

# Algebra Linear Computacional COC473 - Lista 5

Bruno Dantas de Paiva  
DRE: 118048097

October 21, 2020

## 1 Questão 1

```
def numeric_integration(function, a, b, number_of_points, polynomial_integration = True):
    incognitas = []
    result = 0
    if(polynomial_integration):
        delta_x = float(abs(b-a)) / (number_of_points - 1)
        vector_b = []
        for i in range(1, number_of_points+1):
            incognitas.append(a+(i-1)*delta_x)
            vector_b.append(float((b**i-a**i))/i)

        vandermond_matrix = [[0.0 for i in range(number_of_points)] for j in range(number_of_points)]

        for i in range(number_of_points):
            for j in range(number_of_points):
                vandermond_matrix[i][j] = incognitas[j]**i

        matrix_a = ALC_List1.lu_decomposition(vandermond_matrix)
        x = Matrix_Utls.backward_substitution(matrix_a, Matrix_Utls.forward_substitution)

        for i in range(number_of_points):
            result += x[i]*function(incognitas[i])
    else:
        weight = Weight.get(number_of_points).get("weight")
        points = Weight.get(number_of_points).get("points")
        l = b-a

        for i in range(number_of_points):
            incognitas.append((a+b+points[i]*l)/2)

        integral_sum = 0
        for i in range(number_of_points):
            integral_sum += function(incognitas[i])*weight[i]

        result = integral_sum*l/2
    print("Integral:␣" + str(result))
```

Note que esta função possui rotinas secundárias, onde estas são as mesmas utilizadas na lista 1.

Além disso, o parametro "Weight" é na verdade um dicionário de dicionários de pesos, que possuem a seguinte estrutura:

```
{n:{ "points": [ponto_1 ... ponto_n] ,"weight": [ peso_1 ... peso_n] }
```

Onde n varia de 2 até 10.

Ainda sobre esta função, ela engloba a integração polinomial e a quadratura de gauss, sendo escolhida pelo usuário pelo booleano polynomial integration.

## 2 Questão 2

### 2.1 Integral 1

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Quadratura de Gauss:
Integral: 0.341344746068543
Integração Polinomial:
Integral: 0.34134474607651527
```

Figure 1: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

### 2.2 Integral 2

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Quadratura de Gauss:
Integral: 0.4999996589432936
Integração Polinomial:
Integral: 0.49933498808470905
```

Figure 2: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

## 3 Questão 3

Após executar a rotina desenvolvida para ambas as integrais, foi observado que estes resultados precisam mais de 10 pontos de integração para satisfazer a solução. Tendo em vista que o valor varia muito ao alterar o número de pontos utilizados, o que leva a entender que não houve a convergência do resultado. Como pode ser observado abaixo:

### 3.1 Integral 1

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 13.227748281885857
Quadratura de Gauss
Integral: 17.05580203189161
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 52.85924970185418
Quadratura de Gauss
Integral: 6.598559615255326
```

Figure 3: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

### 3.2 Integral 2

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 20.120283031996205
Quadratura de Gauss
Integral: 12.564682282723105
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 65.2099810489997
Quadratura de Gauss
Integral: 5.35698500920671
```

Figure 4: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

## 4 Questão 4

Tal como a questão anterior, após executar a rotina desenvolvida para ambas as integrais, foi observado que estes resultados precisam mais de 10 pontos de integração para satisfazer a solução. Tendo em vista que o valor varia muito ao alterar o número de pontos utilizados, o que leva a entender que não houve a convergência do resultado. Como pode ser observado abaixo:

### 4.1 Integral 1

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 2.644426976286834
Quadratura de Gauss
Integral: 6.591566942728261
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 16.3377076997587
Quadratura de Gauss
Integral: 0.7532902132971433
```

Figure 5: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

### 4.2 Integral 2

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 4.1315531593626815
Quadratura de Gauss
Integral: 4.464310594729089
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 20.17187288309652
Quadratura de Gauss
Integral: 0.48426836843985044
```

Figure 6: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

## 5 Questão 5

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integração polinomial
Integral: 376.0
Quadratura de Gauss
Integral: 194.66666666666666
```

Figure 7: Imagem contendo o resultado do cálculo da integral pelos 2 métodos requisitados.

## 6 Questão 6

```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integral Mid Point: 0.9230769230769231
Integral Trapeze: 1.6500000000000001
Integral Simpson: 1.1653846153846155
```

Figure 8: Imagem contendo o resultado do cálculo da integral pelos 3 métodos requisitados.

## 7 Questão 7

### 7.1 Implementação

Para a implementação deste método foi necessário obter os pesos do método de Gauss-Hermite, onde estes estão sendo armazenados da mesma forma que o método da questão 1.

Além disso, é importante ressaltar que somente com Gauss-Hermite, não seria possível calcular os casos de  $n$  até infinito (onde  $n$  pertence ao conjunto dos números naturais), por isso, também foi necessário fazer a implementação de Gauss-Laguerre, que permite o cálculo de integrais indefinidas de 0 até infinito.

```
def integrate(function, a, b, number_of_points, polynomial_integration = True):
    def laguerre_function(x):
        return function(x)/(math.exp(-x))
    def hermite_function(x):
        return function(x)/(math.exp(-x**2))

    weight_hermite = Weight_Hermite.get(number_of_points).get("weight")
    points_hermite = Weight_Hermite.get(number_of_points).get("points")
    weight_laguerre = Weight_Laguerre.get(number_of_points).get("weight")
    points_laguerre = Weight_Laguerre.get(number_of_points).get("points")
    hermite_sum = 0
    laguerre_sum = 0
    result = 0
    partial_integral = 0

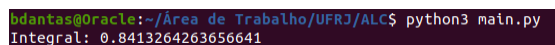
    for i in range(number_of_points):
        hermite_sum += hermite_function(points_hermite[i])*weight_hermite[i]
    for i in range(number_of_points):
        laguerre_sum += laguerre_function(points_laguerre[i])*weight_laguerre[i]

    if(a == "-inf"):
        if(b == "inf"):
            print("Integral:_" + str(hermite_sum))
            return
        elif(b >=0):
            result = hermite_sum - laguerre_sum + numeric_integration(function, 0, b, number_of_points, polynomial_integration)
        else:
            result = hermite_sum - numeric_integration(function, b, 0, number_of_points, polynomial_integration)
    elif(b == "inf"):
        if(a == 0):
            print("Integral:_" + str(laguerre_sum))
            return
        elif(a > 0):
            result = laguerre_sum - numeric_integration(function, 0, a, number_of_points, polynomial_integration)
        else:
            result = numeric_integration(function, a, 0, number_of_points, polynomial_integration) + laguerre_sum
    else:
        result = numeric_integration(function, a, b, number_of_points, polynomial_integration)
    print("Integral:_" + str(result))
```

Note que a função, para utilizar os métodos de Gauss-Laguerre e Gauss-Hermite, tiveram que passar por modificações para ser possível utilizar estas aproximações.

É importante frisar também que uma das rotinas utilizadas é a que está mostrada na questão 1 desta mesma lista.

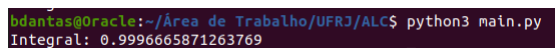
## 7.2 Integral 1



```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integral: 0.8413264263656641
```

Figure 9: Imagem contendo o resultado do cálculo da integral pelo método requisitado

## 7.3 Integral 2



```
bdantas@Oracle:~/Área de Trabalho/UFRJ/ALC$ python3 main.py
Integral: 0.9996665871263769
```

Figure 10: Imagem contendo o resultado do cálculo da integral pelo método requisitado