ANÁLISE NUMÉRICA I - Derivação e Integração Numérica

UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Curso:
Licenciatura em Engenharia Informática

Unidade Curricular:
Análise Numérica I

Docente:
Hermenegildo Borges de Oliveira

Realizado por:
Daniel Maryna (64611)
Miguel Silva (80072)
Francisco Nunes (80061)
Brandon Mejia (79261)

12/2024

CONTEÚDO

INTRODUÇÃO

A integração numérica desempenha um papel fundamental na análise numérica, permitindo a aproximação de integrais definidos em casos onde a integração analítica é inviável. Métodos como as fórmulas de Newton-Cotes são amplamente utilizados para calcular integrais em situações onde os valores da função são conhecidos apenas em pontos específicos ou quando a função integrada não possui uma primitiva elementar.

Este trabalho tem como objetivo implementar, em Python, as fórmulas de Newton-Cotes, tanto abertas quanto fechadas, para diferentes valores de $\boldsymbol{n}$. Além disso, serão realizadas aproximações de integrais de funções dadas em forma de tabelas de pontos ou através de expressões matemáticas. O código permitirá ao utilizador selecionar o método desejado e o grau de precisão $(\boldsymbol{n})$ a ser utilizado.

Ao longo deste relatório, serão apresentadas as fundamentações teóricas que embasam os métodos utilizados, a implementação computacional desenvolvida, os resultados obtidos e as conclusões extraídas a partir da análise das aproximações calculadas.

FUNDAMENTAÇÃO TEÓRICA

A integração numérica é uma técnica essencial para determinar o valor aproximado de integrais definidas, especialmente em casos onde a primitiva da função integrada não pode ser obtida analiticamente. Entre os diversos métodos existentes, destacam-se as **fórmulas de Newton-Cotes**, que se baseiam na substituição da função integrada por um polinómio interpolador.

Fórmulas de Newton-Cotes

As fórmulas de Newton-Cotes podem ser divididas em duas categorias principais: **fechadas** e **abertas**.

Fórmulas Fechadas: Consideram os pontos extremos do intervalo de integração $[a, b]$. Exemplos clássicos incluem:

- **Regra do Trapézio** $(\boldsymbol{n = 1})$: A função é aproximada por uma linha reta entre os pontos a e b.
- **Regra de Simpson** $(\boldsymbol{n = 2})$: Utiliza um polinómio de grau 2 para interpolar os pontos.
- **Regra de Simpson 3/8** $(\boldsymbol{n = 3})$: Baseia-se em um polinómio de grau 3.
- **Regra de Boole** $(\boldsymbol{n = 4})$: Utiliza um polinómio de grau 4 para maior precisão.

Fórmulas Abertas: Excluem os pontos extremos do intervalo, sendo úteis quando a função não está definida nesses pontos.

Exemplos incluem:

Regra do Ponto Médio $(\boldsymbol{n = 0})$.

Fórmulas para $\boldsymbol{n = 1, 2}$ e $\boldsymbol{3}$ que seguem princípios semelhantes às fórmulas fechadas, mas sem incluir os extremos.

Estrutura Matemática

A ideia geral das fórmulas de Newton-Cotes é aproximar a integral $\int_a^b f(x)dx$ por uma soma ponderada dos valores da função nos pontos de interpolação. Matematicamente, temos:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i)$$

onde os coeficientes a_i dependem do método e do número de pontos $n+1$ utilizados.

Para as fórmulas fechadas, os pontos de interpolação são igualmente espaçados no intervalo $[a, b]$, com $h = \frac{b-a}{n}$. Já nas fórmulas abertas, os pontos estão dentro do intervalo, sendo $x_0 = a + h$ e $x_n = b - h$, com $h = \frac{b-a}{n+2}$.

Erro de Aproximação

O erro em cada fórmula depende do grau do polinómio interpolador e do comportamento da derivada da função integrada. Em geral, para n pontos:

- O erro é proporcional a h^{n+2} para fórmulas fechadas.
- Para fórmulas abertas, o erro depende de h^{n+1}.

Aplicação e Implementação

Os métodos de Newton-Cotes são úteis em diversas áreas de aplicação, como engenharia e física, especialmente quando se lida com dados experimentais. A implementação computacional destes métodos permite avaliar integrais de maneira eficiente, ajustando a precisão conforme necessário pelo utilizador.

IMPLEMENTAÇÃO DOS PROGRAMAS

Nesta secção, detalha-se a implementação das fórmulas de Newton-Cotes fechadas e abertas, destacando as principais funções de cálculo e validação.

