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

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.

2. Métodos Numéricos para Derivação

2.1 Fórmulas de Diferenças Finitas

2.1.1 Progressivas

Fórmula:

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

onde:

- $f'(x_k)$ → 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 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):

```
function [x,y,dydx] = NDerivadaDFP(~,f,a,b,h,y) %progressivas em 2 pontos
    x = a:h:b;
    n = length(x);
    if nargin == 5
        y = f(x);
    end
    dydx = zeros(1,n);
    for i = 1:n-1
        dydx(i) = (y(i+1)-y(i))/h;
    end
    dydx(n) = dydx(n-1);

end
```

2.1.2 Regressivas

Fórmula:

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

onde:

- $f'(x_k)$ → 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:

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):

```
function [x,y,dydx]=NDerivadaDFR(~,f,a,b,h,y) %regressivas
    x=a:h:b;
    n=length(x);

    if nargin==5
        y=f(x);
    end

    dydx=zeros(1,n);
    dydx(1)=(y(2)-y(1))/h;

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

2.2 Fórmulas de Diferenças Finitas em 3 pontos

2.2.1 Progressivas

Fórmula:

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

onde:

- $f'(x_k)$ → 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 próxima abcissa;
- $f(x_{k+2})$ →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):

```
function [x,y,dydx]=NDerivadaFPT(~,f,a,b,h,y) %progressiva 3 pontos
    x=a:h:b;
    n=length(x);

    if nargin==5
        y=f(x);
    end

    dydx=zeros(1,n);

    for i=1:n-2
        dydx(i)=(-3*y(i)+4*y(i+1)-y(i+2))/(2*h);
    end
    dydx(n-1)=(y(n-3)-4*y(n-2)+3*y(n-1))/(2*h);
    dydx(n)=(y(n-2)-4*y(n-1)+3*y(n))/(2*h);
end
```

2.2.2 Regressivas

Fórmula:

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

onde:

- $f'(x_k)$ → 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:

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):

```
function [x,y,dydx]=NDerivadaFRT(~,f,a,b,h,y) %regressiva 3 pontos
    x=a:h:b;
    n=length(x);

    if nargin==5
        y=f(x);
    end

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

    for i=3:n
        dydx(i)=(y(i-2)-4*y(i-1)+3*y(i))/(2*h);
    end
end
```

2.2.3 Centradas

Fórmula:

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

onde:

- $f'(x_k)$ → Aproximação do valor da derivada no ponto de abcissa x_k;
- $f(x_{k+1})$ → Valor da função na próxima abcissa;
- $f(x_{k-1})$ → 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):

```
function [x,y,dydx]=NDerivadaFC(~,f,a,b,h,y) %centrada em 3 pontos
    x=a:h:b;
    n=length(x);

    if nargin==5
        y=f(x);
    end

    dydx=zeros(1,n);

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

2.3 2ª Derivada

Fórmula:

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

onde:

- $f''(x_k)$ → 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:

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):

```
function [x,y,dydx]=NDerivadaF2D(~,f,a,b,h,y) %2 derivada
    x=a:h:b;
    n=length(x);

    if nargin==5
        y=f(x);
    end

    dydx=zeros(1,n);

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

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:

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

onde:

- $I_T(f)$ → 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:

1. Calcular ***h***;
2. Calcular x ;
3. Inicializar ***s*** com o valor ***0***;
4. Para ***i*** de ***1*** a ***n-1***:
   a. 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

```
function T = RTrapezios(~,f,a,b,n)
    h = (b-a)/n;
    x = a:h:b;
    s = 0;
    for i = 1:n-1
        s = s+f(x(i));
    end
    T = h/2*(f(a)+2*s+f(b));

end
```

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:

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

onde:

- $I_S(f)$ → Cálculo da Regra de Simpson;
- h → Valor de cada subintervalo (passo);
- $f(x_0)$ → 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 abcissa x_{n-1};
- $f(x_n)$ → Valor da função na abcissa x_n;
- 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***:
   a. Se ***i*** for par, soma-se **2** vezes o valor de ***f(x)*** a ***s***;
   b. 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

```
function S = RSimpson(~,f,a,b,n)
    h = (b-a)/n;
    x = a:h:b;
    s = 0;
    for i = 1:n-1
        if mod(i,2)==0
            s = s+2*f(x(i));
        else
            s = s+4*f(x(i));
        end
    end
    S = h/3*(f(a)+s+f(b));
end
```

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