Máquina de Cálculo Diferencial e Integral

Realizado por:

Pedro da Costa Sherring – 2020126540

Tiago José Gaspar Oliveira - 2022137225

Índice

Índice figuras

Introdução

Este trabalho refere-se à Atividade 5 da cadeira de Análise Matemática II, na qual é solicitado criar uma máquina capaz de utilizar métodos numéricos para realizar a integração e a derivação de funções, bem como verificar se uma função de duas variáveis reais é harmónica ou não.

Neste relatório, explicaremos os conceitos de métodos numéricos para integração e derivação. No primeiro ponto, abordaremos as fórmulas das diferenças finitas em 2 e 3 pontos, implementadas na nossa aplicação. No segundo ponto, discutiremos a regra dos trapézios e a regra de Simpson, também disponíveis na nossa aplicação.

1. Métodos Numéricos Para Derivação

Uma derivada permite calcular o declive da reta tangente ao um ponto x numa determinada função. Sendo o declive = m na equação reduzida da reta ($y = mx + c$).

Estas Fórmulas das Diferenças Finitas permitem calcular a derivada em situações onde não se sabe a função, mas apenas um conjunto de valores respetivos dessa função.

Se *f* for uma função definida em $[a, b]$, suficientemente regular e conhecida num conjunto de pontos de partição uniforme estilo $\mathrm{a} = \mathrm{x0} < \mathrm{x1} < \cdots < \mathrm{xn} = \mathrm{b}$, é possível calcular a sua derivada.

Todos os códigos destas funções começam por:

- Alocar memória para *x;*

- Definir o número de pontos de *n* com base no tamanho de *x*;

- Declaração de uma condição if para verificar se foram recebidos 4 argumentos para a função;

- Alocar memória para $dydx$, que vai ser a derivada.

1.1. Diferenças finitas em 2 pontos

Estas fórmulas permitem-nos aproximar da derivada usando a tal partição finita de pontos.

Progressivas

Fórmula

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

Figura 1 - Fórmula Progressiva das Diferenças Finitas em 2 pontos

Código MATLAB

```
function [x, y, dydx] = NDerivacaoP2(f, a, b, h, y)
x = a: h: b;
n = length(x);

if nargin == 4
    y = f(x);
end

dydx = zeros(1, n);

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

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

Figura 2 - Código da Progressiva das Diferenças Finitas em 2 pontos na nossa Aplicação

Regressivas

Fórmula

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

Figura 3 - Fórmula Regressiva das Diferenças Finitas em 2 pontos

Código MATLAB

```
function [x, y, dydx] = NDerivacaoR2(f, a, b, h, y)
x = a: h: b;
n = length(x);

if nargin == 4
    y = f(x);
end

dydx = zeros(1, n);

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

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

end
```

Figura 4 - Código da Regressiva das Diferenças Finitas em 2 pontos na nossa Aplicação

1.2. Diferenças finitas em 3 pontos

Progressivas

Fórmula

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

Figura 5 - Fórmula Progressiva das Diferenças Finitas em 3 pontos

Código MATLAB

```
function [x, y, dydx] = NDerivacaoP3(f, a, b, h, y)
x = a: h: b;
n = length(x);

if nargin == 4
    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) = (-3 * y(n - 2) + 4 * y(n - 1) - y(n)) / (2 * h);
dydx(n) = (-3 * y(n - 2) + 4 * y(n - 1) - y(n)) / (2 * h);
end
```

Figura 6 - Código da Progressiva das Diferenças Finitas em 3 pontos na nossa Aplicação

Regressivas

Fórmula

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

Figura 7 – Fórmula Regressiva das Diferenças Finitas em 3 pontos

Código MATLAB

```
function [x, y, dydx] = NDerivacaoR3(f, a, b, h, y)
x = a: h: b;
n = length(x);

if nargin == 4
    y = f(x);
end

dydx = zeros(1, n);

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

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

Figura 8 – Código da Regressiva das Diferenças Finitas em 3 pontos na nossa Aplicação