Fórmulas de Newton-Cotes Fechadas

Função regra_do_Trapezio:

```
def regra_do_Trapezio(funcao_ou_Pontos,  intervalo:tuple):
    x0, x1 = intervalo
    h = (x1 - x0)
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        valor_final = h/2*((funcao.subs("x",x0))+ funcao.subs("x",x1))
        p = (funcao.subs("x", x0))
        p = funcao.subs("x", x1)
        return valor_final.n()
    else:
        if len(funcao_ou_Pontos) < 2: raise ValueError("Precisa de minimo 2 
(Pontos)")
        ponto0,ponto1 = funcao_ou_Pontos[0] , funcao_ou_Pontos[1]
        valor_final = h/2*(ponto0.y + ponto1.y)
```

Código 1 - Regra do Trapézio

Descrição:
Esta função implementa a **Regra do Trapézio** ($n = 1$), utilizando dois pontos.
O intervalo de integração é definido pelos extremos x_0 e x_1.

Entrada:
- funcao_ou_Pontos: Pode ser uma expressão matemática ou uma lista de pontos interpolados.
- intervalo: Representa os limites $[x_0, x_1]$.

Lógica:
Calcula $h = x_1 - x_0$.
Caso seja fornecida uma função:
- Calcula os valores $f(x_0)$ e $f(x_1)$ substituindo os pontos na função.
- Aplica a fórmula $\frac{h}{2}[f(x_0) + f(x_1)]$.

Caso sejam fornecidos pontos:
- Utiliza diretamente as ordenadas (y) dos pontos para calcular o valor da integral.

Função regra_de_Simpson_1:

```
def regra_de_Simpson_1(funcao_ou_Pontos , intervalo:tuple):
    x0, x2 = intervalo
    x1 = (x0 + x2) / 2
    h = x1 - x0 and x2 - x1 # passo de ponto a ponto uniformemente
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        valor_final = h/3 *(funcao.subs("x",x0) + 4*funcao.subs("x",x1) +
funcao.subs("x",x2))
        return valor_final.n()
    else: # Funciona com pontos
        if len(funcao_ou_Pontos) < 3: raise ValueError("Precisa de minimo 3
(Pontos)")
        ponto0, ponto1, ponto2 = funcao_ou_Pontos[0], funcao_ou_Pontos[1],
funcao_ou_Pontos[2]
        h = (ponto2.x - ponto0.x)/2
        valor_final = h/3*(ponto0.y + 4*ponto1.y + ponto2.y)
        return valor_final.n()
```

Código 2 - Regra de Simpson

Descrição:

Esta função implementa a **Regra de Simpson** $(\boldsymbol{n = 2})$, interpolando três pontos.

Entrada:

- funcao_ou_Pontos: Função ou pontos interpolados.
- intervalo: Limites $[x_0, x_2]$.

Lógica:

Calcula $x_1 = \frac{x_0+x_2}{2}$ e $h = x_1 - x_0$.

Se for uma função:

- Avalia $f(x_0), f(x_1)$ e $f(x_2)$ na função dada.
- Aplica a fórmula $\frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$.

Com pontos:

- Usa diretamente as ordenadas (y) para calcular o resultado.

Função regra_de_Simpson_2:

```
def regra_de_Simpson_2(funcao_ou_Pontos , intervalo:tuple):
    x0, x3 = intervalo
    h = (x3 - x0) / 3
    x1 = x0 + h
    x2 = x0 + 2*h
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        valor_final = 3*h/8*(funcao.subs("x",x0) + 3*funcao.subs("x",x1) +
3*funcao.subs("x",x2) + funcao.subs("x",x3))
        return valor_final.n()
    else: # Funciona com pontos
        if len(funcao_ou_Pontos) < 4: raise ValueError("Precisa de minimo 4
(Pontos)")
        ponto0, ponto1, ponto2, ponto3 = funcao_ou_Pontos[0],
funcao_ou_Pontos[1], funcao_ou_Pontos[2], funcao_ou_Pontos[3]
        valor_final = 3*h/8*(ponto0.y + 3*ponto1.y + 3*ponto2.y + ponto3.y)
        return valor_final.n()
```

Código 3 - Regra de Simpson 3/8

Descrição:

Esta função implementa a **Regra de Simpson 3/8**, uma fórmula de Newton-Cotes Fechada com $n = 3$.

Utiliza quatro pontos uniformemente espaçados (x_0, x_1, x_2, x_3) para calcular a integral.

Entrada:

- funcao_ou_Pontos: Pode ser uma função matemática ou uma lista de pontos.
- intervalo: Define os limites de integração $[x_0, x_4]$.

Lógica:

Calcula $h = \frac{x_3 - x_0}{3}$ e define os pontos intermediários $x_1 = x_0 + h$ e $x_2 = x_0 + 2h$.

Com uma função:

- Avalia os valores $f(x_0), f(x_1), f(x_2), f(x_3)$.
- Aplica a fórmula: $\int_{x_0}^{x_3} f(x)dx \approx \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$.

