Instituto Superior de Engenharia de Coimbra

Análise Matemática II

Atividade01Trabalho

Métodos Numéricos para Equações Diferenciais Ordinárias e Problemas de Valor Inicial

Licenciatura em Engenharia Informática

2019 / 2020

TheForgotten

someone@isec.pt

Conteúdo

1. Introdução

1.1. Enunciado da atividade proposta e interpretação do mesmo

"Pretende-se com esta atividade que os alunos adquiram conhecimentos sobre os métodos numéricos para resolução de EDO/PVI, programem esses métodos e ao mesmo tempo que desenvolvam competências algorítmicas e de programação em Matlab.

Com base nos ficheiros existentes nas sub-pastas de » ficheiros de suporte à Atividade01 e/ou ficheiros implementados nas aulas práticas sobre este assunto » **implementem os métodos numéricos:**

- **Euler** (versão beta com "alocação" de memória)
- **Euler melhorado ou modificado;**
- **Runge-Kutta de ordem 2** (RK2);
- **Runge-Kutta de ordem 4** (RK4);
- Função que utilize o **ODE45** do Matlab;
- Pesquisa de outro método."

Com este enunciado sinto que o objetivo era óbvio: familiarizar os alunos com a programação em MatLab enquanto que, ao mesmo tempo, consolidam os seus conhecimentos sobre métodos numéricos para resolução de EDO (equações diferenciais ordinárias) e PVI (problemas de valor inicial).

1.2. Definição de PVI

Problema de valor inicial ou problema de Cauchy é uma equação diferencial que se faz acompanhar do valor da função num determinado ponto. Isto é chamado de valor inicial.

A equação geral dum PVI é

$$\begin{cases} y' = f(t,y) \\ t \in [a,b] \\ y(a) = y_0 \\ n = \alpha \end{cases}, a, b, y_0 \in \mathbb{R}\ e\ \alpha \in \mathbb{N}$$

onde:

- t = intervalo de iteração;
- n = número de iterações (no caso de uso de métodos numéricos para aproximação da solução);
- y_0 = valor inicial.

2. Métodos Numéricos para Resolução de PVI

Todas as funções apresentadas (exceto ODE45) têm como parâmetros de entrada o seguinte:

- f - função em t e y que corresponde a y';
- a - limite esquerdo do intervalo de iteração;
- b - limite direito do intervalo de iteração;
- n - número de iterações a realizar;
- y0 - valor inicial.

Para além disso, todas elas começam por calcular o h (partição regular do intervalo) e obter uma matriz t que vai conter todos os valores de x para os quais pretendemos calcular a aproximação de y'.

De seguida alocam memória suficiente para uma matriz y de modo a conseguir guardar nela todas as aproximações que obtivemos com o uso do método em questão. Esta alocação é efetuada com o auxílio da função 'zeros' do MatLab, que vai "encher" o espaço que vamos utilizar na memória com 0. Isto foi realizado em todas ao invés de apenas a função do método de Euler pois melhora a performance da função.

No fim de realizar a alocação de memória atribui o valor inicial do PVI ao valor inicial da matriz y e começa finalmente a calcular as aproximações.

Por fim, todas elas devolvem y.

2.1. Método de Euler

2.1.1. Fórmulas

$$\boldsymbol{h = \frac{b-a}{n}}$$

$$\boldsymbol{y_{i+1} = y_i + hf(t_i, y_i), i = 0, 1, \dots, n-1}$$

2.1.2. Algoritmo / Função

```
function y = NEuler(f, a, b, n, y0)

h = (b - a) / n;

t = a: h: b;


y = zeros(1, n + 1);


y(1) = y0;


for i = 1: n

      y(i + 1) = y(i) + h * f(t(i), y(i));

end

end
```

Depois de realizar o que foi mencionado anteriormente, esta função entra num ciclo onde vai calcular as soluções aproximadas do PVI em questão usando o método de Euler.

