Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

Métodos Numéricos para Derivação e Integração

Relatório de Licenciatura

Autores

Ana Rita Conceição Pessoa – 2023112690

João Francisco de Matos Claro – 2017010293

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Coimbra, maio e 2024

1 ÍNDICE

1.1 Índice de texto

1.2 Índice de figuras

2 LISTA DE SIGLAS, ACRÓNIMOS E SÍMBOLOS

2.1 Lista de siglas e acrónimos

GUI Graphical User Interface

2.2 Lista de símbolos

2.2.1 Exemplos de listas de símbolos

Alfabeto grego

Σ	Soma dos termos da série
ξ	Ponto dentro do intervalo de integração
Δ	Operador Laplaciano
∂	Derivada parcial

3 INTRODUÇÃO

Neste trabalho da unidade curricular de Análise Matemática II - Matemática Computacional, exploramos os conceitos de derivada e integral, além de criar uma ferramenta em MATLAB com uma interface gráfica (GUI) para calcular essas operações analiticamente e numericamente.

Sabemos que a derivada é a velocidade de mudança instantânea, ou seja, a velocidade com que uma função se altera em relação à sua variável independente num ponto específico. Esta é determinada ao calcular o limite da diferença das variações, à medida que o intervalo de variação da variável independente se aproxima de zero.

Adicionalmente, sabemos que a integral é uma operação que envolve a acumulação de quantidades e é o oposto da diferenciação. Calcular integrais é crucial em matemática e física, pois permite encontrar áreas sob curvas, volumes de sólidos com formas irregulares e muitas outras grandezas.

Os métodos de derivação e integração numérica incluem diferenças finitas, a segunda derivada, a regra dos Trapézios e a regra de Simpson, além da derivação e integração simbólica no MATLAB.

A "Máquina para Derivação e Integração" utiliza três segmentos numa GUI: dois para Derivação Numérica e Integração Numérica de funções reais de variável real, e outro para Derivação Numérica de funções reais de 2 variáveis reais.

O projeto é dividido em duas partes: métodos numéricos e simbólicos, e exemplos de aplicação. Visa também melhorar as competências de programação em MATLAB, utilizando as GUIs fornecidas pela disciplina, com arquivos disponíveis no moodle.

4 MÉTODOS NUMÉRICOS PARA DERIVAÇÃO

Entender e calcular derivadas é crucial em várias áreas, mas muitas vezes o processo é computacionalmente demorado. Para lidar com esta questão, surgiram as aproximações numéricas, como as fórmulas de diferenças finitas: progressivas, regressivas e centradas.

A Derivação Numérica é vital em situações onde não se dispõe da função completa, mas apenas de um conjunto de pontos que a representam, ou quando a função não é derivável em todo o seu domínio ou tem uma derivação não trivial.

O método baseia-se na redução de um intervalo h, aproximando-se assim do valor real da derivada. No entanto, mesmo com intervalos pequenos, o método pode ter um erro de arredondamento considerável. Para minimizar este erro, recorre-se à utilização de vários pontos.

O processo envolve começar com um conjunto de pontos que definem um intervalo $[a, b]$ e determinar a função f que os representa, interpolando este conjunto de pontos. Em seguida, é possível calcular a derivada da função f e aplicá-la em qualquer ponto dentro do intervalo $[a, b]$. Quanto mais pontos forem utilizados, mais preciso será o resultado.

4.1 Fórmulas de Diferenças Finitas em 2 pontos

Uma forma simples de aproximar a derivada é através do método das diferenças finitas. Uma diferença finita é uma expressão na forma $f(x+b) - f(x+a)$, que ao ser dividida por $(b-a)$ é conhecida como quociente de diferenças. A técnica das diferenças finitas consiste em estimar a derivada de uma função usando fórmulas discretas que necessitam apenas de um conjunto finito de pares ordenados, onde normalmente representamos $y_i = f(x)$.

4.1.1 Progressivas

Fórmula:

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

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 subintervalo (passo).

Algoritmo:

1. Alocar memória para x;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, $\boldsymbol{y}$ recebe o valor de $\boldsymbol{f(x)}$;
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)
      % Cria um vetor x que vai de a até b com incrementos de h. Este vetor
      contém os pontos nos quais queremos calcular a derivada (alocação de
      memória)
      x = a:h:b;
      % Número de pontos (tamanho do vetor de abcissas)
      n = length(x);

      % Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula
      os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’
      será ‘f(x)’.
      if nargin == 5
            y = f(x); % y é a função de f(x)
      end

      % Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor
      armazenará as derivadas aproximadas (alocação de memória)
      dydx = zeros(1,n);

      % Utiliza um loop for para calcular as derivadas aproximadas de f nos
      pontos de x, exceto no último ponto. A derivada é calculada usando a
      fórmula das diferenças finitas progressivas: (y(i+1) - y(i)) / h.
      for i = 1:n-1
            % Derivada (aproximada) de f no ponto atual
            dydx(i) = (y(i+1)-y(i))/h;
      end
      % Usar regressiva para calcular o último ponto. Isto é feito porque não há
      um ponto após ‘x(n)’ para aplicar a fórmula progressiva.
      dydx(n) = (y(n)-y(n-1))/h;
end
```

4.1.2 Regressivas

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

Onde:

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

Algoritmo:

1. Alocar memória para $\boldsymbol{x}$;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, $\boldsymbol{y}$ recebe o valor de $\boldsymbol{f(x)}$;
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)
	% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)
	x = a:h:b;
	% Número de pontos (tamanho do vetor de abcissas)
	n = length(x);
	% Se a função foi chamada com apenas cinco argumentos (sem o 'y'), calcula os valores de 'y' aplicando a função 'f' aos valores de 'x'. Ou seja, 'y' será 'f(x)'
	if nargin == 5
	y = f(x);
	end

	% Cria um vetor 'dydx' de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)
	dydx = zeros(1,n);
	% Para o primeiro ponto, a derivada é calculada usando a fórmula de diferenças finitas progressivas: '(y(2) - y(1)) / h'. Isto é necessário porque não há um ponto anterior a 'x(1)' para aplicar a fórmula regressiva
	dydx(1)=(y(2)-y(1))/h;

	% Utiliza um loop 'for' para calcular as derivadas aproximadas de 'f' nos pontos de 'x', começando do segundo ponto até o último ponto. A derivada é calculada usando a fórmula das diferenças finitas regressivas: '(y(i) - y(i-1)) / h'
	for i=2:n
		dydx(i) = (y(i)-y(i-1))/h;
	end
end
```

4.2 Fórmulas de Diferenças Finitas em 3 pontos

4.2.1 Progressivas

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

Onde:

- $f'(x_k)$ → Aproximação do valor da derivada no ponto de abcissa x_k;
- $f(x_k)$ → Valor da função na próxima 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 $\boldsymbol{x}$;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, y recebe o valor de $\boldsymbol{f(x)}$;
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] = NDerivadaDFP3P(~,f,a,b,h,y)

	% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)
	x = a:h:b;
	% Número de pontos (tamanho do vetor de abcissas)
	n = length(x);

	% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’
	if nargin == 5
		y = f(x);
	end

	% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)
	dydx = zeros(1,n);

	% Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, começando do segundo ponto até ao 3º ponto. A derivada é calculada usando a fórmula das diferenças finitas progressiva de 3 pontos: ‘(-3*y(i)+4*y(i+1)-y(i+2))/(2*h);’
	for i=2:n-2
		dydx(i) = (-3*y(i)+4*y(i+1)-y(i+2))/(2*h);
	end

	% Cálculo da derivada no penúltimo ponto: Calcula a derivada no penúltimo ponto usando uma fórmula de diferenças finitas regressivas ajustada para garantir a precisão:
	dydx(n-1)=(y(n-3) - 4*y(n-2) + 3*y(n-1))/(2*h);
	% Cálculo da derivada no último ponto: Calcula a derivada no último ponto usando uma fórmula de diferenças finitas regressivas ajustada:
	dydx(n)=(y(n-2) - 4*y(n-1) + 3*y(n))/(2*h);
end
```

4.2.2 Regressivas

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

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 na próxima abcissa atual;
- h → Valor de cada subintervalo (passo).

Algoritmo:

1. Alocar memória para $\boldsymbol{x}$;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, y recebe o valor de $\boldsymbol{f(x)}$;
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] = NDerivadaDFR3(~,f,a,b,h,y)
	% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)
	x = a:h:b;
	% Número de pontos (tamanho do vetor de abcissas)
	n = length(x);

	% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’
	if nargin == 5
		y = f(x);
	end

	% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)
	dydx = zeros(1,n);

	% Cálculo da derivada no primeiro ponto: Calcula a derivada no primeiro ponto usando uma fórmula de diferenças finitas progressivas ajustada. Isto é necessário porque não há pontos anteriores a ‘x(1)’ para aplicar a fórmula regressiva.
	dydx(1)=(-3*y(1) + 4*y(2) - y(3))/(2*h);

	% Cálculo da derivada no segundo ponto: Calcula a derivada no segundo ponto usando uma fórmula de diferenças finitas progressivas ajustada
	dydx(2)=(-3*y(2) + 4*y(3) - y(4))/(2*h);

	% Cálculo das derivadas para os pontos restantes (método das diferenças finitas regressivas de 3 pontos): Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, começando do terceiro ponto até o último ponto. A derivada é calculada usando a fórmula das diferenças finitas regressivas de 3 pontos:
	for i=3:n
		dydx(i)=(y(i-2) - 4*y(i-1) + 3*y(i))/(2*h);
	end
end
```

