

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ENGENHARIA DE CONTROLE E AUTOMAÇÃO
PROGRAMAÇÃO ORIENTADA À OBJETO

SEMINÁRIO EM JAVA
QUARTO DE SUSPENSÃO
Linear

Leonardo Ferreira
00576066
Lucas Yuuki Sasaki Ramos
00588802
David Luís Leite
00591311

Porto Alegre
2025

1. INTRODUÇÃO

Este Capítulo irá explicitar o problema selecionado para aplicação na Programação Orientada à Objetos em Java.

1.1. O QUARTO DE SUSPENSÃO VEICULAR

Um quarto de suspensão é um modelo que abstrai o comportamento de um veículo simplificando a análise para uma única roda, metade de um eixo. Em sua configuração clássica, conforme apresentado na Figura 1.1, a suspensão de um veículo é composta por três elementos principais:

1. Elemento Elástico: Tipicamente uma mola helicoidal, este elemento fornece uma força proporcional e oposta ao alongamento da suspensão, suportando a carga estática do veículo;
2. Elemento de amortecimento: Geralmente um amortecedor hidráulico, oferece uma força dissipativa contra a velocidade de alongamento. Este componente é crucial em situações dinâmicas, mas fornece força insignificante em estados estacionários;
3. Articulações Mecânicas: Esses componentes conectam a carroceria suspensa do veículo à massa não suspensa, cruciais para a funcionalidade geral do sistema de suspensão.

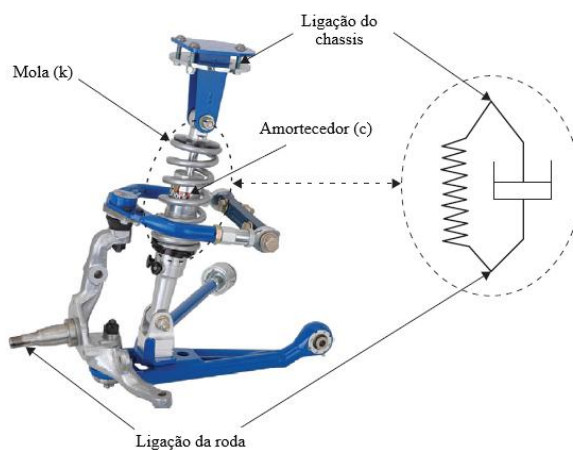


Figura 1.1 – Esquema clássico de suspensão veicular [Adaptado de Saravesi et al., 2010].

Para fins de modelagem, podemos abstrair o conceito para um sistema massa, mola e amortecedor conforme a Figura 1.2.

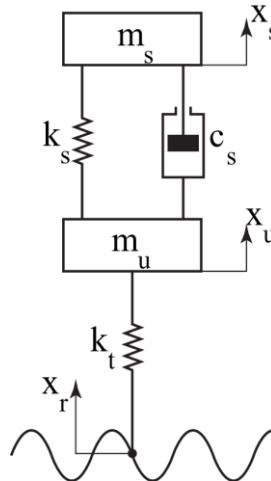


Figura 1.2 – Quarto de suspensão adaptado [Fonte: Adaptado de Ogata, 2010].

Esta abstração inclui:

- Massa suspensa (m_s): a carroceria do veículo suportada pela suspensão;
- Massa não suspensa (m_u): a parte do veículo abaixo da suspensão, incluindo pneus, cubo de roda e eixos;
- Mola (k_s): elemento elástico que suporta a massa suspensa e impactos;
- Amortecedor (c_s): elemento que dissipa energia em forma de calor, reduzindo oscilações;
- Rigidez do pneu (k_t): representa a medida de deflexão do pneu em contato com o solo;
- Deslocamento da estrada (x_r): perfil do solo que a roda segue, pode ser definido por uma função senoidal;
- O amortecimento do pneu será negligenciado;
- (x_u): posição da massa não suspensa;
- (x_s) posição da massa suspensa.

No modelo padrão de um quarto de suspensão, as massas da mola e do amortecedor normalmente não são consideradas explicitamente. Essa simplificação pressupõe que as massas da mola e do amortecedor são insignificantes em comparação com a massa suspensa e a massa não suspensa.

A mola e o amortecedor são agrupados com suas respectivas massas de conexão, o que significa:

- A massa suspensa inclui todos os componentes acima da suspensão, incluindo a carroceria do veículo e a carga do passageiro;
- A massa não suspensa inclui componentes como a roda, o pneu e as peças do sistema de suspensão (por exemplo, o braço de ligação inferior).

Portanto, o objetivo desta modelagem é simular a dinâmica de um sistema de suspensão veicular via Java e levando em conta as especificações da Programação Orientada à Objetos.

2. METODOLOGIA

Este Capítulo abordará a modelagem geral do sistema de suspensão proposto, incluindo as Equações de Movimento que o definem e a Introdução à Representação de Espaço de Estados, onde também se utilizou o *software* MATLAB 2024b em paralelo com a IDE (IntelliJ e Eclipse) Java.

2.1. Equações de Movimento

A modelagem é feita com base na Segunda Lei de Newton ($\sum \vec{F} = m\vec{a}$), podemos equacionar o sistema conforme a Equação 2.1 e a Equação 2.2:

$$m_s \ddot{x}_s = k_s(x_u - x_s) + c_s(\dot{x}_u - \dot{x}_s) \quad (2.1)$$

$$m_u \ddot{x}_u = -k_s(x_u - x_s) - c_s(\dot{x}_u - \dot{x}_s) + k_t(x_r - x_u) \quad (2.2)$$

2.1.1. Variáveis e Parâmetros Necessários

m_s : Massa suspensa (carroceria);

m_u : Massa não suspensa (pneu e eixo);

k_s : Rigidez da suspensão, influencia no conforto veicular e na habilidade de absorção das irregularidades da rua;

c_s : Amortecimento da suspensão, reduz as vibrações (conforto) e garante o contato do pneu com o solo (controle);

k_t : Rigidez do pneu, afeta a transmissão de vibração da rua;

x_r : Deslocamento imposto pelo solo, que pode ser definido como:

- Perfil harmônico (senoidal): $x_r = A \sin(2\pi t)$. Conforme a Figura 2.1.

