Dinâmica de Sistemas Mecânicos em Python - SEPEX 2020 -

Professor Paulo Victor

CEFET/RJ - Campus Angra dos Reis
paulo.gomes@cefet-rj.br

Referências Bibliográficas:
Site do SymPy Mechanics → https://docs.sympy.org/latest/modules/physics/mechanics/index.html
Site do PyDy→ www.pydy.org
Site oficial do Jupyter Notebook → https://jupyter.org/
Repositório com o material do curso → https://github.com/Pi-vi/SEPEX2020_Dinamica

---

Agenda:

- Instalação
- Jupyter Notebook
- Introdução
- Operações Vetoriais
- Cinemática
- Equações de Movimento - Método de Newton-Euler
- Equações de Movimento - Método de Lagrange
- Simulações
- Livros Recomendados

Instalação

Instalação de pacotes no Ubuntu:

Instalação de bibliotecas e do Jupyter Notebook:

```
sudo apt install python3-pip
pip3 install numpy
pip3 install matplotlib
pip3 install sympy
pip3 install pydy
pip3 install scipy
pip3 install notebook
pip3 install jupyterlab
sudo apt install jupyter-core
```

habilitar gráficos iterativos:

```
pip3 install ipywidgets
jupyter nbextension enable --py widgetsnbextension
```

Para executar:

```
jupyter notebook
```

ou:

```
jupyter lab
```

Instalação do Python no Windows:

A maneira mais fácil de instalar o Python no Windows e vários outros pacotes é através do Anaconda.
O Anaconda pode ser baixado em: https://www.anaconda.com/products/individual

https://youtu.be/gAdZ1BABhrY

https://youtu.be/gAdZ1BABhrY

O Anaconda já vem com uma série de pacotes úteis para a engenharia.
Caso algum pacote que se desejar utilizar não venha junto com o Anaconda, esse pacote pode ser instalado de forma separada através da seguinte maneira:
→ Abra o prompt de comando e digite o comando:

```
conda install nome_do_pacote
```

Jupyter Notebook

O Jupyter é uma IDE (interactive development enviroment) baseado na web, ou seja, ela roda a partir do seu navegador.
Site oficial do Jupyter Notebook → https://jupyter.org/
Após ter seguido os passos da seção anterior, basta abrir o Terminal, ou o Prompt de Comando e digitar:

```
jupyter notebook
```

Introdução

Introdução à Dinâmica

- A dinâmica, de modo geral, é responsável por estudar sistemas que evoluem no tempo. Ou seja, sistemas que sofrem variações a medida que o tempo passa.
- Observa-se que na natureza tudo está em movimento, fato este que evidencia a importância do estudo da dinâmica.
- Para a Engenharia Mecânica, que basicamente está interessada em projetar todo tipo de aparato, o estudo da dinâmica é fundamental. Através dele será possível, entre outras coisas, prever o comportamento dos sistemas a serem construídos, determinar forças necessárias para gerar o movimento desejado e, a partir das equações de movimento, estabelecer ações de controle.

A dinâmica se divide basicamente em duas áreas: a cinemática e a cinética. A primeira está preocupada em estudar o movimento, sem se preocupar com a causa do movimento. Já a cinética estuda o que causa o movimento.

Entre as aplicações da dinâmica, destacam-se:

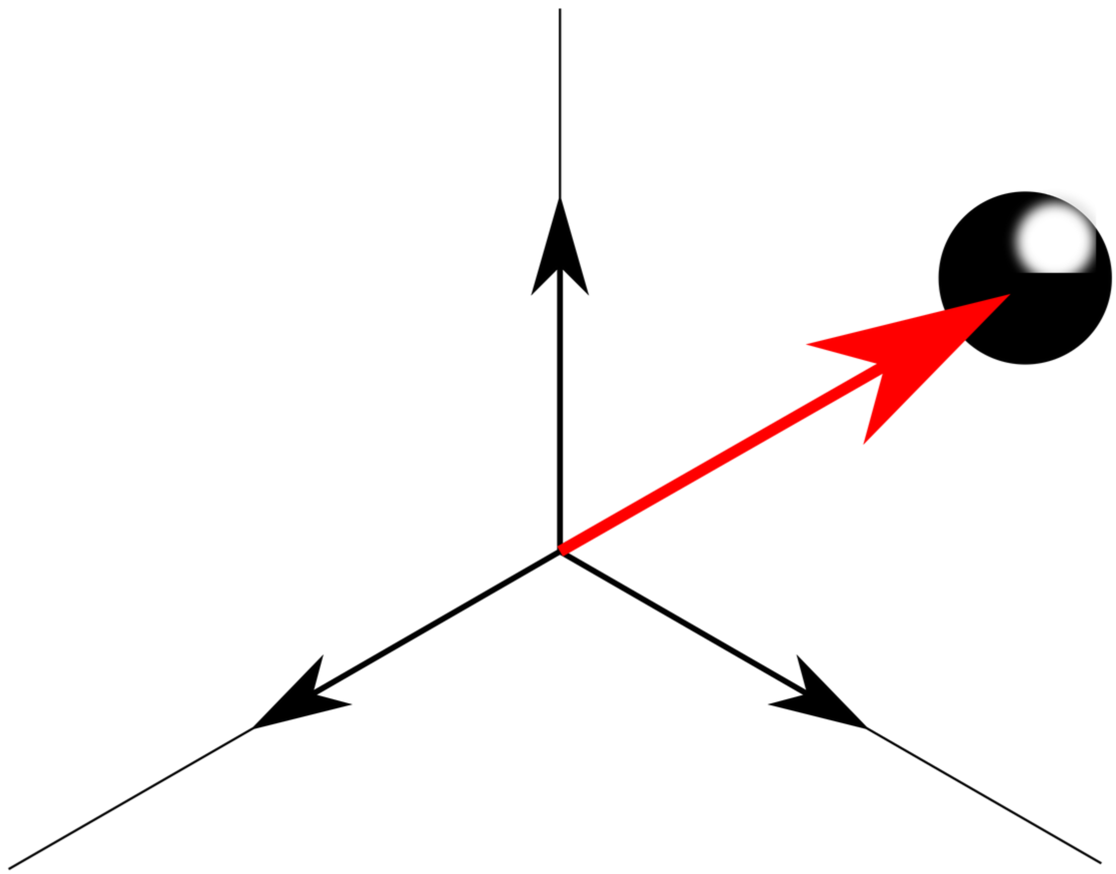
- Dinâmica de Mecanismos e Máquinas
- Dinâmica de Rotores
- Dinâmica de Robôs
- Astrodinâmica
- Dinâmica de Veículos
- Biomecânica.

Modelagem de Sistemas Mecânicos

- A modelagem mecânica é uma representação matemática da realidade.
- Para se modelar um sistema mecânico, deve-se ter, primeiramente, uma visão muito precisa e clara dos objetivos do seu modelo.
- Boas medições são tão raras quanto bons modelos.
- **Modelo é aquilo que não é, mas tudo se passa como se fosse.**

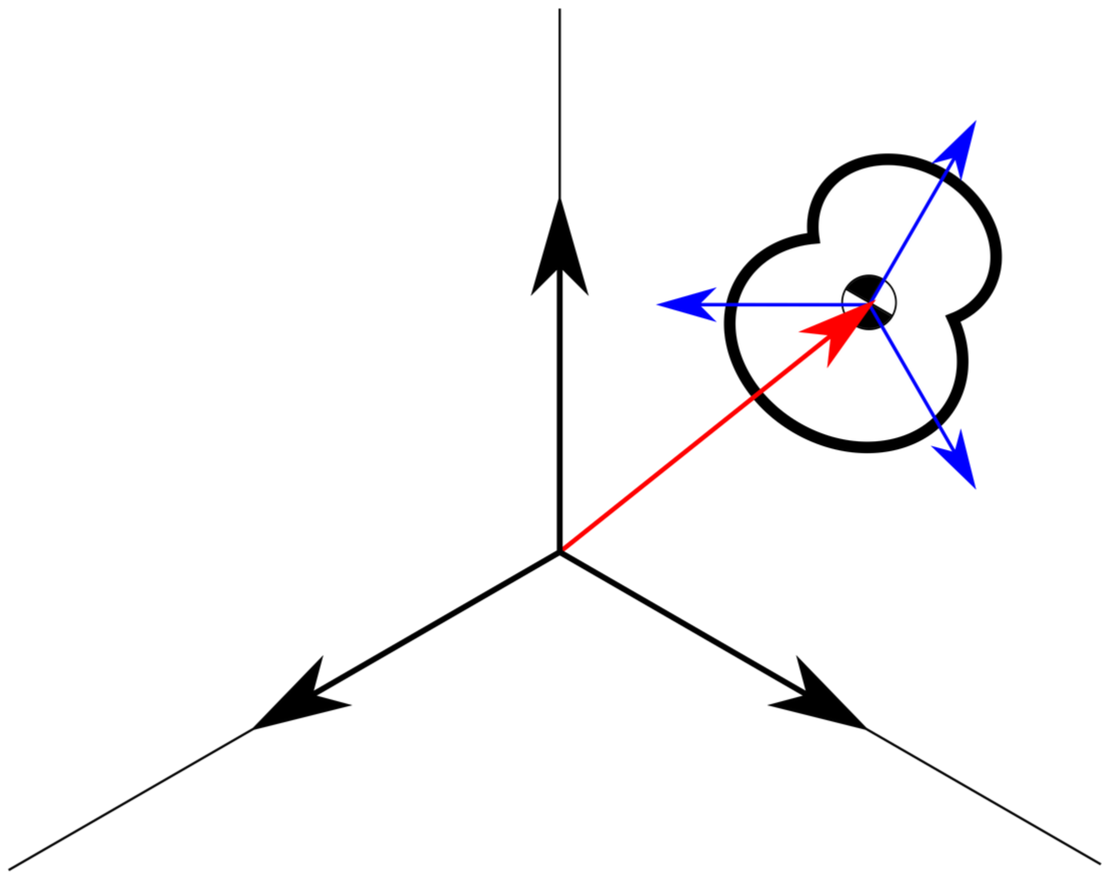
Por hora, podemos considerar dois tipos básicos de elementos mecânicos para modelagem:

