Relatório - Máquina para Derivação e Integração

Departamento de Engenharia Informática e de Sistemas (DEIS)

Bruno Teixeira(2019100036), Rafael Ribeiro(2019131989), Gonçalo Correia(2019150122), Sofia Janeiro(2019132578)

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Neste relatório vamos introduzir a noção de derivada e de integral, e ao mesmo tempo vamos mostrar como foi criada a máquina de derivação e integração, mostrando alguns exemplos práticos com a GUI do MATLAB. Dizemos que derivada é o limite da razão entre o acréscimo de uma função e o acréscimo da variável independente, quando tende para zero. Já o integral de uma função foi criada para determinar uma área sob uma curva no plano cartesiano.

Capítulo 2

Desenvolvimento

Neste capítulo são apresentados os algoritmos em MATLAB referente a cada método que vamos usar.

2.1 Métodos numéricos para a derivação

Uma vez que na maior parte das vezes o cálculo de uma derivada é lento em termos de algoritmo, surgiu a ideia de criar aproximações numéricas. Ao falarmos destas aproximações numéricas, somos introduzidos às diferentes fórmulas de diferenças finitas. Dentro destas fórmulas há as progressivas, regressivas e centradas.

2.2 Fórmulas

2.2.1 Fórmula de diferenças finitas em 2 pontos

Progressivas

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

Regressivas

$$f'(x_k) = \frac{f(x_k) - f(x_k)}{h}$$

2.2.2 Fórmula 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

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

2.3 Funções em MATLAB

2.3.1 Derivada progressiva de diferenças finitas em 2 pontos

```
function [x,y,dydx]=DI_DFProgressivas_2(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;
```

Figura 2.1: Função - Derivada progressiva de diferenças finitas em 2 pontos

2.3.2 Derivada regressiva de diferenças finitas em 2 pontos

```
function [x,y,dydx]=DI_DFRegressivas_2(f,a,b,h,y)
x=a:h:b;
n=length(x);
if nargin==4
    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
```

Figura 2.2: Função - Derivada regressiva de diferenças finitas em 2 pontos

2.3.3 Derivada progressiva de diferenças finitas em 3 pontos

```
function [x,y,dydx]=DI_DFProgressivas_3(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)=( 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);
```

Figura 2.3: Função - Derivada progressiva de diferenças finitas em 3 pontos

2.3.4 Derivada regressiva de diferenças finitas em 3 pontos

```
function [x,y,dydx]=DI_DFRegressivas_3(f,a,b,h,y)
x=a:h:b;
n=length(x);
if nargin==4
    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
```

Figura 2.4: Função - Derivada regressiva de diferenças finitas em 3 pontos

2.3.5 Derivada de 2ª ordem de diferenças finitas em 3 pontos

```
function [x,y,dydx]=DI_DFDerivada_2(f,a,b,h,y)
x=a:h:b;
n=length(x);
if nargin==4
    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);
```

Figura 2.5: Função - Derivada de 2ª ordem de diferenças finitas em 3 pontos

2.3.6 Derivada centrada de diferenças finitas em 3 pontos

```
function [x,y,dydx]=DI_DFCentradas_3(f,a,b,h,y)
x=a:h:b;
n=length(x);
if nargin==4
    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;
```

Figura 2.6: Função - Derivada centrada de diferenças finitas em 3 pontos

2.4 Derivação simbólica no MATLAB

Função diff()
Fazendo uma pesquisa na documentação que o MATLAB nos dá, podemos ver que se fizermos Y=diff(x),calculamos a diferença entre os elementos adjacentes de "X"ao longo da primeira dimensão do array cujo o tamanho é diferente de 1.

Ou seja, com o **diff()** podemos calcular derivadas de qualquer ordem.

```
X = [1 1 2 3 5 8 13 21];
Y = diff(X)

Y = 1×7

     0     1     1     2     3     5     8
```

Figura 2.7: Exemplo da função diff()

No exemplo em baixo estamos a calcular uma derivada de 1ª ordem e comparamos o resultado que o MATLAB nos dá com o Symbolab.

