



Análise Matemática II

Atividade 03 – Métodos Numéricos para Derivação e Integração

Docente: Arménio Correia

Carlos Miguel Parreira Pais nº 2010017171 – LEI

José Fernando Esteves de Almeida nº 2019129077 – LEICE

Pedro Rodrigues nº2019136525 – LEICE

Coimbra, 27 de Maio de 2020

Índice

1. Introdução.....	3
2. Métodos Numéricos para Derivação.....	4
2.1 Fórmulas de Diferenças Finitas em 2 pontos.....	5
2.1.1 Progressivas	6
2.1.2 Regressivas	8
2.2 Fórmulas de Diferenças Finitas em 3 pontos	
2.2.1 Progressivas	10
2.2.2 Regressivas	12
2.2.3 Centradas	14
2.3 2ª Derivada.....	16
3. Derivação simbólica no MATLAB	18
3.1. Diff (MATLAB).....	18
4. Métodos Numéricos para Integração.....	19
4.1 Regra dos Trapézios	20
4.2 Regra de Simpson.....	22
5. Integração simbólica no Matlab	24
5.1 Int (MATLAB).....	24
6. Exemplos de aplicação e teste dos métodos	25
6.1 Problemas de Movimento (Integração)	25
6.2 Integração.....	29
6.3 Problemas de Movimento (Derivação).....	32
6.4 Derivação	34
7. Conclusão.....	37
8. Anexos	38
9. Bibliografia.....	43

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 e 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 GUI em linguagem de programação 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.

Além disso é pretendido apresentar uma breve pesquisa sobre os Métodos Numéricos para Derivação e Integração, assim como a Derivação e Integração Simbólica no MATLAB e mostrar alguns exemplos de aplicação e testes dos métodos.

O trabalho está dividido em duas partes:

- 1ª Parte – Métodos Numéricos para Derivação e Integração e Derivação e Primitivação simbólica no MATLAB;
- 2ª Parte – Exemplos de aplicação e teste dos Métodos Numéricos.

2. Métodos Numéricos para Derivação (Fórmulas das Diferenças Finitas)

Para entendermos a Derivação Numérica temos primeiro de perceber o significado de derivada. A derivada de uma função f em um ponto pode ser descrita graficamente como a inclinação da reta tangente à função naquele ponto.

A Derivação Numérica é utilizada para calcular a derivada em situações onde não está disponível a função, e sim apenas um conjunto de pontos pertencentes a esta, ou para funções que não são deriváveis em todo o seu domínio ou de derivação não trivial.

À medida que h diminui, o valor da derivada numérica aproxima-se do valor real. No entanto, por mais pequeno que seja h , este método irá sempre apresentar um erro de arredondamento grande. Uma maneira de reduzir este erro é utilizar vários pontos.

O objetivo é: partir de um conjunto de pontos, que definem um intervalo $[a,b]$, e determinar a função f que representa tais pontos, ou seja, interpolar este conjunto de pontos. Em seguida, pode-se calcular a derivada da função f e aplicá-la a qualquer ponto pertencente ao intervalo $[a,b]$. Quanto maior for o número de pontos melhor será o resultado. Normalmente utiliza-se fórmulas de 3 a 5 pontos.

Aplicação das F.D.F. em MATLAB

INPUT:

- f - função
- $[a, b]$ - intervalo de derivação
- h - passo da discretização
- y - vetor das ordenadas (opcional)

OUTPUT:

- $[x, y]$ - malha de pontos
- $dydx$ - derivada de f

CÓDIGO:

- Varia de método para método

2.1 Fórmulas de Diferenças Finitas em 2 pontos

O método mais simples para aproximar a derivada é usar o método de diferenças finitas. Uma diferença finita é uma expressão da forma $f(x+b)-f(x+a)$, que ao ser dividida por $(b-a)$ passa a ser chamada de quociente de diferenças. A técnica de **diferenças finitas** consiste em aproximar a derivada de uma função via fórmulas discretas que requerem apenas um conjunto finito de pares ordenados, onde geralmente denotamos $y_i = f(x_i)$

2.1.1 Progressivas

Fórmula:

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

onde:

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

Algoritmo:

1. Alocar memória para x ;
2. Definir o número de pontos (n);
3. Se forem recebidos 4 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 i ésima iteração;
6. Calcular a derivada (aproximada) de f no ponto atual, em n .

Função (MATLAB):

```
function [x,y,dydx] = NDerivacaoDFP(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

Fórmula:

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

onde:

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

Algoritmo:

1. Alocar memória para x ;
2. Definir o número de pontos (n);
3. Se forem inseridos 4 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 i ésima iteração.

Função (MATLAB):

```
function [x,y,dydx] = NDerivacaoDFR(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 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:

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

Algoritmo:

1. Alocar memória para x ;
2. Definir o número de pontos (n);
3. Se forem inseridos 4 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 i ésima 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] = NDerivacaoDFP3(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);
```

```
% Alocação de memória

% Número de pontos (tamanho do vetor de abcissas)

% y é a função f(x)

% Alocação de memória

% Derivada (aproximada) de f no ponto atual

% Derivada (aproximada) de f no ponto atual
% Derivada (aproximada) de f no ponto atual
```

2.2.2 Regressivas

Fórmula:

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

onde:

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

Algoritmo:

1. Alocar memória para x ;
2. Definir o número de pontos (n);
3. Se forem inseridos 4 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 i ésima iteração.

Função (MATLAB):

```
function [x,y,dydx] = NDerivacaoDFR3(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

Fórmula:

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

onde:

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

Algoritmo:

1. Alocar memória para x ;
2. Definir o número de pontos (n);
3. Se forem inseridos 4 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 i ésima iteração;
7. Calcular a derivada (aproximada) de f no ponto atual, em n .

Função (MATLAB):

```
function [x,y,dydx] = NDerivacaoDFC(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

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

2.3 2ª Derivada

Fórmula:

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

onde:

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

Algoritmo:

1. Alocar memória para x ;
2. Definir o número de pontos (n);
3. Se forem inseridos 4 elementos, y recebe o valor de $f(x)$;
4. Alocar memória para a derivada;
5. Calcular a 1ª derivada no ponto: $x = a$;
6. Calcular a 1ª derivada no ponto: $x = a + h$;
7. Calcular a 1ª derivada no ponto: $x = a + h * 2$;
8. Calcular a 2ª derivada no ponto: $x = a$;
9. Para i de 2 a $n-1$, calcular a derivada (aproximada) de f no ponto atual, para a i ésima iteração;
10. Calcular a 1ª derivada no ponto: $x = b - h * 2$;
11. Calcular a 1ª derivada no ponto: $x = b - h$;
12. Calcular a 1ª derivada no ponto: $x = b$;
13. Calcular a 2ª derivada no ponto: $x = b$.

Função (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);

temp1=(-3*y(1) + 4*y(2) - y(3))/(2*h);
temp2=(-3*y(2) + 4*y(3) - y(4))/(2*h);
temp3=(-3*y(3) + 4*y(4) - y(5))/(2*h);

dydx(1)=(-3*temp1 + 4*temp2 - temp3)/(2*h);

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

tempn2=(y(n-4) - 4*y(n-3) + 3*y(n-2))/(2*h);
tempn1=(y(n-3) - 4*y(n-2) + 3*y(n-1))/(2*h);
tempn=(y(n-2) - 4*y(n-1) + 3*y(n))/(2*h);

dydx(n)=(tempn2 - 4*tempn1 + 3*tempn)/(2*h);
```

```
% Alocação de memória

% Número de pontos (tamanho do vetor de abcissas)

% y é a função f(x)

% Alocação de memória

% Primeira derivada no ponto x = a
% Primeira derivada no ponto x = a + h
% Primeira derivada no ponto x = a + 2*h

% Segunda derivada no ponto x = a

% Derivada (aproximada) de f no ponto atual

% Primeira derivada no ponto x = b - 2*h
% Primeira derivada no ponto x = b - h
% Primeira derivada no ponto x = b

% Segunda derivada no ponto x = b
```

3. Derivação Simbólica no MATLAB

Para além de cálculo numérico, o MATLAB também permite efectuar cálculo simbólico. Para esse efeito, existe uma toolbox de matemática simbólica (“Symbolic Math Toolbox”).

Esta toolbox permite efectuar cálculos de diferentes naturezas, designadamente:

- Cálculo propriamente dito (diferenciação, integração, determinação de limites, somatórios, série de Taylor, etc.);
- Álgebra linear (inversa de uma matriz, determinantes, valores próprios, etc.);
- Simplificação de expressões algébricas;
- Obtenção de soluções analíticas de equações algébricas e diferenciais;
- Transformadas de Laplace, Z e de Fourier, etc., etc.

3.1. Diff (MATLAB)

Como visto anteriormente, a *toolbox* de matemática simbólica disponibiliza um conjunto de funções de cálculo, entre as quais se destaca a derivação que pode ser efetuada utilizando o comando *diff*.