4.2.3 Centradas

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

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 $\boldsymbol{x}$;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, $\boldsymbol{y}$ recebe o valor de $\boldsymbol{f(x)}$;
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] = NDerivadaDFC3P(~,f,a,b,h,y)
	% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)
	x = a:h:b;
	% Número de pontos (tamanho do vetor de abcissas)
	n = length(x);

	% Se a função foi chamada com apenas cinco argumentos (sem o ‘y’), calcula os valores de ‘y’ aplicando a função ‘f’ aos valores de ‘x’. Ou seja, ‘y’ será ‘f(x)’
	if nargin == 5
		y = f(x);
	end

	% Cria um vetor ‘dydx’ de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)
	dydx = zeros(1,n);


	% Cálculo da derivada no primeiro ponto: Calcula a derivada no primeiro ponto usando uma fórmula de diferenças finitas progressivas ajustada. Isto é necessário porque não há pontos anteriores a ‘x(1)’ para aplicar a fórmula centrada
	dydx(1)=(-3*y(1) + 4*y(2) - y(3))/(2*h);

	% Cálculo das derivadas para os pontos intermediários (método das diferenças finitas centradas de 3 pontos): Utiliza um loop ‘for’ para calcular as derivadas aproximadas de ‘f’ nos pontos de ‘x’, do segundo ponto até o penúltimo ponto. A derivada é calculada usando a fórmula das diferenças finitas centradas de 3 pontos
	for i=2:n-1
		dydx(i)=(y(i+1)-y(i-1))/(2*h);
	end

	% Cálculo da derivada no último ponto: Calcula a derivada no último ponto usando uma fórmula de diferenças finitas regressivas ajustada
	dydx(n)=(y(n-2) - 4*y(n-1) + 3*y(n))/(2*h);
end
```

4.2.4 Segunda Derivada

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

Onde:

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

Algoritmo:

1. Alocar memória para $\boldsymbol{x}$;
2. Definir o número de pontos (***n***);
3. Se forem recebidos 4 elementos, $\boldsymbol{y}$ recebe o valor de $\boldsymbol{f(x)}$;
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 iésima 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] = SegundaDerivada(~,f,a,b,h,y)
	% Cria um vetor x que vai de a até b com incrementos de h. Este vetor contém os pontos nos quais queremos calcular a derivada (alocação de memória)
	x = a:h:b;
	% Número de pontos (tamanho do vetor de abcissas)
	n = length(x);

	% Se a função foi chamada com apenas cinco argumentos (sem o 'y'), calcula os valores de 'y' aplicando a função 'f' aos valores de 'x'. Ou seja, 'y' será 'f(x)'
	if nargin == 5
		y = f(x);
	end

	% Cria um vetor 'dydx' de zeros com o mesmo comprimento que x. Este vetor armazenará as derivadas aproximadas (alocação de memória)
	dydx = zeros(1,n);

	% Cálculo das derivadas primeiras nos primeiros três pontos: Calcula as primeiras derivadas nos três primeiros pontos usando uma fórmula de diferenças finitas progressivas ajustada
	dydx1=(-3*y(1) + 4*y(2) - y(3))/(2*h);
	dydx2=(-3*y(2) + 4*y(3) - y(4))/(2*h);
	dydx3=(-3*y(3) + 4*y(4) - y(5))/(2*h);

	% Cálculo da segunda derivada no primeiro ponto: Calcula a segunda derivada no primeiro ponto utilizando uma combinação linear das primeiras derivadas calculadas anteriormente
	dydx(1)=(-3*dydx1 + 2*dydx2 - 3*dydx3)/(2*h);

	% Cálculo das segundas derivadas para os pontos intermediários (método das diferenças finitas centradas): Utiliza um loop 'for' para calcular as segundas derivadas aproximadas de 'f' nos pontos de 'x', do segundo ponto até o penúltimo ponto. A segunda derivada é calculada usando a fórmula das diferenças finitas centradas
	for i=2:n-1
		dydx(i)=(y(i+1)-2*y(i)-y(i-1))/(2*h);
	end

	% Cálculo das derivadas primeiras nos últimos três pontos: Calcula as primeiras derivadas nos três últimos pontos usando uma fórmula de diferenças finitas regressivas ajustada
	dydx1=(-3*y(n-4) + 4*y(n-3) - y(n-2))/(2*h);
	dydx2=(-3*y(n-3) + 4*y(n-2) - y(n-1))/(2*h);
	dydx3=(-3*y(n-2) + 4*y(n-1) - y(n))/(2*h);

	% Cálculo da segunda derivada no último ponto: Calcula a segunda derivada no último ponto utilizando uma combinação linear das primeiras derivadas calculadas anteriormente
	dydx(n)=(-3*dydx1 + 2*dydx2 - 3*dydx3)/(2*h);
end
```

4.2 Funções simbólicas no MatLab

O MATLAB, além de permitir o cálculo numérico, oferece também funcionalidades de cálculo simbólico através da sua toolbox de matemática simbólica, conhecida como "Symbolic Math Toolbox". Esta toolbox possibilita a realização de cálculos de várias naturezas, incluindo:

Cálculo Numérico: diferenciação, integração, determinação de limites, somatórios, séries de Taylor, entre outros.

Álgebra linear: operações como a inversa de uma matriz, cálculo de determinantes, obtenção de valores próprios, etc.

Simplificação de expressões algébricas.

Obtenção de soluções analíticas de equações algébricas e diferenciais.

Transformadas: Laplace, Z, Fourier, entre outras.

Esta ferramenta é extremamente útil para a resolução de problemas matemáticos complexos de forma simbólica, ampliando significativamente as capacidades do MATLAB no campo da matemática e da engenharia.

4.2.1 Função *diff* ()

Sintaxe:

$$\boldsymbol{Y = diff(X)}$$
$$\boldsymbol{Y = diff(X, n)}$$
$$\boldsymbol{Y = diff(X, n, dim)}$$

Onde:

$\boldsymbol{X}$ → Array de Entrada que pode ser um vetor, matriz, array multidimensional, tabela ou tabela de horários, com diversos tipos de dados permitidos.

$\boldsymbol{n}$ → Ordem da Diferença e, se não for fornecido, o valor padrão é 1; se $\boldsymbol{n}$ for maior que a dimensão, o comportamento varia.

$\boldsymbol{dim}$ → Dimensão ao longo da qual calcular a diferença; se não for especificado, a operação é realizada na primeira dimensão com tamanho maior que 1.

Considerando um array de entrada bidimensional 'p-por-m', 'A':

- ***diff(A,1,1)*** trabalha nos elementos sucessivos nas colunas de **A** e devolve uma matriz de diferenças de tamanho **(p-1)-por-m**.
- ***diff(A,1,2)*** trabalha nos elementos sucessivos nas filas de **A** e devolve uma matriz de diferenças de **tamanho p-por-(m-1).**

Para $\boldsymbol{Y = diff(X)}$ calcula as diferenças entre elementos adjacentes de \`X\` ao longo da primeira dimensão do array cujo tamanho não seja 1:

- Se \`X\` for um vetor de comprimento \`m\`, então $\boldsymbol{Y = diff(X)}$ devolve um vetor de comprimento \`m-1\`. Os elementos de \`Y\` são as diferenças entre os elementos adjacentes de \`X\`.

  Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]

- Se \`X\` for uma matriz não vazia e não vetorial de tamanho \`p-por-m\`, então $\boldsymbol{Y = diff(X)}$ devolve uma matriz de tamanho \`(p-1)-por-m\`, cujos elementos são as diferenças entre as linhas de \`X\`.

  Y = [X(2,:)-X(1,:); X(3,:)-X(2,:); ... X(p,:)-X(p-1,:)]

- Se \`X\` for uma matriz vazia de 0-por-0, então $\boldsymbol{Y = diff(X)}$ devolve uma matriz vazia de 0-por-0.

- Se \`X\` for uma tabela ou tabela de horários de tamanho \`p-por-m\`, então $\boldsymbol{Y = diff(X)}$ devolve uma tabela ou tabela de horários de tamanho \`(p-1)-por-m\`, cujos elementos são as diferenças entre as linhas de \`X\`. Se \`X\` for uma tabela ou tabela de horários de tamanho 1-por-m, então o tamanho de \`Y\` é 0-por-m.

Para $\boldsymbol{Y = diff(X, n)}$ calcula a diferença de ordem $\boldsymbol{n}$ aplicando o operador diff(X) recursivamente $\boldsymbol{n}$ vezes. Na prática, isto significa que diff(X,2) é o mesmo que diff(diff(X)).

Para $\boldsymbol{Y = diff(X, n, dim)}$ é a diferença de ordem $\boldsymbol{n}$ calculada ao longo da dimensão especificada por ***dim***. O parâmetro ***dim*** é um escalar inteiro positivo.

