



Universidade Federal  
de São João del-Rei

**UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ**  
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEPEL**  
**COORDENAÇÃO DE ENGENHARIA ELÉTRICA - COELE**

**GABRIEL BUENO LEANDRO**

**COMPARAÇÃO ENTRE O CONTROLE PID E MPC: UM ESTUDO DE CASO COM  
*CART POLE E LUNAR LANDER***

Trabalho Final de Curso de graduação submetido  
à Coordenadoria do Curso de Engenharia Elétrica  
da Universidade Federal de São João del-Rei como  
requisito parcial para a obtenção do título de Bacharel  
em Engenharia Elétrica.

**Orientador:** Prof. Dr. Samir Ângelo Milani Martins

**São João del-Rei - MG**  
**MARÇO de 2024**

## RESUMO

Este trabalho consiste em aplicar duas técnicas de controle em ambientes da biblioteca Gymnasium. As técnicas são o Controle-Proporcional-Integral-Derivativo (PID), que mesmo após um século de existência, ainda é o controle mais utilizado na indústria, devido à sua simplicidade e robustez. Outro ponto interessante é que não é necessário um bom modelo da planta para sua implementação. O Controle Preditivo Baseado em Modelo (MPC, do inglês *Model Predictive Control*) é uma técnica mais recente, originada no meio petroquímico, e aos poucos vem ganhando espaço em diversas áreas industriais. Na prática, o MPC possui algumas vantagens quando se trata de sistemas com atraso e/ou com restrições, mas sua ideia é prever o futuro, utilizando um modelo, que para o caso retratado será um modelo paramétrico, ou seja, será utilizado o Controle Preditivo Generalizado (GPC, do inglês *Generalized Predictive Control*) para o qual a obtenção do modelo paramétrico será feita através do Pacote de identificação de sistemas SysIdentPy. Os ambientes retratados serão *Cart Pole* (pêndulo invertido) e o *Lunar Lander* (Sonda Lunar). Apesar de serem uma interface de programação de aplicações para aprendizado de reforço, o significado físico por trás de ambos é significativo, por exemplo, o pêndulo invertido pode representar um robô bípede em pé, ou até mesmo um foguete subindo, já o *Lunar Lander* é mais intuitivo, representando um pouso autônomo na lua. A implementação do PID no *Cart Pole* foi feita utilizando a sua Função de Transferência (FT). Para sua obtenção, foram utilizados conceitos físicos e experimentações. A sintonia do PID foi realizada através do Pacote de GEKKO, um pacote de otimização e aprendizado de máquina. O GPC mostrou-se com uma aplicação intuitiva no *Cart Pole*, sendo a maior dificuldade a obtenção do modelo paramétrico ARX, contornada pelo simples ato de aumentar o número de etapas a serem simuladas. No ambiente *Lunar Lander*, não foi possível obter o modelo caixa branca. A implementação do PID ocorreu de forma mais intuitiva, simplificando os *setpoints*. Os parâmetros de ajuste do PID foram determinados utilizando o método de otimização Subida de Encosta, que também foi usado para estimar os pesos do controle GPC. Aqui, a obtenção dos modelos paramétricos e definição dos *setpoints* foram bastante intuitivas para o GPC. Em ambos os ambientes, as variáveis de processo a serem controladas são inteiras, logo, no caso do PID, sua saída deve passar por uma adequação, utilizando estruturas condicionais como *if* para se ajustar às variáveis de processo. O GPC enfrentou um problema de otimização inteira, sendo utilizado o Método de Pesquisa em Grade para superar essa barreira, apesar de sempre retornar o mínimo global, é computacionalmente custoso. Ao final, ambas as técnicas mostraram-se capazes de controlar o *Cart Pole* e o *Lunar Lander*.

**Palavras-chave:** Controle Preditivo, Controle-Proporcional-Integral-Derivativo, Função de Transferência, Gymnasium, Modelo ARX.

## SUMÁRIO

<b>LISTA DE FIGURAS . . . . .</b>	<b>4</b>
<b>LISTA DE TABELAS . . . . .</b>	<b>5</b>
<b>1 INTRODUÇÃO . . . . .</b>	<b>6</b>
1.1 Considerações Iniciais . . . . .	6
1.2 Objetivos Geral e Específicos . . . . .	7
<b>2 CONCEITOS PRELIMINARES . . . . .</b>	<b>8</b>
2.1 Função de Transferência . . . . .	8
2.2 Controle PID . . . . .	9
2.3 Controle Preditivo Baseado em Modelo . . . . .	10
2.3.1 Controle Preditivo Generalizado . . . . .	12
2.4 Ambientes de Estudo . . . . .	13
2.4.1 <i>Cart Pole</i> . . . . .	13
2.4.2 <i>Lunar Lander</i> . . . . .	14
2.5 Método de pesquisa em grade . . . . .	15
2.6 Algoritmo <i>Hill Climbing</i> (subida da encosta) . . . . .	15
<b>3 METODOLOGIA . . . . .</b>	<b>15</b>
3.1 Considerações Iniciais . . . . .	15
3.2 <i>Cart Pole</i> : Implementação e Sintonia do Controlador PID . . . . .	16
3.2.1 Obtenção da Função de Transferência para o <i>Cart Pole</i> . . . . .	16
3.2.2 Sintonia do PID . . . . .	24
3.3 <i>Cart Pole</i> : Controle Preditivo Generalizado Aplicado . . . . .	26
3.3.1 Controle Preditivo Generalizado: Formulação para o <i>Cart Pole</i> . . . . .	26
3.3.2 Estimação do Modelo ARX . . . . .	27
3.4 <i>Lunar Lander</i> : Implementação e Sintonia do Controlador PID . . . . .	29
3.4.1 <i>Lunar Lander</i> : Aplicando o controle PID . . . . .	29
3.5 <i>Lunar Lander</i> : Controle Preditivo Generalizado Aplicado . . . . .	31
3.5.1 Controle Preditivo Generalizado: Formulação para o <i>Lunar Lander</i> . . . . .	31
3.5.2 Estimação do Modelo ARX . . . . .	33
3.5.3 Determinação dos Pesos da Função Custo . . . . .	34
<b>4 RESULTADOS . . . . .</b>	<b>35</b>
4.0.1 <i>Cart Pole</i> . . . . .	35
4.0.2 <i>Lunar Lander</i> . . . . .	36
<b>5 CONCLUSÕES . . . . .</b>	<b>37</b>
<b>6 PROPOSTAS DE TRABALHOS FUTUROS . . . . .</b>	<b>38</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS . . . . .</b>	<b>39</b>