Este comando permite calcular derivadas de várias ordens, em que em derivadas de ordem superior à primeira é necessária a utilização de mais um parâmetro de entrada.

Exemplo de utilização da função *diff* em MATLAB para o cálculo da Derivada Exata:

```
function f = DerivadaExata(strF)

syms x                                     % cria a variável simbólica x
h = @(x) eval(vectorize(strF));

f = matlabFunction(diff(h(x)));           % Calcula, através da função diff, a derivada exata
```

4. Métodos Numéricos para Integração

Na Matemática existe uma grande variedade de algoritmos cujo principal objetivo é aproximar o valor de uma dada integral definida de uma função sem o uso de uma expressão analítica para a sua primitiva.

Normalmente, estes métodos são constituídos pelas seguintes fases:

- Decomposição do domínio em subintervalos (um intervalo contido de subintervalos);
- Integração aproximada da função de cada subintervalo;
- Soma dos resultados numéricos obtidos.

Razões da necessidade de usar a integração numérica:

- Algumas funções não admitem uma primitiva de forma explícita;
- A primitiva da função é muito complicada para ser avaliada;
- Quando não se dispõe de uma expressão analítica para o integrando, mas conhece-se os seus valores em um conjunto de pontos do domínio.

A Integração Numérica de uma função $f(x)$ num intervalo $[a, b]$ consiste no cálculo da área delimitada por essa função, recorrendo à interpolação polinomial, como forma de obtenção de um polinómio $p_n(x)$.

O método básico envolvido nesta aproximação é chamado de quadratura numérica e consiste na seguinte expressão:

$$\int_a^b f(x)dx \simeq \sum_{i=0}^n \alpha_i f(x_i)$$

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.

Este método está dividido em 3 passos:

- Preencher a parte inferior da função (em cada subintervalo) com trapézios;
- Calcular as suas áreas;
- Somar os resultados do cálculo das várias áreas.

Nota: Quanto mais subintervalos existirem mais precisa se torna a aproximação.

Fórmula:

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

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 abcissa x_{n-1} ;
- $f(x_n)$ → Valor da função na abcissa x_n ;
- h → Valor de cada subintervalo (passo).

Algoritmo:

1. Calcular o passo (h);
2. Atribuir o valor de a a x ;
3. Inicializar s com o valor 0 ;
4. Para i de 1 a $n-1$:
 - a. Somar o passo (h) a x ;
 - b. 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 = INumRTrapezios(f,a,b,n)

h=(b-a)/n;           % Valor de cada subintervalo (passo)
x=a;                 % x recebe o valor de a
s=0;                 % Inicialização da variável s a 0

for i=1:n-1
    x=x+h;           % Ao valor de x é somado o passo (h)
    s=s+f(x);        % Ao valor de s é somado o valor da função
end
T=(h/2)*(f(a)+2*s+f(b)); % Cálculo da Regra dos Trapézios
```

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:

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

Algoritmo:

1. Cálculo do passo (h);
2. Atribuir o valor de a a x ;
3. Inicializar s com o valor 0 ;
4. Para i de 1 a $n-1$:
 - a. Somar o passo (h) a x ;
 - b. Se i for par, soma-se 2 vezes o valor de $f(x)$ a s ;
 - c. 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 out_S = INumRSimpson(f,a,b,n)

h=(b-a)/n;           % Valor de cada subintervalo (passo)
x=a;                 % x recebe o valor de a
s=0;                 % Inicialização da variável s a 0

for i=1:n-1
    x=x+h;           % Ao valor de x é somado o passo (h)
    if mod(i,2) == 0 % Se i for par
        s=s+2*f(x);  % Ao valor de s é somado 2 vezes o valor da função
    else
        s=s+4*f(x);  % Ao valor de s é somado 2 vezes o valor da função
    end
end
out_S=(h/3)*(f(a)+s+f(b)); % Cálculo da Regra de Simpson
```

5. Integração Simbólica no Matlab

Como já referido anteriormente, através da *toolbox* de Matemática simbólica “Symbolic Math Toolbox”, é possível efetuar cálculos de diferentes naturezas. Uma das diversas possibilidades de cálculo é a de Integração.

5.1 Int (MATLAB)

A função *int* pode ser usada para calcular integrais definidos e indefinidos.

Exemplo de utilização da função *int* em MATLAB para o cálculo da solução exata do integral:

```
sExata=int(f(sym('x')),a,b)
```

Onde a e b são os extremos do intervalo de integração $[a, b]$ e f a função integranda.

6. Exemplos de aplicação e teste dos métodos

6.1 Problemas de movimento (com integrais definidos)

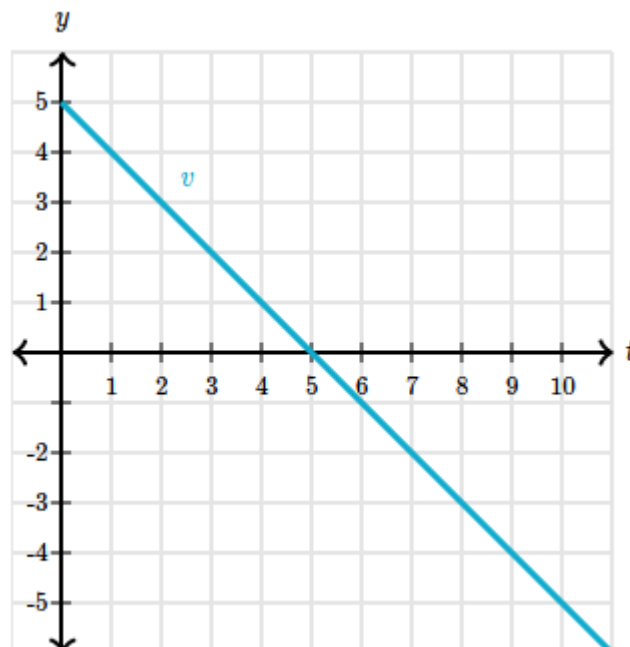
Informação e exercício retirados da “Khan Academy”:

Integrais definidos são muito usados para resolver problemas de movimento, por exemplo para descobrir a posição de um objeto em movimento dadas informações sobre a sua velocidade.

Problemas que envolvem movimento são muito comuns no cálculo. No cálculo diferencial, raciocinamos sobre a velocidade de um objeto em movimento, dada a sua função de posição. No cálculo integral, fazemos o contrário: dada a função de velocidade de um objeto em movimento, raciocinamos sobre a sua posição ou sobre a variação da sua posição.

Enunciado: Velocidade e Integrais Definidos

Digamos que uma partícula se move em linha reta, com velocidade de $v(t) = 5 - t$ metros por segundo, na qual o tempo t é dado em segundos.



Encontre o deslocamento da partícula (ou seja, a variação da sua posição) entre $t = 0$ e $t = 10$.

Resolução:

A variação da posição da partícula é dada pela área entre a função $v(t)$ e o eixo dos xx . Esta área é dada pelo seguinte integral definido:

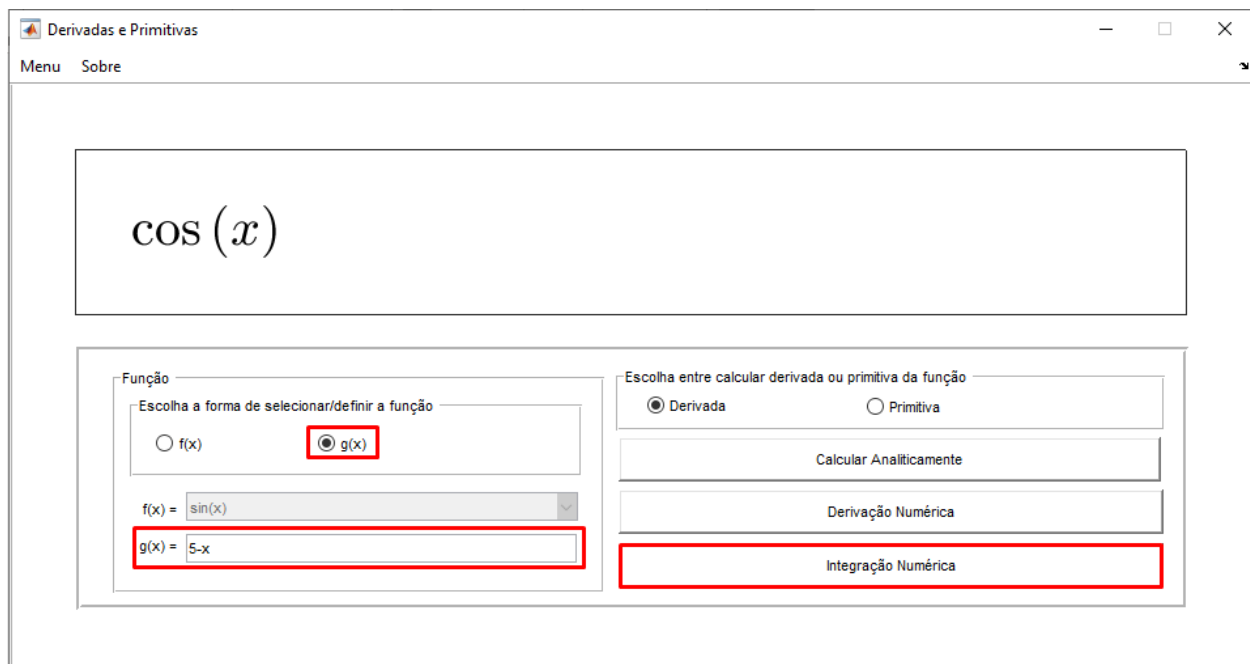
$$\int_0^{10} v(t) dt$$

Com $v(t) = 5 - t$.

