela fórmula:

$$\sum_{i=1}^{\mathrm{N}} w_i (t_i)^{j-1} - \int_a^b x^{j-1}\,dx = 0 \qquad \text{para j=1,2,...,2N} \tag{6}$$

Nesse estudo, as integrais do sistema não linear foram calculadas numéricamente através da regra de Newton-Cotes utilizando a fórmula repetida de Simpson 3/8:

$$\int_a^b f(x)\,dx \approx \frac{3}{8} \times h[f(x_0) + 3f(x_1) + 3(x_2) + f(x_3)] \tag{7}$$

A resolução do sistema não linear pode ser feita pela aplicação do método de Newton-Raphson que estima raízes de uma função através de sua derivada. Para utilizar esse método no presente estudo, foi construída uma matriz Jacobiana $J(w,t)$ com as derivadas parciais de $f(w^k,t^k)$ de w_i^k e t_i^k da seguinte forma:

$$J(w,t) = \begin{bmatrix} \frac{\partial f_1}{\partial w_1} & \cdots & \frac{\partial f_1}{\partial w_N} & \frac{\partial f_1}{\partial t_1} & \cdots & \frac{\partial f_1}{\partial t_N} \\ \frac{\partial f_2}{\partial w_1} & \cdots & \frac{\partial f_2}{\partial w_N} & \frac{\partial f_2}{\partial t_1} & \cdots & \frac{\partial f_2}{\partial t_N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_2 N}{\partial w_1} & \cdots & \frac{\partial f_2 N}{\partial w_N} & \frac{\partial f_2 N}{\partial t_1} & \cdots & \frac{\partial f_2 N}{\partial t_N} \end{bmatrix} \tag{8}$$

Com a matriz Jacobiana montada de acordo com a estrutura descrita anteriormente, é possível construir o sistema linear:

$$J(w,t) \times s(w,t) = -f(w,t) \tag{9}$$

O vetor s de $2N$ coordenadas, possui os valores a serem somados aos vetores w e t, sendo que a primeira metade do vetor s é somada a w e a segunda metade ao vetor t. No estudo presente, o critério de parada escolhido foi a norma do máximo do vetor s.

OBJETIVO

O presente estudo tem como objetivo a criação de um código computacional para calcular os pontos de integração da quadratura de Gauss para qualquer intervalo assim como os pesos associados a cada um dos pontos.

A partir dos pontos de integração e seus respectivos pesos, serão calculadas algumas integrais numericamente para diferentes domínios $[a, b]$.

Esse relatório contemplará os dados utilizados nas simulações e tabelas com os resultados dos codigo computacional.

Chapter 1

O Código

1.1 Linguagem

O código computacional para a resolução numérica de integrais utilizando o método da quadratura de Gauss exige eficiência e otimização. Por envolver um método de resolução de sistemas lineares em uma das etapas do processo, a escolha da linguagem precisa visar velocidade de execução. Dessa forma o código desse estudo foi construído com a linguagem $C++$.

O $C++$ é uma linguagem que possui pré-processamento e pré-compilação, o que garante a ela uma maior velocidade de execução. Em um comparativo entre $C++$ e uma linguagem que é compilada em tempo de execução, *Python*, pode-se observar um rendimento 20 vezes melhor utilizando a primeira opção em um método de resolução de sistemas lineares.

1.2 Algoritmos

O processo descrito na **Introdução** foi feito no código e seccionado em funções definidas por etapas do processo. A primeira etapa é a construção do chute inicial:

Algorithm 1 Chute Inicial

1: **for** $i = 0, \ldots, N$ **do**
2: $w[i] \leftarrow (a-b) * (i+1)/2 \times N$
3: $t[i] \leftarrow (a + (i+1) \times (w[i]/2))$
4: $w[N-1-i] \leftarrow w[i]$
5: $t[N-1-i] \leftarrow (a+b) - t[i]$
6: **end for**
7: **if** N é ímpar **then**
8: $t[N/2] \leftarrow (a+b)/2$
9: $w[N/2] \leftarrow ((b-a) \times N/2)/2N$
10: **end if**

Nesse código, chama-se de N o número de pontos de integração pretendidos, de w o vetor de pesos para cada um dos pontos de integração e de t o vetor de valores das absissas.

