A picture containing text

Description automatically generated



INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR

DE ENGENHARIA

DE COIMBRA

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

**Métodos Numéricos para Derivação e Integração**

Relatório de Licenciatura

Autores

**Ana Rita Conceição Pessoa – 2023112690**

**João Francisco de Matos Claro – 2017010293**

Coimbra, maio e 2024

# Índice

## Índice de texto

[1 Índice 2](#_Toc167961391)

[1.1 Índice de texto 2](#_Toc167961392)

[1.2 Índice de figuras 3](#_Toc167961393)

[2 Lista de siglas, acrónimos e símbolos 4](#_Toc167961394)

[2.1 Lista de siglas e acrónimos 4](#_Toc167961395)

[2.2 Lista de símbolos 4](#_Toc167961396)

[2.2.1 Exemplos de listas de símbolos 4](#_Toc167961397)

[3 Introdução 5](#_Toc167961398)

[4 Métodos Numéricos para Derivação 6](#_Toc167961399)

[4.1 Fórmulas de Diferenças Finitas em 2 pontos 7](#_Toc167961400)

[4.1.1 Progressivas 7](#_Toc167961401)

[4.1.2 Regressivas 9](#_Toc167961402)

[4.2 Fórmulas de Diferenças Finitas em 3 pontos 11](#_Toc167961403)

[4.2.1 Progressivas 11](#_Toc167961404)

[4.2.3 Centradas 15](#_Toc167961405)

[4.2.4 Segunda Derivada 17](#_Toc167961406)

[4.2 Funções simbólicas no MatLab 19](#_Toc167961407)

[4.2.1 Função *diff* () 20](#_Toc167961408)

[4.2.2 Função *int ()* 24](#_Toc167961409)

[5 Métodos Numéricos para Integração 27](#_Toc167961410)

[5.1 Regra dos Trapézios Simples 28](#_Toc167961411)

[5.2 Regra dos Trapézios Composta 30](#_Toc167961412)

[5.3 Regra de Simpson Simples 32](#_Toc167961413)

[Fórmula do erro para a Regra de Simpson Simples: 32](#_Toc167961414)

[5.4 Regra de Simpson Composta 34](#_Toc167961415)

[6 Função Harmónica 37](#_Toc167961416)

[7 Exemplos de aplicação e teste dos métodos 38](#_Toc167961417)

[7.1 Função de Variável Real e Intervalo 38](#_Toc167961418)

[7.1.1 Derivação 38](#_Toc167961419)

[7.1.2 Integração 42](#_Toc167961420)

[7.2 Função de Duas Variáveis Reais e Intervalos 43](#_Toc167961421)

[8 Conclusão 45](#_Toc167961422)

[9 Bibliografia 46](#_Toc167961423)

[10 Autoavaliação e heteroavaliação do trabalho submetido 47](#_Toc167961424)

## Índice de figuras

[Figura 1 - Aproximação de Derivadas com diff 22](#_Toc167833128)

[Figura 2 - Diferenças Finitas Progressivas (2 pontos) 38](#_Toc167833129)

[Figura 3 - Diferenças Finitas Regressivas (2 pontos) 39](#_Toc167833130)

[Figura 4 - Diferenças Finitas Progressivas (3 pontos) 39](#_Toc167833131)

[Figura 5 - Diferenças Finitas Regressivas (3 pontos) 40](#_Toc167833132)

[Figura 6 - Diferenças Finitas Centradas (3 pontos) 40](#_Toc167833133)

[Figura 7 - Diferenças Finitas 2ª Derivada 41](#_Toc167833134)

[Figura 8 - Regra dos Trapézios Composta 42](#_Toc167833135)

[Figura 9 – Regra Simpson Composta 42](#_Toc167833136)

[Figura 10 - Função de Duas Variáveis Reais Não Harmónica 43](#_Toc167833137)

[Figura 11 - Função de Duas Variáveis Reais Harmónica 44](#_Toc167833138)

# Lista de siglas, acrónimos e símbolos

## Lista de siglas e acrónimos

|  |  |
| --- | --- |
| GUI | Graphical User Interface |

## Lista de símbolos

### Exemplos de listas de símbolos

**Alfabeto grego**

|  |  |
| --- | --- |
| Σ | Soma dos termos da série |
|  | Ponto dentro do intervalo de integração |
| Δ | Operador Laplaciano |
| ∂ | Derivada parcial |

# Introdução

Neste trabalho da unidade curricular de Análise Matemática II - Matemática Computacional, exploramos os conceitos de derivada e integral, além de criar uma ferramenta em MATLAB com uma interface gráfica (GUI) para calcular essas operações analiticamente e numericamente.

Sabemos que a derivada é a velocidade de mudança instantânea, ou seja, a velocidade com que uma função se altera em relação à sua variável independente num ponto específico. Esta é determinada ao calcular o limite da diferença das variações, à medida que o intervalo de variação da variável independente se aproxima de zero.

Adicionalmente, sabemos que a integral é uma operação que envolve a acumulação de quantidades e é o oposto da diferenciação. Calcular integrais é crucial em matemática e física, pois permite encontrar áreas sob curvas, volumes de sólidos com formas irregulares e muitas outras grandezas.

Os métodos de derivação e integração numérica incluem diferenças finitas, a segunda derivada, a regra dos Trapézios e a regra de Simpson, além da derivação e integração simbólica no MATLAB.

A "Máquina para Derivação e Integração" utiliza três segmentos numa GUI: dois para Derivação Numérica e Integração Numérica de funções reais de variável real, e outro para Derivação Numérica de funções reais de 2 variáveis reais.

O projeto é dividido em duas partes: métodos numéricos e simbólicos, e exemplos de aplicação. Visa também melhorar as competências de programação em MATLAB, utilizando as GUIs fornecidas pela disciplina, com arquivos disponíveis no moodle.