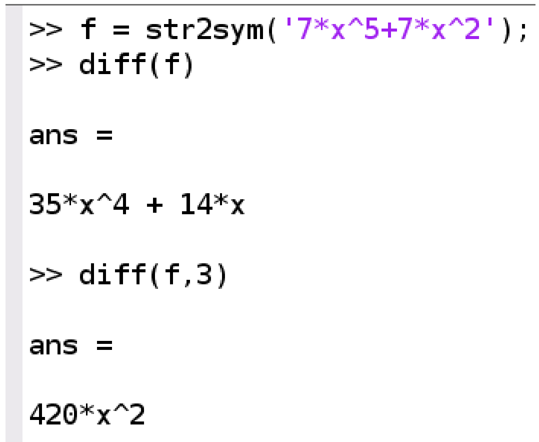
```
>> f = str2sym('4*x^2+2*x^2');
>> diff(f)

ans =

12*x
```

Figura 2.8: Exemplo da função diff() no MATLAB

$$\frac{d}{dx}\left(4x^2+2x^2\right)=12x$$

Figura 2.9: Resultado no Symbolab

Podemos tambem calcular derivadas da ordem que quisermos, basta especificar o número da ordem no segunda parâmetro da função. Em baixo estamos então a mostrar como podemos usar o **diff()** para calcular uma derivada de ordem 3.



```
>> f = str2sym('7*x^5+7*x^2');
>> diff(f)

ans =

35*x^4 + 14*x

>> diff(f,3)

ans =

420*x^2
```

Figura 2.10: Resultado no MATLAB da derivada de 3ª ordem

2.5 Métodos Numéricos para Integração

Existem vários métodos numéricos para a integração. Neste caso iremos falar da **Regra dos Trapézios** e da **Regra de Simpson**.

Regra Dos Trapézios

Trata-se do caso mais simples da **Fórmula de Newton-Cotes fechada**.Neste caso, condiseramos n= 1 e temos dois nós de integração.

Regra de Simpson

A regra de Simpson é mais uma vez um exemplo da **Formula de Newton-Cotes fechada**, mas em vez de considerarmos a aproximação e cada sub-intervalo através de um polinómio interpolador do 1º grau, pensamos numa aproximação melhor considerando um polinómio de 2º grau, ou seja, uma parábola. Para isto precisamos então de um ponto médio para considerarmos a regra de integração simples.

2.5.1 Fórmulas

Regra dos Trapézios

$$T(f) = I_1(f) = (f(a) + f(b))(b - a)/2$$

Regra de Simpson

$$S(f) = I_2(f) = (f(a) + 4f(c) + f(b))h/3$$

2.5.2 Algoritmos e Funções em MATLAB

Em baixo estarão então representados os algoritmos, para a **Regra dos Trapézios** e para a **Regra de Simpson**.

```
Inputs = f,a,b e n
Outputs = 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 2.11: Algoritmo - Trapézios

```
Inputs = f,a,b e n
Outputs = 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 2.12: Algoritmo - Simpson

Uma vez que já temos as fórmulas e os algoritmos, conseguimos agora fazer uma função que aplica essas fórmulas no MATLAB.

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

h = (b-a)/n;
s = 0;
x=a;

for i=1:n-1
    x=x+h;
    s = s+f(x);
end

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

Figura 2.13: Função - Trapézios

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

h = (b-a)/n;
s = 0;
x=a;

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

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

Figura 2.14: Função - Simpson

2.6 Integração simbólica no MATLAB

Função int()
Ao pesquisar no site do MATLAB, podemos ver que:
F = int(expr) calcula o integral indefenido de **expr** e usa a varivel de integração padrão determinada por **symvar(expr,1)**. Se **expr** for uma constante a variável padrão é x.

```
syms x
expr = -2*x/(1+x^2)^2;
```

```
F = int(expr)
```

F =

$$\frac{1}{x^2+1}$$

Figura 2.15: Exemplo da função F = int(expr)

