Universidade Federal de Minas Gerais
Departamento de Engenharia Eletrônica
ELT129 – Oficina de Modelagem e Simulação

Professor: Leonardo Mozelli – lamoz@ufmg.br

Tutorial 6 – Simulação de Sistemas Dinâmicos

1 Introdução

Em geral, a maioria dos sistemas existentes apresenta algum tipo de não linearidade. Neste tutorial são apresentadas maneiras de simular numericamente o comportamento de tais sistemas para diferentes entradas e condições iniciais.

2 Sistemas Discretos

A evolução temporal de sistemas discretos pode ser vista como uma sequência de números (amostras) que é regida pela equação dinâmica do sistema. As amostras do sistema ocorrem em intervalos bem especificados sendo possível realizar a simulação de sistemas discretos por meio de comandos iterativos como o `for` e o `while`. A estrutura vetorial fornecida pelo Matlab é bastante útil para a simulação de tais sistemas.

Considere, por exemplo, a equação logística

$$x[k+1] = rx[k](1 - x[k]), \quad r > 0 \tag{1}$$

que pode ser utilizada para representar a dinâmica populacional em termos percentuais. Para condições iniciais $x[0] \in [0,\ 1]$ e $0 \leq r \leq 4$, a sequência $x[n]$ permanece sempre dentro do intervalo. Igualando (2) a zero, é possível determinar os pontos de equilíbrio do sistema, os quais são

$$\bar{x}_1 = 0 \quad , \quad \bar{x}_2 = 1 - \frac{1}{r}$$

Caso o sistema seja inicializado sobre um desses pontos, $x[k]$ permanecerá no mesmo valor para todos os instantes de tempo k.

Para simular o comportamento do sistema em um horizonte de tempo, o seguinte código pode ser utilizado:

```
r = 2.7;          % valor de r no intervalo [0,4]
T = 30;           % tempo de simulacao
x = zeros(1,T);   % inicializacao do vetor x com zeros
x(1) = 0.95;      % condicao inicial

for k = 1:T-1
    x(k+1) = r*x(k)*(1-x(k));
end
```

Note que, apesar de não ser mandatório, é uma boa prática inicializar, sempre que possível, vetores e matrizes, pois pode-se economizar em tempo de processamento.

O código acima simula a equação (2) ao longo de 30 amostras e armazena, em cada posição do vetor, um "retrato" do sistema no instante k. Para visualizar os resultados da simulação, poderia ser utilizado o comando `plot`. No entanto, esse comando é pouco interessante quando o sistema em estudo é discreto, pois ele realiza a interpolação dos pontos do vetor $x[k]$. A melhor alternativa é utilizar o comando `stem`.

```
tempo = linspace(0,29,30);    % gera 30 pontos igualmente
                              % espacados entre 0 e 29
stem(tempo,x)
xlabel('Tempo (amostras)')    % nomeando o eixo
```

3 Sistemas contínuos

Diferentemente de sistemas discretos, a simulação de sistemas contínuos precisa abranger instantes de tempo não inteiros. Uma possibilidade seria gerar uma quantidade muito grande de pontos em um certo intervalo de tempo $[t_0,\ t_f]$ e tentar proceder como no caso discreto. Esse método, infelizmente, é pouco eficiente e, dependendo do tipo de sistema analisado, pode ser inadequado. A solução é recorrer a métodos de integração numérica para resolver a equação dinâmica do sistema, usualmente, dada na forma

$$\dot{x}(t) = f(t, x(t)) \tag{2}$$

sendo x a variável do sistema, t a variável tempo e $f(t, x)$ uma função, possivelmente não linear, que mapeia x e t para $\dot{x}(t)$.

3.1 Métodos de integração numérica

Integrando, analiticamente, a equação (3) tem-se

$$\int_{t_0}^{t} \dot{x}(\tau)d\tau = \int_{t_0}^{t} f(\tau, x(\tau))d\tau \quad \Rightarrow \quad x(t) = x(t_0) + \int_{t_0}^{t} f(\tau, x(\tau))d\tau \tag{3}$$

Supondo que a função $f(t, x(t))$ seja bem comportada no intervalo $[t_0, t]$, isto é, não possua descontinuidades infinitas e nem singularidades, é possível empregar métodos numéricos para determinar o valor da integral do lado direito da equação (3.1).