# Métodos Numéricos para Derivação

Entender e calcular derivadas é crucial em várias áreas, mas muitas vezes o processo é computacionalmente demorado. Para lidar com esta questão, surgiram as aproximações numéricas, como as fórmulas de diferenças finitas: progressivas, regressivas e centradas.

A Derivação Numérica é vital em situações onde não se dispõe da função completa, mas apenas de um conjunto de pontos que a representam, ou quando a função não é derivável em todo o seu domínio ou tem uma derivação não trivial.

O método baseia-se na redução de um intervalo , aproximando-se assim do valor real da derivada. No entanto, mesmo com intervalos pequenos, o método pode ter um erro de arredondamento considerável. Para minimizar este erro, recorre-se à utilização de vários pontos.

O processo envolve começar com um conjunto de pontos que definem um intervalo e determinar a função que os representa, interpolando este conjunto de pontos. Em seguida, é possível calcular a derivada da função e aplicá-la em qualquer ponto dentro do intervalo Quanto mais pontos forem utilizados, mais preciso será o resultado.

## Fórmulas de Diferenças Finitas em 2 pontos

Uma forma simples de aproximar a derivada é através do método das diferenças finitas. Uma diferença finita é uma expressão na forma , que ao ser dividida por é conhecida como quociente de diferenças. A técnica das diferenças finitas consiste em estimar a derivada de uma função usando fórmulas discretas que necessitam apenas de um conjunto finito de pares ordenados, onde normalmente representamos

### Progressivas

**Fórmula:**

Onde:

* Aproximação do valor da derivada no ponto de abcissa ;
* Valor da função na próxima abcissa;
* Valor da função no ponto de abcissa atual;
* Valor de cada subintervalo (passo).

**Algoritmo:**

1. Alocar memória para ;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, recebe o valor de ;
4. Alocar memória para a derivada;
5. Para ***i*** de ***1*** a ***n-1***, calcular a derivada (aproximada) de no ponto atual, para a iésima iteração;
6. Calcular a derivada (aproximada) de f no ponto atual, em n.

**Função (MatLab):**

function [x,y,dydx] = NDerivadaDFP(~,f,a,b,h,y)

% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)

x = a:h:b;

% Número de pontos (tamanho do vetor de abcissas)

n = length(x);

% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’.

if nargin == 5

y = f(x); % y é a função de f(x)

end

% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)

dydx = zeros(1,n);

% Utiliza um loop for para calcular as derivadas aproximadas de f nos pontos de x, exceto no último ponto. A derivada é calculada usando a fórmula das diferenças finitas progressivas: (y(i+1) - y(i)) / h.

for i = 1:n-1

% Derivada (aproximada) de f no ponto atual

dydx(i) = (y(i+1)-y(i))/h;

end

% Usar regressiva para calcular o último ponto. Isto é feito porque não há um ponto após ‘x(n)’ para aplicar a fórmula progressiva.

dydx(n) = (y(n)-y(n-1))/h;

end

### Regressivas

Onde:

* Aproximação do valor da derivada no ponto de abcissa ;
* Valor da função na próxima abcissa atual;
* Valor da função no ponto de abcissa anterior;
* Valor de cada subintervalo (passo).

**Algoritmo:**

1. Alocar memória para ;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, recebe o valor de ;
4. Alocar memória para a derivada;
5. Calcular a derivada (aproximada) de f no ponto atual, em ***1***.
6. Para ***i*** de ***2*** a ***n***, calcular a derivada (aproximada) de no ponto atual, para a iésima iteração;

**Função (MatLab):**

function [x,y,dydx] = NDerivadaDFR(~,f,a,b,h,y)

% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)

x = a:h:b;

% Número de pontos (tamanho do vetor de abcissas)

n = length(x);

% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’

if nargin == 5

y = f(x);

end

% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)

dydx = zeros(1,n);

% Para o primeiro ponto, a derivada é calculada usando a fórmula de diferenças finitas progressivas: ‘(y(2) - y(1)) / h’. Isto é necessário porque não há um ponto anterior a ‘x(1)’ para aplicar a fórmula regressiva

dydx(1)=(y(2)-y(1))/h;

% Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, começando do segundo ponto até o último ponto. A derivada é calculada usando a fórmula das diferenças finitas regressivas: ‘(y(i) - y(i-1)) / h’

for i=2:n

dydx(i) = (y(i)-y(i-1))/h;

end

end

## 4.2 Fórmulas de Diferenças Finitas em 3 pontos

### 4.2.1 Progressivas

Onde:

* Aproximação do valor da derivada no ponto de abcissa ;
* Valor da função na próxima abcissa atual;
* Valor da função na próxima abcissa;
* Valor da função 2 abcissas à frente;
* Valor de cada subintervalo (passo).

**Algoritmo:**

1. Alocar memória para ;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, recebe o valor de ;
4. Alocar memória para a derivada;
5. Para ***i*** de **1** a ***n-2***, calcular a derivada (aproximada) de no ponto atual, para a iésima iteração;
6. Calcular a derivada (aproximada) de f no ponto atual, em ***n-1***.
7. Calcular a derivada (aproximada) de f no ponto atual, em ***n***.