## LISTA DE FIGURAS

Figura 2.1 – Diagrama de blocos da Função de Transferência. . . . .	8
Figura 2.2 – PID aplicado a uma malha de controle. . . . .	10
Figura 2.3 – Controle Preditivo Baseado em Modelo. . . . .	11
Figura 2.4 – Controle Baseado em Modelo: Representação Gráfica, . . . . .	11
Figura 2.5 – <i>Cart Pole</i> . . . . .	13
Figura 2.6 – <i>Lunar Lander</i> . . . . .	14
Figura 3.1 – Sistema de pêndulo invertido. . . . .	16
Figura 3.2 – Diagrama de corpo livre do pêndulo invertido. . . . .	16
Figura 3.3 – Deslocamento do centro de massa do pêndulo. . . . .	17
Figura 3.4 – Deslocamento rotacional do centro de gravidade do pêndulo, . . . . .	17
Figura 3.5 – Resultados da aplicação de uma força constante à direita. . . . .	19
Figura 3.6 – <i>Cart Pole</i> . . . . .	20
Figura 3.7 – Relação entre momento de inércia e comprimento, para $m = 0,5\text{kg}$ . . . . .	20
Figura 3.8 – Entrada e saída. . . . .	21
Figura 3.9 – Episódios inteiros. . . . .	22
Figura 3.10–RMSE médio devido ao fator $k$ . . . . .	23
Figura 3.11–Diagrama do sistema. . . . .	23
Figura 3.12–Entrada e saída - Validação da FT. . . . .	24
Figura 3.13–Malha de controle para o <i>Cart Pole</i> . . . . .	24
Figura 3.14–Resposta ao impulso para obtenção dos parâmetros do PID. . . . .	26
Figura 3.15–Aplicando o PRBS. . . . .	28
Figura 3.16–Maior episódio. . . . .	29
Figura 3.17– <i>Setpoints</i> para o controle PD. . . . .	30
Figura 3.18–Plano cartesiano do <i>Lunar Lander</i> , . . . . .	32
Figura 3.19– <i>Setpoints</i> para o GPC, . . . . .	32
Figura 3.20–Pesos da Função de Custo para o <i>Lunar Lander</i> . . . . .	34
Figura 4.1 – PID: <i>Cart Pole</i> GIF. . . . .	35
Figura 4.2 – GPC: <i>Cart Pole</i> GIF. . . . .	35
Figura 4.3 – PID: <i>Cart Pole</i> . . . . .	35
Figura 4.4 – GPC: <i>Cart Pole</i> . . . . .	35
Figura 4.5 – PD: <i>Lunar Lander</i> GIF. . . . .	36
Figura 4.6 – GPC: <i>Lunar Lander</i> GIF. . . . .	36
Figura 4.7 – PD: <i>Lunar Lander</i> . . . . .	36
Figura 4.8 – GPC: <i>Lunar Lander</i> . . . . .	36

## LISTA DE TABELAS

Tabela 2.1 – Entrada do <i>Cart Pole</i> . . . . .	14
Tabela 2.2 – Espaço de observação do <i>Cart Pole</i> . . . . .	14
Tabela 2.3 – Entradas do <i>Lunar Lander</i> . . . . .	14
Tabela 2.4 – Espaço de observação do <i>LunarLander</i> . . . . .	15
Tabela 3.1 – Valores dos parâmetros. . . . .	21
Tabela 3.2 – Parâmetros do PID. . . . .	25
Tabela 3.3 – Horizonte de Controle em relação ao Tempo. . . . .	29
Tabela 3.4 – Relação entre entradas e saídas. . . . .	33
Tabela 4.1 – PID x CPC - <i>Cart Pole</i> . . . . .	35
Tabela 4.2 – PID x GPC - <i>Lunar Lander</i> . . . . .	37

# 1 INTRODUÇÃO

## 1.1 Considerações Iniciais

Na atualidade tecnológica, a pesquisa em controle de sistemas dinâmicos ganha proeminência devido à busca constante por aprimoramentos no desempenho de sistemas complexos. Este estudo propõe-se a investigar duas técnicas de controle amplamente reconhecidas, o Controlador-Proporcional-Integral-Derivativo (PID) e o Controle Preditivo Baseado em Modelo (MPC). Ao final da análise, serão realizadas ponderações comparativas entre essas abordagens. Para contextualizar e aplicar essas técnicas, serão utilizadas as interfaces de programação de aplicativos (API's) *Cart Pole* (pêndulo invertido) e *Lunar Lander* (Sonda Lunar). Esses ambientes proporcionarão cenários práticos para avaliação e comparação das técnicas de controle em questão.

O engenheiro russo Nikolai (Nicolas) Fyodorovich Minorsky (1885-1970) é reconhecido como o pioneiro do controlador de três termos, desempenhando um papel crucial na aprimorada navegação das embarcações da marinha norte-americana. Esses três termos correspondem às componentes proporcional, integral e derivativa, formando o que é conhecido como controlador PID [1]. Apesar de ter mais de um século de existência, o controlador PID é atualmente o mais utilizado nas malhas fechadas industriais devido à sua robustez e facilidade de implementação [2]. Por esse motivo, continua sendo amplamente explorado no meio acadêmico.

O Controle Preditivo Baseado em Modelo (MPC) é uma estratégia que se baseia na previsão do comportamento futuro por meio de um modelo do sistema. Neste contexto, foi adotado a abordagem do Controle Preditivo Generalizado (GPC), que é atualmente a técnica mais amplamente reconhecida no âmbito do MPC, originando-se do desenvolvimento do controle de mínima variância [3], o GPC é uma técnica avançada que emprega modelos paramétricos sendo projetado a partir de modelos auto-regressivos, média móvel e com sinal exógeno do sistema (ARMAX, do inglês *Auto-Regressive Moving Average with Exogenous Inputs*), permitindo antecipar o comportamento futuro e otimizar as ações de controle. No presente trabalho, serão aplicadas técnicas de identificação de sistemas utilizando o pacote SysIdentPy [4] para estimar o modelo paramétrico. A ISA (*International Society of Automation*, em inglês) considera que o controle preditivo é uma ferramenta importante capaz de diferenciar entre um bom e um excelente Engenheiro de Controle.

O desafio apresentado pelo pêndulo invertido é um clássico na engenharia de controle, estando presente em alguns livros nessa área, dentre eles, Ogata [5] e Castrucci [6]. Este sistema é frequentemente utilizado como um exemplo didático devido à sua natureza instável e não linear, que pode ser linearizada por meio de aproximações. O pêndulo invertido encontra aplicação prática em diversas áreas, incluindo a estabilidade de robôs ambulantes, controle de atitude de veículos lançadores, sistemas de balanceamento em robótica móvel e até mesmo no transporte de objetos por meio de drones, entre outras aplicações [7].

Para explorar e simular o comportamento do sistema de pêndulo invertido, será utilizado

o Ambiente *Cart Pole* da Gymnasium. Além disso, o desafio do pouso lunar será abordado por meio da simulação no ambiente *Lunar Lander*, onde uma sonda deve realizar um pouso preciso. Esse problema é notável por sua complexidade, uma vez que envolve um sistema MIMO (Múltiplas Entradas e Múltiplas Saídas), acrescentando uma camada adicional de desafio ao controle do sistema.