3.1.1 Método de Euler

O *método de Euler* é um dos mais simples de ser aplicado e consiste em aproximar a integral da função $f(t, x(t))$ (área sob a curva) por meio da soma das áreas de retângulos de "largura" constante, como ilustrado na Figura 1. Neste caso, a "largura" do retângulo é denominada *passo de integração* ou apenas *passo*.

Para um determinado instante de tempo, t_k, sabe-se, por (3.1), que

$$x(t_k) = x(t_0) + \int_{t_0}^{t_k} f(\tau, x(\tau))d\tau$$

para determinar o valor de x um passo à frente, ou seja, $t_{k+1} = t_k + h$, pelo método de Euler procede-se da seguinte forma

$$x(t_{k+1}) = x(t_0) + \int_{t_0}^{t_{k+1}} f(\tau, x(\tau))d\tau$$

$$x(t_{k+1}) = x(t_0) + \int_{t_0}^{t_k+h} f(\tau, x(\tau))d\tau = \underbrace{x(t_0) + \int_{t_0}^{t_k} f(\tau, x(\tau))d\tau}_{x(t_k)} + \int_{t_k}^{t_k+h} f(\tau, x(\tau))d\tau \tag{4}$$

$$x_{k+1} \approx x_k + hf(t_k, x_k)$$

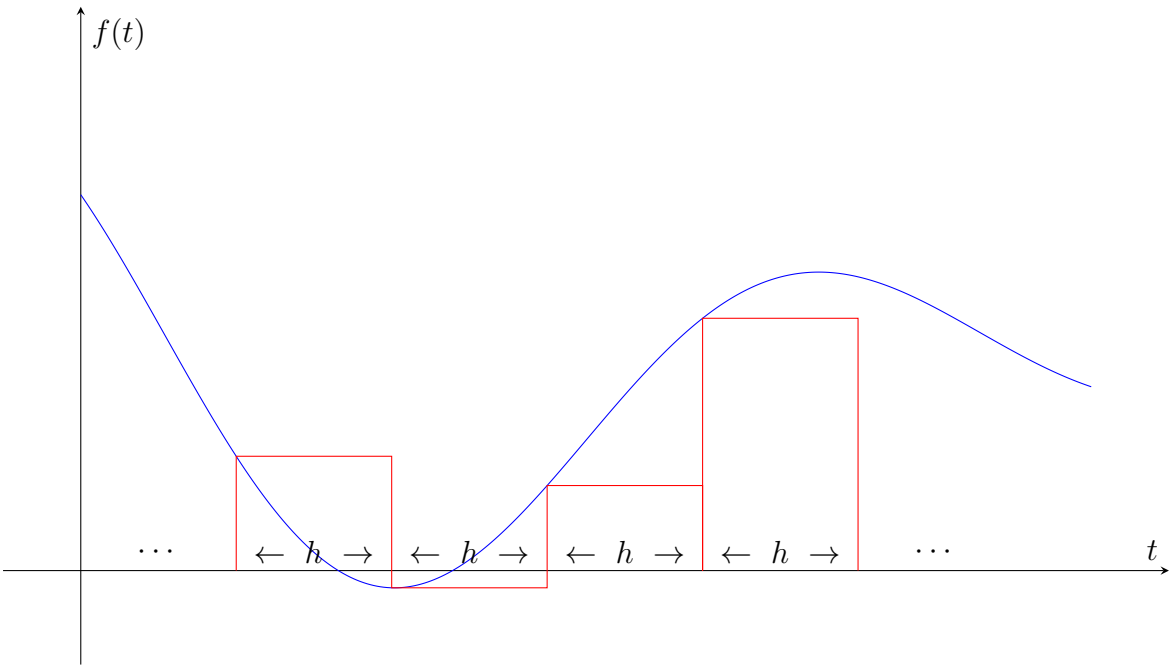

Figura 1: Método de Euler para resolução de integrais.

ou seja, para saber o valor da integral da função um passo à frente, basta conhecer o valor da função no instante atual e somar com $hf(t_k, x_k)$.

Note que há um erro inerente à aproximação adotada que pode ser reduzido adotando valores pequenos para o passo h. Todavia, valores muito pequenos podem esbarrar em erros de aproximação numérica, fazendo o método divergir ou demorar muito para encontrar uma solução.