**Função (MatLab):**

function [x,y,dydx] = NDerivadaDFP3P(~,f,a,b,h,y)

% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)

x = a:h:b;

% Número de pontos (tamanho do vetor de abcissas)

n = length(x);

% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’

if nargin == 5

y = f(x);

end

% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)

dydx = zeros(1,n);

% Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, começando do segundo ponto até ao 3º ponto. A derivada é calculada usando a fórmula das diferenças finitas progressiva de 3 pontos: ‘(-3\*y(i)+4\*y(i+1)-y(i+2))/(2\*h);’

for i=2:n-2

dydx(i) = (-3\*y(i)+4\*y(i+1)-y(i+2))/(2\*h);

end

% Cálculo da derivada no penúltimo ponto: Calcula a derivada no penúltimo ponto usando uma fórmula de diferenças finitas regressivas ajustada para garantir a precisão:

dydx(n-1)=(y(n-3) - 4\*y(n-2) + 3\*y(n-1))/(2\*h);

% Cálculo da derivada no último ponto: Calcula a derivada no último ponto usando uma fórmula de diferenças finitas regressivas ajustada:

dydx(n)=(y(n-2) - 4\*y(n-1) + 3\*y(n))/(2\*h);

end

**4.2.2 Regressivas**

Onde:

* Aproximação do valor da derivada no ponto de abcissa ;
* Valor da função 2 abcissas atrás;
* Valor da função na abcissa anterior;
* Valor da função na próxima abcissa atual;
* Valor de cada subintervalo (passo).

**Algoritmo:**

1. Alocar memória para ;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, recebe o valor de ;
4. Alocar memória para a derivada;
5. Calcular a derivada (aproximada) de f no ponto atual, em ***1***.
6. Calcular a derivada (aproximada) de f no ponto atual, em ***2***.
7. Para ***i*** de **3** a ***n***, calcular a derivada (aproximada) de no ponto atual, para a iésima iteração;

**Função (MatLab):**

function [x,y,dydx] = NDerivadaDFR3(~,f,a,b,h,y)

% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)

x = a:h:b;

% Número de pontos (tamanho do vetor de abcissas)

n = length(x);

% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’

if nargin == 5

y = f(x);

end

% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)

dydx = zeros(1,n);

% Cálculo da derivada no primeiro ponto: Calcula a derivada no primeiro ponto usando uma fórmula de diferenças finitas progressivas ajustada. Isto é necessário porque não há pontos anteriores a ‘x(1)’ para aplicar a fórmula regressiva.

dydx(1)=(-3\*y(1) + 4\*y(2) - y(3))/(2\*h);

% Cálculo da derivada no segundo ponto: Calcula a derivada no segundo ponto usando uma fórmula de diferenças finitas progressivas ajustada

dydx(2)=(-3\*y(2) + 4\*y(3) - y(4))/(2\*h);

% Cálculo das derivadas para os pontos restantes (método das diferenças finitas regressivas de 3 pontos): Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, começando do terceiro ponto até o último ponto. A derivada é calculada usando a fórmula das diferenças finitas regressivas de 3 pontos:

for i=3:n

dydx(i)=(y(i-2) - 4\*y(i-1) + 3\*y(i))/(2\*h);

end

end

### 4.2.3 Centradas

Onde:

* Aproximação do valor da derivada no ponto de abcissa ;
* Valor da função na próxima abcissa;
* Valor da função na abcissa anterior;
* Valor de cada subintervalo (passo).

**Algoritmo:**

1. Alocar memória para ;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, recebe o valor de ;
4. Alocar memória para a derivada;
5. Calcular a derivada (aproximada) de f no ponto atual, em ***1***.
6. Para ***i*** de **2** a ***n-1***, calcular a derivada (aproximada) de no ponto atual, para a iésima iteração;
7. Calcular a derivada (aproximada) de f no ponto atual, em ***n***.

**Função (MatLab):**

function [x,y,dydx] = NDerivadaDFC3P(~,f,a,b,h,y)

% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)

x = a:h:b;

% Número de pontos (tamanho do vetor de abcissas)

n = length(x);

% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’

if nargin == 5

y = f(x);

end

% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)

dydx = zeros(1,n);

% Cálculo da derivada no primeiro ponto: Calcula a derivada no primeiro ponto usando uma fórmula de diferenças finitas progressivas ajustada. Isto é necessário porque não há pontos anteriores a ‘x(1)’ para aplicar a fórmula centrada

dydx(1)=(-3\*y(1) + 4\*y(2) - y(3))/(2\*h);

% Cálculo das derivadas para os pontos intermediários (método das diferenças finitas centradas de 3 pontos): Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, do segundo ponto até o penúltimo ponto. A derivada é calculada usando a fórmula das diferenças finitas centradas de 3 pontos

for i=2:n-1

dydx(i)=(y(i+1)-y(i-1))/(2\*h);

end

% Cálculo da derivada no último ponto: Calcula a derivada no último ponto usando uma fórmula de diferenças finitas regressivas ajustada

dydx(n)=(y(n-2) - 4\*y(n-1) + 3\*y(n))/(2\*h);

end

### 4.2.4 Segunda Derivada

Onde:

* Aproximação do valor da 2ªderivada no ponto de abcissa ;
* Valor da função na próxima abcissa;
* Valor da função na abcissa atual;
* Valor da função na abcissa anterior;
* Valor de cada subintervalo (passo).

**Algoritmo:**

1. Alocar memória para ;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, recebe o valor de ;
4. Alocar memória para a derivada;
5. Calcular a 1ª derivada no ponto: ;
6. Calcular a 1ª derivada no ponto: ;
7. Calcular a 1ª derivada no ponto: ;
8. Calcular a 2ª derivada no ponto:;
9. Para ***i*** de **2** a ***n-1***, calcular a derivada (aproximada) de no ponto atual, para a iésima iteração;
10. Calcular a 1ª derivada no ponto: ;
11. Calcular a 1ª derivada no ponto: ;
12. Calcular a 1ª derivada no ponto: ;
13. Calcular a 2ª derivada no ponto:;

**Função (MatLab):**

function [x,y,dydx] = SegundaDerivada(~,f,a,b,h,y)

% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)

x = a:h:b;

% Número de pontos (tamanho do vetor de abcissas)

n = length(x);

% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’

if nargin == 5

y = f(x);

end

% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)

dydx = zeros(1,n);