Partícula



Partícula

Corpo rígido



Corpo Rígido

Computação Numérica x Computação Simbólica

Computação numérica - exemplos → Matlab*, Octave, Scilab, Fortran, C
Computação simbólica - exemplos → Wolfram Mathematica, Maple

Operações Vetoriais

Importar bibliotecas pydy para o python:

```
import sympy.physics.mechanics as me
```

Criando uma Sistema Referencial

Para se definir a posição de uma partícula no espaço, um sistema de referência torna-se necessário. Para esse propósito, três linhas ortogonais que se interceptam em um ponto comum, chamado origem, são necessárias.
A posição da partícula pode ser definida em termos da distância ao longo dessas linhas.
Podemos definir vetores unitários alinhados a essas retas.

```
N = me.ReferenceFrame('N')
```

Vetores unitários

$\hat{n}_x$
$\hat{n}_y$
$\hat{n}_z$

```
N.x

N.y

N.z
```

Composição/decomposição de um vetor

Podemos compor/decompor um vetor de infinitas maneiras. Uma maneira muito útil é a composição/decomposição através de uma base ortonormal.
Composição:
$\mathbf{a} = 3\hat{n}_x + 2\hat{n}_y + 4\hat{n}_z$
$\mathbf{b} = 5\hat{n}_x + 7\hat{n}_y - 3\hat{n}_z$
Decomposição:

```
ax_N = a.express(N).args[0][0][0] # componente de a projetada em N.x

ay_N = a.express(N).args[0][0][1] # componente de a projetada em N.y

az_N = a.express(N).args[0][0][2] # componente de a projetada em N.z
```

Para escrever o vetor da seguinte forma, use o comando:

$$^{N}\mathbf{a} = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$$

```
a.express(N).args[0][0]
```

Adição

$\mathbf{c} = \mathbf{a} + \mathbf{b}$

```
c = a + b
```

Produto escalar

$\mathbf{a} \cdot \mathbf{b}$

```
me.dot(a,b)
```

Ou:

```
a.dot(b)
```

Produto vetorial

$\mathbf{a} \times \mathbf{b}$

```
me.cross(a,b)
```

Produtos múltiplos

Produto misto:

$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$

```
me.dot(a,me.cross(b,c))
```

Duplo produto vetorial:

$\mathbf{a} \times (\mathbf{b} \times \mathbf{c})$

```
me.cross(a,me.cross(b,c))
```

Diferenciação de vetores

$$\frac{{}^{N}\partial \mathbf{r}}{\partial x}$$

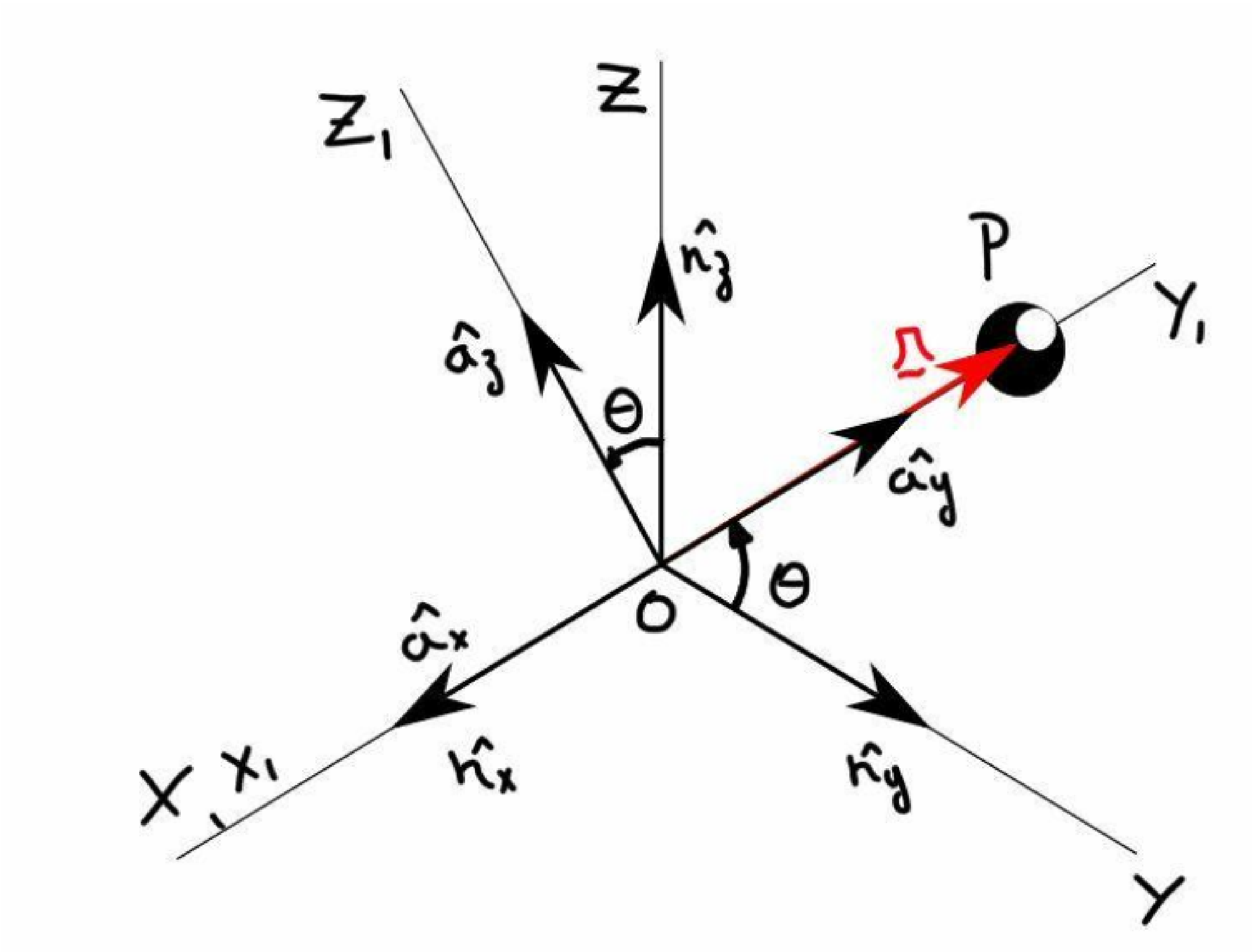
```
r.diff(x,N)
```

Cinemática

A cinemática desempenha um papel fundamental no estudo da dinâmica.
→ Ela descreve o movimento de sistemas mecânicos

→ Para se obterem as equações de movimento de um sistema qualquer, será sempre necessário expressar, antes, as velocidades e acelerações de seus elementos como funções das coordenadas que descrevem a configuração do sistema.

Para facilitar a análise cinemática, podemos criar referenciais que se movem.



