UNIVERSIDADE LUSÓFONA

Controle de Pêndulo Invertido

Nome: Mauro Carvalho – 22402165

Professor: Plínio Lopez

Data: 09/06/2025

Repositório GitHub: https://github.com/CMauro1910/Projeto-individual3.git

Controle do Pêndulo Invertido

1. Introdução

O controle de pêndulo invertido é um problema clássico da engenharia de controle. Consiste em equilibrar um pêndulo na posição vertical instável, geralmente montado sobre uma base móvel. Neste trabalho, utilizamos o simulador Webots para modelar esse sistema e implementar um controlador PID que permita estabilizar o pêndulo automaticamente.

2. Metodologia

Foi utilizado o Webots com um modelo de robô contendo rodas motorizadas e um pêndulo com sensor de posição. A linguagem de programação usada foi Python. O algoritmo implementado se baseia em controle PID, com parâmetros variaveis de Kp, Ki e Kd. O objetivo foi avaliar o comportamento do sistema com valores constantes durante o tempo de simulação.

3. Explicação do Código

O código inicia criando um objeto da classe Robot e obtendo o tempo de passo da simulação.

Em seguida, o sensor de posição do pêndulo é ativado com o tempo de amostra no site localhost(localhost:1234/robot_windows/inverted_pendulum_benchmark/inverted_pendulum_benchmark.html?name=robot) que é criado automaticamente ao se iniciar a simulação, e os motores das rodas são configurados para controle de velocidade (posição infinita).

São definidos os parâmetros do controlador PID: ganho proporcional (Kp), integral (Ki) e

derivativo (Kd), além de variáveis auxiliares como o erro anterior e a integral acumulada.

Dentro do laço principal, o código lê o ângulo do pêndulo e calcula a ação de controle com base no PID. Essa ação é aplicada nas rodas para tentar corrigir a inclinação do pêndulo. Se o ângulo ultrapassar 90 graus, o controle é interrompido.

Esse tipo de controle é bastante sensível, exigindo ajustes finos para que o pêndulo permaneça em equilíbrio.

Inicialização do robô

```
robot = Robot()

timestep = int(robot.getBasicTimeStep())
```

Função: Cria um objeto do robô e obtém o tempo de passo da simulação. Esse valor determina de quanto em quanto tempo o controlador será executado.

Sensor do pêndulo

```
ps = robot.getDevice('pendulum sensor')

ps.enable(timestep)
```

Função: Ativa o sensor de posição do pêndulo, que vai medir seu ângulo durante a simulação.

Motores das rodas

```
leftMotor = robot.getDevice("left wheel motor")

rightMotor = robot.getDevice("right wheel motor")

leftMotor.setPosition(float('+inf'))

rightMotor.setPosition(float('+inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)
```

Função: Obtém os motores das rodas do robô e configura para modo de controle por velocidade (posição infinita). Inicializa com velocidade zero.

Parâmetros do controlador PID

KP = 100

KI = 90

KD = 0

integral = 0.0

previous_position = 0.0

Função: Define os ganhos do controle PID e inicializa variáveis auxiliares:

KP: ganho proporcional;

KI: ganho integral;

KD: ganho derivativo;

integral: acumula o erro ao longo do tempo;

previous_position: usado para calcular a derivada do erro

Leitura do ângulo do pêndulo

position = ps.getValue()

Função: Obtém a posição angular atual do pêndulo.

Parar o robô se o pêndulo cair

if math.fabs(position) > math.pi * 0.5:

  leftMotor.setVelocity(0.0)

  rightMotor.setVelocity(0.0)

  break

Função: Se o ângulo do pêndulo ultrapassar 90° para qualquer lado, o robô para — sinal de que o pêndulo caiu.

Controle PID

integral = integral + (position + previous_position) * 0.5 / timestep

derivative = (position - previous_position) / timestep

speed = KP * position + KI * integral + KD * derivative

Função: Calcula a ação de controle com base na posição atual:

1. A integral acumula o erro ao longo do tempo;
2. A derivada estima a velocidade da inclinação;
3. A saída speed é a combinação ponderada dos três termos do PID.

Limitando a velocidade

if speed > maxSpeed:

  speed = maxSpeed

elif speed < -maxSpeed:

  speed = -maxSpeed

Função: Garante que a velocidade não ultrapasse os limites físicos do motor.

Aplicando a ação de controle

leftMotor.setVelocity(-speed)

rightMotor.setVelocity(-speed)

Função: Define a velocidade das rodas com o valor calculado pelo controlador (em direções iguais para mover o robô para frente ou para trás).

4. Resultados

Com os parâmetros utilizados em um dos intervalos, o robô conseguiu manter o pêndulo estável por cerca de 0.77 segundos antes de ocorrer a queda. Este resultado mostra que a ação do controlador é relativamente fraca e insuficiente para corrigir rapidamente os desvios da posição vertical.

	Kp=200, Ki=90	Kp=300, Ki=280	Kp= 410, Ki=400	Kp=550, Ki=600	Kp=100, Ki=90
Taxa de sucesso (%)	**50%**	55.6%	90%	60%	**100%**

5. Conclusão

O controlador PID é uma ferramenta poderosa para estabilizar sistemas instáveis como o pêndulo invertido, porém exige um ajuste fino dos parâmetros. Nesta simulação, os valores baixos de Kp e Ki resultaram em uma resposta lenta, não sendo suficientes para manter o pêndulo equilibrado por muito tempo. Em testes passados com valores diferentes, mostrou que a variação sistemática desses parâmetros para alcançar melhor desempenho.