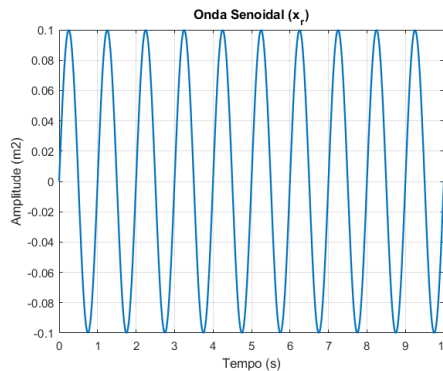


Figura 2.1 – Perfil senoidal usado como entrada [Fonte: Elaborado pelos autores, 2024].

Onde a amplitude (A) representa a altura dos solavancos em metros. Essa entrada senoidal impacta na força transmitida ao conjunto de suspensão.

2.1.2. UML

Desta forma o UML do programa foi feito seguindo o que era necessário para rodar fazer os cálculos, conforme a Figura 2.2 abaixo.

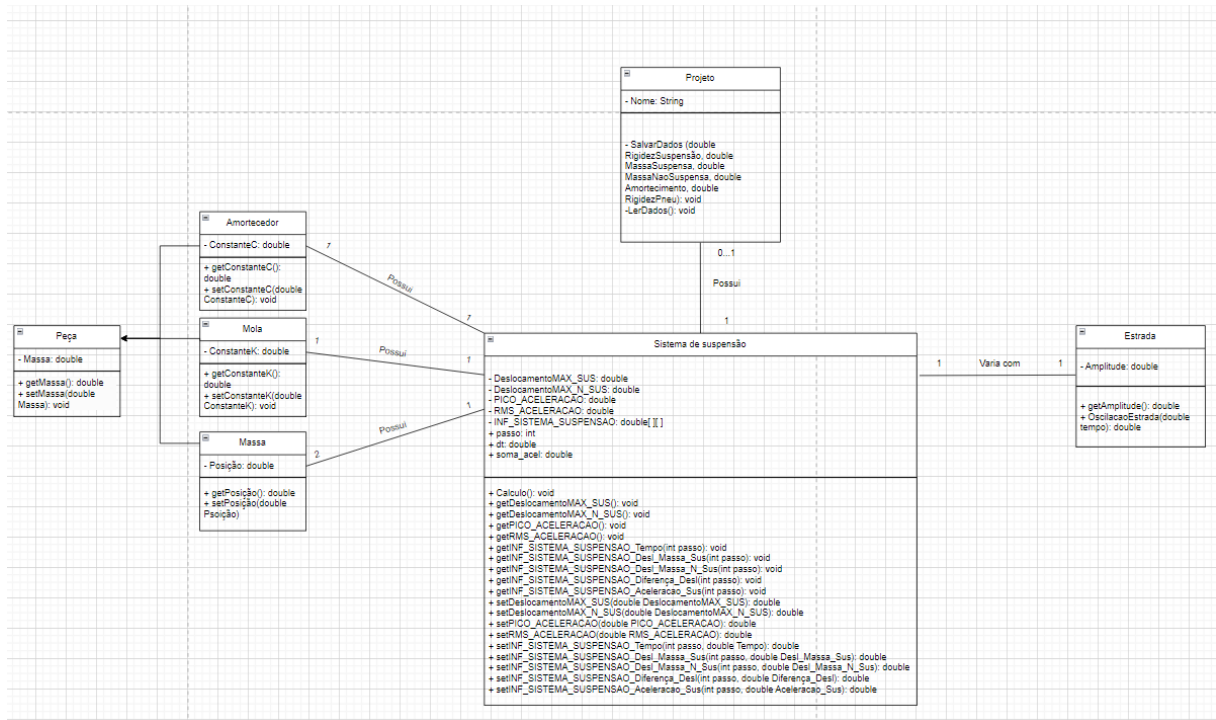


Figura 2.2 – UML completo [Fonte: Elaborado pelos autores, 2025].

Para uma melhor visualização, foi feito um UML sem os *getters* e *setters* das variáveis da classe sistema de suspensão, isso pode ser visto na Figura 2.3 abaixo.

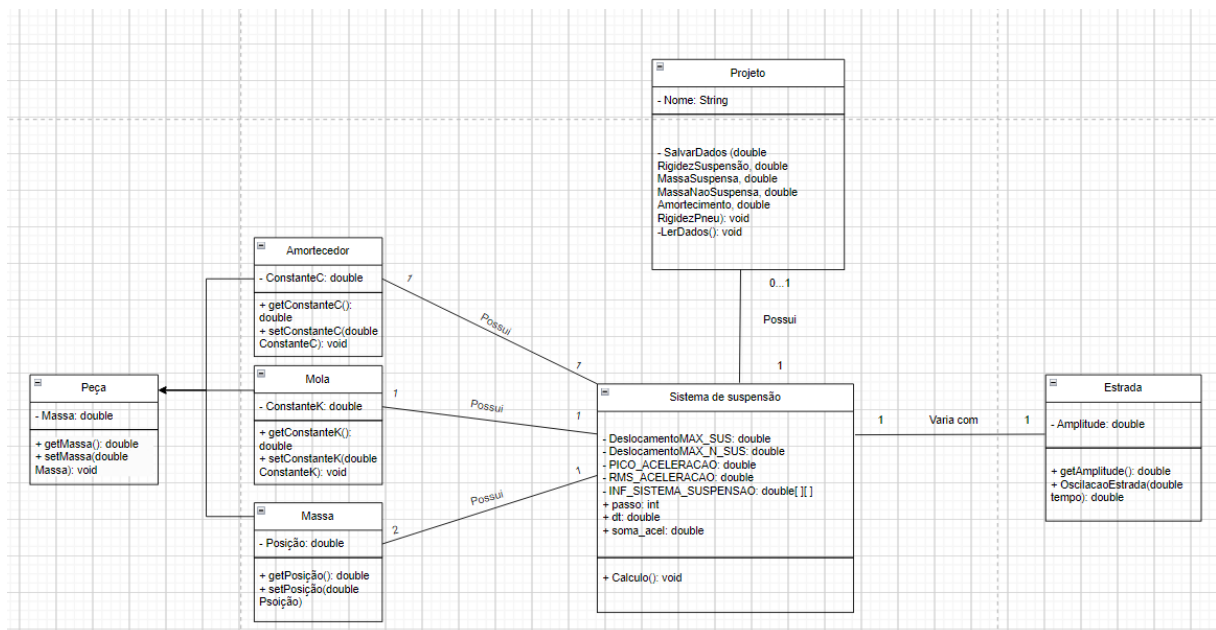


Figura 2.3 – UML sem os *getters* e *setters* [Fonte: Elaborado pelos autores, 2025].