% Cálculo das derivadas primeiras nos primeiros três pontos: Calcula as primeiras derivadas nos três primeiros pontos usando uma fórmula de diferenças finitas progressivas ajustada

dydx1=(-3\*y(1) + 4\*y(2) - y(3))/(2\*h);

dydx2=(-3\*y(2) + 4\*y(3) - y(4))/(2\*h);

dydx3=(-3\*y(3) + 4\*y(4) - y(5))/(2\*h);

% Cálculo da segunda derivada no primeiro ponto: Calcula a segunda derivada no primeiro ponto utilizando uma combinação linear das primeiras derivadas calculadas anteriormente

dydx(1)=(-3\*dydx1 + 2\*dydx2 - 3\*dydx3)/(2\*h);

% Cálculo das segundas derivadas para os pontos intermediários (método das diferenças finitas centradas): Utiliza um loop ‘for’ para calcular as segundas derivadas aproximadas de ‘f’ nos pontos de ‘x’, do segundo ponto até o penúltimo ponto. A segunda derivada é calculada usando a fórmula das diferenças finitas centradas

for i=2:n-1

dydx(i)=(y(i+1)-2\*y(i)-y(i-1))/(2\*h);

end

% Cálculo das derivadas primeiras nos últimos três pontos: Calcula as primeiras derivadas nos três últimos pontos usando uma fórmula de diferenças finitas regressivas ajustada

dydx1=(-3\*y(n-4) + 4\*y(n-3) - y(n-2))/(2\*h);

dydx2=(-3\*y(n-3) + 4\*y(n-2) - y(n-1))/(2\*h);

dydx3=(-3\*y(n-2) + 4\*y(n-1) - y(n))/(2\*h);

% Cálculo da segunda derivada no último ponto: Calcula a segunda derivada no último ponto utilizando uma combinação linear das primeiras derivadas calculadas anteriormente

dydx(n)=(-3\*dydx1 + 2\*dydx2 - 3\*dydx3)/(2\*h);

end

## Funções simbólicas no MatLab

O MATLAB, além de permitir o cálculo numérico, oferece também funcionalidades de cálculo simbólico através da sua toolbox de matemática simbólica, conhecida como "Symbolic Math Toolbox". Esta toolbox possibilita a realização de cálculos de várias naturezas, incluindo:

**Cálculo Numérico:** diferenciação, integração, determinação de limites, somatórios, séries de Taylor, entre outros.

**Álgebra linear:** operações como a inversa de uma matriz, cálculo de determinantes, obtenção de valores próprios, etc.

**Simplificação de expressões algébricas.**

**Obtenção de soluções analíticas** de equações algébricas e diferenciais.

**Transformadas:** Laplace, Z, Fourier, entre outras.

Esta ferramenta é extremamente útil para a resolução de problemas matemáticos complexos de forma simbólica, ampliando significativamente as capacidades do MATLAB no campo da matemática e da engenharia.

### Função *diff* ()

**Sintaxe:**

Onde:

Array de Entrada que pode ser um vetor, matriz, array multidimensional, tabela ou tabela de horários, com diversos tipos de dados permitidos.

Ordem da Diferença e, se não for fornecido, o valor padrão é 1; se ***n*** for maior que a dimensão, o comportamento varia.

Dimensão ao longo da qual calcular a diferença; se não for especificado, a operação é realizada na primeira dimensão com tamanho maior que 1.

Considerando um array de entrada bidimensional ‘p-por-m’, ‘A’:

* ***diff(A,1,1)*** trabalha nos elementos sucessivos nas colunas de **A** e devolve uma matriz de diferenças de tamanho **(p-1)-por-m**.
* ***diff(A,1,2)*** trabalha nos elementos sucessivos nas filas de **A** e devolve uma matriz de diferenças de **tamanho p-por-(m-1).**

Para calcula as diferenças entre elementos adjacentes de `X` ao longo da primeira dimensão do array cujo tamanho não seja 1:

* Se `X` for um vetor de comprimento `m`, então devolve um vetor de comprimento `m-1`. Os elementos de `Y` são as diferenças entre os elementos adjacentes de `X`.

Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]

* Se `X` for uma matriz não vazia e não vetorial de tamanho `p-por-m`, então devolve uma matriz de tamanho `(p-1)-por-m`, cujos elementos são as diferenças entre as linhas de `X`.

Y = [X(2,:)-X(1,:); X(3,:)-X(2,:); ... X(p,:)-X(p-1,:)]

* Se `X` for uma matriz vazia de 0-por-0, então devolve uma matriz vazia de 0-por-0.
* Se `X` for uma tabela ou tabela de horários de tamanho `p-por-m`, então devolve uma tabela ou tabela de horários de tamanho `(p-1)-por-m`, cujos elementos são as diferenças entre as linhas de `X`. Se `X` for uma tabela ou tabela de horários de tamanho 1-por-m, então o tamanho de `Y` é 0-por-m.

Para calcula a diferença de ordem ***n*** aplicando o operador diff(X) recursivamente ***n*** vezes. Na prática, isto significa que diff(X,2) é o mesmo que diff(diff(X)).

Para é a diferença de ordem ***n*** calculada ao longo da dimensão especificada por ***dim***. O parâmetro ***dim*** é um escalar inteiro positivo.

**Exemplos**

**Diferenças Entre Elementos de um Vetor**

Cria um vetor e calcula as diferenças entre os elementos.

X = [1 1 2 3 5 8 13 21];

Y = diff(X)

Y = *1×7*

0 1 1 2 3 5 8

**Diferenças Entre Linhas de uma Matriz**

Cria uma matriz 3x3 e calcula a primeira diferença entre as linhas.

X = [1 1 1; 5 5 5; 25 25 25];

Y = diff(X)

Y = *2×3*

4 4 4  
 20 20 20

**Diferenças Múltiplas**

Cria um vetor e calcula a diferença de segunda ordem entre os elementos.

