EP2 - Fórmulas de Integração Numérica de Gauss

MAP3121 - Métodos Numéricos e Aplicações

Prof° Renato Vicente

Igor Costa D'Oliveira - 11391446

Paulo Gomes Ivonica - 11804532

05 de Junho de 2022

---

1. Introdução

Este relatório descreve as atividades ao estudar problemas de cálculos de integrais duplas usando o método de Gauss. Neste exercício computacional, analisou-se e implementou-se um algoritmo que calcula a integral dupla de funções no intervalo [-1;1] e que posteriormente a transpõe para outros intervalos.

2. Fórmulas de Gauss

Para a resolução de integrais em um intervalo [a ; b] é necessário que se conheça a sua primitiva, fato este que nem sempre é possível. Dado este contexto, ao longo da história vários matemáticos se dedicaram a obter métodos numéricos para obtenção de integrais.

O método contemplado neste trabalho é o de Gauss, que consiste no uso de uma função peso W que estipula a relevância de certos pontos da função. Pontos estes que são chamados de nós, ao calcular a função nestes nós e multiplicar o resultado pela função peso naquele ponto, temos uma parcela da integral. O resultado total da integral é dado pela somatória dessas parcelas acima descritas , tal que :

$$\int_a^b f(x)\,dx = \sum_{j=1}^{n} \omega_j f(x_j)$$

2.1 Mudança de Variável

A fim de minimizar o erro na aproximação, os nós e pesos são determinados em intervalos fixos e graças aos estudos do matemático francês Adrien-Marie Legendre é fácil determinar bons pesos e nós no intervalo [-1;1] , havendo extensas tabelas na internet para diferentes quantidades de valores calculados.

Porém, dada a infinidade de extremos de integração que uma função pode ter, esses valores calculados no intervalo [-1;1] deve ser transposto linearmente para outro intervalo [a , b], e seu peso deve ser multiplicado por um fator de escala.

Logo, se o intervalo for [0,4] os valores entre -1 e 1 devem ser distribuídos neste novo intervalo de forma linear, o ponto médio que era 0 passa a ser o 2 e o peso passa a ser o dobro do anterior já que o intervalo tem o dobro do tamanho do anterior.

2.2 Comentários sobre o código

Para facilitar o reuso de trechos do código, foram definidas várias funções neste, a fim de facilitar o cálculo e exibição dos resultados obtidos. Essas funções estão devidamente comentadas no código fonte. Nas primeiras 30 linhas foi feito o cabeçalho com nossos nomes e NUSP's, a importação da biblioteca Math além do armazenamento dos nós e pesos em vetores.

A primeira função definida foi a **funcao(x , y , ex),** e ela foi feita para facilitar a atribuição dos valores das funções que serão integradas nas tarefas. Note que os argumentos **X** e **Y** são as variáveis da função e o argumento **EX** é o que determina qual é o exercício requerido.

Salienta-se também que dada a existência de dois tópicos por tarefa, essas foram subdivididas usando, por exemplo, **EX = 1.0** para o cálculo do volume do Cubo e **EX = 1.1** para o cálculo do volume do Tetraedro, dois itens diferentes de uma mesma tarefa.

obs : Os valores atribuídos para cada função serão discutidos posteriormente em cada uma das tarefas.

Figura 1 - Trecho da função **funcao(x , y ,ex)** implementada no código.

```python
def funcao(x, y, ex):  # função para facilitar a atribuição da função a ser integrada
    # a variável EX recebida pelo usuário define qual o exercício ,logo :
    # a função no integrando e os seus extremos de integração também

    if ex == 1.0:
        f = 1  # ex = 1.0 define a F no cálculo do volume do cubo
    elif ex == 1.1:
        f = -x - y + 1  # ex = 1.1 define a F no cálculo do volume do tetraedro

    elif ex == 2.0:  # ex = 2.0 define a F no cálculo da Aréa integrando em Y depois em X
        f = 1
    elif ex == 2.1:  # ex = 2.1 define a F no cálculo da Aréa integrando em X depois em Y
        f = 1

    elif ex == 3.0:  # ex = 3.0 define a F para o cálculo da superfície da função no ex 3.1
        f = (((np.exp(2 * y / x)) * (x * x + y * y) + x ** 4) ** (1 / 2)) / (x ** 2)
    elif ex == 3.1:  # ex = 3.1 define a F para o cálculo do volume na região R
        f = 2.718282 ** (y / x)
    elif ex == 4.0:  # ex = 4.0 define a F que é a distância até o eixo de revolução Y
        f = math.sqrt(1 - y ** 2)
    elif ex == 4.1:  # ex = 4.1 define a F que é a distância até o eixo de revolução Y
        f = f = np.exp(-1 * (y * y))
    return f
```