Assim, através da nossa GUI, este integral é facilmente calculável, em alguns simples passos:

Primeiro, seleccionar a opção $g(x)$, que nos permite introduzir, agora, a função $v(t) = 5 - t \leftrightarrow v(x) = 5 - x$.

Seguidamente, carregar no botão “Integração Numérica”, que nos levará para outra figura.

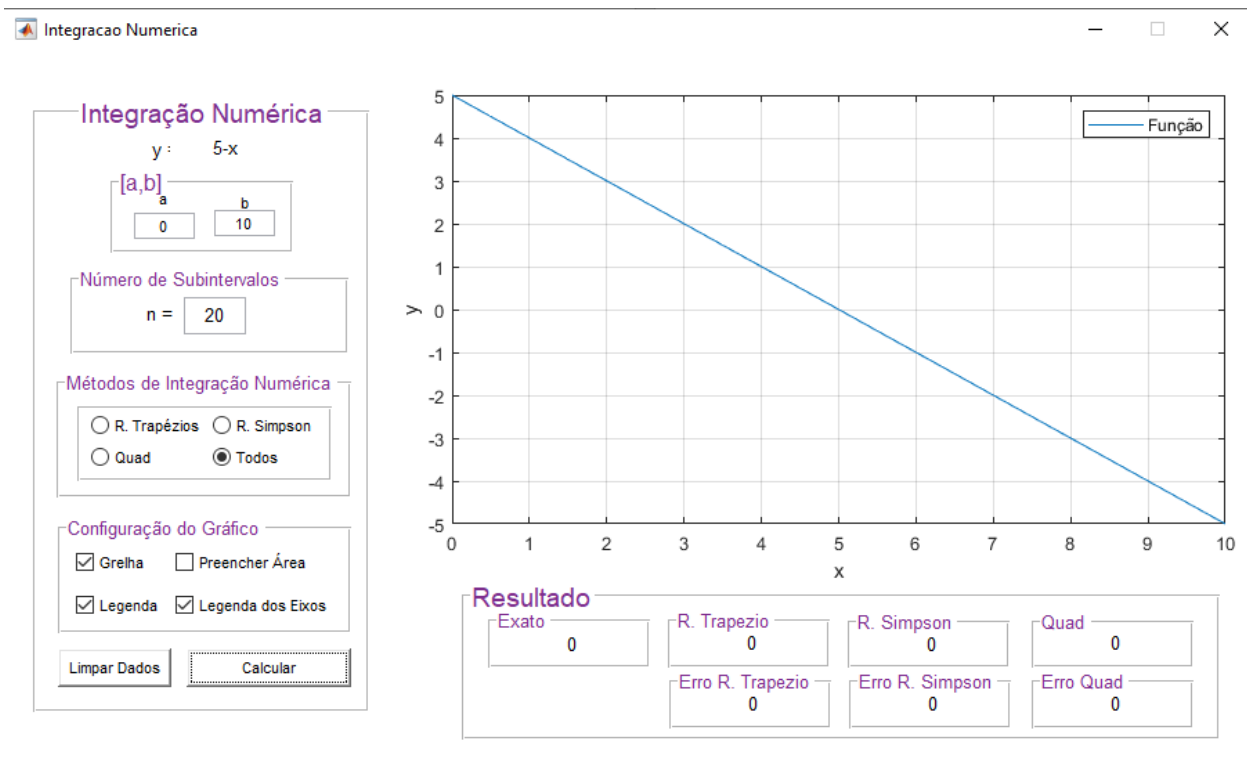


Como podemos observar, a nova figura automaticamente associou a y a função que introduzimos.

Escolhemos o intervalo pretendido. No nosso caso, temos o intervalo $[0,10]$ (ou seja, $a = 0$ e $b = 10$).

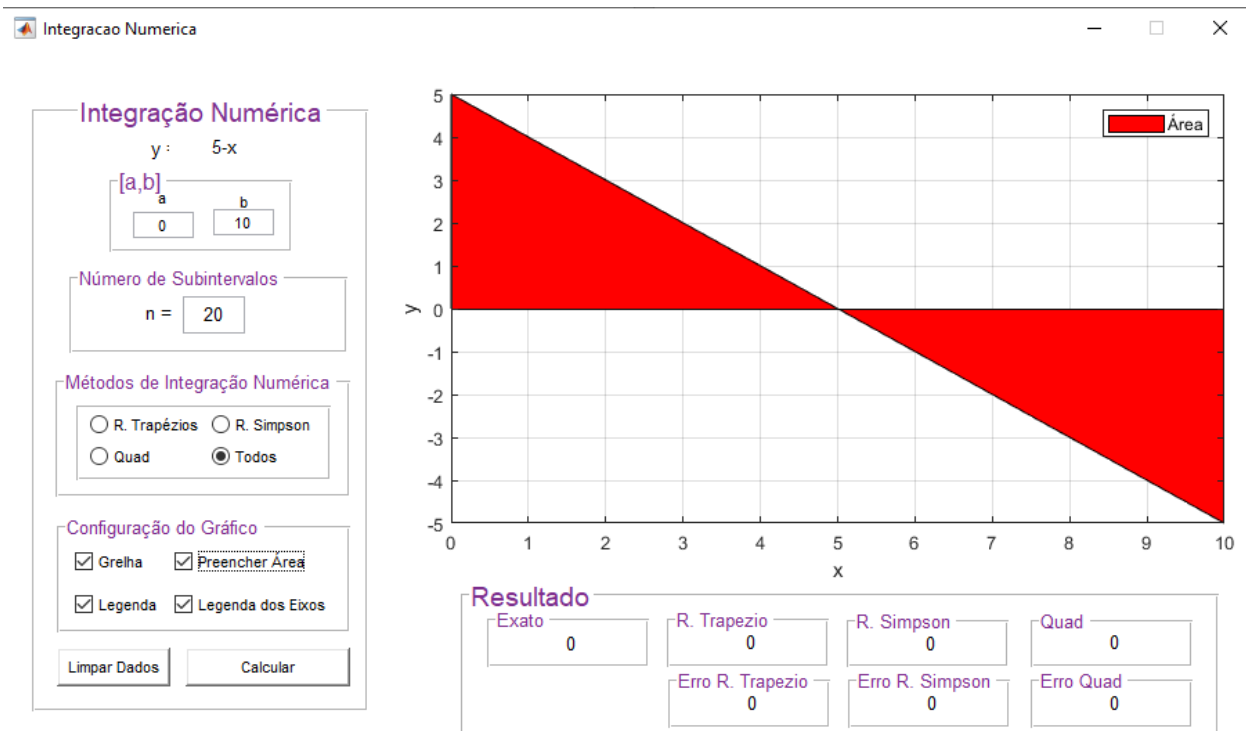
Temos agora que tomar a decisão de qual Regra usar para aproximar numericamente o nosso resultado (ou até mesmo escolher todos), assim como o número de subintervalos (quanto maior, menor o erro, ou seja, melhor a aproximação ao valor real). Existem também várias opções que podemos seleccionar, relativamente ao aspeto do gráfico.

Por fim, carregamos no botão “Calcular” e obtemos o seguinte:



Et voilà! Aqui está o resultado, que neste caso é zero, ou seja, deslocamento ao fim de 10 segundos totalizou zero metros (a partícula voltou à origem).

Para percebermos o porquê, podemos seleccionar a opção “Preencher Área” das “Configurações do Gráfico” e obter uma resposta visualmente intuitiva:



O programa mostra-nos a área cujo valor foi calculado, a vermelho. Verificamos que exatamente metade dessa área está acima da reta $y = 0$ (parte “positiva”) e a restante metade abaixo da reta (parte “negativa”). Por coincidência, estas áreas, devido a terem exatamente o mesmo valor mas “sinais” diferentes, anulam-se, e obtemos o deslocamento da partícula: zero.

