Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE / DEPARTMENT OF
INFORMÁTICA E SISTEMAS

Métodos Numéricos para resolução de Sistemas de ED

Relatório de Licenciatura em / in Engenharia Informática

Autor / Author

António Domingos Gonçalves Pedroso

António Miguel Grangeiro Rocha

Samuel Frazão Pinto Costa

Supervisor

Arménio António da Silva Correia

Coimbra, Maio 2025

INSTITUTO POLITÉCNICO DE
COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Resumo

Este relatório documenta o desenvolvimento da Atividade 4 da unidade curricular de Análise Matemática 2, focada nos métodos numéricos para a resolução de Sistemas de Equações Diferenciais (SED). São abordados os métodos de Euler, Euler Melhorado (Heun), Runge-Kutta de ordem 2 (RK2) e Runge-Kutta de ordem 4 (RK4), bem como a função `ode45` do Matlab e um método numérico adicional pesquisado. Para cada método, é apresentada a formulação matemática, implementada em Matlab, e analisada a sua eficácia, precisão e aplicabilidade na resolução de SED com condições iniciais. Foram considerados problemas de aplicação, como sistemas mecânicos (mola-massa com e sem amortecimento), circuitos elétricos e o modelo do pêndulo, comparando, sempre que possível, os resultados numéricos com a solução exata. A atividade consolidou competências teóricas e práticas na resolução numérica de SED, destacando a relevância destas ferramentas em modelação matemática e simulação computacional.

Abstract

This report documents the development of Activity 4 from the course unit Mathematical Analysis 2, centered on numerical methods for solving Systems of Differential Equations (SDEs). The methods covered include Euler's method, the improved Euler method (Heun), Runge-Kutta methods of order 2 (RK2) and order 4 (RK4), the `ode45` solver from Matlab, and an additional researched numerical method. Each method is presented with its mathematical formulation, implemented in Matlab, and evaluated for effectiveness, accuracy, and applicability in solving SDEs with initial conditions. Practical applications, such as mechanical systems (mass-spring with and without damping), electrical circuits, and the pendulum model, were analyzed, comparing numerical results with exact solutions whenever feasible. This activity reinforced theoretical and practical skills in the numerical resolution of SDEs, emphasizing their significance in mathematical modeling and computational simulation.

Epígrafe

Tzzzzz Tzzzzzzz Tzzzzzz.
Arménio, 2025

Agradecimentos

Arménio António da Silva Correia

Rui Manuel Carreira Rodrigues

Índice

Índice de figuras

Lista de Abreviaturas

a	Limite inferior do intervalo
b	Limite superior do intervalo
$f(t,u,v)$	Função que define a primeira equação do SED, ou seja, $u' = f(t,u,v)$
$g(t,u,v)$	Função que define a segunda equação do SED, ou seja, $v' = g(t,u,v)$
h	Tamanho do passo (step size)
IEEE	*Institute of Electrical and Electronics Engineers*
ISEC	Instituto Superior de Engenharia de Coimbra
$k_{1u}, k_{1v}, k_{2u}, k_{2v}, k_{3u}, k_{3v}, k_{4u}, k_{4v}$	Avaliações intermediárias das funções do SED
MATLAB	*Matrix Laboratory*
`ode45`	Função do MATLAB para resolver SED com método adaptativo
PVI	Problema de Valor Inicial
RK2	Método de Runge-Kutta de Ordem 2
RK3	Método de Runge-Kutta de Ordem 3
RK4	Método de Runge-Kutta de Ordem 4
RK45	Método de Runge-Kutta de Ordem 4(5), usado pelo `ode45`
SED	Sistema de Equações Diferenciais
t	Variável independente (tempo)
u	Primeira variável dependente (componente da solução do SED)
u_0	Condição inicial da primeira variável dependente
v	Segunda variável dependente (componente da solução do SED)
v_0	Condição inicial da segunda variável dependente

1 Introdução

No presente documento, será registado todo o processo envolvido na realização da Atividade 4: Métodos Numéricos para Sistemas de Equações Diferenciais (SED), proposta no âmbito da Unidade Curricular Análise Matemática 2.

Os assuntos abordados neste relatório baseiam-se, principalmente, nos conhecimentos transmitidos nas aulas teórico-práticas das 1ª e 2ª semanas de abril, sendo complementados com pesquisa e trabalho realizado fora do ambiente letivo.

Ao longo deste documento, será possível acompanhar o desenvolvimento da atividade, incluindo a implementação em Matlab dos métodos numéricos, a análise de problemas de aplicação como sistemas mecânicos, circuitos elétricos e o modelo do pêndulo, as dificuldades enfrentadas e a descrição detalhada de todas as suas componentes.

2 Sistemas de Equações Diferenciais (SED)

O estudo de Sistemas de Equações Diferenciais (SED) centra-se na resolução de sistemas da forma:

$$\begin{cases} \mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t)) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases} \tag{2.1}$$

em que $\mathbf{x}(t)$ é um vetor de funções e $\mathbf{f}$ é uma função vetorial que descreve a dinâmica do sistema. O objetivo é determinar o vetor $\mathbf{x}(t)$ para $t > t_0$, dado o valor inicial $\mathbf{x}_0$ no instante t_0. Estes sistemas são amplamente utilizados na modelação de fenómenos complexos, como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento), circuitos elétricos e o movimento de pêndulos, entre outros.

Na prática, a solução analítica de SEDs é frequentemente inviável, especialmente para sistemas não lineares ou de maior dimensão, pelo que se recorre a métodos numéricos para obter aproximações das soluções.

3 Método de Euler

Fórmulas

O Método de Euler é um procedimento numérico de primeira ordem para aproximar a solução de Sistemas de Equações Diferenciais (SED) com condições iniciais, da forma:

$$\begin{cases} u'(t) = f(t, u(t), v(t)) \\ v'(t) = g(t, u(t), v(t)) \\ u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases} \tag{3.1}$$

em que $u(t)$ e $v(t)$ são as componentes do vetor solução $\mathbf{x}(t) = [u(t), v(t)]$, e f e g são funções que definem a dinâmica do sistema. O objetivo é determinar $u(t)$ e $v(t)$ para $t > t_0$, dadas as condições iniciais u_0 e v_0. Estes sistemas modelam fenómenos complexos, como sistemas mecânicos (e.g., mola-massa) ou circuitos elétricos.