Exemplos

Diferenças Entre Elementos de um Vetor

Cria um vetor e calcula as diferenças entre os elementos.

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

```
Y = 1×7
     0     1     1     2     3     5     8
```

Diferenças Entre Linhas de uma Matriz

Cria uma matriz 3x3 e calcula a primeira diferença entre as linhas.

```
X = [1 1 1; 5 5 5; 25 25 25];
Y = diff(X)
```

```
Y = 2×3
     4     4     4
    20    20    20
```

Diferenças Múltiplas

Cria um vetor e calcula a diferença de segunda ordem entre os elementos.

```
X = [0 5 15 30 50 75 105];
Y = diff(X,2)
```

```
Y = 1×5
     5     5     5     5     5
```

Diferenças Entre Colunas de uma Matriz

Cria uma matriz 3x3 e calcula a diferença de primeira ordem entre as colunas.

```
X = [1 3 5;7 11 13;17 19 23];
Y = diff(X,1,2)
```

```
Y = 3×2
     2     2
     4     2
     2     4
```

Aproximação de Derivadas com diff

Usa a função diff para aproximar derivadas parciais com a sintaxe Y = diff(f)/h, onde f é um vetor de valores da função avaliados sobre algum domínio X, e h é um tamanho de passo apropriado.

Por exemplo, a primeira derivada de sin(x) em relação a x é cos(x), e a segunda derivada em relação a x é -sin(x). Podes usar diff para aproximar estas derivadas.

```
h = 0.001;       % passo
X = -pi:h:pi;    % domínio
f = sin(X);      % função
Y = diff(f)/h;   % 1ª derivada
Z = diff(Y)/h;   % 2ª derivada
plot(X(:,1:length(Y)),Y,'r',X,f,'b', X(:,1:length(Z)),Z,'k')
```

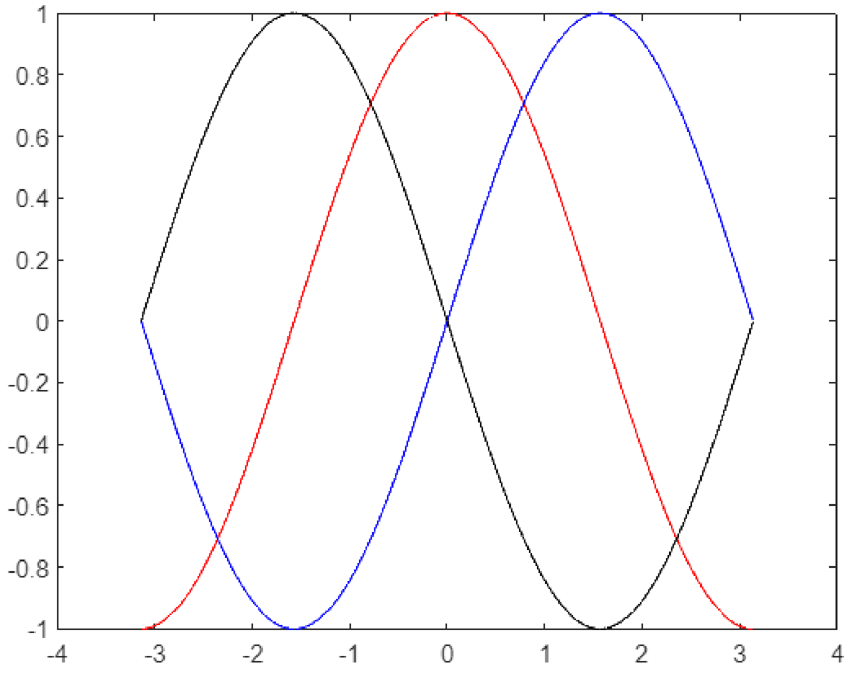


Figura 1 - Aproximação de Derivadas com diff

Neste gráfico, a linha azul corresponde à função original, sin. A linha vermelha corresponde à primeira derivada calculada, cos, e a linha preta corresponde à segunda derivada calculada, -sin.

Diferenças Entre Valores de Data e Hora

Cria uma sequência de valores de data e hora igualmente espaçados e encontra as diferenças de tempo entre eles.

```
t1 = datetime('now');
t2 = t1 + minutes(5);
t = t1:minutes(1.5):t2
```

```
t = 1×4 datetime
27-May-2024 11:11:1127-May-2024 11:12:4127-May-2024 11:14:1127-May-2024 11:15:41
```

```
dt = diff(t)
```

```
dt = 1×3 duration
00:01:30    00:01:30    00:01:30
```

4.2.2 Função *int ()*

Integral Indefinida de Expressão Univariada

F = int(expr) calcula a integral indefinida de expr. int usa a variável de integração padrão determinada por symvar(expr,1). Se expr for uma constante, então a variável de integração padrão é x.

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

F =

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

Integrais Indefinidas de Função Multivariada

F = int(expr,var) calcula a integral indefinida de expr em relação à variável escalar simbólica var.

```
% Define uma função multivariada com as variáveis x e z
syms f(x,z)
f(x,z) = x/(1+z^2);
% Encontra as integrais indefinidas da expressão multivariada em relação às variáveis x e z
Fx = int(f,x)
```

Fx(x, z) =

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

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

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

```
% Se for especificada a variável de integração, então a função int utiliza a primeira variável retornada por symvar como a variável de integração
Fz(x, z) = atan(z)
```

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

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

var =

x

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

F(x, z) =

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

Integrais Definidas de Expressões Simbólicas

F = int(expr,a,b) calcula a integral definida de expr de a para b. int usa a variável de integração padrão determinada por symvar(expr,1). Se expr for uma constante, então a variável de integração padrão é x.

int(expr,[a b]) é equivalente a **int(expr,a,b)**

```
% Integra uma expressão simbólica de 0 a 1
syms x
expr = x*log(1+x);
F = int(expr,[0 1])
```

F =

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

```
% Integra outra expressão de sin(t) a 1
syms t
F = int(2*x,[sin(t) 1])
```

F = $\cos(t)^2$

```
% Quando int não consegue calcular o valor de uma integral definida, aproxima numericamente a integral usando vpa.
syms x
f = cos(x)/sqrt(1 + x^2);
Fint = int(f,x,[0 10])
```

Fint =

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

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

Fvpa =

0.37570628299079723478493405557162

```
% Para aproximar integrais diretamente, utiliza vpaintegral em vez de vpa. A função vpaintegral é mais rápida e fornece controlo sobre as tolerâncias de integração.
Fvpaint = vpaintegral(f,x,[0 10])
```

Fvpaint =

0.375706

Integrais dos Elementos de uma Matriz

F = int(expr,var,a,b) calcula a integral definida de expr em relação à variável escalar simbólica var de a para b.

int(expr,var,[a b]) é equivalente a **int(expr,var,a,b)**

```
% Integra uma expressão simbólica de 0 a 1
syms x
expr = x*log(1+x);
F = int(expr,[0 1])
```

F =

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

```
% Integra outra expressão de sin(t) a 1
syms t
F = int(2*x,[sin(t) 1])
```

F = $\cos(t)^2$

```
% Quando int não consegue calcular o valor de uma integral definida, aproxima numericamente a integral usando vpa
syms x
f = cos(x)/sqrt(1 + x^2);
Fint = int(f,x,[0 10])
```

Fint =

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

```
% Para aproximar integrais diretamente, utiliza vpaintegral em vez de vpa. A função vpaintegral é mais rápida e fornece controlo sobre as tolerâncias de integração
Fvpa = vpa(Fint)
```

Fvpa =

0.37570628299079723478493405557162

5 MÉTODOS NUMÉRICOS PARA INTEGRAÇÃO

Na Matemática, muitos algoritmos visam aproximar o valor de uma integral definida sem usar uma expressão analítica para a sua primitiva. Estes métodos geralmente seguem três fases:

1. Decomposição do domínio em subintervalos.
2. Integração aproximada da função em cada subintervalo.
3. Soma dos resultados numéricos obtidos.

A integração numérica é necessária quando não é possível encontrar uma primitiva explícita para uma função, quando a primitiva é demasiado complexa para ser avaliada ou quando não se dispõe de uma expressão analítica para a função, mas conhecem-se os seus valores em pontos específicos do domínio. Durante a Integração Numérica de uma função $\boldsymbol{f(x)}$ num intervalo $[a, b]$, a área sob a curva é calculada utilizando interpolação polinomial para aproximar a função e obter um polinómio $p_n(x)$. Este método básico é conhecido como quadratura numérica:

$$\int_{a}^{b} f(x)dx \cong \sum_{i=0}^{n} \alpha_i f(x_i)$$

Existem vários métodos numéricos para a integração, sendo a **Regra dos Trapézios** e a **Regra de Simpson** os mais destacados e utilizados.

- **Regra dos Trapézios:** Aproxima a função $f(x)$ por um polinómio de ordem 1 (reta), resultando num valor aproximado para a área de uma determinada região limitada $[a, b]$ com a forma de um trapézio.