## 1.2 Objetivos Geral e Específicos

O trabalho tem como objetivo aplicar conceitos de controle clássico e estudar o controle preditivo baseado em modelo (MPC), um tema não abordado na grade curricular. Utilizando API's que simulam sistemas complexos, a intenção é desenvolver a capacidade de aplicar na prática técnicas de controle, integrando conhecimentos tradicionais com abordagens mais contemporâneas.

Em síntese, os objetivos específicos, são:

- Derivar a FT específica do sistema *Cart Pole*, proporcionando uma compreensão aprofundada de sua dinâmica;
- Refinar o desempenho do sistema *Cart Pole* por meio da sintonia precisa do controlador PID, visando estabilidade e resposta otimizada;
- Identificar e elaborar um modelo paramétrico abrangente para descrever o comportamento da saída do *Cart Pole* (representada pelo ângulo  $\theta$ ) em relação à variável de entrada, com o intuito de fornecer uma representação sólida e precisa;
- Explorar a teoria do Controle Preditivo Baseado em Modelo (MPC), com foco especial na abordagem Controle Preditivo Generalizado (GPC), aprofundando a compreensão teórica para aplicações práticas;
- Implementar de maneira eficaz o GPC para o sistema *Cart Pole*, integrando os conceitos teóricos com a prática da engenharia de controle;
- Desenvolver um modelo intuitivo e abrangente para o sistema *Lunar Lander*, proporcionando uma base sólida para futuras análises e otimizações;
- Sintonizar o controlador PID do sistema *Lunar Lander* com precisão, visando aprimorar seu comportamento dinâmico e garantir estabilidade;
- Identificar modelos paramétricos que descrevam de forma precisa o comportamento da posição horizontal ( $p_x$ ), posição vertical ( $p_y$ ), ângulo ( $\theta$ ) e velocidade linear na direção vertical ( $v_y$ ) em resposta às entradas do *Lunar Lander*;

- Implementar de forma coerente o GPC para o sistema *Lunar Lander*, aplicando os princípios teóricos na prática de controle avançado;
- Conduzir uma análise comparativa abrangente entre os controladores PID e GPC, avaliando seus desempenhos de forma holística e identificando possíveis áreas de melhoria.

## 2 CONCEITOS PRELIMINARES

Estudos comparativos entre controladores PID e MPC, como discutido por Prata et al. em 2020 [8], destacaram a eficácia do MPC, formulado a partir do espaço de estados, evidenciando menor erro relativo e maior velocidade na estabilização dos sinais de saída em comparação com o PID, que foi ajustado usando *auto-tuning* do MATLAB®. Sugere-se que um PID melhor parametrizado poderia se equiparar ao desempenho do MPC, especialmente para plantas com pouco ou nenhum atraso.

O estudo de Taketa et al. em 2018 [9] aborda a aplicação do GPC em sistemas mecânicos, destacando benefícios como a atenuação de vibrações e menor erro no rastreamento de posição. A pesquisa, abrangendo aspectos teóricos, experimentais e simulados, ressalta a eficácia do GPC, mesmo para controle de vibrações, o qual é menos comum, evidenciando sua capacidade de lidar efetivamente com esse desafio.

A proposta deste trabalho é combinar as ideias de Prata et al. em 2020 [8] e Taketa et al. em 2018 [9]. Serão comparados os controladores PID e MPC, como em Prata et al. em 2020 [8], em sistemas menos comuns, conforme discutido em Taketa et al. em 2018 [9], como nas API's, como o *Cart Pole* e *Lunar Lander*, usualmente resolvidos por algoritmos de aprendizado por reforço.

### 2.1 Função de Transferência

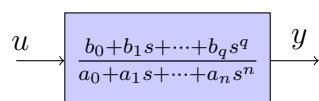
Os processos dinâmicos são comumente descritos por equações diferenciais. Ao conhecer as equações diferenciais que descrevem o sistema, a função de transferência pode ser obtida ao aplicar a transformada de Laplace com condições iniciais nulas.

A função de transferência de um sistema é, por definição, a transformada de Laplace da resposta ao impulso, descrevendo como uma entrada é dinamicamente "transferida" para a saída do sistema [10]. A FT é matematicamente expressa pela Equação 2.1:

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_0 + b_1 s + \dots + b_q s^q}{a_0 + a_1 s + \dots + a_n s^n}, \quad (2.1)$$

além da representação matemática, pode também ser expressa por meio de diagrama de blocos:

Figura 2.1 – Diagrama de blocos da Função de Transferência.



Fonte: Autor, adaptado de [10].

Os zeros de  $H(s)$  são os zeros do polinômio  $N(s)$ . De maneira semelhante, os polos são as raízes do polinômio  $D(s)$ .

## 2.2 Controle PID

Um controlador PID é composto por três elementos ajustáveis, os quais são adaptados com base na discrepância entre um ponto de ajuste definido ( $r(t)$ ) e uma variável de processo medida ( $y_m(t)$ ) em malha fechada:

$$e(t) = r(t) - y_m(t). \quad (2.2)$$

A saída de um controlador PID é determinada pela soma ponderada dos termos proporcional, integral e derivativo, representados por ( $K_P$ ,  $K_I$ , e  $K_D$ ), respectivamente. Essas constantes ajustáveis podem ser modificadas para otimizar o desempenho do controlador:

$$g_c(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(t)dt + K_D \cdot \frac{de(t)}{dt}, \quad (2.3)$$

a Equação 2.3 será reescrita no padrão ISA:

$$g(t) = K_C \cdot e(t) + \frac{K_C}{\tau_I} \cdot \int_0^t e(t)dt + K_C \cdot \tau_D \cdot \frac{de(t)}{dt}, \quad (2.4)$$

