1.7 Solução de equações diferenciais

Prof. Dr. Sidney Bruce Shiki
e-mail: bruce@ufscar.br
Prof. Dr. Vitor Ramos Franco
e-mail: vrfranco@ufscar.br

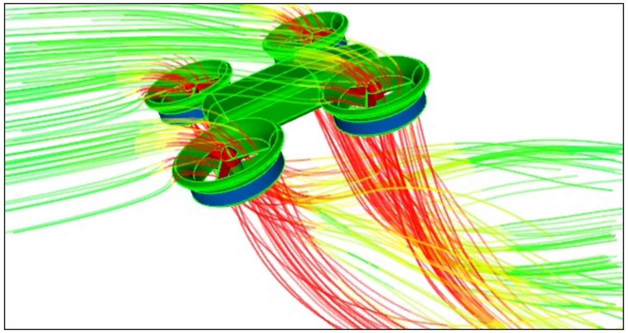
UFSCar – Universidade Federal de São Carlos
DEMec - Departamento de Engenharia Mecânica

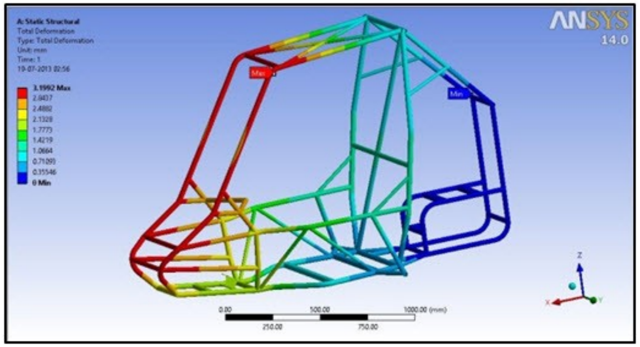
Conteúdo

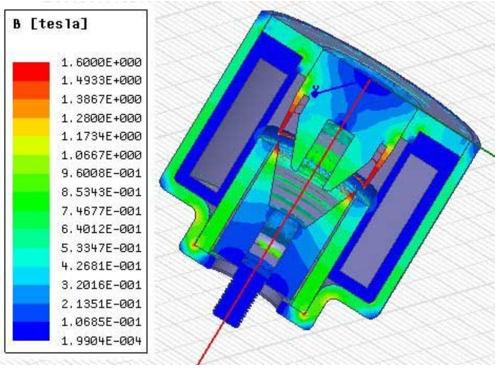
- Introdução
- Solução via método de Euler
- Representação de EDOs no MATLAB
- Solução de EDOs
- Exercícios

Introdução

- Boa parte das equações e modelos matemáticos que lidamos em engenharia são equações diferenciais;
- Softwares de modelagem mais complexas também resolvem equações diferenciais.







Introdução

- Nem sempre é possível obter soluções analíticas fechadas para as EDOs que tratamos;
- Soluções numéricas são empregadas para se obter aproximações dessas soluções;
- O MATLAB possui diversas rotinas para resolver EDOs;
- Implementaremos o método de Euler para resolver EDOs e posteriormente usaremos a famosa função `ode45` do MATLAB para esse tipo de solução.

Solução via método de Euler

- Considere a seguinte equação diferencial:

$$\frac{dy}{dt} = \sin t + 1$$

- A EDO acima tem solução analítica? Se sim, como obtê-la?

Solução via método de Euler

- Considere a seguinte equação diferencial:

$$\frac{dy}{dt} = \sin t + 1$$

- A EDO acima tem solução analítica? Se sim, como obtê-la?
  - Separação de variáveis:

$$\boxed{y(t) = t - \cos t + y_0 + 1}$$

$$y_0 = y(\mathrm{t} = 0)$$

Solução via método de Euler

- Mas e se a ED for muito complicada ?

$$\frac{d^2y}{dt^2}+\frac{dy}{dt}+y+y^3=e^{t^2}$$

- Solução numérica aproxima as derivadas com funções mais fáceis de serem trabalhadas;

Solução via método de Euler

- Método de Euler (mais básico) aproxima derivadas com diferenças finitas:

$$\frac{dy}{dt} \approx \frac{y_n - y_{n-1}}{t_n - t_{n-1}}$$

- Onde a diferença de tempos é o passo (*h*):