Neste método a aplicação da fórmula é direta.

Este método é pouco preciso, mas é o mais fácil de aplicar.

2.2. Método de Euler Melhorado ou Modificado

2.2.1. Fórmulas

$$\boldsymbol{h = \frac{b - a}{n}}$$

$$\boldsymbol{\tilde{y}_{i+1} = y_i + hf(t_i, y_i), i = 0, 1, \dots, n - 1}$$

$$\boldsymbol{y_{i+1} = y_i + \frac{h}{2} f(t_i, y_i) + f(t_{i+1}, \tilde{y}_{i+1}), i = 0, 1, \dots, n - 1}$$

2.2.2. Algoritmo / Função

```
function y = NEuler_M(f, a, b, n, y0)

    h = (b - a) / n;

    t = a: h: b;


    y = zeros(1, n + 1);


    y(1) = y0;


    for i = 1: n

        y(i + 1) = y(i) + h * f(t(i), y(i));

        y(i + 1) = y(i) + (h/2) * (f(t(i), y(i)) + ...

                f(t(i + 1), y(i + 1)));

    end

end
```

Depois de realizar o que foi mencionado anteriormente, esta função entra num ciclo onde vai calcular as soluções aproximadas do PVI em questão usando o método de Euler melhorado ou modificado.

Neste método, para aplicar a fórmula, é preciso previamente calcular o valor intermédio. Apenas após isso é que se pode aplicar a fórmula.

Este método, apesar de ser chamado de um Euler melhorado, é muito mais preciso que o original. Na minha opinião, este é o método que oferece melhor precisão por dificuldade.

2.3. Método de Runge-Kutta de ordem 2 (RK2)

2.3.1. Fórmulas

$$\boldsymbol{h = \frac{b - a}{n}}$$

$$\boldsymbol{k_1 = hf(t_i, y_i);\ k_2 = hf(t_{i+1}, y_{i+1})}$$

$$\boldsymbol{y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2), i = 0, 1, \ldots, n - 1}$$

2.3.2. Algoritmo / Função

```
function y = NRK2(f, a, b, n, y0)
    h = (b - a) / n;
    t = a: h: b;

    y = zeros(1, n + 1);

    y(1) = y0;

    for i = 1: n
        k1 = h * f(t(i), y(i));
        k2 = h * f(t(i + 1), y(i) + k1);

        y(i + 1) = y(i) + (1/2) * (k1 + k2);
    end
end
```

Depois de realizar o que foi mencionado anteriormente, esta função entra num ciclo onde vai calcular as soluções aproximadas do PVI em questão usando o método de Runge-Kutta de ordem 2 (RK2).

Neste método, para aplicar a fórmula, é preciso previamente calcular k_1 e k_2. Apenas após a obtenção desses dois valores se pode aplicar a fórmula.

Este método é mais preciso que o método original de Euler, mas menos que a versão melhorada.

2.4. Método de Runge-Kutta de ordem 4 (RK4)

2.4.1. Fórmulas

$$\boldsymbol{h = \frac{b-a}{n}}$$

$$\boldsymbol{k_1 = hf(t_i, y_i);\ k_2 = hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right);\ k_3 = hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_2\right)}$$
$$\boldsymbol{k_4 = hf(t_{i+1}, y_i + k_3)}$$

$$\boldsymbol{y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2), i = 0, 1, \dots, n-1}$$

2.4.2. Algoritmo / Função

```
function y = NRK4(f, a, b, n, y0)

    h = (b - a) / n;

    t = a: h: b;


    y = zeros(1, n + 1);


    y(1) = y0;


    for i = 1: n

        k1 = h * f(t(i), y(i));

        k2 = h * f(t(i) + (h/2), y(i) + (1/2) * k1);

        k3 = h * f(t(i) + (h/2), y(i) + (1/2) * k2);

        k4 = h * f(t(i + 1), y(i) + k3);


        y(i + 1) = y(i) + (1/6) * (k1 + 2 * k2 + 2 * k3 +
k4);

    end

end
```

Depois de realizar o que foi mencionado anteriormente, esta função entra num ciclo onde vai calcular as soluções aproximadas do PVI em questão usando o método de Runge-Kutta de ordem 4 (RK4).