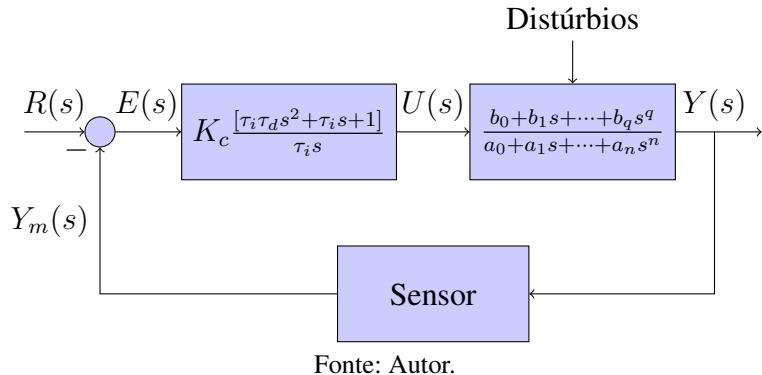
observe que  $K_P$  é igual a  $K_C$ ,  $K_I$  é equivalente a  $\frac{K_C}{\tau_I}$ , e  $K_D$  é dado por  $K_C \cdot \tau_D$ . Portanto, no domínio de Laplace, a expressão para um controlador PID pode ser representada de acordo com a Equação 2.5:

$$G(s) = K_c \frac{[\tau_i \tau_d s^2 + \tau_i s + 1]}{\tau_i s}, \quad (2.5)$$

note que a FT que representa o Controlador Proporcional-Integral-Derivativo (PID) na Equação 2.5 é uma função imprópria, uma vez que apresenta dois zeros e apenas um polo. Essa característica implica na impossibilidade de encontrar sistemas físicos que se enquadrem nesse modelo [10], pois tal condição resulta em um sistema passa altas para frequências  $\omega \rightarrow \infty$ . No entanto, ao determinar a sintonia do PID, é possível empregar a FT apresentada na Equação 2.5. Os parâmetros obtidos podem então ser aplicados em um controlador implementável, sendo essa abordagem geralmente eficaz.

Abaixo, tem a representação ID aplicado a uma malha de controle em diagramas de blocos:

Figura 2.2 – PID aplicado a uma malha de controle.



Fonte: Autor.

Para ajustar os parâmetros do controlador PID, ou seja, determinar os valores de  $K_P$ ,  $K_I$  e  $K_D$ , será utilizado o pacote GEKKO[11]. Este pacote, uma ferramenta em Python dedicada ao aprendizado de máquina e otimização de inteiros mistos, assim como a equações algébricas diferenciais.

Na Equação 2.4, foi fixado os valores de  $\tau_i$  e  $\tau_d$ , mantendo apenas  $K_c$  como variável independente. Em seguida, o pacote GEKKO calculará o valor correspondente de  $K_C$  por meio de um procedimento de minimização. A metodologia adotada visa minimizar o erro da variável de processo em relação ao valor desejado (conhecido como *setpoint*), expresso pela equação  $e(t) = r(t) - y_m(t)$ . Neste contexto,  $e(t)$  representa o erro,  $r(t)$  é o valor de referência e  $y_m(t)$  é o valor medido da variável de processo.

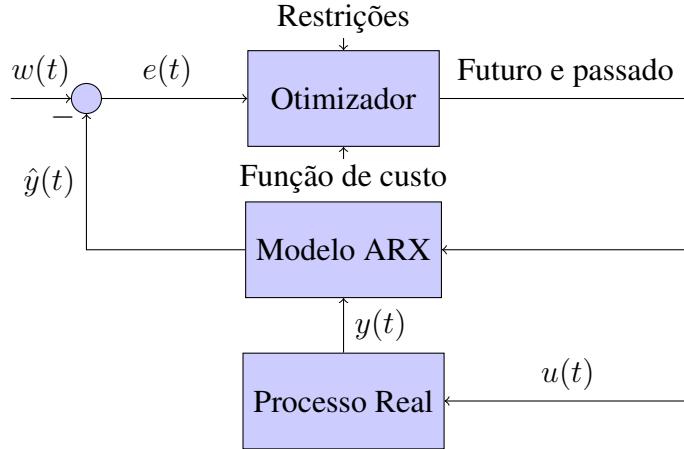
### 2.3 Controle Preditivo Baseado em Modelo

O controle preditivo baseado em modelo (MPC) é uma estratégia de controle com as seguintes características:

- Utiliza um modelo explícito do processo para prever sua saída em um horizonte finito;
- Calcula as ações de controle futuras minimizando uma função objetivo específica;
- Apresenta um horizonte de previsão deslizante, ou seja, a cada período de amostragem, o horizonte é deslocado um passo à frente. O sinal de controle no instante atual é aplicado ao processo, desconsiderando o restante do horizonte de controle.

Assim, a representação do MPC por meio de um diagrama de blocos pode ser visualizada na Figura 2.3:

Figura 2.3 – Controle Preditivo Baseado em Modelo.



Fonte: Autor, adaptado de [12].

Trajetória de referência ( $w(t)$ ): representa o comportamento do sinal desejado para a saída futura. É o conhecimento prévio desta trajetória que garante ao controlador uma característica antecipativa;

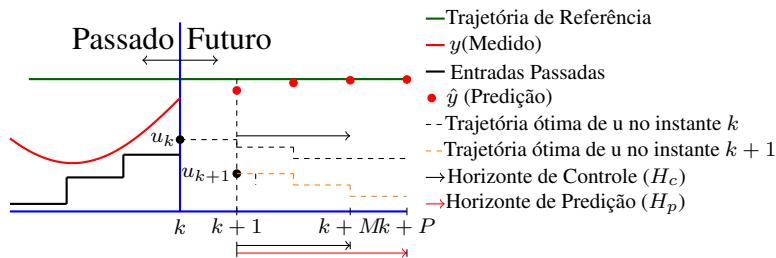
Modelo do processo: deve ser capaz de representar com precisão suficiente o comportamento dinâmico do sistema. Neste trabalho, o modelo ARX será empregado, obtido a partir do pacote de identificação de sistemas SysIdentPy;

Otimizador: minimiza a função custo a cada período de amostragem, de forma a obter uma ação de controle que garanta um desempenho adequado ao sistema;

Processo real: Os ambientes escolhidos para este estudo serão o *Cart Pole* (pêndulo invertido) e o *Lunar Lander* (Sonda Lunar) do pacote Gymnasium.

De forma resumida, o MPC busca, por meio de um modelo, estimar a saída futura de um sistema. Essa saída pode ser fragmentada em dois termos, sendo  $\hat{y} = f_p(u_{\text{Passado}}) + f_t(\Delta u_{\text{Futuro}}) = y_{\text{Liv}} + y_{\text{For}}$ , onde  $y_{\text{Liv}}$  depende das condições iniciais na amostra  $k$ . Logo, essa parcela não pode ser alterada. O que pode ser feito é alterar  $y_{\text{For}}$  por meio da entrada  $\Delta u_{\text{Futuro}}$ , de modo que  $\hat{y}$  atinja a trajetória de referência, conforme a Figura 2.4:

Figura 2.4 – Controle Baseado em Modelo: Representação Gráfica,



Fonte: Autor, adaptado de [13].

note que se  $y_{\text{Liv}}$  conseguir perseguir a referência, então  $\Delta u_{\text{Futuro}} = 0$ , caso contrário, é necessário incrementar  $\Delta u_{\text{Futuro}}$  minimizando uma Função Custo, normalmente considerando critérios como o erro em relação à trajetória de referência e o esforço de controle.