$$\dot{y} \approx \frac{y_n - y_{n-1}}{h}$$

$$\boxed{y_n \approx y_{n-1} + h\dot{y}}$$

Solução via método de Euler

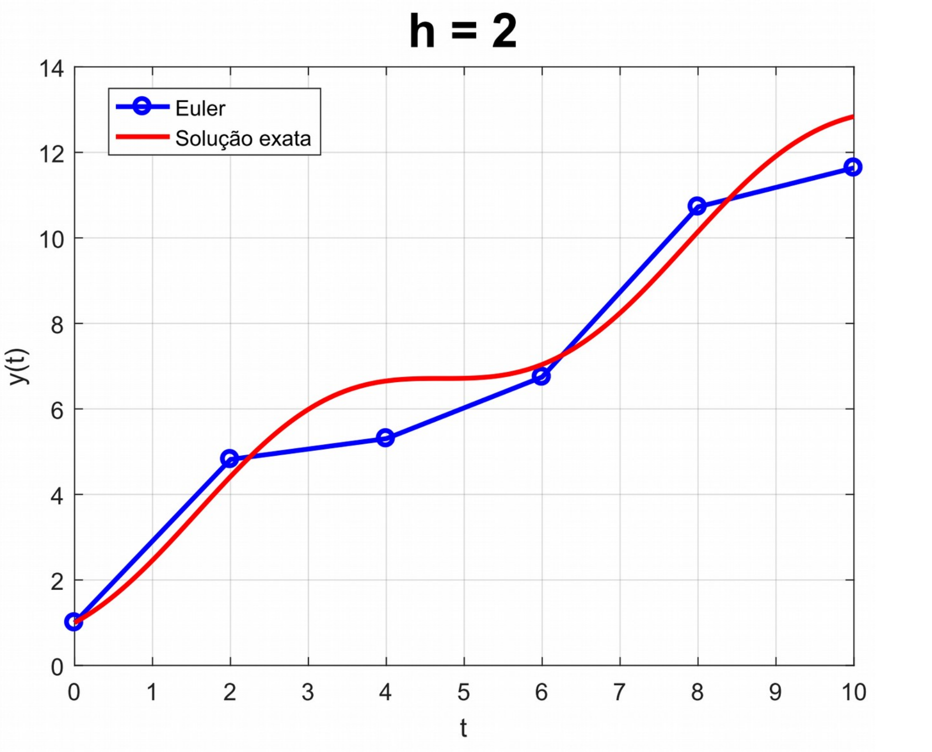
- Solução via Euler

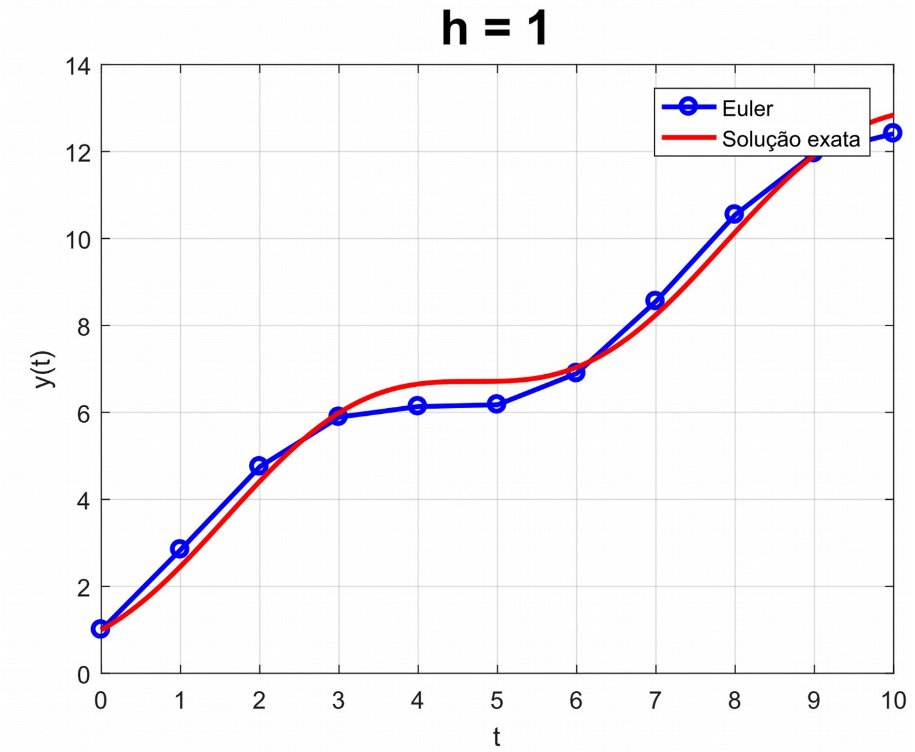
```
y0 = 1; % Valor inicial
dydt = @(t) sin(t)+1;    % Equação diferencial
% Vetor de tempos
h = 1;   % passo
t = 0:h:10;
ta = 0:0.001:10;
% Solução analítica
ya = ta-cos(ta)+y0+1;
% Solução numérica
y = zeros(size(t));
y(1) = y0;
for n = 2:length(y)
    y(n) = y(n-1) + h*dydt(t(n));
end

figure;
plot(t,y,'b-o');hold on;plot(ta,ya,'r');
xlabel('t');ylabel('y(t)');
legend('Euler','Solução exata');grid on
```