Interpretando o problema através de uma perspetiva física, considerando que a função representada é a velocidade da partícula, percebemos que decorreu metade do tempo (até $t = 5$ segundos) com velocidade superior a zero, mas a diminuir constantemente (aceleração negativa constante), ou seja, a partícula afastava-se da origem mas cada vez mais lentamente, até que a sua velocidade se torna negativa e começa a deslocar-se na direção contrária, ou, por outras palavras, a aproximar-se da origem, onde acaba o seu movimento no nosso intervalo, em $t = 10$ segundos.

6.2 Integração

(sem contexto físico)

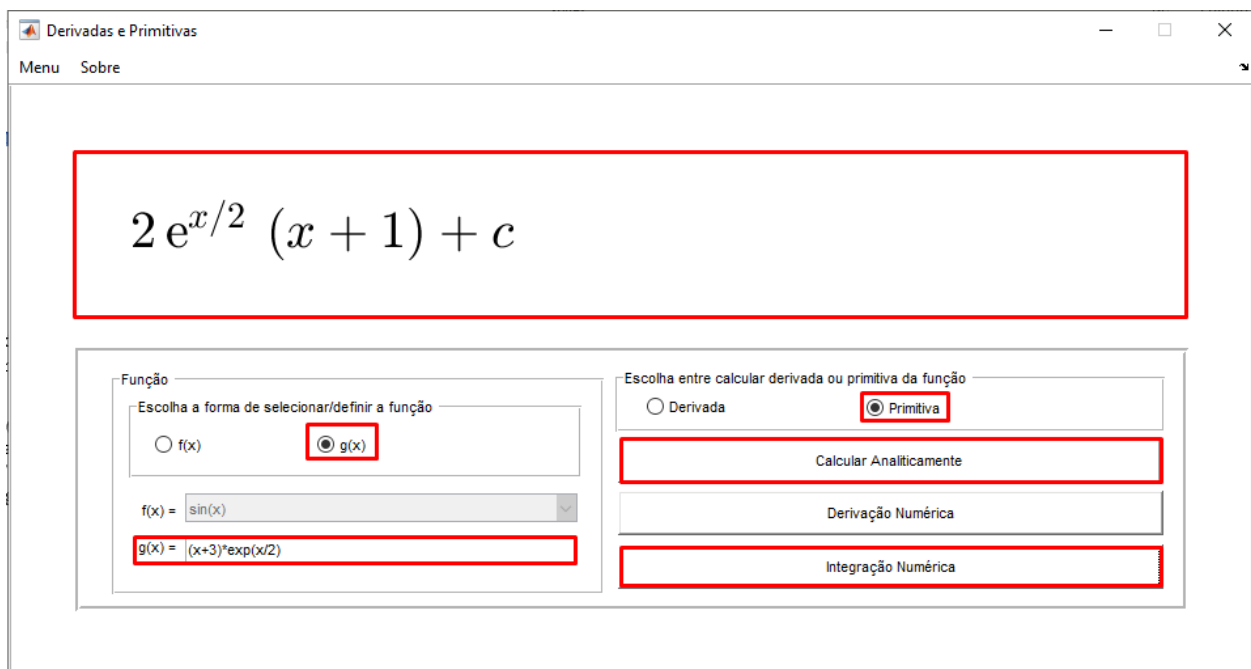
Consideremos, por exemplo, o seguinte integral:

$$\int_{-2}^5 (x + 3) e^{\frac{x}{2}} dx$$

Para calcular o seu valor analiticamente, teríamos de usar primitivação por partes, técnica que é um pouco trabalhosa. Para evitar isso, podemos facilmente usar a nossa GUI.

Como no exercício anterior, escolhemos a opção $g(x)$ e inserimos a nossa função, tendo em atenção que na sintaxe MATLAB a função $e^{\frac{x}{2}}$ designa-se por $\exp(x/2)$.

Depois, usamos novamente o botão “Integração Numérica”. Como curiosidade adicional, podemos também usar o botão “Calcular Analiticamente” (depois de seleccionar a opção “Primitiva”), que resolve a primitiva associada a este integral.



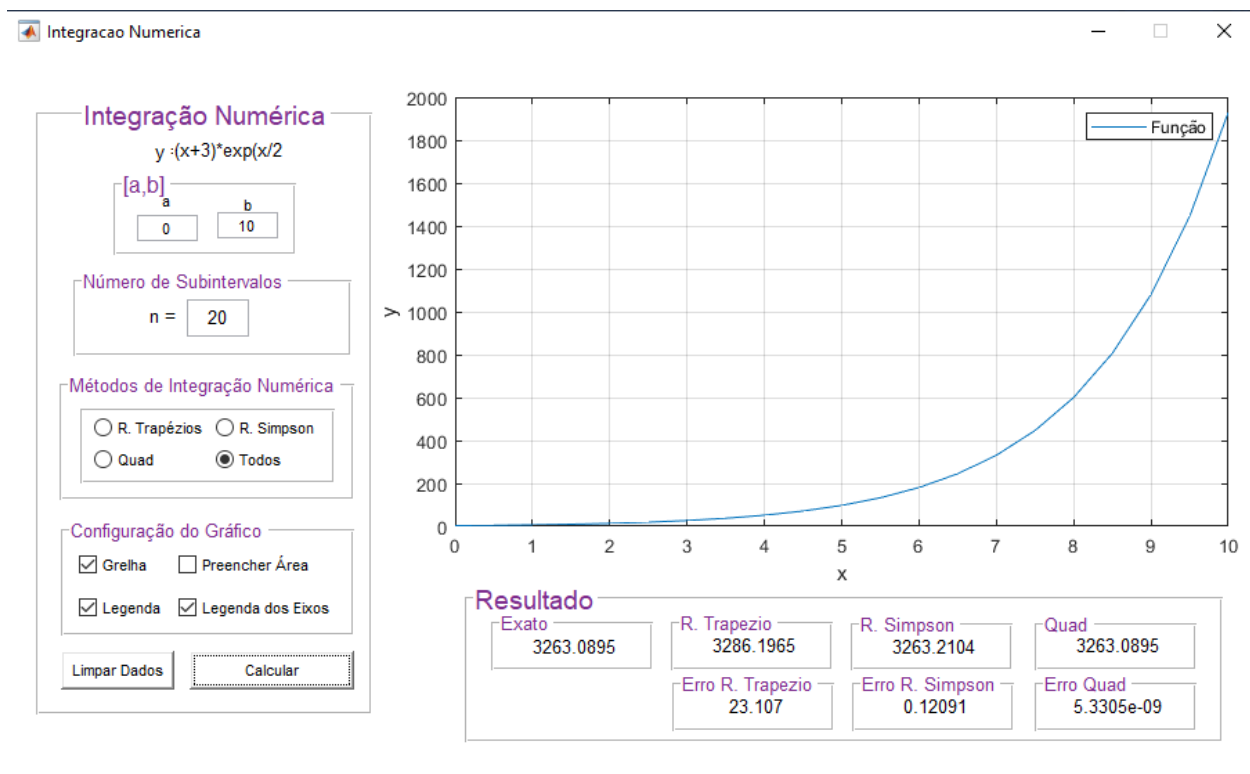
The screenshot shows a MATLAB GUI titled "Derivadas e Primitivas". The main display area shows the result of the integration: $2e^{x/2} (x + 1) + c$. Below this, there are two main sections. The left section, labeled "Função", allows the user to choose between "f(x)" and "g(x)". The "g(x)" option is selected. Below this, there are input fields for "f(x)" (containing "sin(x)") and "g(x)" (containing "(x+3)*exp(x/2)"). The right section, labeled "Escolha entre calcular derivada ou primitiva da função", allows the user to choose between "Derivada" and "Primitiva". The "Primitiva" option is selected. Below this, there are three buttons: "Calcular Analiticamente", "Derivação Numérica", and "Integração Numérica". The "Integração Numérica" button is highlighted with a red box.

Como podemos observar, a nova figura automaticamente associou a y a função que introduzimos.