3.1.2 Método do trapézio

Uma forma de minimizar o erro de aproximação da integral do método de Euler é, por exemplo, tomar mais pontos dentro do intervalo de interesse. Se os pontos no início e no final do intervalo forem considerados, como ilustrado na Figura 2, a aproximação de (3.1.1) é tal que

$$x_{k+1} = x_k + \int_{t_k}^{t_k+h} f(\tau, x(\tau))d\tau \approx x_k + \frac{h}{2}\left(f(t_k, x_k) + f(t_{k+1}, x_{k+1}))\right) \tag{5}$$

sendo esse método denominado *Método do trapézio*.

Note que x_{k+1} aparece dos dois lados da relação (3.1.2), sendo necessário resolver a relação implicitamente para obter x_{k+1} de forma isolada. Por esse motivo, o método do trapézio é dito ser um *método implícito*, enquanto o método de Euler é um *método explícito*.

3.1.3 Métodos de Runge-Kutta

Métodos de Runge-Kutta correspondem a uma grande classe de métodos que podem ser escritos na forma

$$x_{k+1} = x_k + h\sum_{i=1}^{m} \gamma_i \kappa_i$$

$$\kappa_i = f\left(t_k + \alpha_i h, x_k + h\sum_{j=1}^{i-1} \beta_j \kappa_j\right), \quad i = 1, \ldots, m$$

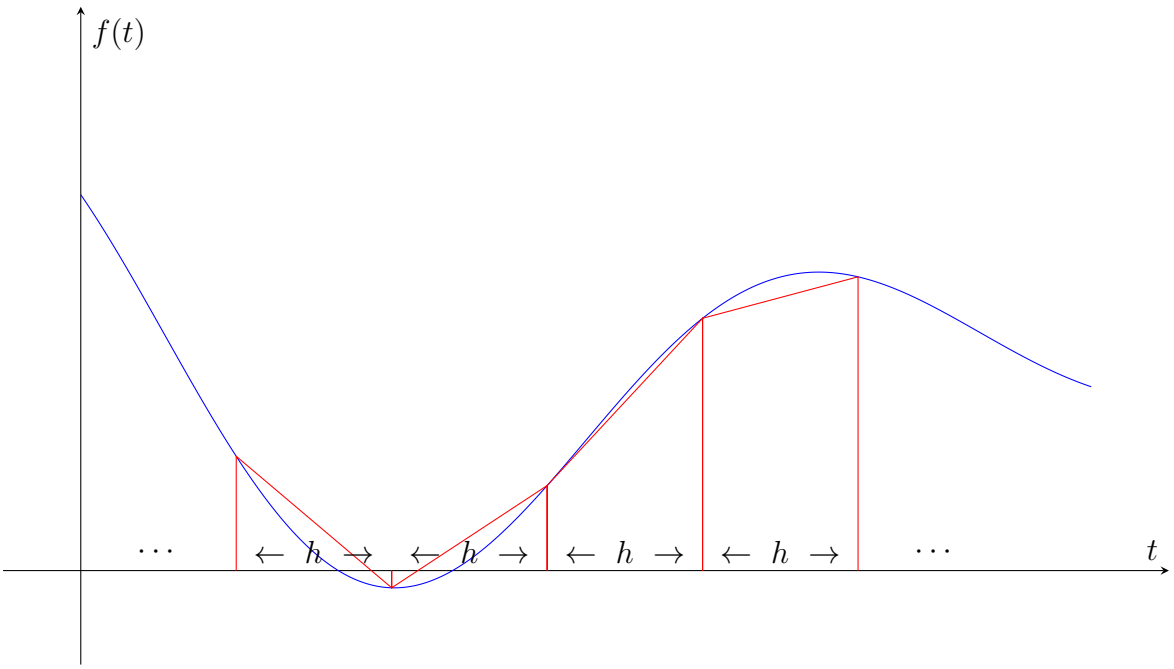

Figura 2: Método do trapézio para resolução de integrais.

Note que tais métodos requerem m avaliações da função $f(t, x(t))$.

