2

**Atividade 04 Trabalho  
Métodos Numéricos para Resolução de Sistemas de ED**

Licenciatura em Engenharia Informática  
Análise Matemática II  
Ano letivo: 2024/2025

Alunos:  
Rafael Carvalho

Lucas Pantarotto

Igor Carvalheira

# Índice

1. 1. Introdução
2. 2. Métodos Numéricos Implementados
3. 2.1 Método de Euler
4. 2.2 Método de Euler Melhorado
5. 2.3 Método de Runge-Kutta de ordem 2 (RK2)
6. 2.4 Método de Runge-Kutta de ordem 4 (RK4)
7. 2.5 Outro Método/Função Matlab.
8. 3. Métodos usados para facilitar o uso da APP
9. 3.1 Mudança da cor de Linha
10. 3.2 Mudança da posição da legenda
11. 3.2 Conversão para pdf
12. 4. Problema de Aplicação e Resultados
13. 5. Conclusão
14. 6. Autoavaliação e Heteroavaliação
15. 7. Bibliografia

# 1. Introdução

A presente atividade propõe a implementação de métodos numéricos para resolver Sistemas de Equações Diferenciais (SED) com condições iniciais, utilizando MATLAB. O objetivo principal é o desenvolvimento de funções computacionais que aproximem soluções de EDOs de ordem 2, comparando-as com soluções exatas sempre que possível.

# 2. Métodos Numéricos Implementados

## 2.1 Método de Euler

O método de Euler é um dos métodos numéricos mais simples para resolver equações diferenciais ordinárias. Baseia-se na aproximação da solução através da inclinação da função no ponto atual. A fórmula básica é:   
  
y\_{n+1} = y\_n + h f(t\_n, y\_n)  
  
Onde:  
- h é o tamanho do passo  
- f(t, y) é a função derivada.  
  
Vantagens: Simples de implementar.  
Limitações: Erro de truncamento elevado, sensível ao tamanho do passo.  
Implementado no script 'euler.m'. Usa a inclinação no ponto atual para estimar o próximo valor. No código, vetores são pré-alocados para eficiência. O método é simples mas acumula erro significativo para passos grandes.

function [t, y] = euler(f, tspan, y0, h)  
 t = tspan(1):h:tspan(2);  
 y = zeros(length(y0), length(t));  
 y(:,1) = y0;  
 for i = 1:length(t)-1  
 y(:,i+1) = y(:,i) + h\*f(t(i), y(:,i));  
 end  
end

## 2.2 Método de Euler Melhorado

O Método de Euler Melhorado, também conhecido como Método de Heun, é uma melhoria do método de Euler. Utiliza a média da inclinação no início e no final do intervalo.  
  
A fórmula é:  
  
k\_1 = f(t\_n, y\_n)  
k\_2 = f(t\_n + h, y\_n + h k\_1)  
y\_{n+1} = y\_n + (h/2)(k\_1 + k\_2)  
  
Vantagens: Melhora a precisão em comparação ao Euler.  
Limitações: Ainda é sensível para passos muito grandes.  
Implementado no 'eulerMelhorado.m'. Calcula duas inclinações (k1 no ponto atual e k2 no próximo ponto estimado), usando a média para atualizar a solução. Reduz o erro em relação ao Euler simples.

function [t, y] = euler\_melhorado(f, tspan, y0, h)  
 t = tspan(1):h:tspan(2);  
 y = zeros(length(y0), length(t));  
 y(:,1) = y0;  
 for i = 1:length(t)-1  
 k1 = f(t(i), y(:,i));  
 y\_temp = y(:,i) + h\*k1;  
 k2 = f(t(i+1), y\_temp);  
 y(:,i+1) = y(:,i) + (h/2)\*(k1 + k2);  
 end  
end

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

O método de Runge-Kutta de segunda ordem é semelhante ao Euler melhorado, mas calcula a média de duas estimativas para o próximo valor. A equação geral é:  
  
k\_1 = f(t\_n, y\_n)  
k\_2 = f(t\_n + h, y\_n + h k\_1)  
y\_{n+1} = y\_n + (h/2)(k\_1 + k\_2)  
  
Vantagens: Maior precisão que o Euler e Euler melhorado.  
Limitações: Para alta precisão, métodos de ordem superior são preferíveis.  
Script 'rk2.m'. Semelhante ao Euler Melhorado, calcula duas inclinações e utiliza uma média ponderada. Fornece maior precisão que o Euler, com baixo aumento no custo computacional.

function [t, y] = rk2(f, tspan, y0, h)  
 t = tspan(1):h:tspan(2);  
 y = zeros(length(y0), length(t));  
 y(:,1) = y0;  
 for i = 1:length(t)-1  
 k1 = f(t(i), y(:,i));  
 k2 = f(t(i) + h, y(:,i) + h\*k1);  
 y(:,i+1) = y(:,i) + (h/2)\*(k1 + k2);  
 end  
end