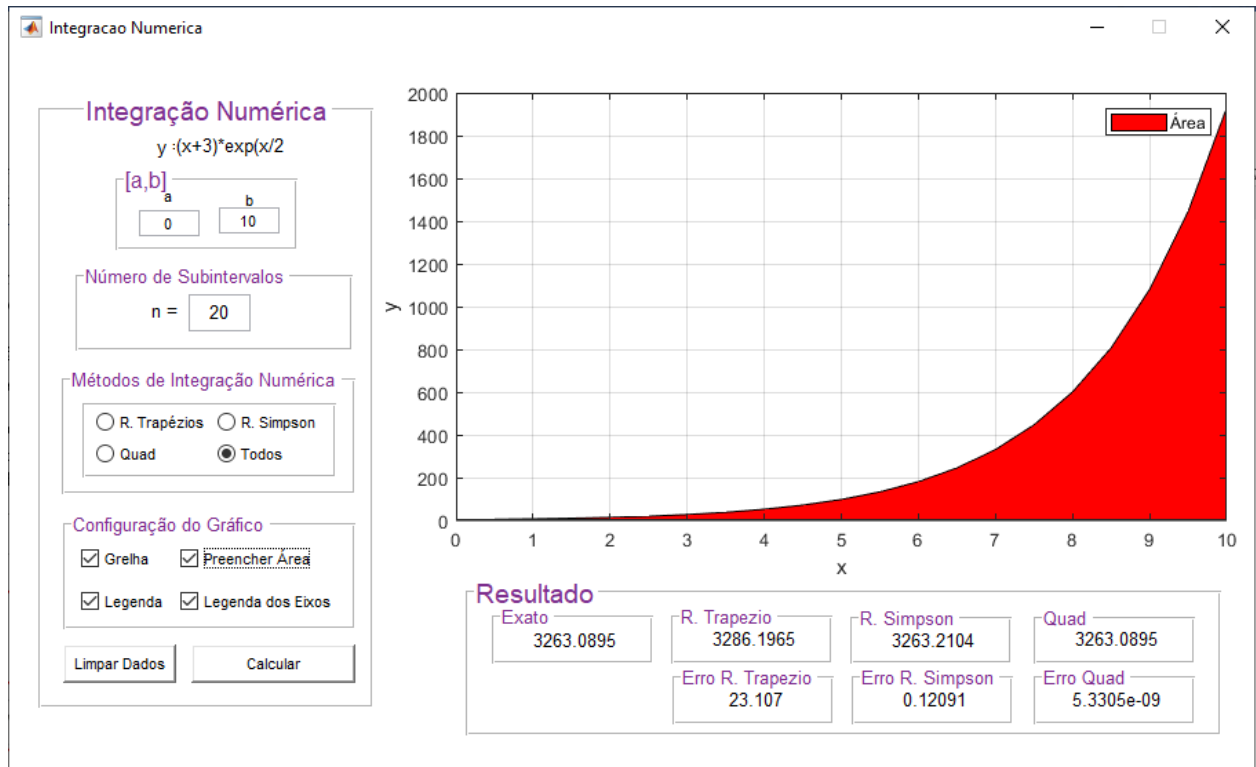
Escolhemos o intervalo pretendido. No nosso caso, temos o intervalo $[0,10]$ (ou seja, $a = 0$ e $b = 10$).

Temos agora que tomar a decisão de qual Regra usar para aproximar numericamente o nosso resultado (ou até mesmo escolher todos), assim como o número de subintervalos (quanto maior, menor o erro, ou seja, melhor a aproximação ao valor real). Existem também várias opções que podemos seleccionar, relativamente ao aspeto do gráfico.

Por fim, carregamos no botão “Calcular” e obtemos o seguinte:



Embora, neste caso, não tenhamos um contexto físico associado ao exercício, podemos seleccionar a opção “Preencher Área” das “Configurações do Gráfico” e obter a representação física deste integral definido:



6.3 Problemas de movimento (com derivadas)

Consideremos novamente a equação de velocidade usada no exercício do ponto 6.1:

$$v(t) = 5 - t$$

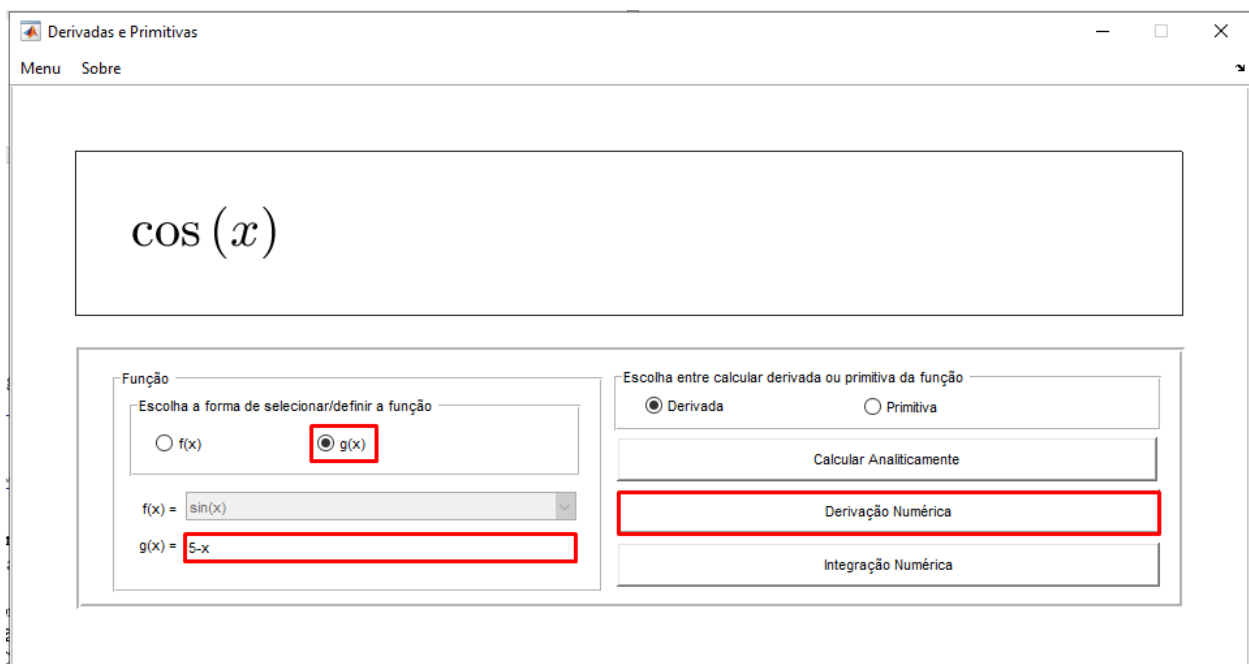
A velocidade trata-se da variação instantânea da posição da partícula, ou seja, em termos matemáticos, a função velocidade é a derivada da função posição. Por extensão, podemos querer a variação instantânea da velocidade da partícula, também chamada de aceleração.

A função aceleração pode ser obtida pelo cálculo da derivada da função velocidade, e é imediatamente obtida de maneira analítica:

$$a(t) = v'(t) = (5 - t)' = -1$$

Ou seja, a aceleração é constante (como comprovamos facilmente com uma análise ao gráfico da velocidade)

No entanto, vamos usar a nossa GUI, até porque esta nos permite vermos o gráfico da função aceleração. Para isso, como antes, escolhemos a opção $g(x)$ e inserimos a nossa função velocidade. De seguida, usamos o botão “Derivação Numérica”.

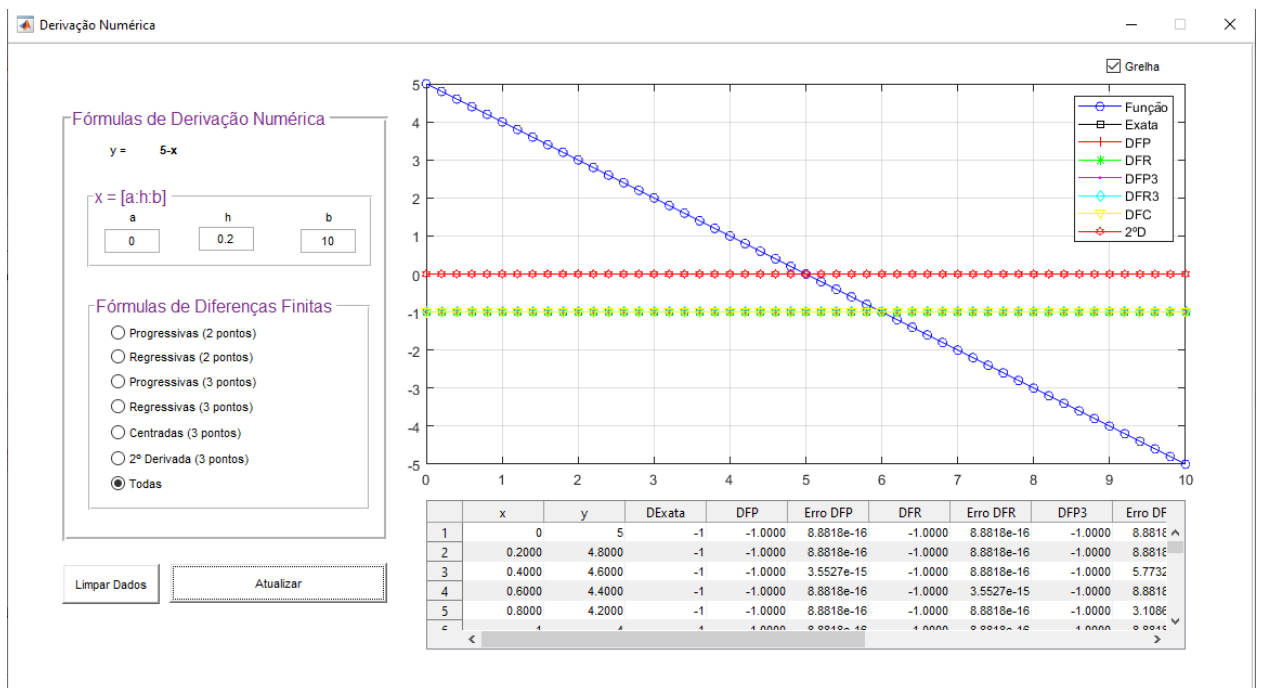


Como podemos observar, a nova figura automaticamente associou a y a função que introduzimos.

