Análise Matemática II | Engenharia Informática

Trabalho Prático Nº3

Máquina para Derivação e Integração

Diogo Silva a2020138438
Hugo Ferreira a2020128305
Rúben Mendes a2020138473

2020/2021

Índice

Índice de Imagens

1. Introdução

Para aproximar o valor de derivada num ponto temos que aplicar uma das fórmulas de diferenças finitas. Com este trabalho, para além de termos de aplicar estas fórmulas, com esta atividade desenvolvemos também a derivação numérica e simbólica em Matlab bem como a implementação da regra dos trapezios e de simpsons.

O professor Arménio Correia propôs a seguinte atividade:

Fórmulas de diferenças finitas em 2 pontos:

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

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

Fórmulas de diferenças finitas em 3 pontos:

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

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

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

2ª derivada » $f''(x_k) := \dfrac{f(x_{k+1}) - 2f(x_k) + f(x_{k-1})}{h^2}$

Integração Numérica

$$\int_a^b f(x)\,dx \approx$$

Regra dos Trapézios

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

$$|E_T| \leq \frac{b-a}{12}h^2 M_2, \quad M_2 = \max_{x\in[a,b]}|f''(x)|$$

Regra de Simpson

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

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

Trapezios rule Algorithm	**Simpson's 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*)+2*s*+*f*(*b*))
	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*))

2. Métodos Numéricos para Derivação

2.1. Formulas

Fórmulas de diferenças finitas em 2 pontos:

Progressivas:

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

Regressivas:

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

Fórmulas de diferenças finitas em 3 pontos:

Progressivas:

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

Regressivas:

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

Centradas:

$$\frac{f(x_{k+1}) - f(x_{k-1})}{2h}$$

2ª Derivada:

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

2.2. Algoritmo/Função

Algoritmo:

DFProgressivas_2

Input: f,a,b,h,y Output: x,y,dydx

x <- Inicializar vetor de a até b com um salto de h

n <- Comprimento de x

Se nº de argumentos de entrada = 5

y <- f(x)

Fim Se

dydx <- Vetor de zero de 1 até n

Equanto i=1 ≤ n-1

dydx(i) <- (y (i+1) – y (i)) / h

Fim Enquanto

dydx(n) <- (y(n) – y(n-1)) / h

Função:

```
function [x,y,dydx]=DFProgressivas_2(f,a,b,h,y)
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)=(y(n)-y(n-1))/h;
```

Algoritmo:

DFRegressivas_2

Input: f,a,b,h,y

Output: x,y,dydx

x <- Inicializar vetor de a até b com um salto de h

n <- Comprimento de x

Se nº de argumentos de entrada = 5

y <- f(x)

Fim Se

dydx <- Vetor de zero de 1 até n

dydx(1) <- (y(2) – y(1)) / h

Equanto i=2 ≤ n

dydx(i) <- (y (i) – y (i-1)) / h

Fim Enquanto

Função:

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

Algoritmo:

DFProgressivas_3
Input: f,a,b,h,y
Output: x,y,dydx
x <- Inicializar vetor de a até b com um salto de h
n <- Comprimento de x

Se nº de argumentos de entrada = 5
y <- f(x)
Fim Se

Equanto i=1 ≤ n-2

dydx(i) <- ((-3) * y(i) + 4*y(i+1) – y(i+2)) / (2*h)

Fim Enquanto

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-1)) / (2*h)

Função:

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

Algoritmo:

DFRegressivas_3
Input: f,a,b,h,y
Output: x,y,dydx
x <- Inicializar vetor de a até b com um salto de h
n <- Comprimento de x

Se nº de argumentos de entrada = 5
    y <- f(x)
Fim Se

dydx <- Vetor de zero de 1 até 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)

Equanto i=3 ≤ n

    dydx(i) <- (y(i-2) - 4 * y(i-1) + 3*yi)) / (2*h)
Fim Enquanto

Função:

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

Algoritmo:

DFCentradas_3
Input: f,a,b,h,y
Output: x,y,dydx
x <- Inicializar vetor de a até b com um salto de h
n <- Comprimento de x
Se nº de argumentos de entrada = 5
    y <- f(x)
Fim Se

dydx <- Vetor de zero de 1 até n
dydx(1) <- (y(2) – Y(1)) / (2*h)
Equanto i=2 ≤ n-1

    dydx(i) <- (y(i+1) - y(i-1)) / (2*h)