O Método de Euler para SED é dado pelas seguintes fórmulas gerais:

$$u_{i+1} = u_i + hf(t_i, u_i, v_i) \tag{3.2}$$

$$v_{i+1} = v_i + hg(t_i, u_i, v_i) \tag{3.3}$$

onde:

- u_{i+1}, v_{i+1} → Próximos valores aproximados das componentes da solução na abscissa t_{i+1};
- u_i, v_i → Valores aproximados das componentes da solução na abscissa atual t_i;
- h → Tamanho do passo (subintervalo);
- $f(t_i, u_i, v_i), g(t_i, u_i, v_i)$ → Valores das funções do sistema em t_i, u_i e v_i.

Algoritmo/Função

Algoritmo:

1. Definir o intervalo $[a, b]$, o número de passos n, e calcular o tamanho do passo $h = \frac{b-a}{n}$;

2. Criar vetores t, u e v para armazenar os pontos do intervalo e as soluções, atribuindo $u(1) = u_0$ e $v(1) = v_0$;
3. Para cada i de 1 a n, calcular:
   - $u_{i+1} = u_i + hf(t_i, u_i, v_i)$;
   - $v_{i+1} = v_i + hg(t_i, u_i, v_i)$;
4. Retornar os vetores t, u e v com os resultados.

A implementação em Matlab do Método de Euler para SED é apresentada abaixo:

```
function [t, u, v] = NEulerSED(~, f, g, a, b, n, u0, v0)
    h = (b-a)/n;
    t = a:h:b;
    u = zeros(1, n+1);
    v = zeros(1, n+1);
    u(1) = u0;
    v(1) = v0;
    for i = 1:n
        u(i+1) = u(i) + h * f(t(i), u(i), v(i));
        v(i+1) = v(i) + h * g(t(i), u(i), v(i));
    end
end
```

4 Método de Euler Melhorado ou Modificado (Método de Heun)

Fórmulas

O Método de Euler Melhorado, também conhecido como Método de Heun, é um procedimento numérico equivalente a um método de Runge-Kutta de ordem 2, utilizado para aproximar a solução de Sistemas de Equações Diferenciais (SED) com condições iniciais, da forma:

$$\begin{cases} u'(t) = f(t, u(t), v(t)) \\ v'(t) = g(t, u(t), v(t)) \\ u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases} \tag{4.1}$$

em que $u(t)$ e $v(t)$ são as componentes do vetor solução $\mathbf{x}(t) = [u(t), v(t)]$, e f e g são funções que definem a dinâmica do sistema. O objetivo é determinar $u(t)$ e $v(t)$ para $t > t_0$, dadas as condições iniciais u_0 e v_0. Estes sistemas são aplicados em modelações como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento) ou circuitos elétricos.

O Método de Heun para SED é dado pelas seguintes fórmulas gerais:

$$u_{i+1} = u_i + \frac{h}{2}(k_{1u} + k_{2u}) \tag{4.2}$$

$$v_{i+1} = v_i + \frac{h}{2}(k_{1v} + k_{2v}) \tag{4.3}$$

Cálculo de k_{1u} e k_{1v}:

$$k_{1u} = f(t_i, u_i, v_i) \tag{4.4}$$

$$k_{1v} = g(t_i, u_i, v_i) \tag{4.5}$$

Cálculo de k_{2u} e k_{2v}:

$$k_{2u} = f(t_i + h, u_i + hk_{1u}, v_i + hk_{1v}) \tag{4.6}$$

$$k_{2v} = g(t_i + h, u_i + hk_{1u}, v_i + hk_{1v}) \tag{4.7}$$

onde:

- u_{i+1}, v_{i+1} → Próximos valores aproximados das componentes da solução na abscissa t_{i+1};
- u_i, v_i → Valores aproximados das componentes da solução na abscissa atual t_i;
- h → Tamanho do passo (subintervalo);
- k_{1u}, k_{1v} → Inclinações no início do intervalo para u e v;
- k_{2u}, k_{2v} → Inclinações no fim do intervalo para u e v;
- $f(t_i, u_i, v_i), g(t_i, u_i, v_i)$ → Valores das funções do sistema em t_i, u_i e v_i.

Algoritmo/Função

Algoritmo:

1. Definir o intervalo $[a, b]$, o número de passos n, e calcular o tamanho do passo $h = \frac{b-a}{n}$;
2. Criar vetores t, u e v para armazenar os pontos do intervalo e as soluções, atribuindo $u(1) = u_0$ e $v(1) = v_0$;
3. Para cada i de 1 a n, calcular:
   - $k_{1u} = f(t_i, u_i, v_i)$, $k_{1v} = g(t_i, u_i, v_i)$;
   - $k_{2u} = f(t_i + h, u_i + hk_{1u}, v_i + hk_{1v})$;
   - $k_{2v} = g(t_i + h, u_i + hk_{1u}, v_i + hk_{1v})$;
   - $u_{i+1} = u_i + \frac{h}{2}(k_{1u} + k_{2u})$;
   - $v_{i+1} = v_i + \frac{h}{2}(k_{1v} + k_{2v})$;
4. Retornar os vetores t, u e v com os resultados.