2.1.3. Métricas Analíticas

Para avaliar a dinâmica veicular, focaremos em:

- Conforto de veicular: Medido através da aceleração da massa suspensa (\ddot{x}_s). Acelerações baixas melhoram o conforto veicular;
- Deslocamento da suspensão: Medido pelo deslocamento relativo entre a massa suspensa e não suspensa ($x_u - x_s$). O curso excessivo leva a restrições mecânicas.

2.1.4. Valores Ideais para os Parâmetros

Para um típico veículo de passageiros:

- Massa suspensa: 250 – 500 kg (1/4 da massa do veículo);
- Massa não suspensa: 25 – 75 kg;
- Rigidez da suspensão: 10 000 – 50 000 N/m;
- Amortecimento da suspensão: 1 000 – 5 000 Ns/m;
- Rigidez do pneu: 150 000 – 250 000 N/m;

2.2. Introdução à Representação de Espaço de Estados

A representação de Espaço de Estados modela a dinâmica do sistema de suspensão a partir do conjunto de equações diferenciais de primeira ordem estabelecidos nas equações 2.1 e 2.2. É útil para:

- Representar sistemas com múltiplas entradas e múltiplas saídas;
- Descreve o comportamento do sistema no domínio tempo em uma forma matricial compacta.

A forma geral é dada pelas equações 2.3 e 2.4:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u \quad (2.3)$$

$$y = \mathbf{C}x + \mathbf{D}u \quad (2.4)$$

Onde:

- x : Vetor de estado (variáveis do sistema que capturam o comportamento dinâmico do sistema);
- u : Vetor de entrada (forças externas);
- y : Vetor de saída (quantidades de interesse);
- \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} : Matrizes que definem as dinâmicas do sistema.

Após derivação das equações de movimento (2.1 e 2.2), reformulando em equações de primeira ordem e combinando na forma vetorial, temos as matrizes do Espaço de Estados da seguinte forma:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_s}{m_s} & -\frac{c_s}{m_s} & \frac{k_s}{m_s} & \frac{c_s}{m_s} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{m_u} & \frac{c_s}{m_u} & -\frac{k_s + k_t}{m_u} & -\frac{c_s}{m_u} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{k_t}{m_u} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

O código e a simulação feitos no MATLAB estão no Anexo A deste Trabalho. Seguindo o código que foi utilizado no MATLAB, tentou-se extrair as informações necessárias para poder fazer a mesma aplicação em Java, como será demonstrado na próxima Seção.

2.3. Aplicação em Java

Inicialmente, como essa aplicação necessita de uma manipulação de matrizes, dessa forma, foi necessário adicionar uma biblioteca para suprir esta necessidade, então a biblioteca “*org.apache.commons.math3.linear*” foi adicionada ao programa.

2.3.1. Cálculo Espaço de Estados

Aqui será mais aprofundado o cálculo da sistema de Espaço de Estados ($\text{sys} = \text{ss}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$; conforme código no MATLAB), já que esta é uma função que não está presente em nenhuma das bibliotecas adicionadas, portanto seu cálculo receberá mais atenção.

As matrizes **A**, **B**, **C** e **D** são definidas de uma forma que fazendo certa operação entre elas seja possível retornar as equações iniciais de movimento (2.1 e 2.2).

Seguindo a Equação 2.3 e a Equação 2.4, na qual **A**, **B**, **C** e **D** são as matrizes apresentadas anteriormente, “*x*” é um vetor de deslocamento, sendo que “*x*[0]” é o deslocamento da massa suspensa, “*x*[1]” é a velocidade da massa suspensa, “*x*[2]” é o deslocamento da massa não suspensa e “*x*[3]” é a velocidade da massa não suspensa e todas elas começaram zeradas já que o sistema parte do repouso, logo as velocidades são zero, e como a oscilação da estrada segue uma função senoidal. Em uma função harmônica, quando o tempo é igual a zero, a função ‘será zero’, portanto o deslocamento também será. Além disso, “*u*” é um vetor relacionado à função senoidal, que dentro da função do Java foi nomeado como deslocamento.

Partindo para o cálculo, a primeira função apresentada calcula a derivada de “*x*”, e não o “*x*” em si, portanto dentro do programa foi feito uma função que apenas calcula esta derivada, para não se ter que preocupar com seu cálculo várias vezes, ela foi nomeada de *Calculardervada(RealMatrix A, RealMatrix B, RealMatrix C, RealMatrix Oscilação)*. Além disso foi adicionado uma função que lida com o tempo, já que precisamos lidar com o deslocamento em função do tempo.

2.3.2. Cálculo do “x”

Mas ainda é necessário fazer uma lógica para calcular o “x”, já que a função “*CalcularDerivada*” recebe como parâmetro o “x” e, por enquanto, possui-se apenas o valor do x inicial, que é zero. Para isso será usado o método de Runge-Kutta de quarta ordem para lidar com matrizes de dimensão quatro, basicamente ele segue a Equação 2.5.

$$x = x + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)dt \quad (2.5)$$

Onde o novo “x” é igual ao antigo “x” acrescido os parâmetros do método de Runge Kutta.

2.3.3. Cálculo do Deslocamento

Por fim, pode-se calcular a segunda equação do Espaço de Estados que gera como resultado o deslocamento, entretanto, como a matriz **D** possui apenas os valores de 0, ela não entrará na equação, pois quando ela for multiplicada com a matriz “u”, gerará uma matriz nula, dessa forma a matriz do deslocamento ser dada apenas pela matriz **C** multiplicada por “x”.

2.3.4. Cálculo da Aceleração da Massa Suspensa

O cálculo da aceleração da massa suspensa foi feito seguindo a Equação 2.6:

$$\ddot{x}_s = \frac{-k_s(x_s - x_u) - c_s(\dot{x}_s - \dot{x}_u)}{m_s} \quad (2.6)$$

Desta forma, foi possível calcular a aceleração para cada instante de tempo sem usar o tempo diretamente na equação, mas usando os deslocamentos e as velocidades das massas suspensa e não suspensa que variam conforme o tempo. Nesta parte, era imprescindível que as velocidades e os deslocamentos fossem calculados corretamente, pois a aceleração seria calculada a cada instante de tempo diferente, e se esses parâmetros fossem calculados de forma errônea, a aceleração levaria esse erro juntamente.

O próximo passo foi calcular a medida de RMS da aceleração, já que essa medida apresenta mais relevância que apenas a medida da média simples. Desta forma, quando o programa chegar no último intervalo de tempo ele pegará a soma de todos os valores das acelerações elevados um a um ao quadrado, dividirá esse valor pelo número de intervalos de tempo e tirará a raiz quadrada desse valor, calculando assim o RMS da aceleração.