X = [0 5 15 30 50 75 105];

Y = diff(X,2)

Y = *1×5*

5 5 5 5 5

**Diferenças Entre Colunas de uma Matriz**

Cria uma matriz 3x3 e calcula a diferença de primeira ordem entre as colunas.

X = [1 3 5;7 11 13;17 19 23];

Y = diff(X,1,2)

Y = *3×2*

2 2  
 4 2  
 2 4

**Aproximação de Derivadas com diff**

Usa a função diff para aproximar derivadas parciais com a sintaxe Y = diff(f)/h, onde f é um vetor de valores da função avaliados sobre algum domínio X, e h é um tamanho de passo apropriado.

Por exemplo, a primeira derivada de sin(x) em relação a x é cos(x), e a segunda derivada em relação a x é -sin(x). Podes usar diff para aproximar estas derivadas.

h = 0.001; % passo

X = -pi:h:pi; % domínio

f = sin(X); % função

Y = diff(f)/h; % 1ª derivada

Z = diff(Y)/h; % 2ª derivada

plot(X(:,1:length(Y)),Y,'r',X,f,'b', X(:,1:length(Z)),Z,'k')

Uma imagem com file, Gráfico, diagrama, texto

Descrição gerada automaticamente

Figura 1 - Aproximação de Derivadas com diff

Neste gráfico, a linha azul corresponde à função original, sin. A linha vermelha corresponde à primeira derivada calculada, cos, e a linha preta corresponde à segunda derivada calculada, -sin.

**Diferenças Entre Valores de Data e Hora**

Cria uma sequência de valores de data e hora igualmente espaçados e encontra as diferenças de tempo entre eles.

t1 = datetime('now');

t2 = t1 + minutes(5);

t = t1:minutes(1.5):t2

t = *1×4 datetime*

27-May-2024 11:11:1127-May-2024 11:12:4127-May-2024 11:14:1127-May-2024 11:15:41

dt = diff(t)

dt = *1×3 duration*

00:01:30 00:01:30 00:01:30

### Função *int ()*

**Integral Indefinida de Expressão Univariada**

**F = int(expr)** calcula a integral indefinida de expr. int usa a variável de integração padrão determinada por symvar(expr,1). Se expr for uma constante, então a variável de integração padrão é x.

syms x

expr = -2\*x/(1+x^2)^2;

F = int(expr)

F =

Uma imagem com preto, Tipo de letra, captura de ecrã, design

Descrição gerada automaticamente

**Integrais Indefinidas de Função Multivariada**

**F = int(expr,var)** calcula a integral indefinida de expr em relação à variável escalar simbólica var.

% Define uma função multivariada com as variáveis x e z

syms f(x,z)

f(x,z) = x/(1+z^2);

% Encontra as integrais indefinidas da expressão multivariada em relação às variáveis x e z

Fx = int(f,x)

Fx(x, z) =

Uma imagem com relógio, Tipo de letra, preto, captura de ecrã

Descrição gerada automaticamente

Fz = int(f,z)

Fz(x, z) = 

% Se for especificada a variável de integração, então a função int utiliza a primeira variável retornada por symvar como a variável de integração

Fz(x, z) = atan(z)

Fz(x, z) = 

var = symvar(f,1)

var =

*x*

F = int(f)

F(x, z) =

Uma imagem com relógio, Tipo de letra, preto, captura de ecrã

Descrição gerada automaticamente

**Integrais Definidas de Expressões Simbólicas**

**F = int(expr,a,b)** calcula a integral definida de expr de a para b. int usa a variável de integração padrão determinada por symvar(expr,1). Se expr for uma constante, então a variável de integração padrão é x.

**int(expr,[a b])** é equivalente a **int(expr,a,b)**

% Integra uma expressão simbólica de 0 a 1

syms x

expr = x\*log(1+x);

F = int(expr,[0 1])

F =



% Integra outra expressão de sin(t) a 1

syms t

F = int(2\*x,[sin(t) 1])

F = 

% Quando int não consegue calcular o valor de uma integral definida, aproxima numericamente a integral usando vpa.

syms x

f = cos(x)/sqrt(1 + x^2);

Fint = int(f,x,[0 10])

Fint =

Uma imagem com Tipo de letra, preto, escrita à mão, design

Descrição gerada automaticamente

Fvpa = vpa(Fint)

Fvpa =

0.37570628299079723478493405557162

% Para aproximar integrais diretamente, utiliza vpaintegral em vez de vpa. A função vpaintegral é mais rápida e fornece controlo sobre as tolerâncias de integração.

Fvpaint = vpaintegral(f,x,[0 10])

Fvpaint =

0.375706

**Integrais dos Elementos de uma Matriz**

**F = int(expr,var,a,b)** calcula a integral definida de expr em relação à variável escalar simbólica var de a para b.

**int(expr,var,[a b])** é equivalente a **int(expr,var,a,b)**

% Integra uma expressão simbólica de 0 a 1

syms x

expr = x\*log(1+x);

F = int(expr,[0 1])

F =



% Integra outra expressão de sin(t) a 1

syms t

F = int(2\*x,[sin(t) 1])

F = 

% Quando int não consegue calcular o valor de uma integral definida, aproxima numericamente a integral usando vpa

syms x

f = cos(x)/sqrt(1 + x^2);

Fint = int(f,x,[0 10])

Fint =

Uma imagem com Tipo de letra, preto, escrita à mão, design

Descrição gerada automaticamente

% Para aproximar integrais diretamente, utiliza vpaintegral em vez de vpa. A função vpaintegral é mais rápida e fornece controlo sobre as tolerâncias de integração

Fvpa = vpa(Fint)

Fvpa =

0.37570628299079723478493405557162

# Métodos Numéricos para Integração

Na Matemática, muitos algoritmos visam aproximar o valor de uma integral definida sem usar uma expressão analítica para a sua primitiva. Estes métodos geralmente seguem três fases:

1. Decomposição do domínio em subintervalos.
2. Integração aproximada da função em cada subintervalo.
3. Soma dos resultados numéricos obtidos.

