Lagrange

June 12, 2019

```
In [17]: import sympy as sp
         import numpy as np
         from IPython.display import display

         #display(yourobject)
```

1 Dedução das equações dinâmicas por Lagrange

1.1 Introdução

```
In [2]: sp.init_printing()
```

```
In [3]: t = sp.Symbol("t", positive=True)

        # Propriedades da cabeça
        m_c, J_c = sp.symbols("m_c J_c", positive=True)
        x_c = sp.Symbol("x_c", real=True)(t)
        y_c = sp.Symbol("y_c", real=True)(t)
        theta_c = sp.Symbol("theta_c", real=True)(t)

        # Propriedades do torço
        m_t, J_t, L_t = sp.symbols("m_t J_t L_t", positive=True)
        x_t = sp.Symbol("x_t", real=True)(t)
        y_t = sp.Symbol("y_t", real=True)(t)
        theta_t = sp.Symbol("theta_t", real=True)(t)

        # Propriedades dos membros inferiores
        x_i = sp.Symbol("x_i", real=True)(t)
        m_i = sp.Symbol("m_i", positive=True)

        # Interação air bag-cabeça
        k_ab, c_ab = sp.symbols("k_ab c_ab", positive=True)

        # Propriedades do pescoço
        ## Deformação do pescoço
        x_p = sp.Symbol("x_p", positive=True)(t)
        ## Mola linear
```

```python
k_p, c_p = sp.symbols("k_p c_p", positive=True)
## Mola de torção
k_rp, c_rp = sp.symbols("k_rp c_rp", positive=True)

# Propriedades do cinto de segurança
k_s, c_s = sp.symbols("k_s c_s", positive=True)

# Propriedades do braço
k_b, c_b = sp.symbols("k_b c_b", positive=True)

# Interação air_bag-torso
k_ab, c_ab = sp.symbols("k_ab c_ab", positive=True)

# Ligamento torço-membros inferiores
k_ri, c_ri = sp.symbols("k_ri c_ri", positive=True)

# Interações de amortecimento dos membros inferiores
k_i, c_i = sp.symbols("k_i c_i", positive=True)

# Desaceleração do carro
a = sp.Symbol("a", real=True)

g = sp.Symbol("g", positive=True)
```

Parâmetro	Unidade	Explicação
m_c	kg	Massa da cabeça
J_c	kg*mš	Momento de inércia da cabeça
x_c	m	Posição do centro de massa da cabeça
y_c	m	Posição do centro de massa da cabeça
θ_c	rad	Inclinação da cabeça
m_t	kg	Massa do torso
J_t	kg*mš	Momento de inércia do torso
L_t	m	Distância do centro de massa do torso até o banco
x_t	m	Posição do centro de massa do torso
y_t	m	Posição do centro de massa do torso
θ_t	rad	Inclinação do torso
x_i	m	Posição do centro de massa dos membros inferiores
m_i	kg	Massa dos membros inferiores
x_p	m	Deformação do pescoço
k_p	N/m	Propriedade elástica linear do pescoço
c_p	N*s/m	Propriedade viscosa linear do pescoço
k_{rp}	N*m	Constante da mola de torção do pescoço
c_{rp}	N*s*m	Constante do amortecedor de torção do pescoço
k_s	N/m	Propriedades do cinto de segurança
c_s	N*s/m	Propriedades do cinto de segurança
k_b	N/m	Propriedades do braço
c_b	N*s/m	Propriedades do braço

Parâmetro	Unidade	Explicação
k_{ab}	N/m	Interação do air bag
c_{ab}	N*s/m	Interação do air bag
k_{ri}	N*m	Ligamento dos membros inferiores com o torso
c_{ri}	N*s*m	Ligamento dos membros inferiores com o torso
k_i	N/m	Interações de amortecimento dos membros inferiores
c_i	N*s/m	Interações de amortecimento dos membros inferiores