3. FUNCIONALIDADES

Este Capítulo apresentará algumas funcionalidades (por consequência, resultados) obtidas através da implementação desta modelagem em Java, inclusive a Interface Gráfica. Para isso serão simulados quatro diferentes cenários com base na realidade física de determinados tipos de veículos. Desta forma será possível verificar os resultados das simulações realizadas em Java, comparando brevemente com os resultados obtidos em MATLAB, a partir de casos mais ‘realistas’ e com base nos valores de pico e RMS (*Root Mean Square*) de cada variável em análise. Os cenários serão:

- Cenário 1: Carro popular.

Parâmetros:

$$m_s = 300 \text{ kg};$$

$$m_u = 40 \text{ kg};$$

$$k_s = 12\,000 \text{ N/m};$$

$$k_t = 200\,000 \text{ N/m};$$

$$c_s = 1\,500 \text{ Ns/m}.$$

- Cenário 2: Carro de luxo.

Parâmetros:

$$m_s = 500 \text{ kg};$$

$$m_u = 50 \text{ kg};$$

$$k_s = 10\,000 \text{ N/m};$$

$$k_t = 180\,000 \text{ N/m};$$

$$c_s = 3\,500 \text{ Ns/m}.$$

- Cenário 3: Carro esportivo.

Parâmetros:

$$m_s = 250 \text{ kg};$$

$$m_u = 30 \text{ kg};$$

$$k_s = 40\,000 \text{ N/m};$$

$$k_t = 250\,000 \text{ N/m};$$

$$c_s = 2\,000 \text{ Ns/m}.$$

- Cenário 4: Veículo Off-Road.

Parâmetros:

$$m_s = 400 \text{ kg};$$

$$m_u = 70 \text{ kg};$$

$$k_s = 15\,000 \text{ N/m};$$

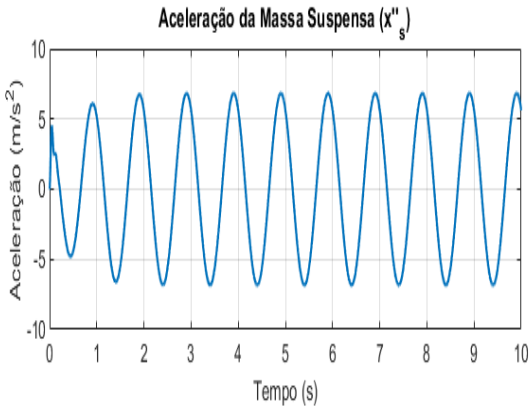
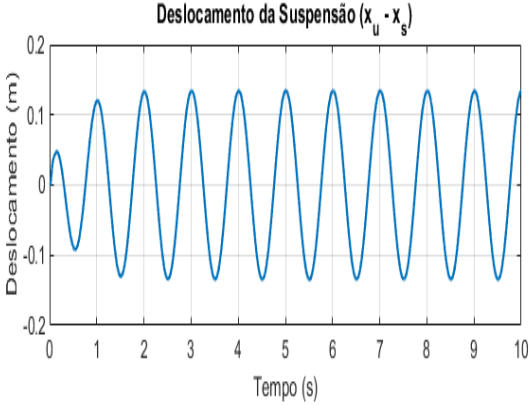
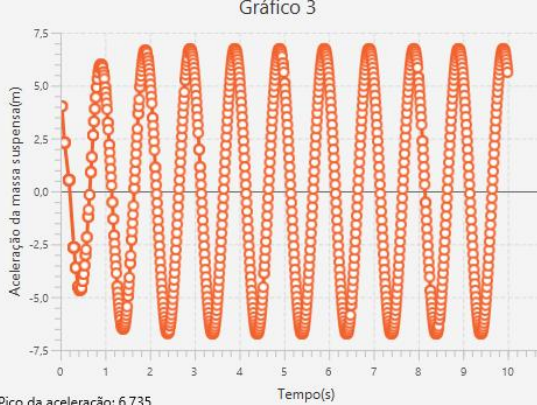
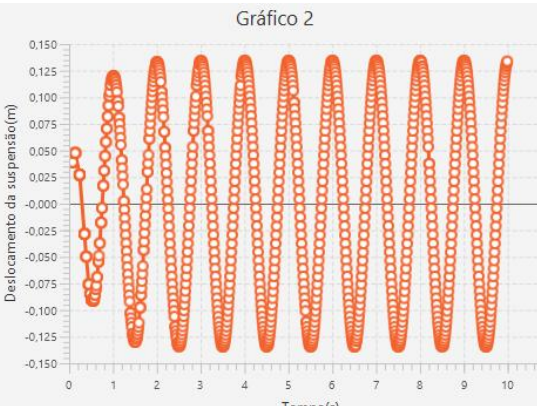
$$k_t = 160\,000 \text{ N/m};$$