A integração numérica é necessária quando não é possível encontrar uma primitiva explícita para uma função, quando a primitiva é demasiado complexa para ser avaliada ou quando não se dispõe de uma expressão analítica para a função, mas conhecem-se os seus valores em pontos específicos do domínio. Durante a Integração Numérica de uma função num intervalo , a área sob a curva é calculada utilizando interpolação polinomial para aproximar a função e obter um polinómio . Este método básico é conhecido como quadratura numérica:

Existem vários métodos numéricos para a integração, sendo a **Regra dos Trapézios** e a **Regra de Simpson** os mais destacados e utilizados.

* **Regra dos Trapézios:** Aproxima a função por um polinómio de ordem 1 (reta), resultando num valor aproximado para a área de uma determinada região limitada com a forma de um trapézio.
* **Regra de Simpson:** Oferece uma melhor aproximação ao considerar um polinómio de 2º grau (parábola) em vez de um polinómio de 1º grau. Este método é especialmente eficaz quando o intervalo de integração é pequeno. A Regra de Simpson composta divide o intervalo de integração em subintervalos menores e aplica a fórmula de Simpson a cada um desses subintervalos, somando os resultados para obter uma aproximação mais precisa.

## Regra dos Trapézios Simples

**Fórmula:**

Onde:

* Representa a aproximação da integral da função no intervalo usando a Regra dos Trapézios Simples;
* É o valor da função f no ponto , ou seja, o valor da função no extremo esquerdo do intervalo de integração;
* É o valor da função f no ponto , ou seja, o valor da função no extremo direito do intervalo de integração;
* Representa a largura do intervalo de integração. É a diferença entre os limites superior e inferior do intervalo ;
* Média dos valores da função nos pontos e . Esta média é utilizada para calcular a altura do trapézio na aproximação da área sob a curva.
* Multiplicado pela média , calcula a área do trapézio que aproxima a integral da função no intervalo.

**Fórmula do erro para a regra dos trapézios simples:**

, com

Onde:

* Erro da aproximação da integral da função utilizando a Regra do Trapézio Simples;
* Indica que o erro é proporcional ao comprimento do intervalo elevado ao cubo. Quanto maior o intervalo, maior será o erro, assumindo que a segunda derivada não varia significativamente.
* A segunda derivada dá uma medida de quanto a função está curvada. O erro da aproximação aumenta com a magnitude da segunda derivada, pois funções mais curvas são mais difíceis de aproximar com segmentos de linha reta (o que a Regra do Trapézio faz).

**Algoritmo:**

1. Calcular os valores da função nos extremos do intervalo:
2. Calcular a largura do intervalo:
3. Aplicar a fórmula da Regra do Trapézio:

## Regra dos Trapézios Composta

**Fórmula:**

Onde:

* **e**  Os limites de integração inferior e superior, respectivamente;
* A função que está a ser integrada;
* O elemento de variação infinitesimal na variável de integração;
* O tamanho do intervalo , onde *n* é o número de subintervalos;
* Os pontos de integração dentro do intervalo .
* Os valores da função avaliados nos pontos de integração.

**Fórmula do |erro| para a regra dos trapézios:**

, com

Onde:

* O erro absoluto na aproximação da integral usando a regra dos trapézios;
* O valor absoluto do erro;
* **e**  Os limites de integração inferior e superior, respectivamente ℎ;
* O tamanho do intervalo de integração;
* O valor máximo da segunda derivada da função no intervalo

**Algoritmo:**

1. Ler ***n***
2. Ler ***(a,b)***
3. Para ***i*** de ***1*** até fazer
4. Escrever ***T***

**Função em MatLab:**

function Tsol = RTrapezios(~,f,a,b,h)

% Esta linha cria um vetor t com valores espaçados uniformemente de a a b com um passo

t = a:h:b;

% Inicializa a variável s como zero, que será usada para acumular a soma dos valores da função f(x)

s = 0;

% Calcula o número de elementos no vetor t, que é o número total de subintervalos

n = length(t);

% Inicia um loop que percorre todos os subintervalos, exceto o último

for i=1:n-1

% Para cada subintervalo, calcula o valor da função f(x) no ponto médio do subintervalo e adiciona-o à soma acumulada s

s = s + f(t(i));

end

% Calcula a integral aproximada usando a regra dos trapézios, onde Tsol é a solução aproximada. Esta solução é obtida multiplicando o tamanho do subintervalo h pela média ponderada dos valores da função nos extremos dos intervalos e nos pontos médios, conforme prescrito pela regra dos trapézios

Tsol = h/2\*(f(a)+2\*s+f(b));

end

## Regra de Simpson Simples

**Fórmula:**

Onde:

* Representa a aproximação da integral da função no intervalo usando a Regra de Simpson Simples;
* Integral da função no intervalo;
* São os limites de integração, onde é o limite inferior e é o limite superior;
* É o valor da função no ponto ;
* É o valor da função no ponto médio do intervalo ;
* É o valor da função f no ponto .
* É um fator de normalização que ajusta a soma ponderada dos valores da função para aproximar a integral.

### Fórmula do erro para a Regra de Simpson Simples:

, com

Onde:

* Erro da aproximação ao usar a Regra de Simpson Simples para calcular a integral de uma função no intervalo ;
* largura do intervalo de integração. Representa a diferença entre o limite superior e o limite inferior ;
* Representa a quarta derivada da função , avaliada no ponto dentro do intervalo de integração . Esta parte da fórmula captura a variação da função e sua curvatura dentro do intervalo.