O método clássico de Runge-Kutta possui a seguinte expressão

$$\begin{aligned} x_{k+1} &= x_k + \frac{h}{6}(\kappa_1 + \kappa_2 + \kappa_3 + \kappa_4) \\ \kappa_1 &= f(t_k, x_k) \\ \kappa_2 &= f(t_k + h/2, x_k + h\kappa_1/2) \\ \kappa_3 &= f(t_k + h/2, x_k + h\kappa_2/2) \\ \kappa_4 &= f(t_k + h, x_k + h\kappa_3) \end{aligned}$$

que corresponde a avaliações da função $f(t, x(t))$ no início, no final e em dois pontos intermediários do intervalo de integração. Importante salientar que o método clássico de Runge-Kutta é explícito.

Ademais, os métodos de Euler e do trapézio podem ser vistos como casos particulares dos métodos de Runge-Kutta.

3.1.4 Métodos de passo variável

Todos os métodos previamente mencionados consideram que o passo de integração é fixo. No entanto, há situações nas quais a função sendo integrada varia rapidamente em certos intervalos e, em outros intervalos, a sua variação é muito lenta. Nesses casos, é interessante utilizar um método que adapte o tamanho do passo segundo algum critério, buscando otimizar o processo de integração.

Tipicamente, deseja-se que o erro cometido entre o valor $x(t_k)$ e sua aproximação x_k seja menor que uma certa tolerância, ou seja,

$$e_k = \|x_k - x(t_k)\| \leq \text{TOL}$$

Sabendo que o erro cometido por um método de ordem[1] p pode ser aproximado por $e_k \approx$

[1] A ordem de um método tem relação direta com a magnitude do erro de aproximação. Quanto maior a ordem do método, maior a sua precisão e, consequentemente, menor o seu erro. Por exemplo, o método de Euler é de ordem 1, ao passo que o método do trapézio é de ordem 2.

Ch^{p+1}, sendo C uma constante de proporcionalidade, o passo ótimo h_{opt} para o método é tal que $Ch_{\text{opt}}^{p+1} \approx \text{TOL}$. Assim, o passo ótimo a cada instante de tempo pode ser determinado por meio da seguinte relação

$$h_{\text{opt}} = h \left(\frac{\text{TOL}}{e_k} \right)^{\frac{1}{p+1}}$$

O *método de Dormand-Prince* é um método de Runge-Kutta de quarta ordem com passo variável, sendo o passo ajustado de acordo com um método de quinta ordem, utilizado para avaliar o erro de integração. No Matlab, esse método é implementado sob o nome de `ode45` e é um dos métodos de integração mais utilizados, devido à sua boa acurácia e à sua velocidade de execução. A Tabela 1 apresenta um resumo dos principais métodos de integração (comandos) presentes no Matlab[2].

Tabela 1: Principais métodos de integração disponíveis no Matlab.

Método	**Classificação**	**Acurácia**	**Uso**
`ode45`	Explícito	Média	Primeiro método a se tentar.
`ode23`	Explícito	Baixa	Maiores tolerâncias ou problemas moderadamente *stiff*.
`ode15s`	Implícito	Baixa a média	Se `ode45` é lento devido a *stiffness*.
`ode23s`	Implícito	Baixa	Problemas *stiff* com altas tolerâncias. Mais estável que `ode15s`.
`ode113`	Explícito (múltiplos passos)	Alta	Baixas tolerâncias ou problemas computacionalmente intensos.

3.2 Exemplo

Considere o circuito RC apresentado na Figura 3, que pode ser utilizado para a descarga de um capacitor. A EDO que descreve o comportamento desse sistema é dada por

$$RC\dot{x} + x = 0$$

com $x(t)$ a tensão medida nos polos do capacitor ao longo do tempo.

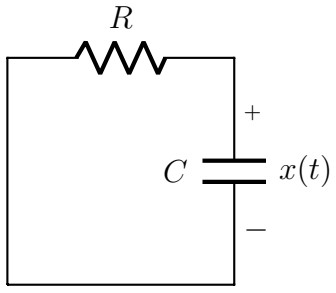

Figura 3: Circuito RC.

[2]Problemas do tipo *stiff* são aqueles nos quais a EDO possui modos com escalas de tempo separadas por diversas ordens de magnitude.