Fim Enquanto

dydx(n) <- (y(n) – y(n-2)) / (2*h)

Função:

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

Algoritmo:

DFDerivadas_2
Input: f,a,b,h,y
Output: x,y,dydx
x <- Inicializar vetor de a até b com um salto de h
n <- Comprimento de x
Se nº de argumentos de entrada = 5
    y <- f(x)
Fim Se

dydx <- Vetor de zero de 1 até n
dydx(1) <- (y(2) – Y(1)) / (2*h)
Equanto i=2 ≤ n-1

    dydx(i) <- (y(i+1) - y(i-1)) / (2*h)

Fim Enquanto

dydx(n) <- (y(n) – y(n-2)) / (2*h)

Função:

```
function [x,y,dydx]=DI_DFDerivada2(f,a,b,h,y)
x=a:h:b;
```

```
n=length(x);
if nargin==5
    y=f(x);
end;
dydx=zeros(1,n);
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. Derivação Simbólica no Matlab

3.1. Diff

Funciona também com a função diff mas neste caso calcula as diferenças entre elementos adjacentes de X ao longo de um array cujo tamanho não é igual a 1. Pode ser representada das seguintes maneiras:

Y = diff(X)   Y = diff(X, n)   Y = diff(X, n, dim)

4. Métodos Numéricos para integração

4.1. Formulas

Regra dos Trapézios:

$$I_T(f) = \frac{h}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]$$
$$|E_T| \leq \frac{b-a}{12} h^2 M_2, \quad M_2 = \max_{x \in [a,b]} |f''(x)|$$

Regra de Simpson:

$$I_S(f) = \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$$
$$|E_S| \leq \frac{b-a}{180} h^4 M_4, \quad M_4 = \max_{x \in [a,b]} \left|f^{(4)}(x)\right|$$

4.2. Algoritmo/Função

Algoritmo:

```
RSimpson
Input: f,a,b,n
Output: t
h <- (b – a) / n
s <- 0
x <- a

Equanto i=1 ≤ n-1

        x <- x + h
        Se i/2 der resto 0
        s <- s + 2 * f(x)
        Se Não

                s <- s + 4 * f(x)

        Fim Se
Fim Enquanto
        s <- h * ( f(a) + f(b) + s ) /3
```

Função:

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

Algoritmo:

RTrapezios
Input: f,a,b,n
Output: t
h <- (b – a) / n
t <- 0
x <- a
Equanto i=1 ≤ n-1
    x <- x + h
    t <- t + 2 * f(x)

Fim Enquanto

t <- h * (f(a) + f(b) + t) / 2

Função:

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

5. Verificar se a função é harmónica

Função:

```
function verifica = harmonica(f)

syms x y;

