

RELATÓRIO

Atividade03Trabalho

Cálculo Diferencial e Integral

Licenciatura em Engenharia Informática

“

O cálculo diferencial e integral, é um ramo importante da matemática que se dedica ao estudo de taxas de variação de grandezas e acumulação de quantidades.

Trabalho realizado por:

Tomás Gomes Silva - 2020143845

Tomás da Cunha Pinto - 2020144067

Francisco Santos Seabra Mendes - 2020143982

Análise Matemática II – Trabalho 3

Índice

Índice	3
Introdução.....	4
Métodos Numéricos para Derivação.....	5
Fórmulas de Diferenças Finitas em 2 pontos	6
Progressivas	6
Regressivas	7
Fórmulas de Diferenças Finitas em 3 pontos	8
Progressivas	8
Regressivas	9
Centradas.....	10
2 ^a Derivada	11
Métodos Numéricos para Integração.....	12
Regra dos Trapézios	12
Regra de Simpson	13
Derivação simbólica no MATLAB	14
Função <i>diff</i> do MATLAB	14
Integração simbólica no MATLAB	15
Função <i>int</i> do MATLAB	15
Manual de Utilização	16
Conclusão.....	21

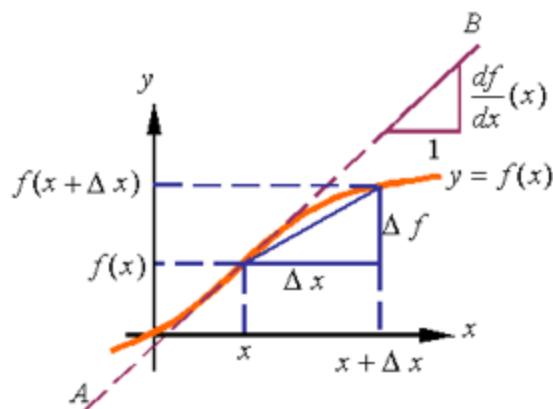
Introdução

O terceiro trabalho proposto para a unidade curricular de Análise Matemática 2 consiste em criar uma aplicação para efetuar **Cálculo Diferencial e Integral (CDI)** e também outras operações relacionadas com o tema.

Este relatório irá abordar todos os métodos utilizados e alguns pontos chave da criação da aplicação em MATLAB.

Para calcular uma derivada numericamente, utilizámos 6 fórmulas de diferenças divididas finitas: [Progressivas \(2 pontos\)](#), [Regressivas \(2 pontos\)](#), [Progressivas \(3 pontos\)](#), [Regressivas \(3 pontos\)](#), [Centradas \(3 pontos\)](#) e [2^a Derivada \(3 pontos\)](#).

No caso do cálculo integral, calculado de forma numérica, foram implementadas 2 regras: a [Regra dos Trapézios](#) e a [Regra de Simpson](#).



Métodos Numéricos para Derivação

Em várias situações, o cálculo analítico da derivada de uma função pode demorar bastante a ser calculada. Proveniente dessa necessidade, foi necessário arranjar métodos fiáveis que fossem capazes de se aproximarem às soluções exatas.

Exemplo de um método numérico:

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

Fórmula Diferenças Finitas Progressivas (2 pontos)

Fórmulas de Diferenças Finitas

em 2 pontos

Progressivas

Fórmulas:

A fórmula de diferenças finitas **Progressivas (2 pontos)** é um procedimento numérico que permite aproximar a solução da derivada de uma função num dado intervalo.

Para utilizarmos este método basta-nos apenas passar como parâmetros: a função a derivar, o intervalo e a amplitude de cada subintervalo. Corremos a fórmula num ciclo para que este cálculo seja efetuado em vários pontos da função.