Supondo o capacitor inicialmente carregado $x(0) = 10\,\text{V}$, $R = 33\,\text{k}\Omega$ e $C = 10\mu\text{F}$, o seguinte código pode ser utilizado para simular o comportamento do sistema durante a descarga do capacitor. A curva de descarga é apresentada na Figura 4

```
function descargaCapacitor

R = 33e3;
C = 10e-6;
x0 = 10;
tfinal = 2;

[tout, xout] = ode45(@(t,x)simulaCircuitoRC(t,x,R,C),[0 tfinal
   ],x0);

plot(tout,xout)
xlabel('Tempo [s]')
ylabel('Tensao [V]')
title('Curva de descarga de um capacitor')

end

function dx = simulaCircuitoRC(t,x,R,C)

dx = -x/(R*C);

end
```

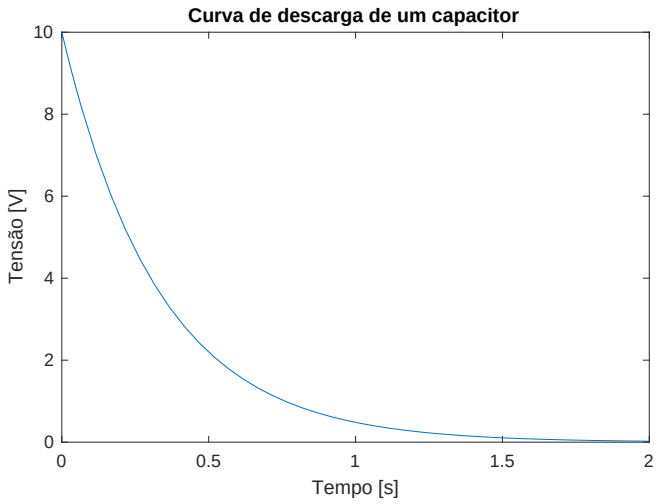

Figura 4: Curva de descarga de um capacitor.

A função `ode45` em sua forma mais simples recebe três parâmetros (a função que se deseja integrar, um vetor com os tempos inicial e final e um vetor de condições iniciais) e retorna dois vetores: um com os tempos de simulação e outro com os valores da função ao longo do tempo.

Como o interesse é passar uma função definida pelo usuário, faz-se necessário utilizar a notação

```
@(t,x)simulaCircuitoRC(t,x,R,C)
```

O símbolo `@` indica que se está passando o *handler* de uma função, `(t,x)` é um argumento mandatório que corresponde aos vetores de tempo e condições iniciais passados para o `ode45`

(segundo e terceiro argumentos) e a função `simulaCircuitoRC` deve receber como primeiros argumentos `t` e `x` nessa ordem. Os demais argumentos `R` e `C` são adicionais e podem ser passados em qualquer ordem.

Um outro ponto interessante de se notar é que há duas funções declaradas dentro do mesmo arquivo. A primeira função, `descargaCapacitor`, possui o mesmo nome do arquivo e é a principal. A segunda função, `simulaCircuitoRC`, é uma função auxiliar utilizada pelo `ode45` e sua visibilidade está restrita ao escopo do arquivo. Se for desejável construir uma função de simulação passível de ser acessada por outras funções ou *scripts*, recomenda-se criar um arquivo separado contendo somente a função de interesse.

Mude os valores de R e C e verifique como eles impactam na curva de descarga.

3.3 Resolução de equações diferenciais de ordem superior

Nas seções anteriores foi apresentado como resolver numericamente equações diferenciais de primeira ordem na forma $\dot{x} = f(t, x(t))$. No entanto, muitos sistemas são representados por equações dinâmicas cuja ordem é ao menos dois, por exemplo, sistemas mecânicos.

Os métodos numéricos previamente apresentados são limitados a resolução de equações diferenciais de primeira ordem, mas podem ser aplicados a sistemas descritos na forma vetorial. Essa é a chave para a resolução numérica de equações diferenciais de ordem superior: converter a equação para uma série de equações diferenciais com múltiplas variáveis. Essa abordagem usualmente é denominada *espaço de estados*, possuindo, também, outros tipos de aplicação.