- **Regra de Simpson:** Oferece uma melhor aproximação ao considerar um polinómio de 2º grau (parábola) em vez de um polinómio de 1º grau. Este método é especialmente eficaz quando o intervalo de integração $[a, b]$ é pequeno. A Regra de Simpson composta divide o intervalo de integração em subintervalos menores e aplica a fórmula de Simpson a cada um desses subintervalos, somando os resultados para obter uma aproximação mais precisa.

5.1 Regra dos Trapézios Simples

Fórmula:

$$\boldsymbol{T(f) = I_1(f) = (f(a) + f(b)) \cdot \frac{b-a}{2}}$$

Onde:

- $\boldsymbol{T(f) = I_1(f)}$ → Representa a aproximação da integral da função f no intervalo $[a, b]$ usando a Regra dos Trapézios Simples;
- $\boldsymbol{f(a)}$ → É o valor da função f no ponto a, ou seja, o valor da função no extremo esquerdo do intervalo de integração;
- $\boldsymbol{f(b)}$ → É o valor da função f no ponto b, ou seja, o valor da função no extremo direito do intervalo de integração;
- $\boldsymbol{b - a}$ → Representa a largura do intervalo de integração. É a diferença entre os limites superior e inferior do intervalo $[a, b]$;
- $\boldsymbol{\frac{f(a)+f(b)}{2}}$ → Média dos valores da função f nos pontos a e b. Esta média é utilizada para calcular a altura do trapézio na aproximação da área sob a curva.
- $\boldsymbol{(b - a)}$ → Multiplicado pela média $\boldsymbol{\frac{f(a)+f(b)}{2}}$, calcula a área do trapézio que aproxima a integral da função f no intervalo$[a, b]$.

Fórmula do erro para a regra dos trapézios simples:

$$\boldsymbol{E(f) = \frac{(b-a)^3}{12} f''(\xi)} \text{ , com } \boldsymbol{\xi} \in \,]a, b[$$

Onde:

- $\boldsymbol{E(f)}$ → Erro da aproximação da integral da função f utilizando a Regra do Trapézio Simples;
- $\boldsymbol{(b - a)^3}$ → Indica que o erro é proporcional ao comprimento do intervalo elevado ao cubo. Quanto maior o intervalo, maior será o erro, assumindo que a segunda derivada não varia significativamente.
- $\boldsymbol{f''(\xi)}$ →A segunda derivada dá uma medida de quanto a função está curvada. O erro da aproximação aumenta com a magnitude da segunda derivada, pois funções mais curvas são mais difíceis de aproximar com segmentos de linha reta (o que a Regra do Trapézio faz).

Algoritmo:

1. Calcular os valores da função $f(x)$ nos extremos do intervalo:
   - $f(a)$
   - $f(b)$
2. Calcular a largura do intervalo:
   - $h = b - a$
3. Aplicar a fórmula da Regra do Trapézio:
   - $T(f) = I_1(f) = (f(a) + f(b)) \cdot \frac{b-a}{2}$

5.2 Regra dos Trapézios Composta

Fórmula:

$$\int_a^b f(x)dx \approx \frac{h}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]$$

Onde:

- a e b → Os limites de integração inferior e superior, respectivamente;
- $f(x)$ → A função que está a ser integrada;
- dx → O elemento de variação infinitesimal na variável de integração;
- h → O tamanho do intervalo $h = \frac{b-a}{n}$, onde *n* é o número de subintervalos;
- $x_0, x_1, \dots, x_n$ → Os pontos de integração dentro do intervalo $[a, b]$.
- $f(x_0), f(x_1), \dots, f(x_n)$ → Os valores da função $f(x)$ avaliados nos pontos de integração.

Fórmula do |erro| para a regra dos trapézios:

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

Onde:

- $e_T(f)$ → O erro absoluto na aproximação da integral usando a regra dos trapézios;
- $|e_T(f)|$ → O valor absoluto do erro;
- a e b → Os limites de integração inferior e superior, respectivamente h;
- h → O tamanho do intervalo de integração;
- M_2 → O valor máximo da segunda derivada da função $f(x)$ no intervalo $[a, b]$.

Algoritmo:

4. Ler ***n***
5. Ler ***(a,b)***
6. $h \leftarrow \frac{b-a}{n}$
7. $x \leftarrow a$
8. $s \leftarrow 0$
9. Para $\boldsymbol{i}$ de $\boldsymbol{1}$ até $\boldsymbol{n-1}$ fazer
   - $x \leftarrow x + h$
   - $s \leftarrow s + f(x)$
10. $T \leftarrow \frac{h}{2}(f(a) + 2 \cdot s + f(b))$
11. Escrever $\boldsymbol{T}$

Função em MatLab:

```
function Tsol = RTrapezios(~,f,a,b,h)
      % Esta linha cria um vetor t com valores espaçados uniformemente de a a b
      com um passo
      t = a:h:b;
      % Inicializa a variável s como zero, que será usada para acumular a soma
      dos valores da função f(x)
      s = 0;
      % Calcula o número de elementos no vetor t, que é o número total de
      subintervalos
      n = length(t);

      % Inicia um loop que percorre todos os subintervalos, exceto o último
      for i=1:n-1
            % Para cada subintervalo, calcula o valor da função f(x) no ponto
            médio do subintervalo e adiciona-o à soma acumulada s
            s = s + f(t(i));
      end
      % Calcula a integral aproximada usando a regra dos trapézios, onde Tsol é a
      solução aproximada. Esta solução é obtida multiplicando o tamanho do
      subintervalo h pela média ponderada dos valores da função nos extremos dos
      intervalos e nos pontos médios, conforme prescrito pela regra dos trapézios
      Tsol = h/2*(f(a)+2*s+f(b));
end
```

5.3 Regra de Simpson Simples

Fórmula:

$$\boldsymbol{S(f) = \int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]}$$

Onde:

- $\boldsymbol{S(f)}$ → Representa a aproximação da integral da função f no intervalo usando a Regra de Simpson Simples;
- $\boldsymbol{\int_a^b f(x)}$ → Integral da função f no intervalo$[a, b]$;
- $\boldsymbol{a}$ e $\boldsymbol{b}$ → São os limites de integração, onde a é o limite inferior e b é o limite superior;
- $\boldsymbol{f(a)}$ → É o valor da função f no ponto a;
- $\boldsymbol{f\left(\frac{a+b}{2}\right)}$ → É o valor da função f no ponto médio do intervalo $[a, b]$;
- $\boldsymbol{f(b)}$ → É o valor da função f no ponto b.
- $\boldsymbol{\frac{b-a}{6}}$ → É um fator de normalização que ajusta a soma ponderada dos valores da função para aproximar a integral.

Fórmula do erro para a Regra de Simpson Simples:

$$\boldsymbol{E(f) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)}\text{ , com } \boldsymbol{\xi} \in]a, b[$$

Onde:

- $\boldsymbol{E(f)}$ → Erro da aproximação ao usar a Regra de Simpson Simples para calcular a integral de uma função $f(x)$ no intervalo $[a, b]$;
- $\boldsymbol{(b-a)}$ → largura do intervalo de integração. Representa a diferença entre o limite superior b e o limite inferior a;
- $\boldsymbol{f^{(4)}(\xi)}$ → Representa a quarta derivada da função $f(x)$, avaliada no ponto ξ dentro do intervalo de integração $[a, b]$. Esta parte da fórmula captura a variação da função e sua curvatura dentro do intervalo.

Algoritmo:

1. **Entrada:**
   - Função $f(x)$ a ser integrada;
   - Limites de integração a e b;
2. **Cálculos:**
   - Calcular o ponto médio do intervalo: $\frac{a+b}{2}$;
   - Avaliar a função nos pontos médio, a, e b: $f(a)$, $f(\frac{a+b}{2})$, $f(b)$.
3. **Aplicar a Fórmula da Regra de Simpson:**

$$\boldsymbol{S(f) = \int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]}$$

5.4 Regra de Simpson Composta

Fórmula:

$$\int_a^b \boldsymbol{f(x)dx} \approx \boldsymbol{\frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]}$$

Onde:

- $\boldsymbol{a}$ e $\boldsymbol{b}$ → Os limites de integração inferior e superior, respectivamente;
- $\boldsymbol{f(x)}$ → A função que está a ser integrada;
- $\boldsymbol{dx}$ → O elemento de variação infinitesimal na variável de integração;
- $\boldsymbol{\frac{h}{3}}$ → Representa a largura de cada subintervalo usado na integração. O valor de h é o tamanho do passo entre os pontos de integração e $1/3$ é um fator constante na regra de Simpson;
- $\boldsymbol{f(x_0)}$ → Valor da função na abcissa x_0;
- $\boldsymbol{f(x_1)}$ → Valor da função na abcissa x_1;
- $\boldsymbol{f(x_{n-1})}$ → Valor da função na abcissa x_{n-1};
- $\boldsymbol{f(x_n)}$ → Valor da função na abcissa x_n;
- $\boldsymbol{n}$ → Número de subintervalos.