As quatro próximas funções possuem a mesma funcionalidade, todas são usadas para facilitar a determinação dos extremos de integração dos exercícios requeridos.

obs : Os valores atribuídos para cada extremo serão discutidos posteriormente em cada uma das tarefas.

Figuras 2 e 3 - Trechos do código das funções **limite_a** e **limite_b.**

```python
def limite_a(ex):
    if ex == 1.0:
        f = 0
    elif ex == 1.1:
        f = 0

    elif ex == 2.0:
        f = 0
    elif ex == 2.1:
        f = 0

    elif ex == 3.0:
        f = 0.1
    elif ex == 3.1:
        f = 0.1

    elif ex == 4.0:
        f = 0.75
    elif ex == 4.1:
        f = -1
    return f
```

```python
def limite_b(ex):
    if ex == 1.0:
        f = 1
    elif ex == 1.1:
        f = 1

    elif ex == 2.0:
        f = 1
    elif ex == 2.1:
        f = 1

    elif ex == 3.0:
        f = 0.5
    elif ex == 3.1:
        f = 0.5

    elif ex == 4.0:
        f = 1
    elif ex == 4.1:
        f = 1

    return f
```

Note que em ambas as funções os intervalos possuem extremos numéricos. Recordando os conceitos de cálculo 3, observa-se que pelos Teoremas de Fubini é necessário que pelo menos uma das duas integrais tenha extremos numéricos para que o resultado final seja numérico.

Figuras 4 e 5 - Trechos do código das funções **limite_c** e **limite_d.**

```
def limite_c(x, ex):
    if ex == 1.0:
        f = 0
    elif ex == 1.1:
        f = 0

    elif ex == 2.0:
        f = 0
    elif ex == 2.1:
        f = 0

    elif ex == 3.0:
        f = x * x * x

    elif ex == 3.1:
        f = x * x * x

    elif ex == 4.0:
        f = 0
    elif ex == 4.1:
        f = 0
    return f
```

```
def limite_d(x, ex):
    if ex == 1.0:
        f = 1
    elif ex == 1.1:
        f = -x + 1

    elif ex == 2.0:
        f = 1 - x * x
    elif ex == 2.1:
        f = (1 - x) ** (1 / 2)

    elif ex == 3.0:
        f = x * x
    elif ex == 3.1:
        f = x * x

    elif ex == 4.0:
        f = (1 - x ** 2) ** 1 / 2
    elif ex == 4.1:
        f = np.exp(-1 * (x * x))
    return f
```

Note que em ambas as funções os intervalos possuem extremos variáveis segundo o argumento **X**. Recordando os conceitos de cálculo 3, a região de integração nestes casos possuem extremos definidos por funções e por valores numéricos.

A próxima função implementada foi a que itera o método duas vezes a **equacao_gauss_2_integrais(nos,peso,limit,ex)** perceba que os argumentos da função são os vetores contendo os nós e os pesos, a variável **limit** estipula quantas vezes o número de vezes que o loop for acontecerá. Já a variável **EX** como visto anteriormente determina qual o exercício em apreço, e consequentemente os seus extremos de integração, tal qual a sua função no integrando.

Vale ressaltar que a função analisada também tem em seu escopo a mudança de variável, como será visto logo a seguir.

