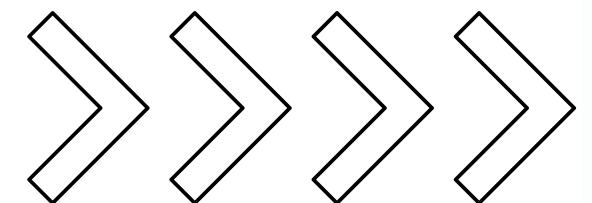


# **CONTROLADOR POR ALOCAÇÃO DE PÓLOS**

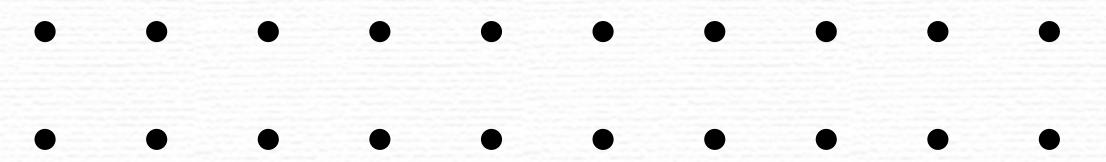


**Aluno:** Pedro H. Grossi da Silva

# Índice

- 03** Problematização
- 04** Artigo Base
- 05** Modelagem matematica
- 08** Design do controlador

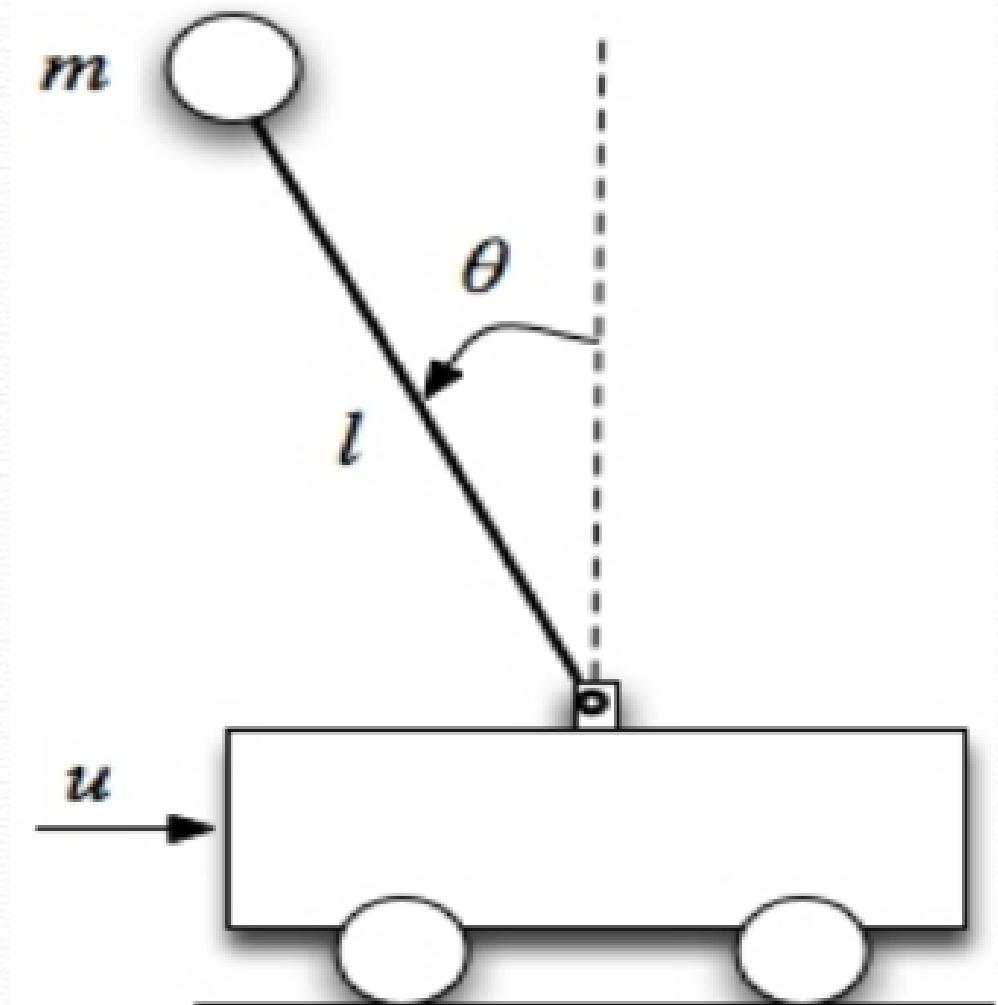
- 11** Adaptação do artigo
- 12** Trechos do codigo
- 14** Comparação com PID



# Problematização

## Equilibrio de um pendulo invertido

- Pêndulo fixado a um carrinho móvel.
- Objetivo: estabilizar o pêndulo na posição vertical (invertida).
- Problematica: sistema instável por natureza, exigindo controle preciso para manter o pêndulo equilibrado.