if(diff(f,x,2)+diff(f,y,2)==0)
verifica = 1;
else
     verifica = 0;
end
```

6. Exemplos de aplicação e teste dos métodos

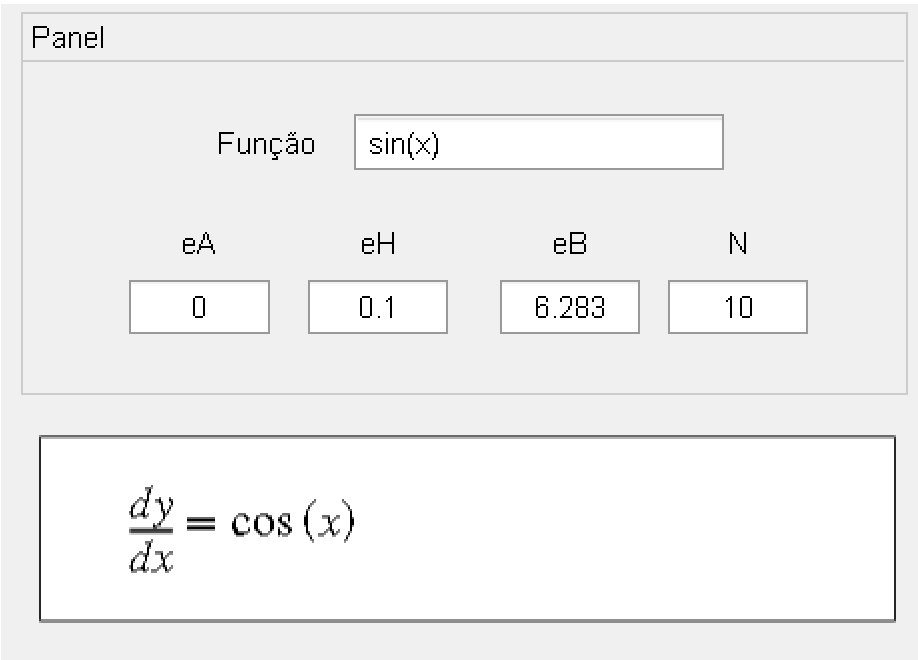

Figura 1: GUI - Derivadas

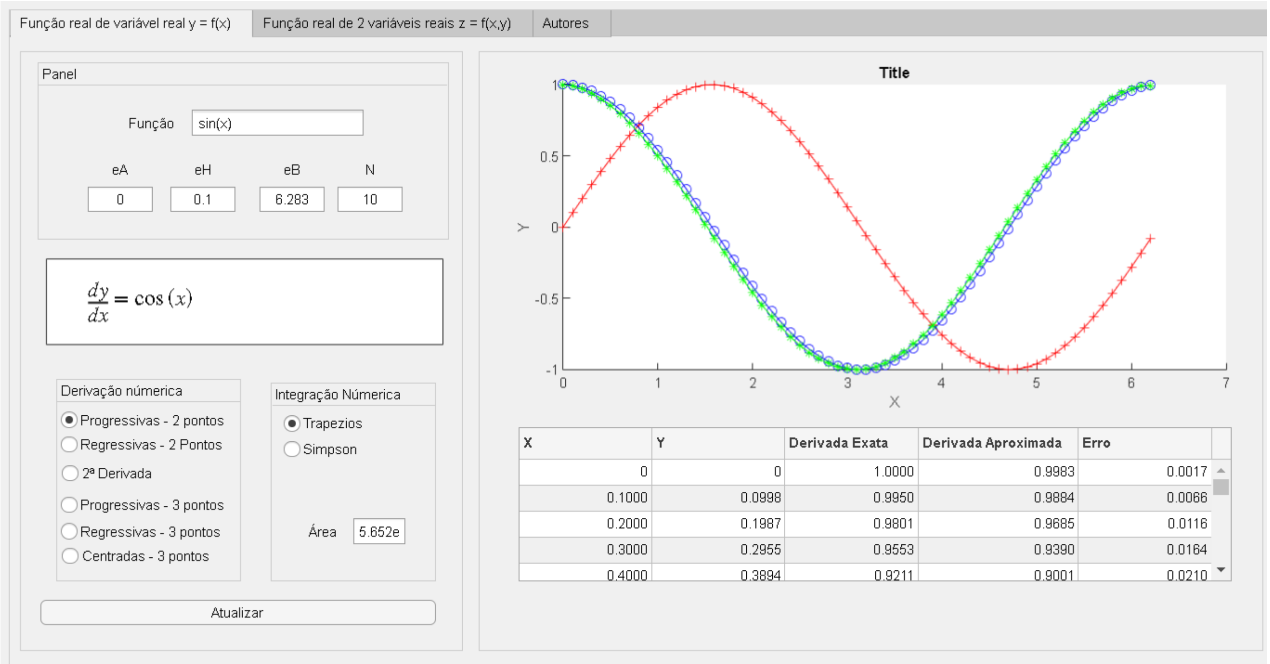

Figura 2: Função representada pelo Método das Diferenças Progressivas em 2 pontos