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

$f(x_{k+1})$: Próximo ponto da função f

$f(x_k)$: Ponto atual da função f

h : Valor de cada subintervalo

Algoritmo/Função:

1. Criar um vetor que contém os valores de **a** a **b** com passo **h**
2. Criar uma variável para armazenar o tamanho do vetor anterior
3. Pré alocar memória para o array que vai guardar os valores calculados
4. Fazer um ciclo iterativo que percorra desde **1** a **n-1** para calcular a solução aproximada utilizando a fórmula das diferenças **Progressivas (2 pontos)**
5. Calcular caso específico no final pois este não é calculado no ciclo

```

x = a:h:b; % Vetor com valores de A a B com step H
n = length(x); % Tamanho do vetor anterior

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

dydx = zeros(1,n); % Pre-alocação de memória para os valores calculados

for i=1:n-1 % Percorre elementos do vetor X de 1 a n-1
    dydx(i) = (y(i+1) - y(i)) / h; % Aplica a fórmula (Progressiva)
end

dydx(n) = (y(n) - y(n-1)) / h; % Caso específico no final para o elemento
% que não foi corrido no ciclo em cima

```

Figura 1 - Algoritmo da fórmula das diferenças Progressivas (2 pontos)

Regressivas

Fórmulas:

A fórmula de diferenças finitas **Regressivas (2 pontos)** é um procedimento numérico que permite aproximar a solução da derivada de uma função num dado intervalo.

Para utilizarmos este método basta-nos apenas passar como parâmetros: a função a derivar, o intervalo e a amplitude de cada subintervalo. Corremos a fórmula num ciclo para que este cálculo seja efetuado em vários pontos da função.

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

- $f(x_{k-1})$: Ponto anterior da função f
- $f(x_k)$: Ponto atual da função f
- h : Valor de cada subintervalo

Algoritmo/Função:

1. Criar um vetor que contém os valores de **a** a **b** com passo **h**
2. Criar uma variável para armazenar o tamanho do vetor anterior
3. Pré alocar memória para o array que vai guardar os valores calculados
4. Calcular caso específico no início pois este não será calculado no ciclo
5. Fazer um ciclo iterativo que percorra desde **2** a **n** para calcular a solução aproximada utilizando a fórmula das diferenças **Regressivas (2 pontos)**

```

x = a:h:b; % Vetor com valores de A a B com step H
n = length(x); % Tamanho do vetor anterior

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

dydx = zeros(1,n); % Pre-alocação de memória para os valores calculados

dydx(1) = (y(2) - y(1)) / h; % Caso específico no início para o elemento
% que não vai ser corrido no ciclo em baixo

for i=2:n % Percorre elementos do vetor X de 2 a n
    dydx(i) = (y(i) - y(i-1)) / h; % Aplica a fórmula (Regressiva)
end

```

Figura 5 - Algoritmo da fórmula das diferenças Regressivas (2 pontos)

Fórmulas de Diferenças Finitas

em 3 pontos

Progressivas

Fórmulas:

A fórmula de diferenças finitas **Progressivas (3 pontos)** é um procedimento numérico que permite aproximar a solução da derivada de uma função num dado intervalo.

Para utilizarmos este método basta-nos apenas passar como parâmetros: a função a derivar, o intervalo e a amplitude de cada subintervalo. Corremos a fórmula num ciclo para que este cálculo seja efetuado em vários pontos da função.

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

- $f(x_{k-1})$: Ponto anterior da função f
- $f(x_k)$: Ponto atual da função f
- $f(x_{k+2})$: Dois pontos a seguir ao ponto atual da função f
- h : Valor de cada subintervalo

Algoritmo/Função:

1. Criar um vetor que contém os valores de **a** a **b** com passo **h**
2. Criar uma variável para armazenar o tamanho do vetor anterior
3. Pré alocar memória para o array que vai guardar os valores calculados
4. Fazer um ciclo iterativo que percorra desde **1** a **n-2** para calcular a solução aproximada utilizando a fórmula das diferenças **Progressivas (3 pontos)**
5. Calcular casos específicos no final pois estes não foram calculados no ciclo

```
x = a:h:b; % Vetor com valores de A a B com step H
n = length(x); % Tamanho do vetor anterior

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

dydx = zeros(1,n); % Pre-alocação de memória para os valores calculados

for i=1:n-2 % Percorre elementos do vetor X de 1 a n-2
    dydx(i) = (-3 * y(i) + 4 * y(i+1) - y(i+2)) / (2*h); % Aplica a fórmula (Progressiva)
end

% Casos específicos no final para os elementos
% que não foram corridos no ciclo em cima
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 6 - Algoritmo da fórmula das diferenças Progressivas (3 pontos)

Regressivas

Fórmulas:

A fórmula de diferenças finitas **Regressivas (3 pontos)** é um procedimento numérico que permite aproximar a solução da derivada de uma função num dado intervalo.

Para utilizarmos este método basta-nos apenas passar como parâmetros: a função a derivar, o intervalo e a amplitude de cada subintervalo. Corremos a fórmula num ciclo para que este cálculo seja efetuado em vários pontos da função.

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

- $f(x_{k-2})$: Dois pontos anteriores ao ponto atual da função f
- $f(x_{k-1})$: Ponto anterior da função f
- $f(x_k)$: Ponto atual da função f
- h : Valor de cada subintervalo

Algoritmo/Função:

1. Criar um vetor que contém os valores de **a** a **b** com passo **h**
2. Criar uma variável para armazenar o tamanho do vetor anterior
3. Pré alocar memória para o array que vai guardar os valores calculados
4. Calcular casos específicos no início pois estes não serão calculados no ciclo
5. Fazer um ciclo iterativo que percorra desde **3** a **n** para calcular a solução aproximada utilizando a fórmula das diferenças **Regressivas (3 pontos)**

```

x = a:h:b; % Vetor com valores de A a B com step H
n = length(x); % Tamanho do vetor anterior

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

dydx = zeros(1,n); % Pre-alocação de memória para os valores calculados

% Casos específicos no início para os elementos
% que não irão ser corridos no ciclo em baixo
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 % Percorre elementos do vetor X de 3 a n
    dydx(i) = (y(i-2) - 4 * y(i-1) + 3 * y(i)) / (2*h); % Aplica a fórmula (Regressiva)
end

```

Figura 7 - Algoritmo da fórmula das diferenças Regressivas (3 pontos)

Centradas

Fórmulas:

A fórmula de diferenças finitas **Centradas (3 pontos)** é um procedimento numérico que permite aproximar a solução da derivada de uma função num dado intervalo.

Para utilizarmos este método basta-nos apenas passar como parâmetros: a função a derivar, o intervalo e a amplitude de cada subintervalo. Corremos a fórmula num ciclo para que este cálculo seja efetuado em vários pontos da função.

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

- $f(x_{k-1})$: Ponto anterior da função f
- $f(x_k)$: Ponto atual da função f
- $f(x_{k+1})$: Ponto seguinte da função f
- h : Valor de cada subintervalo

Algoritmo/Função:

1. Criar um vetor que contém os valores de **a** a **b** com passo **h**
2. Criar uma variável para armazenar o tamanho do vetor anterior
3. Pré alocar memória para o array que vai guardar os valores calculados
4. Calcular caso específico no início pois este não será calculado no ciclo
5. Fazer um ciclo iterativo que percorra desde **2** a **n-1** para calcular a solução aproximada utilizando a fórmula das diferenças **Centradas (3 pontos)**
6. Calcular caso específico no final pois este não é calculado no ciclo

```

x = a:h:b; % Vetor com valores de A a B com step H
n = length(x); % Tamanho do vetor anterior