Para realizar a conversão para o espaço de estados, considere, por exemplo, a seguinte equação diferencial de terceira ordem e condições iniciais de um sistema dinâmico

$$\dddot{y} + \alpha\ddot{y} + \beta\dot{y} + \gamma y = 0 \quad , \quad y(0) = y_0,\ \dot{y}(0) = \dot{y}_0,\ \ddot{y}(0) = \ddot{y}_0 \tag{6}$$

com α, β e γ valores conhecidos. O primeiro passo é arbitrar quais serão os *estados* para o sistema transformado. Usualmente, escolhem-se, como estados, as variáveis sendo derivadas, sendo necessário um número de estados igual à ordem da equação diferencial. No exemplo considerado, 3 estados são necessários. Assim,

$$x_1 = y \quad , \quad x_2 = \dot{y} \quad \text{e} \quad x_3 = \ddot{y} \tag{7}$$

Note que

$$\dot{x}_1 = x_2 \quad \text{e} \quad \dot{x}_2 = x_3$$

ademais, substituindo (3.3) em (3.3) e rearranjando, tem-se

$$\dot{x}_3 = -\alpha x_3 - \beta x_2 - \gamma x_1 \quad , \quad x_1(0) = y_0,\ x_2(0) = \dot{y}_0,\ x_3(0) = \ddot{y}_0$$

Portanto, o sistema em sua forma de estados pode ser escrito, de modo genérico, como

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} x_2 \\ x_3 \\ -\alpha x_3 - \beta x_2 - \gamma x_1 \end{bmatrix}}_{f(t,x(t))} \quad , \quad x(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} y_0 \\ \dot{y}_0 \\ \ddot{y}_0 \end{bmatrix}$$

Importante salientar que a inter-relação entre as variáveis de estado deve ser preservada e aparecer na estrutura final, sob a pena de o sistema transformado não mais corresponder ao sistema original. Além disso, escolhas diferentes para as variáveis de estado levam a representações distintas. De forma geral, *há diversas representações em espaço de estados para um mesmo sistema dinâmico.*

3.4 Exemplo – Simulação de um pêndulo simples

Considere um sistema pendular como ilustrado na Figura 5. A haste do pêndulo possui comprimento L, a massa m está concentrada na parte de baixo e o pêndulo oscila em torno de um ponto fixo preso ao teto. Supondo que o atrito do ar seja desprezível e que não haja atrito no ponto de engaste, o modelo desse sistema é descrito por

$$mL\ddot{\theta} + mg\text{sen}\,(\theta) = 0$$

sendo g a aceleração da gravidade e θ o ângulo do pêndulo com a vertical.

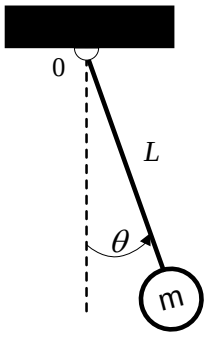

Figura 5: Pêndulo simples.

Antes de simular o sistema, é necessário descrevê-lo no espaço de estados. Tome, então,

$$x_1 = \theta \quad \text{e} \quad x_2 = \dot{\theta}$$

desse modo, a representação de estados é tal que

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{L}\text{sen}\,(x_1) \end{bmatrix} \tag{8}$$

Tomando a representação (3.4), pode-se escrever o seguinte programa para simular o comportamento do pêndulo para uma certa condição inicial `x0` fornecida como entrada:

```
function simulaPenduloSimples(x0)

g = 9.81;    % m/s^2
L = 0.5;  % m

tfinal = 5;

[tout, xout] = ode45(@(t,x)simulaPendulo(t,x,g,L),[0 tfinal],x0
   );

plot(tout,xout)
xlabel('Tempo [s]')
ylabel('Angulo [rad] e velocidade [rad/s]')
title('Comportamento de um pendulo simples')
legend({'$\theta$','$\dot{\theta}$'},'Interpreter','Latex')

end
```

```
function dx = simulaPendulo(t,x,g,L)

dx = zeros(2,1);
dx(1) = x(2);
dx(2) = -(g/L)*sin(x(1));

end
```

Neste caso, é importante definir, dentro da função `simulaPendulo` a dimensão do vetor de saída `dx`. Isso porque a função `ode45` espera como retorno um vetor coluna e, por padrão, se nada for previamente informado, o Matlab cria vetores linha.

O gráfico apresentado na Figura 6 ilustra o comportamento do sistema ao longo do tempo para uma condição inicial `x0 = [pi/4; 0]`.