Rotação no eixo X

$\mathbf{p}^{P/O} = r\hat{\mathbf{a}}_y$

$^{N}\dfrac{d\theta\hat{\mathbf{n}}_x}{dt} = \dot{\theta}\hat{\mathbf{n}}_x = ^{N}\ \omega^{A}$ ← vetor velocidade angular

$$\frac{^{N}d\mathbf{p}^{\frac{P}{O}}}{dt} = \dot{r}\hat{\mathbf{a}}_y + r^{N}\omega^{A} \times \hat{\mathbf{a}}_y$$

Criando referencial móvel e o orientando em relação a outro referencial:

```
A = me.ReferenceFrame('A')
A.orient(N,'Axis',[theta,N.x])
```

Ou

```
A = N.orientnew('A', 'Axis',(theta,N.x))
```

Criando pontos

Criando um ponto estacionário:

```
O = me.Point('O')
O.set_vel(N,0) # define que a velocidade de O em relação a N é zero
```

Criando uma constante simbólica:

```
import sympy as sy
r = sy.symbols('r')
```

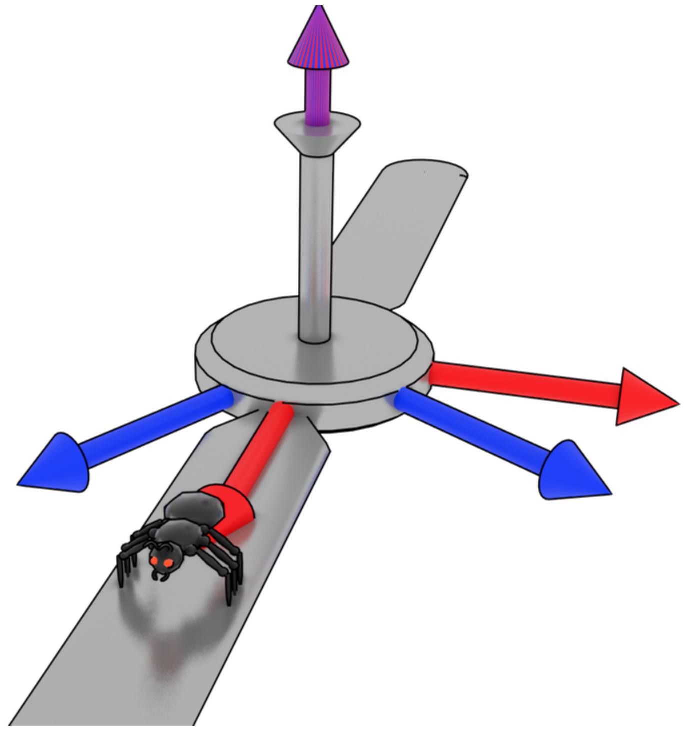
Criando um ponto móvel:

```
P = me.Point('P')
P.set_pos(O,r*A.y)
P.set_vel(N, r*A.y.dt(N))
```

Extraindo informações cinemáticas de um ponto:

```
P.pos_from(O)
P.vel(N)
P.acc(N)
```

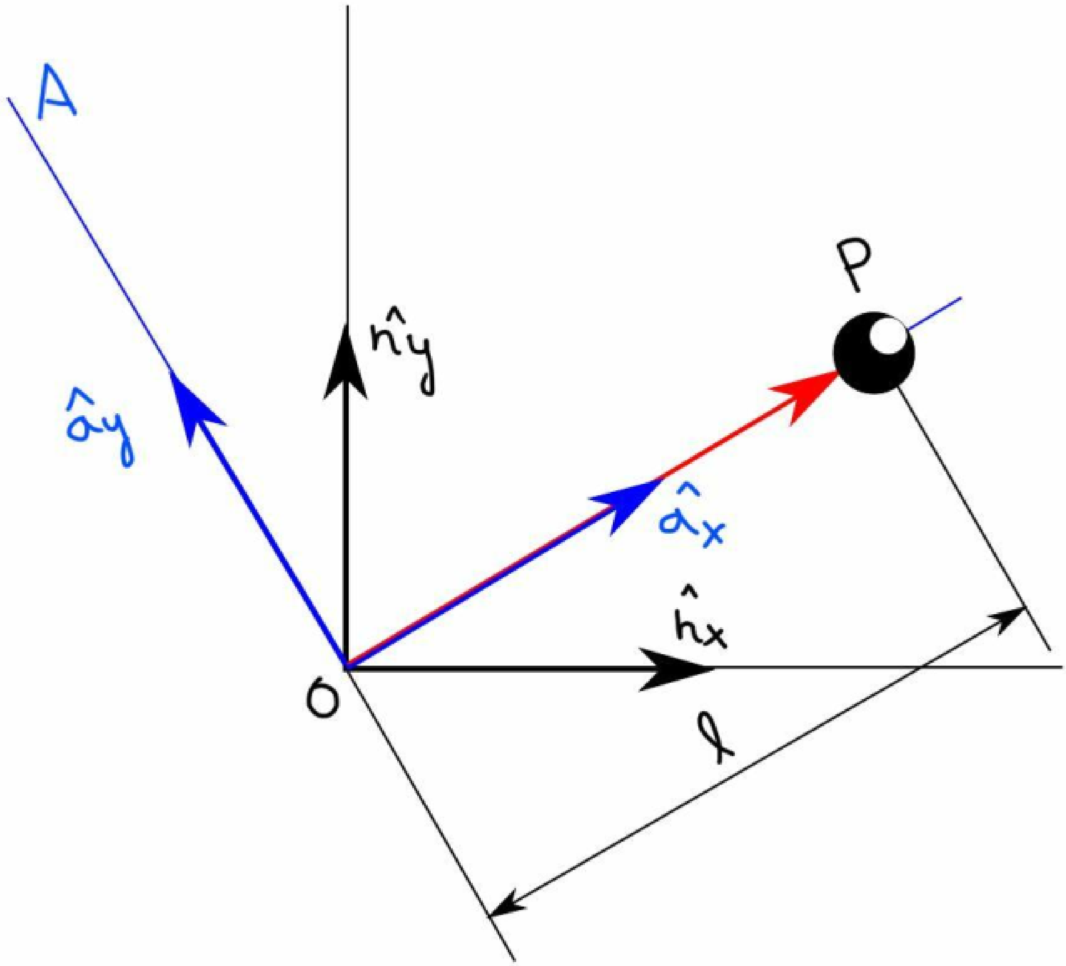
Exemplo1 : Obtenha a aceleração absoluta da formiga no ventilador.



Exemplo 2: Considere agora que a formiga está andando no ventilador. Obtenha a velocidade e aceleração da figura em relação a N e em relação a A.

Transformação de Coordenadas

Quando lidamos com vários referenciais, é interessante aprendermos a converter os vetores de base entre os diferentes referenciais.



$\mathbf{p}^{P/O} = l\hat{\mathbf{a}}_x$ ← queremos o vetor posição na base N
Para isso podemos utilizar as matrizes de rotações.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

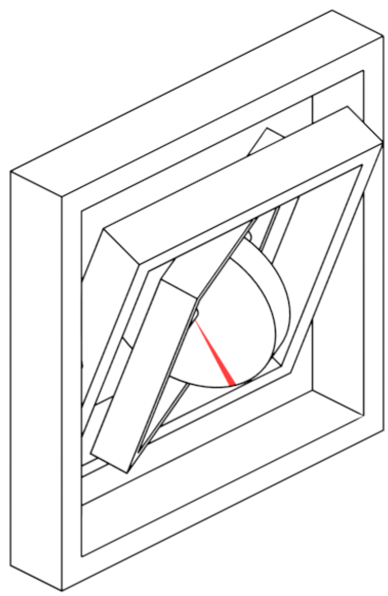
Para escrever o vetor $\mathbf{r}$ em outra base:

```
r.express(A)
```

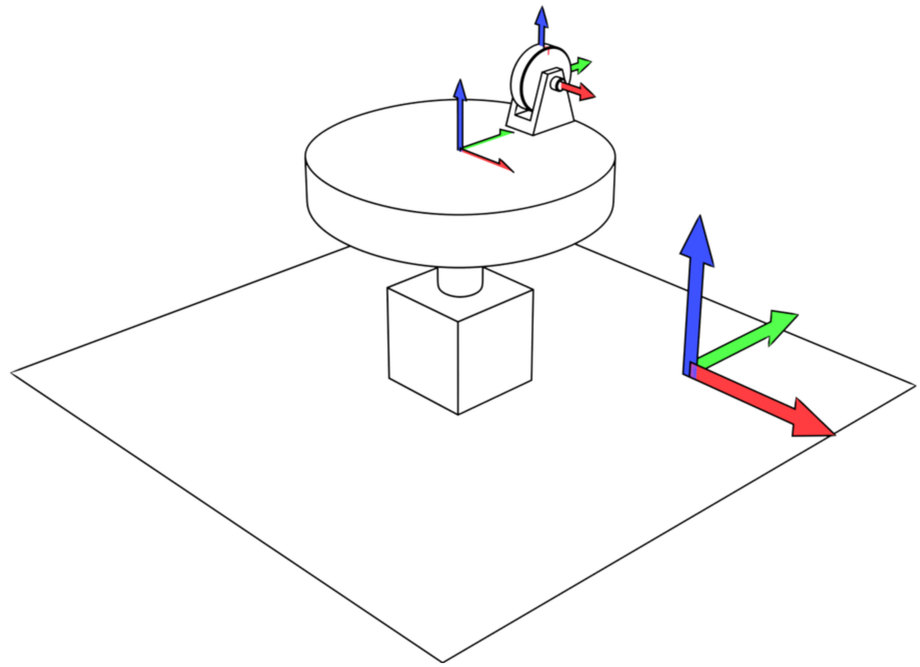
Para obter a matriz de rotação do referencial A em relação ao N:

```
A.dcm(N)
```

Exemplo 3: Obtenha a matriz de rotação do referencial fixo no disco em relação ao referencial inercial.