$$c_s = 2\,500 \text{ Ns/m}.$$

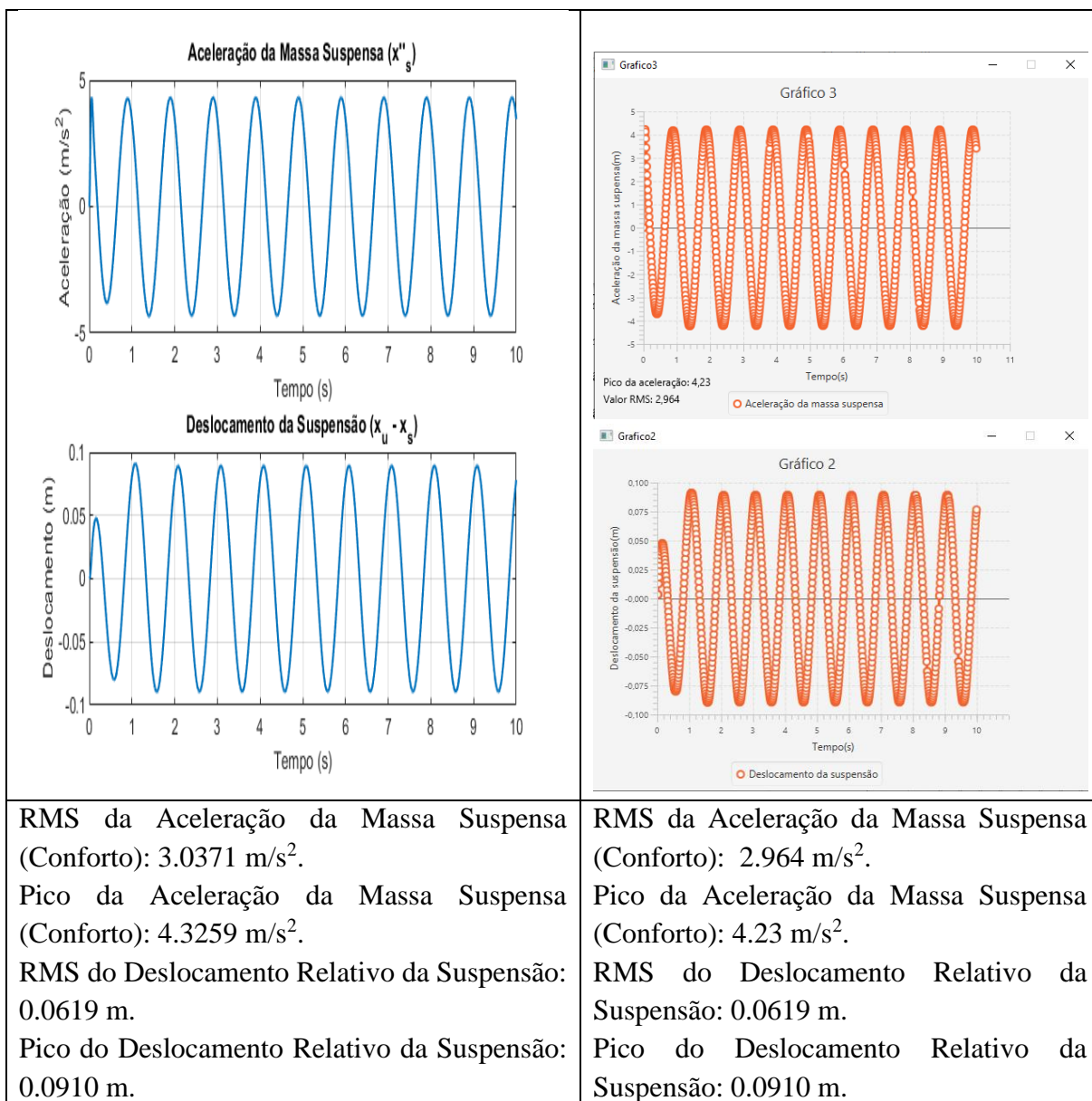
3.1. Aceleração da Massa Suspensa (\ddot{x}_s) e Deslocamento da Suspensão ($x_u - x_s$)

Para o primeiro cenário, os gráficos e resultados numéricos obtidos estão ilustrados na ‘tabela’ abaixo. À esquerda resultados obtidos no MATLAB para fins de comparação, à direita resultados obtidos em Java.

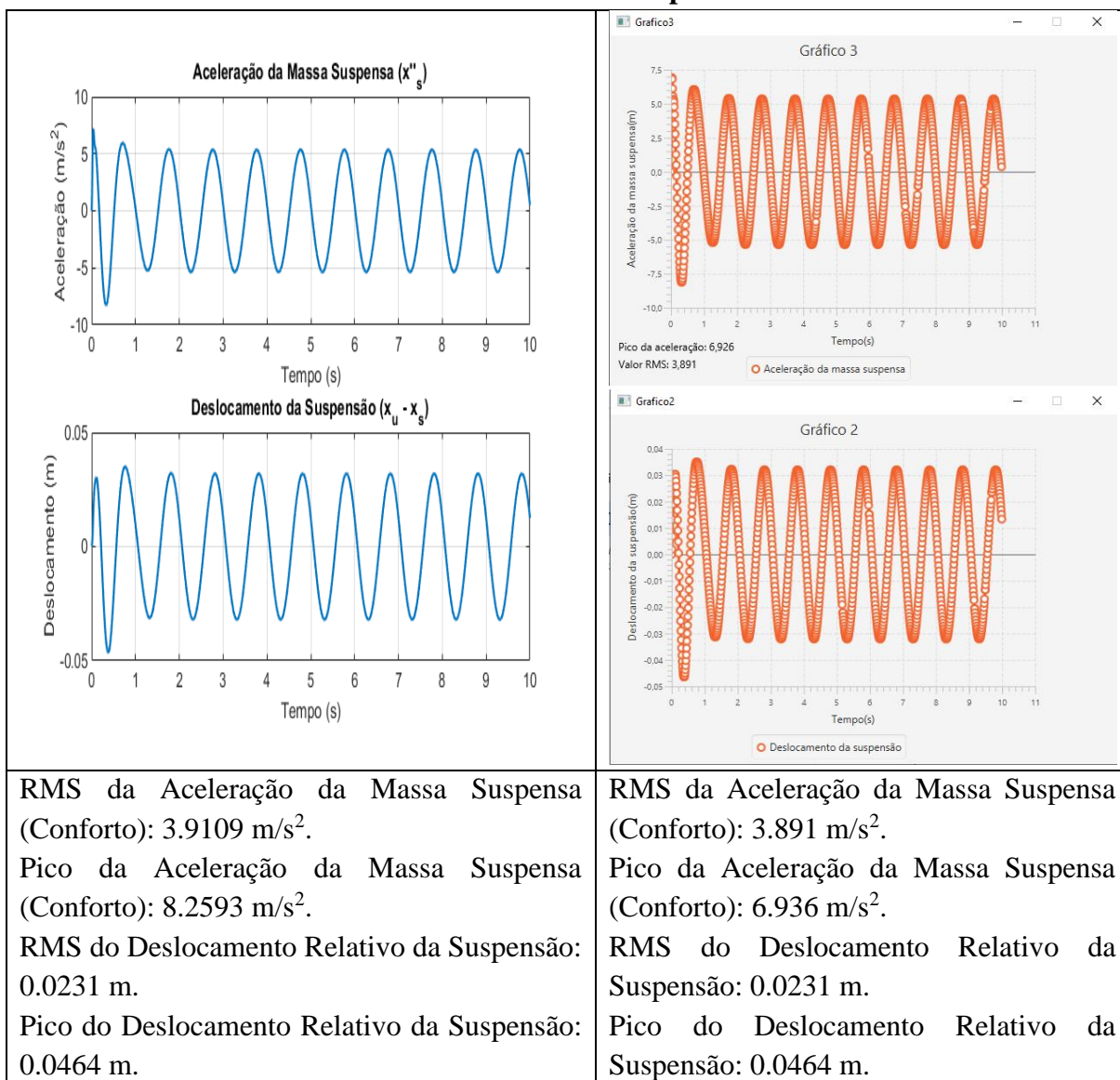
CENÁRIO 1 – Carro popular.