Neste método, para aplicar a fórmula, é preciso previamente calcular k_1, k_2, k_3 e k_4. Apenas após a obtenção desses dois valores se pode aplicar a fórmula.

Este método é o mais complicado de aplicar, mas é também o que oferece melhor precisão dos métodos apresentados.

2.5. Método de 2 passos Adams–Bashforth

2.5.1. Fórmulas

$$\boldsymbol{h = \frac{b - a}{n}}$$

$$\boldsymbol{y_{i+1} = y_i + hf(t_i, y_i), i = 0, 1, \dots, n - 1}$$

$$\boldsymbol{y_{i+2} = y_{i+1} + \frac{3}{2}hf(t_{i+1}, y_{i+1}) - \frac{1}{2}hf(t_i, y_i), i = 0, 1, \dots, n - 1}$$

2.5.2. Algoritmo / Função

```
function y = NAdams(f, a, b, n, y0)
    h = (b - a) / n;
    t = a: h: b;
    y(1) = y0;
    y(2) = y(1) + h * f(t(1), y(1));

    for i = 1: n - 1
        y(i + 2) = y(i + 1) + (3/2) * h * ...
           f(t(i + 1), y(i + 1)) - (1/2) * h * f(t(i),
y(i));
    end
end
```

Depois de realizar o que foi mencionado anteriormente, esta função entra num ciclo onde vai calcular as soluções aproximadas do PVI em questão usando o método de 2 passos Adams–Bashforth.

Neste método, como se trata dum método que calcula aproximações em passos de dois, temos que obter o y_{i+1} através dum método à escolha. Para este propósito, pessoalmente, escolhi o método de Euler por se tratar dum método simples mantendo, assim, a simplicidade do método. Apenas após a obtenção desse valor se pode aplicar a fórmula.

Este método é fácil de aplicar e é de fraca precisão.

Escolhi este método pois, apesar da sua fraca precisão, achei interessante a ideia de calcular as aproximações em passos de dois, ao contrário dos outros métodos lecionados que as obtêm em passos de um.

2.6. Função ODE45 do MatLab

'ODE45' trata-se duma função de MatLab cujo objetivo é resolver problemas de variável inicial.

Esta é uma de muitas as funções que o MatLab fornece para resolver PVI, sendo outras, por exemplo, 'ODE23', 'ODE113' e 'ODE15s'.

Apesar da sua enorme precisão, quando comparada com as suas "irmãs", a precisão desta é classificada como média.

Esta função é chamada da seguinte maneira

```
[x, y] = ode45(f, t, y0);
```

onde:

- f - função em t e y que corresponde a y';
- t - matriz com os valores de x para os quais pretendemos calcular a aproximação;
- y0 - valor inicial.

Isto vai devolver 2 vetores, x e y, onde:

- x - vetor com o valor das abcissas
- y - vetor com o valor das ordenadas

Como neste caso apenas pretendemos obter o vetor com o valor das ordenadas, ou seja, o vetor que contém as aproximações, podemos chamar a função da seguinte forma:

```
[~, y] = ode45(f, t, y0);
```

Assim, conseguimos obter o vetor que pretendemos sem ter que guardar dados que não precisamos.

Apesar de ser bastante fácil a aplicação desta função ela levanta um pequeno problema derivado do facto que ela devolve o vetor y como uma coluna em vez duma linha. Mesmo que fosse fácil moldar o meu código para converter todas as matrizes linha menos a matriz do ODE45 para matrizes colunas. Mas, de modo a manter uma certa coerência, achei por bem inserir a seguinte linha a seguir à chamada da função:

```
y = y.';
```

Deste modo podemos manter o nosso código mais coerente e mais conciso contornando este pequeno problema com apenas uma simples linha de código.