Exemplo 4: Obtenha a aceleração absoluta de um ponto na borda do disco menor:



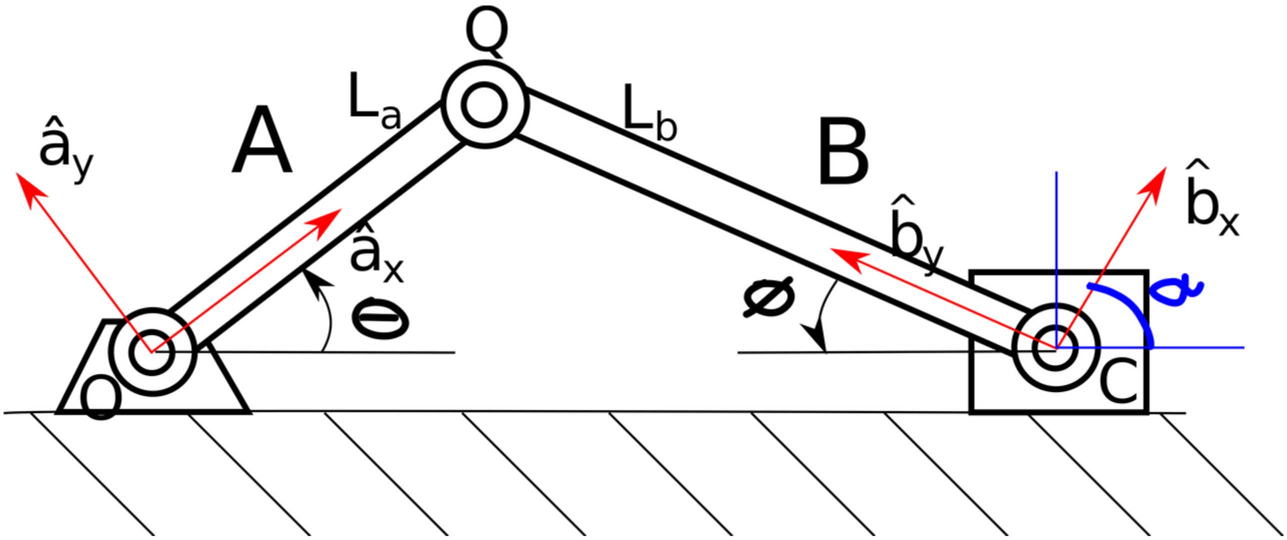
Mecanismos

- Quando um conjunto S de partículas ou corpos (ou ambos) se move em relação a um dado referencial R de tal modo que haja uma parcial ou total interdependência entre seus movimentos, tem se um mecanismo.
- A determinação da configuração do sistema se faz por meio de um conjunto de funções escalares do tempo, $q_1(t), q_2(t), ..., q_r(t)$ denominadas **coordenadas generalizadas**, ou simplesmente, coordenadas do sistema.

→ O número de coordenadas generalizadas mutuamente independentes de um sistema mecânico é conhecido como **número de graus de liberdade** do sistema.

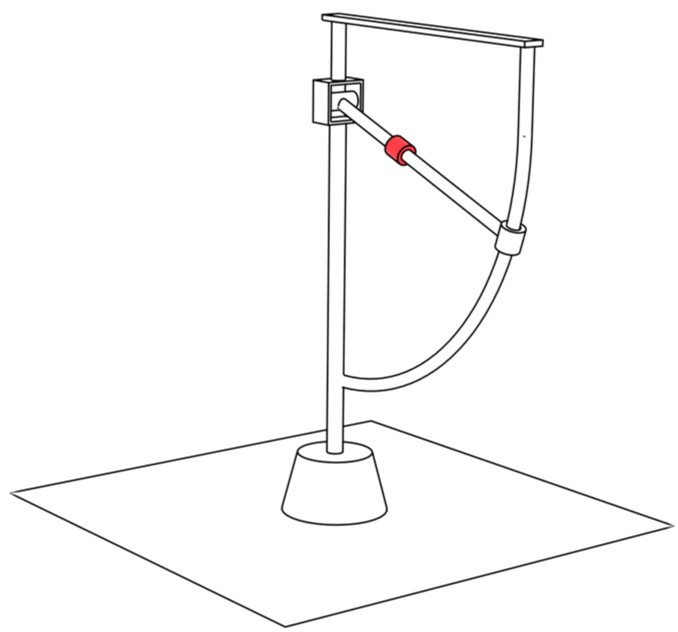
→As relações de interdependência entre os diferentes elos de um mecanismos podem ser obtidas através **Equações de Restrições Cinemáticas** ou **Equações de Vínculos**.

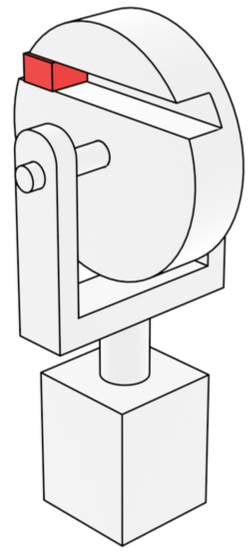
Exemplo 6: Obtenha a velocidade do pistão para o instante de tempo $t = 5s$.
Considere: $\theta = 0,5trad$; $\dot{\theta} = 0.5rad/s$; $L_a = 0,3m$ e $L_b = 0,7m$



Exercícios:

1 - Obtenha a aceleração absoluta dos cursores (objeto vermelho) das figuras abaixo:





Equações de Movimento - Método de Newton-Euler

Dinâmica da Partícula

Leis de Newton:

1. Todo corpo permanece em seu estado de repouso ou de movimento uniforme ao longo de uma reta, a menos que seja impelido a mudar de estado por forças sobre ele atuantes. (Princípio da inércia)

Vetor quantidade de movimento da partícula:

$$^{N}\mathbf{G}^{P} = m\,^{N}\mathbf{v}^{P}$$

→ Na ausência de forças externas, o vetor quantidade de movimento linear se conserva.

2. A mudança de movimento é proporcional à força motriz aplicada, e ocorre ao longo da linha reta onde esta força é aplicada.

$$\frac{{}^N d\mathbf{G}^P}{dt} = \frac{{}^N d\,(m^N\mathbf{v}^P)}{dt}$$

Se m for constante:
$${}^N\dot{\mathbf{G}}^P = m^N\dot{\mathbf{v}}^P$$

Logo, de acordo com a segunda lei de Newton:

$$\mathbf{R} = {}^N\dot{\mathbf{G}}^P$$

3. A uma ação corresponde sempre uma ação igual e contrária; ou, as ações mútuas entre dois corpos são sempre iguais e dirigidas em sentidos opostos.

Criando uma partícula:

```
Par = me.Particle('Par',P,m)
```

Podemos definir a energia potencial da partícula:

```
Par.potential_energy =
```

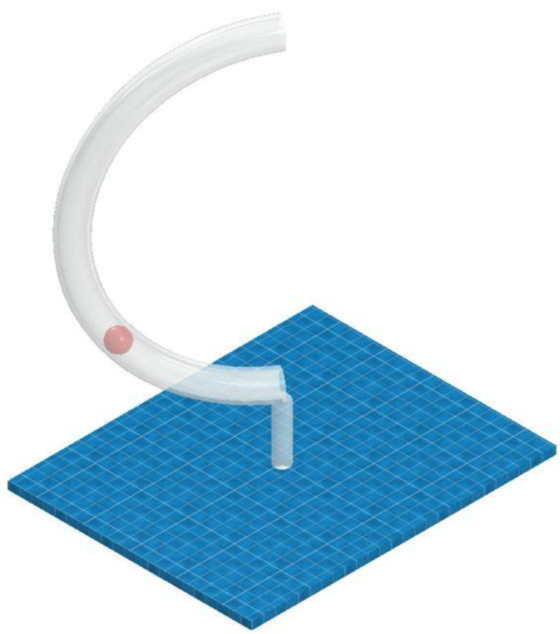
Extraindo informações:

```
Par.linear_momentum(N)
Par.angular_momentum(O, N)
Par.kinetic_energy(N)
Par.potential_energy
```

Diagrama de Corpo Livre

Ao utilizar o método de Newton, é importante desenhar, de modo esquemático, todos os corpos e partículas separados dos demais e com todas as forças as quais eles estão submetidos. Essa representação é conhecida como **Diagrama de Corpo Livre**.