Fórmula do |erro| para a regra dos trapézios:

$$|\boldsymbol{e_S(f)}| \leq \boldsymbol{\frac{b-a}{180} h^4 M_4} \text{, com } M_4 = \max_{x\in[a,b]} |f^{(4)}(x)|$$

Onde:

- $|\boldsymbol{e_S(f)}|$ → Módulo do erro absoluto na aproximação da integral usando a regra de Simpson;
- $\boldsymbol{a}$ e $\boldsymbol{b}$ → Os limites de integração inferior e superior, respectivamente h;
- $\boldsymbol{h}$ → O tamanho do intervalo de integração;
- $\boldsymbol{M_4}$ → Constante que representa o valor máximo da quarta derivada de $f(x)$ no intervalo de integração $[a, b]$;
- $\boldsymbol{h^4}$ → Quarta potência do tamanho do subintervalo;
- $\boldsymbol{\frac{b-a}{180}}$ → Fator de ponderação.

Algoritmo:

1. Ler ***n***
2. Ler ***(a,b)***
3. $h \leftarrow \frac{b-a}{n}$
4. $x \leftarrow a$
5. $s \leftarrow 0$
6. Para ***i*** de ***1*** até $\boldsymbol{n-1}$ fazer
    a. $x \leftarrow x + h$
7. Se ***i*** par

    Então $s \leftarrow s + 2f(x)$

    Senão $s \leftarrow s + 4f(x)$
8. $S \leftarrow \frac{h}{3}(f(a) + 2 \cdot s + f(b))$
9. Escrever ***S***

Função em MatLab:

```
function out_S = RSimpson(~,f,a,b,h)
    % Esta linha cria um vetor t com valores espaçados uniformemente de a a b
    com um passo
    t = a:h:b;
    % Inicializa a variável s como zero, que será usada para acumular a soma
    dos valores da função f(x)
    s = 0;
    % Calcula o número de elementos no vetor t, que é o número total de
    subintervalos
    n = length(t);

    % Inicia um loop que percorre todos os subintervalos, exceto o último
    for i=1:n-1

        % Se o resto da divisão por 2 for 0, então i é par

        % Verifica se i é par usando a função mod(i, 2), que retorna o resto
        da divisão de i por 2. O operador ~ nega o resultado, portanto, esta
        condição é verdadeira se i for par.
        if ~(mod(i, 2))
        % Se i for par, adiciona duas vezes o valor de f(xi)f(xi) à soma s.
        Isso corresponde à regra de Simpson para pontos de índices pares.
        s = s + 2*f(t(i));

        % Se i não for par, ou seja, se for ímpar:
        Else
        % Adiciona quatro vezes o valor de f(xi)à soma s. Isso também
        corresponde à regra de Simpson, mas para pontos de índices ímpares.
        s = s + 4*f(t(i));
```

```
            end
        end

        % Calcula a aproximação da integral definida usando a regra de Simpson. É uma soma ponderada dos valores da função nos extremos e nos pontos intermediários. A constante h/3 é um fator de ponderação da regra de Simpson. O resultado é armazenado em out_S e é retornado como saída da função
        out_S = h/3*(f(a)+s+f(b));
        end
end
```

6 FUNÇÃO HARMÓNICA

Uma função harmónica é uma função que satisfaz a equação de Laplace, ou seja, a soma das segundas derivadas parciais da função em relação a todas as suas variáveis é zero. Estas funções são comuns em problemas de física e engenharia, como na teoria do potencial, eletrostática e mecânica dos fluidos, descrevendo fenómenos como o campo elétrico e o fluxo de calor.

Equação de Laplace:

$$\boldsymbol{\nabla}^2 u = 0$$

Onde:

$\boldsymbol{\nabla^2}$ → Operador Laplaciano $\boldsymbol{u}$ → Função $u(x, y)$

Pela definição, sabemos que é necessário mostrar que a soma das suas segundas derivadas parciais é igual a zero, ou seja:

$$\boldsymbol{\nabla}^2 u = 0 \Leftrightarrow \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Usando a equação $u(x, y) = x^2 - y^2$ como exemplo:

1ºPasso: Calcular a primeira derivada parcial de $\boldsymbol{u}$:

$$\frac{\partial u}{\partial x} = 2x \qquad \frac{\partial u}{\partial y} = -2y$$

2ºPasso: Calcular a segunda derivada parcial de $\boldsymbol{u}$:

$$\frac{\partial^2 u}{\partial x^2} = 2 \qquad \frac{\partial^2 u}{\partial y^2} = -2$$

3ºPasso: Somar as segundas derivadas parciais

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 2 + (-2) = 0$$

4ºPasso: Conclusão

É uma função harmónica.

7 EXEMPLOS DE APLICAÇÃO E TESTE DOS MÉTODOS

Para ilustrar a aplicação de cada método de diferenciação e integração vamos usar um problema fictício de velocidade ao longo do tempo.

Vamos escolher uma função de variável real $y = f(x)$ e uma função de duas variáveis reais $z = f(x, y)$. Em seguida, aplicaremos os métodos de diferenciação e integração solicitados.

7.1 Função de Variável Real e Intervalo

$$f(x) = x^2 + 3x + 2$$

$$[a, b] = [0,5]$$

$$h = 1$$

7.1.1 Derivação

7.1.1.1 Diferenças Finitas Progressivas (2 pontos) para $f(x)$

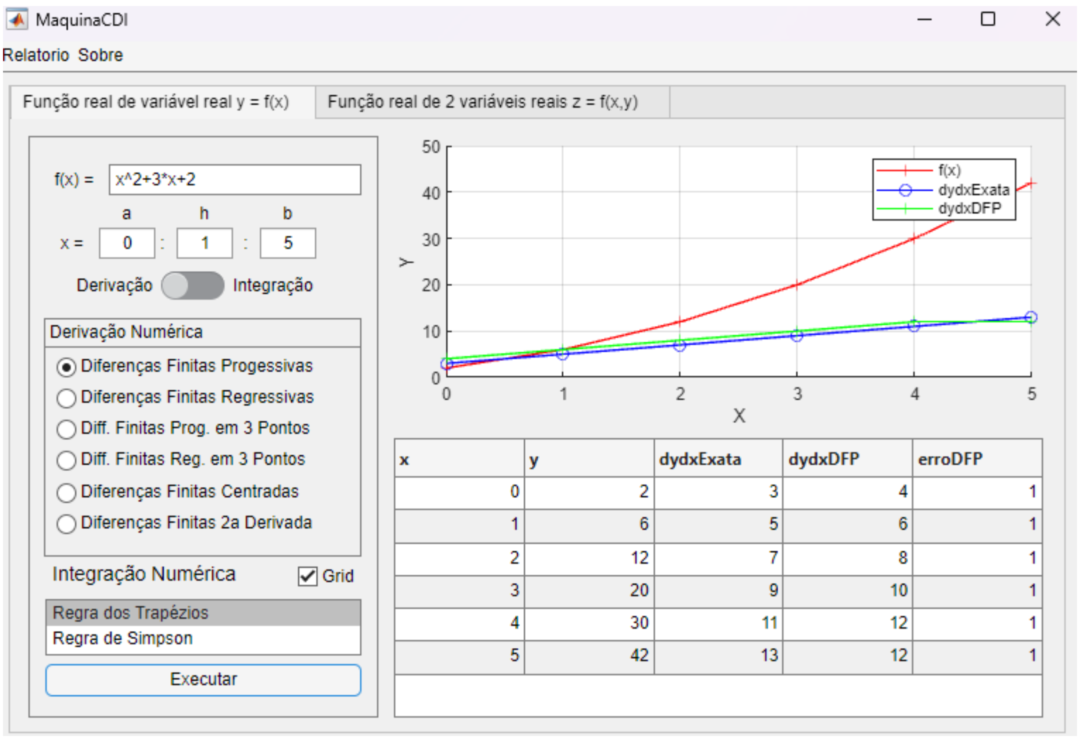


Figura 2 - Diferenças Finitas Progressivas (2 pontos)