Solução via método de Euler

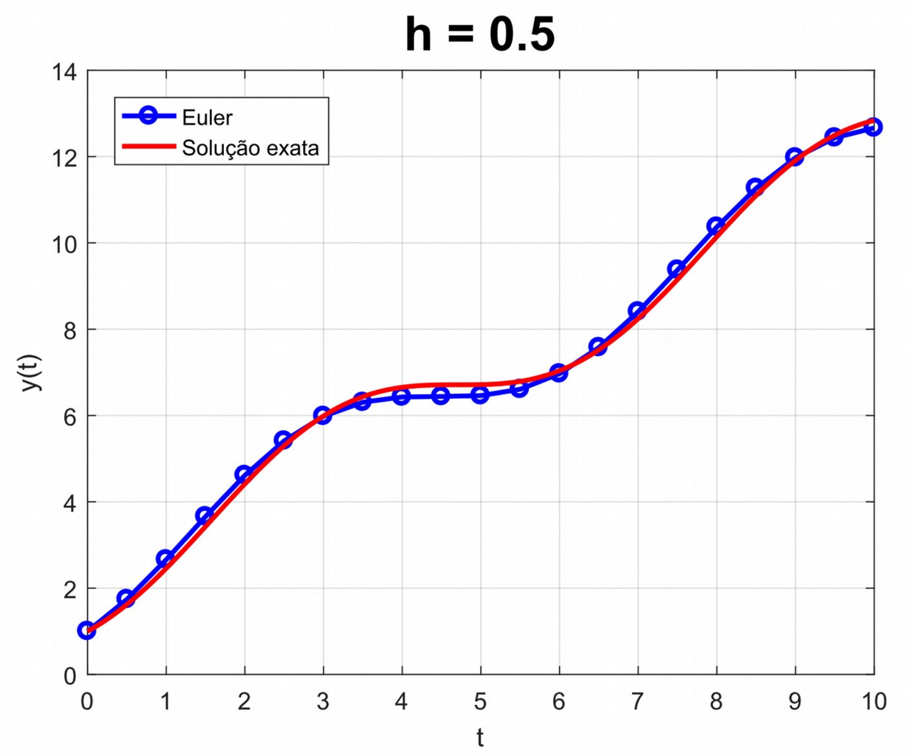
- Tamanho do passo tem efeito importante no método de Euler:

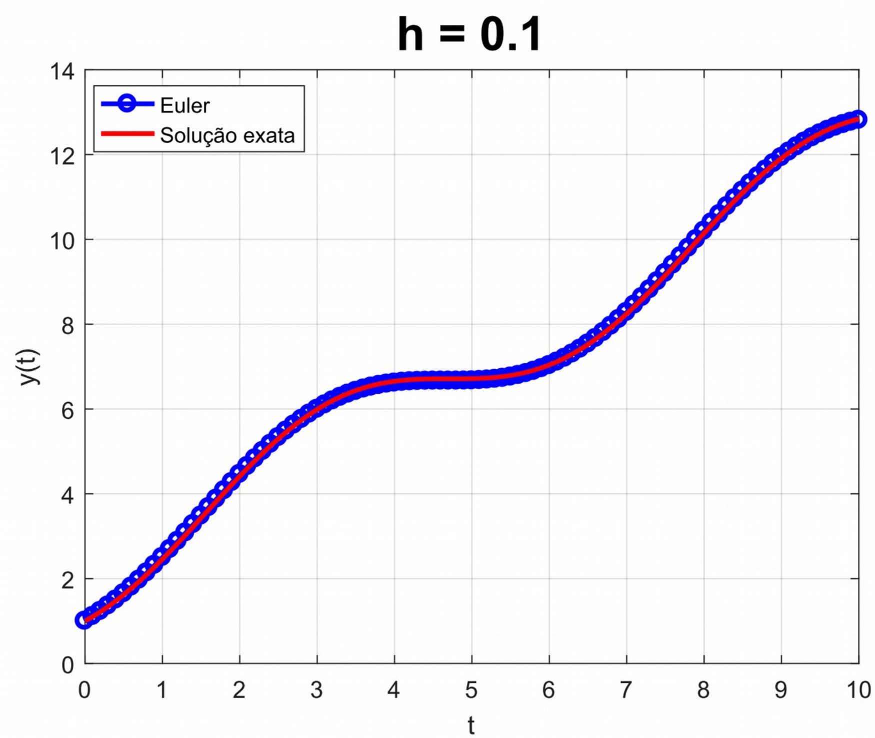




Solução via método de Euler

- Tamanho do passo tem efeito importante no método de Euler:





Solução via método de Euler

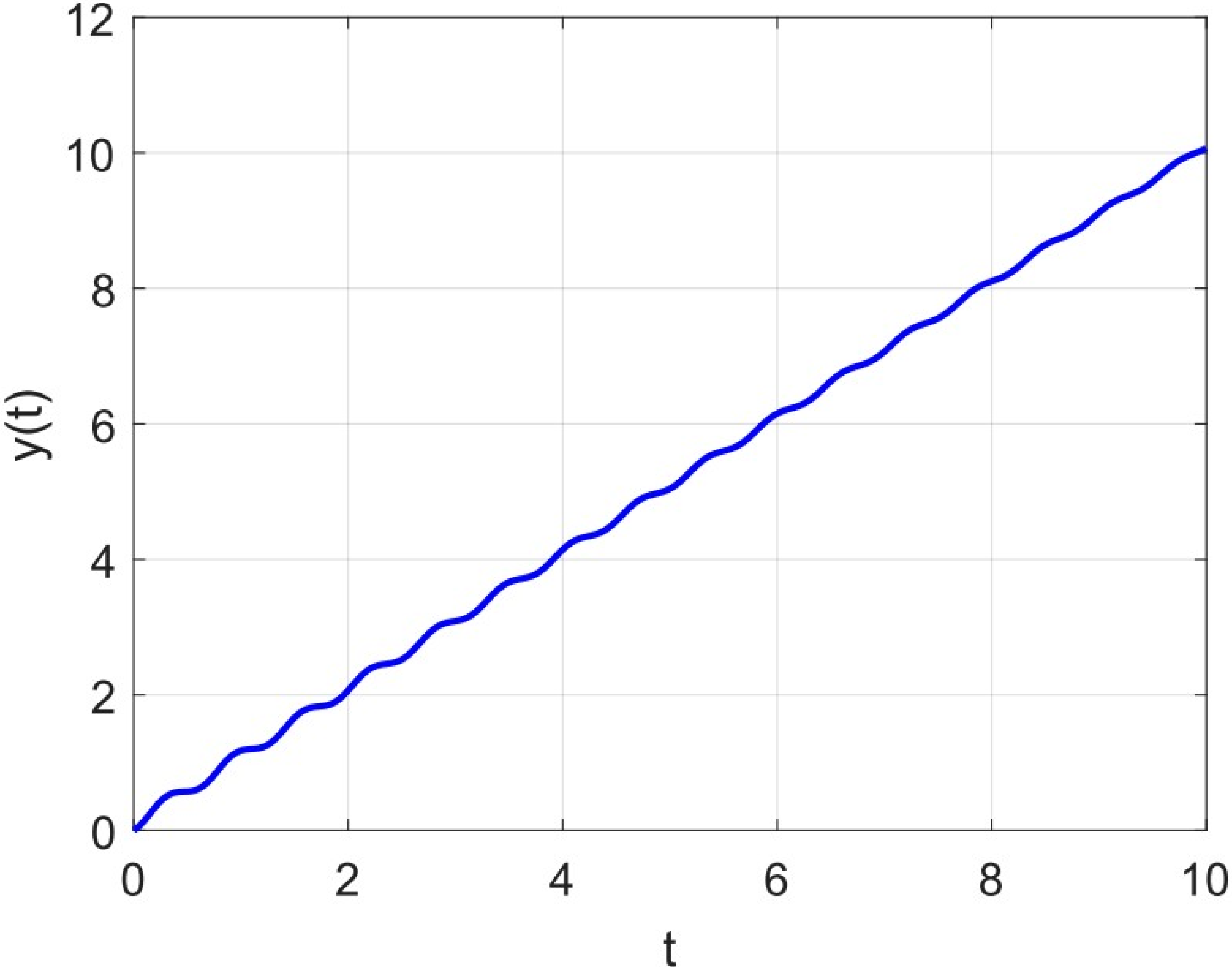
- Exercício 01 – Resolva numericamente a equação abaixo com o método de Euler para uma faixa de tempo de 0 até 5 segundos (escolha o passo de tempo que achar adequado):

$$\frac{dy}{dt} = e^{-0.5\,t} \sin\left(10t\right) + 1$$

$$y(t = 0) = y_0 = 0$$

Solução via método de Euler

- Exercício 01 –Resposta:



Solução via método de Euler

- Exercício 02 – Resolva numericamente a EDO abaixo que representa um circuito RC simples com voltagem de entrada e_i e de saída e_0

$$RC\frac{de_0}{dt}+e_0=e_i$$

$$R=100000\,\Omega$$

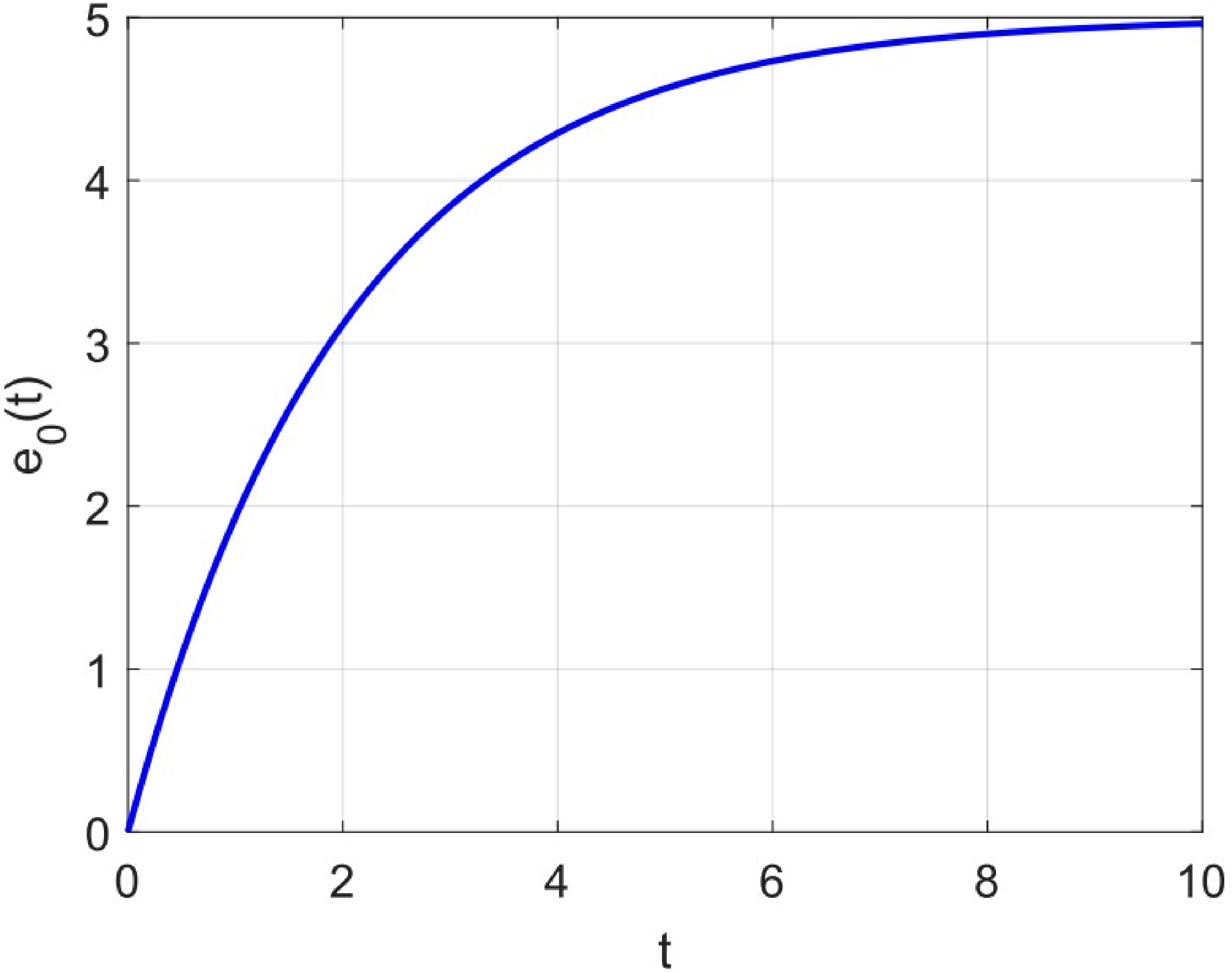
$$C=2\times10^{-5}\ \mathrm{F}$$

$$e_i=5\,\mathrm{V}$$

$$e_0(t=0)=0$$

Solução via método de Euler

- Exercício 02 – Resposta



Representação de EDOs no MATLAB

- Embora o método de Euler possa ser usado, ele não é o mais indicado (erro depende fortemente do passo utilizado);

- Outros métodos são análogos mas possuem melhores aproximações (veja métodos de Runge-Kuta, Preditor-corretor, etc).

Representação de EDOs no MATLAB

- Os métodos numéricos geralmente são usados para se resolver EDOs de primeira ordem (apenas com derivada de primeira ordem);

- No entanto pode-se "transformar" uma EDO de ordem N, em N equações diferenciais de primeira ordem.

