Universidade de São Paulo

Instituto de Matemática e Estatística: Dep. Mat. Aplicada

Escola Politécnica

Luciano Chaparin Luisi - 9016866
luciano.luisi@usp.br
Bruno Prasinos Bernal - 10355141
brunobernal@usp.br

Fórmulas de Integração Numérica de Gauss: EP2

Trabalho apresentado como avaliação da disciplina
MAP 3121 - Métodos Numéricos e Aplicações

São Paulo

2022

RESUMO

Neste trabalho foi estudada a implementação de fórmulas de integração numérica conhecidas como fórmulas de integração de Gauss e suas aplicações visando o cálculo de integrais duplas.

1 INTRODUÇÃO

1.1 CONCEITOS

Uma fórmula geral de integração numérica para se aproximar a integral de uma função f de a até b, é dada por:

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

onde os nós xj ∈ [a, b] e os pesos ωj são determinados impondo-se condições para o erro E(f). No caso das fórmulas de Gauss, os pesos e os nós são escolhidos de forma que a aproximação seja exata (E(f) = 0) para polinômios de maior grau possível.

1.2 OBJETIVOS

O objetivo do exercício programa consiste em implementar o cálculo de integrais duplas em regiões R do plano, descritas por fórmulas iteradas. Usaremos as fórmulas de Gauss com n nós para as integrações numéricas. Os nós e os pesos foram fornecidos para o intervalo [−1, 1] e o programa deverá fazer os ajustes necessários para outros intervalos. Usaremos precisão dupla, e testaremos o programa nos exemplos fornecidos, usando fórmulas de Gauss com n = 6, 8 e 10.

2 IMPLEMENTAÇÃO E TESTES

2.1 ARQUIVOS DO PROJETO

2.1.1 requirements

Diretório contendo arquivos listando os módulos e versões instalados no projeto, testado e executado em ambiente Conda.

2.1.2 testes_exemplos (referente ao EP1)

Diretório que possui arquivos .CSV utilizados para os testes apresentados neste relatório (outros testes foram realizados com diferentes matrizes e dimensões, mas não anexados ao projeto).

2.1.3 LEIAME.txt

Documentação e instruções pertinentes ao uso do programa.

2.1.4 custom_functions.py

Módulo de funções usadas para recebimento, cálculo e devolução dos resultados, pertinentes ao exercício programa.

2.1.5 ep2.py

Executável que realiza o recebimento e retorno de dados do usuário num terminal através de linha de comando.

2.1.6 main.py (referente ao EP1)

Executável que realiza o recebimento e retorno de dados do usuário num terminal através de linha de comando.

2.2 ALGORITMO

O programa foi escrito e testado em Python 3.7, utilizando a biblioteca NumPy para a realização dos cálculos.

2.2.1 gaussIntegrate(f,a,b,n)

Essa função do módulo de funções customizadas calcula uma integral simples de uma função num intervalo definido utilizando fórmulas de Gauss, recebendo como argumentos uma função *f*, os limites de integração *a* e *b*, e um número *n* de nós para o método de Gauss.

```python
def gaussIntegrate(f, a, b, n: int) -> float:
    '''Integra uma função `f(x)` no intervalo `x = [a,b]` usando fórmulas de Gauss com `n` nós ···

    if n == 6: ···
    elif n == 8: ···
    elif n == 10: ···

    # Guarda novos vetores com valores negativos das abscissas
    x = np.append(np.flip(x),-x)
    w = np.append(np.flip(w),w)

    # Transportando linearmente para os limites desejados, [-1,1] para [a,b]:
    # # x = (b-a)*t/2 + (b+a)/2
    # # dx = (b-a)/2 dt
    # # Mudança de variável de x para t
    # # int[a,b](f(x))dx = int[-1,1](f(((b-a)/2)*t+((b+a)/2))*((b-a)/2))dt = (b-a)/2)*int[-1,1](f(((b-a)/2))*t+((b+a)/2)))dt

    A = np.array(list(map(a,x))) if callable(a) else a
    B = np.array(list(map(b,x))) if callable(b) else b
    x = ((B-A)/2)*x + ((B+A)/2)
    f = np.array(list(map(f, x)))   # Aplica a função f sobre cada elemento do vetor m*t+c
    soma = np.sum(np.multiply(w,f)) # Somatória dos w[i]*f(x[i]) termos, i de 0 até n
    return ((B-A)/2)*soma
```

2.2.2 gaussDoubleIntegrate(f,a,b,c,d,n)

Esta função permite calcular numericamente uma integral dupla de uma função em intervalos retangulares, recebendo como argumentos uma função *f*(x,y), os limites *a* e *b* de x e os limites *c* e *d* de y, e o número *n* de nós para a fórmula de Gauss, aproveitando a função de integração simples de maneira recursiva.

```python
def gaussDoubleIntegrate(f, a, b, c, d, n: int) -> float:
    '''Calcula a integral dupla de uma função `f(x,y)` no intervalo

    def g(x): return gaussIntegrate(lambda y: f(x, y), c, d, n)
    return gaussIntegrate(g, a, b, n)
```