A implementação em MATLAB do Método de Euler Melhorado para SED é apresentada abaixo:

```
function [t, u, v] = NEulerMSED(~, f, g, a, b, n, u0, v0)
    h = (b-a)/n;
    t = a:h:b;
    u = zeros(1, n+1);
    v = zeros(1, n+1);
    u(1) = u0;
    v(1) = v0;
    for i = 1:n
        k1u = f(t(i), u(i), v(i));
        k1v = g(t(i), u(i), v(i));
```

```
        k2u = f(t(i+1), u(i) + h*k1u, v(i) + h*k1v);
        k2v = g(t(i+1), u(i) + h*k1u, v(i) + h*k1v);
        u(i+1) = u(i) + h*(k1u + k2u)/2;
        v(i+1) = v(i) + h*(k1v + k2v)/2;
    end
end
```

5 Método de Runge-Kutta de Ordem 2

Fórmulas

O Método de Runge-Kutta de Ordem 2 (RK2) é um método numérico de passo simples que utiliza derivadas de primeira ordem para fornecer aproximações precisas para a solução de Sistemas de Equações Diferenciais (SED) com condições iniciais, da forma:

$$\begin{cases} u'(t) = f(t, u(t), v(t)) \\ v'(t) = g(t, u(t), v(t)) \\ u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases} \tag{5.1}$$

em que $u(t)$ e $v(t)$ são as componentes do vetor solução $\mathbf{x}(t) = [u(t), v(t)]$, e f e g são funções que definem a dinâmica do sistema. O objetivo é determinar $u(t)$ e $v(t)$ para $t > t_0$, dadas as condições iniciais u_0 e v_0. Estes sistemas modelam fenómenos complexos, como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento) ou circuitos elétricos.

O Método de Runge-Kutta de Ordem 2 para SED é dado pelas seguintes fórmulas gerais:

$$u_{i+1} = u_i + \frac{1}{2}(k_{1u} + k_{2u}) \tag{5.2}$$

$$v_{i+1} = v_i + \frac{1}{2}(k_{1v} + k_{2v}) \tag{5.3}$$

Cálculo de k_{1u} e k_{1v}:

$$k_{1u} = hf(t_i, u_i, v_i) \tag{5.4}$$

$$k_{1v} = hg(t_i, u_i, v_i) \tag{5.5}$$

Cálculo de k_{2u} e k_{2v}:

$$k_{2u} = hf(t_i + h, u_i + k_{1u}, v_i + k_{1v}) \tag{5.6}$$

$$k_{2v} = hg(t_i + h, u_i + k_{1u}, v_i + k_{1v}) \tag{5.7}$$

onde:

- u_{i+1}, v_{i+1} → Próximos valores aproximados das componentes da solução na abscissa t_{i+1};

- $u_i, v_i \rightarrow$ Valores aproximados das componentes da solução na abscissa atual t_i;
- $h \rightarrow$ Tamanho do passo (subintervalo);
- $k_{1u}, k_{1v} \rightarrow$ Inclinações no início do intervalo para u e v;
- $k_{2u}, k_{2v} \rightarrow$ Inclinações no fim do intervalo para u e v;
- $f(t_i, u_i, v_i), g(t_i, u_i, v_i) \rightarrow$ Valores das funções do sistema em t_i, u_i e v_i.

Algoritmo/Função

Algoritmo:

1. Definir o intervalo $[a, b]$, o número de passos n, e calcular o tamanho do passo $h = \frac{b-a}{n}$;
2. Criar vetores t, u e v para armazenar os pontos do intervalo e as soluções, atribuindo $u(1) = u_0$ e $v(1) = v_0$;
3. Para cada i de 1 a n, calcular:
   - $k_{1u} = hf(t_i, u_i, v_i)$, $k_{1v} = hg(t_i, u_i, v_i)$;
   - $k_{2u} = hf(t_i + h, u_i + k_{1u}, v_i + k_{1v})$;
   - $k_{2v} = hg(t_i + h, u_i + k_{1u}, v_i + k_{1v})$;
   - $u_{i+1} = u_i + \frac{1}{2}(k_{1u} + k_{2u})$;
   - $v_{i+1} = v_i + \frac{1}{2}(k_{1v} + k_{2v})$;
4. Retornar os vetores t, u e v com os resultados.

A implementação em MATLAB do Método de Runge-Kutta de Ordem 2 para SED é apresentada abaixo:

```
function [t, u, v] = NRK2SED(~, f, g, a, b, n, u0, v0)
    h = (b-a)/n;
    t = a:h:b;
    u = zeros(1, n+1);
    v = zeros(1, n+1);
    u(1) = u0;
    v(1) = v0;
    for i = 1:n
        k1u = h*f(t(i), u(i), v(i));
        k1v = h*g(t(i), u(i), v(i));
        k2u = h*f(t(i+1), u(i)+k1u, v(i)+k1v);
        k2v = h*g(t(i+1), u(i)+k1u, v(i)+k1v);
```

```
        u(i+1) = u(i)+(k1u+k2u)/2;
        v(i+1) = v(i)+(k1v+k2v)/2;
    end
end
```

6 Método de Runge-Kutta de Ordem 3

Fórmulas

O Método de Runge-Kutta de Ordem 3 (RK3) é um método numérico que fornece uma aproximação mais precisa em relação ao RK2, utilizando três avaliações da função por iteração para resolver Sistemas de Equações Diferenciais (SED) com condições iniciais, da forma:

$$\begin{cases} u'(t) = f(t, u(t), v(t)) \\ v'(t) = g(t, u(t), v(t)) \\ u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases} \tag{6.1}$$

em que $u(t)$ e $v(t)$ são as componentes do vetor solução $\mathbf{x}(t) = [u(t), v(t)]$, e f e g são funções que definem a dinâmica do sistema. O objetivo é determinar $u(t)$ e $v(t)$ para $t > t_0$, dadas as condições iniciais u_0 e v_0. Estes sistemas modelam fenómenos complexos, como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento) ou circuitos elétricos.