Muito bem, senhoras e senhores, iremos agora deduzir as equações dinâmicas para o nosso modelo de *crash test* utilizando o método de Lagrange. Garanto que será uma experiência intensa e engrandecedora. Não sei dizer se *de fato* engrandecedora, mas certamente será intensa. No mais, confie na matemática e lá vamos nós

1.2 Dedução de T (energia cinética)

$$T = \sum_{i \subset C} \frac{m_i * v_{gi}^2}{2} + \frac{J_i * \omega_i^2}{2}$$

Onde:

Parâmetro	Significado
C	conjunto de corpos de sistema
v_{gi}	velocidade do centro de massa do corpo i
m_i	massa do corpo i
J_i	momento de inércia do corpo i
ω_i	velocidade angular do corpo i

```
In [5]: class Corpo():
            def __init__(self, massa, mom_inercia, x, y, theta):
                self.massa        = massa
                self.mom_inercia  = mom_inercia

                self.x = x
                self.y = y
                self.theta = theta

                try:
                    self.y_ponto = y.diff()
                except:
                    self.y_ponto = 0

                try:
                    self.x_ponto = x.diff()
                except:
                    self.x_ponto = 0

                try:
```

```
            self.theta_ponto = theta.diff()
        except:
            self.theta_ponto = 0

        self.quad_vel_lin = self.x_ponto**2+self.y_ponto**2
        self.quad_vel_ang = self.theta_ponto**2

    def cinetica(self):
        return (self.massa*self.quad_vel_lin)/2 \
                + (self.mom_inercia*self.quad_vel_ang)/2

    def potencial(self, a_x, a_y):
        return self.massa*(a_x*self.x + a_y*self.y)

    def show(self):
        return (self.massa, self.quad_vel_lin,
                self.mom_inercia, self.quad_vel_ang)

Corpos = [Corpo(m_c, J_c, x_c, y_c, theta_c),
          Corpo(m_t, J_t, x_t, y_t, theta_t),
          Corpo(m_i, 0, x_i, 0, 0)]
```

```
In [6]: for corpo in Corpos:
            display(corpo.show())
```

$$\left(m_c, \quad \left(\frac{d}{dt}\mathrm{x_c}(t)\right)^2 + \left(\frac{d}{dt}\mathrm{y_c}(t)\right)^2, \quad J_c, \quad \left(\frac{d}{dt}\theta_c(t)\right)^2\right)$$

$$\left(m_t, \quad \left(\frac{d}{dt}\mathrm{x_t}(t)\right)^2 + \left(\frac{d}{dt}\mathrm{y_t}(t)\right)^2, \quad J_t, \quad \left(\frac{d}{dt}\theta_t(t)\right)^2\right)$$

$$\left(m_i, \quad \left(\frac{d}{dt}\mathrm{x_i}(t)\right)^2, \quad 0, \quad 0\right)$$

```
In [7]: display(Corpos[0].cinetica())
```

$$\frac{J_c}{2}\left(\frac{d}{dt}\theta_c(t)\right)^2 + \frac{m_c}{2}\left(\left(\frac{d}{dt}\mathrm{x_c}(t)\right)^2 + \left(\frac{d}{dt}\mathrm{y_c}(t)\right)^2\right)$$

```
In [8]: T = 0
        for corpo in Corpos:
            T += corpo.cinetica()

        display(T)
```

$$\frac{J_c}{2}\left(\frac{d}{dt}\theta_c(t)\right)^2 + \frac{J_t}{2}\left(\frac{d}{dt}\theta_t(t)\right)^2 + \frac{m_c}{2}\left(\left(\frac{d}{dt}\mathrm{x_c}(t)\right)^2 + \left(\frac{d}{dt}\mathrm{y_c}(t)\right)^2\right) + \frac{m_i}{2}\left(\frac{d}{dt}\mathrm{x_i}(t)\right)^2 + \frac{m_t}{2}\left(\left(\frac{d}{dt}\mathrm{x_t}(t)\right)^2 + \left(\frac{d}{dt}\right.\right.$$

1.3 Dedução de V (energia potencial)

$$V = \sum_{i \subset M} \frac{k_i * q_i^2}{2} + \sum_{i \subset C} m_i(x_i * a + y_i * g)$$