7.1.1.2 Diferenças Finitas Regressivas (2 pontos) para $f(x)$

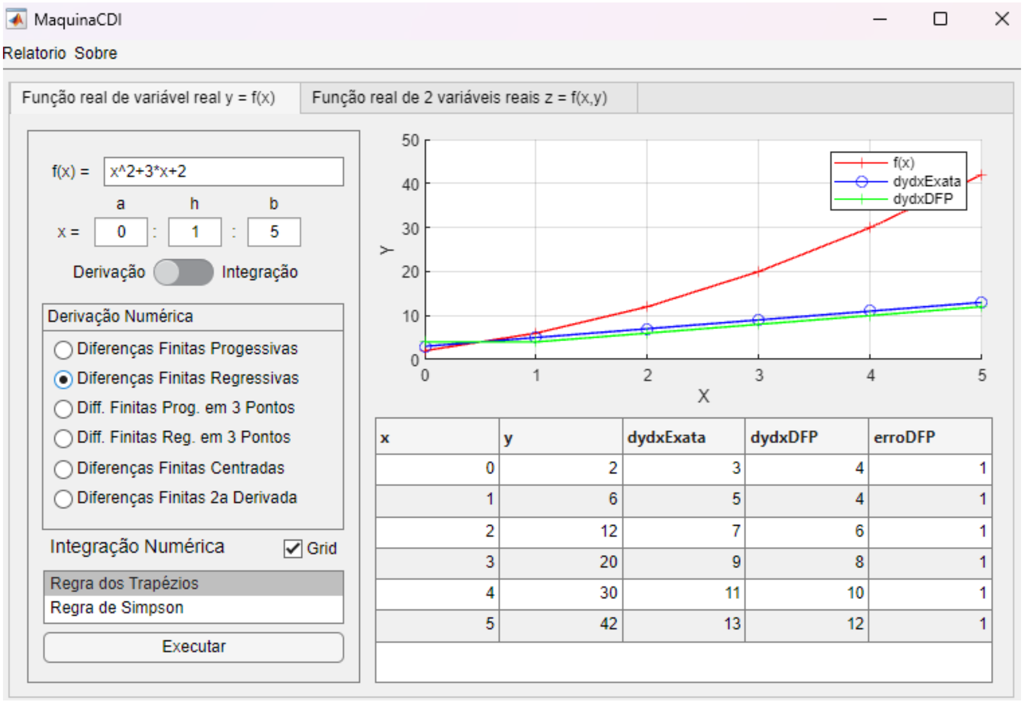


Figura 3 - Diferenças Finitas Regressivas (2 pontos)

7.1.1.3 Diferenças Finitas Progressivas (3 pontos) para $f(x)$

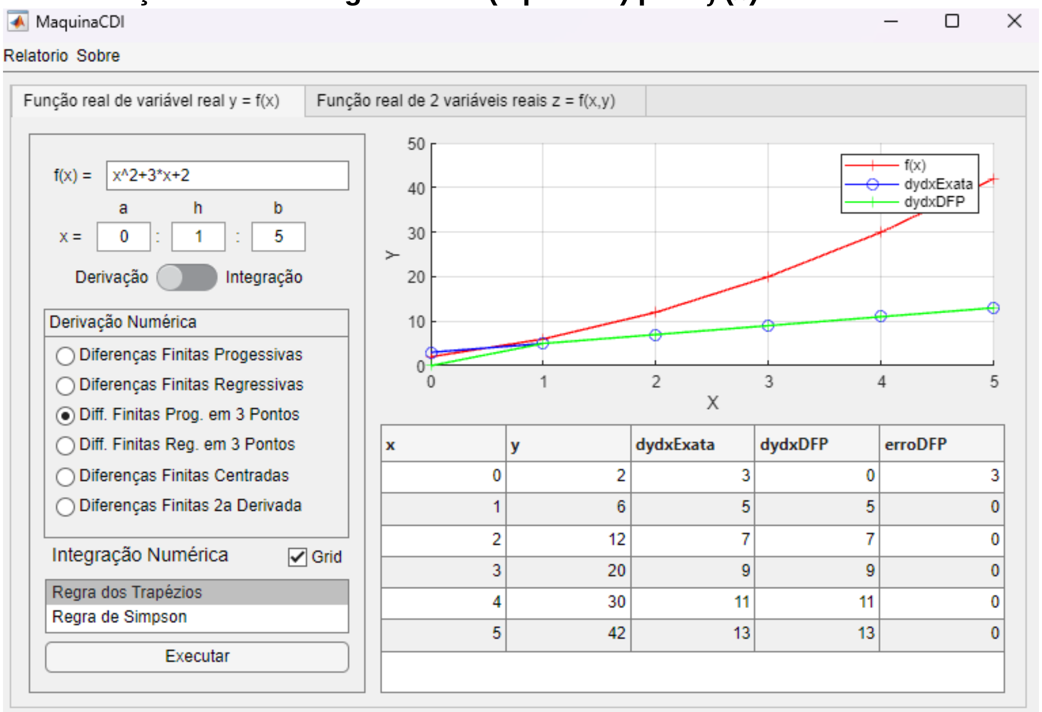


Figura 4 - Diferenças Finitas Progressivas (3 pontos)

7.1.1.4 Diferenças Finitas Regressivas (3 pontos) para $f(x)$

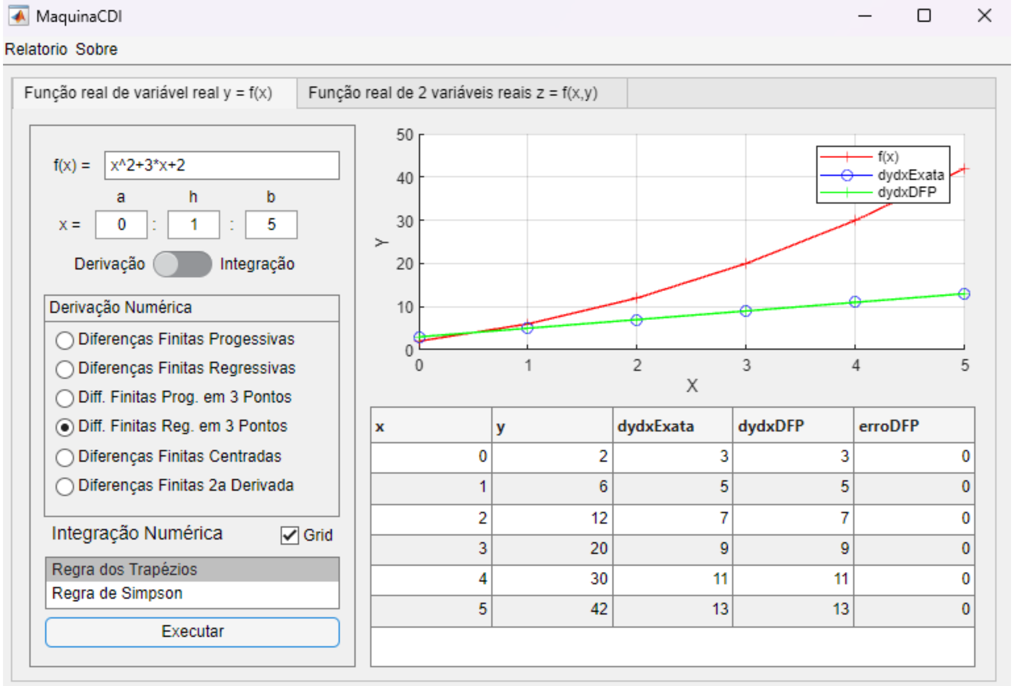


Figura 5 - Diferenças Finitas Regressivas (3 pontos)

7.1.1.5 Diferenças Finitas Centradas (3 pontos) para $f(x)$

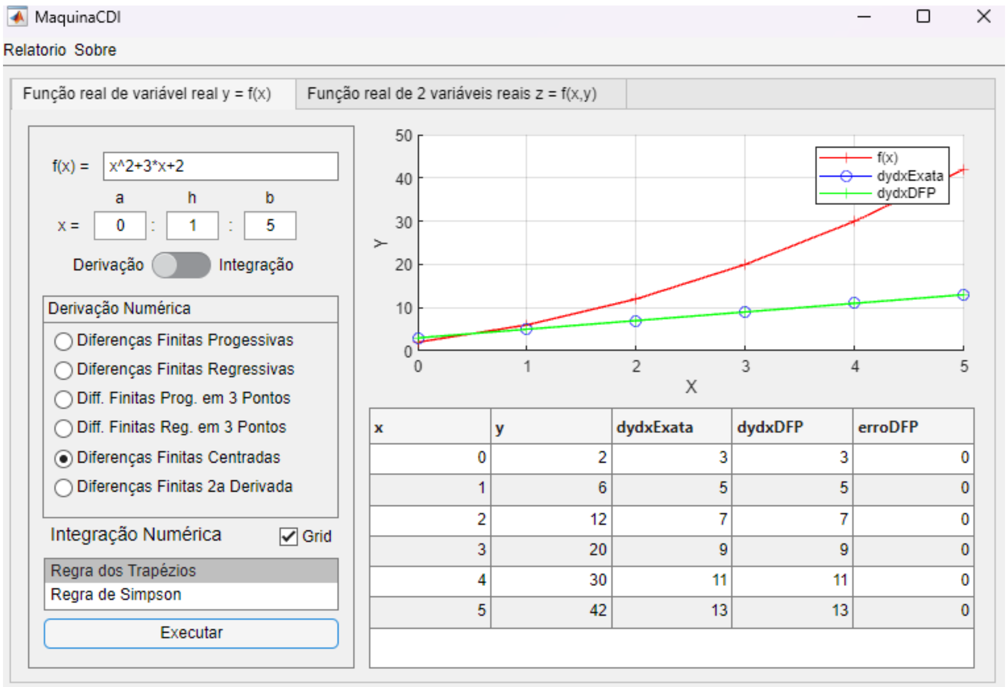


Figura 6 - Diferenças Finitas Centradas (3 pontos)