O Método de Runge-Kutta de Ordem 3 para SED é dado pelas seguintes fórmulas gerais:

$$u_{i+1} = u_i + \frac{1}{6}(k_{1u} + 4k_{2u} + k_{3u}) \tag{6.2}$$

$$v_{i+1} = v_i + \frac{1}{6}(k_{1v} + 4k_{2v} + k_{3v}) \tag{6.3}$$

Cálculo de k_{1u} e k_{1v}:

$$k_{1u} = hf(t_i, u_i, v_i) \tag{6.4}$$

$$k_{1v} = hg(t_i, u_i, v_i) \tag{6.5}$$

Cálculo de k_{2u} e k_{2v}:

$$k_{2u} = hf\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right) \tag{6.6}$$

$$k_{2v} = hg\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right) \tag{6.7}$$

Cálculo de k_{3u} e k_{3v}:

$$k_{3u} = hf(t_i + h, u_i - k_{1u} + 2k_{2u}, v_i - k_{1v} + 2k_{2v}) \tag{6.8}$$

$$k_{3v} = hg(t_i + h, u_i - k_{1u} + 2k_{2u}, v_i - k_{1v} + 2k_{2v}) \tag{6.9}$$

onde:

- u_{i+1}, v_{i+1} → Próximos valores aproximados das componentes da solução na abscissa t_{i+1};
- u_i, v_i → Valores aproximados das componentes da solução na abscissa atual t_i;
- h → Tamanho do passo (subintervalo);
- k_{1u}, k_{1v} → Inclinações no início do intervalo para u e v;
- k_{2u}, k_{2v} → Inclinações no ponto médio do intervalo para u e v;
- k_{3u}, k_{3v} → Inclinações no fim do intervalo para u e v;
- $f(t_i, u_i, v_i), g(t_i, u_i, v_i)$ → Valores das funções do sistema em t_i, u_i e v_i.

Algoritmo/Função

Algoritmo:

1. Definir o intervalo $[a, b]$, o número de passos n, e calcular o tamanho do passo $h = \frac{b-a}{n}$;
2. Criar vetores t, u e v para armazenar os pontos do intervalo e as soluções, atribuindo $u(1) = u_0$ e $v(1) = v_0$;
3. Para cada i de 1 a n, calcular:
   - $k_{1u} = hf(t_i, u_i, v_i)$, $k_{1v} = hg(t_i, u_i, v_i)$;
   - $k_{2u} = hf\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right)$;
   - $k_{2v} = hg\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right)$;
   - $k_{3u} = hf(t_i + h, u_i - k_{1u} + 2k_{2u}, v_i - k_{1v} + 2k_{2v})$;
   - $k_{3v} = hg(t_i + h, u_i - k_{1u} + 2k_{2u}, v_i - k_{1v} + 2k_{2v})$;
   - $u_{i+1} = u_i + \frac{1}{6}(k_{1u} + 4k_{2u} + k_{3u})$;
   - $v_{i+1} = v_i + \frac{1}{6}(k_{1v} + 4k_{2v} + k_{3v})$;
4. Retornar os vetores t, u e v com os resultados.

A implementação em MATLAB do Método de Runge-Kutta de Ordem 3 para SED é apresentada abaixo:

```
function [t, u, v] = NRK3SED(~, f, g, a, b, n, u0, v0)
    h = (b-a)/n;
    t = a:h:b;
    u = zeros(1, n+1);
    v = zeros(1, n+1);
    u(1) = u0;
    v(1) = v0;
    for i = 1:n
        k1u = h*f(t(i), u(i), v(i));
        k1v = h*g(t(i), u(i), v(i));
        k2u = h*f(t(i)+h/2, u(i)+k1u/2, v(i)+k1v/2);
        k2v = h*g(t(i)+h/2, u(i)+k1u/2, v(i)+k1v/2);
        k3u = h*f(t(i)+h, u(i)-k1u+2*k2u, v(i)-k1v+2*k2v);
        k3v = h*g(t(i)+h, u(i)-k1u+2*k2u, v(i)-k1v+2*k2v);
        u(i+1) = u(i) + (k1u + 4*k2u + k3u)/6;
        v(i+1) = v(i) + (k1v + 4*k2v + k3v)/6;
    end
end
```

7 Método de Runge-Kutta de Ordem 4

Fórmulas

O Método de Runge-Kutta de Ordem 4 (RK4) é um dos métodos mais utilizados para resolver Sistemas de Equações Diferenciais (SED) devido à sua alta precisão e simplicidade de implementação. O método utiliza quatro avaliações da função por iteração para aproximar a solução de SED com condições iniciais, da forma:

$$\begin{cases} u'(t) = f(t, u(t), v(t)) \\ v'(t) = g(t, u(t), v(t)) \\ u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases} \tag{7.1}$$

em que $u(t)$ e $v(t)$ são as componentes do vetor solução $\mathbf{x}(t) = [u(t), v(t)]$, e f e g são funções que definem a dinâmica do sistema. O objetivo é determinar $u(t)$ e $v(t)$ para $t > t_0$, dadas as condições iniciais u_0 e v_0. Estes sistemas modelam fenómenos complexos, como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento) ou circuitos elétricos.

O Método de Runge-Kutta de Ordem 4 para SED é dado pelas seguintes fórmulas gerais:

$$u_{i+1} = u_i + \frac{1}{6}(k_{1u} + 2k_{2u} + 2k_{3u} + k_{4u}) \tag{7.2}$$

$$v_{i+1} = v_i + \frac{1}{6}(k_{1v} + 2k_{2v} + 2k_{3v} + k_{4v}) \tag{7.3}$$

Cálculo de k_{1u} e k_{1v}:

$$k_{1u} = hf(t_i, u_i, v_i) \tag{7.4}$$

$$k_{1v} = hg(t_i, u_i, v_i) \tag{7.5}$$

Cálculo de k_{2u} e k_{2v}:

$$k_{2u} = hf\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right) \tag{7.6}$$

$$k_{2v} = hg\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right) \tag{7.7}$$

Cálculo de k_{3u} e k_{3v}:

$$k_{3u} = hf\left(t_i + \frac{h}{2}, u_i + \frac{k_{2u}}{2}, v_i + \frac{k_{2v}}{2}\right) \tag{7.8}$$

$$k_{3v} = hg\left(t_i + \frac{h}{2}, u_i + \frac{k_{2u}}{2}, v_i + \frac{k_{2v}}{2}\right) \tag{7.9}$$

Cálculo de k_{4u} e k_{4v}:

$$k_{4u} = hf(t_i + h, u_i + k_{3u}, v_i + k_{3v}) \tag{7.10}$$

$$k_{4v} = hg(t_i + h, u_i + k_{3u}, v_i + k_{3v}) \tag{7.11}$$

onde:

- u_{i+1}, v_{i+1} → Próximos valores aproximados das componentes da solução na abscissa t_{i+1};
- u_i, v_i → Valores aproximados das componentes da solução na abscissa atual t_i;
- h → Tamanho do passo (subintervalo);
- k_{1u}, k_{1v} → Inclinações no início do intervalo para u e v;
- k_{2u}, k_{2v} → Inclinações no primeiro ponto médio do intervalo para u e v;
- k_{3u}, k_{3v} → Inclinações no segundo ponto médio do intervalo para u e v;
- k_{4u}, k_{4v} → Inclinações no fim do intervalo para u e v;
- $f(t_i, u_i, v_i), g(t_i, u_i, v_i)$ → Valores das funções do sistema em t_i, u_i e v_i.