Escolhemos o intervalo pretendido. Neste exemplo, vamos usar o intervalo $[0,10]$ (ou seja, $a = 0$ e $b = 10$).

Temos agora que tomar a decisão de qual Fórmula usar para aproximar numericamente o nosso resultado (ou até mesmo escolher todos), assim como o tamanho h de cada subintervalo (quanto menor, menor o erro, ou seja, melhor a aproximação ao valor real).

Por fim, carregamos no botão “Atualizar” e obtemos o seguinte:



Observamos que, como esperado, a função é constante, de valor $a(t) = -1$.

NOTA: A GUI apresentou também a segunda derivada aproximada da função velocidade, ou seja, a derivada aproximada da função aceleração. Como a aceleração é constante, a derivada da constante (variação da aceleração) é zero.

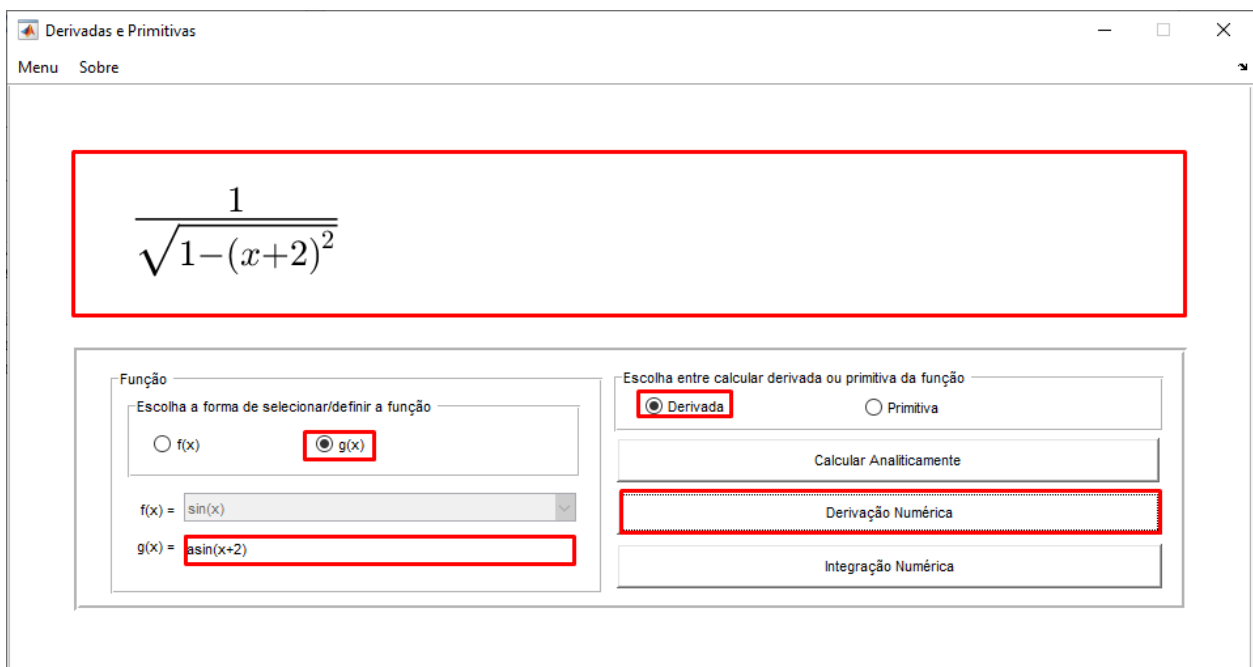
6.4 Derivação (sem contexto físico)

Consideremos, por exemplo, a seguinte função:

$$g(x) = \arcsen(x + 2)$$

Se quisermos evitar o cálculo analítico da sua derivada, podemos utilizar a nossa GUI. Como nos exercícios anteriores, escolhemos a opção $g(x)$ e inserimos a nossa função, tendo em atenção que na sintaxe MATLAB a função *arcsen* designa-se por *asin*.

Depois, usamos novamente o botão “Derivação Numérica”. Como curiosidade adicional, podemos também usar o botão “Calcular Analiticamente” (depois de selecionar a opção “Derivada”), que calcula analiticamente e automaticamente a derivada da função introduzida.



Como podemos observar, a nova figura automaticamente associou a y a função que introduzimos.

Neste caso, não podemos escolher qualquer intervalo $[a,b]$, pois o domínio da função não o permite. Vamos calcular o domínio da derivada da função introduzida, ou seja, o domínio de:

$$g'(x) = \frac{1}{\sqrt{1 - (x + 2)^2}}$$

O denominador não pode ser zero, e o radicando tem de ser maior ou igual a zero. Como $\sqrt{0} = 0$, ficamos com a restrição de que o radicando tem de ser maior que zero, ou seja:

$$D_{g'} = \{(x, y) \in R^2 : 1 - (x + 2)^2 > 0\}$$

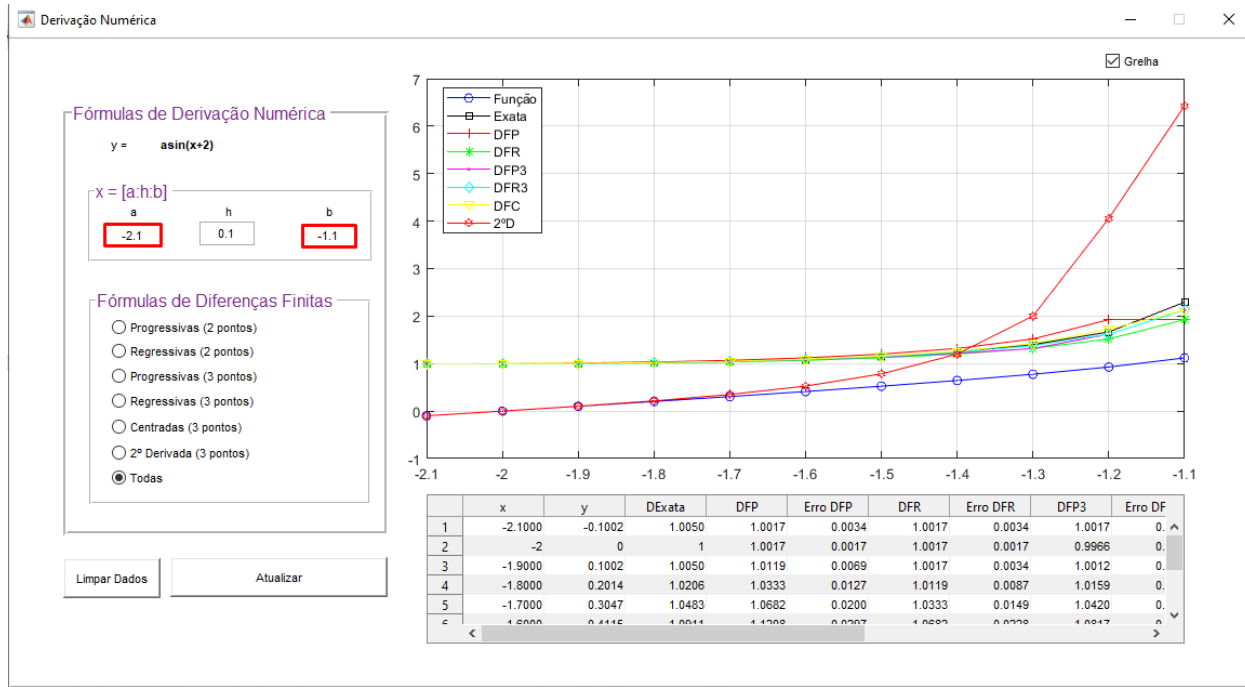
Ou seja, simplificando:

$$D_{g'} = \{(x, y) \in R^2 : -3 < x < -1\}$$

Portanto, podemos, por exemplo, escolher o intervalo $[-2.1, -1.1]$ (ou seja, $a = -2.1$ e $b = -1.1$).

Temos agora que introduzir o tamanho h de cada subintervalo (quanto menor for, menor o erro, ou seja, melhor a aproximação ao valor real), e tomar a decisão de qual Fórmula usar para aproximar numericamente o nosso resultado (ou escolher todos).

Por fim, carregamos no botão “Atualizar” e obtemos o seguinte:



NOTA: Se introduzirmos um intervalo fora do domínio da função, obtemos valores imaginários como resultado.