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

dydx = zeros(1,n); % Pre-alocação de memória para os valores calculados

dydx(1) = (y(2) - y(1)) / h; % Caso específico no início para o elemento
% que não irá ser corrido no ciclo em baixo

for i=2:n-1 % Percorre elementos do vetor X de 2 a n-1
    dydx(i) = (y(i+1) - y(i-1)) / (2*h); % Aplica a fórmula (Centradas)
end

dydx(n) = (y(n) - y(n-1)) / h; % Caso específico no final para o elemento
% que não foi corrido no ciclo em cima

```

Figura 8 - Algoritmo da fórmula das diferenças Centradas (3 pontos)

2^a Derivada

Fórmulas:

A fórmula de diferenças finitas de **2^a Derivada (3 pontos)** é um procedimento numérico que permite aproximar a solução da segunda derivada de uma função num intervalo.

Para utilizarmos este método basta-nos apenas passar como parâmetros: a função a derivar, o intervalo e a amplitude de cada subintervalo. Corremos a fórmula num ciclo para que este cálculo seja efetuado em vários pontos da função.

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

- $f(x_{k-1})$: Ponto anterior da função f
- $f(x_k)$: Ponto atual da função f
- $f(x_{k+1})$: Ponto seguinte da função f
- h : Valor de cada subintervalo

Algoritmo/Função:

1. Criar um vetor que contém os valores de **a** a **b** com passo **h**
2. Criar uma variável para armazenar o tamanho do vetor anterior
3. Pré alocar memória para o array que vai guardar os valores calculados
4. Calcular caso específico no início pois este não será calculado no ciclo
5. Fazer um ciclo iterativo que percorra desde **2** a **n-1** para calcular a solução aproximada utilizando a fórmula das diferenças de **2^a Derivada (3 pontos)**
6. Calcular caso específico no final pois este é calculado no ciclo

```
x = a:h:b; % Vetor com valores de A a B com step H
n = length(x); % Tamanho do vetor anterior

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

dydx = zeros(1,n); % Pre-alocação de memória para os valores calculados

dydx(1) = (y(3) - 2*y(2) + y(1)) / (h^2); % Caso específico no início para o elemento
% que não irá ser corrido no ciclo em baixo

for i=2:n-1
    dydx(i) = (y(i+1) - 2*y(i) + y(i-1)) / h^2; % Aplica a fórmula (2a Derivada)
end

dydx(n) = (y(n) - 2*y(n-1) + y(n-2)) / (h^2); % Caso específico no final para o elemento
% que não foi corrido no ciclo em cima
```

Figura 9 - Algoritmo da fórmula das diferenças de 2^a Derivada (3 pontos)

Métodos Numéricos para Integração

Para calcularmos a primitiva de uma função podemos recorrer a métodos numéricos de integração que aproximam a solução exata do problema. Os métodos implementados neste trabalho foram: a [Regra dos Trapézios](#) e a [Regra de Simpson](#).

Regra dos Trapézios

O método de funcionamento da [Regra dos Trapézios](#), como o nome indica, consiste em calcular a área de vários trapézios divididos em subintervalo de modo a aproximar a solução exata da primitiva de uma função.

$$\int_a^b f(x)dx \approx \frac{h}{2}f(a) + 2s + f(b)$$

$$s = s + f(x)$$

- $f(a)$: Valor da função no ponto inicial
- $f(b)$: Valor da função no ponto final
- s : Soma da área dos trapézios
- h : Valor de cada subintervalo

```

h = (b-a)/n; % Amplitude dos subintervalos
x = a; % X começa em A (primeiro ponto do intervalo)
s = 0; % Soma começa a zero

for i=1:n-1 % Executa o ciclo de 1 a n-1
    x = x + h; % Próximo ponto
    s = s + f(x); % Soma é igual à soma mais valor da função no ponto atual
end

T = (h/2)*(f(a)+2*s+f(b)); % Aplica a Regra dos Trapézios
end

```

Figura 10 - Algoritmo da Regra dos Trapézios

Regra de Simpson

No que toca à **Regra de Simpson** sabemos que esta também se trata de outro exemplo de Fórmula de Newton-Cotes fechada. Desta vez, ao invés de aproximarmos a solução exata com recurso a trapézios fazemo-lo utilizando polinómios interpoladores de 1º grau (ou seja, uma reta).

$$\int_a^b f(x)dx \approx \frac{h}{3}f(a) + s + f(b)$$

Se n for par: $s = s + 2f(x)$

Se n for ímpar: $s = s + 4f(x)$

- $f(a)$: Valor da função no ponto inicial
- $f(b)$: Valor da função no ponto final
- s : Soma da área dos trapézios
- h : Valor de cada subintervalo