Centradas

Fórmula

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

Figura 9 – Fórmula Centrada das Diferenças Finitas em 3 pontos

Código MATLAB

```
function [x, y, dydx] = NDerivacaoC3(f, a, b, h, y)
x = a: h: b;
n = length(x);

if nargin == 4
    y = f(x);
end

dydx = zeros(1, n);

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

dydx(1) = (y(3) - y(1)) / (2 * h);
dydx(n) = (y(n) - y(n - 2)) / (2 * h);
end
```

Figura 10 – Código da Centrada das Diferenças Finitas em 3 pontos na nossa Aplicação

2º Derivada

Fórmula

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

Figura 11 - Fórmula da 2º Derivada das Diferenças Finitas em 3 pontos

Código MATLAB

```
function [x, y, dydx] = NDerivacaoD2(f, a, b, h, y)
x = a: h: b;
n = length(x);

if nargin == 4
    y = f(x);
end

dydx = zeros(1, n);

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

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

end
```

Figura 12 - Código da 2º Derivada das Diferenças Finitas em 3 pontos na nossa Aplicação

2. Métodos Numéricos para Integração

Tal como as fórmulas de derivação anteriores, existem algoritmos para nos aproximarmos do valor desejado no que toca à integração. Estes algoritmos recorrem à descomposição da integral em subintervalos $(x_a, x_1, x_2, \dots, x_{b)}$ no intervalo $[a, b]$ para realizarem a integração aproximada.

A integração numérica através destes algoritmos é especialmente útil nos casos em que as funções que pretendemos integrar são demasiado complexas, não admitem uma primitiva direta ou apenas se conhecem uma partição de valores de um intervalo da função.

2.1. Regras dos Trapézios

A Regra dos Trapézios calcula a integral de uma função num certo intervalo com base à interpolação polinomial, obtendo assim uma aproximação razoável através dos polinómios. Quantos mais intervalos em $[a, b]$, maior será a sua precisão.

Fórmula

Regra dos Trapézios

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

$$|E_{\mathrm{T}}| \leq \frac{b-a}{12} h^2 M_2, \quad M_2 = \max_{x \in [a,b]} |f''(x)|$$

Figura 13 - Fórmula da Regra dos Trapézios

Código MATLAB

Trapezios rule Algorithm

```
Input parameters: f, a, b e n
Output parameter: T

h := (b-a)/n;
x := a;
s := 0;
For i by 1 to n-1 do
   x := x + h;
   s := s + f(x);
End for
T := h/2(f(a)+2s+f(b))
```

Figura 14 - Código da Regra dos Trapézios fornecida no Moodle da UC

2.2. Regra de Simpson

Tal como a Regra dos Trapézios, calcula uma aproximação da integral de uma função f, mas desta vez, calcula-o através de um polinómio interpolador de 2ª grau, isto é, uma parábola. Como visto anteriormente, quantos mais intervalos em $[a, b]$, maior será a sua precisão.

Fórmula

Regra de Simpson

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

$$|E_{\mathrm{S}}| \leq \frac{b-a}{180} h^4 M_4, \qquad M_4 = \max_{x\in[a,b]} \left|f^{(4)}(x)\right|$$

Figura 15 - Fórmula da Regra de Simpson

Código MATLAB

```
Simpson's rule Algorithm
Input parameters: f, a, b e n
Output parameter: out_S

h := (b-a)/n;
x := a;
s := 0;
For i by 1 to n-1 do
   x := x + h;
   If i is even
   Then s := s + 2f(x);
   Else s := s + 4f(x);
End for
out_S :=h/3(f(a)+s+f(b))
```

Figura 16 - Código da Regra de Simpson fornecida no Moodle da UC

3. Função Harmónica

Uma Função Harmónica é definida usando a equação de Laplace e a função é harmónica é verdadeira se este for igual a zero.