Com pontos:

- Usa diretamente as ordenadas (y) dos pontos para calcular a integral.

Função regra_de_Boole:

```
def regra_de_Boole(funcao_ou_Pontos , intervalo:tuple):
    x0, x4 = intervalo
    h = (x4 - x0) / 4
    x1 = x0 + h
    x2 = x0 + 2*h
    x3 = x0 + 3*h
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        valor_final = 2*h/45*(7*funcao.subs("x",x0) +
32*funcao.subs("x",x1) + 12*funcao.subs("x",x2) + 32*funcao.subs("x",x3) +
7*funcao.subs("x",x4))
        return valor_final.n()
    else: # Funciona com pontos
        if len(funcao_ou_Pontos) < 5: raise ValueError("Precisa de minimo
5 (Pontos)")
        ponto0, ponto1, ponto2, ponto3, ponto4 = funcao_ou_Pontos[0],
funcao_ou_Pontos[1], funcao_ou_Pontos[2], funcao_ou_Pontos[3],
funcao_ou_Pontos[4]
        valor_final = 2*h/45*(7*ponto0.y + 32*ponto1.y + 12*ponto2.y +
32*ponto3.y + 7*ponto4.y)
        return valor_final.n()
```

Código 4 - Regra de Boole

Descrição:
Implementa a **Regra de Boole** ($\boldsymbol{n = 4}$), utilizando cinco pontos $(x_0, x_1, x_2, x_3, x_4)$.
É usada para aproximações, com um polinómio de grau 4 que modela a função.
Entrada:
- funcao_ou_Pontos: Aceita uma função ou lista de pontos uniformemente espaçados.
- intervalo: Limites de integração $[x_0, x_4]$.

Lógica:
Calcula $h = \frac{x_4 - x_0}{4}$ e determina os pontos x_1, x_2, x_3.
Se for uma função:
- Avalia a função nos cinco pontos: $f(x_0), f(x_1), f(x_2), f(x_3), f(x_4)$.
- Aplica a fórmula: $\int_{x_0}^{x_4} f(x)dx \approx \frac{2h}{45}[7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)]$

Com pontos:
- Utiliza diretamente as ordenadas (y) fornecidas para calcular a integral.

Fórmulas de Newton-Cotes Abertas

Função regra_do_ponto_medio:

```
def regra_do_ponto_medio(funcao_ou_Pontos, intervalo:tuple):
    a, b = intervalo
    h = (b-a)/2
    x0 = a + h
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        valor_final = 2*h*funcao.subs("x",x0)
        return valor_final.n()
    else:
        if len(funcao_ou_Pontos) < 1: raise ValueError("Precisa de minimo 1 
(Pontos) sem contar com os pontos dos extremos")
        ponto0 = funcao_ou_Pontos[0]
        valor_final = 2*h*ponto0.y
        return valor_final.n()
```

Código 5 - Regra do Ponto Médio

Descrição:

Implementa a **Regra do Ponto Médio** $(\boldsymbol{n})$, utilizando um único ponto interior.

Entrada:

- funcao_ou_Pontos: Função simbólica ou lista de pontos.
- intervalo: Define os limites $[a, b]$.

Lógica:

- Calcula $h = \frac{b-a}{2}$ e $x_0 = a + h$.
- Avalia a função em x_0 ou utiliza a ordenada (y) do ponto fornecido.
- Aplica a fórmula: $\int_a^b f(x)dx \approx 2h \times f(x_0)$.

Função regra_do_grau_1:

```
def regra_do_grau_1(funcao_ou_Pontos , intervalo:tuple):
    a, b = intervalo
    h = (b-a)/3
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        x0 = a + h
        x1 = a + 2*h
        valor_final = 3*h/2 *(funcao.subs("x",x0) + funcao.subs("x",x1) )
        return valor_final.n()
    else:
        if len(funcao_ou_Pontos) < 2: raise ValueError("Precisa de minimo 2
(Pontos) sem contar com os pontos dos extremos")
        ponto0, ponto1 = funcao_ou_Pontos[0], funcao_ou_Pontos[1]
        valor_final = 3*h/2*(ponto0.y + ponto1.y)
        return valor_final.n()
```

Código 6 - Regra de 1° Grau

Descrição:

Implementa a **Regra do Grau 1** ($\boldsymbol{n = 1}$), usando dois pontos interiores.

Entrada:

- funcao_ou_Pontos: Função simbólica ou lista de pontos.
- intervalo: Define os limites $[a, b]$.

Lógica:

- Calcula $h = \frac{b-a}{3}$ e $x_0 = a + h, x_1 = a + 2h$.
- Avalia a função em x_0 e x_1 ou usa as ordenadas (y) dos pontos fornecidos.
- Aplica a fórmula: $\int_a^b f(x)dx \approx 3h[f(x_0) + f(x_1)]$.