3. Exemplos de aplicação e teste dos métodos

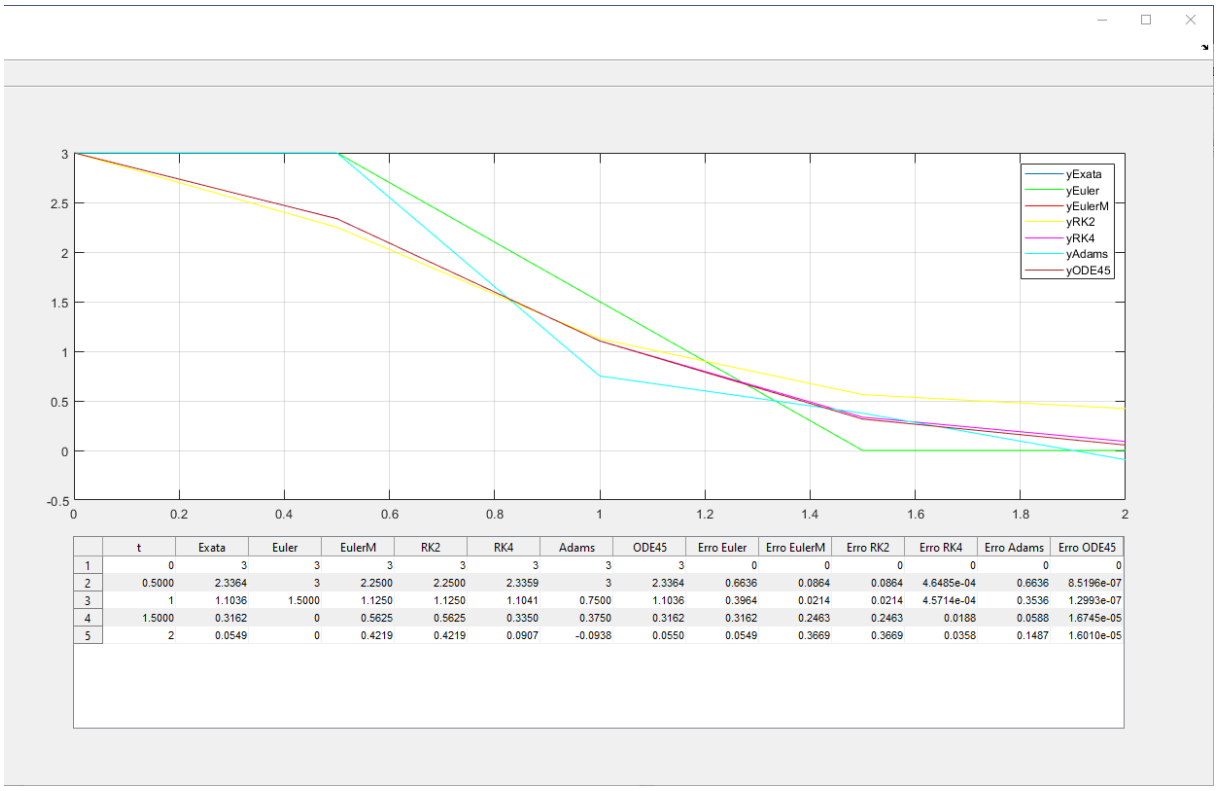
3.1. Exercício 4 de um teste A de 2015 / 2016

O teste escolhido foi "TesteIntercalarA_v1_AM2_1516_INFECS". Apenas foi resolvida a alínea c) por achar que é a única que se adequa.

