EP2 - Fórmulas de Integração Numérica de Gauss

MAP3121- Métodos Numéricos e Aplicações

Arthur Pedroso Porto Belli - 11804608
Letícia Harumi Furusawa - 11965585

10 de junho de 2022

1 Introdução

Este relatório descreve os resultados obtidos no exercício programa, no qual estudamos as fórmulas de integração numérica de Gauss para o caso de integrais duplas, assim como a resulção dos exercícios e exemplos propostos.

O grupo optou pela implementação em Python 3.9.7 e utilizou apenas a biblioteca Numpy 1.21.1, além de uma coletânea de funções próprias (*utils.py*).

2 Exercícios propostos

Para um polinômio genérico de grau até 5: $f(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + c_3 \cdot x^3 + c_4 \cdot x^4 + c_5 \cdot x^5$, sabemos que a integral exata dele no intervalo $[-1, 1]$ será dada por:

$$I = \int_{-1}^{1} (c_0 + c_1 \cdot x + c_2 \cdot x^2 + c_3 \cdot x^3 + c_4 \cdot x^4 + c_5 \cdot x^5)dx = 2c_0 + \frac{2}{3}c_2 + \frac{2}{5}c_4$$

visto que os termos de expoente ímpar são funções ímpares e em um intervalo simétrico têm integral nula.

Agora, podemos supor que podemos realizar o cálculo aproximado dessa integral por meio da soma ponderada de certos nós da curva. Como fornecido pelo enunciado, precisaremos de $n = 3$ nós, assim como $n = 3$ pesos. Denotando por J o cálculo aproximado, ω_i os pesos e p_i os nós, temos que:

$$J = \omega_1 f(p_1) + \omega_2 f(p_2) + \omega_3 f(p_3)$$

Substituindo os valores na expressão genérica do polinômio de grau até 5 e igualando à integral exata, obtemos o seguinte sistema de 6 equações e 6 incognitas:

$$\begin{pmatrix} \omega_1 + \omega_2 + \omega_3 \\ \omega_1 p_1 + \omega_2 p_2 + \omega_3 p_3 \\ \omega_1 p_1^2 + \omega_2 p_2^2 + \omega_3 p_3^2 \\ \omega_1 p_1^3 + \omega_2 p_2^3 + \omega_3 p_3^3 \\ \omega_1 p_1^4 + \omega_2 p_2^4 + \omega_3 p_3^4 \\ \omega_1 p_1^5 + \omega_2 p_2^5 + \omega_3 p_3^5 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ \frac{2}{3} \\ 0 \\ \frac{2}{5} \\ 0 \end{pmatrix}$$

É da resolução desse sistema que se obtém os valores tabelados indicados anteriormente no enunciado, e de onde se conclui que, para um polinômio genérico de grau menor ou igual a 5, a igualdade expressa na equação abaixo é válida:

$$I = J = \int_{-1}^{1} f(x)dx = \frac{5}{9}f(-\sqrt{0.6}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{0.6})$$

Caso desejarmos realizar essa integração em um intervalo genérico $[a, b]$, transportamos linearmente esse intervalo por meio de uma transformação afim que leva $x \in [-1, 1]$ até $t \in [a, b]$. Rapidamente, encontramos a expressão $t = a + \frac{(b-a)}{2}(x+1)$. Assim, tem-se que

$$\int_{-1}^{1} f(x)dx = \int_{a}^{b} f(t(x))\frac{2}{b-a}dt$$

uma vez que $dx = \frac{2}{(b-a)}dt$. Finalmente,

$$\int_{a}^{b} f(t)dt = \frac{b-a}{2} \cdot \int_{-1}^{1} f(x)dx = \frac{b-a}{2} \cdot \left(\sum_{i=1}^{n} \omega_i \cdot f(p_i)\right)$$

Portanto, podemos reescrever essa última expressão como:

$$I = \sum_{i=1}^{n} \left(\frac{b-a}{2} \cdot \omega_i\right) \cdot f(p_i'),\ \ p_i' = a + \frac{b-a}{2}(p_i + 1)$$

De onde se evidencia o fator de escala que deve ser utilizado para modificar os pesos tabelados do intervalo $[-1, 1]$ e a transformação linear que leva o intervalo $[-1, 1]$ ao intervalo $[a, b]$ aplicada aos nós.