```

for i=1:n-1 % Executa o ciclo de 1 a n-1
    x = x + h; % Próximo ponto
    if(~mod(i,2)) % Se o número da iteração for par
        s = s + 2*f(x); % Soma é igual à soma vezes 2 vezes a o valor da função no ponto atual
    else % Se o número da iteração for ímpar
        s = s + 4*f(x); % Soma é igual à soma vezes 4 vezes a o valor da função no ponto atual
    end
end
out_S = (h/3)*(f(a)+s+f(b)); % Aplica a Regra de Simpson
end

```

Figura 11 - Algoritmo da Regra de Simpson

Derivação simbólica no MATLAB

Função *diff* do MATLAB

Graças à poderosa toolbox, *Symbolic Math Toolbox*, que faz parte do MATLAB, podemos utilizar a função **diff** que permite calcular derivadas de uma função da ordem que quisermos. A função retorna uma expressão simbólica que representada a expressão da derivada da função simbólica introduzida nos parâmetros.

$$f(x)' \dots = \text{diff}(f, \text{dim})$$

$$f(x)'' = \text{diff}(\cos(x), 2)$$

f : Função simbólica a ser derivada

dim : Ordem da derivada

```
d = diff(f,x);
g = @(x) eval(vectorize(d));
```

Integração simbólica no MATLAB

Função *int* do MATLAB

De forma a integrarmos uma função analiticamente podemos recorrer, mais uma vez, às potencialidades da *Symbolic Math Toolbox* que conta com a função **int** que nos permite calcular a solução exata da primitiva de uma função. Para além disso podemos também calcular integrais definidos num intervalo de **a** a **b**.

$$\int_a^b f(x)dx = \mathbf{int}(\mathbf{f}, \mathbf{a}, \mathbf{b})$$

$$\int_a^b f(x)dx = \mathbf{int}(\cos(x), 2, 3)$$

- f** : Função simbólica a ser primitivada
- a** : Limite esquerdo do integral definido (opcional)
- b** : Limite direito do integral definido (opcional)