7.1.1.6 Diferenças Finitas 2ªDerivada para $f(x)$

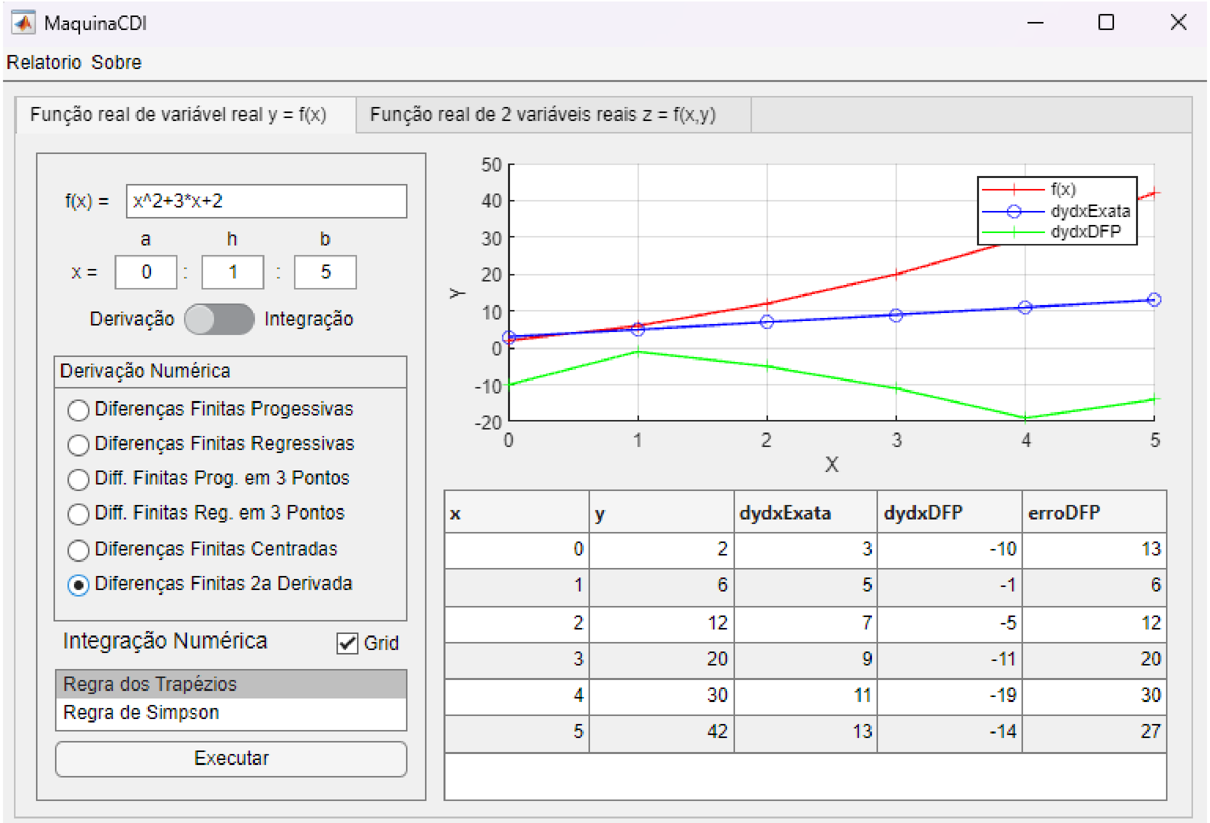


Figura 7 - Diferenças Finitas 2ª Derivada

7.1.2 Integração

7.1.2.1 Regra dos Trapézios Composta

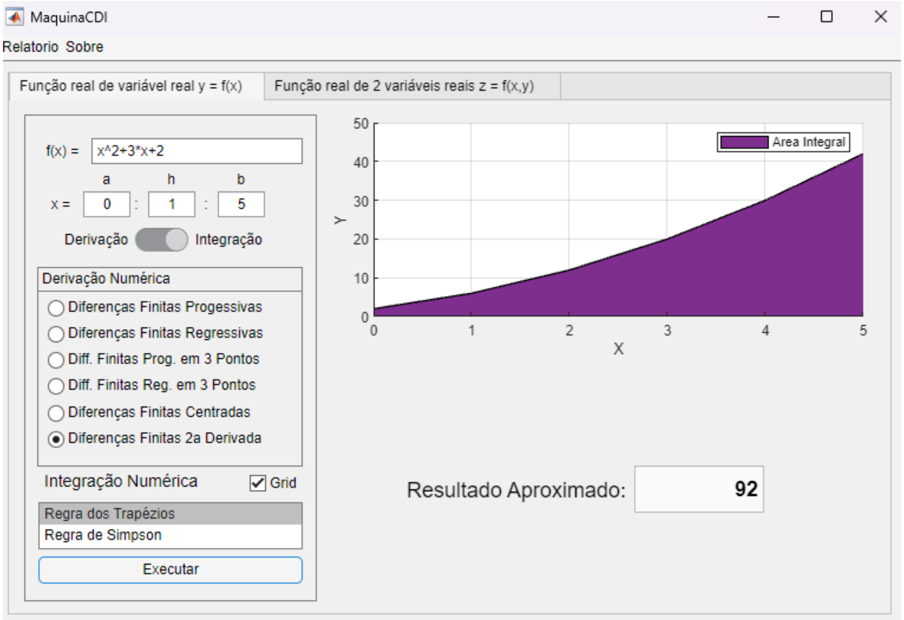


Figura 8 - Regra dos Trapézios Composta

7.1.2.2 Regra Simpson Composta

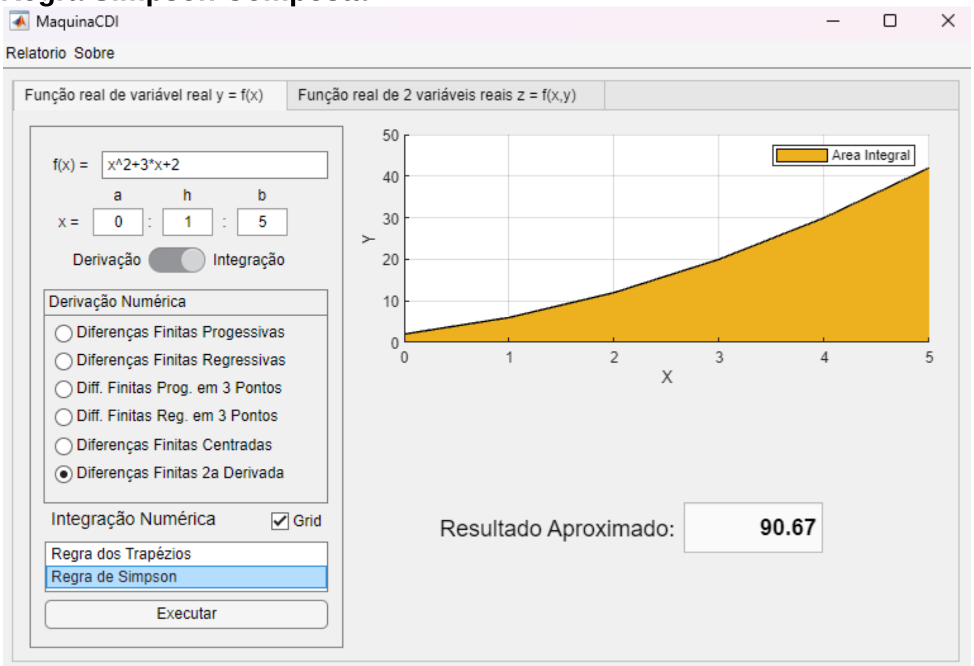


Figura 9 – Regra Simpson Composta

7.2 Função de Duas Variáveis Reais e Intervalos

$$f(x,y) = x^2 + y^2$$

$$[a,b] = [0{,}5] \text{ com passo } h1 = 1$$

$$[c,d] = [0{,}5] \text{ com passo } h2 = 1$$

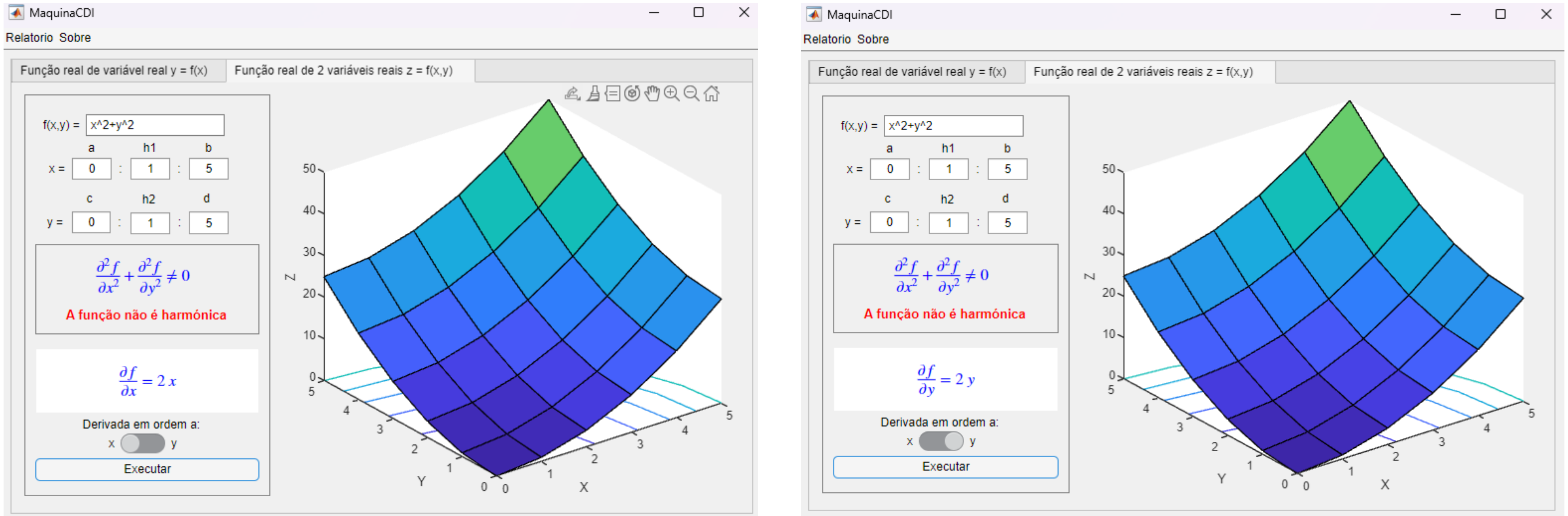
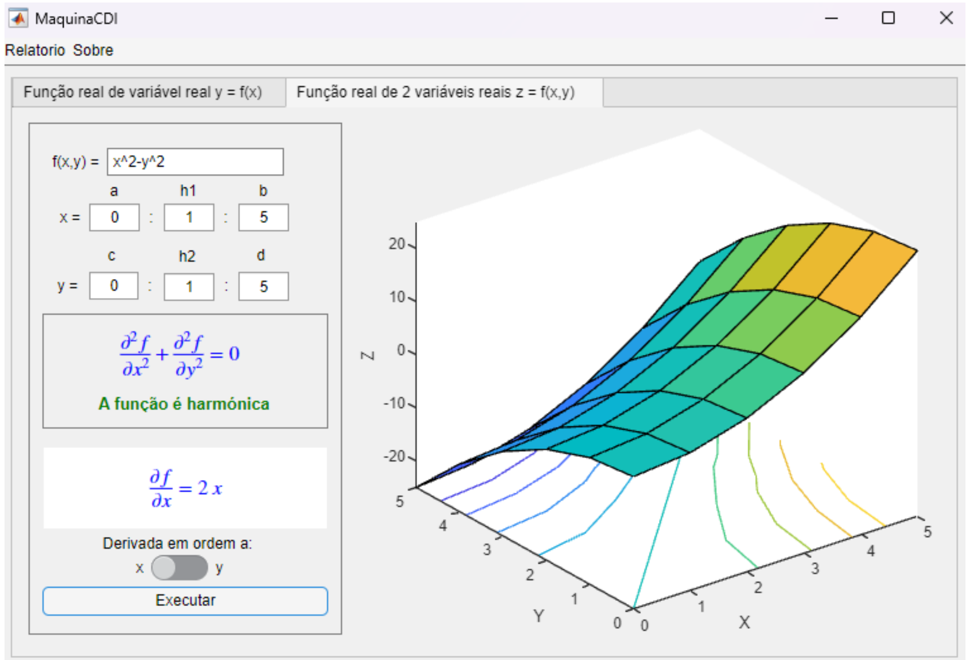


Figura 10 - Função de Duas Variáveis Reais Não Harmónica

$f(x,y) = x^2 - y^2$

$[a,b] = [0{,}5]$ com passo $h1 = 1$

$[c,d] = [0{,}5]$ com passo $h2 = 1$



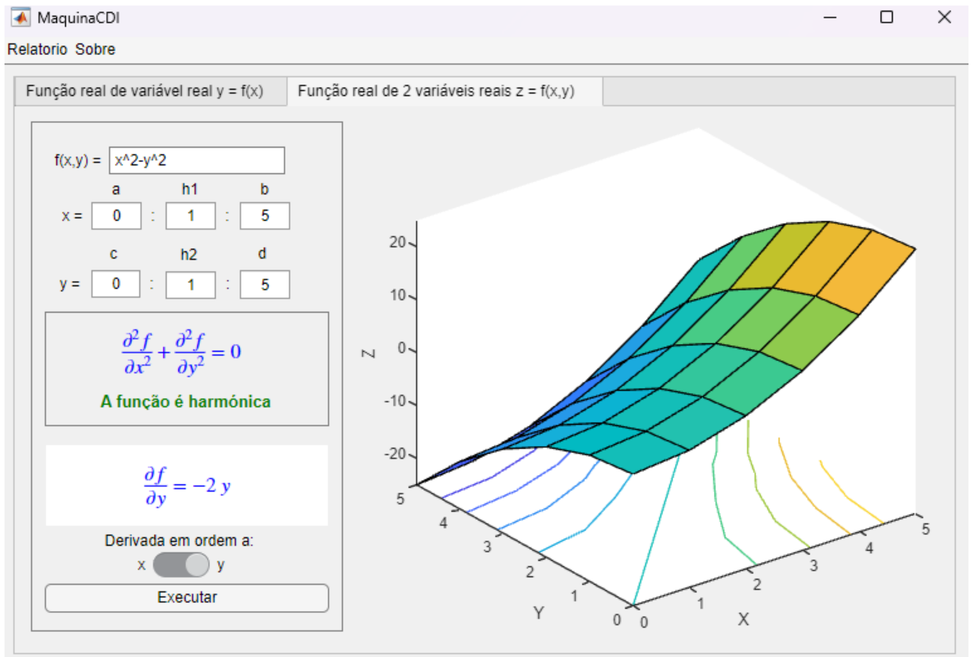


Figura 11 - Função de Duas Variáveis Reais Harmónica

8 CONCLUSÃO

Os métodos numéricos têm uma vasta gama de aplicações, tanto para derivadas como para integrais, o que facilita a resolução de problemas complexos em várias áreas da ciência e além. Muitas vezes, o cálculo de integrais de forma analítica não é possível, tornando os métodos numéricos indispensáveis.

No contexto da derivação numérica, as fórmulas que utilizam 3 pontos fornecem uma melhor aproximação do valor real em comparação com aquelas que utilizam apenas 2 pontos, que frequentemente apresentam o dobro do erro ou mais.

Para integrais, a Regra de Simpson é superior à Regra dos Trapézios para valores elevados de n e grandes intervalos (ou seja, pequeno h). No entanto, para valores pequenos de n e intervalos reduzidos (grande h), a diferença é menos perceptível, e por vezes a Regra dos Trapézios pode ter erros menores. Com n suficientemente grande, a Regra de Simpson tende a ter erros menores.

Este trabalho prático demonstra a importância das equações diferenciais ordinárias na modelação de problemas científicos e o papel crucial dos métodos numéricos na sua resolução. Os computadores são ferramentas indispensáveis no estudo destas equações, permitindo a execução de algoritmos para obter aproximações numéricas das soluções.

9 BIBLIOGRAFIA