F = int(expr, var) calcula o integral indefenido de **expr** tendo em conta a variável simbólica escalar var.

```
syms x z
f(x,z) = x/(1+z^2);
```

```
Fx = int(f,x)
```

Fx(x, z) =

$$\frac{x^2}{2\,(z^2+1)}$$

```
Fz = int(f,z)
```

Fz(x, z) = $x\,\mathrm{atan}(z)$

```
var = symvar(f,1)
```

var = x

```
F = int(f)
```

F(x, z) =

$$\frac{x^2}{2\,(z^2+1)}$$

Figura 2.16: Exemplo da função F = int(expr, var)

F = int(expr, a, b) calcula o integral indefenido de **expr** de a até b e usa variável de integração determinada por **symvar(expr,1)**. Se **expr** for uma constante a variável padrão é x.

```
syms x
expr = x*log(1+x);
F = int(expr,[0 1])
```

F =

$$\frac{1}{4}$$

```
syms t
F = int(2*x,[sin(t) 1])
```

F = $\cos(t)^2$

Figura 2.17: Exemplo da função F = int(expr, a, b)

F = int(expr,var,a,b) calcula o integral indefenido de **expr** tendo em conta a variável simbólica escalar var de **a** até **b**.

```
syms x
f = cos(x)/sqrt(1 + x^2);
Fint = int(f,x,[0 10])
```

Fint =

$$\int_0^{10} \frac{\cos(x)}{\sqrt{x^2+1}}\,dx$$

```
Fvpa = vpa(Fint)
```

Fvpa = 0.37570628299079723478493405557162

Figura 2.18: Exemplo da função F = int(expr,var,a,b)

2.7 Exemplos de aplicação

2.7.1 Derivação

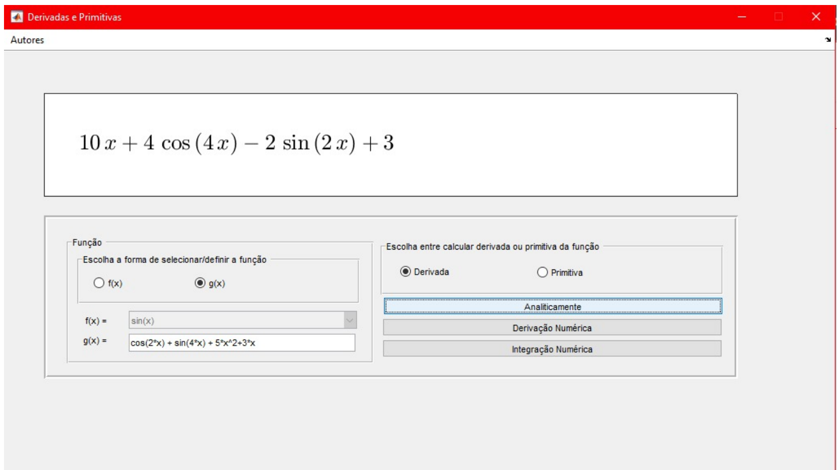


Figura 2.19: Funcionamento de uma derivada de g(x)