**Algoritmo:**

1. **Entrada:**
   * Função a ser integrada;
   * Limites de integração e ;
2. **Cálculos:**
   * Calcular o ponto médio do intervalo: ;
   * Avaliar a função nos pontos médio, , e : , , .
3. **Aplicar a Fórmula da Regra de Simpson:**

## Regra de Simpson Composta

**Fórmula:**

Onde:

* **e**  Os limites de integração inferior e superior, respectivamente;
* A função que está a ser integrada;
* O elemento de variação infinitesimal na variável de integração;
* Representa a largura de cada subintervalo usado na integração. O valor de é o tamanho do passo entre os pontos de integração e 1/3 é um fator constante na regra de Simpson;
* Valor da função na abcissa ;
* Valor da função na abcissa ;
* Valor da função na abcissa ;
* Valor da função na abcissa ;
* Número de subintervalos.

**Fórmula do |erro| para a regra dos trapézios:**

, com

Onde:

* Módulo do erro absoluto na aproximação da integral usando a regra de Simpson;
* **e**  Os limites de integração inferior e superior, respectivamente ℎ;
* O tamanho do intervalo de integração;
* Constante que representa o valor máximo da quarta derivada de no intervalo de integração
* Quarta potência do tamanho do subintervalo;
* Fator de ponderação.

**Algoritmo:**

1. Ler ***n***
2. Ler ***(a,b)***
3. Para ***i*** de ***1*** até fazer
4. Se ***i*** par

Então

Senão

1. Escrever ***S***

**Função em MatLab:**

function out\_S = RSimpson(~,f,a,b,h)

% Esta linha cria um vetor t com valores espaçados uniformemente de a a b com um passo

t = a:h:b;

% Inicializa a variável s como zero, que será usada para acumular a soma dos valores da função f(x)

s = 0;

% Calcula o número de elementos no vetor t, que é o número total de subintervalos

n = length(t);

% Inicia um loop que percorre todos os subintervalos, exceto o último

for i=1:n-1

% Se o resto da divisão por 2 for 0, então i é par

% Verifica se i é par usando a função mod(i, 2), que retorna o resto da divisão de i por 2. O operador ~ nega o resultado, portanto, esta condição é verdadeira se i for par.

if ~(mod(i, 2))

% Se i for par, adiciona duas vezes o valor de 𝑓(𝑥𝑖)f(xi​) à soma s. Isso corresponde à regra de Simpson para pontos de índices pares.

s = s + 2\*f(t(i));

% Se i não for par, ou seja, se for ímpar:

Else

% Adiciona quatro vezes o valor de f(xi)à soma s. Isso também corresponde à regra de Simpson, mas para pontos de índices ímpares.

s = s + 4\*f(t(i));

end

end

% Calcula a aproximação da integral definida usando a regra de Simpson. É uma soma ponderada dos valores da função nos extremos e nos pontos intermediários. A constante h/3 é um fator de ponderação da regra de Simpson. O resultado é armazenado em out\_S e é retornado como saída da função

out\_S = h/3\*(f(a)+s+f(b));

end

end

# Função Harmónica

Uma função harmónica é uma função que satisfaz a equação de Laplace, ou seja, a soma das segundas derivadas parciais da função em relação a todas as suas variáveis é zero. Estas funções são comuns em problemas de física e engenharia, como na teoria do potencial, eletrostática e mecânica dos fluidos, descrevendo fenómenos como o campo elétrico e o fluxo de calor.

**Equação de Laplace:**

Onde:

Operador Laplaciano

Função

Pela definição, sabemos que é necessário mostrar que a soma das suas segundas derivadas parciais é igual a zero, ou seja:

Usando a equação como exemplo:

**1ºPasso: Calcular a primeira derivada parcial de :**

**2ºPasso: Calcular a segunda derivada parcial de :**

**3ºPasso: Somar as segundas derivadas parciais**

**4ºPasso: Conclusão**

É uma função harmónica.

# Exemplos de aplicação e teste dos métodos

Para ilustrar a aplicação de cada método de diferenciação e integração vamos usar um problema fictício de velocidade ao longo do tempo.

Vamos escolher uma função de variável real e uma função de duas variáveis reais . Em seguida, aplicaremos os métodos de diferenciação e integração solicitados.

## Função de Variável Real e Intervalo

### Derivação

#### Diferenças Finitas Progressivas (2 pontos) para 𝑓(𝑥)

Uma imagem com texto, captura de ecrã, diagrama, número

Descrição gerada automaticamente

Figura 2 - Diferenças Finitas Progressivas (2 pontos)

#### Diferenças Finitas Regressivas (2 pontos) para 𝑓(𝑥)

Uma imagem com texto, captura de ecrã, número, diagrama

Descrição gerada automaticamente

Figura 3 - Diferenças Finitas Regressivas (2 pontos)

#### Diferenças Finitas Progressivas (3 pontos) para 𝑓(𝑥)

Uma imagem com texto, captura de ecrã, número, diagrama

Descrição gerada automaticamente

Figura 4 - Diferenças Finitas Progressivas (3 pontos)

#### Diferenças Finitas Regressivas (3 pontos) para 𝑓(𝑥)

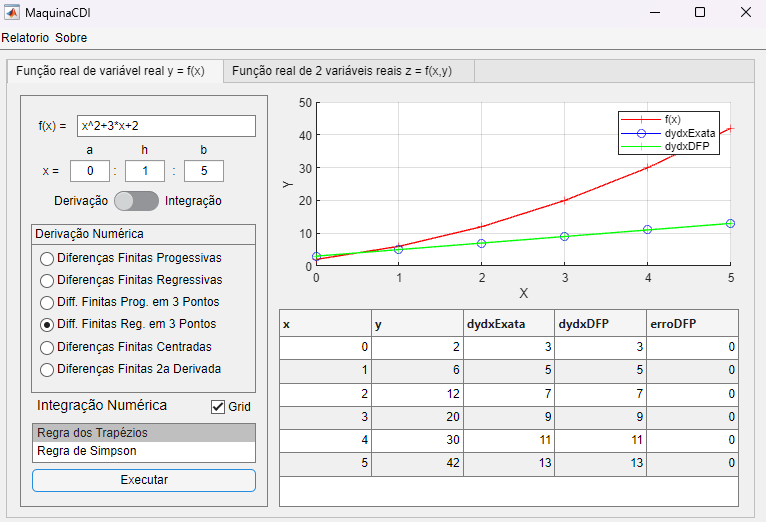


Figura 5 - Diferenças Finitas Regressivas (3 pontos)