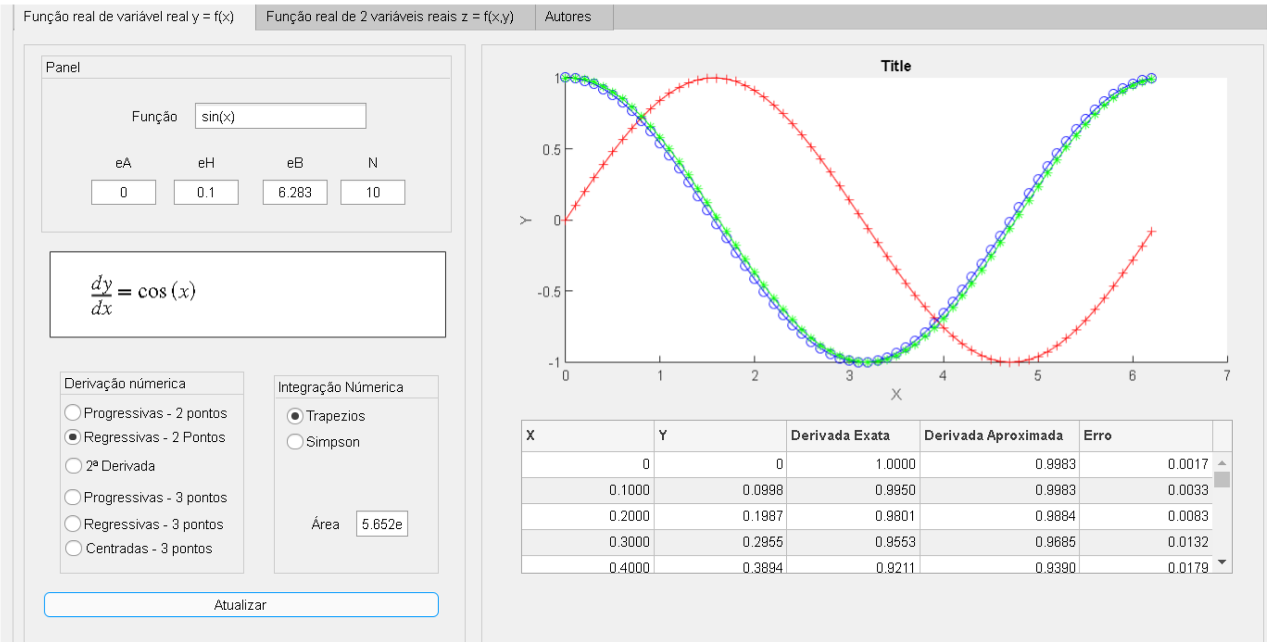

Figura 3: Função representada pelo Método das Diferenças Regressivas em 2 pontos

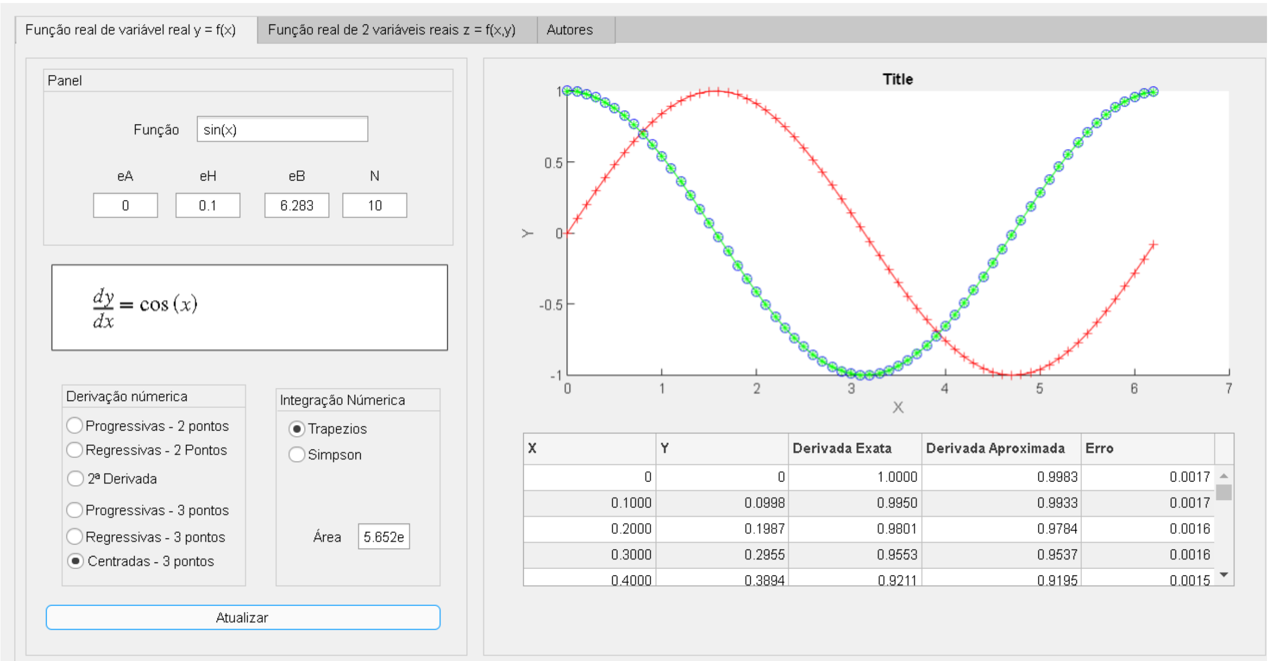

Figura 4: Função representada pelo Método das Diferenças Centradas

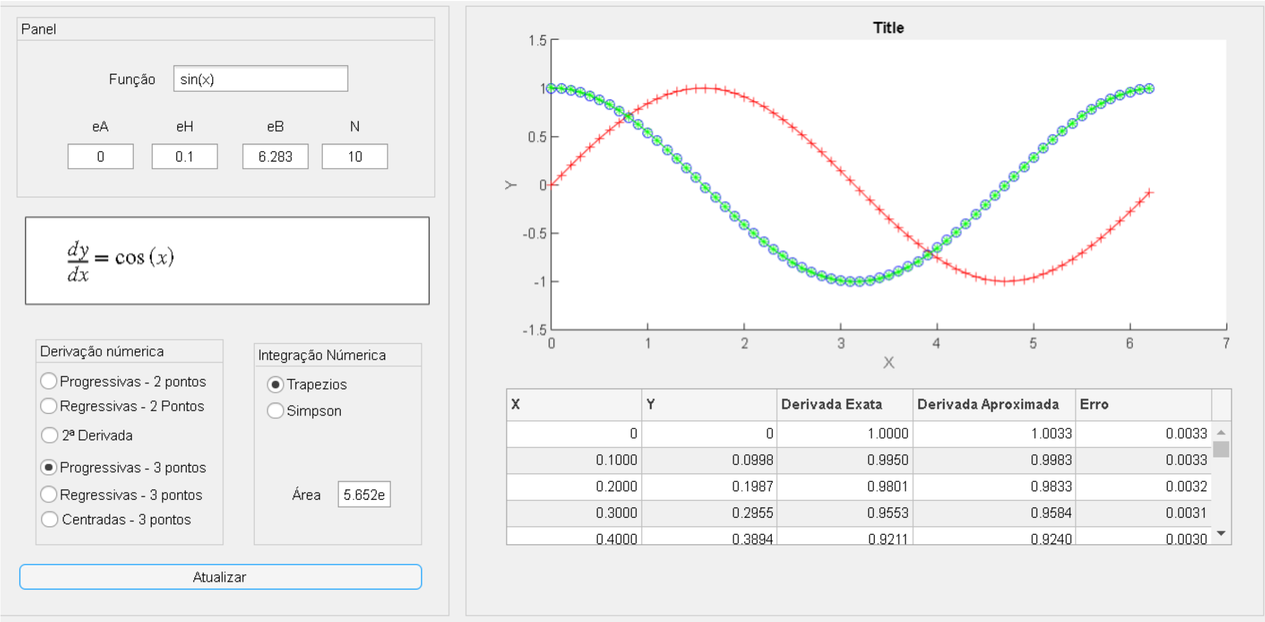