- Correia, A. (2024, 29 de abril). *Matlab .: Atividade05Trabalho - MáquinaCDI*. Moodle.https://pt.wikipedia.org/wiki/Diferencia%C3%A7%C3%A3o_num%C3%A9rica
- Contribuidores dos projetos da Wikipedia. (2004, 9 de março). Método das diferenças finitas. Wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Método_das_diferenças_finitas
- *Diferenças finitas*. (s.d.). Inicial — UFRGS | Universidade Federal do Rio Grande do Sul. https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/dn-diferencas_finitas.html
- *Diferenciação numérica*. (2017, 5 de setembro). Métodos Matemáticos e Computacionais. https://metmatcom.blogspot.com/2017/09/diferenciacao-numerica.html
- *Differences and approximate derivatives - MATLAB diff*. (s.d.). MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. https://www.mathworks.com/help/matlab/ref/diff.html
- *Definite and indefinite integrals - MATLAB int*. (s.d.). MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. https://www.mathworks.com/help/symbolic/sym.int.html
- Contribuidores dos projetos da Wikipedia. (2008a, 19 de fevereiro). *Função harmônica*. Wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Função_harmônica
- Contribuidores dos projetos da Wikipedia. (2007, 17 de julho). *Integração numérica*. Wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Integração_numérica
- *Integração Numérica*. (s.d.). Department of Mathematics — Técnico, Lisboa. https://www.math.tecnico.ulisboa.pt/~calves/courses/integra/
- Contribuidores dos projetos da Wikipedia. (2008b, 15 de março). *Diferenciação numérica – Wikipédia, a enciclopédia livre*. Wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Diferenciação_numérica
- Correia, A. (s.d.). *Cap5_IntegracaoNumerica_rascunho*.

10 AUTOAVALIAÇÃO E HETEROAVALIAÇÃO DO TRABALHO SUBMETIDO

Tendo em conta o que foi feito ao longo do trabalho e que o mesmo vale 5 valores, concluímos assim as seguintes auto e hétero avaliações:

Autoavaliação:

Ana Rita Conceição Pessoa – 4 valores

João Francisco de Matos Claro – 5 valores

Heteroavaliação:

Ana Rita Conceição Pessoa – 4 valores

João Francisco de Matos Claro – 5 valores

Instituto Superior
de Engenharia
Politécnico de Coimbra