Depois do cálculo dos valores dos vetores w e t, é feito o cálculo da integral $\int_a^b x^{j-1}\,dx$ que será apelidada de g no código. O cálculo dessa integral pode ser feito de duas maneiras: utilizando um método de aproximação numérica Newton-Cotes, ou calculando a integral de maneira exata. Nesse caso, é possível calcular de maneira exata a integral, por tratar-se de uma integral de polinômio.

Algorithm 2 Newton-Cotes: Simpson 3/8 (repetida)

1: $I \leftarrow 0$
2: $h \leftarrow (b-a)/nparticoes$
3: **for** $i = 0, 3, 6, \ldots, nparticoes$ **do**
4: $\quad I \leftarrow I + (3 \times h/8) \times [(a + i * h)^{pot} + 3 \times (a + (i+1) * h)^{pot} + 3 \times (a + (i + 2) * h)^{pot} + (a + (i+3) * h)^{pot}]$
5: **end for**
6: return I

Nessa função, são passados como parâmetros os limites a e b e a potência $j-1$ que está sendo cálculada.

Algorithm 3 Integral exata

1: return $\frac{b^{pot+1}}{pot+1} - \frac{a^{pot+1}}{pot+1}$

Os parâmetros da função imediatamente acima são os mesmos passados para a função do método numérico de Newton-Cotes descrito anteriormente.

O cálculo da integral exata previne alguns erros de ponto flutuante, que se fazem presentes no método numérico de Simpson 3/8. Essa variação de ponto flutuante, pode ser observada na tabela abaixo, que reúne resultados do cálculo da integral g pelos dois métodos para cada valor de j.

Table 1.1: Comparação de resultados para a solução da integral g no intervalo [-1,1]

j	Exata	Simpson 3/8 (1500 partições)
1	2	2
2	0	4.66E-20
3	0.666667	0.666667
4	0	8.98E-20
5	0.4	0.4
6	0	7.54E-20
7	0.285714	0.285714
8	0	8.22E-20
9	0.222222	0.222222
10	0	7.75E-20

Nesse caso particular de cálculo de integral, a solução exata aparenta ser mais adequada por não exigir um grande custo computacional e apresentar o resultado exato, portanto nesse estudo será proposta uma comparação entre os resultados finais obtidos através do cálculo numérico e do cálculo exato.

Com os vetores w, t e g preenchidos de maneira adequada, o próximo passo do código é cacular os valores de f_j conforme a equação:

$$f_j(w,t) = \sum_{i=1}^{\mathrm{N}} w_i (t_i)^{j-1} - g_j \tag{1.1}$$

O último passo para o cálculo dos pontos de integração e pesos é a execução do método de Newton-Rapshon:

Algorithm 4 Newton-Rapshon

```
1: while solution(w, t) > TOL do
2:     funtion ← −f(w, t, g)
3:     j ← Jacobiana(w, t, g)
4:     solution ← metodoSisLinear(j, fntion, 2)
5:     for i=0,...,N do
6:         w[i] = w[i] + solution[i]
7:         t[i] = t[i] + solution[i]
8:     end for
9: end while
10: return w, t
```

O cálculo da matriz Jacobiana necessita da aproximação das derivadas parciais

$\frac{\partial f_j}{\partial w_i}$ e $\frac{\partial f_j}{\partial t_i}$ que será feita através do cálculo:

$$\frac{\partial f_j}{\partial w_i} \approx \frac{f_j(w_1, w_2, \ldots, w_i + \varepsilon, \ldots, w_N, t_1, \ldots, t_N) - f_j(w_1, \ldots, w_N, t_1, \ldots, t_N}{\varepsilon} \tag{1.2}$$

$$\frac{\partial f_j}{\partial t_i} \approx \frac{f_j(w_1, \ldots, w_N, t_1, t_2, \ldots, t_i + \varepsilon, \ldots, t_N) - f_j(w_1, \ldots, w_N, t_1, \ldots, t_N}{\varepsilon} \tag{1.3}$$

Após a execução do código, o resultado obtido nos vetores w e t são armazenados em um arquivo do tipo *.csv* que pode ser aberto em formato de uma planilha.