```
d = int(f,x);
g = @(x) eval(vectorize(d));
```

Manual de Utilização

A aplicação é constituída por quatro separadores:

- **Menu (1 variável real)** – Este menu permite ao utilizador introduzir os dados necessários para a utilização dos métodos numéricos existentes de modo. A solução exata também é calculada e apresentada neste menu
- **Menu (2 variáveis reais)** – O segundo separador é uma adaptação do menu anterior só que desta vez para funções reais de 2 variáveis reais. O utilizador pode introduzir os dados nos campos destinados a esse efeito e escolher o tipo de representação gráfica que quer
- **Função real de 1 variável real: $y = f(x)$** – Este separador mostra a representação gráfica da função exata, da função derivada/primitivada e do método utilizado. A tabela mostra os valores associados à representação gráfica e ao cálculo numérico efetuado previamente com recurso às fórmulas/regras
- **Função real de 2 variáveis reais: $z = f(x,y)$** – O último separador conta apenas com um gráfico onde será representada a superfície 3D correspondente à função inserida no menu apropriado

Análise Matemática II – Trabalho 3

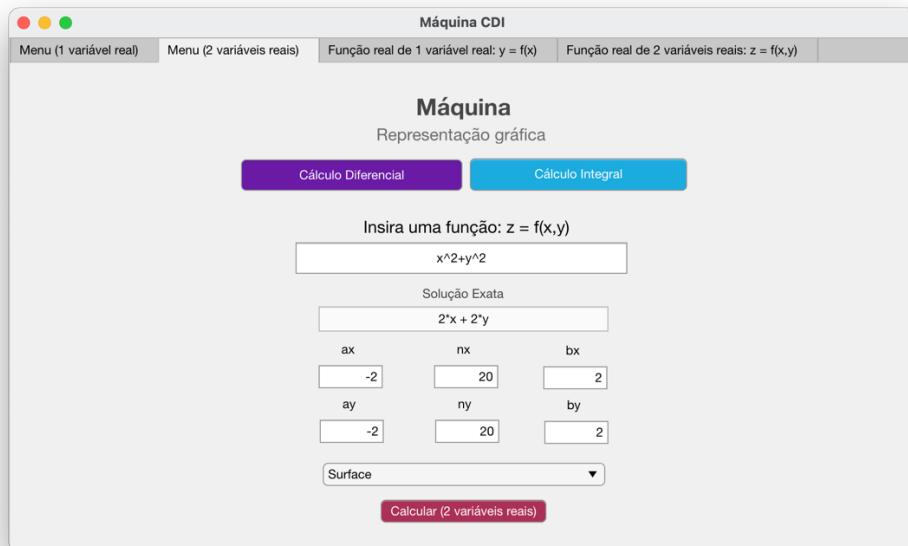
Menu (1 variável real)



No **Menu (1 variável real)** temos dois botões para escolher se queremos derivar ou primitivar a função inserida no campo abaixo. No campo da **solução exata** será necessário preencher os campos **a**, **n** e **b** assim como é necessário escolher um método para utilizar.

Análise Matemática II – Trabalho 3

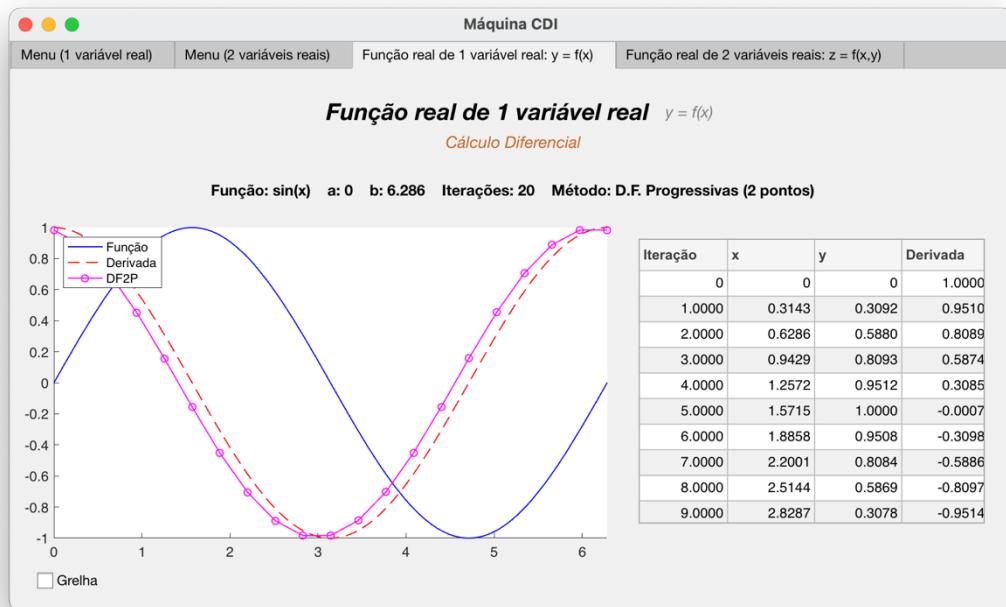
Menu (2 variáveis reais)



Quanto ao separador **Menu (2 variáveis reais)** a interface é praticamente igual ao menu anteriormente exposto e as únicas diferenças são os campos para introduzir os valores para **ay**, **ny**, **by** para calcular a solução numericamente. O dropdown no fundo da página permite ao utilizador escolher o método de representação gráfica com o qual quer que a superfície 3D seja gerada.