Função regra_de_Milne:

```
def regra_de_Milne(funcao_ou_Pontos , intervalo:tuple):
    a, b = intervalo
    h = (b-a)/4
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        x0 = a + h
        x1 = a + 2*h
        x2 = a + 3*h
        valor_final = 4*h/3*(2*funcao.subs("x",x0) - funcao.subs("x",x1) +
2*funcao.subs("x",x2))
        return valor_final.n()
    else:
        if len(funcao_ou_Pontos) < 3: raise ValueError("Precisa de minimo 3
(Pontos) sem contar com os pontos dos extremos")
        ponto0, ponto1, ponto2 = funcao_ou_Pontos[0], funcao_ou_Pontos[1],
funcao_ou_Pontos[2]
        valor_final = 4*h/3*(2*ponto0.y - ponto1.y + 2*ponto2.y)
        return valor_final.n()
```

Código 7 - Regra de Milne

Descrição:

Implementa a **Regra de Milne** $(\boldsymbol{n = 2})$, usando três pontos interiores.

Entrada:

- funcao_ou_Pontos: Função simbólica ou lista de pontos.
- intervalo: Define os limites $[a, b]$.

Lógica:

- Calcula $h = \frac{b-a}{4}$ e $x_0 = a + h, x_1 = a + 2h, x_2 = a + 3h$.
- Avalia a função nos pontos ou usa as ordenadas (y) fornecidas.
- Aplica a fórmula: $\int_a^b f(x)dx \approx \frac{4h}{3}[2f(x_0) - f(x_1) + 2f(x_2)]$.

Função regra_do_grau_3:

```
def regra_do_grau_3(funcao_ou_Pontos, intervalo:tuple):
    a, b = intervalo
    h = (b-a)/5
    if not isinstance(funcao_ou_Pontos, list) :
        funcao = funcao_ou_Pontos
        x0 = a + h
        x1 = a + 2*h
        x2 = a + 3*h
        x3 = a + 4*h
        valor_final = 5*h/24*(11*funcao.subs("x",x0) + funcao.subs("x",x1) +
funcao.subs("x",x2) + 11*funcao.subs("x",x3) )
        return valor_final.n()
```

Código 8 - Regra do 3º Grau

Descrição:

Implementa a **Regra do Grau 3** ($\boldsymbol{n = 3}$), utilizando quatro pontos interiores.

Entrada:

- funcao_ou_Pontos: Função simbólica ou lista de pontos.
- intervalo: Define os limites $[a, b]$.

Lógica:

- Calcula $h = \frac{b-a}{5}$, x_0, x_1, x_2, x_3 como pontos uniformemente espaçados.
- Avalia a função ou utiliza as ordenadas (y).
- Aplica a fórmula: $\int_a^b f(x)dx \approx \frac{5h}{24}[11f(x_0) + f(x_1) + f(x_2) + 11f(x_3)]$

FLUXO DE EXECUÇÃO

1. Início e Interface com o Utilizador

O programa inicia e apresenta opções ao utilizador para escolher:

- O tipo de fórmula a ser usada:
  - **Fechada** (inclui os extremos do intervalo).
  - **Aberta** (exclui os extremos do intervalo).
- O grau do método desejado:
  - Para fórmulas fechadas:
    - $n = 1$: Regra do Trapézio.
    - $n = 2$: Regra de Simpson.
    - $n = 3$: Regra de Simpson 3/83/83/8.
    - $n = 4$: Regra de Boole.
  - Para fórmulas abertas:
    - $n = 0$: Regra do Ponto Médio.
    - $n = 1$: Regra do Grau 1.
    - $n = 2$: Regra de Milne.
    - $n = 3$: Regra do Grau 3.

2. Entrada de Dados

O utilizador insere:

- Uma **função matemática** ou **valores de pontos** (abscissas e ordenadas).
- O **intervalo de integração**.

O programa processa os dados:

- **Funções** são convertidas para um formato simbólico utilizável pelo SymPy.
- **Pontos** são validados para garantir uniformidade e consistência.

3. Seleção da Classe e Método

Com base na escolha do utilizador:

- Para **Fórmulas Fechadas**:
    - A classe Newton_Cotes_Fechadas é instanciada, e o método correspondente ao grau é selecionado.
- Para **Fórmulas Abertas**:
    - A classe Newton_Cotes_Abertas é instanciada, e o método correspondente ao grau é selecionado.

4. Execução do Método

A função ou pontos fornecidos são usados para calcular a integral no intervalo indicado.

- O método apropriado é chamado com os dados fornecidos.
- Os valores são substituídos nas fórmulas implementadas no programa, seguindo as especificações de cada regra (Trapézio, Simpson, etc.).