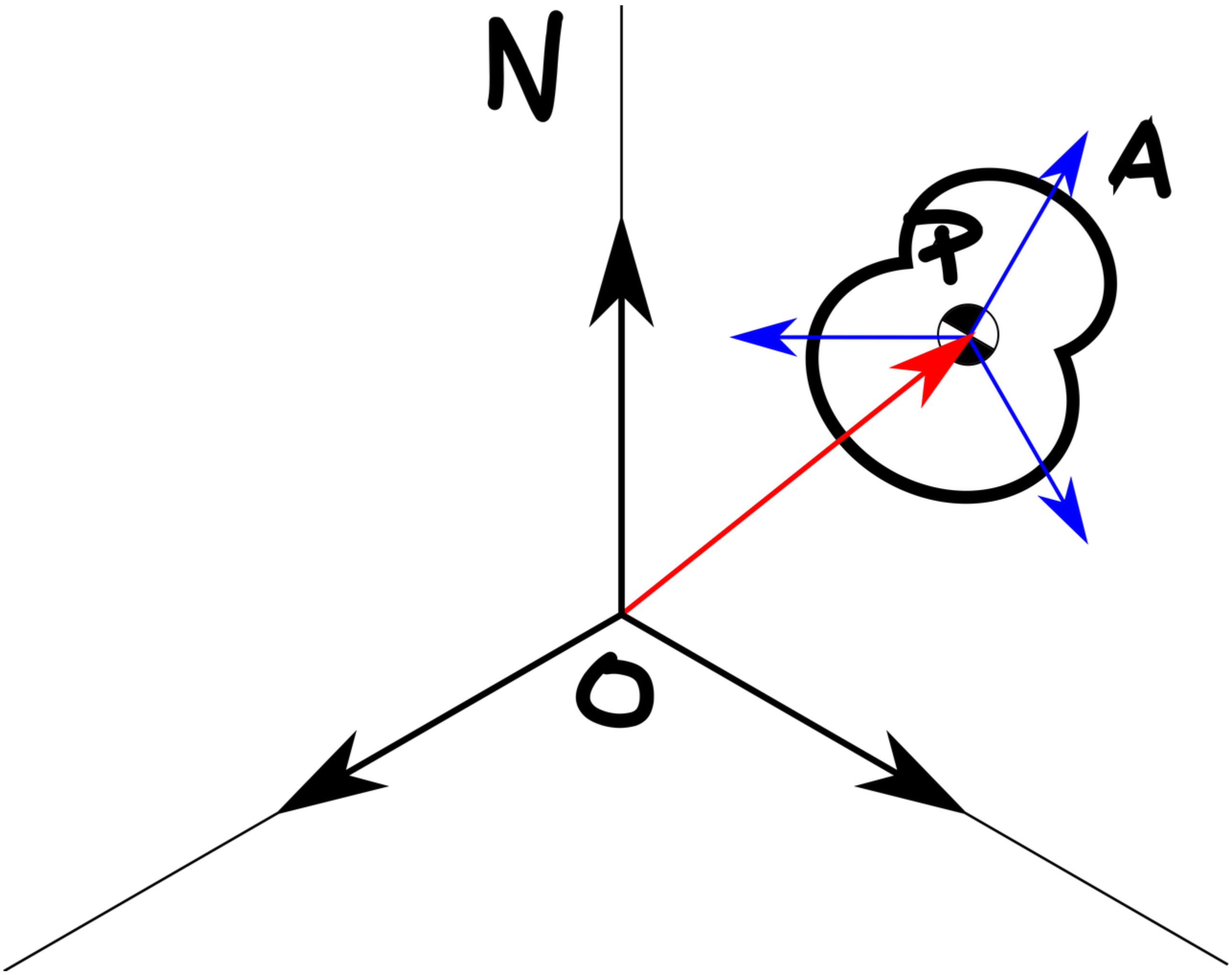
Exemplo 1: Obtenha as equações de movimento da partícula P, conforme mostrado na figura a seguir. Para a simulação considere os seguintes parâmetros:
Ângulo inicial: $\theta = \frac{\pi}{3} rad$
Velocidade angular inicial: $\dot{\theta} = 0 rad/s$
Velocidade angular do tubo: $\omega = 10 rad/s$
Raio de curvatura do tubo: $r = 0.5 m$
Gravidade: $g = 9.8 m/s^2$
Massa da partícula: $m = 1 kg$



Dinâmica do Corpo Rígido

Para a dinâmica de corpos rígidos, as equações de Newton são complementadas pelas equações de Euler.
Define-se um vetor quantidade de movimento angular da seguinte forma:

$$ {}^{N}\mathbf{H}^{P} = \mathbf{I}^{A/P} \cdot {}^{N}\omega^{A} $$



O **tensor de inércia** é uma matriz composta pelos momentos de inércia do corpo. É usual escrever esses momentos de inércia em relação a base móvel atrelada ao corpo rígido.

$$\mathbf{I}^{A/P} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

Definimos o momento da resultante das forcas e torques atuando no corpo A em relação ao ponto P como $\mathbf{M}^P$

$$\mathbf{M}^P = {}^N\dot{H}^P$$

Podemos criar um corpo rígido no python com os seguintes comandos:
Criando um corpo rígido:

```
I = me.inertia(A,Ixx,Iyy,Izz) # A é um referencial móvel atrelado ao corpo

D = me.RigidBody('D',P,A,m,(I,P)) # onde P é o ponto onde centro de massa do corpo D está localizado
```

Podemos definir a energia potencial do corpo com o comando:

```
D.potential_energy =
```

Extraindo informações:

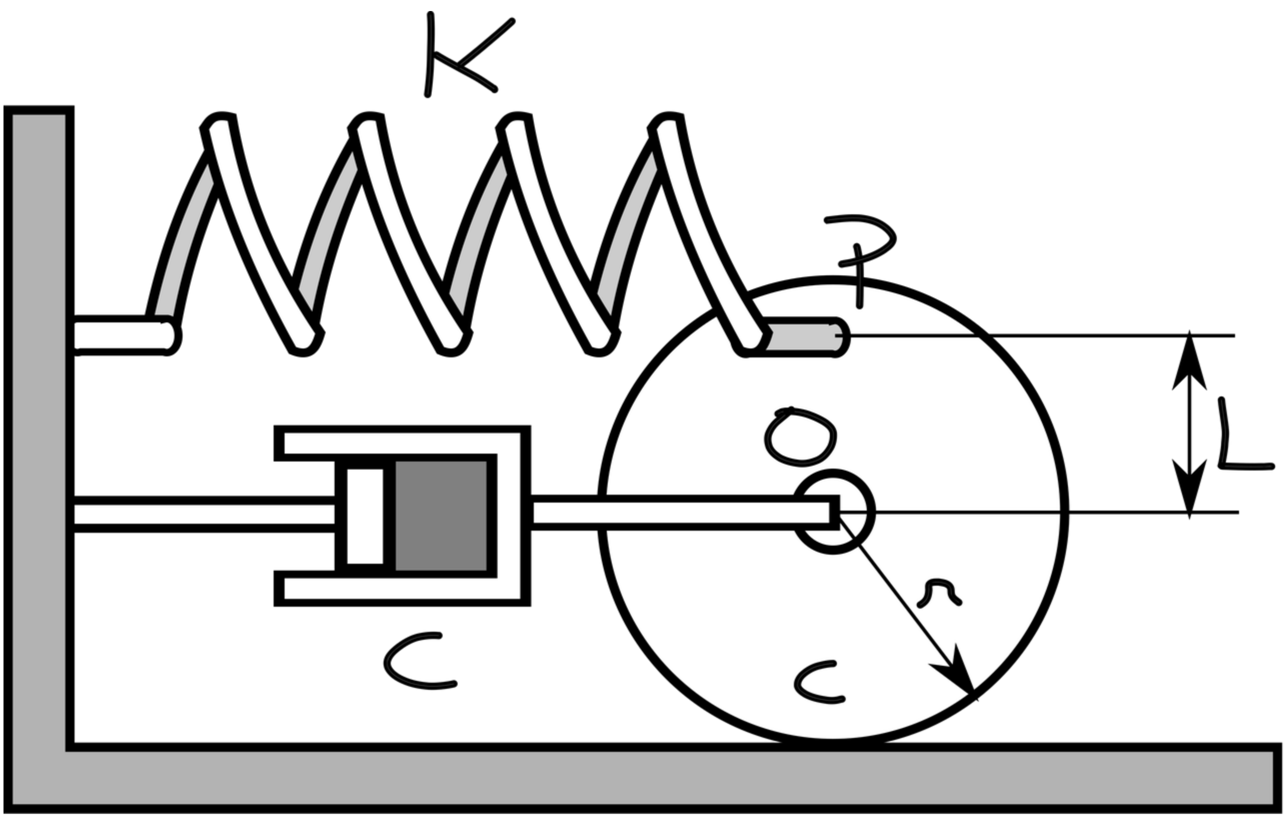
```
D.linear_momentum(N)

D.angular_momentum(O, N) # retorna o vetor quantidade de movimento angular em relação ao ponto O e ao referencial N

D.kinetic_energy(N)

D.potential_energy
```

Exemplo 2: Obtenha a equação de movimento do sistema mostrado abaixo:



Equações de Movimento - Método de Lagrange

- Através do Método de Lagrange se obtém as equações de movimento através de uma abordagem energética.
- Esse método é muito vantajoso para se obter as equações de movimento para sistemas de vários corpos e partículas.