| | |
|---|--|
|   | <p>Gráfico 3</p>  <p>Pico da aceleração: 6,735 Valor RMS: 4,651</p> <p>○ Aceleração da massa suspensa</p> <p>Gráfico 2</p>  <p>○ Deslocamento da suspensão</p> |
| <p>RMS da Aceleração da Massa Suspensa (Conforto): 4.7207 m/s^2. Pico da Aceleração da Massa Suspensa (Conforto): 6.8332 m/s^2. RMS do Deslocamento Relativo da Suspensão: 0.0918 m. Pico do Deslocamento Relativo da Suspensão: 0.1344 m.</p> | <p>RMS da Aceleração da Massa Suspensa (Conforto): 4.651 m/s^2. Pico da Aceleração da Massa Suspensa (Conforto): 6.735 m/s^2. RMS do Deslocamento Relativo da Suspensão: 0.0918 m. Pico do Deslocamento Relativo da Suspensão: 0.1344 m.</p> |

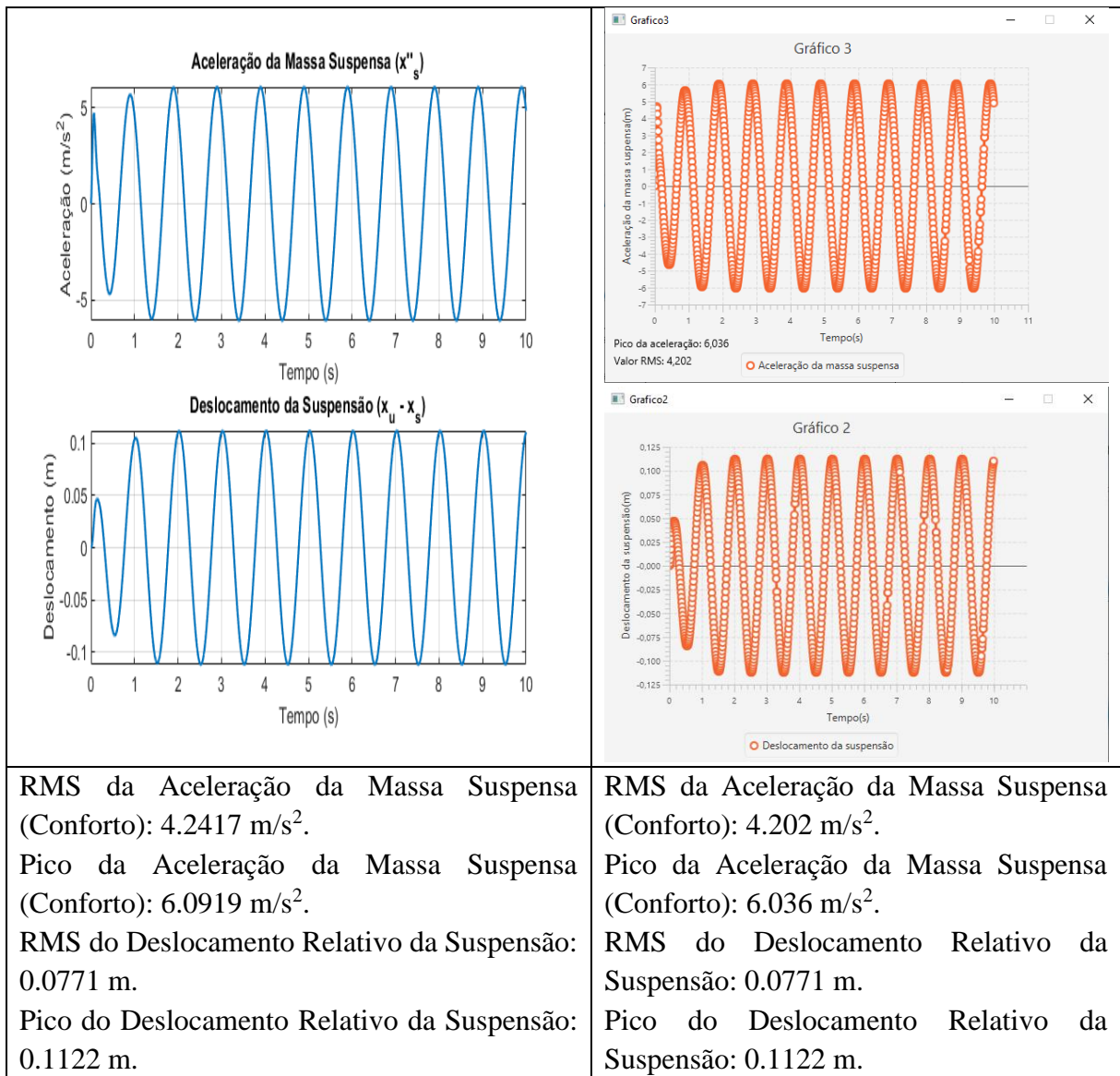
CENÁRIO 2 – Carro de luxo.



CENÁRIO 3 – Carro esportivo.



CENÁRIO 4 – Veículo Off-Road.



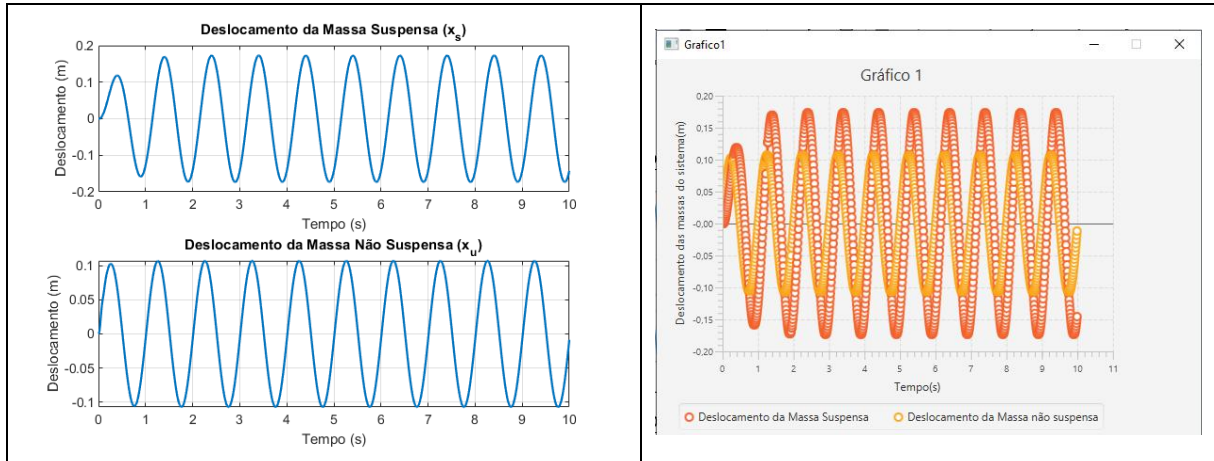
Diante dos cenários expostos acima, verifica-se que os parâmetros essenciais:

- Valores mais altos de m_s tipicamente reduzem a aceleração da massa suspensa, devido a inércia aumentada, melhorando o conforto. Valores mais baixos aumentam a sensibilidade a irregularidades da rua, piorando o conforto;
- Valores inferiores de k_s resultam em suspensões mais macias, melhorando os valores de aceleração da massa suspensa (conforto), mas aumentando o deslocamento da suspensão. Enquanto valores superiores deixam a suspensão mais rígida, o que melhora a dirigibilidade, mas prejudica o conforto devido a altas acelerações na massa suspensa;
- Valores pequenos de c_s levam a oscilações constantes, reduzindo a estabilidade e conforto. Valores muito grandes de amortecimento aumentam a estabilidade, mas reduzem a responsividade da suspensão, potencialmente prejudicando a dirigibilidade.

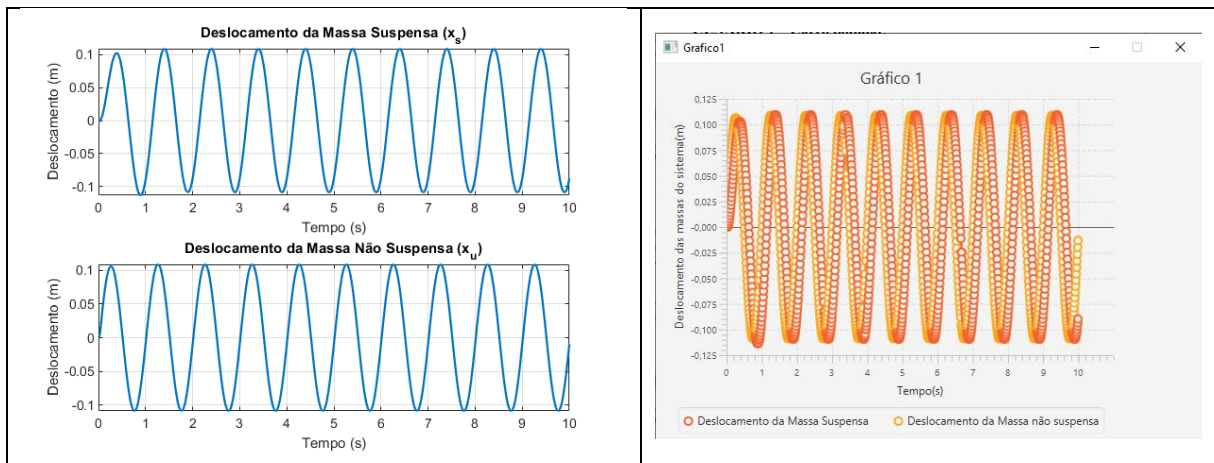
3.2. Deslocamento da Massas (x_u e x_s)

Abaixo os resultados específicos de deslocamento de cada massa (suspensa e não suspensa).

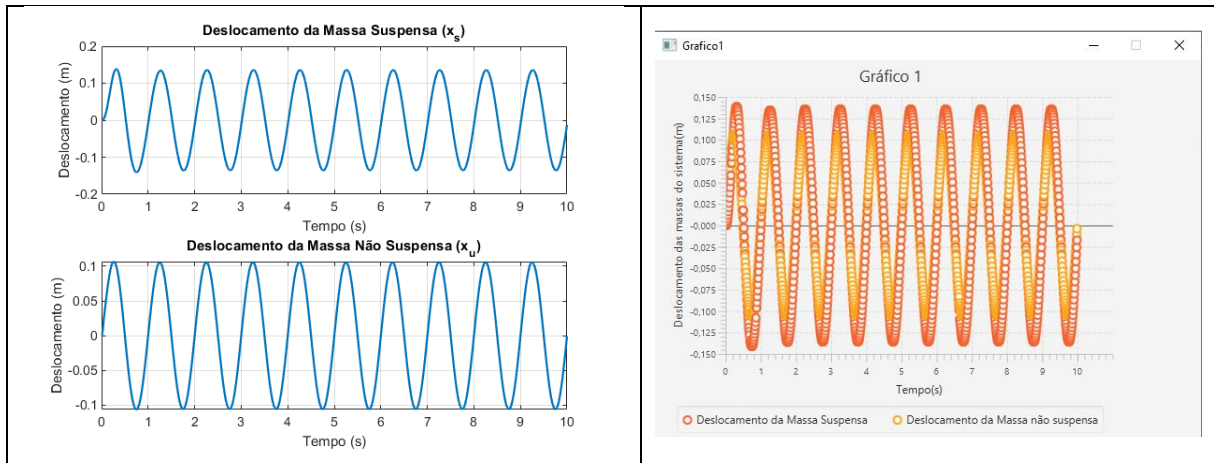
CENÁRIO 1 – Carro popular.



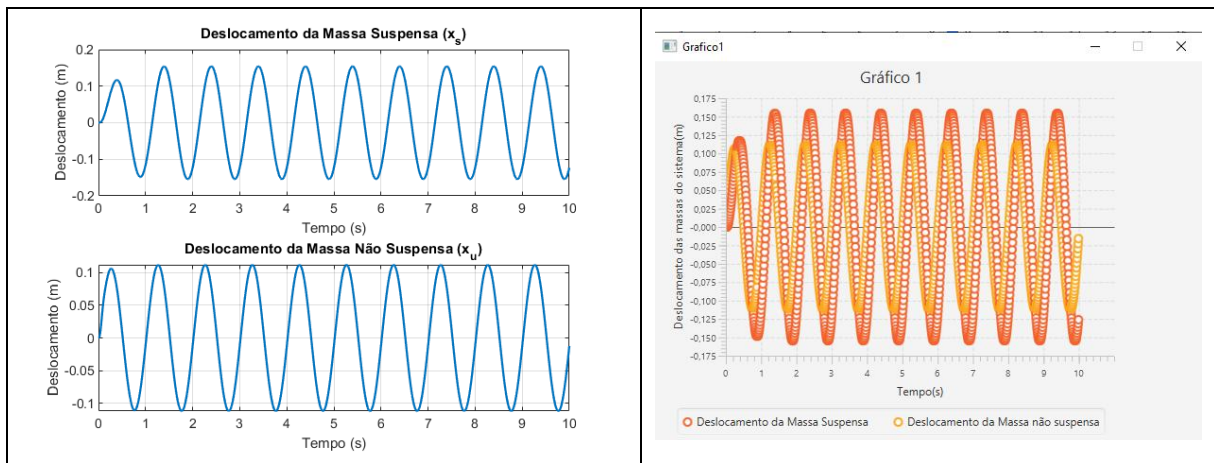
CENÁRIO 2 – Carro de luxo.