Análise Matemática II – Trabalho 3

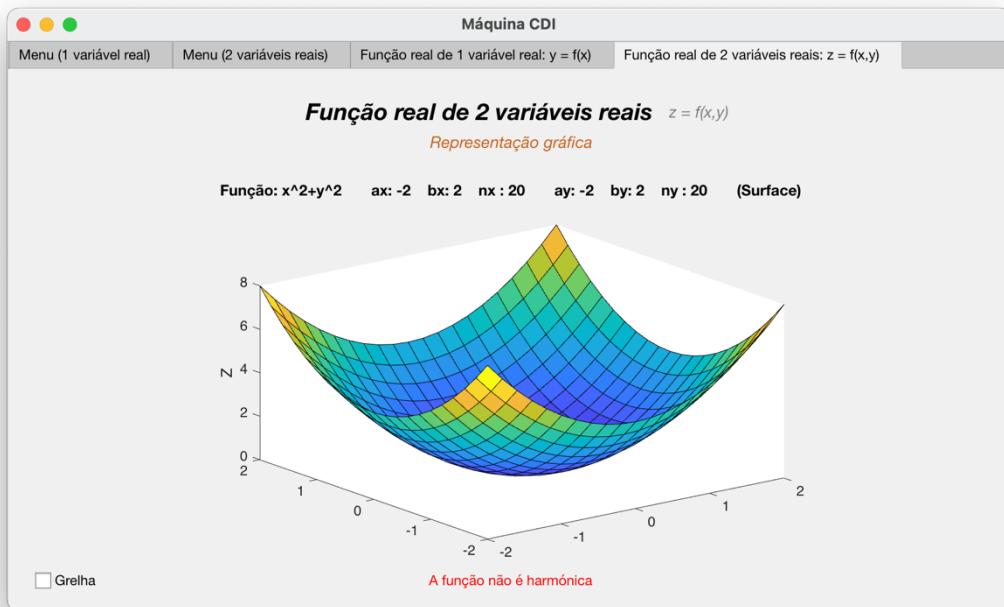
Função real de 1 variável real: $y = f(x)$



Este separador mostra os resultados dos cálculos sob forma de um gráfico e de uma tabela com os valores computados. O utilizador pode escolher mostrar ou esconder a grelha do gráfico selecionando ou desselecionando, respetivamente, a caixa no canto inferior esquerdo.

Análise Matemática II – Trabalho 3

Função real de 2 variáveis reais: $z = f(x,y)$



Neste separador é representado a superfície 3D relativa à função introduzida no menu de introdução de valores. O utilizador pode escolher mostrar ou esconder a grelha do gráfico seleccionando ou desselecccionando, respetivamente, a caixa no canto inferior esquerdo.

Na parte de baixo da aplicação, centrado, encontra-se um indicador. Este componente informa-nos se a função é harmónica ou não.

Conclusão

Este trabalho, na nossa opinião, foi bem mais desafiante do que os dois trabalhos anteriores que realizámos devido ao facto de termos tido de implementar uma grande variedade de funcionalidades numa só aplicação. Foi interessante debater e chegar a um consenso quanto à interligação dos vários elementos que formam a nossa aplicação.

Ao contrário dos outros trabalhos, em que haviam métodos que se destacavam mais que outros, neste trabalho todos os métodos se portaram muito bem. Todos as fórmulas de diferenças finitas conseguiram aproximar-se bastante bem das soluções exatas calculadas com o MATLAB. A mesma coisa se aplica às regras implementadas, como a do [Trapézio](#) e a de [Simpson](#). Ambas produziram resultados bastante interessantes.

Como sempre, a realização deste trabalho permitiu-nos aplicar tanto os nossos conhecimentos matemáticos como os conhecimentos de programação em MATLAB de modo a criar uma aplicação com uma utilidade exorbitante com a vantagem de termos adicionado os nossos toques especiais à mesma de maneira a torná-la única.