Define-se o Lagrangeano L como:

$$L = T - U$$

Onde:
T → Energia cinética total do sistema
U → Energia potencial total do sistema

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = 0$$

Onde:
$q_i = q_1, q_2, ..., q_n$ ← em que n é o número de graus de liberdade do sistema
$\dot{q}_i = \frac{dq_i}{dt}$

Para criar o Lagrangeano no Python usamos o comando:

```
Lag = me.Lagrangian(Referencial_inercial,Corpos_e_partículas_separados_por_virgulas)
```

A seguir devemos criar um array com as coordenadas generalizadas:

```
q = [q1,q2,...]
```

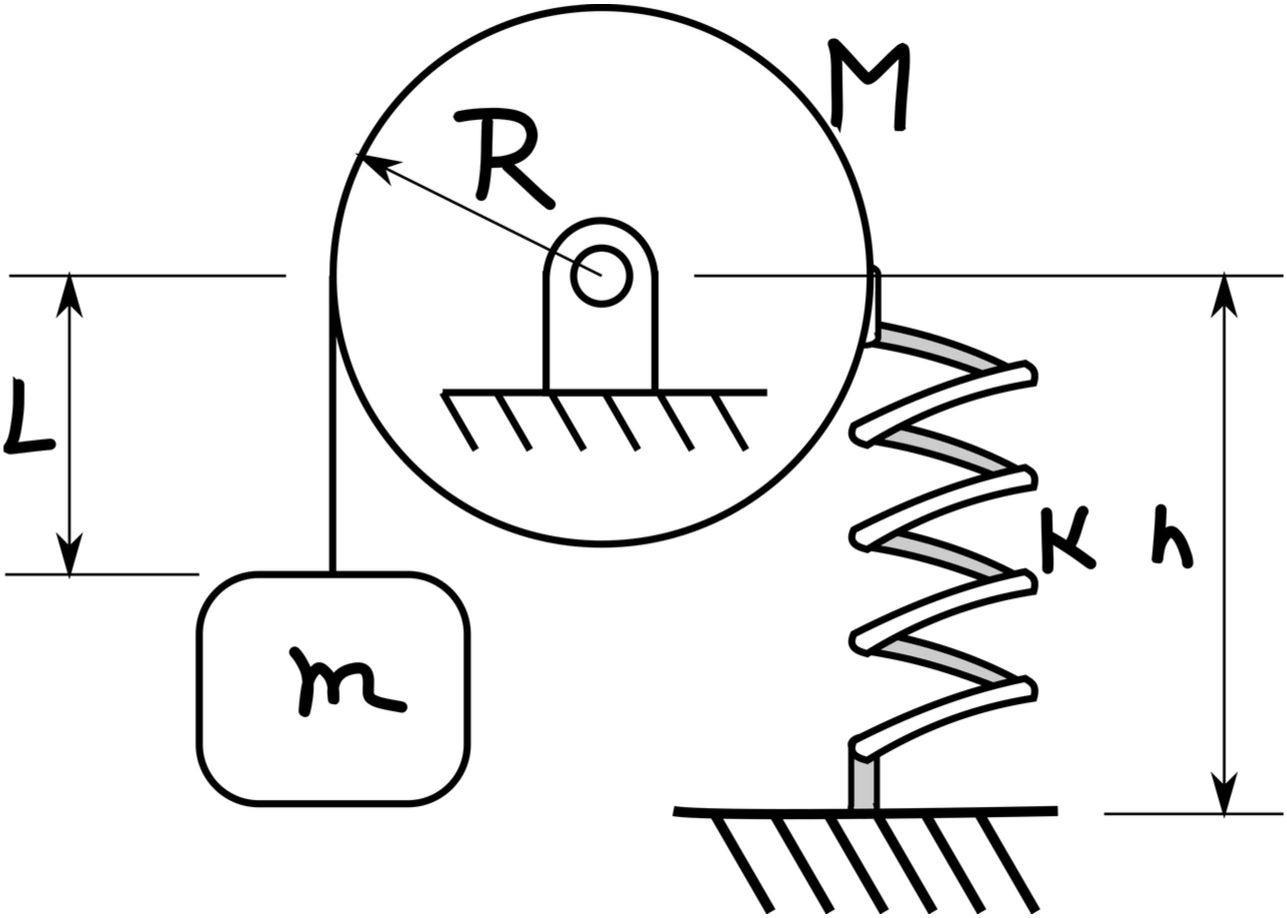
Em seguida usamos o seguinte comando:

```
l = me.LagrangesMethod(Lag,q)
```

Finalmente, utilizamos o seguinte comando para obter as equações de movimento:

```
le = l.form_lagranges_equations()
```

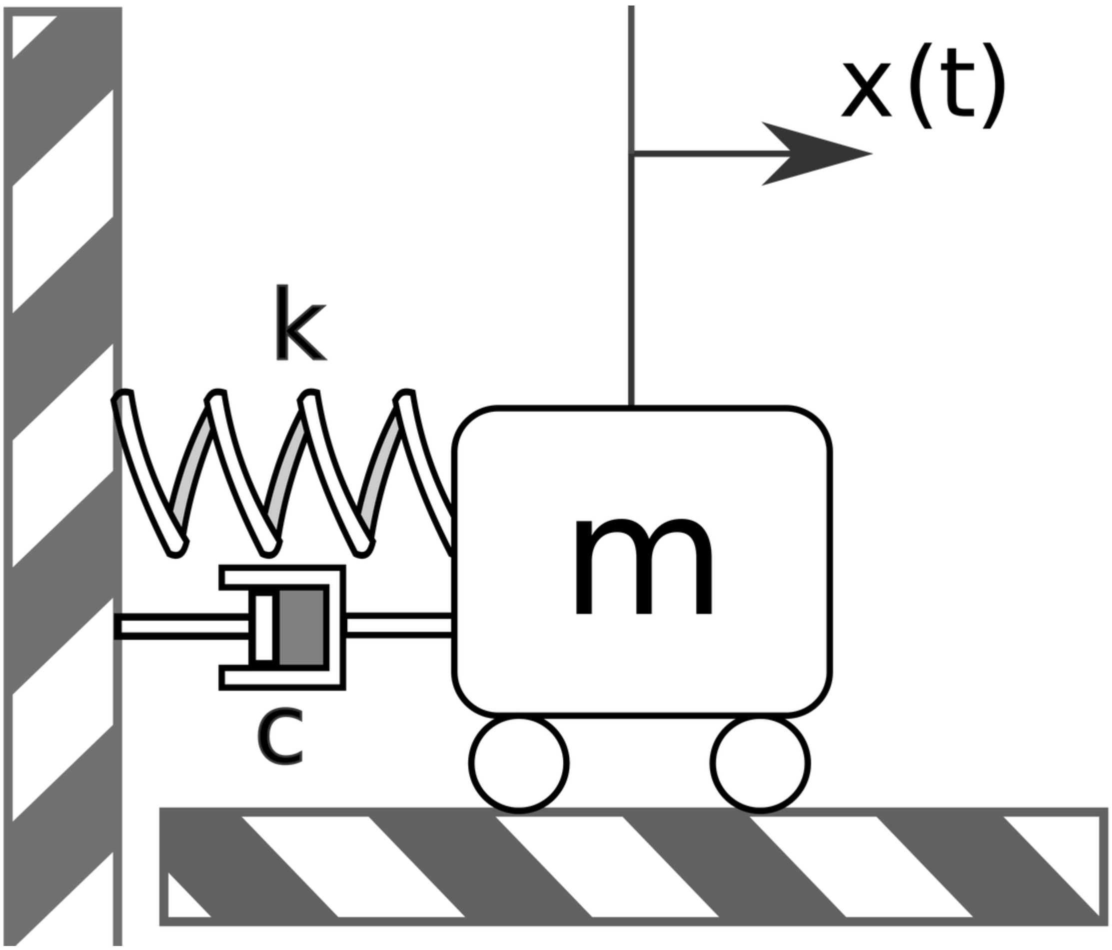
Exemplo 1:
Obtenha a equação de movimento do sistema abaixo:



Simulações

- Simular um sistema consiste em resolver suas equações diferenciais características.
- Para o caso da dinâmica de sistemas mecânicos, temos **equações diferenciais de segunda ordem.**
- De modo geral, não conseguimos resolver essas equações de modo analítico, por isso temos que recorrer a métodos numéricos.
- Para integrar numericamente essa EDO de segunda ordem, precisamos **transformá-la em duas EDOs de primeira ordem.**

Exemplificaremos este procedimento com o sistema massa-mola-amortecedor:



A equação de movimento desse sistema pode ser escrita da seguinte maneira:
$m\ddot{x} + c\dot{x} + kx = 0$
Para integrarmos numericamente essa equação, precisamos reescrevê-la da seguinte forma:

$\dot{x} = v$
$\dot{v} = (-cv - kx)/m$

Podemos armazenar essas variáveis em um *array* → $X = [x, v]$ e $\dot{X} = [\dot{x}, \dot{v}]$

Integração Numérica no Python

Primeiramente, precisamos substituir as constantes simbólicas por constantes numéricas nas equações.
Um jeito prático de fazer isso é através de dicionários e do comando subs.
Exemplo de criação de um dicionário para: $m = 1kg, L = 2m$ e $g = 9.8m/s^2$

```
dicionario = {m:1,L:2,g:9.8}
```

Para substituir esses valores nas equações basta usar o comando *subs*.

```
Eqs.subs{dicionario}
```

Precisaremos importar a biblioteca de computação numérica:

```
import numpy as np
```

Criaremos um tempo numérico que se inicia em 0 segundos, termina em 20 segundos e possui 1000 pontos:

```
T = np.linspace(0,20,1000)
```

Precisamos definir as condições iniciais do nosso sistema:

```
x0 = 0.0001

v0 = 0.0001

X0 = [x0,v0]
```

Em seguida, precisamos criar um função no Python que funcione de acordo com o sistema das 2 EDOs de primeira ordem:

```
def Modelo (X,T)

 dx_dt = X[1]

 dv_dt = Eq.subs({x:X[0],x.diff(t):X[1]})

 return [dx_dt,dv_dt]
```

