Ano Letivo: 2022/2023

Atividade 05- Métodos Numéricos para Derivação e Integração

Relatório

Análise Matemática II

Docente: Arménio Correia

Trabalho realizado por:

Martim Alexandre Vieira Antunes

nº: 2022141890 **Curso:** LEI

Pedro Lino Neves Faneca

nº: 2022134142 **Curso:** LEI

Índice

1.Introdução

Este trabalho surge do âmbito da unidade curricular de Análise Matemática 2, do curso de Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

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

2.Métodos Numéricos para Derivação

Existem vários métodos numéricos para a derivação, que são utilizados para calcular aproximações das derivadas de funções. Esses métodos são úteis quando não é possível calcular a derivada analiticamente ou quando se deseja obter uma estimativa numérica precisa da derivada.

O método mais simples para aproximar a derivada é usar o método de diferenças finitas. O método das diferenças finitas é um método de resolução de equações diferenciais que se baseia na aproximação de derivadas por diferenças finitas.

2.1- Diferenças Finitas em 2 pontos

As diferenças finitas em 2 pontos são um conjunto de métodos para estimar derivadas de uma função utilizando os valores da função em dois pontos consecutivos.

2.1.1- Progressivas

Este método utiliza a diferença entre os valores da função em dois pontos consecutivos para estimar a derivada.

Fórmula:

$$f'(x_k) := \frac{f(x_{k+1}) - f(x_k)}{h}$$

Legenda:

- $f'(x_k)$ = Aproximação do valor da derivada no ponto de abcissa x_k;
- $f(x_k+1)$ = Valor da função na próxima abcissa;
- $f(x_k)$ = Valor da função no ponto de abcissa atual;
- h = Valor de cada subintervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function [x,y,dydx] = DFProgressivas_2(f,a,b,h,y)

x = a:h:b;                  % Alocação de memória
n = length(x);              % Número de pontos (tamanho do vetor de abcissas)

if nargin==4
    y = f(x);               % y é a função f(x)
end

dydx = zeros(1,n);           % Alocação de memória

for i = 1:n-1
    dydx(i) = (y(i+1)-y(i))/h;  % Derivada (aproximada) de f no ponto atual
end

dydx(n) = (y(n)-y(n-1))/h;      % Derivada (aproximada) de f no ponto atual
```

2.1.2- Regressivas

Este método também utiliza a diferença entre os valores da função em dois pontos consecutivos, mas agora os pontos são deslocados para trás em relação ao ponto de interesse.

Fórmula:

$$f'(x_k) := \frac{f(x_k) - f(x_{k-1})}{h}$$

Legenda:

- $f'(x_k)$ = Aproximação do valor da derivada no ponto de abcissa x_k;
- $f(x_k)$ = Valor da função no ponto de abcissa atual;
- $f(x_{k-1})$ = Valor da função na abcissa anterior;
- *h* = Valor de cada sub-intervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function [x,y,dydx] = DFRegressivas_2(f,a,b,h,y)

x = a:h:b;                  % Alocação de memória
n = length(x);              % Número de pontos (tamanho do vetor de abcissas)

if nargin==4
    y = f(x);                % y é a função f(x)
end

dydx = zeros(1,n);           % Alocação de memória

dydx(1) = (y(2)-y(1))/h;     % Derivada (aproximada) de f no ponto atual

for i = 2:n
    dydx(i) = (y(i)-y(i-1))/h;   % Derivada (aproximada) de f no ponto atual
end
```

2.2-Diferenças Finitas em 3 pontos

As diferenças finitas em 3 pontos são um conjunto de métodos para estimar derivadas de uma função utilizando os valores da função em três pontos consecutivos.

2.2.1-Progressivas

Fórmula:

$$f'(x_k) := \frac{-3f(x_k) + 4f(x_{k+1}) - f(x_{k+2})}{2h}$$

Legenda:

- $f'(x_k)$ = Aproximação do valor da derivada no ponto de abcissa $\boldsymbol{x_k}$;
- $f(x_k)$ = Valor da função no ponto de abcissa atual;
- $f(x_{k+1})$ = Valor da função na próxima abcissa;
- $f(x_{k+2})$ =Valor da função 2 abcissas à frente;
- h = Valor de cada subintervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function [x,y,dydx] = DFProgressivas_3(f,a,b,h,y)

x = a:h:b;                % Alocação de memória
n = length(x);            % Número de pontos (tamanho do vetor de abcissas)

if nargin==4
    y = f(x);             % y é a função f(x)
end

dydx = zeros(1,n);        % Alocação de memória

for i = 1:n-2
    dydx(i) = ((-3)*y(i) + 4*y(i+1) - y(i+2))/(2*h);  % Derivada (aproximada) de f no ponto atual
end