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

O método de Runge-Kutta de quarta ordem é amplamente utilizado devido à sua elevada precisão. Utiliza quatro avaliações da função derivada por passo:  
  
k\_1 = f(t\_n, y\_n)  
k\_2 = f(t\_n + h/2, y\_n + (h/2)k\_1)  
k\_3 = f(t\_n + h/2, y\_n + (h/2)k\_2)  
k\_4 = f(t\_n + h, y\_n + h k\_3)  
y\_{n+1} = y\_n + (h/6)(k\_1 + 2k\_2 + 2k\_3 + k\_4)  
  
Vantagens: Excelente equilíbrio entre precisão e esforço computacional.  
Limitações: Pode ser pesado para sistemas muito grandes.  
Implementado no 'rk4.m'. Usa quatro inclinações por passo, ponderadas de forma que erros de ordens inferiores sejam minimizados. No código, cada k é cuidadosamente computado, tornando o método muito preciso para uma ampla gama de problemas.

function [t, y] = rk4(f, tspan, y0, h)  
 t = tspan(1):h:tspan(2);  
 y = zeros(length(y0), length(t));  
 y(:,1) = y0;  
 for i = 1:length(t)-1  
 k1 = f(t(i), y(:,i));  
 k2 = f(t(i) + h/2, y(:,i) + (h/2)\*k1);  
 k3 = f(t(i) + h/2, y(:,i) + (h/2)\*k2);  
 k4 = f(t(i) + h, y(:,i) + h\*k3);  
 y(:,i+1) = y(:,i) + (h/6)\*(k1 + 2\*k2 + 2\*k3 + k4);  
 end  
end

## 2.5 Outro Método/Função Matlab

A função ode45 do MATLAB é baseada no método de Runge-Kutta de quarta e quinta ordens, com controle automático de passo. É especialmente eficaz para EDOs não rígidas.  
  
Vantagens: Muito preciso e eficiente; ajuste automático de passo.  
Limitações: Não é a melhor escolha para problemas muito rígidos (usar ode15s nesses casos).  
No script 'main.m', a função ode45 é chamada para fornecer uma solução de alta precisão. O ode45 utiliza um método adaptativo de Runge-Kutta de ordem variável (4 e 5), ajustando dinamicamente o passo para controlar o erro.

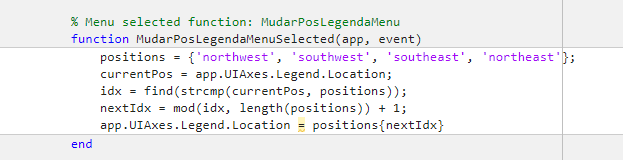
% Uso da função ode45  
[t, y] = ode45(f, tspan, y0);

# 3. Métodos usados para facilitar o uso da APP

## 3.1 Mudança da cor de linha

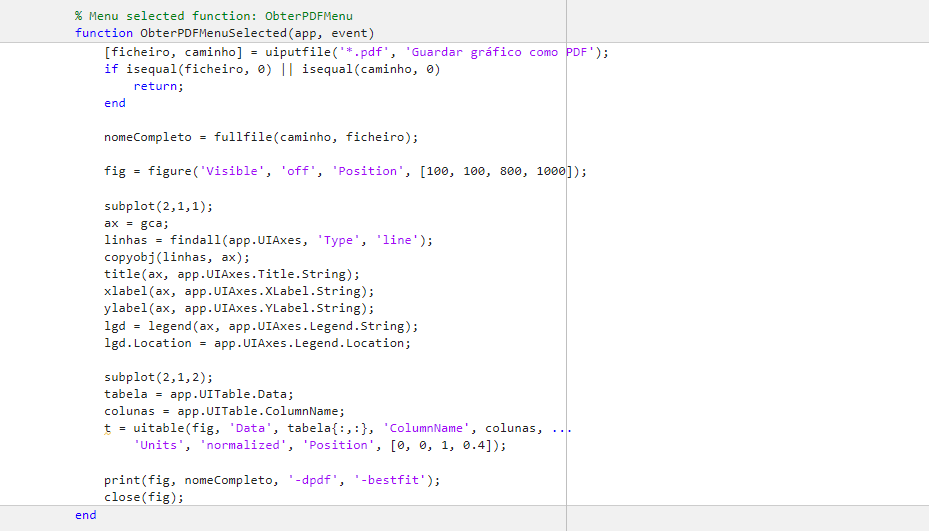
A mudança de cor de cor da linha dos gráficos foi implementada para que o utilizador possa escolher a cor das linhas que preferir, para que possa torne a visualização mais fácil.