CENÁRIO 3 – Carro esportivo.



CENÁRIO 4 – Veículo Off-Road.



3.3. Interface Gráfica

A interface permite entrar com valores para ser calculado ou importar os valores de um arquivo .txt, além de poder salvar os valores digitados em um arquivo .txt. A interface pode ser visualizada na Figura 3.1 abaixo.

Simulador de um quarto de suspensão veicular

[SIMULADOR DE UM QUARTO DE SUSPENSÃO VEICULAR]

Digite o valor da rigidez da suspensão (N/m): Ok

Digite o valor da massa suspensa (kg): Ok

Digite o valor da massa não suspensa (kg): Ok

Digite o coeficiente de amortecimento (Ns/m): Ok

Digite a rigidez do pneu (N/m): Ok

Você deseja salvar esses dados em um arquivo?

Você deseja importar um arquivo com os dados?

Figura 3.1 – Interface [Fonte: Elaborado pelos autores, 2025].

ANEXO A

Abaixo está o código, comentado, sobre como é feita a simulação via MATLAB. Os parâmetros utilizados, no código abaixo, são arbitrários. Esse código possui funções específicas do MATLAB que realizam o cálculo das equações do Espaço de Estados (*lsim* e *ss*).

```
clc; clearvars; close all;
```

```
%% IMPLEMENTAÇÃO AUTOMÁTICA
```

```
% Parâmetros
```

```
m_s = 300;      % Massa suspensa (kg) - 250 a 500 kg
m_u = 40;       % Massa não suspensa (kg) - 25 a 75 kg
k_s = 12000;    % Rigidez da suspensão (N/m) - 10 000 a 50 000 N/m
k_t = 200000;   % Rigidez do pneu (N/m) - 150 000 a 250 000 N/m
c_s = 1500;     % Amortecimento da suspensão (Ns/m) - 1 000 a 5 000 Ns/m
```

```
% Matrizes do Espaço de Estados
```

```
A = [0, 1, 0, 0;
     -k_s/m_s, -c_s/m_s, k_s/m_s, c_s/m_s;
     0, 0, 0, 1;
     k_s/m_u, c_s/m_u, -(k_s+k_t)/m_u, -c_s/m_u];
B = [0; 0; 0; k_t/m_u];
C = [1, 0, 0, 0; 0, 0, 1, 0];
D = [0; 0];
```

```
% Função do MATLAB (sys) para definir o Sistema das matrizes do Espaço de Estados
```

```
sys = ss(A, B, C, D);
```

```
% Parâmetros para Simulação
```

```
A = 0.1;        % Amplitude do solavanco, valor arbitrário
t = 0:0.01:10;  % Período/Tempo de 0 a 10 segundos com degrau (step) de 0.01 s
```

```
% Simulação
```

```
u = A * sin(2 * pi * t);      % Excitação da rua de 0.1 m (altura do solavanco)
[y, t, x] = lsim(sys, u, t);  % lsim é uma função linear do MATLAB para simular
                                sistemas no domínio tempo
```

```
% --- Cálculo da Aceleração da Massa Suspensa ---
```

```
x_s = x(:,1);      % Deslocamento da massa suspensa
dx_s = x(:,2);     % Velocidade da massa suspensa
x_u = x(:,3);      % Deslocamento da massa não suspensa
dx_u = x(:,4);     % Velocidade da massa não suspensa
```

```
acel_suspensa = (-k_s * (x_s - x_u) - c_s * (dx_s - dx_u)) / m_s; % Aceleração da
massa suspensa
```

```
% --- Cálculo do Deslocamento da Suspensão ---
```

```
desl_susp = x_u - x_s; % Deslocamento relativo (x_u - x_s)
```



```

% --- Plots ---
% Deslocamentos
figure;
subplot(2,1,1);
plot(t, y(:,1), 'LineWidth', 1.5);
title('Deslocamento da Massa Suspensa (x_s)');
xlabel('Tempo (s)');
ylabel('Deslocamento (m)');
grid on;

subplot(2,1,2);
plot(t, y(:,2), 'LineWidth', 1.5);
title('Deslocamento da Massa Não Suspensa (x_u)');
xlabel('Tempo (s)');
ylabel('Deslocamento (m)');
grid on;

% Aceleração e Deslocamento da Suspensão
figure;
subplot(2,1,1);
plot(t, acel_suspensa, 'LineWidth', 1.5);
title('Aceleração da Massa Suspensa (x''''_s)');
xlabel('Tempo (s)');
ylabel('Aceleração (m/s^2)');
grid on;

subplot(2,1,2);
plot(t, desl_susp, 'LineWidth', 1.5);
title('Deslocamento da Suspensão (x_u - x_s)');
xlabel('Tempo (s)');
ylabel('Deslocamento (m)');
grid on;

% Estatísticas de conforto (RMS da aceleração da massa suspensa)
rms_acel_suspensa = rms(acel_suspensa);
fprintf('RMS da Aceleração da Massa Suspensa (Conforto): %.4f m/s^2\n',
rms_acel_suspensa);
max_acel_suspensa = max(abs(acel_suspensa));
fprintf('Pico da Aceleração da Massa Suspensa (Conforto): %.4f m/s^2\n',
max_acel_suspensa);
% Pico do Deslocamento Relativo da Suspensão
rms_desl_susp = rms(desl_susp);
fprintf('RMS do Deslocamento Relativo da Suspensão: %.4f m\n', rms_desl_susp);
max_desl_susp = max(abs(desl_susp));
fprintf('Pico do Deslocamento Relativo da Suspensão: %.4f m\n', max_desl_susp);

% figure;
% plot(t, u, 'LineWidth', 1.5);
% title('Onda Senoidal (x_r)');
% xlabel('Tempo (s)');
% ylabel('Amplitude (m2)');
% grid on;

```