dydx(n-1) = (y(n-3) - 4*y(n-2) + 3*y(n-1))/(2*h);  % Derivada (aproximada) de f no ponto atual
dydx(n) = (y(n-2) - 4*y(n-1) + 3*y(n))/(2*h);      % Derivada (aproximada) de f no ponto atual
```

2.2.2-Regressivas

Fórmula:

$$f'(x_k) := \frac{f(x_{k-2}) - 4f(x_{k-1}) + 3f(x_k)}{2h}$$

Legenda:

- $f'(x_k)$ = Aproximação do valor da derivada no ponto de abcissa x_k;
- $f(x_{k-2})$ = Valor da função 2 abcissas atrás;
- $f(x_{k-1})$ = Valor da função na abcissa anterior;
- $f(x_k)$ = Valor da função no ponto de abcissa atual;
- h = Valor de cada subintervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function [x,y,dydx] = DFRegressivas_3(f,a,b,h,y)

x=a:h:b;                  % Alocação de memória
n=length(x);              % Número de pontos (tamanho do vetor de abcissas)

if nargin==4
    y=f(x);               % y é a função f(x)
end

dydx=zeros(1,n);          % Alocação de memória

dydx(1)=((-3)*y(1) + 4*y(2) - y(3))/(2*h);     % Derivada (aproximada) de f no ponto atual
dydx(2)=((-3)*y(2) + 4*y(3) - y(4))/(2*h);     % Derivada (aproximada) de f no ponto atual

for i=3:n
    dydx(i)=(y(i-2)- 4*y(i-1) + 3*y(i))/(2*h);    % Derivada (aproximada) de f no ponto atual
end
```

2.2.3-Centradas

Este método utiliza a diferença simétrica entre os valores da função em pontos adjacentes para obter uma aproximação mais precisa da derivada.

Fórmula:

$$f'(x_k) := \frac{f(x_{k+1}) - f(x_{k-1})}{2h}$$

Legenda:

- *$f'(x_k)$* = Aproximação do valor da derivada no ponto de abcissa $\boldsymbol{x_k}$;
- *$f(x_{k+1})$* = Valor da função na próxima abcissa;
- *$f(x_{k-1})$* = Valor da função na abcissa anterior;
- *h* = Valor de cada subintervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function [x,y,dydx] = DFCentradas_3(f,a,b,h,y)

x = a:h:b;                    % Alocação de memória
n = length(x);                % Número de pontos (tamanho do vetor de abcissas)

if nargin==4
    y=f(x);                   % y é a função f(x)
end

dydx = zeros(1,n);            % Alocação de memória

dydx(1) = (y(2)-y(1))/h;      % Derivada (aproximada) de f no ponto atual

for i = 2:n-1
    dydx(i) = (y(i+1)-y(i-1))/(2*h);        % Derivada (aproximada) de f no ponto atual
end

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

2.3-2ªDerivada

Fórmula:

$$f''(x_k) := \frac{f(x_{k+1}) - 2f(x_k) + f(x_{k-1})}{h^2}$$

Legenda:

- $f''(x_k)$ = Aproximação do valor da 2ª derivada no ponto de abcissa x_k;
- $f(x_{k+1})$ = Valor da função na próxima abcissa;
- $f(x_k)$ = Valor da função no ponto de abcissa atual;
- $f(x_{k-1})$ = Valor da função na abcissa anterior;
- h = Valor de cada subintervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function [x,y,dydx] = DFDerivada2_3(f,a,b,h,y)

x=a:h:b;                    % Alocação de memória
n=length(x);                % Número de pontos (tamanho do vetor de abcissas)

if nargin==4
    y=f(x);                  % y é a função f(x)
end

dydx=zeros(1,n);             % Alocação de memória

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

for i=2:n-1
    dydx(i)=(y(i+1)-2*y(i)+y(i-1))/(h^2);
end
dydx(n)=(y(n)-2*y(n-1)+y(n-2))/(h^2);
```

3.Métodos Numéricos para Integração

Integração numérica é uma técnica utilizada para calcular uma aproximação numérica de uma integral definida de uma função. O integral definido é uma operação matemática que envolve o cálculo da área sob a curva de uma função em um intervalo específico.

A necessidade de técnicas de integração numérica surge quando não é possível encontrar uma solução analítica para a integral, ou quando a função é muito complexa para ser integrada de forma exata.

Existem vários métodos de integração numérica, cada um com suas próprias características e aplicabilidades. Alguns dos métodos mais comuns incluem:

- Regra do Trapézio: Este método divide o intervalo de integração em vários segmentos de igual comprimento e aproxima a área sob a curva em cada segmento por um trapézio. A soma das áreas dos trapézios fornece uma estimativa da integral.
- Regra de Simpson: Este método também divide o intervalo de integração em segmentos, mas em vez de utilizar trapézios, utiliza polinómios de grau 2 para aproximar a curva em cada segmento. A soma das áreas dos polinómios de grau 2 fornece uma estimativa mais precisa do integral.
- Quadratura de Gauss: Este método utiliza uma abordagem baseada em pontos de integração específicos e pesos correspondentes. Esses pontos e pesos são escolhidos de forma a obter uma estimativa precisa do integral para uma determinada classe de funções.
- Regra de Monte Carlo: Este método utiliza amostragem aleatória para estimar o integral. São gerados pontos aleatórios dentro do intervalo de integração, e a média dos valores da função nesses pontos é utilizada para estimar o integral.

Esses são apenas alguns exemplos de métodos de integração numérica. A escolha do método mais adequado depende da função a ser integrada, das características do problema e do nível de precisão desejado. A integração numérica desempenha um papel importante em várias áreas, como física, engenharia, economia, entre outras, onde a integração analítica pode ser difícil ou impossível de ser realizada.

$$\int_a^b f(x)\mathrm{d}x \approx$$

3.1-Regra dos Trapézios

A regra dos trapézios é um método numérico para calcular uma aproximação da integral definida de uma função. Ela divide o intervalo de integração em vários trapézios e estima a área sob a curva da função utilizando a soma das áreas desses trapézios. Quanto mais subintervalos existirem mais precisa se torna a aproximação.

Fórmula:

$$I_T(f) = \frac{h}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]$$

Legenda:

- $I_T(f)$ = Cálculo da Regra dos Trapézios;
- $f(x_0)$ = Valor da função na abcissa x_0;
- $f(x_1)$ = Valor da função na abcissa x_1;
- $f(x_{n-1})$ = Valor da função na abcisa x_{n-1};
- $f(x_n)$ = Valor da função na abcissa x_n;
- h = Valor de cada subintervalo (passo).

Algoritmo:

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

Função (MATLAB):