O MPC trabalha com horizonte deslizante, onde a cada amostra, o horizonte avança. Outros termos importantes são o horizonte de controle ( $H_c$ ) e horizonte de previsão ( $H_p$ ), onde  $H_c \leq H_p$

(causalidade). Alguns pontos importantes são: quanto maior o horizonte de controle, mais complicado é do ponto de vista computacional, pois em casos em que há restrições de entrada e/ou saída,  $\Delta u_{\text{Futuro}}$  não pode ser obtido de forma algébrica (caso sem restrições), mas sim utilizando algoritmos de otimização, normalmente o quadprog. Por outro lado, se o horizonte de controle for muito pequeno, significa que o controlador está tomando decisões de controle de curto prazo, considerando apenas um número limitado de passos à frente.

### 2.3.1 Controle Preditivo Generalizado

O Controle Preditivo Generalizado (GPC) foi proposto inicialmente em 1987 [14, 15] e tornou-se uma das estratégias mais amplamente adotadas tanto no meio acadêmico quanto na indústria [12]. Neste contexto, será considerado um modelo *Single-Input Single-Output* (SISO) e linear, representado por um modelo ARMAX (CARMA):

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + C(z^{-1})\nu(k), \quad (2.6)$$

como de praxe,  $y(k)$  e  $u(k)$  representam a saída e a entrada, respectivamente,  $z$  é operador atraso, logo  $A(z^{-1})$ ,  $B(z^{-1})$ , e  $C(z^{-1})$  são definidos como:

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{na} z^{-na} \\ B(z^{-1}) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{nb} z^{-nb} \\ C(z^{-1}) &= 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{nc} z^{-nc}. \end{aligned} \quad (2.7)$$

esse modelo pode ser usado para fazer previsões. Para facilitar, será considerado  $C(q^{-1}) = 1$  por ser uma característica estocástica desconhecida. Dessa forma, todo erro (modelagem, perturbação, ruído) entre o processo real e o modelo será atribuído a  $\epsilon$ , o modelo passa a ser ARX (do inglês, *AutoRegressive with eXogenous inputs*):

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \nu(k), \quad (2.8)$$

considerando que  $\nu$  seja um ruído branco com média zero e não correlacionado, a melhor previsão para a saída de uma variável com comportamento desconhecido é a esperança matemática. Esta, por sua vez, possui o mesmo valor da média, ou seja, 0:

$$A(z^{-1})y(k) = B(z^{-1})u(k-1). \quad (2.9)$$

A proposta é empregar este modelo para efetuar previsões e minimizar uma função de custo. A função custo ( $J$ ) é definida como a soma dos erros entre a previsão da saída ( $\hat{y}$ ) e a referência ( $w$ ), com a opção de penalizar ou não o esforço de controle ( $\Delta u$ ). Considere a função custo:

$$J(k) = \sum_{j=d}^{H_p} \sqrt{\delta[\hat{y}(j+k|k) - w(j+k)]^2} + \sum_{j=1}^{H_c} \lambda \Delta u(j+k-1), \quad (2.10)$$

onde  $H_p$  e  $d$  são os horizontes de previsão máximo e mínimo, marcam os instantes onde é desejável que a saída siga a referência  $w$ . E  $H_c$  é o horizonte de controle,  $\delta$  e  $\lambda$  são ponderações e do esforço de controle respectivamente e geralmente são escolhidos constantes ou exponenciais ao longo do horizonte.

A abordagem fundamental do GPC consiste em correlacionar os termos em negrito ( $\hat{y}$  e  $\Delta u$ ) por meio de um modelo paramétrico. Deve-se encontrar uma maneira para que ( $\hat{y}$ ) avance até o horizonte de previsão em função de ( $\Delta u$ ), que, por sua vez, avança até o horizonte de controle, separando a resposta forçada da resposta livre do sistema. A Equação 2.11 que descreve essa relação:

$$\hat{y} = \overbrace{G\Delta u}^{\text{Forçada}} + \overbrace{f}^{\text{Forçada}}, \quad (2.11)$$

o processo de derivar essa equação a partir de um modelo paramétrico pode ser encontrado na literatura, como no livro *Model Predictive Control* de Camacho e Bordons [12].

## 2.4 Ambientes de Estudo

O estudo utiliza os ambientes de simulação *Cart Pole* e *Lunar Lander* do Gymnasium, uma biblioteca de Python para aprendizado por reforço. O Gymnasium oferece uma API padronizada para facilitar a interação entre algoritmos e ambientes de aprendizado, sendo uma bifurcação da biblioteca Gym da OpenAI.

### 2.4.1 *Cart Pole*

Neste cenário, o pêndulo é posicionado verticalmente sobre o carrinho, tendo como meta equilibrar o pêndulo ao aplicar forças nas direções esquerda e direita no carro [16]. O *Cart Pole* é um problema clássico que envolve um pêndulo invertido:

Figura 2.5 – *Cart Pole*.

Fonte: Autor.

A interação do *Cart Pole* com algoritmo se dá por meio de sua entrada:

Tabela 2.1 – Entrada do *Cart Pole*.

Número	Ação
0	Empurre o carrinho para à esquerda
1	Empurre o carrinho para à direita

Fonte: Autor, adaptado de [17].

O espaço de observação constitui a saída, e essas informações são utilizadas para a aplicação de ações futuras:

Tabela 2.2 – Espaço de observação do *Cart Pole*.

Número	Observação	Mínimo	Máximo
0	Posição do carrinho	-4,8	4,8
1	Velocidade do carrinho	-Inf	Inf
2	Ângulo do polo	-0,418 rad (-24°)	0,418 rad (24°)
3	Velocidade angular do polo	-Inf	Inf

Fonte: Autor, adaptado de [17].

O objetivo é manter o poste ereto o máximo de tempo possível, com uma recompensa atribuída a cada passo, incluindo a etapa final, limitada a 500 pontos.

#### 2.4.2 *Lunar Lander*

Neste ambiente, a tarefa consiste em realizar um pouso suave de uma sonda espacial na superfície lunar, lidando com desafios como a gravidade lunar, obstáculos e utilizando informações cruciais, como a posição, ângulo e velocidade da nave:

Figura 2.6 – *Lunar Lander*.

Fonte: Autor.

A interação do LunaLander com algoritmo se dá por meio de suas entradas:

Tabela 2.3 – Entradas do *Lunar Lander*.

Número	Ação
0	Não fazer nada
1	Motor de orientação à esquerda da sonda
2	Acionamento do motor principal
3	Motor de orientação à direita da sonda

Fonte: Autor, adaptado de [18].

espaço de observação do *Lunar Lander* é composto pelos itens da Tabela 3.2:

Tabela 2.4 – Espaço de observação do LunarLander.