Assim, a expressão exata para o cálculo da integral numérica de um polinômio de grau menor ou igual a 5 no intervalo $[a, b]$ é dada por:

$$I = J = \int_{a}^{b} f(x)dx = \frac{5 \cdot (b-a)}{9} f\left(a + \frac{b-a}{2}(1-\sqrt{0.6})\right) + \frac{8 \cdot (b-a)}{9} f\left(a + \frac{b-a}{2}\right) + \frac{5 \cdot (b-a)}{9} f\left(a + \frac{b-a}{2}(1+\sqrt{0.6})\right)$$

3 Funções utilizadas

3.1 find_lin_scale_transp(a, b)

Calcula os fatores de correção dos pesos e dos nós a partir dos extremos de integração a e b.

```
def find_lin_scale_transp(a, b):
    return np.double((b-a)/2), np.double((a+b)/2)
```

3.2 x_w(n)

Função entrega os valores dos nós e dos pesos para o número n passado como parâmetro.

```
def x_w(n):
    if n == 6:
        x = [-0.9324695142031520278123016,-0.6612093864662645136613996,-0.
                                                    2386191860831969086305017,
             0.2386191860831969086305017,0.6612093864662645136613996,0.
                                                         9324695142031520278123016]
        w = [0.1713244923791703450402961,0.3607615730481386075698335,0.
                                                    4679139345726910473898703,
             0.4679139345726910473898703,0.3607615730481386075698335,0.
                                                         1713244923791703450402961]

    if n == 8 :
        x = [-0.9602898564975362316835609,-0.7966664774136267395915539,-0.
                                                    5255324099163289858177390,
             -0.1834346424956498049394761,0.1834346424956498049394761,0.
                                                         5255324099163289858177390,
             0.7966664774136267395915539,0.9602898564975362316835609]
        w = [0.1012285362903762591525314,0.2223810344533744705443560,0.
                                                    3137066458778872873379622,
             0.3626837833783619829651504,0.3626837833783619829651504,0.
                                                         3137066458778872873379622,
             0.2223810344533744705443560,0.1012285362903762591525314]

    if n == 10:
        x = [-0.9739065285171717200779640,-0.8650633666889845107320967,-0.
                                                    6794095682990244062343274,
             -0.4333953941292471907992659,-0.1488743389816312108848260,0.
                                                         1488743389816312108848260,
             0.4333953941292471907992659,0.6794095682990244062343274,0.
                                                         8650633666889845107320967,0.
                                                         9739065285171717200779640]
        w = [0.0666713443086881375935688,0.1494513491505805931457763,0.
                                                    2190863625159820439955349,
             0.2692667193099963550912269,0.2955242247147528701738930,0.
                                                         2955242247147528701738930,
             0.2692667193099963550912269,0.2190863625159820439955349,0.
                                                         1494513491505805931457763,0.
                                                         0666713443086881375935688]

    return x, w
```

3.3 gauss(f, a, b, n, x, w)

Realiza integração numérica simples de acordo com a expressão deduzida anteriormente. Serve apenas como prova de conceito e foi implementada antes da integral dupla como etapa intermediária no desenvolvimento do Exercício-Programa. Recebe como parâmetros:

- f: função a ser integrada;
- a: extremo inferior de integração;
- b: extremo superior de integração;
- n: número escolhido de nós e pesos;
- x, w: nós e pesos no intervalo $[-1, 1]$ correspondentes.

```
def gauss(f, a, b, n, x, w):
    linear_scaling, linear_transposition = find_lin_scale_transp(a, b)
    sum = 0
    v = [linear_scaling*w[i] for i in range(n)] # pesos adaptados
    y = [(linear_scaling*x[i]+linear_transposition) for i in range(n)] # nos adaptados
    for i in range (n): # soma iterada da integral
        sum += v[i]*f(y[i])

    return sum
```

3.4 gauss_2(f, a, b, c, d, n, x, w)

Principal função do código. Realiza integral dupla iterada de acordo com as expressões obtidas anteriormente na sessão de Exercícios Propostos. Seus parâmetros são análogos aos da **gauss(f, a, b, n, x, w)**, com exceção de c, d que são os extremos de integração da integral mais interna. Podem ser tanto funções que dependem da variável a ser integrada por último, como constantes (domínio retangular). Seu funcionamento é também análogo à gauss para uma variável, a única diferença é a existência de laços *for* aninhados, por conta da dupla somatória iterada apresentada no enunciado.

```
def gauss_2(f, a, b, c, d, n, x, w):
    linear_scaling_ext, linear_transposition_ext = find_lin_scale_transp(a, b) #
                                                fatores de correcao para a integral
                                                externa
    pesos_ext = [linear_scaling_ext*w[i] for i in range(n)] # pesos da integral
                                                externa
    nos_ext = [(linear_scaling_ext*x[i]+linear_transposition_ext) for i in range(n)] #
                                                 nos da integracao externa

    sum_ext = 0
    for i in range(n): #para cada no externo, definiremos o intervalo de integracao,
                                                seus nos e seus pesos
        a_int = c(nos_ext[i]) # extremo inferior de integracao interno a cada iteracao
        b_int = d(nos_ext[i]) # extremo superior de integracao interno a cada iteracao
        linear_scaling_int, linear_transposition_int = find_lin_scale_transp(a_int,
                                                    b_int) # fatores de correcao para a
                                                     integral interna a cada iteracao
        pesos_int = [linear_scaling_int*w[i] for i in range(n)] # pesos da integral
                                                    interna
        nos_int = [linear_scaling_int*x[i]+linear_transposition_int for i in range(n)]
                                                     # nos da integral interna
        sum_int = 0
        for j in range(n):
            sum_int += pesos_int[j]*f(nos_ext[i], nos_int[j])
        sum_ext += pesos_ext[i]*sum_int

    return sum_ext
```