```
function T = RTrapezios(f,a,b,n)

h = (b-a)/n;           % Valor de cada subintervalo (passo)
x = a;                 % 'x' recebe o valor de 'a' (primeira abcissa)
s = 0;                 % Inicializacao da variavel 's' a 0

for i = 1:n-1
    x = x+h;           % Ao valor de 'x' e somado o passo ('h')
    s = s+f(x);        % Ao valor de 's' e somado 1 vez o valor da funcao
end

T = h/2*((f(a) +2*s + f(b)));
```

3.2-Regra de Simpson

A regra de Simpson é um método numérico para calcular uma aproximação do integral definido de uma função. Esta regra utiliza uma fórmula de interpolação polinomial para aproximar a curva da função por meio de segmentos de parábolas. O integral sob cada parábola é calculado e somado para obter a aproximação do integral total.

Fórmula:

$$I_S(f) = \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$$

Legenda:

- $I_S(f)$ → Cálculo da Regra de Simpson;
- h → Valor de cada subintervalo (passo);
- $f(x_0)$ → Valor da função na abcissa $\boldsymbol{x_0}$;
- $f(x_1)$ → Valor da função na abcissa $\boldsymbol{x_1}$;
- $f(x_{n-1})$ → Valor da função na abcissa $\boldsymbol{x_{n-1}}$;
- $f(x_n)$ → Valor da função na abcissa $\boldsymbol{x_n}$;
- n → Número de subintervalos.

Algoritmo:

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

Função (MATLAB)

```
function out_S = RSimpson(f,a,b,n)

h = (b-a)/n;                 % Valor de cada subintervalo (passo)
x=a;                         % 'x' recebe o valor de 'a' (primeira abcissa)
s = 0;                       % Inicializacao da variavel 's' a 0
for i = 1:n-1
    x = x+h;                 % Ao valor de 'x' e somado o passo ('h')
    if mod(i,2)==0           % Se i for par
        s = s+2*f(x);        % Ao valor de 's' e somado 2 vezes o valor da funcao
    else
        s = s+4*f(x);        % Ao valor de 's' e somado 2 vezes o valor da funcao
    end
end

out_S = h/3*((f(a)+s+f(b)));
```

4.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 mais complicados das diferentes áreas da matemática e ciências.

Como já visto anteriormente, verificamos que quanto maior for o número de subintervalos ***n***, menor é o erro dos métodos. Já com o tamanho de cada subintervalo, ***h*** , o efeito é o contrário: quanto menor o tamanho introduzido, menor o erro dos métodos.

Relativamente à comparação entre os vários métodos da integração numérica, verificamos que o que apresenta menor erro é, e consequentemente, melhor aproximação ao valor exato, é a **Regra de Simpson** comparando com a Regra dos Trapézios.

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.

5.Bibliografia

- Ficheiros de suporte disponibilizados pelo professor;
- Formulário da cadeira;

6.Autoavalição e heteroavaliação

Chegando ao fim do trabalho, estamos contentes pelo resultado final.

Pelo esforço e trabalho aplicados a esta atividade, achamos que merecemos numa escala de 0 a 5 valores, um 4.25.

A nível de grupo, não houve quaisquer problemas e ambos trabalhámos bem. O aluno Martim Antunes foi quem distribui as tarefas, explicou como secalhar ficava melhor e quem preocupava-se de sempre colocar um "dedinho" dele no final, mesmo não lhe tivesse sido atribuído aquela tarefa, aperfeiçoar ainda mais.

Por isso achamos que o aluno Martim Antunes mereça um 4.25 e o aluno Pedro Faneca um 3.75.