Algoritmo/Função

Algoritmo:

1. Definir o intervalo $[a, b]$, o número de passos n, e calcular o tamanho do passo $h = \frac{b-a}{n}$;
2. Criar vetores t, u e v para armazenar os pontos do intervalo e as soluções, atribuindo $u(1) = u_0$ e $v(1) = v_0$;
3. Para cada i de 1 a n, calcular:
   - $k_{1u} = hf(t_i, u_i, v_i)$, $k_{1v} = hg(t_i, u_i, v_i)$;
   - $k_{2u} = hf\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right)$;
   - $k_{2v} = hg\left(t_i + \frac{h}{2}, u_i + \frac{k_{1u}}{2}, v_i + \frac{k_{1v}}{2}\right)$;
   - $k_{3u} = hf\left(t_i + \frac{h}{2}, u_i + \frac{k_{2u}}{2}, v_i + \frac{k_{2v}}{2}\right)$;

- $k_{3v} = hg\left(t_i + \frac{h}{2}, u_i + \frac{k_{2u}}{2}, v_i + \frac{k_{2v}}{2}\right)$;
- $k_{4u} = hf(t_i + h, u_i + k_{3u}, v_i + k_{3v})$;
- $k_{4v} = hg(t_i + h, u_i + k_{3u}, v_i + k_{3v})$;
- $u_{i+1} = u_i + \frac{1}{6}(k_{1u} + 2k_{2u} + 2k_{3u} + k_{4u})$;
- $v_{i+1} = v_i + \frac{1}{6}(k_{1v} + 2k_{2v} + 2k_{3v} + k_{4v})$;

4. Retornar os vetores t, u e v com os resultados.

A implementação em MATLAB do Método de Runge-Kutta de Ordem 4 para SED é apresentada abaixo:

```
function [t, u, v] = NRK4SED(~, f, g, a, b, n, u0, v0)
    h = (b-a)/n;
    t = a:h:b;
    u = zeros(1,n+1);
    v = zeros(1,n+1);
    u(1) = u0;
    v(1) = v0;
    for i = 1:n
        k1u = h*f(t(i), u(i), v(i));
        k1v = h*g(t(i), u(i), v(i));
        k2u = h*f(t(i)+h/2, u(i)+k1u/2, v(i)+k1v/2);
        k2v = h*g(t(i)+h/2, u(i)+k1u/2, v(i)+k1v/2);
        k3u = h*f(t(i)+h/2, u(i)+k2u/2, v(i)+k2v/2);
        k3v = h*g(t(i)+h/2, u(i)+k2u/2, v(i)+k2v/2);
        k4u = h*f(t(i)+h, u(i)+k3u, v(i)+k3v);
        k4v = h*g(t(i)+h, u(i)+k3u, v(i)+k3v);
        u(i+1) = u(i) + (k1u + 2*k2u + 2*k3u + k4u)/6;
        v(i+1) = v(i) + (k1v + 2*k2v + 2*k3v + k4v)/6;
    end
end
```

8 Função `ode45` do MATLAB

Descrição

A função `ode45` do MATLAB é um dos métodos numéricos mais utilizados para resolver Sistemas de Equações Diferenciais (SED). Ela é baseada no método de Runge-Kutta de ordem 4(5), combinando métodos de Runge-Kutta de ordens 4 e 5 para controle adaptativo do passo de integração, garantindo alta precisão e eficiência. A função é adequada para a maioria dos problemas não-stiff, como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento) ou circuitos elétricos, fornecendo soluções com alta precisão.

O sistema a ser resolvido tem a forma:

$$\begin{cases} u'(t) = f(t, u(t), v(t)) \\ v'(t) = g(t, u(t), v(t)) \\ u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases} \tag{8.1}$$

em que $u(t)$ e $v(t)$ são as componentes do vetor solução $\mathtt{x}(t) = [u(t), v(t)]$, e f e g são funções que definem a dinâmica do sistema.

Sintaxe

```
[t, y] = ode45(@(t, y) [f(t, y(1), y(2)); g(t, y(1), y(2))], tspan, y0)
```

onde:

- `@(...) [f(t, y(1), y(2)); g(t, y(1), y(2))]` → Handle da função que define o SED, com $y(1) = u(t)$ e $y(2) = v(t)$;
- `tspan` → Intervalo de integração, geralmente definido como `[a b]` ou um vetor de pontos `a:h:b`;
- `y0` → Vetor de condições iniciais $[u_0, v_0]$;
- `t` → Vetor de tempos nos quais a solução foi avaliada;
- `y` → Matriz cujas colunas contêm as soluções numéricas aproximadas para $u(t)$ e $v(t)$ nos tempos definidos em `t`.

Exemplo de Uso

Exemplo: Resolver o sistema de equações diferenciais $u' = -2u + v$**,** $v' = u - 2v$**, com condições iniciais** $u(0) = 1$**,** $v(0) = 0$**, no intervalo** $t \in [0, 5]$**.**

```
f = @(t, u, v) -2*u + v;
g = @(t, u, v) u - 2*v;
[t, u, v] = ODE45SED([], f, g, 0, 5, 100, 1, 0);
plot(t, u, 'b-', t, v, 'r--')
xlabel('t')
ylabel('Soluções u(t) e v(t)')
title('Solução do SED usando ode45')
legend('u(t)', 'v(t)')
```

A implementação da função auxiliar em Matlab para resolver o SED é apresentada abaixo:

```
function [t, u, v] = ODE45SED(~, f, g, a, b, n, u0, v0)
    h = (b-a)/n;
    tspan = a:(b-a)/n:b;
    y0 = [u0; v0];
    [T, Y] = ode45(@(t,y) [f(t,y(1),y(2)); g(t,y(1),y(2))], tspan, y0);
    t = T';
    u = Y(:,1)';
    v = Y(:,2)';
end
```

Vantagens

- Método adaptativo: ajusta automaticamente o passo de integração para otimizar precisão e eficiência;
- Simples de usar, exigindo apenas a definição da função do sistema e condições iniciais;
- Ideal para SED com soluções suaves e bem comportadas;
- Alta precisão devido à combinação de métodos de ordens 4 e 5.