3.1.1. PVI – Equação Diferencial de 1ª Ordem e Condições Iniciais

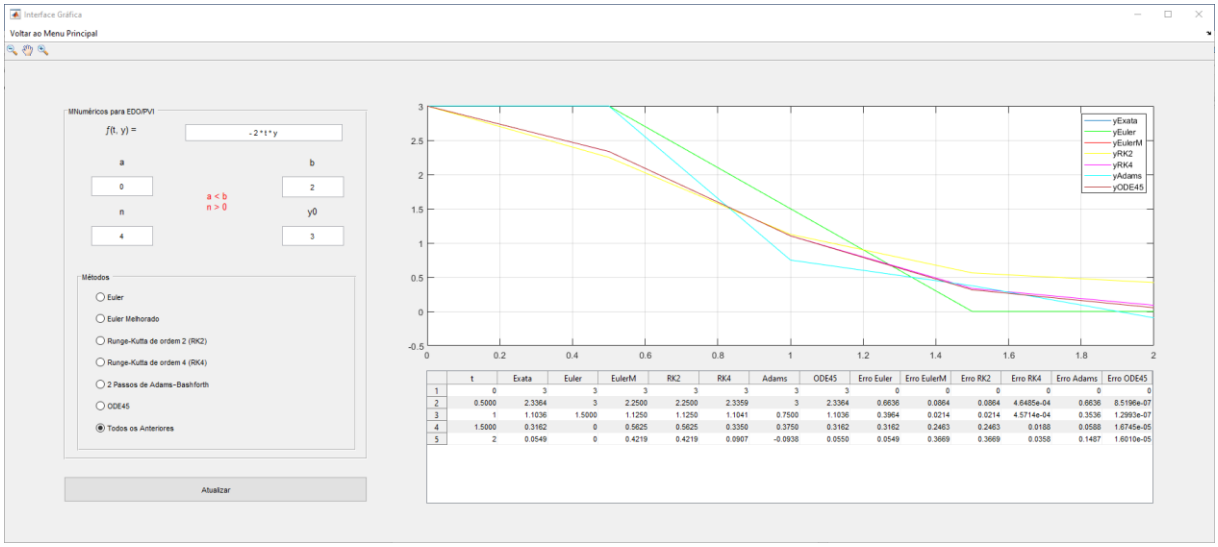
$$\begin{cases} y' + 2ty = 0 \\ t \in [0,2] \\ y(0) = 3 \\ n = 4 \end{cases} \Leftrightarrow \begin{cases} y' = -2ty \\ t \in [0,2] \\ y(0) = 3 \\ n = 4 \end{cases}$$