Por fim, basta importar o comando que fará a integração numérica e usá-lo.

```
from scipy.integrate import odeint

X = odeint(Modelo,X0,T)
```

Visualização de Resultados

Para plotar os resultados usaremos a biblioteca matplotlib.pyplot

```
import matplotlib.pyplot as plt
```

Podemos criar uma gráfico da seguinte forma:

```
plt.figure()
plt.plot(T,X[:,0]) # plota o gráfico de x por t, para plotar v
por t use -> X[:,1]
plt.xlabel('legenda do eixo horizontal')
plt.ylabel('legenda do eixo vertical')
plt.title('Título do gráfico')
plt.grid(True) # cria uma grade
```

Podemos plotar o deslocamento e a velocidade juntos como:

```
plt.figure()
plt.plot(T,X)
plt.xlabel('legenda do eixo horizontal')
plt.ylabel('legenda do eixo vertical')
plt.title('Título do gráfico')
plt.grid(True) # cria uma grade
plt.legend(['x','v']) # plota uma legenda de acordo com as cor
es
```

Livros Recomendados

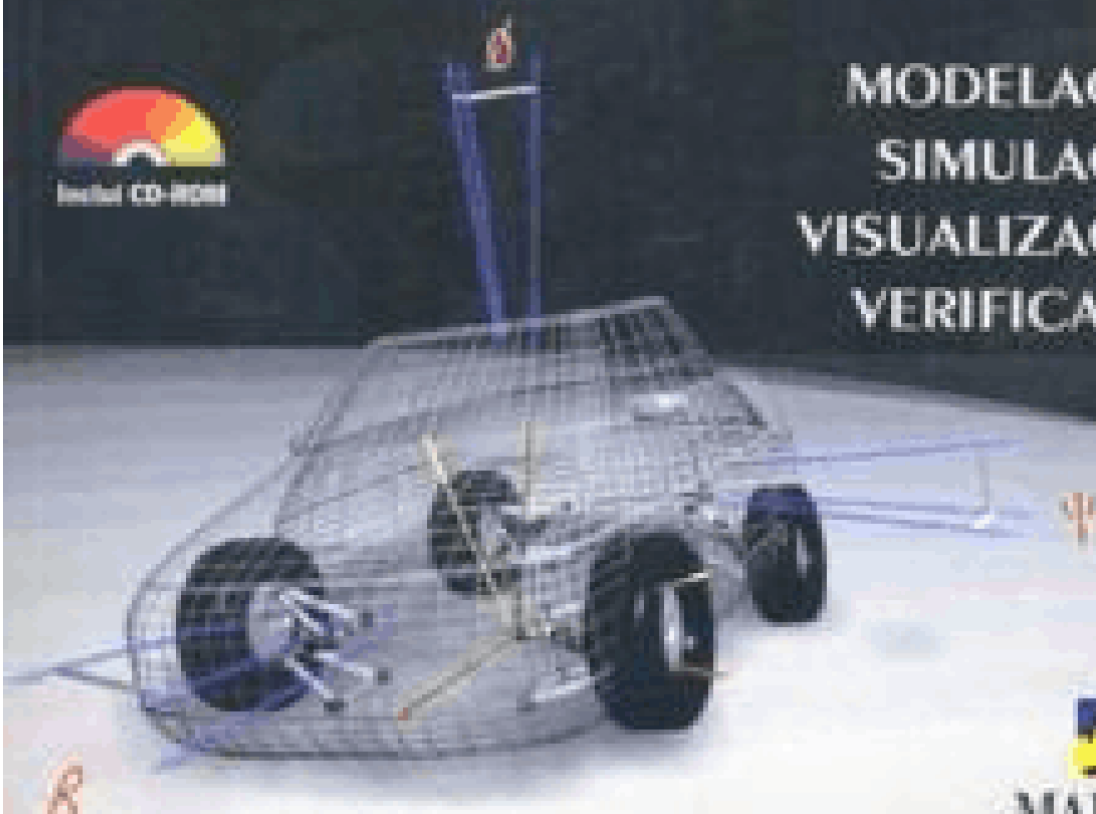
Dinâmica de Sistemas Mecânicos(Imar Santos)

- SANTOS, Ilmar Ferreira. **Dinâmica de sistemas mecânicos: modelagem, simulação, visualização, verificação.** Makron, 2001.

DINÂMICA DE SISTEMAS MECÂNICOS

Inclui CD-ROM

MODELAGEM,
SIMULAÇÃO,
VISUALIZAÇÃO,
VERIFICAÇÃO

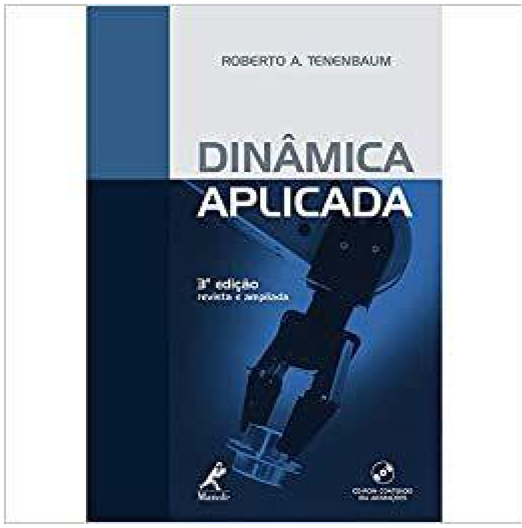


ILMAR FERREIRA SANTOS

MAKRON
Books

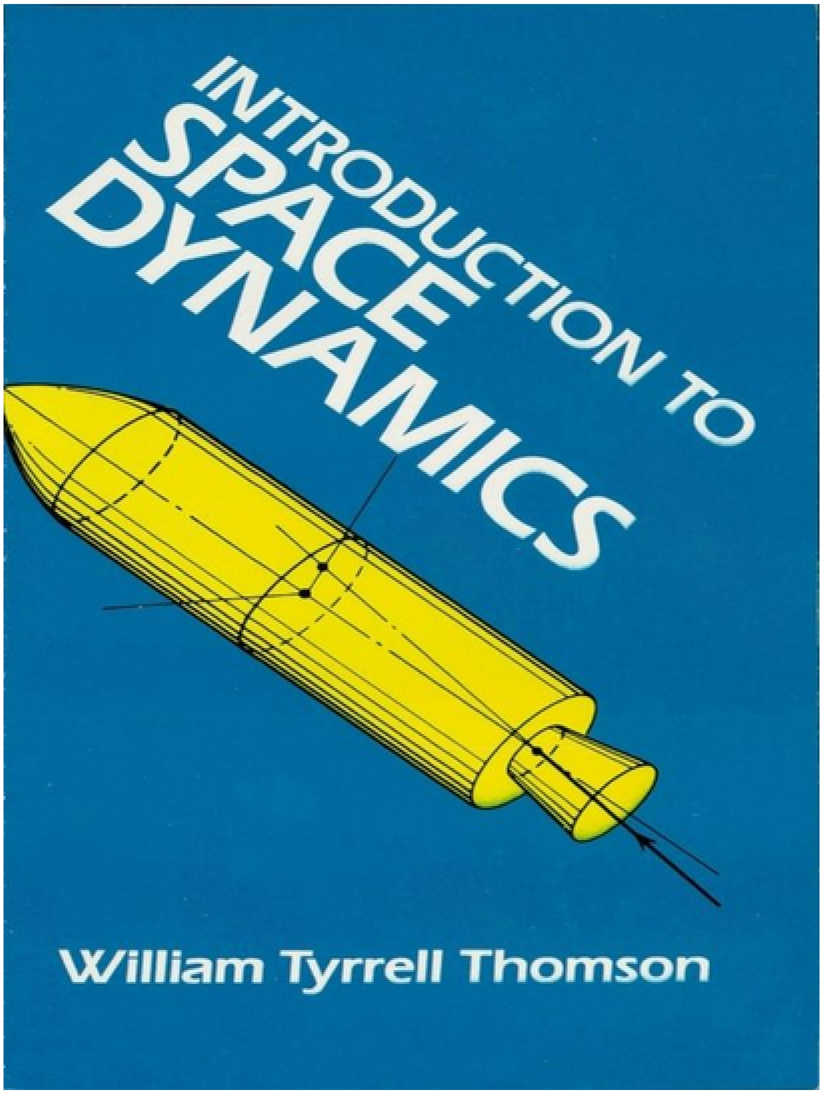
Dinâmica Aplicada (Roberto A. Tenenbaum)

- TENENBAUM, Roberto A. **Dinâmica aplicada.** Editora Manole Ltda, 2006.