5. Apresentação dos Resultados

O valor aproximado da integral é calculado e arredondado a 8 casas decimais, conforme configurado no programa.

- O resultado é exibido no terminal ou interface, junto com mensagens de validação (se aplicável).

6. Validações e Tratamento de Erros

O programa verifica:

- **Uniformidade dos pontos**: Para garantir que os métodos funcionem corretamente.
- **Coerência dos extremos**: Confirma que o intervalo e os dados fornecidos estão consistentes.
- **Número mínimo de pontos**: Valida que o método escolhido possui os pontos necessários.

Mensagens de erro são exibidas se:

- O utilizador escolher um método sem fornecer pontos suficientes.
- Os pontos fornecidos não forem uniformemente espaçados.

RESULTADOS

Fórmulas Fechadas

```
--------------------------------------------------------------------
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
1
Qual o grau da formula de Newton Cotes-Fechadas?
 1)Regra do Trapezio
 2)Regra de Simpson 1/3
 3)Regra de Simpson 3/8
 4)Regra de Boole
1
--------------------------------------------------------------------
--> O metodo escolhido precisa de minimo 2 pontos
Inserir as Abscissas(separadas com espaço) ou a Função
x^2
--------------------------------------------------------------------
Inserir o Intervalo de integração
0 2
--------------------------------------------------------------------
--> Resultado: 4.00000000000000
```

Ilustração 1 - Fórmula Fechada (R. Trapézio)

Regra do Trapézio $(\boldsymbol{n} = \mathbf{1})$
Função fornecida: $f(x^2)$.
Intervalo de integração: $[0{,}2]$.
Resultado calculado: 4.0.
Conclusão: O resultado é o valor exato da integral no intervalo.

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
1
Qual o grau da formula de Newton Cotes-Fechadas?
 1)Regra do Trapezio
 2)Regra de Simpson 1/3
 3)Regra de Simpson 3/8
 4)Regra de Boole
2
--------------------------------------------------------------------
--> O metodo escolhido precisa de minimo 3 pontos
Inserir as Abscissas(separadas com espaço) ou a Função
sin(x)
--------------------------------------------------------------------
Inserir o Intervalo de integração
0 pi
--------------------------------------------------------------------
--> Resultado: 2.09439510
```

Ilustração 2 - Fórmula Fechada (R. Simpson 1/3)

Regra de Simpson 1/3 $(\boldsymbol{n} = \mathbf{2})$
Função fornecida: $f(x) = \sin(x)$.
Intervalo de integração: $[0, \pi]$.
Resultado calculado: 2.09439510.
Conclusão: O resultado aproxima o valor da integral, com alta precisão.

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
1
Qual o grau da formula de Newton Cotes-Fechadas?
 1)Regra do Trapezio
 2)Regra de Simpson 1/3
 3)Regra de Simpson 3/8
 4)Regra de Boole
3
--------------------------------------------------------------------
--> O metodo escolhido precisa de minimo 4 pontos
Inserir as Abscissas(separadas com espaço) ou a Função
e^x
--------------------------------------------------------------------
Inserir o Intervalo de integração
0 1
--------------------------------------------------------------------
--> Resultado: 1.71854015
```

Ilustração 3 - Fórmula Fechada (Regra Simpson 3/8)

Regra de Simpson 3/8 $(\boldsymbol{n} = \mathbf{2})$
Função fornecida: $f(e^x)$.
Intervalo de integração: $[0{,}1]$.
Resultado calculado: 1.71854015.
Conclusão: O resultado é muito próximo do valor real da integral.

```
--------------------------------------------------------------------
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
1
Qual o grau da formula de Newton Cotes-Fechadas?
 1)Regra do Trapezio
 2)Regra de Simpson 1/3
 3)Regra de Simpson 3/8
 4)Regra de Boole
4
--------------------------------------------------------------------
--> O metodo escolhido precisa de minimo 5 pontos
Inserir as Abscissas(separadas com espaço) ou a Função
x^4
--------------------------------------------------------------------
Inserir o Intervalo de integração
0 1
--------------------------------------------------------------------
--> Resultado: 0.20000000
```

Ilustração 4 - Fórmula Fechada (R. Boole)

Regra de Boole $(\boldsymbol{n} = \mathbf{4})$
Função fornecida: $f(x^4)$.
Intervalo de integração: $[0{,}1]$.
Resultado calculado: 0.2.
Conclusão: O resultado é exato para a integral da função.