3.1.2. Exemplos de output – GUI com gráfico e tabela

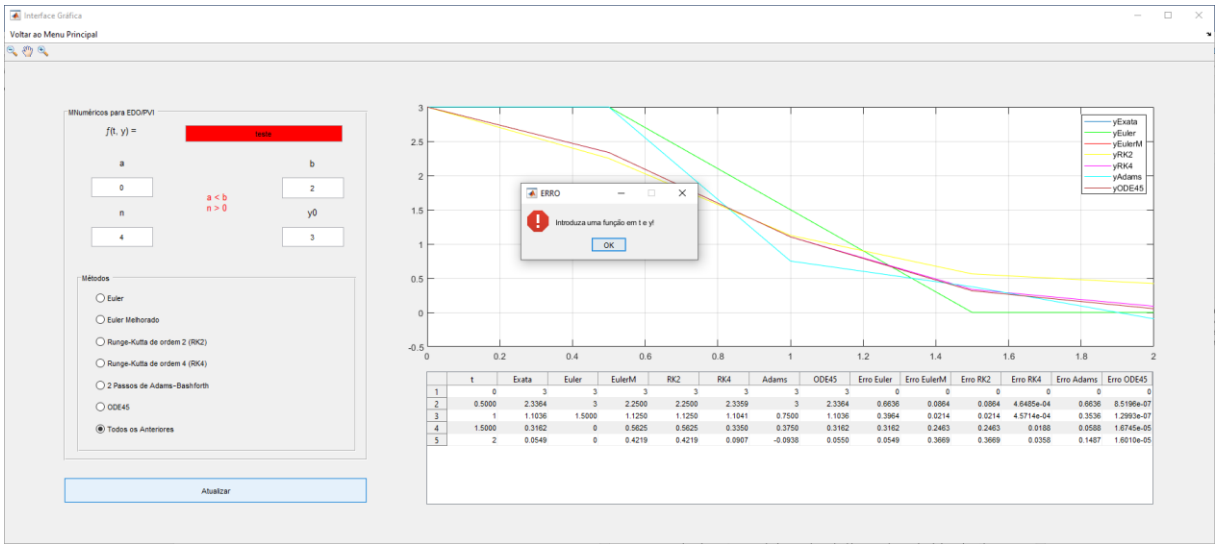


	t	Exata	Euler	EulerM	RK2	RK4	Adams	ODE45	Erro Euler	Erro EulerM	Erro RK2	Erro RK4	Erro Adams	Erro ODE45
1	0	3	3	3	3	3	3	3	0	0	0	0	0	0
2	0.5000	2.3364	3	2.2500	2.2500	2.3359	3	2.3364	0.6636	0.0864	0.0864	4.6485e-04	0.6636	8.5196e-07
3	1	1.1036	1.5000	1.1250	1.1250	1.1041	0.7500	1.1036	0.3964	0.0214	0.0214	4.5714e-04	0.3536	1.2993e-07
4	1.5000	0.3162	0	0.5625	0.5625	0.3350	0.3750	0.3162	0.3162	0.2463	0.2463	0.0188	0.0588	1.6745e-05
5	2	0.0549	0	0.4219	0.4219	0.0907	-0.0938	0.0550	0.0549	0.3669	0.3669	0.0358	0.1487	1.6010e-05

Em cima pode ser visto o gráfico e a tabela obtida pelo programa que desenvolvi. Esta captura de ecrã é um recorte da original para facilitar a visualização dos dados e do gráfico obtido. A captura completa pode ser vista na imagem seguinte.



Aqui podemos visualizar a GUI completa com a introdução dos dados fornecidos pelo PVI nos respetivos espaços. Todos os espaços têm as respetivas validações para os parâmetros inseridos e, quando algo de errado acontece, recebemos uma mensagem de erro a dizer qual é o problema e a caixa de texto onde ele acontece é realçada a vermelho, como podemos visualizar em baixo.



3.2. Problema de aplicação (sugerido pelo professor)

3.2.1. Modelação Matemática do Problema

Exercício 1:

$$\begin{cases} m\frac{dv}{dt} = mg - kv^2, k > 0 \\ k = 0.125 \\ m = 5 \\ g = 32ft/s^2 \\ h = 1 \\ v(0) = 0 \end{cases} \Leftrightarrow \begin{cases} m\frac{dv}{dt} = mg - kv^2, k > 0 \\ k = 0.125 \\ m = 5 \\ g = 32ft/s^2 \\ h = 1 \\ v(0) = 0 \end{cases} \Leftrightarrow$$

$$\begin{cases} v' = g - \frac{kv^2}{m}, k > 0 \\ k = 0.125 \\ m = 5 \\ g = 32ft/s^2 \\ h = 1 \\ v(0) = 0 \end{cases} \Leftrightarrow \begin{cases} v' = 32 - \frac{0.125v^2}{5} \\ h = 1 \\ v(0) = 0 \end{cases} \Leftrightarrow \begin{cases} v' = 32 - \frac{1}{40}v^2, k > 0 \\ h = 1 \\ v(0) = 0 \end{cases}$$