Número	Observação	Mínimo	Máximo
0	Coordenada x	-1,5	1,5
1	Coordenada y	-1,5	1,5
2	Velocidade linear em x	-5	5
3	Velocidade linear em y	-5	5
4	Ângulo	$-\pi$ ( $-360^\circ$ )	$\pi$ ( $360^\circ$ )
5	Velocidade angular	-5	5
6	Contato pé esquerdo	0	1
7	Contato pé direito	0	1

Fonte: Autor, adaptado de [18].

Um episódio é considerado bem-sucedido se alcançar pelo menos 200 pontos.

## 2.5 Método de pesquisa em grade

Como discutido anteriormente, os ambientes têm entradas inteiras, o que sugere a minimização de uma função custo por meio de otimização inteira. Dada a quantidade limitada de entradas, o Método de Pesquisa em Grade é viável. Apesar de uma complexidade  $O(n!)$ , indicando crescimento fatorial do tempo de execução devido a  $n$ , sua aplicação é factível para horizontes de controle mais curtos. Este método, que examina todas as combinações possíveis de variáveis, evitando convergir erroneamente para mínimos/máximos locais ao buscar o valor mínimo/máximo de uma função.

## 2.6 Algoritmo *Hill Climbing* (subida da encosta)

O Método de Subida de Encosta é um algoritmo clássico para otimização [19], mostrando-se altamente eficaz na identificação de máximos ou mínimos locais. No processo desse algoritmo, inicia-se a partir de um ponto aleatório  $X$  e realiza-se sua avaliação. Em seguida, ocorre o deslocamento do ponto original  $X$  para um novo ponto vizinho, designado como  $X'$ . Se o novo ponto  $X'$  representar uma solução superior à do ponto anterior, permanece-se nele e o processo é repetido. Contudo, se for inferior, retorna-se ao ponto inicial  $X$  e tenta-se explorar outro vizinho. Uma das principais "restrições" do Método de Subida de Encosta é sua incapacidade de aceitar valores negativos, em outras palavras, ele sempre busca pontos vizinhos no espaço de solução que possam aprimorar seu estado atual. Caso não encontre tal aprimoramento, a execução é interrompida.

# 3 METODOLOGIA

## 3.1 Considerações Iniciais

A elaboração da metodologia incorporará a totalidade do conteúdo abordado na Seção de Conceitos Preliminares, englobando também a modelagem de sistemas dinâmicos por meio de

equações diferenciais e modelos ARX. Durante foram utilizados pacotes na linguagem Python, destacando-se o Gymnasium, SysIdentPy e GEKKO.

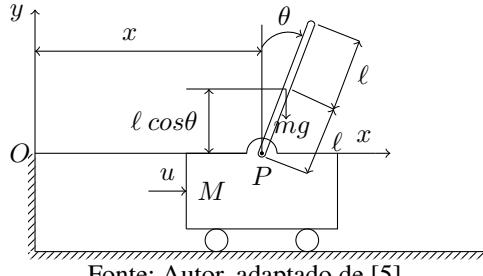
### 3.2 *Cart Pole*: Implementação e Sintonia do Controlador PID

Nesta seção, será abordada a implementação do controlador PID para o *Cart Pole*.

#### 3.2.1 Obtenção da Função de Transferência para o *Cart Pole*

O primeiro ambiente a ser abordado é o *Cart Pole*, o primeiro passo é obter a sua FT, conforme discutido por Ogata [5]. Este passo é crucial para a subsequente implementação do controlador PID, visando controlar o sistema *Cart Pole* [20]. O sistema em questão refere-se a um pêndulo invertido montado em um carro motorizado (*Cart Pole*):

Figura 3.1 – Sistema de pêndulo invertido.

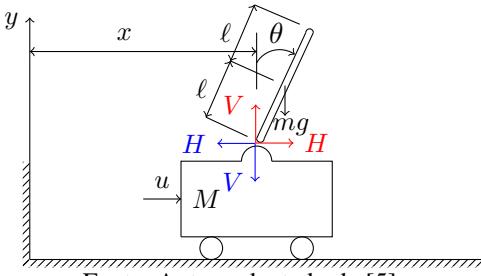


Fonte: Autor, adaptado de [5].

O primeiro passo é modelar o sistema de pêndulo invertido, a ideia é encontrar como a entrada é dinamicamente transferida para a saída, encontrando a FT do sistema, que para o sistema pêndulo invertido a entrada é a força aplicada na lateral ( $u$ ) e a saída o ângulo do pêndulo ( $\theta$ ), conforme a Figura 2.1.

Para deduzir as equações de movimento do sistema, será realizado o estudo do diagrama de corpo livre do sistema de pêndulo invertido, conforme ilustrado na Figura 3.2:

Figura 3.2 – Diagrama de corpo livre do pêndulo invertido.



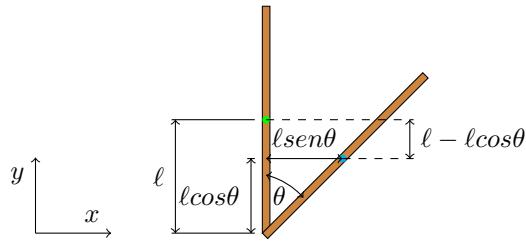
Fonte: Autor, adaptado de [5].

O deslocamento horizontal do carro é descrito pela Equação 3.1:

$$M \frac{d^2x}{dt^2} = u - H. \quad (3.1)$$

Já o deslocamento do centro de massa do pêndulo ( $x_c$ ) é ilustrado na Figura 3.3:

Figura 3.3 – Deslocamento do centro de massa do pêndulo.



Fonte: Autor, adaptado de [5].

Na Figura 3.3, observa-se que a coordenada  $x_c$  do carro é obtida pela soma do deslocamento horizontal do centro de massa da coordenada que se encontra, representado como  $x_c = x + \ell \sin \theta$ , e tomando a segunda derivada de  $x_c$  efetuando o balanço das forças horizontais, encontra-se a Equação 3.2:

$$m \frac{d^2}{dt^2}(x + \ell \sin \theta) = H. \quad (3.2)$$

O  $y_c$  pode ser encontrado subtraindo  $\ell \cos \theta$  de  $\ell$ , agora basta tomar a sua segunda derivada e igual a resultante de forças verticais ( $mg - V$ ):

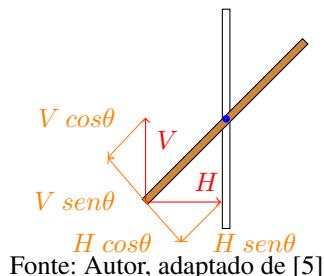
$$m \frac{d^2}{dt^2}(\ell - \ell \cos \theta) = mg - V, \quad (3.3)$$

no entanto, como  $\ell$  é uma constante e sua segunda derivada é zero, a Equação 3.3 pode ser reformulada conforme a Equação 3.4:

$$m \frac{d^2}{dt^2}(-\ell \cos \theta) = mg - V \implies m \frac{d^2}{dt^2}(\ell \cos \theta) = V - mg. \quad (3.4)$$

O próximo passo consiste em determinar o deslocamento rotacional:

Figura 3.4 – Deslocamento rotacional do centro de gravidade do pêndulo,



Fonte: Autor, adaptado de [5].

o movimento rotacional da haste do pêndulo ao redor de seu centro de gravidade é descrito pela Equação 3.5:

$$I \frac{d^2\theta}{dt^2} = V \ell \sin \theta - H \ell \cos \theta, \quad (3.5)$$

onde  $I$  representa o momento de inércia da haste em relação ao centro de gravidade.