Figura 6 - Primeira parte da implementação da equação de gauss.

```
def equacao_gauss_2_integrais(nos, peso, limit, ex):
    i = 0  # declara uma variável oomo contadora
    y = 0  # declara uma variável para armazenar o valor da funçaõ após a mudança de variável
    xi = [0] * 10  # vetor com os novos nós
    ui = [0] * 10  # vetor com os novos pesos

    while i != limit:
        # desloca os nós linearmnete para outro intervalo e os armazena em um vetor
        xi[i] = ((limite_a(ex) + limite_b(ex)) / 2) + ((limite_b(ex) - limite_a(ex)) * nos[i] / 2)
        # multiplica os pesos por um fator de escalaa e os armazena em um vetor
        ui[i] = peso[i] * (limite_b(ex) - limite_a(ex)) / 2
        # calcula o valor da função para a nova variável
        y = y + funcao(xi[i], 0, ex) * ui[i]
        i += 1  # incrementa o contador
```

Note que na estrutura do while é feita a mudança de intervalo nos nós **Xi[i]** e e também a mudança de do peso ao ser multiplicado por um fator de escala apropriado. É notável também que a variável **Y** recebe o valor da função com os novos nós e com os novos pesos.

Figura 7 - Segunda parte da implementação da equação de gauss.

```
I = 0  # declaração da variável que armazenará o valor final da integração
i = 0  # declaração das variáveis contadoras
j = 0
yij = [[0] * 10] * 10  # declaração de uma matriz para receber os n² nós
vij = [[0] * 10] * 10  # declaração de uma matriz para receber os n² pesos

while i != limit:
    j = 0  # zera o contador
    F = 0  # armazena o valor calulado para cada iteração em j
    while j != limit:
        # desloca os nós  linearmente para outro intervalo
        yij[i][j] = ((limite_c(xi[i], ex) + limite_d(xi[i], ex)) / 2) + (
                (limite_d(xi[i], ex) - limite_c(xi[i], ex)) * nos[j] / 2)
        # multiplica os pesos por um fator de escala
        vij[i][j] = peso[j] * (limite_d(xi[i], ex) - limite_c(xi[i], ex)) / 2
        # soma o valor calculado com os valores anteriormente calculados
        F = F + funcao(xi[i], yij[i][j], ex) * vij[i][j]
        j += 1  # incrementa o contador
    I = I + F * ui[i]  # adiciona o falor da função multiplicado pelo peso
                       # aos valores anteriormente calculados
    i += 1  # incrementa o contador
return I
```

Note que o método para duas integrais é muito semelhante ao primeiro while, apenas repetindo o procedimento dependendo do valor de **limit** ao quadrado.

Observa-se que toda a função foi pensada para depender de cada exercício, dado que não foi devidamente elucidado pelo enunciado se o usuário deveria ser livre para adicionar as funções e extremos que bem entendesse. Portanto, o grupo optou pela programação defensiva e organizou o código para a mínima interferência do usuário e priorizou a resolução dos exercícios propostos.

A última função é a **Main( )** que apenas possui como entrada uma variável **opcao** que recebe valores entre 1 e 5, sendo que os valores de 1 a 4 direcionam para a resolução das tarefas numericamente correspondentes. Já a opção 5 é responsável por finalizar a execução do programa.

Figura 8 - print da interface da **main** no terminal.