9 Exercícios

Mola-Massa s/ Amortecimento

Mola-Massa c/ Amortecimento

Circuito Elétrico

Pêndulo

Oscilador Simples (Ext.)

Sistema amortecido simples (Ext.)

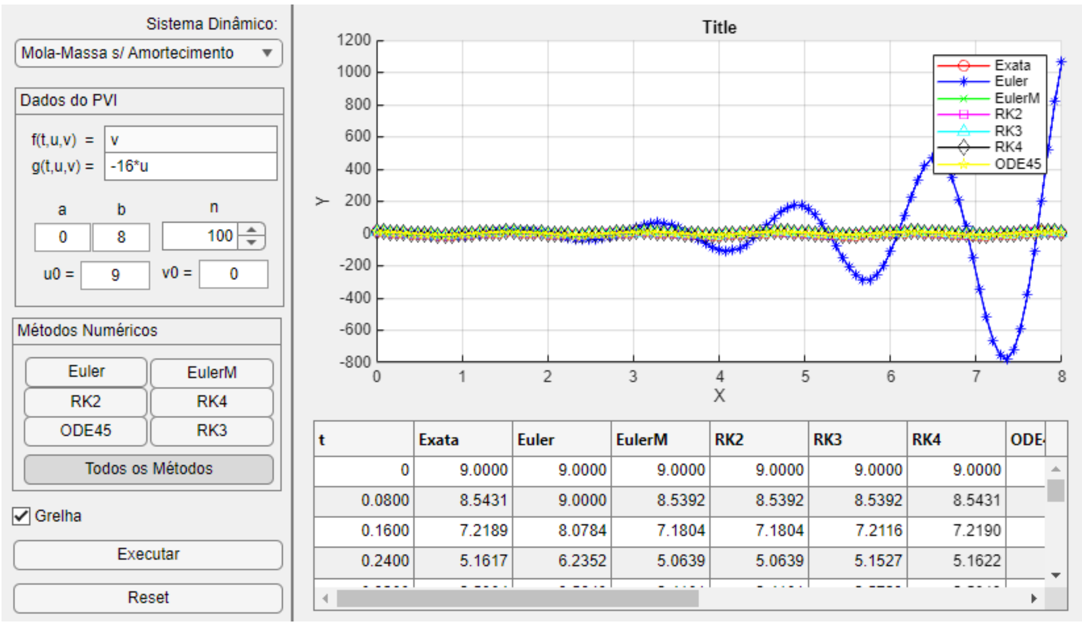

Figura 9.1: Mola-Massa s/ Amortecimento

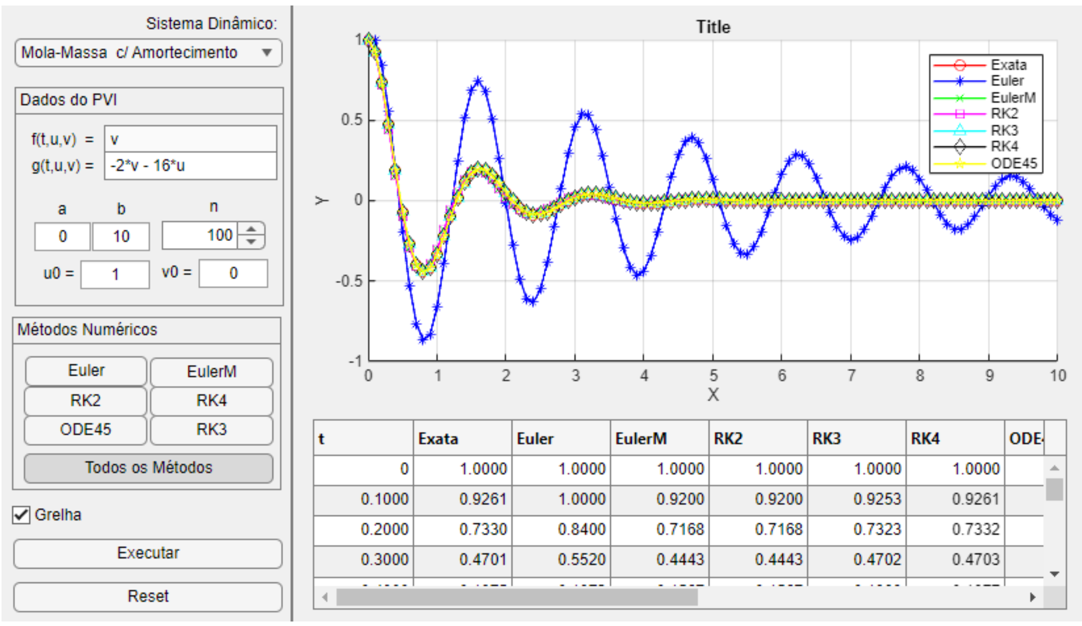