Se temos uma função $f(x, y)$, podemos afirmar que ela é harmónica se $f''(x) + f''(y) = 0$, ou seja, se o Laplaciano for igual a zero.

$$\frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2} + \cdots + \frac{\partial^2 f}{\partial x_n^2} = 0$$

Figura 17 - Equação de Laplace

Aqui está um exemplo de um exercício para provar se uma função real de duas variáveis é harmónica ou não:

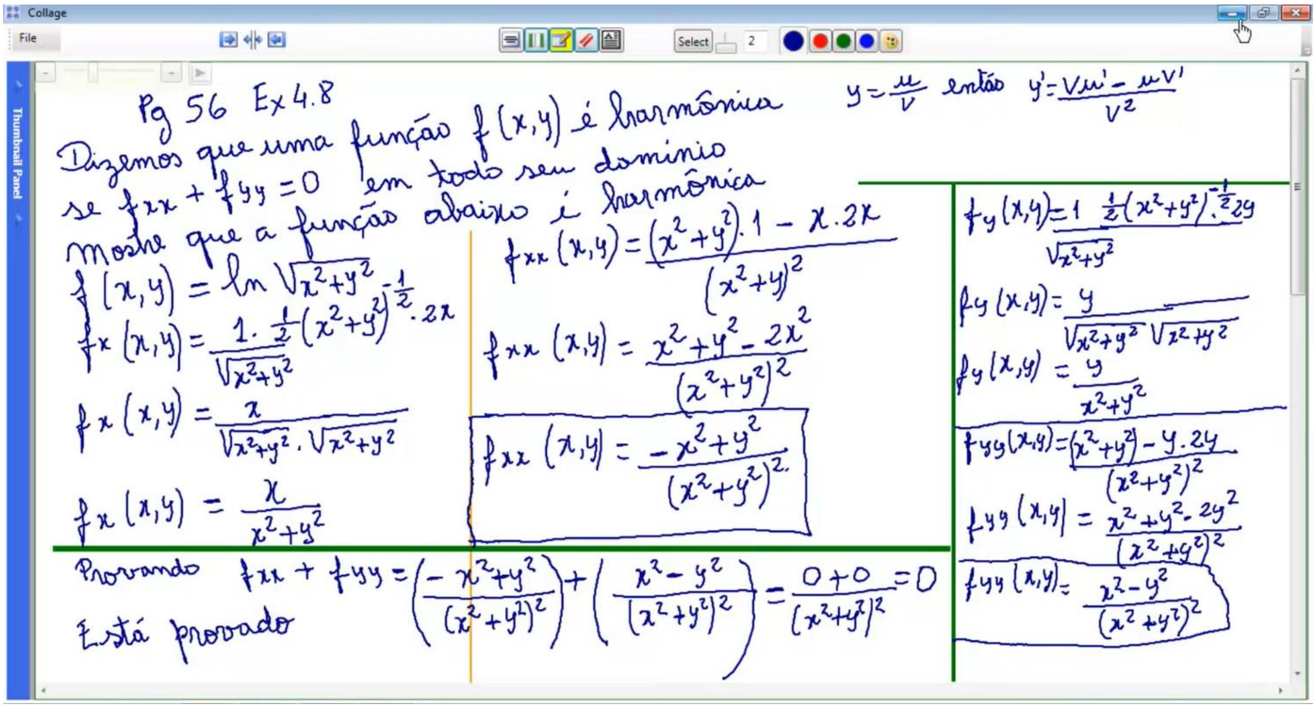

Figura 18 - Exercício Exemplo da Função Harmónica

4. Conclusão

Os métodos de derivação utilizando fórmulas de diferenças finitas e os métodos de integração através da regra dos trapézios e da regra de Simpson são técnicas numéricas úteis para aproximar derivadas e integrais de funções. Estes métodos são amplamente usados em cálculos numéricos e podem fornecer resultados precisos com um número adequado de subintervalos.

No entanto, é importante lembrar que estas técnicas são aproximações e podem ter erros associados. Por isso, é crucial avaliar a precisão desses métodos antes de os aplicar.

Em relação aos métodos de derivação através das fórmulas de diferenças finitas, verificamos que as diferenças finitas em 3 pontos têm uma precisão maior do que o método em 2 pontos.

5. Bibliografia

https://www.youtube.com/watch?v=cejEsc5o7zs

https://math.tecnico.ulisboa.pt/

https://moodle.isec.pt/moodle/mod/assign/view.php?id=254106