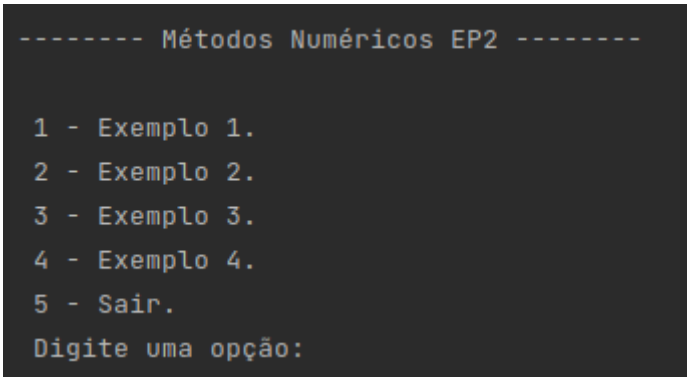

Figura 9 - primeira parte da função main, obtenção da variável **opcao**.

```
def main():  # a main basicamente recebe do usuário qual é o teste o qual ele quer verificar
    opcao = '1'
    while opcao != '5':  # printa para o usuário que tarefa que cada opção executa
        print("\n-------- Métodos Numéricos EP2 --------\n\n"
              " 1 - Exemplo 1.\n 2 - Exemplo 2."
              "\n 3 - Exemplo 3. \n 4 - Exemplo 4. \n 5 - Sair.")
        opcao = input(" Digite uma opção: ")
        while opcao != '1' and opcao != '2' and opcao != '3' and opcao != '4' and opcao != '5':
            opcao = input("\n Opção Inválida! Digite uma opção: ")  # recebe a variável opção com entrada
```

E na sequência para cada opção chama-se o método para uma tarefa específica, e os valores obtidos são exibidos no terminal.

Figura 10 - Chamou-se o método para a primeira tarefa e seus resultados são exibidos na tela.

```
if opcao == '1':  # o método é usado para calular o volume do cubo e do tetraedro
    v6 = equacao_gauss_2_integrais(nos_n6, peso_n6, 6, 1.0)
    v8 = equacao_gauss_2_integrais(nos_n8, peso_n8, 8, 1.0)
    v10 = equacao_gauss_2_integrais(nos_n10, peso_n10, 10, 1.0)
    t6 = equacao_gauss_2_integrais(nos_n6, peso_n6, 6, 1.1)
    t8 = equacao_gauss_2_integrais(nos_n8, peso_n8, 8, 1.1)
    t10 = equacao_gauss_2_integrais(nos_n10, peso_n10, 10, 1.1)

    # exibição dos resultados
    print("\n-------- Exemplo 1 --------")
    print("\n 1. Volume do Cubo: \nPara n = 6 o volume calculado V = ", v6,
          "m, V exato = 1m.\nPara n = 8 o volume calculado V = ", v8, "m, V exato = 1m."
                                                                    "\nPara n = 10 o volume calculado V = ",
          v10, "m, V exato = 1m."
               "\n 2. Volume do tetraedro: \nPara n = 6 o volume calculado V = ", t6, "m, V exato = 1/6m."
                                                                                    "\nPara n = 8 o volume calculado V = ",
          t8, "m, V exato = 1/6m.\nPara n = 10 o volume calculado V = ", t10, "m, V exato = 1/6m.\n")
```

3. Resolução dos Exemplos

3.1. Exemplo 1

O exemplo 1 é dividido em dois itens. No item (a) é necessário calcular o volume de um cubo cujas arestas têm comprimento 1. Para isso foi feita a integral dupla dos limites de integração do cubo, que são entre 0 e 1 para ‘a’ e ‘b’, 0 e 1 para ‘c’ e ‘d’ e a função que será integrada é o valor 1. A integral realizada é a seguinte: $\int_0^1 \int_0^1 1 \, dx \, dy$. Na imagem 11 ilustra os valores encontrados com o código em python, que estão de acordo com o esperado teoricamente.

Figura 11 - Volume do Cubo calculado.

```
 1. Volume do Cubo:
Para n = 6 o volume calculado V =  1.0 m, V exato = 1m.
Para n = 8 o volume calculado V =  1.0 m, V exato = 1m.
Para n = 10 o volume calculado V =  1.0 m, V exato = 1m.
```

Já o item (b) precisa calcular o volume de um tetraedro com vértices (0,0,0), (1,0,0), (0,1,0) e (0,0,1). Para isso é necessário encontrar os limites de integração, o grupo decidiu fazer a integral de fora com os limites do eixo x, logo a e b são 0 e 1 respectivamente, e os limites de dentro são em função de y, logo c e d são 0 e -x + 1. Já a função é o

plano composto pelos três eixos e será isolado z, logo f(x,y) = -x -y +1. Portanto, a integral dupla para se encontrar o volume é a seguinte: $\int_0^{1}\int_0^{-x+1}(-x-y+1)\,dy\,dx$. Na figura 12 ilustra os volumes calculados pelo software, e podemos observar que os valores foram de acordo com o volume exato do sólido que é igual ao volume de um tetraedro regular dividido por 4, logo $V = \frac{a^3\sqrt{2}}{24} = \frac{4}{24} = \frac{1}{6} = 0,166666667m.$