Figura 5: Função representada pelo Método das Diferenças Progressivas em 3 pontos

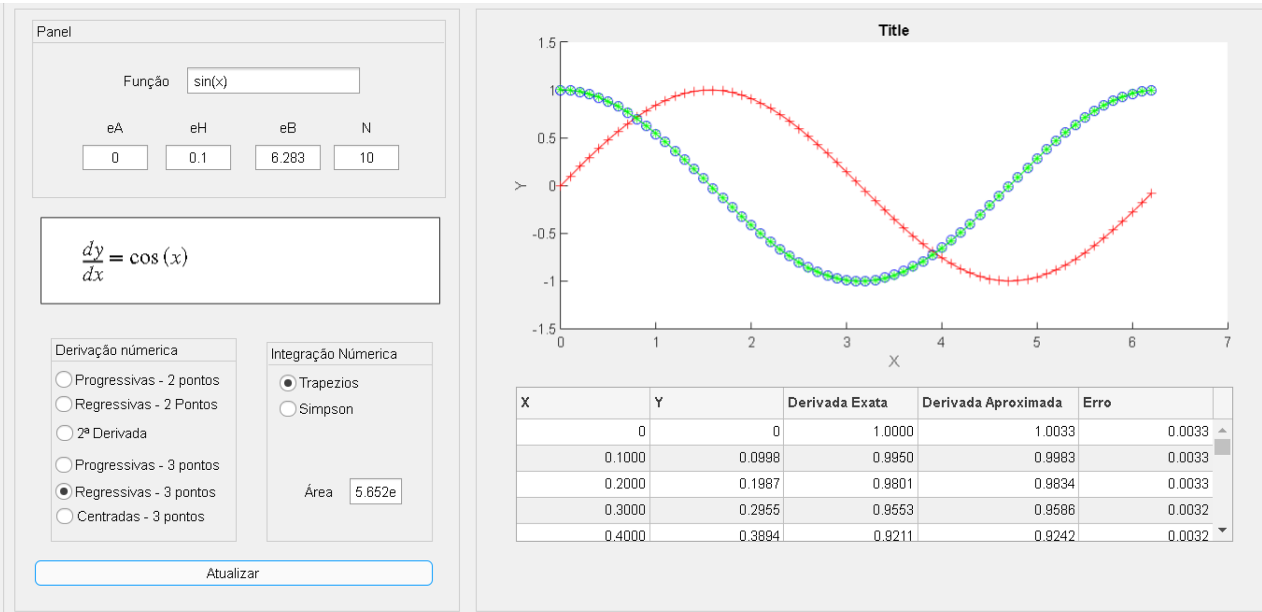

Figura 6: Função representada pelo Método das Diferenças Regressivas em 3 pontos

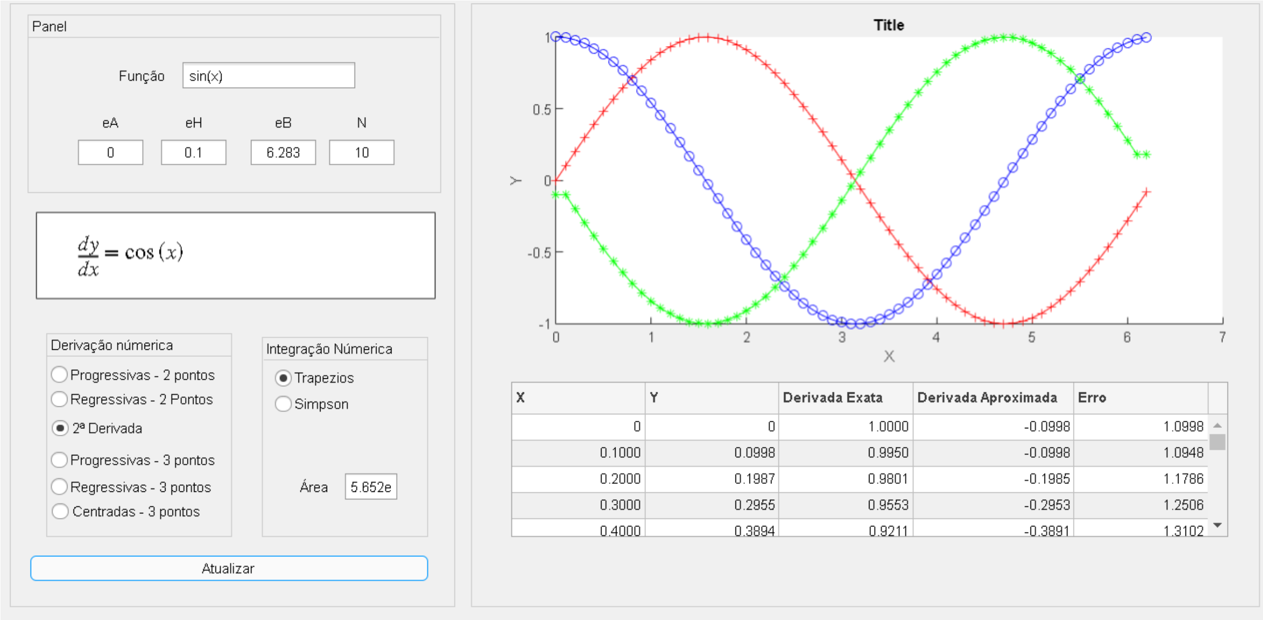

Figura 7: Função representada pelo Método da Segunda Derivada

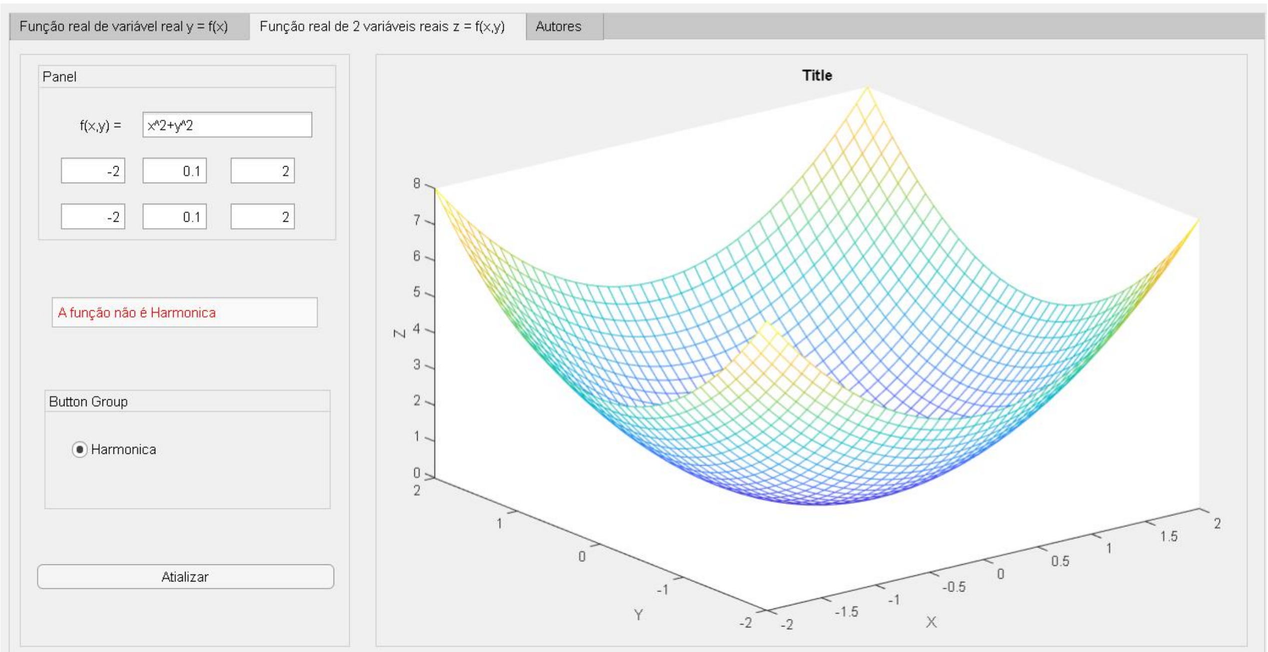

Figura 8:Verificar se a função é harmónica e construir o gráfico

7. Conclusão

Com a realização desta atividade, ficámos a conhecer melhor os métodos numéricos para derivação e integração e também ganhámos mais experiência a trabalhar com o MATLAB.

Aprendemos também a funcionar com APPS, sendo esta uma componente bastante prática que nos permite uma melhor perceção do conteúdo lecionado em aula.

Cumprindo todos os objetivos propostos pelo professor Arménio Correia, podemos concluir que todos os métodos nos dão uma boa aproximação dos valores pretendidos, de uma maneira mais simples e rápida.