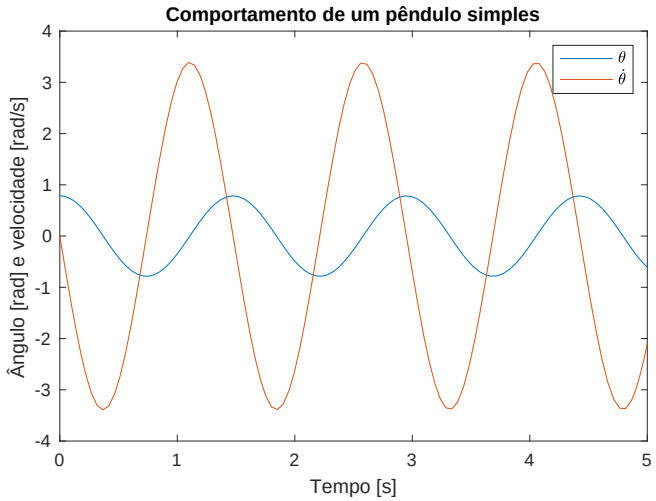

Figura 6: Simulação do pêndulo simples para uma condição inicial $x(0) = [\pi/4,\ 0]^T$.

4 Exercícios

1. Considere um sistema de primeira ordem descrito por

$$\dot{x} = f(t, x) = -x^3, \quad x(0) = x_0$$

Defina um certo horizonte de tempo T e escolha uma condição inicial não nula. Utilizando o que foi apresentado neste tutorial, simule o sistema para ao menos duas condições iniciais e apresente em um gráfico a evolução ao longo do tempo de $x(t)$.

2. Seja um sistema não linear descrito pela equação diferencial

$$\ddot{y} + 0.02\dot{y} + y + 5y^3 = 8\cos(0.5t)$$

Pede-se:

a) Apresente a descrição em espaço de estados do sistema.

b) Supondo condições iniciais nulas e um horizonte de tempo de, ao menos, $T = 200\,\text{s}$, simule o comportamento do sistema e apresenta os gráficos de $y(t)$ e $\dot{y}(t)$ ao longo do tempo. Plote também o gráfico $\dot{y}(t) \times y(t)$, o qual é denominado *diagrama de fase*. O que é possível observar?

3. Considere o sistema massa-mola-amortecedor apresentado no Tutorial 3, o qual é reproduzido na Figura 7 e cujo modelo matemático em torno do ponto de equilíbrio estático é

$$m\ddot{y}(t) + c\dot{y}(t) + ky(t) = f(t)$$

Tomando $m = 3\,\text{kg}$, $c = 1\,\text{Ns/m}$ e $k = 10\,\text{N/m}$, pede-se:

a) Apresente a descrição em espaço de estados dos sistema.

b) Simule o sistema para $f(t) = 3u(t)$ e apresente a evolução da posição vertical da massa m ao longo do tempo.

c) Repita o item anterior, mantendo $k = 10\,\text{N/m}$, mas escolha dois valores distintos para c, um maior e outro menor. O que é possível observar em cada caso?

d) Repita o item anterior, matendo $c = 1\,\text{Ns/m}$, mas escolha dois valores distintos para k, um maior e outro menor. O que é possível observar em cada caso?

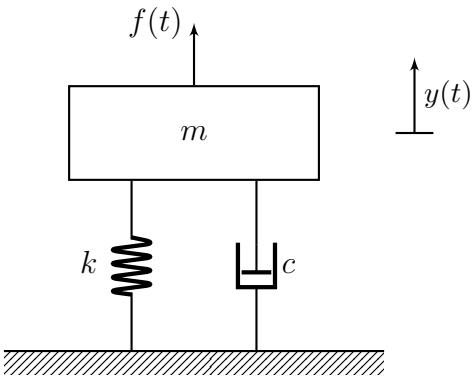

Figura 7: Sistema massa-mola-amortecedor.

4. Considere o sistema descrito pelo seguinte par de EDOs

$$\begin{aligned} \ddot{x} + 4x - y &= 0 \\ \ddot{y} + cy &= 0 \end{aligned}$$

com $x(0) = \dot{x}(0) = y(0) = \dot{y}(0) = 1$.

a) Para $c = 9$ e $t \in [0,\ 200]$, simule o sistema e plote o diagrama de fase $x \times \dot{x} \times y$. **Dica:** utilize o comando `plot3`.

b) Repita o item acima para $c = 10$ e com um horizonte de simulação maior. O que é possível observar com relação a ambas as figuras?