Introduction to Space Dynamics (William Tyrrell Thomson)

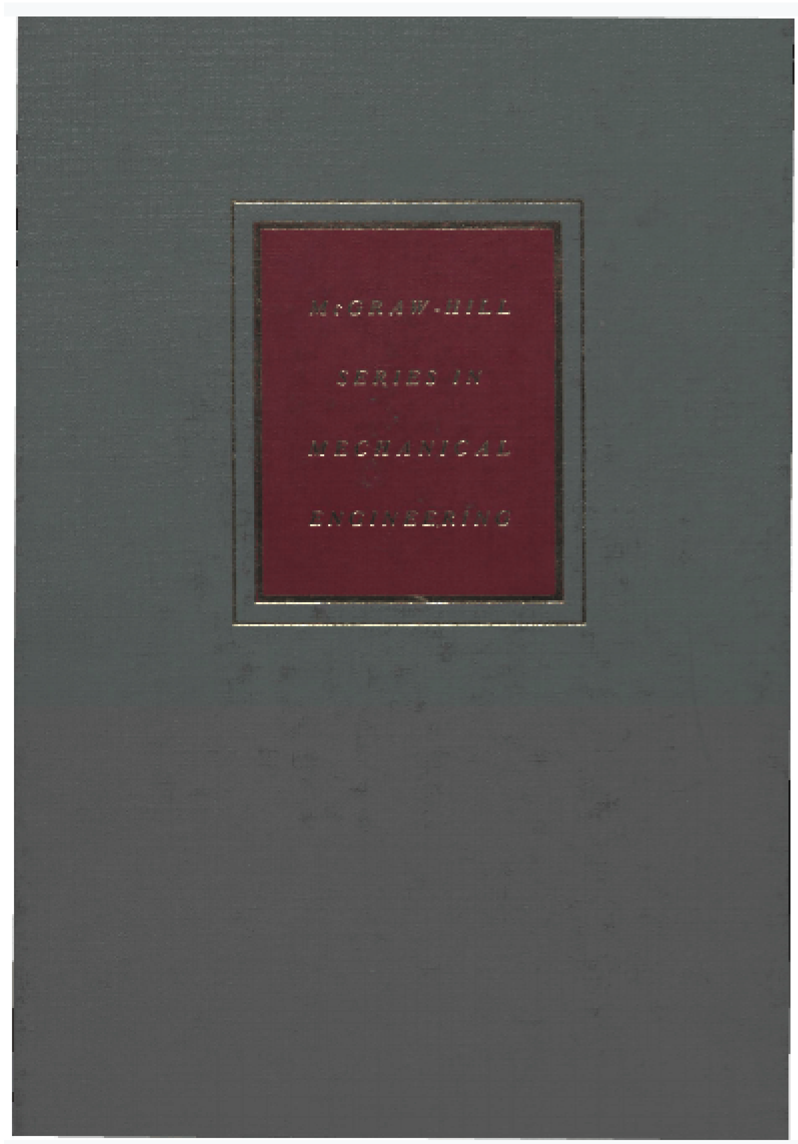
- THOMSON, William Tyrrell. **Introduction to space dynamics.** Courier Corporation, 2012.



Dynamics, Theory and Application (Thomas R. Kane e David A. Levinson)

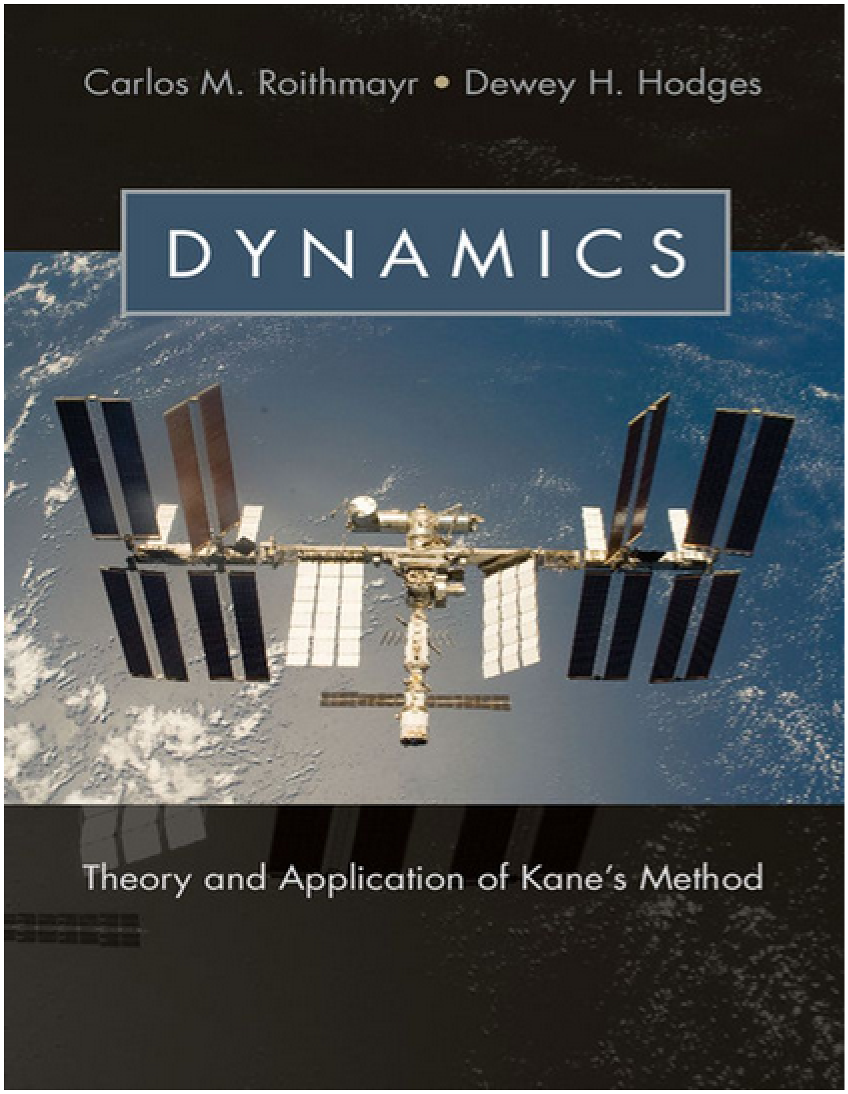
- KANE, Thomas R.; LEVINSON, David A. **Dynamics, theory and applications**. McGraw Hill, 1985.

→ Download legal gratuito em: https://ecommons.cornell.edu



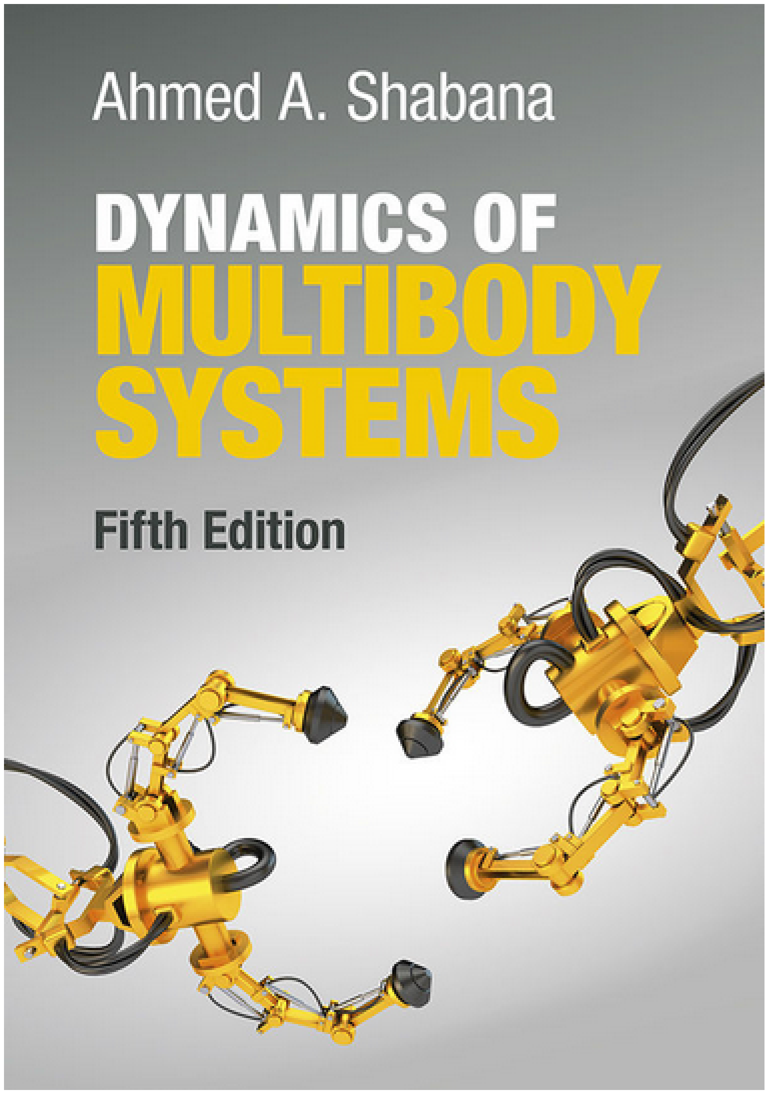
Dynamics, Theory and Application of Kane’s Method (Carlos M. Roithmayr e Dewey H. Hodges)

- ROITHMAYR, Carlos M.; HODGES, Dewey H. Dynamics: theory and application of Kane's method. 2016.



Dynamics of Multibody Systems (Ahmed Shabana)

- SHABANA, Ahmed. **Dynamics of multibody systems**. Cambridge university press, 2020.


Ahmed A. Shabana
DYNAMICS OF
MULTIBODY
SYSTEMS
Fifth Edition