Representação de EDOs no MATLAB

- Exemplo: $\ddot{y}+\dot{y}+y=0$
  - Cria-se uma variável adicional y_2:

$$y_2=\dot{y}$$

  - Desse modo:

$$\dot{y}_2=-y_2-y$$

  - Temos um sistema de equações diferenciais de ordem 1

$$\begin{aligned}\dot{y}&=y_2\\ \dot{y}_2&=-y_2-y\end{aligned} \longrightarrow \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\dot{y}\\ \dot{y}_2\end{bmatrix}=\begin{bmatrix}y_2\\ -y_2-y\end{bmatrix}$$

Representação de EDOs no MATLAB

- É conveniente escrever a expressão em forma matricial

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{A}\underbrace{\begin{bmatrix} \dot{y} \\ \dot{y}_2 \end{bmatrix}}_{\dot{\mathbf{y}}}=\underbrace{\begin{bmatrix} y_2 \\ -y_2-y \end{bmatrix}}_{b}$$

- Ou ainda:

$$A\dot{\mathbf{y}}=b \longrightarrow \dot{\mathbf{y}}=A^{-1}b$$

Representação de EDOs no MATLAB

- Note que:

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y} \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} \xrightarrow{\text{logo}} \mathbf{y} = \begin{bmatrix} y \\ y_2 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

- Podemos expressar as condições iniciais da EDO também em um vetor:

$$\mathbf{y}_0 = \begin{bmatrix} y(0) \\ \dot{y}(0) \end{bmatrix}$$

Representação de EDOs no MATLAB

- A EDO deve ser representada por uma `function`:

```
function[dotY] = edoex01(t,Y)
% ED: d2y/dt2 + dy/dt + y = 0
% Entradas
% Y(1) - y
% Y(2) - dy/dt
% Saídas
% dotY(1) - dy/dt
% dotY(2) - d2y/dt2

A = [1 0;0 1];
b = [Y(2);
    -Y(2)-Y(1)];
dotY = inv(A)*b;

end
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -y_2 - y \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y \\ y_2 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \qquad \texttt{Y}$$

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y} \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} \qquad \texttt{dotY}$$

Solução de EDOs

- A função básica para resolução de EDOs tem a seguinte sintaxe:

```
[t,Y] = ode45(@odefun,t,x0)
```

`t` – vetor de tempo ou da variável independente
`Y` – matriz com respostas da EDO
`@odefun` – equação diferencial expressa em uma function
`t` – vetor de tempo ou então faixa de tempo (e.g. [0 10])
`x0` – vetor com condições iniciais do problema

- Essa função é uma implementação do algoritmo Runge-Kutta de 4ª-5ª ordem, embora existam outras funções implementadas com outros métodos.

Solução de EDOs

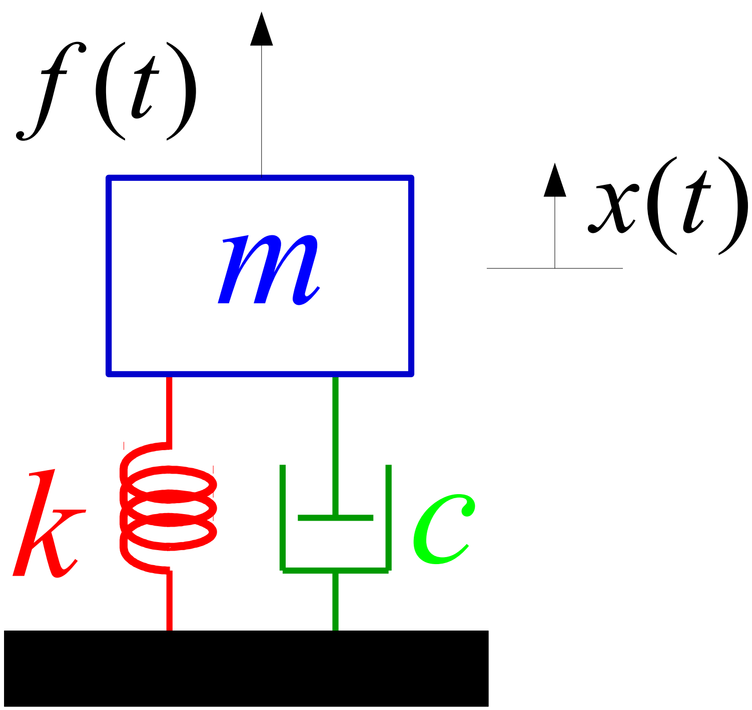
- Exemplo de criação (usando a `function` implementada antes):

```
h = 0.001;   % Passo [s]
tfim = 10;   % Tempo final [s]
t = 0:h:tfim;    % Vetor de tempo [s]
y0 = [1 0];      % Condições iniciais

