**2. A MODELAGEM**

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 (), podemos equacionar o sistema conforme a Equação 2.1 e a Equação 2.2:

(2.1)

(2.2)

**2.1.1. Variáveis e Parâmetros Necessários**

*ms*​​: Massa suspensa (carroceria) – [kg];

*mu*​: Massa não suspensa (pneu e eixo) – [kg];

*ks*: Rigidez da suspensão, influencia no conforto veicular e na habilidade de absorção das irregularidades da rua – [N/m];

*cs*: Amortecimento da suspensão, reduz as vibrações (conforto) e garante o contato do pneu com o solo (controle) – [Ns/m];

*kt*​: Rigidez do pneu, afeta a transmissão de vibração da rua – [N/m];

*xr*: Deslocamento imposto pelo solo, que pode ser definido como:

* Perfil harmônico (senoidal): ou .

Onde a amplitude (*A*) representa a altura dos solavancos em metros e a frequência angular (*f*) afeta a resposta dinâmica. A velocidade (*r*) impacta na força transmitida ao conjunto de suspensão.

**2.1.2. Métricas Analíticas**

Para avaliar a dinâmica veicular, focaremos em:

* Conforto de veicular: Medido através da aceleração da massa suspensa (*s*). Acelerações baixas melhoram o conforto veicular. Faixa ideal: 0,3 – 0,5 𝑚/𝑠 (valor RMS) para veículos de passeio em estradas normais;
* Deslocamento da suspensão: Medido pelo deslocamento relativo entre a massa suspensa e não suspensa . O curso excessivo leva a restrições mecânicas. Curso Máximo da Suspensão:
  + Veículos de passageiros: 50 – 100 mm;
  + Carros esportivos: 30 – 50 mm (suspensão mais rígida, menos curso);
  + Veículos off-road: 200 – 300 mm (curso maior para terrenos acidentados).

**2.1.3. 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;
* Amortecimento do pneu: Negligenciado.

**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:

(2.3)

(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);
* **A**, **B**, **C**, **D**: Matrizes que definem as dinâmicas do sistema.

Após derivaçãodas 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:

, , , 

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 (*sys = ss(A, B, C, 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 *Calcularderivada(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, o código inicial pode ser analisado no Anexo B deste Trabalho.

**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.

 (2.5)

Onde o novo “*x*” é igual ao antigo “*x*” acrescido os parâmetros do método de Runge Kutta, esta parte do programa consta no Anexo C deste Trabalho.

**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*”. Esta parte do código consta no Anexo D deste Trabalho.