Figura 9.2: Mola-Massa c/ Amortecimento

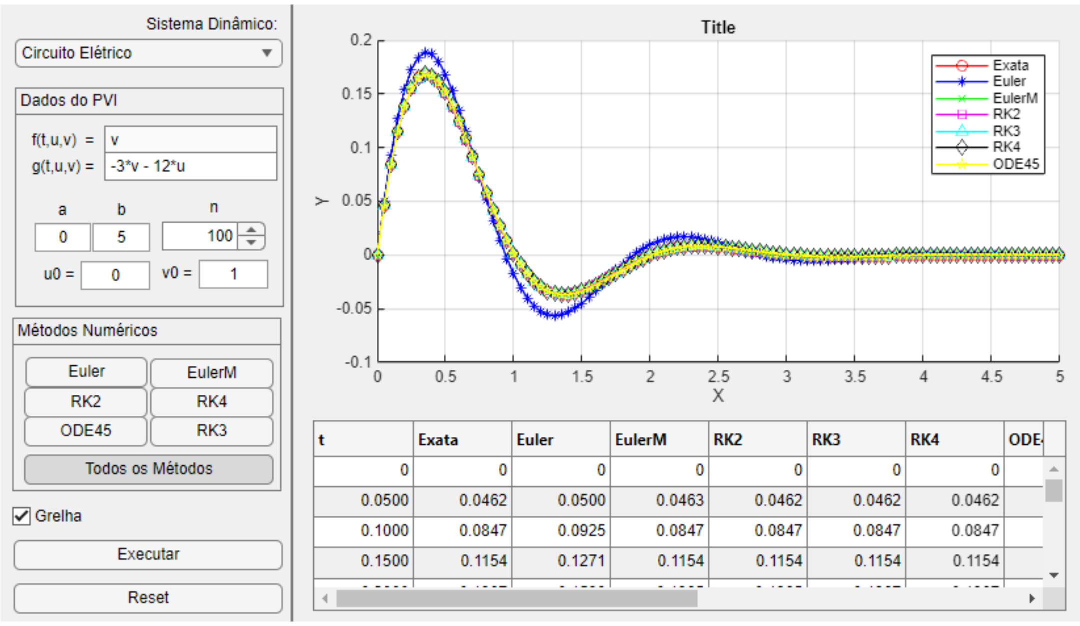

Figura 9.3: Circuito Elétrico

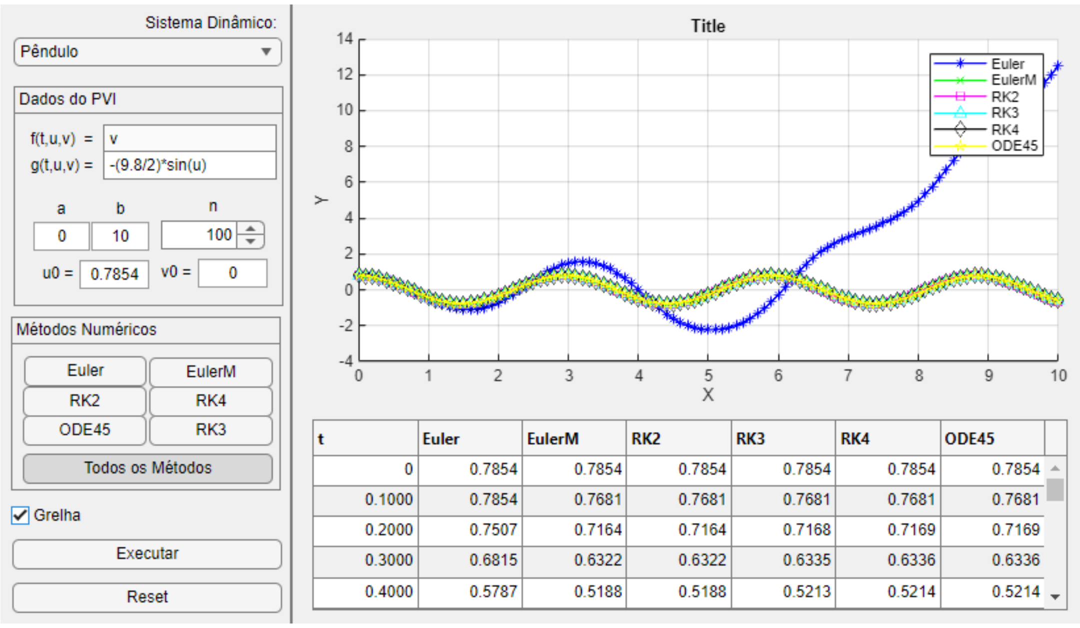

Figura 9.4: Pêndulo

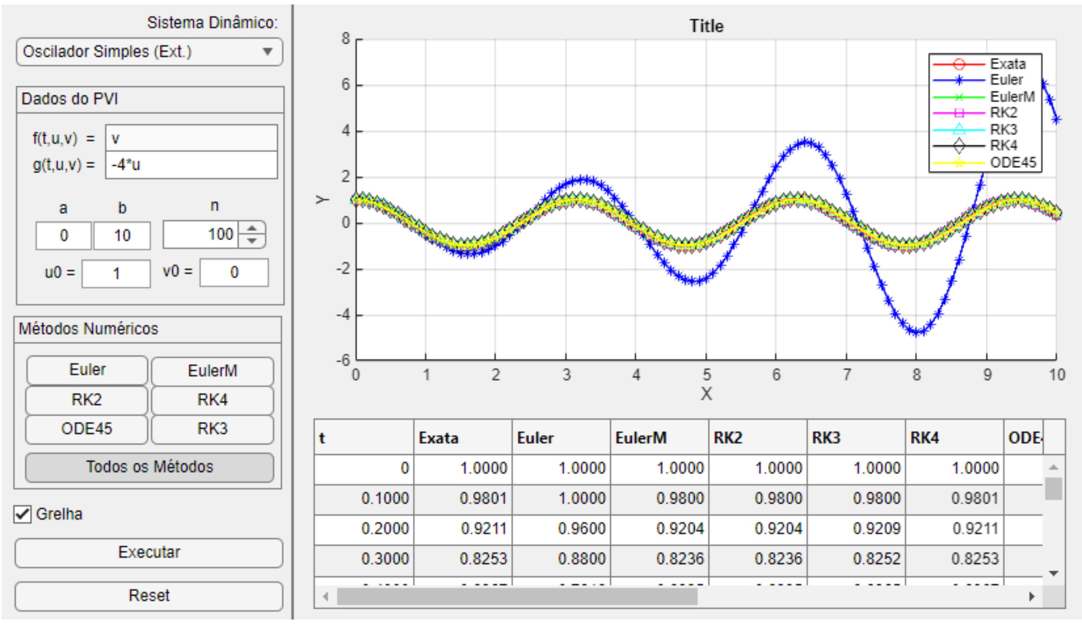

Figura 9.5: Oscilador Simples (Ext.)