[t,y] = ode45(@edoex01,t,y0);          % Roda ODE45

figure;
subplot(211);
plot(t,y(:,1),'linewidth',2);
xlabel('Tempo [s]');
ylabel('y');grid on;
subplot(212);
plot(t,y(:,2),'linewidth',2);
xlabel('Tempo [s]');
ylabel('dy/dt');grid on;
```

- Exercício 03 – Simule a resposta de um sistema massa-mola-amortecedor excitado por uma força senoidal durante 20 segundos com os parâmetros mostrados abaixo. Plote *x* e *dx/dt* para poder analisar a resposta:



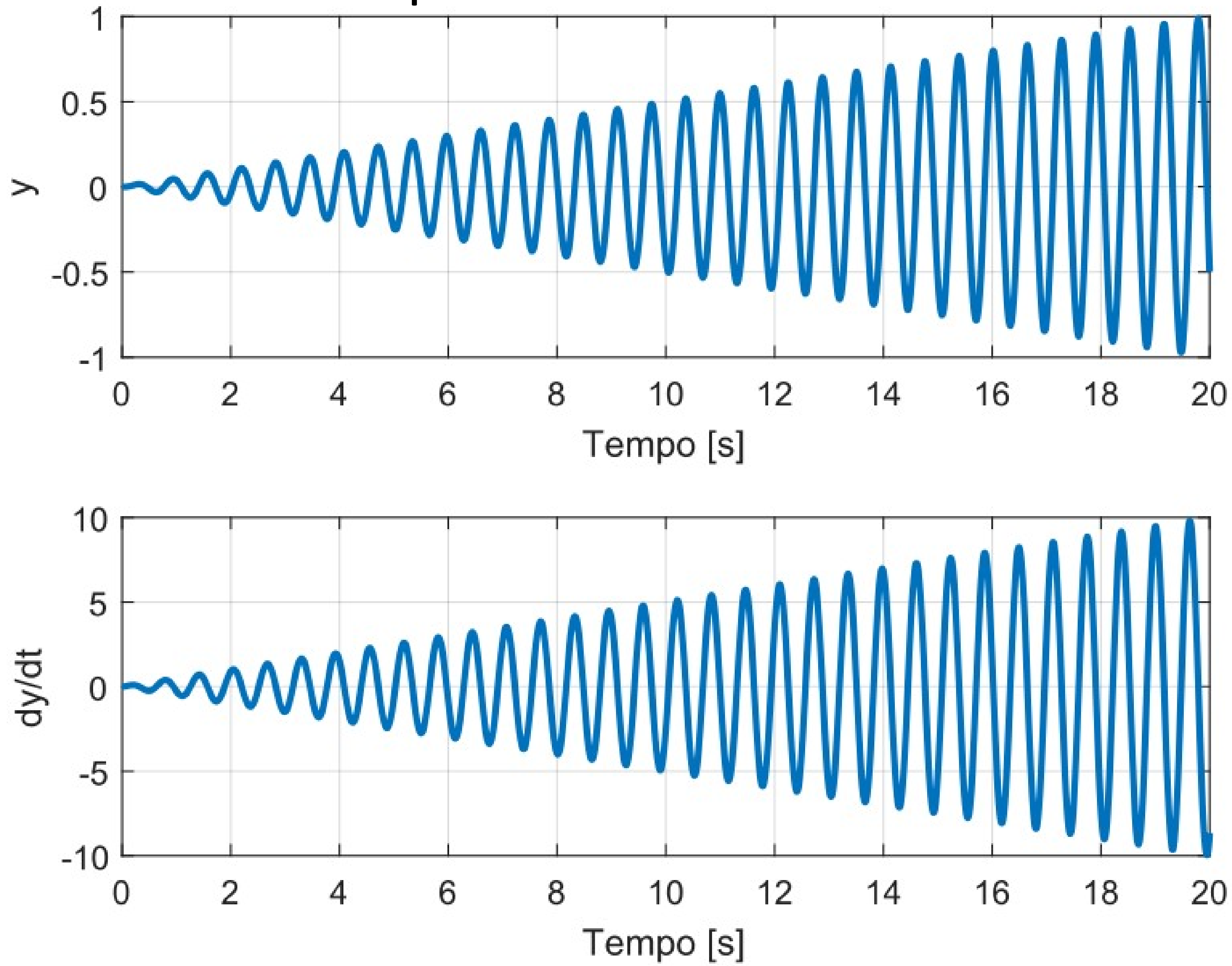
$$m = 1\,\text{kg}; c = 0\,\text{Ns/m}; k = 100\,N/m$$

$$f(t) = \sin\left(t\sqrt{k/m}\right)$$

$$\boxed{m\ddot{x} + c\dot{x} + kx = f(t)}$$

Solução de EDOs

- Exercício 03 – Resposta:



Solução de EDOs