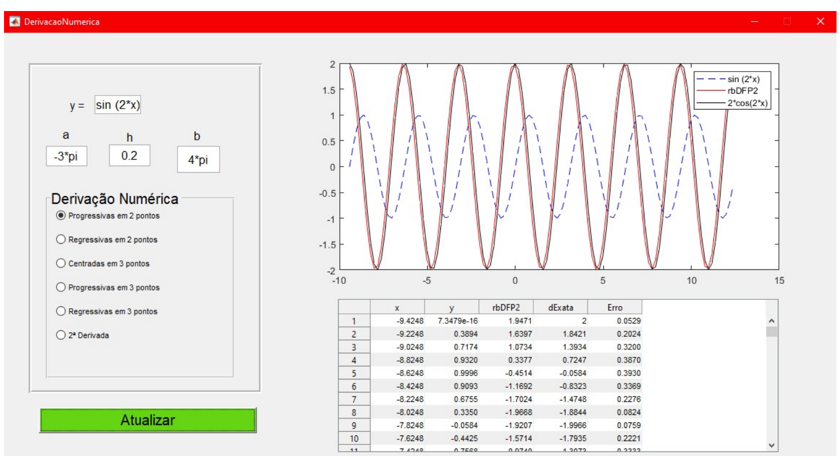


Figura 2.20: Representação gráfica - Progressivas em 2 pontos

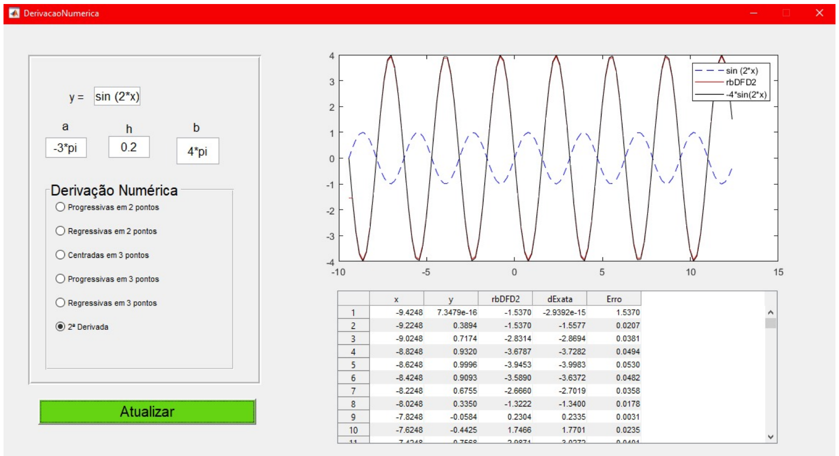


Figura 2.21: Representação gráfica - 2ª derivada

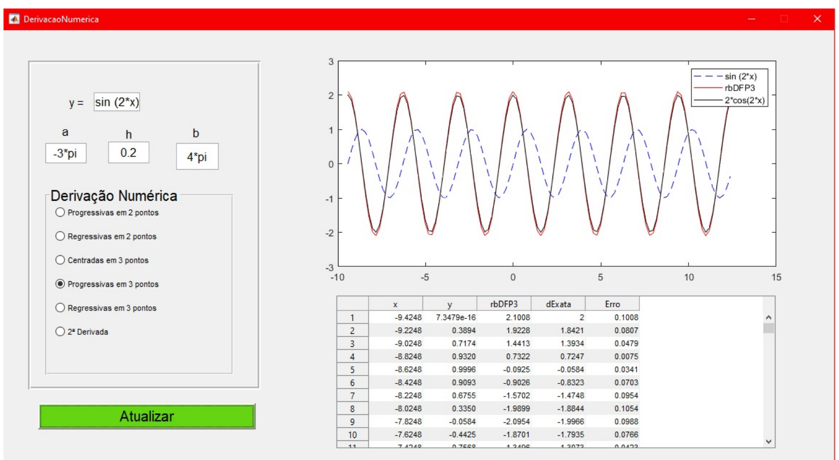


Figura 2.22: Representação gráfica - Progressivas em 3 pontos

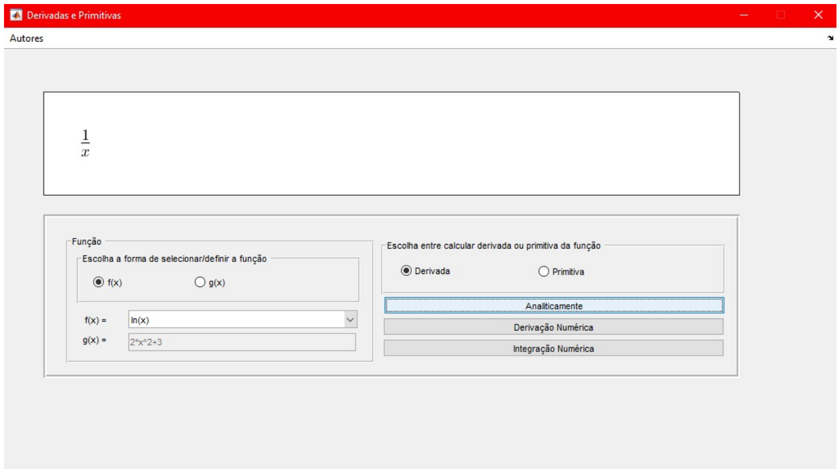


Figura 2.23: Funcionamento de uma derivada de f(x)