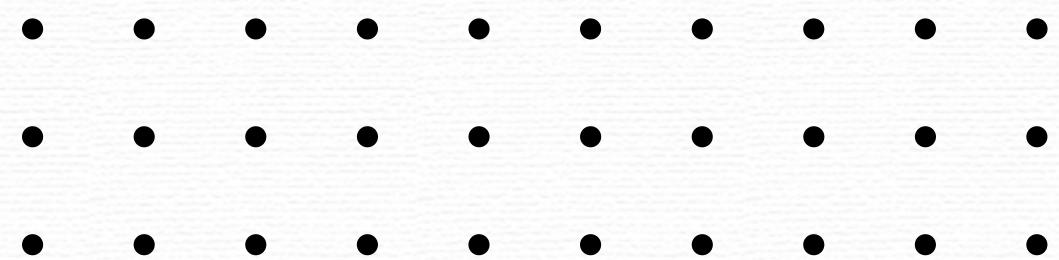
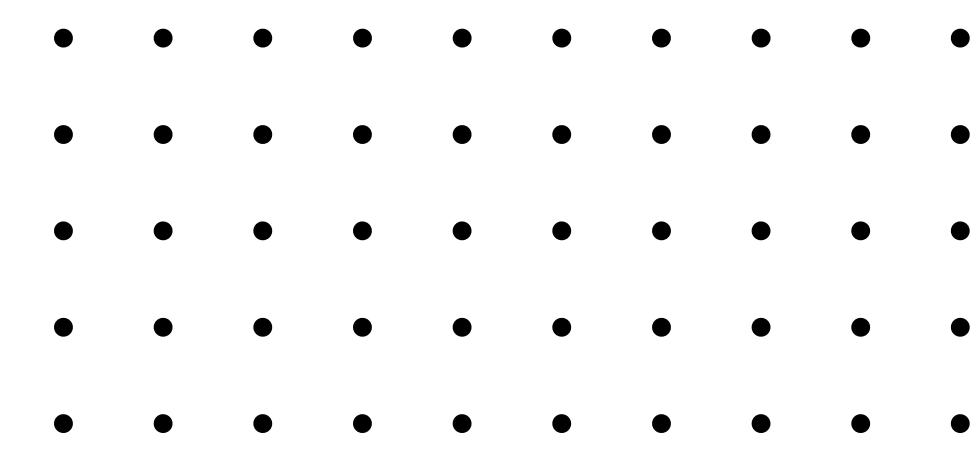
# Artigo base

**01** Stabilization Of Inverted Pendulum On Cart  
Based On Pole Placement and LQR  
Ramashis Banerjee, Arnab Pal

# Modelagem matematica

## 01 Premissas para modelagem

- O sistema é um corpo rígido.
- Não há deslizamento relativo presente em todo o sistema.
- O atrito entre o carrinho e o trilho é diretamente proporcional à velocidade do carrinho



# Modelagem matemática

## 02 Equações

$$M \frac{d^2 y}{dt^2} + m \frac{d^2 x_G}{dt^2} = u$$

Equilíbrio de forças em x

$$x_G = x + l \sin \theta$$

$$y_G = l \cos \theta$$

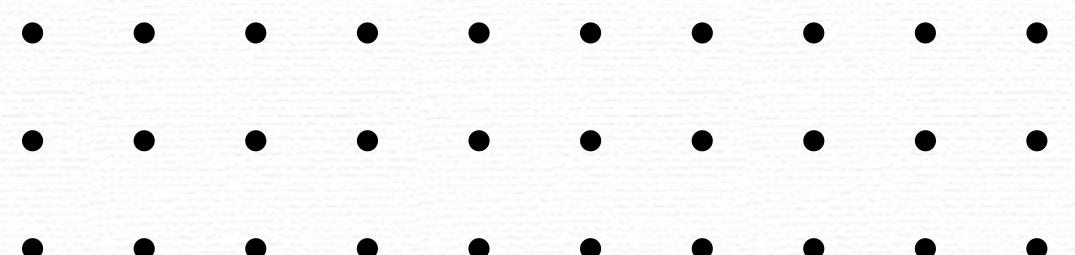
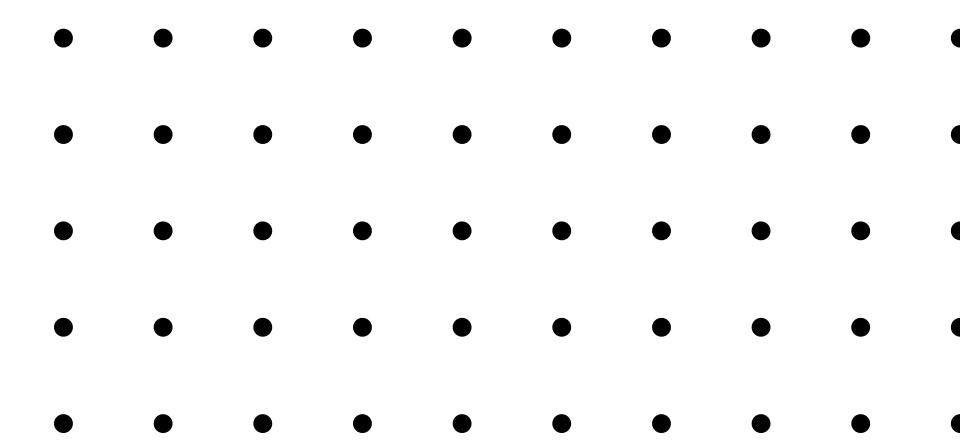
Coordenadas do centro de massa do pêndulo