Para uma boa precisão de ponto flutuante, será utilizado o tipo de dado "long double" primitivo do $C++$. Esse tipo de dado utiliza estrutura de 16 bytes para representar números reais.

Chapter 2

Execução e resultados

Executando o código, inicialmente no intervalo [-1, 1] com $N = 5$ pontos de integração pode-se fazer uma comparação de resultados obtidos utilizando o método numérico de Simpson 3/8 e o cálculo exato para cálcular a integral $g = \int_a^b x^{j-1}\, dx$.

Table 2.1: Valores dos pontos de integração t e seus respectivos pesos w utilizando o método numérico de Newton-Cotes Simpson 3/8 repetido com 1500 partições

w	t
0.236927	-0.90618
0.478629	-0.538469
0.568889	-6.26535E-19
0.478629	0.538469
0.236927	0.90618

Table 2.2: Valores dos pontos de integração t e seus respectivos pesos w utilizando o cálculo da integral g de maneira exata

w	t
0.236927	-0.90618
0.478629	-0.538469
0.568889	3.12219E-19
0.478629	0.538469
0.236927	0.90618

Ambos os métodos necessitaram de 5 iterações do método Newton-Rapshon para alcançar o critério de parada $\|solution(w,t)\|_\infty \leq TOL$ com $TOL = 10^{-8}$. Dessa forma, ambos os métodos de cálculo de g entregam valores suficientemente próximos para o cálculo de w e t. O estudo foi continuado usando o método numérico de Newton-Cotes.

Executando o código para o cálculo de N=5 pontos de integração, variando

o intervalo [a,b] e utilizando $\|solution(w,t)\|_\infty \leq TOL$ com $TOL = 10^{-8}$ como critério de parada, obtemos os seguintes resultados:

Table 2.3: Valores dos pontos de integração t e seus respectivos pesos w no intervalo [-2,2]

w	t
0.473854	-1.81236
0.957257	-1.07694
1.13778	5.00925E-19
0.957257	1.07694
0.473854	1.81236

Table 2.4: Valores dos pontos de integração t e seus respectivos pesos w no intervalo [-3,3]

w	t
0.710781	-2.71854
1.43589	-1.61541
1.70667	1.83711E-18
1.43589	1.61541
0.710781	2.71854

Table 2.5: Valores dos pontos de integração t e seus respectivos pesos w no intervalo [-10,10]

w	t
2.36927	-9.0618
4.78629	-5.38469
5.68889	5.5501E-18
4.78629	5.38469
2.36927	9.0618

Table 2.6: Valores dos pontos de integração t e seus respectivos pesos w no intervalo [-10,5]

w	t
1.77695	-9.29635
3.58972	-6.53852
4.26667	-2.5
3.58972	1.53852
1.77695	4.29635

Calculando os pontos de integração e os seus respectivos pesos para o intervalo $[-1, 1]$ variando N =2,. . . , 7, com critério de parada $\|solution(w, t)\|_\infty \leq TOL$ com $TOL = 10^{-8}$:

Table 2.7: Valores dos N = 2 pontos de integração t e seus respectivos pesos w no intervalo [-1,1]

w	t
1	-0.577350269189626
1	0.577350269189626

Table 2.8: Valores dos N = 3 pontos de integração t e seus respectivos pesos w no intervalo [-1,1]

w	t
0.555555555555556	-0.774596669241483
0.888888888888889	-4.31039784069857E-18
0.555555555555556	0.774596669241483