2.2.3 gaussDoubleIntegrateVar(f,a,b,c,d,n)

Esta função calcula numericamente uma integral dupla de uma função em intervalos variáveis, recebendo como argumentos uma função *f*(x,y), os limites *a* e *b* de x e os limites *c(x)* e *d(x)* de y, e o número *n* de nós para a fórmula de Gauss, aproveitando a função de integração simples de maneira recursiva. A função, em seu estado final, não foi capaz de computar corretamente os valores esperados para os casos exemplificados.

```python
def gaussDoubleIntegrateVar(f, a, b, c, d, n: int,) -> float:
    '''Calcula a integral dupla de uma função `f(x,y)` no intervalo

    if n == 6:⋯
    elif n == 8:⋯
    elif n == 10:⋯

    # Guarda novos vetores com valores negativos das abscissas
    x = np.append(np.flip(x),-x)
    w = np.append(np.flip(w),w)

    soma = 0
    for i in range(len(x)):
        C = c(x[i]) if callable(c) else c
        D = d(x[i]) if callable(d) else d
        F = gaussIntegrate(lambda y: f(x[i],y),C,D,n)
        soma += w[i]*F
    return ((b-a)/2)*soma
```

2.3 TESTES

A seguir são apresentados algumas capturas de tela do resultado de cada tipo de ação do programa, que apresenta ao usuário seus dados de entrada e resultados formatados. Os testes aqui apresentados não contemplam todos os testes, sendo apenas um demonstrativo simples do modelo de saída do programa.

2.3.1 Exemplo 1

```
Escolha a ação desejada (1 a 5): 1
Opção escolhida: Exemplo 1 - Volume de poliedros
Volume de um cubo com arestas de comprimento 1
(n=6 nós): Vol.cubo = 1.000000
Volume de um cubo com arestas de comprimento 1
(n=8 nós): Vol.cubo = 1.000000
Volume de um cubo com arestas de comprimento 1
(n=10 nós): Vol.cubo = 1.000000
Volume de um tetraedro com vértices (0,0,0), (1,0,0), (0,1,0) e (0,0,1)
(n=6 nós): Vol.tetraedro = 0.666667
Volume de um tetraedro com vértices (0,0,0), (1,0,0), (0,1,0) e (0,0,1)
(n=8 nós): Vol.tetraedro = 0.666667
Volume de um tetraedro com vértices (0,0,0), (1,0,0), (0,1,0) e (0,0,1)
(n=10 nós): Vol.tetraedro = 0.666667
MENU PRINCIPAL -----------------------------------------------------
```

2.3.2 Exemplo 2

```
Escolha a ação desejada (1 a 5): 2
Opção escolhida: Exemplo 2 - Área limitada por funções
Área A do primeiro quadrante entre os eixos e a curva y = 1-x²
(n=6 nós): Área A (Idydx)= 0.666667
Área A do primeiro quadrante entre os eixos e a curva y = 1-x²
(n=8 nós): Área A (Idydx)= 0.666667
Área A do primeiro quadrante entre os eixos e a curva y = 1-x²
(n=10 nós): Área A (Idydx)= 0.666667
Área A do primeiro quadrante entre os eixos e a curva x = (1-y)**0.5
(n=6 nós): Área A (Idxdy)= 0.943346
Área A do primeiro quadrante entre os eixos e a curva x = (1-y)**0.5
(n=8 nós): Área A (Idxdy)= 0.943048
Área A do primeiro quadrante entre os eixos e a curva x = (1-y)**0.5
(n=10 nós): Área A (Idxdy)= 0.942935
MENU PRINCIPAL -----------------------------------------------------
```

2.3.3 Exemplo 3

```
Escolha a ação desejada (1 a 5): 3
Opção escolhida: Exemplo 3 - Área de uma superfície e volume sob esta
Área da superfície descrita por z = exp(y/x)
(n=6 nós): Área A= 0.251990
Área da superfície descrita por z = exp(y/x)
(n=8 nós): Área A= 0.251990
Área da superfície descrita por z = exp(y/x)
(n=10 nós): Área A= 0.251990
```

2.3.4 Exemplo 4

```
Escolha a ação desejada (1 a 5): 4
Opção escolhida: Exemplo 4 - Volume de sólido de rotação
Volume da calota esférica
(n=6 nós): V= 6.134135
Volume da calota esférica
(n=8 nós): V= 6.124626
Volume da calota esférica
(n=10 nós): V= 6.121065
```

4 RESULTADOS

Sucessos e limitações observadas

O programa apresentou resultados consistentes e adequados para integrações em uma variável e para integrais duplas em duas variáveis quando os limites de integração são definidos por constantes. O programa, porém, se frustrou ao calcular integrais duplas quando os limites de integração se deram por funções. Não conseguimos determinar ao certo em qual(quais) pontos o algoritmo apresentou erros, e após inúmeras tentativas e correções o código ainda deixou a desejar. A maior dificuldade foi em definir um intervalo y = [c(x),d(x)] para cada x[i] tal que y fosse uma função de x (y = f(x[i])) e calcular a integral a partir disso, mesmo nas tentativas computacionais realizadas manualmente.