#### Uma imagem com texto, captura de ecrã, número, Paralelo Descrição gerada automaticamenteDiferenças Finitas Centradas (3 pontos) para 𝑓(𝑥)

Figura 6 - Diferenças Finitas Centradas (3 pontos)

#### Diferenças Finitas 2ªDerivada para 𝑓(𝑥)

Uma imagem com texto, captura de ecrã, número, Paralelo

Descrição gerada automaticamente

Figura 7 - Diferenças Finitas 2ª Derivada

### Integração

#### Regra dos Trapézios Composta

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Figura 8 - Regra dos Trapézios Composta

#### Regra Simpson Composta

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

Figura 9 – Regra Simpson Composta

## Uma imagem com texto, diagrama, captura de ecrã, Gráfico Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, diagrama, Gráfico Descrição gerada automaticamenteFunção de Duas Variáveis Reais e Intervalos

Figura 10 - Função de Duas Variáveis Reais Não Harmónica

com passo

com passo

com passo

com passo

Uma imagem com texto, diagrama, captura de ecrã, design

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, diagrama, design

Descrição gerada automaticamente

Figura 11 - Função de Duas Variáveis Reais Harmónica

# Conclusão

Os métodos numéricos têm uma vasta gama de aplicações, tanto para derivadas como para integrais, o que facilita a resolução de problemas complexos em várias áreas da ciência e além. Muitas vezes, o cálculo de integrais de forma analítica não é possível, tornando os métodos numéricos indispensáveis.

No contexto da derivação numérica, as fórmulas que utilizam 3 pontos fornecem uma melhor aproximação do valor real em comparação com aquelas que utilizam apenas 2 pontos, que frequentemente apresentam o dobro do erro ou mais.

Para integrais, a Regra de Simpson é superior à Regra dos Trapézios para valores elevados de e grandes intervalos (ou seja, pequeno ). No entanto, para valores pequenos de e intervalos reduzidos (grande ), a diferença é menos perceptível, e por vezes a Regra dos Trapézios pode ter erros menores. Com suficientemente grande, a Regra de Simpson tende a ter erros menores.

Este trabalho prático demonstra a importância das equações diferenciais ordinárias na modelação de problemas científicos e o papel crucial dos métodos numéricos na sua resolução. Os computadores são ferramentas indispensáveis no estudo destas equações, permitindo a execução de algoritmos para obter aproximações numéricas das soluções.

# Bibliografia

* Correia, A. (2024, 29 de abril). *Matlab .: Atividade05Trabalho - MáquinaCDI*. Moodle.<https://pt.wikipedia.org/wiki/Diferencia%C3%A7%C3%A3o_num%C3%A9rica>
* Contribuidores dos projetos da Wikipedia. (2004, 9 de março). Método das diferenças finitas. Wikipédia, a enciclopédia livre. [https://pt.wikipedia.org/wiki/Método\_das\_diferenças\_finitas](https://pt.wikipedia.org/wiki/M%C3%A9todo_das_diferen%C3%A7as_finitas)
* *Diferenças finitas*. (s.d.). Inicial — UFRGS | Universidade Federal do Rio Grande do Sul. <https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/dn-diferencas_finitas.html>
* *Diferenciação numérica*. (2017, 5 de setembro). Métodos Matemáticos e Computacionais. <https://metmatcom.blogspot.com/2017/09/diferenciacao-numerica.html>
* *Differences and approximate derivatives - MATLAB diff*. (s.d.). MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. <https://www.mathworks.com/help/matlab/ref/diff.html>
* *Definite and indefinite integrals - MATLAB int*. (s.d.). MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. <https://www.mathworks.com/help/symbolic/sym.int.html>
* Contribuidores dos projetos da Wikipedia. (2008a, 19 de fevereiro). *Função harmônica*. Wikipédia, a enciclopédia livre. [https://pt.wikipedia.org/wiki/Função\_harmônica](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_harm%C3%B4nica)
* Contribuidores dos projetos da Wikipedia. (2007, 17 de julho). *Integração numérica*. Wikipédia, a enciclopédia livre. [https://pt.wikipedia.org/wiki/Integração\_numérica](https://pt.wikipedia.org/wiki/Integra%C3%A7%C3%A3o_num%C3%A9rica)
* *Integração Numérica*. (s.d.). Department of Mathematics — Técnico, Lisboa. <https://www.math.tecnico.ulisboa.pt/~calves/courses/integra/>
* Contribuidores dos projetos da Wikipedia. (2008b, 15 de março). *Diferenciação numérica – Wikipédia, a enciclopédia livre*. Wikipédia, a enciclopédia livre. [https://pt.wikipedia.org/wiki/Diferenciação\_numérica](https://pt.wikipedia.org/wiki/Diferencia%C3%A7%C3%A3o_num%C3%A9rica)
* Correia, A. (s.d.). *Cap5\_IntegracaoNumerica\_rascunho*.

# Autoavaliação e heteroavaliação do trabalho submetido

Tendo em conta o que foi feito ao longo do trabalho e que o mesmo vale 5 valores, concluímos assim as seguintes auto e hétero avaliações:

**Autoavaliação:**

Ana Rita Conceição Pessoa – 4 valores

João Francisco de Matos Claro – 5 valores

**Heteroavaliação:**

Ana Rita Conceição Pessoa – 4 valores

João Francisco de Matos Claro – 5 valores

Text

Description automatically generated with medium confidence