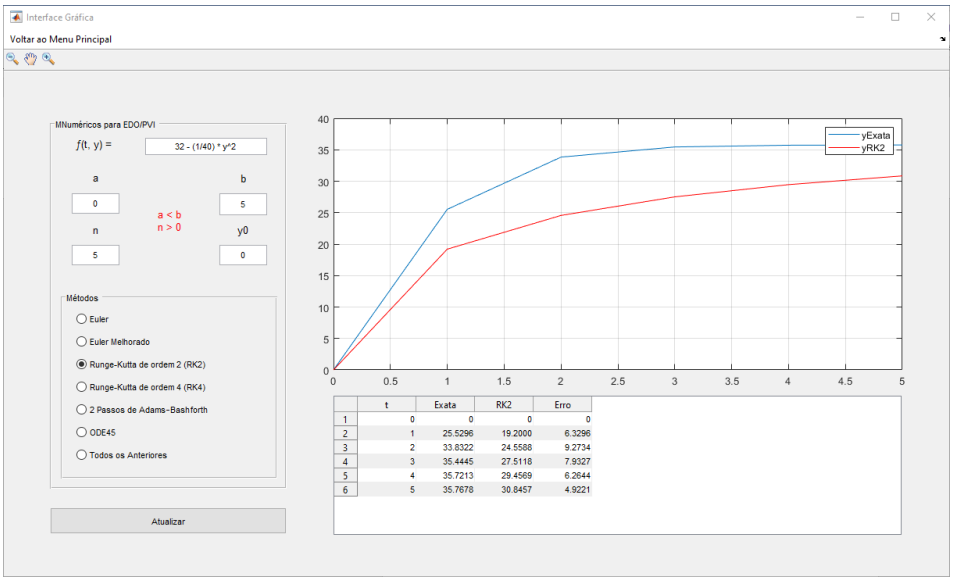
Exercício 2:

$$\begin{cases} \frac{dA}{dt} = A(2.128 - 0.0432A) \\ h = 0.5 \\ A(0) = 0.24cm^2 \end{cases} \Leftrightarrow \begin{cases} A' = A(2.128 - 0.0432A) \\ h = 0.5 \\ A(0) = 0.24cm^2 \end{cases}$$

3.2.2. Resolução Através da Aplicação Criada

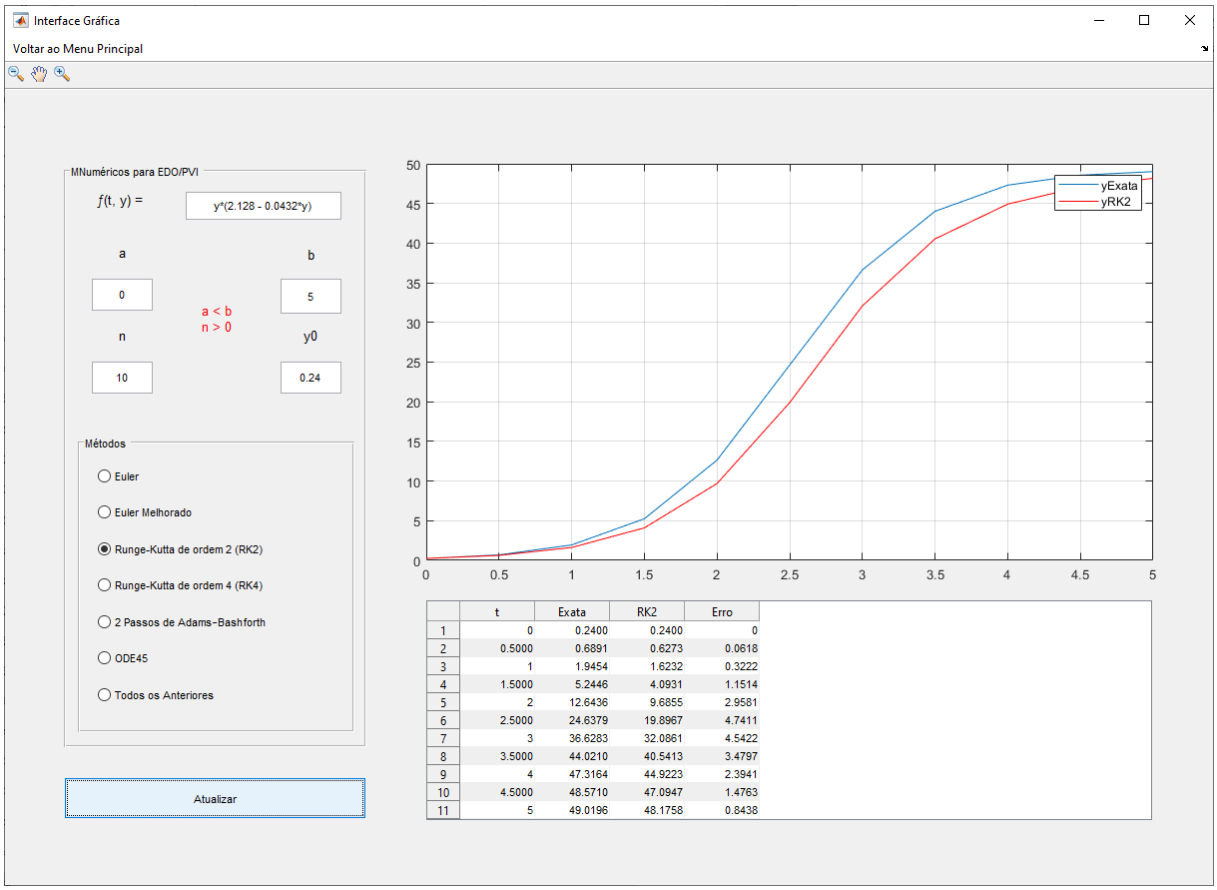
Exercício 1:

Neste exercício é pedido que usemos $h = 1$. Para este efeito, decidi fazer o intervalo $t = [0,5]$ com $n = 5$. No enunciado pede para usarmos um método de Runge-Kutta sem especificar qual. Decidi usar o método de ordem 2. Deste modo, obtemos a seguinte solução:



Exercício 2:

Neste exercício é pedido que usemos $h = 0.5$. Para este efeito, decidi fazer o intervalo $t = [0,5]$ com $n = 10$. No enunciado pede para usarmos um método de Runge-Kutta sem especificar qual. Decidi usar o método de ordem 2. Deste modo, obtemos a seguinte solução:



	t	Exata	RK2	Erro
1	0	0.2400	0.2400	0
2	0.5000	0.6891	0.6273	0.0618
3	1	1.9454	1.6232	0.3222
4	1.5000	5.2446	4.0931	1.1514
5	2	12.6436	9.6855	2.9581
6	2.5000	24.6379	19.8967	4.7411
7	3	36.6283	32.0861	4.5422
8	3.5000	44.0210	40.5413	3.4797
9	4	47.3164	44.9223	2.3941
10	4.5000	48.5710	47.0947	1.4763
11	5	49.0196	48.1758	0.8438

Apesar de neste exercício a tabela oferecer valores para $t = 0, 0.5, 1$ ..., apenas devem ser vistos os valores inteiros, ou seja, $t = 0, 1, 2$ ...

Isto acontece devido ao facto que o exercício exige que façamos as iterações de 0.5 em 0.5, ou seja, de meio em meio dia, querendo apenas contabilizar a área no final de cada dia.

4. Conclusão

Após a realização deste trabalho sinto-me muito mais à vontade com o uso de métodos numéricos e a aplicação dos mesmos. Para além disso, também aprendi bastante sobre programação em MatLab e a lidar com os pequenos desafios e limitações que esta linguagem de programação nos proporciona.

Apesar de ter sido um trabalho demoroso, sinto que valeu a pena cada segundo posto devido ao facto de ter consolidado os meus conhecimentos tanto sobre os métodos como sobre a programação.

Mesmo com todos estes métodos e havendo uns que são mais precisos que outros, considero que não podemos esquecer-nos que nem todos são realistas de serem aplicados em papel. Tal como mencionei anteriormente, para mim, o melhor método para ser aplicar em papel é o método de Euler melhorado. Tendo em conta a facilidade de calcular as aproximações com o mesmo, este oferece uma excelente precisão.