Fórmulas Abertas

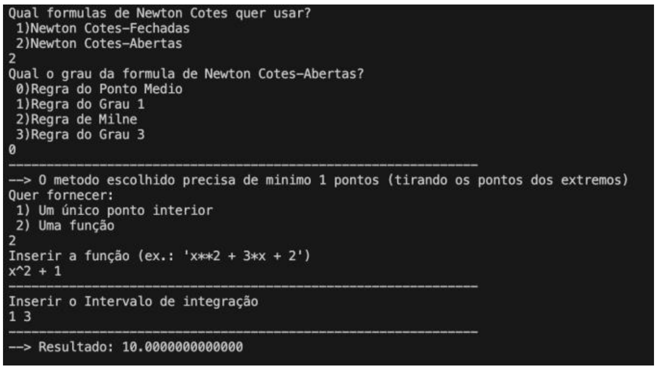

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
0
-------------------------------------------------------------
--> O metodo escolhido precisa de minimo 1 pontos (tirando os pontos dos extremos)
Quer fornecer:
 1) Um único ponto interior
 2) Uma função
2
Inserir a função (ex.: 'x**2 + 3*x + 2')
x^2 + 1
-------------------------------------------------------------
Inserir o Intervalo de integração
1 3
-------------------------------------------------------------
--> Resultado: 10.0000000000000
```

Ilustração 5 - Fórmula Aberta (R. Ponto Médio)

Regra do Ponto Médio $(\boldsymbol{n} = \mathbf{0})$
Função fornecida: $f(x^2 + 1)$.
Intervalo de integração: $[0,3]$.
Resultado calculado: 10.0.
Conclusão: O resultado representa o valor da integral da função no intervalo especificado.

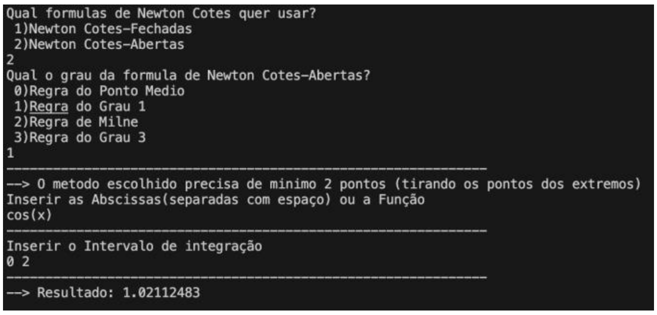

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
1
-------------------------------------------------------------
--> O metodo escolhido precisa de minimo 2 pontos (tirando os pontos dos extremos)
Inserir as Abscissas(separadas com espaço) ou a Função
cos(x)
-------------------------------------------------------------
Inserir o Intervalo de integração
0 2
-------------------------------------------------------------
--> Resultado: 1.02112483
```

Ilustração 6 - Fórmula Aberta (R. 1º Grau)

Regra do Grau 1 $(\boldsymbol{n} = \mathbf{1})$
Função fornecida: $f(x) = \cos{(x)}$.
Intervalo de integração: $[0,2]$.
Resultado calculado: 1.02112483.
Conclusão: O valor da integral aproxima a área sob o gráfico da função no intervalo dado.