3.5 main()

A função main ficou reservada apenas para a interface entre o usuário e a função gauss_2. É por ela que ele escolhe a função a ser integrada a partir de uma lista de funções fornecidas, além de escolher os intervalos de integração e o número de nós e pesos a serem utilizados. Para a seleção das funções, pede-se um input de texto com a forma da função. Esse input é passado para um dicionário (estrutura de dados em Python) que retorna a referência para a função correspondente. A definição das funções, assim como do dicionário estão no arquivo "utils.py".

```
def main():

    utils.print_funcas()
    f = input("Escolha a fun   o da lista que deseja integrar: ")
    d = input("Da mesma lista de funcoes, escolha agora o extremo de integracao
                                                superior: ")
    c = input("Da mesma lista de funcoes, escolha agora o extremo de integracao
                                                inferior: ")
    b = np.double(input("O extremo de integracao superior exterior: "))
    a = np.double(input("O extremo de integracao inferior exterior: "))

    for n in [6, 8, 10]:
        I = gauss_2(utils.funcas[f], a, b, utils.funcas[c], utils.funcas[d], n, *x_w(n
                                                    ))
        print(f"n == {n}")
        print(f"int_({a})^({b}) int_({c})^({d}) {f} dydx = {I}")
```

Outras funções secundárias foram incluídas, entretanto, não serão expostas nesse relatório, visto que são de implementação trivial, ou de efeito puramente estético e são usadas em poucos casos.

4 Resultados

Com as principais funções implementadas, o grupo computou as integrais sugeridas no enunciado como exercício, obtendo valores extremamente próximos, quando não exatos, com exceção da segunda parte do exercício 2, visto que a curva $f(y) = \sqrt{1-y}$ não se comporta exatamente como uma curva suave (classe de funções de interesse desse projeto), de forma que houve certa variação nos valores da integral, conforme se variava o número de nós.

A seguir, está mostrado o *output* do programa com os resultados dos exercícios.

```
EP2 - FORMULAS DE INTEGRACAO NUMERICA DE GAUSS

ARTHUR PEDROSO PORTO BELLI NUSP: 11804608
LETICIA HARUMI FURUSAWA    NUSP: 11965585


--------- Exercicios do enunciado ---------

1.1 Volume do cubo de aresta unitaria
Para n == 6:     integral == 1.0
Para n == 8:     integral == 1.0
Para n == 10:    integral == 1.0

1.2 Volume do tetraedro de arestas unitarias
Para n == 6:     integral == 0.16666666666666666
Para n == 8:     integral == 0.16666666666666666
Para n == 10:    integral == 0.16666666666666669

2.1 Area do primeiro quadrante [0,1] x [0, 1-x^2]
Para n == 6:     integral == 0.6666666666666667
Para n == 8:     integral == 0.6666666666666666
Para n == 10:    integral == 0.6666666666666667

2.2 Area do primeiro quadrante [0,1] x [0, sqrt(1-y)]
Para n == 6:     integral == 0.6670464379156135
Para n == 8:     integral == 0.6668355801001766
Para n == 10:    integral == 0.6667560429365088

3.1 Volume de e^(y/x) para [0.1, 0.5] x [x^3, x^2]
Para n == 6:     integral == 0.03330556611623718
Para n == 8:     integral == 0.033305566116232074
Para n == 10:    integral == 0.03330556611623208

3.2 Area de e^(y/x) para [0.1, 0.5] x [x^3, x^2]
Para n == 6:     integral == 0.10549788240049787
Para n == 8:     integral == 0.10549788240051994
Para n == 10:    integral == 0.10549788240051994

4.1 Volume da calota esferica de raio 1 e altura 1/4
Para n == 6:     integral == 0.1799870791119152
Para n == 8:     integral == 0.17998707911191525
Para n == 10:    integral == 0.17998707911191525

4.2 Volume de [0, e^(-y^2)] x [-1,1]
Para n == 6:     integral == 3.758249664214696
Para n == 8:     integral == 3.758249634237622
Para n == 10:    integral == 3.758249634231835
```

5 Conclusões

O presente exercício-programa implementado pelo grupo atingiu os objetivos propostos e foi uma ótima oportunidade para aprendizagem ativa dos conteúdos adquiridos em sala de aula.