Figura 12 - Volume do Tetraedro calculado.

```
 2. Volume do tetraedro:
Para n = 6 o volume calculado V =  0.16666666666666666 m, V exato = 1/6m.
Para n = 8 o volume calculado V =  0.16666666666666669 m, V exato = 1/6m.
Para n = 10 o volume calculado V =  0.16666666666666669 m, V exato = 1/6m.
```

3.2. Exemplo 2

O exemplo 2 pede para calcular a área A da região no primeiro quadrante limitada pelos eixos e pela curva $y = 1 - x^2$, que pode ser obtida pela integral dupla: $A = \int_0^{1}\int_0^{1-x^2} 1\,dy\,dx = \int_0^{1}\int_0^{\sqrt{1-y}} 1\,dx\,dy$. O valor exato dessa área A é igual a $\frac{2}{3}m^2$, logo o resultado encontrado pelo código deve ser próximo desse valor.

A figura 13 ilustra os resultados encontrados tanto para a primeira integral, quanto para a segunda. É possível observar que a primeira integral, que fez a integração de dy e depois dx, obteve valores muito mais precisos do que a segunda integral. Isto pode ser explicado analisando os limites de integração das duas integrais, onde na primeira possui um polinômio de segundo grau e na segunda uma função dentro de uma raiz quadrada, isto faz com que o seu resultado final perca precisão. Entretanto, ambas as integrais chegaram em valores próximos ao valor exato de $\frac{2}{3}m^2$.

Figura 13 - Área A da região.

```
 1. Integral dydx:
Para n = 6 a integral calculada I =  0.6666666666666667
Para n = 8 a integral calculada I =  0.6666666666666666
Para n = 10 a integral calculada I =  0.6666666666666667
 2. Integral do dxdy:
Para n = 6 a integral calculada I =  0.6670464379156135
Para n = 8 a integral calculada I =  0.6668355801001766
Para n = 10 a integral calculada I =  0.6667560429365088
```

3.3. Exemplo 3

Já no exemplo 3 também possui dois itens. O item (a) pede para calcular a área de uma superfície descrita por $z = e^{y/x}$, $0.1 \leq x \leq 0.5$, $x^3 \leq y \leq x^2$. Para isso, foi encontrado os limites de integração que são 0,1 e 0,5 para a e b, e x^3 e x^2 para c e d, a sua função é o próprio $e^{y/x}$. Logo, a integral dupla que será utilizada para se calcular a área é a seguinte: $\int_{0.1}^{0.5} \int_{x^3}^{x^2} e^{y/x}\, dy\, dx$.

Realizando essa integral em alguns softwares disponíveis na internet, entre eles o Symbolab e o Wolfram, o resultado encontrado para ambos foi de aproximadamente $0.0333\ m^2$. Portanto, o valor que o código desenvolvido pelo grupo deve chegar deve ser próximo a esse número. A figura 14 ilustra a área encontrada pelo nosso código e o valor foi exatamente ao esperado.

Figura 14 - Área A da região primeira região.

```
Para n = 6  a área calculada foi A =  0.03330556666976598 m^2.
Para n = 8 a área calculada foi A =  0.03330556666976088 m^2.
Para n = 10 a área calculada foi A =  0.03330556666976088 m^2.
```

Já o item (b) pede para calcular a área de uma superfície descrita por $z = f(x, y)$, $(x, y) \in R$, que é igual a seguinte integral: $\iint_R \sqrt{f_x(x,y)^2 + f_y(x,y)^2 + 1}\, dy\, dx$. As funções descritas dentro da raiz

quadrada são as derivadas parciais da função do item (a), logo substituindo pelos valores finais a integral dupla é igual a: $\int_{0.1}^{0.5}\int_{x^3}^{x^2} \frac{\sqrt{e^{2y/x}.(x^2+y^2).x^4}}{x^2}\, dy\, dx$. Na figura 15 está ilustrada o valor da área encontrada usando o código em python resolvendo a integral dupla do último parágrafo, o seu valor aproximado foi de $0.0155\ m^2$. Para saber se esta é a área certa o grupo resolveu a integral usando outros dois softwares já validados (Symbolab e Wolfram) e os seus valores foram de $0.0155\ m^2$, logo a função calculou as áreas de acordo com o esperado.