7. 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 densos de variadas áreas da ciência (e não só), onde, por vezes, necessitamos de calcular integrais analiticamente, algo que pode não ser possível.

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.

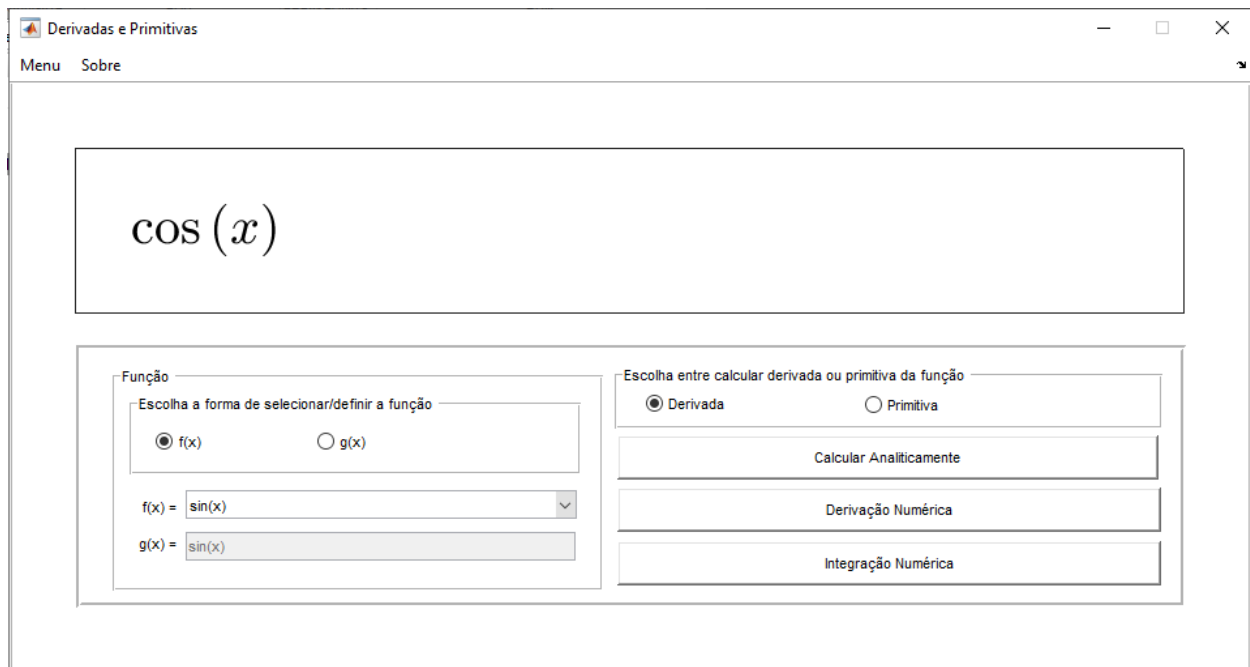
Relativamente à comparação entre os vários métodos da integração numérica, verificamos que o que apresenta menor erro é, consequentemente, melhor aproximação ao valor exato, é em regra geral, o Quad do MATLAB. O erro da Quad pode ser ajustado, mas na nossa GUI o seu valor é sempre aproximadamente $8.6 * 10^{-9}$.

Para valores elevados de n e grandes intervalos (ou seja, pequeno h), a Regra de Simpson é claramente melhor do que a Regra dos Trapézios, mas, para valores pequenos de n e pequenos intervalos (ou seja, grande h), não se nota tanto essa diferença, sendo que por vezes a Regra dos Trapézios obtém erros menores. Para valores suficientemente grandes de n , como o erro da Quad é constante, temos que a Regra de Simpson obtém erros menores, na nossa GUI.

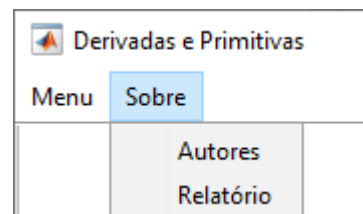
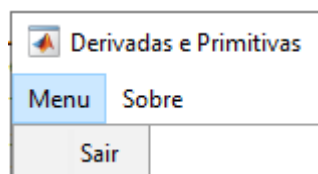
Em relação às fórmulas da derivação numérica, a comparação encontrada foi que a melhor aproximação ao valor real se origina a partir das fórmulas que utilizam 3 pontos, comparativamente às que apenas utilizam 2 pontos (que muitas vezes apresentam o dobro do erro, ou mais).

8. Anexos

Vista da GUI “Derivadas e Primitivas”