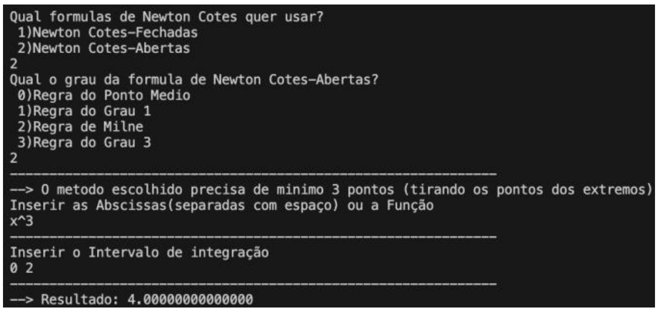

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
2
-------------------------------------------------------------
--> O metodo escolhido precisa de minimo 3 pontos (tirando os pontos dos extremos)
Inserir as Abscissas(separadas com espaço) ou a Função
x^3
-------------------------------------------------------------
Inserir o Intervalo de integração
0 2
-------------------------------------------------------------
--> Resultado: 4.00000000000000
```

Ilustração 7 - Fórmula Aberta (R. Milne)

Regra de Milne $(\boldsymbol{n} = \mathbf{2})$
Função fornecida: $f(x^3)$.
Intervalo de integração: $[0,2]$.
Resultado calculado: 4.0.
Conclusão: O resultado é exato e coincide com o valor analítico da integral da função.

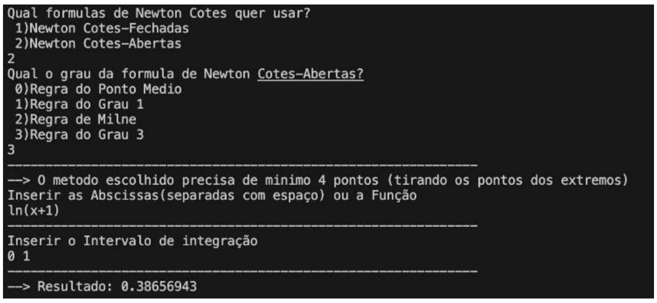
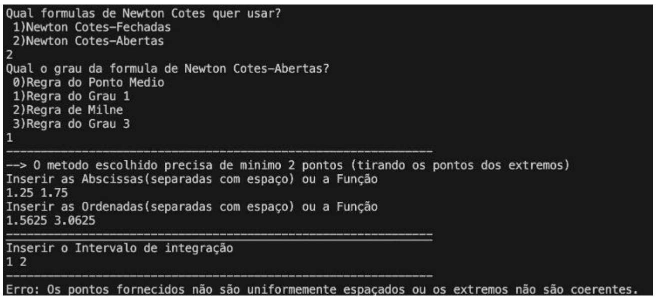

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
3
-------------------------------------------------------------
--> O metodo escolhido precisa de minimo 4 pontos (tirando os pontos dos extremos)
Inserir as Abscissas(separadas com espaço) ou a Função
ln(x+1)
-------------------------------------------------------------
Inserir o Intervalo de integração
0 1
-------------------------------------------------------------
--> Resultado: 0.38656943
```

Ilustração 8 - Fórmula Aberta (R. 3º Grau)

Regra do Grau 3 $(\boldsymbol{n} = \mathbf{3})$
Função fornecida: $f(x) = \ln{(x + 1)}$.
Intervalo de integração: $[0,1]$.
Resultado calculado: 0.38656943.
Descrição: O resultado aproxima o valor da integral com alta precisão.

Resultados de Erro



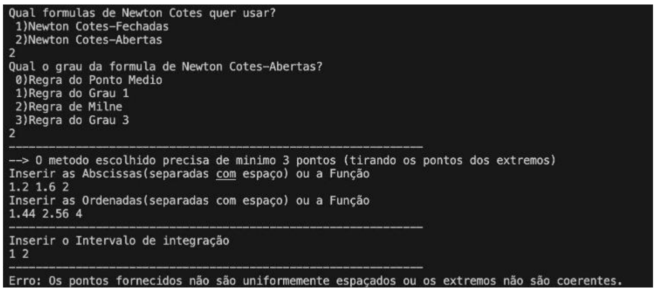
```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
1
--------------------------------------------------------------
--> O metodo escolhido precisa de minimo 2 pontos (tirando os pontos dos extremos)
Inserir as Abscissas(separadas com espaço) ou a Função
1.25 1.75
Inserir as Ordenadas(separadas com espaço) ou a Função
1.5625 3.0625
--------------------------------------------------------------
Inserir o Intervalo de integração
1 2
--------------------------------------------------------------
Erro: Os pontos fornecidos não são uniformemente espaçados ou os extremos não são coerentes.
```

Ilustração 9 - Execução de Erro 1

Erro ao usar a Regra do Grau 1 $(n = 1)$
Abscissas fornecidas: $1.25, 1.75$.
Ordenadas fornecidas: $1.6525, 3.4625$.
Intervalo de integração: $[1,2]$.
Erro: "Os pontos fornecidos não são uniformemente espaçados ou os extremos não são coerentes."

Explicação:

- Para aplicar a **Regra do Grau 1**, os pontos fornecidos (abscissas e ordenadas) devem ser **uniformemente espaçados**.
- O espaçamento entre as abscissas 1.25 e 1.75 é 0.50, mas o intervalo definido $[1,2]$ não é coerente com este espaçamento.
- Além disso, os extremos do intervalo não coincidem com a posição esperada dos pontos uniformes calculados.



```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
2
--------------------------------------------------------------
--> O metodo escolhido precisa de minimo 3 pontos (tirando os pontos dos extremos)
Inserir as Abscissas(separadas com espaço) ou a Função
1.2 1.6 2
Inserir as Ordenadas(separadas com espaço) ou a Função
1.44 2.56 4
--------------------------------------------------------------
Inserir o Intervalo de integração
1 2
--------------------------------------------------------------
Erro: Os pontos fornecidos não são uniformemente espaçados ou os extremos não são coerentes.
```

Ilustração 10 - Execução de Erro 2

Erro ao usar a Regra de Milne $(n = 2)$
Abscissas fornecidas: $1.2, 1.6$.
Ordenadas fornecidas: $1.44, 2.56$.
Intervalo de integração: $[1,2]$.
Erro: "Os pontos fornecidos não são uniformemente espaçados ou os extremos não são coerentes."