Figura 15 - Área S da segunda região.

```
Para n = 6  a área da superficie calculada foi S =  0.10549788240049787 m^2.
Para n = 8  a área da superficie calculada foi S =  0.10549788240051995 m^2.
Para n = 10 a área da superficie calculada foi S =  0.10549788240051994 m^2.
```

3.4. Exemplo 4

Por último, o exemplo 4 possui dois itens. No primeiro item é necessário calcular o volume da calota esférica de altura ¼ da esfera de raio 1. Para isso será usada a fórmula do volume $V = 2\pi \int\int_R d_\gamma(x, y)\, dx\, dy$. A função final fica igual a $\int_{3/4}^{1}\int_{0}^{\sqrt{(1-x^2)}} \sqrt{1-y^2}\, dy\, dx$. Na figura 16 está ilustrado os valores encontrados para o volume da calota, que foram próximos de $0,1792\ m^3$, ao calcular esta integral usando os softwares Symbolab e Wolfram foi encontrado o mesmo valor de volume, logo o resultado está certo.

Figura 16 - Volume da calota.

```
Para n = 6  o volume calculado da calota foi  =  0.17923671892594825
Para n = 8  o volume calculado da calota foi   =  0.17923671892594836
Para n = 10 o volume calculado da calota foi  =  0.17923671892594834
```

Já para o segundo item é necessário calcular o volume do sólido de revolução obtido da rotação da região, em torno do eixo y, delimitada

por $x = 0$, $x = e^{-y^2}$, $y = -1$, $y = 1$. Logo a integral a ser calculada é a seguinte: $\int_{-1}^{1} \int_{0}^{e^{-x^2}} e^{-y^2}\, dy\, dx$. A figura 17 ilustra o volume do sólido encontrado pelo código em python, o volume foi de $3,8421\ m^3$. Utilizando outros softwares já validados para o cálculo desta integral foi visto que o valor obtido está certo.

Figura 17 - Volume do sólido.

```
Para n = 6  o volume calculado do sólido foi  =  3.8420849706625697
Para n = 8  o volume calculado do sólido foi   =  3.84214029877275
Para n = 10 o volume calculado do sólido foi  =  3.842140850074193
```

4. Conclusão

Neste exercício-programa foi visto como implementar as fórmulas de integração numérica conhecidas como Fórmulas de Integração de Gauss e usá-las para o cálculo de algumas integrais simples e duplas. O grupo implementou o algoritmo usando a linguagem de programação Python visto que o EP1 já foi feito com esta linguagem e pela versatilidade da linguagem.

A função fundamental do código é a “equacao_gauss_2_integrais” e ela pede como parâmetro os nós, os pesos, o valor de “n”, a função para se integrar e os 4 limites de integração. Logo, com esta função é possível escolher os limites de integração e a sua função, porém o grupo decidiu fazer uma interface no qual o usuário pudesse apenas escolher um dos exemplos mostrados no tópico 3 deste trabalho. Entretanto, como a implementação das fórmulas de gauss serão usadas na tarefa 3 desta disciplina, o grupo fez um código que pudesse ser facilmente implementado como função para outro código.

É possível concluir que a implementação das fórmulas de integração de gauss em um código em python foi um sucesso, isto porque todos os valores encontrados nos exemplos feitos foram de acordo com os resultados teóricos esperados, logo foi possível validar o código desenvolvido neste trabalho.

5. Referências

[1] Enunciado Exercício-Proposto 1. Visto em: https://edisciplinas.usp.br/.
Acesso em 1 de Junho de 2022.

[2] Métodos Numéricos. Visto em: https://dlmf.nist.gov/
Acesso em 1 de Junho de 2022.

[3] Software para resolução de integrais. Visto em:
https://www.wolframalpha.com/
Acesso em 1 de Junho de 2022.

[4] Software para resolução de integrais. Visto em: https://pt.symbolab.com/
Acesso em 1 de Junho de 2022.