A picture containing logo

Description automatically generated

**Análise Matemática II**

**2021/2022**

Atividade 03 – Métodos Numéricos para Derivação e Integração

Pedro Emanuel Dinis Serrano nº2016017926

Inês Filipa dos Reis Moreira nº 2019120254

Índice

[1. Introdução 3](#_Toc107163323)

[2. Métodos Numéricos para Derivação 4](#_Toc107163324)

[2.1 Fórmulas de Diferenças Finitas 4](#_Toc107163325)

[2.1.1 Progressivas 4](#_Toc107163326)

[2.1.2 Regressivas 6](#_Toc107163327)

[2.2.1 Progressivas 8](#_Toc107163328)

[2.2.2 Regressivas 10](#_Toc107163329)

[2.2.3 Centradas 12](#_Toc107163330)

[2.3 2ª Derivada 14](#_Toc107163331)

[4. Métodos Numéricos para Integração 16](#_Toc107163332)

[4.1 Regra dos Trapézios 16](#_Toc107163333)

[4.2 Regra de Simpson 18](#_Toc107163334)

[5. Conclusão 20](#_Toc107163335)

[6. Bibliografia 21](#_Toc107163336)

# Introdução

Este trabalho consiste na implementação em MATLAB de métodos de Derivação e Integração Numérica.

O principal objetivo é a implementação de funções, através do desenvolvimento de uma app de MATLAB para algumas fórmulas de Derivação e Integração Numérica, nomeadamente: diferenças finitas em 2 e 3 pontos (Progressivas, Regressivas e Centradas), 2ª derivada e também regra dos Trapézios e regra de Simpson.

# Métodos Numéricos para Derivação

## 2.1 Fórmulas de Diferenças Finitas

### 2.1.1 Progressivas

#### Fórmula:

![Uma imagem com relógio

Descrição gerada automaticamente]()

onde:

* *f’(xk)*🡪 Aproximação do valor da derivada no ponto de abcissa ***xk***;
* *f(xk+1)*🡪 Valor da função na próxima abcissa;
* *f(xk)*🡪 Valor da função no ponto de abcissa atual;
* *h* 🡪 Valor de cada sub-intervalo (passo).

#### Algoritmo:

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

#### Função (MATLAB):

Text, letter

Description automatically generated



### 2.1.2 Regressivas

#### Fórmula:



onde:

* *f’(xk)*🡪 Aproximação do valor da derivada no ponto de abcissa ***xk***;
* *f(xk)*🡪 Valor da função no ponto de abcissa atual;
* *f(xk-1)*🡪 Valor da função na abcissa anterior;
* *h* 🡪 Valor de cada sub-intervalo (passo).

#### Algoritmo:

1. Alocar memória para ***x***;
2. Definir o número de pontos (***n***);
3. Se forem inseridos 5 elementos, ***y*** recebe o valor de ***f(x)***;
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 f no ponto atual, para a iésima iteração.

#### Função (MATLAB):

Text, letter

Description automatically generated

2.2 Fórmulas de Diferenças Finitas em 3 pontos

### 2.2.1 Progressivas

#### Fórmula:



onde:

* *f’(xk)*🡪 Aproximação do valor da derivada no ponto de abcissa ***xk***;
* *f(xk)*🡪 Valor da função no ponto de abcissa atual;
* *f(xk+1)*🡪 Valor da função na próxima abcissa;
* *f(xk+2)*🡪Valor da função 2 abcissas à frente;
* *h* 🡪 Valor de cada subintervalo (passo).

#### Algoritmo:

1. Alocar memória para ***x***;
2. Definir o número de pontos (***n***);
3. Se forem inseridos 5 elementos, ***y*** recebe o valor de ***f(x)***;
4. Alocar memória para a derivada;
5. Para ***i*** de ***1*** a ***n-2***, calcular a derivada (aproximada) de f 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):

Graphical user interface, text, application, letter, email

Description automatically generated

### 2.2.2 Regressivas

#### Fórmula:

![Uma imagem com faca

Descrição gerada automaticamente]()

onde:

* *f’(xk)*🡪 Aproximação do valor da derivada no ponto de abcissa ***xk***;
* *f(xk-2)*🡪 Valor da função 2 abcissas atrás;
* *f(xk-1)*🡪 Valor da função na abcissa anterior;
* *f(xk)*🡪 Valor da função no ponto de abcissa atual;
* *h* 🡪 Valor de cada subintervalo (passo).

#### Algoritmo:

1. Alocar memória para ***x***;
2. Definir o número de pontos (***n***);
3. Se forem inseridos 5 elementos, ***y*** recebe o valor de ***f(x)***;
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 f no ponto atual, para a iésima iteração.

#### Função (MATLAB):

Text, letter

Description automatically generated

### 2.2.3 Centradas

#### Fórmula:



onde:

* *f’(xk)*🡪 Aproximação do valor da derivada no ponto de abcissa ***xk***;
* *f(xk+1)*🡪 Valor da função na próxima abcissa;
* *f(xk-1)*🡪 Valor da função na abcissa anterior;
* *h* 🡪 Valor de cada subintervalo (passo).