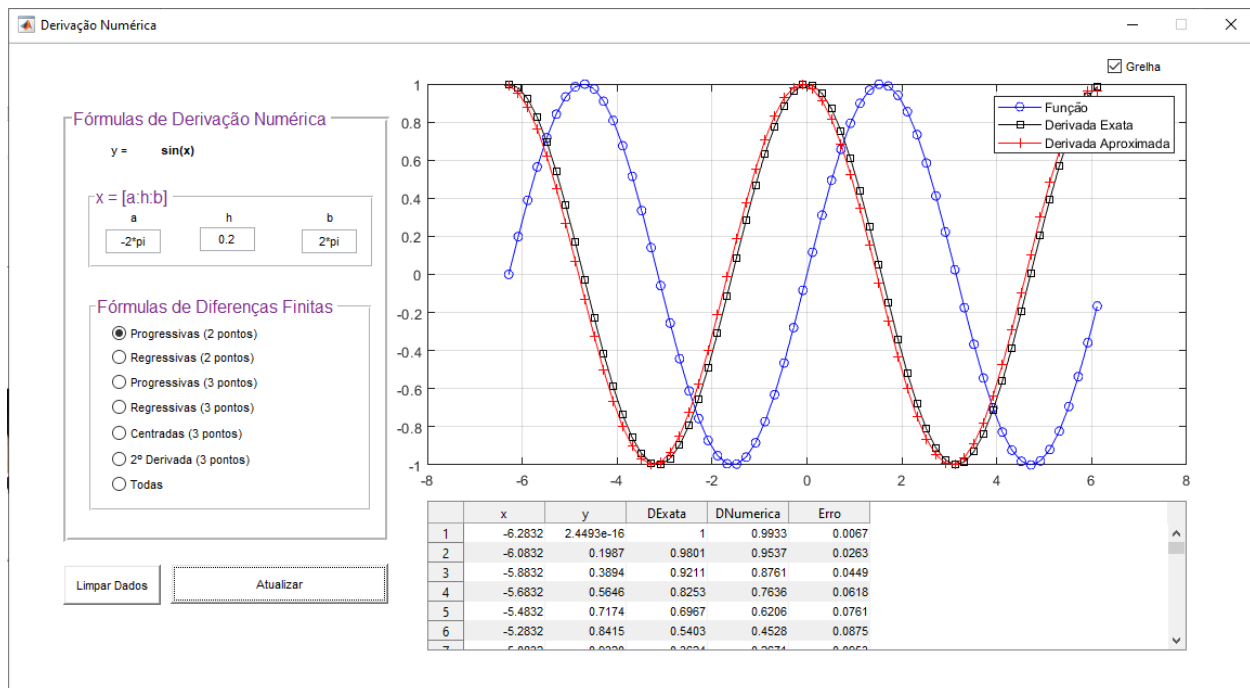
Menus



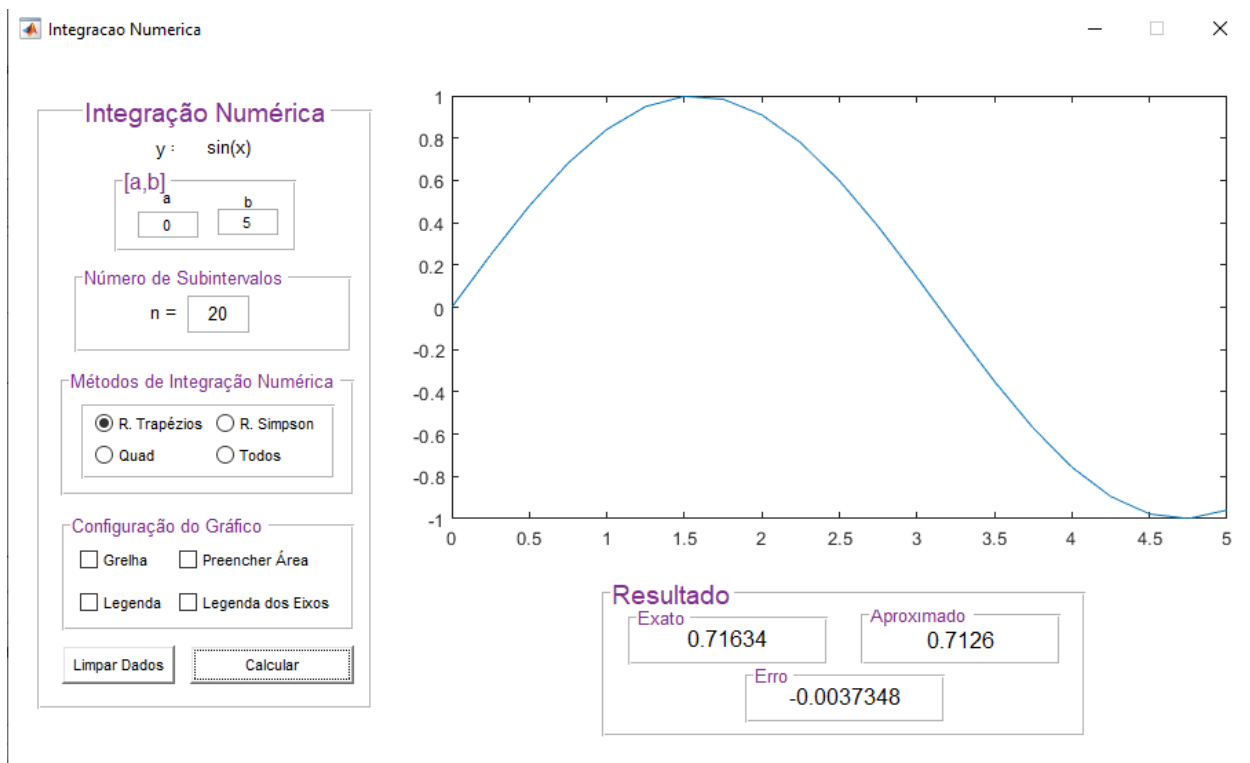
GUI Autores



Vista da GUI “Derivação Numérica”



Vista da GUI “Integração Numérica”



Breve Explicação da GUI “Derivação Numérica”

O valor da **função y**, assim como os valores do intervalo $[a, b]$ e o tamanho dos subintervalos (h) são enviados das edit texts correspondentes para a função do Fórmula seleccionada (neste caso, para a F.D.F. Progressivas (2 pontos)) utilizando o botão **Atualizar**.

Fórmulas das D.F.
Progressivas (2 pontos)
Regressivas (2 pontos)
Progressivas (3 pontos)
...

Os resultados da função são guardados numa tabela e enviados de volta para o programa, (assim como o(s) erro(s) da(s) fórmula(s) seleccionada(s) e o valor exato em cada ponto) que vai mostrar a tabela e o(s) gráfico(s) correspondente(s).

Fórmulas de Derivação Numérica

y =

x = [a:h:b]

a	h	b
<input type="text" value="-2*pi"/>	<input type="text" value="0.2"/>	<input type="text" value="2*pi"/>

Fórmulas de Diferenças Finitas

☒ Progressivas (2 pontos)

☐ Regressivas (2 pontos)

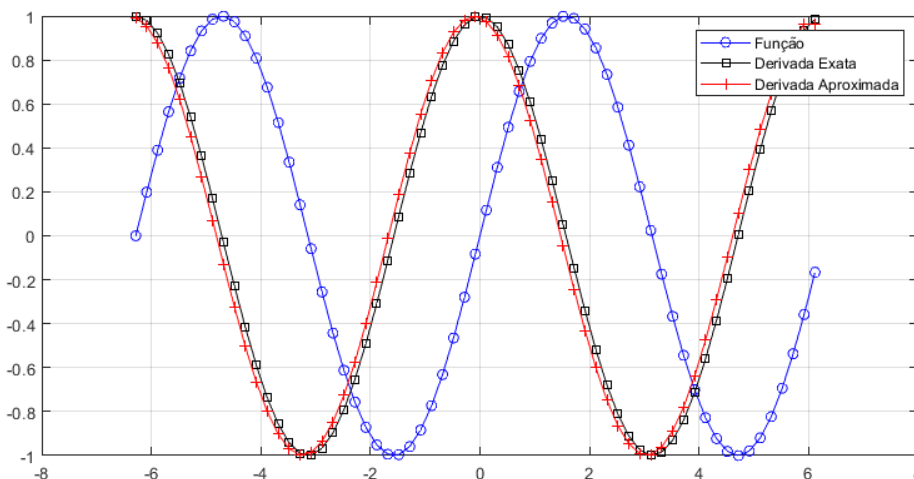
☐ Progressivas (3 pontos)

☐ Regressivas (3 pontos)

☐ Centradas (3 pontos)

☐ 2º Derivada (3 pontos)

☐ Todas



	x	y	DExata	DNumerica	Erro
1	-6.2832	2.4493e-16	1	0.9933	0.0067
2	-6.0832	0.1987	0.9801	0.9537	0.0263
3	-5.8832	0.3894	0.9211	0.8761	0.0449
4	-5.6832	0.5646	0.8253	0.7636	0.0618
5	-5.4832	0.7174	0.6967	0.6206	0.0761
6	-5.2832	0.8415	0.5403	0.4528	0.0875
7	-5.0832	0.9333	0.3674	0.2674	0.0659

Breve Explicação da GUI “Integração Numérica”

O valor da **função y**, assim como os valores do intervalo $[a, b]$ e o número de subintervalos (n) são enviados das edit texts correspondentes para a função do Método selecionado (neste caso, para a Regra dos Trapézios) utilizando o botão **Calcular**.

Fórmulas das D.F.
Regra dos Trapézios
Regra de Simpson
Quad
Todos

Integração Numérica

y :

[a,b]

a b

Número de Subintervalos

n =

Métodos de Integração Numérica

☒ R. Trapézios ☐ R. Simpson

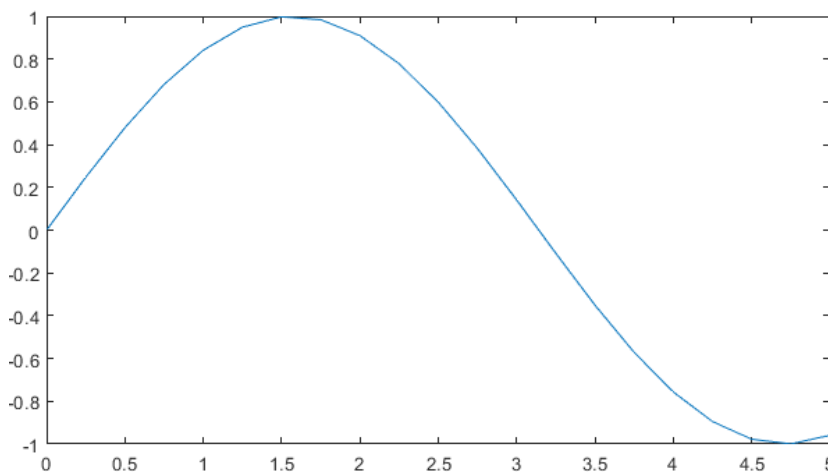
☐ Quad ☐ Todos

Configuração do Gráfico

☐ Grelha ☐ Preencher Área

☐ Legenda ☐ Legenda dos Eixos

Os resultados da função são guardados em static texts, assim como o erro do(s) método(s) selecionado(s), e enviados de volta para o programa, que vai mostrar o(s) resultado(s) aproximado(s), o resultado exato e o gráfico da função introduzida.



Resultado

Exato	Aproximado
<input type="text" value="0.71634"/>	<input type="text" value="0.7126"/>
Erro	
<input type="text" value="-0.0037348"/>	

9. Bibliografia

- <http://educacao.globo.com/fisica/assunto/mecanica/analise-grafica-dos-movimentos.html>
- <http://www.mat.uc.pt/~alma/aulas/matcomp/documentos/MatlabModuloAvancado.pdf>
- http://www.univasf.edu.br/~jorge.cavalcanti/8CN_integracao.pdf
- https://en.wikipedia.org/wiki/Simpson%27s_rule
- https://en.wikipedia.org/wiki/Trapezoidal_rule
- <https://moodle.isec.pt/moodle/mod/forum/discuss.php?d=23998>
- <https://pt.khanacademy.org/math/ap-calculus-ab/ab-applications-of-integration-new/ab-8-2/a/analyzing-problems-involving-definite-integrals-and-motion>
- [https://pt.symbolab.com/solver/function-domain-calculator/dom%C3%ADnio%20f%5Cleft\(x%5Cright\)%3D%5Cfrac%7B1%7D%7B%5Csqrt%7B1-%5Cleft\(x%2B2%5Cright\)%5E%7B2%7D%7D%7D](https://pt.symbolab.com/solver/function-domain-calculator/dom%C3%ADnio%20f%5Cleft(x%5Cright)%3D%5Cfrac%7B1%7D%7B%5Csqrt%7B1-%5Cleft(x%2B2%5Cright)%5E%7B2%7D%7D%7D)
- 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.ufrgs.br/reamat/CalculoNumerico/livro-sci/dn-diferencas_finitas.html
- https://www.ufsj.edu.br/portal2-repositorio/File/nepomuceno/mn/09MN_Derivacao.pdf

Consultados a 05/06/2020