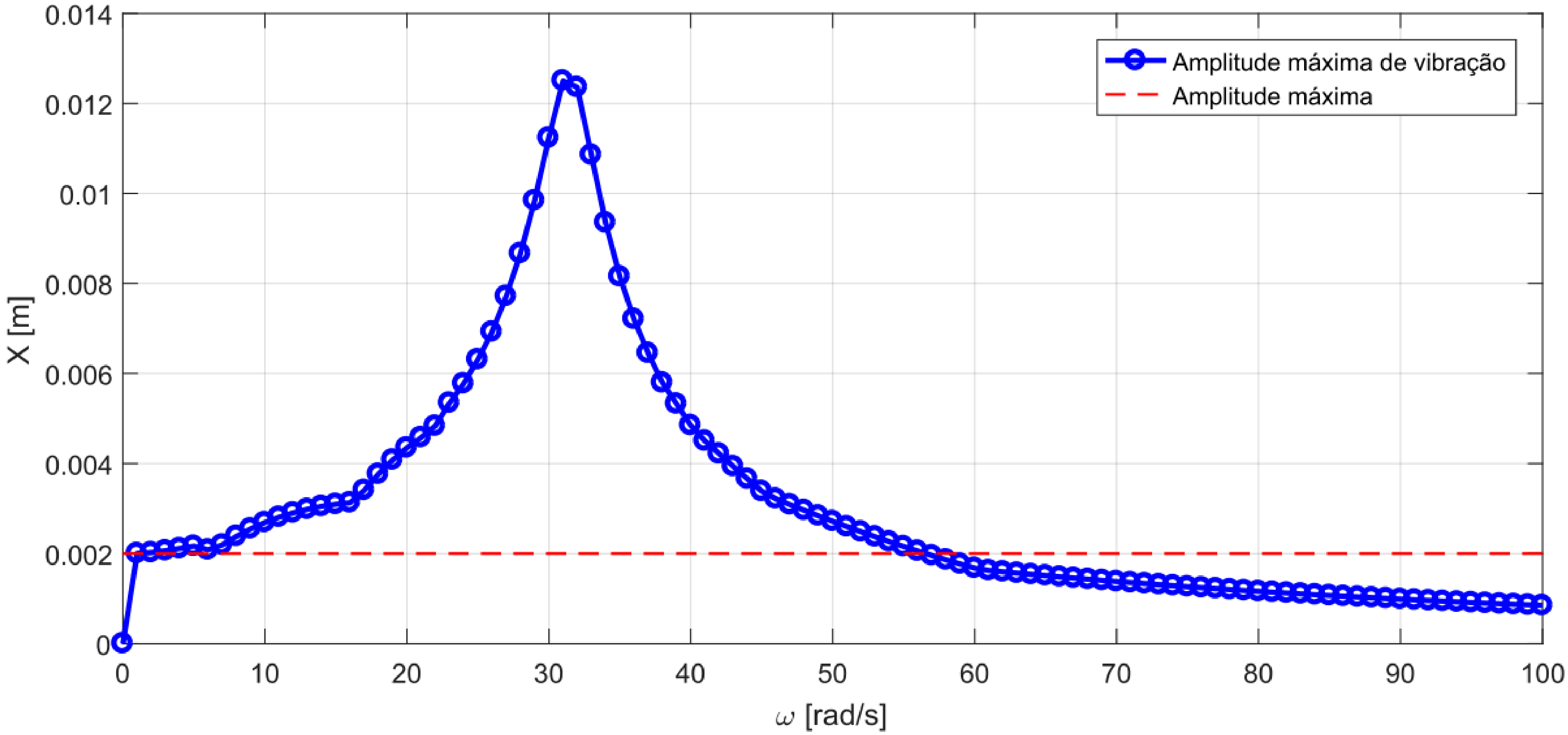
- Exercício 04 – A EDO representa as vibrações do compressor abaixo sob uma rotação ω [rad/s]. Essa máquina está instalada em uma estrutura que pode aguentar um máximo deslocamento de 0,002 metros. Encontre as rotações que essa máquina pode ter sem afetar a estrutura.

$$\ddot{x}+5\dot{x}+1000x=2\sin(\omega t)$$

Dicas: usar algum tipo de loop pode ajudar a fazer os cálculos. Se precisar variar algum parâmetro dentro da @odefun você pode criar uma variável global dentro da function e da rotina principal (sintaxe: `global a;`)

Fonte: https://www.royalmaquinas.com.br/compressor-ar-10br-100l-csv-2cv-mono-schulz.html

- Exercício 04 – Resposta



Perguntas ?