Explicação:

- Para a **Regra de Milne**, é necessário que as abscissas sejam uniformemente espaçadas dentro do intervalo $[1,2]$.
- Os espaçamentos entre as abscissas fornecidas $(1.2, 1.6)$ não são consistentes:
- Os extremos do intervalo $[1,2]$ também não correspondem ao espaçamento calculado para o método.

Resultado Específico

```
Qual formulas de Newton Cotes quer usar?
 1)Newton Cotes-Fechadas
 2)Newton Cotes-Abertas
2
Qual o grau da formula de Newton Cotes-Abertas?
 0)Regra do Ponto Medio
 1)Regra do Grau 1
 2)Regra de Milne
 3)Regra do Grau 3
0
--------------------------------------------------------------
--> O metodo escolhido precisa de minimo 1 pontos (tirando os pontos dos extremos)
Quer fornecer:
 1) Um único ponto interior
 2) Uma função
1
Inserir a abscissa (o ponto interior):
1.5
Inserir a ordenada correspondente (valor da função no ponto):
2.25
--------------------------------------------------------------
Inserir o Intervalo de integração
1 2
--------------------------------------------------------------
--> Resultado: 2.25000000
```

Ilustração 11 - Execução Específica

O que aconteceu?
Neste caso, o utilizador escolheu a **Regra do Ponto Médio**, que exige um único ponto interior para calcular a aproximação da integral.

Entrada de Dados:

- A abscissa 1.5 foi inserida como o ponto interior, localizada no meio do intervalo $[1,2]$.
- A ordenada correspondente, 2.25, representa o valor da função no ponto $x = 1.5$.

Cálculo:

- O programa utiliza a fórmula da **Regra do Ponto Médio**:

$$\int_a^b f(x)dx \approx 2h \times f(x_0)$$

onde $h = \frac{b-a}{2}$, x_0 é o ponto médio, e $f(x_0)$ é o valor da função no ponto interior.

Neste exemplo:

- $a = 1, b = 2$ e $h = \frac{2-1}{2} = 0.5$.
- O valor calculado foi: $2 \times 0.5 \times 2.25 = 2.25$.

Por que é um caso específico?

Este exemplo mostra que a Regra do Ponto Médio pode ser aplicada diretamente com um único ponto interior, usando o valor da função no ponto médio. É um caso simples e eficiente para intervalos pequenos ou funções bem-comportadas.

CONCLUSÃO

O presente trabalho permitiu explorar a aplicação prática das **fórmulas de Newton-Cotes**, tanto **fechadas** quanto **abertas**, para o cálculo de integrais definidas. A implementação computacional desenvolvida em Python demonstrou a eficácia dos métodos numéricos na aproximação de integrais, mesmo em cenários onde a solução analítica é difícil ou inviável.

Os resultados obtidos validaram a precisão dos métodos de maior grau, como a **Regra de Boole** e a **Regra do Grau 3**, que se mostraram altamente confiáveis para funções polinomiais e suaves. Por outro lado, métodos mais simples, como a **Regra do Trapézio** e a **Regra do Ponto Médio**, apresentaram desempenho satisfatório em cenários de menor complexidade e com funções bem-comportadas.

Adicionalmente, a verificação e validação dos dados de entrada reforçaram a importância de garantir uniformidade nos pontos fornecidos e coerência nos intervalos de integração, evitando erros computacionais. Casos de erro foram devidamente tratados, demonstrando a robustez do programa.

Este trabalho também destacou o equilíbrio entre simplicidade computacional e precisão numérica, evidenciando que a escolha do método mais adequado depende tanto das características da função quanto do nível de precisão desejado.

Como possível continuidade, sugere-se:

1. Implementar métodos adaptativos, como a subdivisão automática do intervalo para melhorar a precisão.
2. Comparar os métodos de Newton-Cotes com outras técnicas, como quadraturas gaussianas, em diferentes tipos de funções.

Para concluir. O presente projeto contribuiu para um maior entendimento sobre a integração numérica e seu papel crucial em resolver problemas matemáticos em contextos computacionais.

REFERÊNCIAS

Burden, R. L., & Faires, J. D. (2011). *Análise Numérica*. Cengage Learning.

- Utilizado como base teórica para as fórmulas de Newton-Cotes e métodos numéricos.

Chapra, S. C., & Canale, R. P. (2015). *Métodos Numéricos para Engenharia*. McGraw-Hill.

- Referência complementar sobre integração numérica e aplicações práticas.

SymPy Documentation (2024). Disponível em: https://docs.sympy.org

- Consultado para o uso de operações simbólicas no Python.

Python Official Documentation (2024). Disponível em: https://docs.python.org

- Documentação oficial do Python utilizada no desenvolvimento do programa.

Material de Aula - Capítulo 6: Derivação e Integração Numérica (2024).

- Fornecido pelo docente da disciplina como apoio teórico e prático.