#### Algoritmo:

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

#### Função (MATLAB):

Graphical user interface, text, application

Description automatically generated

### 2.3 2ª Derivada

#### Fórmula:



onde:

* *f’’(xk)*🡪 Aproximação do valor da 2ª derivada no ponto de abcissa ***xk***;
* *f(xk+1)*🡪 Valor da função na próxima abcissa;
* *f(xk)*🡪 Valor da função no ponto de abcissa atual;
* *f(xk-1)*🡪 Valor da função na abcissa anterior;
* *h* 🡪 Valor de cada subintervalo (passo).

#### Algoritmo:

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

#### Função (MATLAB):

Graphical user interface, text

Description automatically generated

## 4. Métodos Numéricos para Integração

## 4.1 Regra dos Trapézios

A Regra dos Trapézios é um método de Integração Numérica utilizado para aproximar o integral definido e pode também ser visto como o resultado da média da parte esquerda e direita da soma de Riemann, e por vezes pode mesmo ser definido desta forma.

#### Fórmula:



onde:

* *IT (f)* 🡪 Cálculo da Regra dos Trapézios;
* *f(x0)*🡪 Valor da função na abcissa ***x0***;
* *f(x1)*🡪 Valor da função na abcissa ***x1***;
* *f(xn-1)*🡪 Valor da função na abcisa ***xn-1***;
* *f(xn)*🡪 Valor da função na abcissa ***xn***;
* *h* 🡪 Valor de cada subintervalo (passo).

#### Algoritmo:

1. Calcular ***h***;
2. Calcular x ;
3. Inicializar ***s*** com o valor ***0***;
4. Para ***i*** de ***1*** a ***n-1***:
   1. Somar o valor da função em x (***f(x)***) a ***s***.
5. Cálculo da Regra dos Trapézios.

#### Função (MATLAB):

INPUT:

* f – função integranda
* [a, b] - intervalo de derivação
* n - número de subintervalos

OUTPUT:

* area - Valor da área calculada pela Regra dos Trapézios

Text, letter

Description automatically generated

## 4.2 Regra de Simpson

A Regra de Simpson é um método de Integração Numérica utilizado para a aproximação de integrais definidos e baseia-se em aproximar o integral definido pela área sob arcos de parábola que interpolam a função.

Para conseguir uma melhor aproximação deve-se dividir o intervalo de integração em intervalos mais pequenos, aplicar a fórmula de Simpson para cada um deles e somar os resultados.

#### Fórmula:



onde:

* *IS (f)* 🡪 Cálculo da Regra de Simpson;
* *h* 🡪 Valor de cada subintervalo (passo);
* *f(x0)*🡪 Valor da função na abcissa ***x0***;
* *f(x1)*🡪 Valor da função na abcissa ***x1***;
* *f(xn-1)*🡪 Valor da função na abcissa ***xn-1***;
* *f(xn)*🡪 Valor da função na abcissa ***xn***;
* *n* 🡪 Número de subintervalos.

#### Algoritmo:

1. Cálculo do passo (***h***);
2. Calcular ***x***;
3. Inicializar ***s*** com o valor ***0***;
4. Para ***i*** de ***1*** a ***n-1***:
   1. Se ***i*** for par, soma-se **2** vezes o valor de ***f(x)*** a ***s***;
   2. Se ***i*** for ímpar, soma-se 4 vezes o valor de ***f(x)*** a ***s***;
5. Calcular a Regra de Simpson.

#### Função (MATLAB):

INPUT:

* f – função integranda
* [a, b] - intervalo de derivação
* n - número de subintervalos

OUTPUT:

* area - Valor da área calculada pela Regra de Simpson

Text, letter

Description automatically generated

# 5. Conclusão

Com este trabalho podemos concluir que os métodos numéricos têm diversas aplicações, quer para derivadas, quer para integrais, o que facilita a resolução de problemas de várias áreas da ciência onde necessitamos de calcular integrais.

Como já visto anteriormente, verificamos que quanto maior for o número de subintervalos ***n***, menor é o erro dos métodos / fórmulas. A introdução do tamanho de cada subintervalo ***h*** tem o efeito contrário: quanto menor o tamanho introduzido, menor o erro dos métodos / fórmulas.

Para valores elevados de ***n*** e grandes intervalos, a Regra de Simpson é melhor do que a Regra dos Trapézios, mas para valores pequenos de ***n*** e pequenos intervalos não se nota tanto essa diferença, sendo que por vezes a Regra dos Trapézios obtém erros menores.

Em relação às fórmulas da derivação numérica, a comparação encontrada foi que a melhor aproximação ao valor real se origina a partir das fórmulas que utilizam 3 pontos, comparativamente às que apenas utilizam 2 pontos.

# 6. Bibliografia

* <https://en.wikipedia.org/wiki/Simpson%27s_rule>
* <https://en.wikipedia.org/wiki/Trapezoidal_rule>
* <https://pt.wikipedia.org/wiki/Diferencia%C3%A7%C3%A3o_num%C3%A9rica>
* <https://pt.wikipedia.org/wiki/Integra%C3%A7%C3%A3o_num%C3%A9rica>
* <https://www.ufsj.edu.br/portal2-repositorio/File/nepomuceno/mn/09MN_Derivacao.pdf>