As Equações 3.1, 3.2, 3.4 e 3.5 descrevem o comportamento dinâmico do pêndulo invertido. No entanto, essas equações envolvem funções trigonométricas, como seno e cosseno. Para linearizá-las, será adotado um valor muito pequeno para  $\theta$ . A ideia é manter o sistema estável, permitindo assim aproximações úteis, tais como:  $\cos \theta \approx 1$  e  $\sin \theta \approx \theta$ . Portanto:

$$I \frac{d^2\theta}{dt^2} = V\ell\theta - H\ell, \quad (3.6)$$

$$M \frac{d^2x}{dt^2} = u - H, \quad (3.7)$$

$$m \frac{d^2}{dt^2}(x + \ell\theta) = H, \quad (3.8)$$

$$m \frac{d^2}{dt^2}(\ell) = V - mg \implies 0 = V - mg. \quad (3.9)$$

Note que  $H$  e  $V$  são desconhecidos, mas a priori não se trata de um problema, pois pela Equação 3.8 é possível expressar  $H$ . Pela Equação 3.9, tem-se que  $V = mg$ , levando ao próximo passo, substituir esses valores na Equação 3.6, obtendo:

$$I \frac{d^2\theta}{dt^2} = mg\theta\ell - m \frac{d^2}{dt^2}(x + \ell\theta)\ell, \quad (3.10)$$

ao igualar a zero, tem-se:

$$(I + m\ell^2) \frac{d^2\theta}{dt^2} + m\ell \frac{d^2x}{dt^2} - mg\ell\theta = 0, \quad (3.11)$$

substituindo a Equação 3.8 na Equação 3.7, obtém-se:

$$M \frac{d^2x}{dt^2} = u - m \frac{d^2}{dt^2}(x + \ell\theta), \quad (3.12)$$

reorganizando os termos:

$$M \frac{d^2x}{dt^2} + m \frac{d^2}{dt^2}(x + \ell\theta) = u. \quad (3.13)$$

Agora, será necessário obter a função de transferência do sistema. Logo, naturalmente, é preciso transformar as Equações 3.11 e 3.13 para o domínio de Laplace com condições iniciais nulas, conforme segue:

$$(I + m\ell^2)s^2\Theta(s) + m\ell s^2X(s) - mg\ell\Theta(s) = 0, \quad (3.14)$$

$$(M + m)s^2X(s) + m\ell s^2\Theta(s) = U(s). \quad (3.15)$$

A ideia volta-se para obter a relação da saída ( $\Theta(s)$ ) em relação à entrada ( $U(s)$ ). Portanto, o  $X(s)$  deve ser eliminado. Considerando a Equação 3.14, tem-se:

$$((I + m\ell^2)s^2 - mg\ell)\Theta(s) + m\ell s^2X(s) = 0, \quad (3.16)$$

isolando  $X(s)$ , a Equação 3.16 pode ser reescrita como:

$$X(s) = -\frac{\{(I + m\ell^2)s^2 - mg\ell\}}{m\ell s^2} \Theta(s), \quad (3.17)$$

substituindo a Equação 3.17 na Equação 3.15, chega-se a:

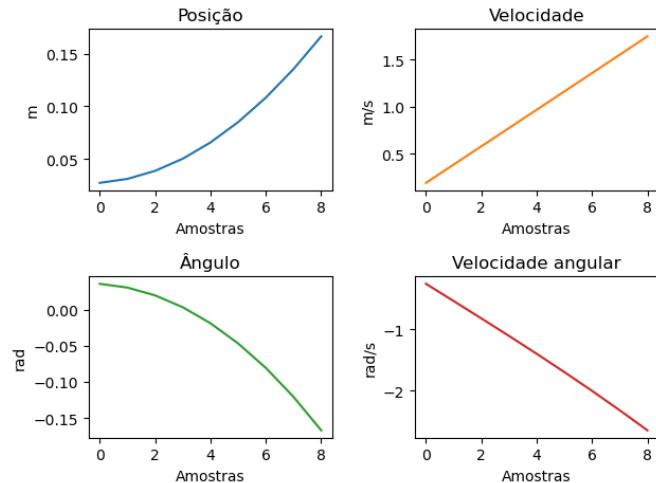
$$\begin{aligned} -(M+m)s^2 \frac{\{(I + m\ell^2)s^2 - mg\ell\}}{m\ell s^2} \Theta(s) + m\ell s^2 \Theta(s) &= U(s), \\ \frac{[m^2\ell^2 s^2 - (M+m)\{(I + m\ell^2)s^2 - mg\ell\}]}{m\ell} \Theta(s) &= U(s). \end{aligned} \quad (3.18)$$

Por fim, tem-se a FT:

$$\frac{\Theta(s)}{U(s)} = \frac{m\ell}{(m^2\ell^2 - (M+m)(I + m\ell^2))s^2 + (M+m)mg\ell}. \quad (3.19)$$

O equacionamento já está pronto, sendo necessário agora saber quais são os valores de massa, comprimento e momento de inércia. Na verdade, não é preciso conhecer o valor exato de cada uma dessas grandezas, pode-se aproxima-las com simulações e um pouco de física. Portanto, será aplicada uma força à direita (aplicar 1 a entrada, conforme a Tabela 4.1) e será traçado como o sistema se comporta. Como o ambiente fornece os valores de velocidade e velocidade angular, é possível plotá-los para encontrar alguns valores para as variáveis supracitadas:

Figura 3.5 – Resultados da aplicação de uma força constante à direita.



Fonte: Autor, adaptado de [20].

Não se pode realmente obter muitas informações úteis da posição ou ângulo, mas o fato de que a velocidade e a velocidade angular serem lineares é muito importante. Isso sugere que a simulação é fisicamente precisa, pois uma força constante está produzindo uma aceleração constante tanto linear quanto angularmente, conforme descrito pela Segunda Lei de Newton.

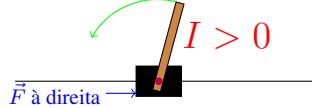
Agora, serão trabalhadas as acelerações linear e angular, sendo a inclinação da velocidade (uma reta linear) em seus respectivos gráficos. Para isso, será utilizado o *linregress* do pacote *Scipy*. Os resultados encontrados foram:

$$\ddot{x} = 0,19524 \frac{m}{s^2} \quad \text{e} \quad \ddot{\theta} = -0,29775 \frac{rad}{s^2}, \quad (3.20)$$

um fato interessante, é que esses valores permanecem constantes, independentemente do número de simulações, sugerindo que o ambiente *Cart Pole* possui um embasamento físico subjacente.