2.7.2 Primitivação

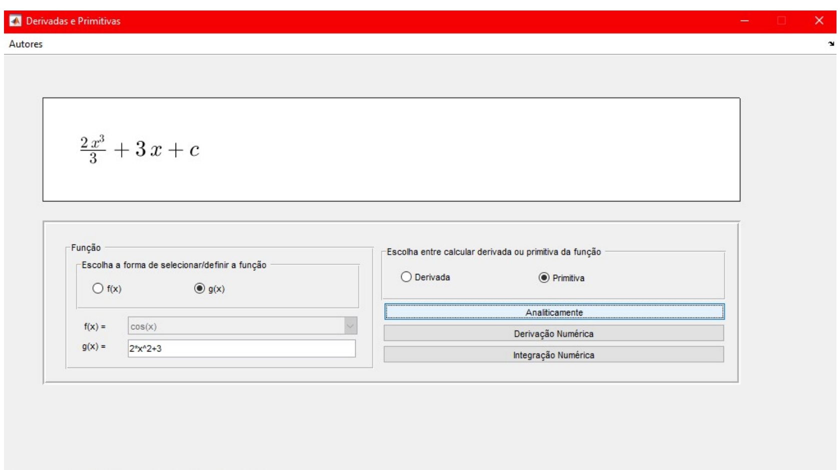


Figura 2.24: Funcionamento de uma primitiva de g(x)

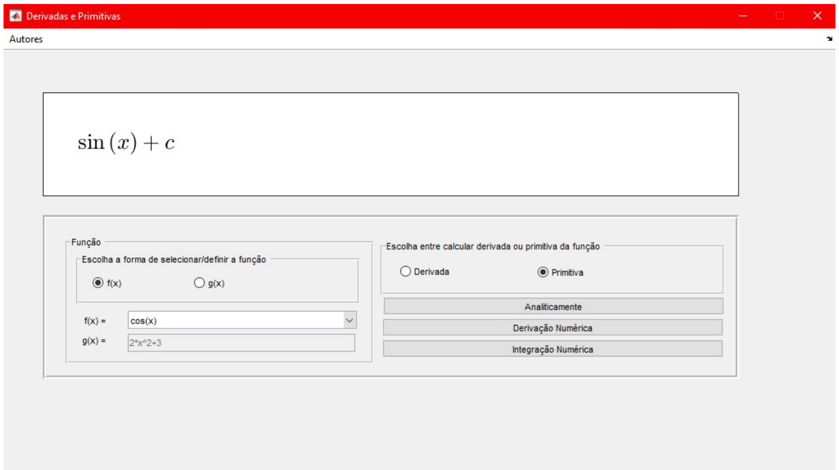


Figura 2.25: Funcionamento de uma primitiva de f(x)

Capítulo 3

Conclusão

Este trabalho prático reflete a importância das equações diferencias ordinárias na modelagem de problemas em diversas áreas científicas , bem como o uso de métodos númericos para resolver estas equações.

Atualmente, os meios de navegação, como os computadores, são uma ferramente indispensável no estudo destas equações diferenciais, uma vez que é através deles que temos a facilidade e a possibilidade de executar algoritmos de maneira a construir aproximações numéricas para obter as soluções destas equações.

Para resolver os problemas que nos foram apresentados, utilizámos o software MATLAB, cujo aprendemos nas aulas da disciplina de AM2, tendo por base ficheiros disponibilizados pelos professores da disciplina.

Assim, podemos concluir que os métodos utilizados neste trabalho nos apresentam resultados aproximados de uma maneira eficaz, mantendo a simplicidade que o programa requer.

DEIS Departamento de Engenharia Informática e de Sistemas