Table 2.9: Valores dos N = 4 pontos de integração t e seus respectivos pesos w no intervalo [-1,1]

w	t
0.347854845137454	-0.861136311594053
0.652145154862546	-0.339981043584856
0.652145154862546	0.339981043584856
0.347854845137454	0.861136311594053

Table 2.10: Valores dos N = 5 pontos de integração t e seus respectivos pesos w no intervalo [-1,1]

w	t
0.236926885056189	-0.906179845938664
0.478628670499367	-0.538469310105683
0.568888888888889	3.12219031339347E-19
0.478628670499367	0.538469310105683
0.236926885056189	0.906179845938664

Table 2.11: Valores dos N = 6 pontos de integração t e seus respectivos pesos w no intervalo [-1,1]

w	t
0.17132449237917	-0.932469514203152
0.360761573048139	-0.661209386466264
0.467913934572691	-0.238619186083197
0.467913934572691	0.238619186083197
0.360761573048139	0.661209386466264
0.17132449237917	0.932469514203152

Table 2.12: Valores dos N = 7 pontos de integração t e seus respectivos pesos w no intervalo [-1,1]

w	t
0.12948496616887	-0.949107912342758
0.279705391489277	-0.741531185599394
0.381830050505119	-0.405845151377397
0.417959183673469	-1.25635091245077E-17
0.381830050505119	0.405845151377397
0.279705391489277	0.741531185599394
0.12948496616887	0.949107912342758

Chapter 3

Utilizando resultados do experimento no cálculo da Quadratura de Gauss-Legendre

Com os pontos de integração calculados no capítulo anterior e com os seus respectivos pesos, podemos aplicar o método de Quadratura de Gauss para aproximar o cálculo da integral de algumas funções. Para estabelecer comparações entre a solução exata e a aproximada, calcula-se o erro como $|exata - aproximada|$. A primeira função que será calculada, será $f(x) = \exp(a \times x + b)$ em diferentes intervalos, com $N = 1, \ldots, 7$, critério de parada $TOL = 10^{-8}$, pertubação para cálculo de derivadas na matriz Jacobiana $\varepsilon = 10^{-8}$ e 1500 partições para o cálculo da integral g com Simpson 3/8.

O primeiro intervalo a ser testado é [-1,1]:

Table 3.1: Comparação de solução exata da com solução aproximada pelo método da Quadratura de Gauss para diferentes quantidades de pontos de integração para o intervalo [-1, 1]

N	Solução Exata	Solução Aproximada	Erro
2	6.38905609893065	6.36810820536712	0.020947893563536
3	6.38905609893065	6.38887816398712	0.000177934943532
4	6.38905609893065	6.3890552966808	8.02249847886571E-07
5	6.38905609893065	6.38905609668867	2.24197638232226E-09
6	6.38905609893065	6.38905609892639	4.2632564145606E-12
7	6.38905609893065	6.38905609893064	1.00E-14

Com a mesma função $f(x) = \exp(a \times x + b)$ e paramêtros supracitados, calcula-se sua integral no intervalo [-2, 2]:

Table 3.2: Comparação de solução exata da com solução aproximada pelo método da Quadratura de Gauss para diferentes quantidades de pontos de integração para o intervalo [-2, 2]

N	exata	aproximada	erro
2	201.646729104749	150.25960903242	51.3871200723289
3	201.646729104749	195.459331807837	6.18739729691185
4	201.646729104749	201.235666672594	0.411062432155319
5	201.646729104749	201.629377214948	0.017351889800921
6	201.646729104749	201.646222936565	0.000506168184216
7	201.646729104749	201.646718299877	1.08048719482667E-05

Mantendo a mesma função e os mesmos parâmetros das tabelas anteriores, mas alterando o intervalo para [-3, 2]:

Table 3.3: Comparação de solução exata da com solução aproximada pelo método da Quadratura de Gauss para diferentes quantidades de pontos de integração para o intervalo [-3, 2]

N	exata	aproximada	erro
2	19958.0411331863	6289.21207807342	13668.8290551129
3	19958.0411331863	15410.0468444919	4547.99428869445
4	19958.0411331863	19072.3990285216	885.642104664668
5	19958.0411331863	19843.1899402444	114.851192941922
6	19958.0411331863	19947.4070513815	10.6340818048557
7	19958.0411331863	19957.3041433806	0.736989805700432

Obs: Observa-se que para todos os intervalos testados, conforme o número de pontos de integração aumenta, a grandeza do erro diminui. Isso se deve a uma melhor cobertura de pontos de integração pelo intervalo.

Calculando a integral da função $h(x) = sen(a \times x + b) + cos(b \times x + a)$ em diferentes intervalos [a,b], variando N = 1, ..., 7, com critério de parada $TOL = 10{-}8$, com pertubação no cálculo das derivadas parciais da matriz Jacobiana $\varepsilon = 10^{-8}$ e 1500 partições para o cálculo da integral g com Simpson 3/8.

Para o intervalo [-1, 1], obtem-se os seguintes resultados:

Table 3.4: Comparação de solução exata da com solução aproximada pelo método da Quadratura de Gauss para diferentes quantidades de pontos de integração para o intervalo [-1, 1]

N	Solução Exata	Solução Aproximada	erro
2	2.32544426337282	2.31560836690097	0.009835896471859
3	2.32544426337282	2.32552935036206	8.50869892410344E-05
4	2.32544426337282	2.32544387520532	3.88167503473369E-07
5	2.32544426337282	2.32544426446636	1.09353415211899E-09
6	2.32544426337282	2.32544426337073	2.09166017839379E-12
7	2.32544426337282	2.32544426337283	1.00E-14

Com os mesmos parâmetros da última tabela, porém alterando o intervalo da função $h(x)$ para [-2, 2]:

Table 3.5: Comparação de solução exata da com solução aproximada pelo método da Quadratura de Gauss para diferentes quantidades de pontos de integração para o intervalo [-2, 2]

N	Solução Exata	Solução Aproximada	erro
2	-0.373217597285376	-1.32806816508159	0.954850567796216
3	-0.373217597285376	-0.218155319204586	0.155062278080791
4	-0.373217597285376	-0.385665698530798	0.012448101245422
5	-0.373217597285376	-0.372619781940795	0.000597815344581
6	-0.373217597285376	-0.373236738338926	1.9141053549887E-05
7	-0.373217597285376	-0.373217158940093	4.38345283404473E-07

Da mesma forma que a função calculada anteriormente, o erro entre a solução exata e aproximada da integral de $h(x)$ diminui conforme o número de pontos de integração aumenta.

Chapter 4

Conclusão

Ao término do estudo sobre o cálculo dos valores de pontos de integração e seus respectivos pesos para integrar uma função $f(x)$ utilizando o método de Quadratura de Gauss-Legendre, é possível concluir sobre a eficácia da aproximação do método.

Nos testes executados durante essa experimentação, observa-se que para intervalos pequenos a aproximação do método é boa o suficiente para oferecer uma aproximação com erro inferior a 10^{-8} com apenas 7 pontos de integração. Para diversas aplicações cotidianas, tal resultado já pode ser utilízado como um resultado prático para a integração da função $f(x)$.

KAMERMANS, Mike "pomax". Gaussian Quadrature Weights and Abscissae. Pomax, 2011. Disponível em: https://pomax.github.io/bezierinfo/legendre-gauss.html. Acesso em: 07 jan. 2023.

P., Davis; P., Rabinowitz. Abscissas and Weights for Gaussian Quadratures of High Order. Research Paper 2645. Washington: Journal of Research of the National Bureau of Standards, 1956. v. 56.