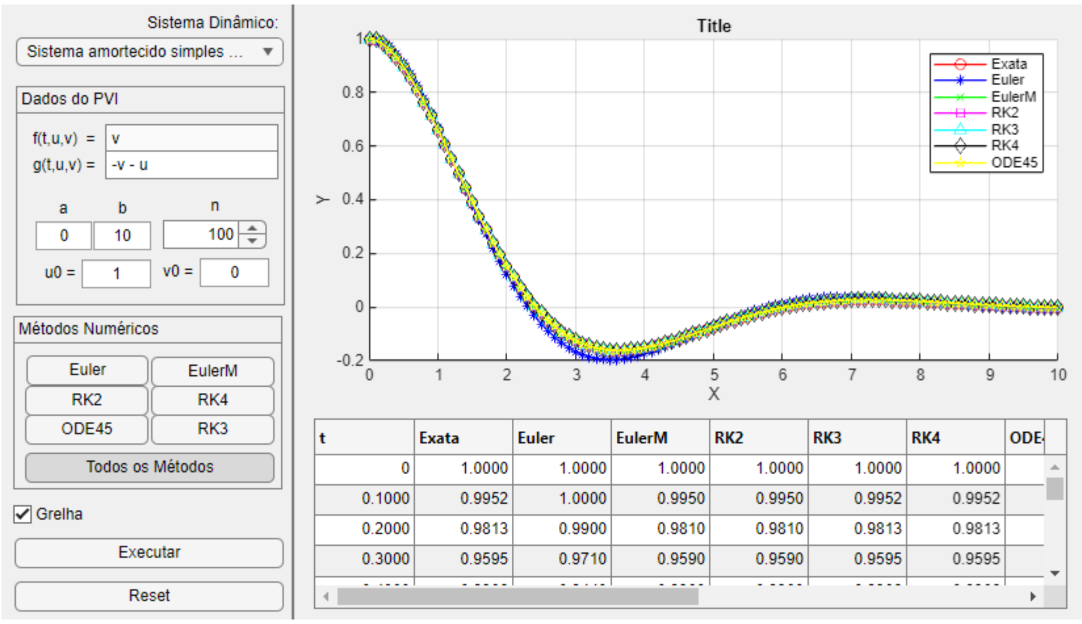

Figura 9.6: Sistema amortecido simples (Ext.)

10 Conclusão

A realização desta atividade permitiu aprofundar os conhecimentos sobre métodos numéricos para a resolução de Sistemas de Equações Diferenciais (SED), com especial foco nos métodos de Euler, Euler Melhorado, Runge-Kutta de ordens 2, 3 e 4, e na função `ode45` do Matlab.

Foi possível compreender a fundamentação teórica destes métodos, bem como as suas implementações práticas em Matlab, analisando as diferenças em termos de precisão, estabilidade e custo computacional. A aplicação destes métodos a problemas práticos, como sistemas mecânicos (e.g., mola-massa com ou sem amortecimento) e circuitos elétricos, permitiu avaliar a sua eficácia em contextos reais de modelação matemática.

A comparação entre os resultados obtidos por diferentes métodos evidenciou a importância de balancear a complexidade computacional com a precisão desejada, considerando o controle do erro associado à discretização. Os métodos de Runge-Kutta, especialmente o de quarta ordem e a função `ode45`, destacaram-se pela sua elevada precisão em problemas com soluções suaves, sendo ideais para a maioria das aplicações práticas. Por outro lado, métodos mais simples, como o de Euler, mantêm relevância em cenários onde a simplicidade e o baixo custo computacional são prioritários.

Em suma, esta atividade contribuiu significativamente para a consolidação de competências na área da análise numérica, nomeadamente na aplicação prática de técnicas fundamentais para a simulação de sistemas dinâmicos descritos por SED, reforçando a importância de ferramentas computacionais como o Matlab na resolução de problemas complexos.

Referências bibliográficas

[1] Wikipédia, a enciclopédia livre. (2025) Método de euler para sistemas de equações diferenciais. Acedido em 4 de abril de 2025. [Online]. Available: https://pt.wikipedia.org/wiki/M%C3%A9todo_de_Euler

[2] ——. (2025) Método de heun para sistemas de equações diferenciais. Acedido em 4 de abril de 2025. [Online]. Available: https://en.wikipedia.org/wiki/Heun%27s_method

[3] ——. (2025) Métodos de runge–kutta para sistemas de equações diferenciais. Acedido em 4 de abril de 2025. [Online]. Available: https://pt.wikipedia.org/wiki/M%C3%A9todo_de_Runge-Kutta

[4] ——. (2025) Função `ode45` para resolução de sistemas de equações diferenciais. Acedido em 4 de abril de 2025. [Online]. Available: https://en.wikipedia.org/wiki/ODE45

[5] MathWorks. (2025) `ode45` – resolver sistemas de equações diferenciais não-rígidos – método de ordem média. Acedido em 4 de abril de 2025. [Online]. Available: https://www.mathworks.com/help/matlab/ref/ode45.html

[6] C. Álves. (2025) Capítulo 5 – sistemas de equações diferenciais. Acedido em 4 de abril de 2025. [Online]. Available: https://www.math.tecnico.ulisboa.pt/~calves/MC/cap5.html

[7] F. Coutinho, P. Silva, and R. Rodrigues, "Como escrever um bom relatório em latex," ISEC/IPC, Coimbra, Tech. Rep., Nov. 2022.

[8] A. Silva and J. Pereira, "Avanços em métodos numéricos para sistemas de equações diferenciais: Aplicações em modelação física," *Revista de Matemática Aplicada*, vol. 12, no. 3, pp. 45–62, 2025, acedido em 4 de abril de 2025. [Online]. Available: https://www.revmataplicada.pt/artigos/2025/silva_sed

[9] R. Mendes, *Sistemas de Equações Diferenciais: Teoria e Aplicações em Modelação Dinâmica*. Lisboa: Editora Universitária de Lisboa, 2025, acedido em 4 de abril de 2025.

Instituto Superior
de Engenharia
Politécnico de Coimbra