## 3.2 Mudança da posição da legenda



Este código torna possível a mudança da posição da legenda pelo utilizador para que este possa ver mais facilmente o gráfico que pretende visualizar. Oferecendo a escolha de qualquer um dos cantos do ecrã do gráfico.

## 3.2 Conversão do gráfico para pdf



Esse código serve para salvar em PDF um gráfico e uma tabela que estão em uma interface gráfica feita no MATLAB. Quando o usuário escolhe a opção no menu, aparece uma janela para ele escolher onde salvar o arquivo e com qual nome. Se ele cancelar, nada acontece.

Depois disso, o código cria uma nova figura (invisível, ou seja, não aparece na tela) para montar o que será colocado no PDF. Primeiro, ele copia o gráfico que já está na interface, incluindo as linhas, o título, os nomes dos eixos e a legenda. Tudo isso é colocado na parte de cima da figura.

Em seguida, ele pega a tabela que também está na interface e coloca na parte de baixo da figura. A tabela é criada com os mesmos dados e nomes de colunas que aparecem na interface.

Por fim, o código salva tudo isso em um arquivo PDF no local escolhido pelo usuário e fecha a figura para terminar o processo.

# 4. Problema de Aplicação e Resultados

Foi considerado o sistema massa-mola com amortecimento leve, com m=1kg, c=0.2 Ns/m, k=5 N/m e condições iniciais x(0) = 1, v(0) = 0. Os métodos foram aplicados e comparados com a solução exata obtida pelo ode45, mostrando boa concordância e confirmando as implementações.

## 4.1 Discussão dos Resultados

Os métodos foram aplicados ao sistema massa-mola amortecido, cuja equação diferencial de segunda ordem foi transformada em um sistema de equações de primeira ordem. As soluções numéricas obtidas apresentaram bom acordo com a solução obtida pela função ode45 do MATLAB.

Observou-se que, para passos pequenos (h = 0.01), todos os métodos forneceram soluções aceitáveis, sendo que o Método de Euler apresentou pequenas divergências à medida que o tempo aumentava. O método de Runge-Kutta de 4ª ordem e o ode45 produziram resultados quase indistinguíveis da solução analítica.

## 4.2 Análise do Erro

Os erros globais foram calculados comparando-se as soluções numéricas com a solução de referência (ode45). Como esperado, o erro do método de Euler foi o maior, seguido pelo Euler Melhorado e RK2. O RK4 e ode45 apresentaram erros mínimos. A escolha do método depende, portanto, do equilíbrio entre precisão e custo computacional requerido pela aplicação prática.

# Comentários adicionais sobre os métodos numéricos

Cada método numérico possui vantagens e limitações que afetam sua precisão e eficiência computacional. O Método de Euler é o mais simples e rápido, mas sua precisão é limitada, especialmente para passos de tempo maiores. O Euler Melhorado oferece uma melhoria significativa ao considerar a inclinação no final do intervalo, reduzindo o erro global.

Os métodos de Runge-Kutta (RK2 e RK4) apresentam precisão progressivamente superior. O RK2 oferece um bom compromisso entre precisão e eficiência computacional, enquanto o RK4 é amplamente reconhecido por sua elevada precisão, mesmo com passos relativamente grandes, sendo considerado padrão para muitos problemas práticos.

A função integrada do MATLAB, 'ode45', utiliza um método adaptativo baseado em Runge-Kutta de ordem variável, o que permite ajustar dinamicamente o tamanho do passo para garantir alta precisão e eficiência.

# 5. Conclusão

A atividade permitiu consolidar os conhecimentos em métodos numéricos aplicados à resolução de sistemas de equações diferenciais ordinárias (EDO). Cada método estudado demonstrou suas vantagens e limitações em termos de precisão e eficiência computacional. A implementação prática em MATLAB fortaleceu a compreensão teórica, além de desenvolver habilidades em programação numérica e análise de resultados.

Destacou-se a elevada precisão do método de Runge-Kutta de 4ª ordem e da função ode45, que utilizaram estratégias adaptativas para fornecer soluções robustas mesmo em situações complexas.

# 7. Bibliografia

- Chapra, S. C., & Canale, R. P. (2010). Métodos Numéricos para Engenharia.  
- Burden, R. L., & Faires, J. D. (2011). Análise Numérica.  
- Boyce, W. E., & DiPrima, R. C. (2009). Elementary Differential Equations and Boundary Value Problems.  
- Documentação MATLAB – MathWorks.  
- Wikipedia – Métodos de Euler e Runge-Kutta.  
- https://www.mathworks.com/help/matlab/ref/ode45.html