Onde:

Parâmetro	Significado
M	conjunto de molas do sistema
C	conjunto de corpos de sistema
k_i	constante elástica da mola i
q_i	coordenada de distenção da mola i
m_i	massa do corpo i
x_i	coordenadas do centro de massa do corpo i
y_i	coordenadas do centro de massa do corpo i
g	aceleração da gravidade
a	aceleração do carro

In [10]:
```
class Mola:
    def __init__(self, k, c, q):
        self.k = k
        self.c = c
        self.q = q

    def potencial(self):
        return (self.k*self.q**2)/2

    def dissipador(self):
        return (self.c*self.q.diff()**2)/2

Molas = [Mola(k_p, c_p, x_p),
         Mola(k_rp, c_rp, theta_c-theta_t),
         Mola(k_s + k_ab + k_b, c_s + c_ab + c_b, x_t),
         Mola(k_i, c_i, x_i),
         Mola(k_ri, c_ri, theta_t)]
```

In [28]:
```
V = 0

for mola in Molas:
    V += mola.potencial()

for corpo in Corpos:
    V += corpo.potencial(a, g)

display(V)
```

$$am_i \mathrm{x_i}(t) + \frac{k_i}{2}\mathrm{x_i}^2(t) + \frac{k_p}{2}\mathrm{x_p}^2(t) + \frac{k_{ri}}{2}\theta_t^2(t) + \frac{k_{rp}}{2}(\theta_c(t) - \theta_t(t))^2 + m_c(a\,\mathrm{x_c}(t) + g\,\mathrm{y_c}(t)) + m_t(a\,\mathrm{x_t}(t) + g\,\mathrm{y_t}(t)) +$$

1.4 Dedução de D (dissipação de Rayleigh)

$$D = \sum_{i \subset A} \frac{c_i * q_i^2}{2}$$

Onde:

Parâmetro	Significado
A	conjunto de amortecedores do sistema
c_i	constante viscosa do amortecedor i
q_i	coordenada de distenção do amortecedor i

In [16]:
```
D = 0
for mola in Molas:
    D += mola.dissipador()

display(D)
```

$$\frac{c_i}{2}\left(\frac{d}{dt}\mathrm{x_i}(t)\right)^2 + \frac{c_p}{2}\left(\frac{d}{dt}\mathrm{x_p}(t)\right)^2 + \frac{c_{ri}}{2}\left(\frac{d}{dt}\theta_t(t)\right)^2 + \frac{c_{rp}}{2}\left(\frac{d}{dt}\theta_c(t) - \frac{d}{dt}\theta_t(t)\right)^2 + \frac{1}{2}\left(c_{ab} + c_b + c_s\right)\left(\frac{d}{dt}\mathrm{x_t}(t)\right)^2$$

Estamos tratando de um sistem de alta complexidade, portanto é essencial que nos atentemos aos conceitos envolvidos e a progressão lógica envolvida, uma vez que a complexidade das expressões finais tornam impossível sua revisão direta

Muito bem, dito isso, peço que analizem com carinho a lógica de cada etapa do processo, se preocupando em linhas gerais com o resultado final. No mais, essas são as funções equações dinâmicas que descrevem o nosso sistema. Antes de passarmos às simulações numéricas, podemos aplicar algumas restrições geométricas

1.5 Simplificações geométricas

In [26]:
```
sp.Eq(x_t, L_t*sp.sin(theta_t) + x_i)
```

Out[26]:

$$\mathrm{x_t}(t) = L_t \sin(\theta_t(t)) + \mathrm{x_i}(t)$$

In [23]:
```
sp.Eq(y_t, L_t*sp.cos(theta_t))
```

Out[23]:

$$\mathrm{y_t}(t) = L_t \cos(\theta_t(t))$$

In [24]:
```
sp.Eq(sp.tan(theta_c), x_c/y_c)
```

Out[24]:

$$\tan(\theta_c(t)) = \frac{\mathrm{x_c}(t)}{\mathrm{y_c}(t)}$$