$$\ddot{x} = \frac{u + ml(\sin \theta)\dot{\theta}^2 - mg \sin \theta \cos \theta}{M + m - m(\cos \theta)^2}$$

Aceleração do carrinho

$$\ddot{\theta} = \frac{u \cos \theta - (M + m)g \sin \theta + ml\dot{\theta}^2 \sin \theta \cos \theta}{ml(\cos \theta)^2 - (M + m)l}$$

Aceleração angular do pendulo



# Modelagem matematica

## 03 Espaço de estados

$$\ddot{x} = \frac{u + ml(\sin \theta)\dot{\theta}^2 - mg \sin \theta \cos \theta}{M + m - m(\cos \theta)^2}$$

$$\ddot{\theta} = \frac{u \cos \theta - (M + m)g \sin \theta + ml\dot{\theta}^2 \sin \theta \cos \theta}{ml(\cos \theta)^2 - (M + m)l}$$

- Assumindo θ proximo de zero
- Massa do carrinho = 1 kg
- Massa do pendulo = 0.1 kg
- Compimento do pendulo = 0.5m
- Gravidade = 9.8m/s<sup>2</sup>

$$\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 15.8180 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 1.4674 \end{bmatrix} u$$

Espaço de estados apos linearização

$$y = [1 \ 0] \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

y é a saida, θ é a posição angular e  
θ(ponto)□ é a velocidade angular

# Design do controlador

## Objetivo

Projetar um controlador capaz de estabilizar o pêndulo invertido (naturalmente instável) e melhorar sua resposta transitória.

## Modelo no espaço de estados e realimentação

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

$$u = -kx$$

$$\dot{x} = (A - Bk)x$$

## Polos

Os auto valores da matriz são chamados de polos

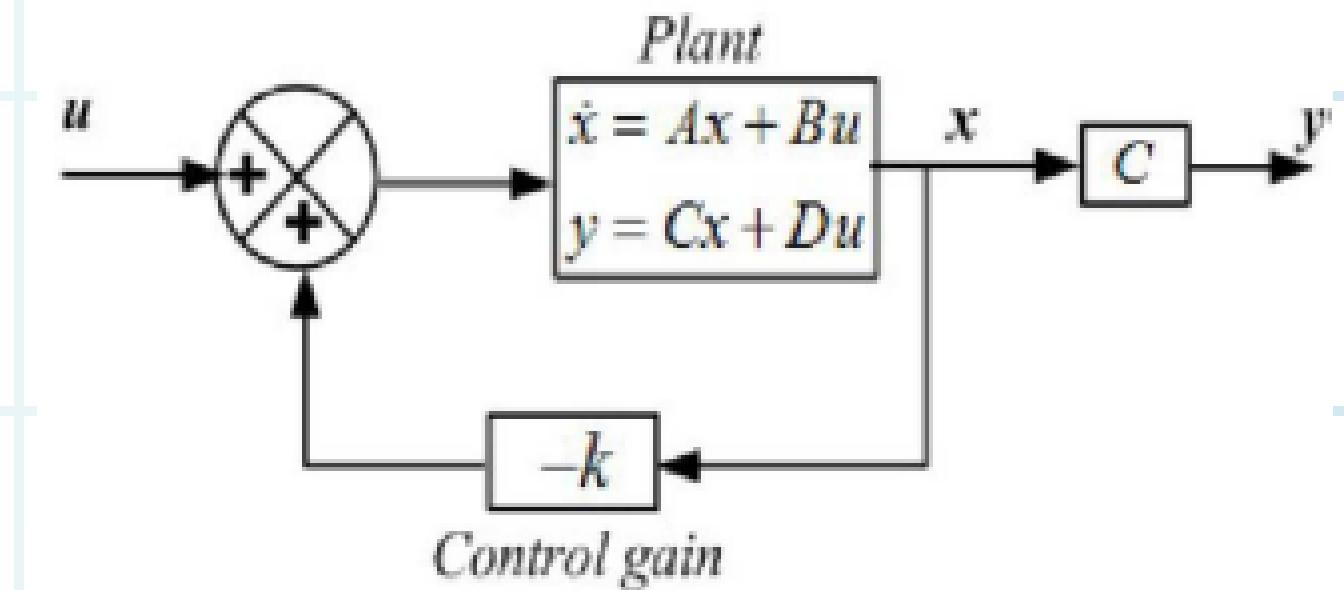
$$|SI - (A - Bk)| = 0$$

Os polos determinam:

- Estabilidade (parte real negativa = sistema estável)
- Rapidez de resposta (mais à esquerda no plano complexo, mais rápido o sistema)
- Oscilações (polos complexos → comportamento oscilatório)

# Design do controlador (pole placement)

“Projetar o controlador significa selecionar a matriz de ganho em feedback de modo que os polos de malha fechada do sistema sejam colocados na localização desejada no plano complexo.”



# Design do controlador (pole placement)

- A matriz de controlabilidade é calculada
- Tempo de estabilização da posição angular = 2.2s
- Velocidade angular = 2.5s
- Polos dominantes =  $(-2,5 + 2,5j)$  e  $(-2,5 - 2,5j)$

Por tentativa e erro é verificado que o sistema é estável quando o valor da matriz de ganho de realimentação de estado é calculado usando a fórmula de Ackermann.

$$k = [19.2988 \quad 3.4075]$$

# Adaptação do artigo

## ➤ Ampliação dos estados de controle

Ampliação do sistema para 4 estados, incluindo o movimento do carrinho.

Os estados agora são:

- $x \rightarrow$  posição do carrinho
- $x(\text{ponto}) \rightarrow$  velocidade do carrinho
- $\theta \rightarrow$  ângulo do pêndulo
- $\theta(\text{ponto}) \rightarrow$  velocidade angular

$$(M + m)\ddot{x} + ml \cos \theta \ddot{\theta} - ml \sin \theta \dot{\theta}^2 = u - k\dot{x}$$

$$ml \cos \theta \ddot{x} + ml^2 \ddot{\theta} - mgl \sin \theta = -c\dot{\theta}$$

Modelo não linear de segunda ordem

Novos polos:

- $(-2.5 + 2.5j)$  e  $(-2.5 - 2.5j) \rightarrow$  reutilizados do artigo
- $(-3)$  e  $(-3.5) \rightarrow$  por tentativa e erro

# Trechos do código

```
M = 0.5 # Cart mass [kg]
k = 0.1 # Cart friction [N.s/m]

m = 0.2 # Pendulum mass [kg]
c = 0.01 # Pendulum friction [N.s/rad]
l = 0.3 # Length [m]

g = 9.81 # Gravity [m/s^2]

Lt = 50 # Simulation time [s]
dt = 0.01 # Discretization [s]
```

```
A = np.array([
    [0, 1, 0, 0],
    [0, 0, -m * g / M, 0],
    [0, 0, 0, 1],
    [0, 0, g * (M + m) / (M * l), 0]
], dtype=float)

B = np.array([0, 1.0 / M, 0, -1.0 / (M * l)], dtype=float).reshape(4, 1)

p_des = np.array([-2.5 + 2.5j, -2.5 - 2.5j, -3, -3.5])

K = place_poles(A, B, p_des).gain_matrix # 1x4
K = np.atleast_2d(K)
K = K.reshape(1, 4)
# print("K = ", K)
```

# Trechos do código

```
# (M + m) * xdd + m*l*cos(theta)*thetadd - m*l*sin(theta)*thetadot^2 = u - k*xdot
# m*l*cos(theta)*xdd + m*l^2*thetadd - m*g*l*sin(theta) = - c*thetadot
def dynamics(state, u):
    x, xdot, th, thdot = state
    s, cth = np.sin(th), np.cos(th)
    D = np.array([[M + m,      m * l * cth],
                  [m * l * cth, m * l**2]], dtype=float)
    rhs1 = u - k * xdot + m * l * s * thdot**2
    rhs2 = m * g * l * s - c * thdot
    rhs = np.array([rhs1, rhs2], dtype=float)
    xdd, thdd = np.linalg.solve(D, rhs)
    return np.array([xdot, xdd, thdot, thdd], dtype=float)
```

```
# u = -K (x - x_ref)
# ref = [spx, 0, 0, 0]
for i in range(Nt):
    ref = np.array([spx[i], 0.0, 0.0, 0.0])
    err = state - ref
    u = float(-K @ err)
    u_total = u + np.random.randn()
    u_total = max(min(u_total, u_max), -u_max)
    us[i] = u_total
    hist[i, :] = state
    state = rk4_step(state, u_total, dt)
```

# Comparação com PID