Considere a Figura 3.6:

Figura 3.6 – *Cart Pole*.



Fonte: Autor, adaptado de [17].

O momento de inércia deve ser sempre positivo, conforme a Figura 3.6. Para tal análise, será utilizada a Equação 3.21:

$$(I + m\ell^2) \frac{d^2\theta}{dt^2} + m\ell \frac{d^2x}{dt^2} - mg\ell\theta = 0, \quad (3.21)$$

como o intuito é mantê-lo equilibrado,  $\theta \approx 0$ , abrindo a possibilidade de reescrever a Equação 3.21 da seguinte forma:

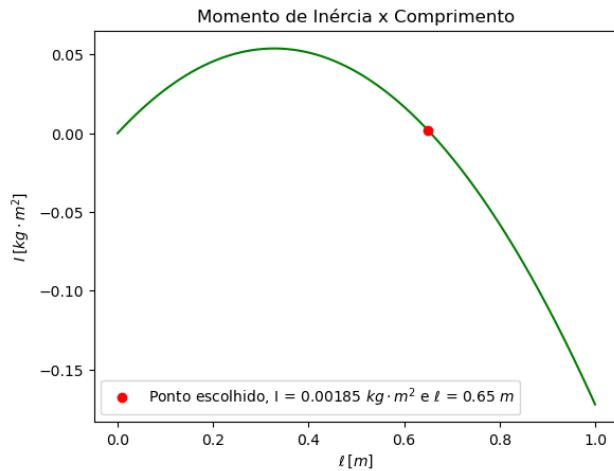
$$(I + m\ell^2) \frac{d^2\theta}{dt^2} = -m\ell \frac{d^2x}{dt^2}, \quad (3.22)$$

onde  $\frac{d^2\theta}{dt^2} = -0,29775 \frac{\text{rad}}{\text{s}^2}$  e  $\frac{d^2x}{dt^2} = 0,19524 \frac{\text{m}}{\text{s}^2}$ , o valor adotado para  $m$  será  $0,5\text{kg}$ . Logo:

$$I = \frac{0,0976\ell - 0,1488\ell^2}{0,29775}, \quad (3.23)$$

a Equação 3.23 expressa  $I$  em função de  $\ell$ . Ao variar  $\ell$  de 0 a 1m, obtém-se o Gráfico:

Figura 3.7 – Relação entre momento de inércia e comprimento, para  $m = 0,5\text{kg}$ .



Fonte: Autor.

Note que o valor escolhido de  $\ell$  apresenta  $I$  positivo.

Agora, será estimada a massa do carro ( $M$ ) a partir da Equação 3.24:

$$(M + m) \frac{d^2x}{dt^2} + m\ell \frac{d^2\theta}{dt^2} = u, \quad (3.24)$$

substituindo os valores conhecidos, como  $m = 0,5\text{kg}$ ,  $u = 1\text{N}$  e  $\ell = 0,65\text{m}$ , obtém-se:

$$M = \frac{u - m\ell\ddot{\theta} - m\ddot{x}}{\ddot{x}} = \frac{1 + 0,5 \cdot 0,65 \cdot 0,29775 - 0,5 \cdot 0,19524}{0,19524} = 5,11754 \text{ kg}. \quad (3.25)$$

Os resultados encontrados/fixados foram:

Tabela 3.1 – Valores dos parâmetros.

$M[\text{kg}]$	$m[\text{kg}]$	$\ell[m]$	$I[\text{kg} \cdot \text{m}^2]$
5,11754	0,5	0,65	$1,8537 \times 10^{-3}$

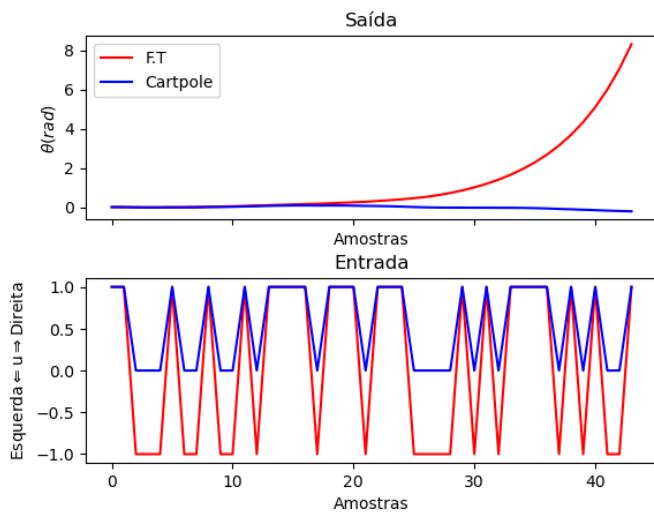
Fonte: Autor.

Ao substituí-los na Função de transferência da Equação 3.19, chega-se:

$$\frac{\Theta(s)}{U(s)} = \frac{0,325}{-1,0914s^2 + 17,9101}. \quad (3.26)$$

A função de transferência faz sentido a priori, pois uma força para a direita deve resultar em um ângulo  $\theta$  negativo. No entanto, é necessário validar a função de transferência com o ambiente *Cart Pole* do Gymnasium. Portanto, será aplicada a mesma entrada tanto no ambiente *Cart Pole* quanto na função de transferência, considerando apenas um episódio (até o sistema perder a estabilidade). A cada episódio, o ambiente é reiniciado. Para adequar a entrada, será empregada a seguinte formulação: no *Cart Pole*, a entrada 1 significa ir para a direita, enquanto 0 é o comando para ir à esquerda. Para a função de transferência, o movimento para a direita também será representado por 1, porém, para a esquerda, será representado por -1. Assim, tem-se:

Figura 3.8 – Entrada e saída.



Fonte: Autor.

A partir da Figura 3.2.1, nota-se que a FT não conseguiu representar o sistema adequadamente. A FT responde de forma muito lenta aos estímulos de entrada. Portanto, é necessário ajustá-la para obter respostas mais abruptas. Ao observar os passos anteriores, a aceleração considerada levou em conta 8 amostras, com um intervalo de 1s entre amostras, para realmente encontrar a inclinação. No entanto, devido à natureza computacional, o tempo é significativamente menor.

Assim, as acelerações em  $x$  e angular serão multiplicadas por um fator ( $k$ ) até que a FT alcance um desempenho satisfatório. Dessa forma, será retornado a equação da inércia:

$$(I + m\ell^2) \frac{d^2\theta}{dt^2} = -m\ell \frac{d^2x}{dt^2}, \quad (3.27)$$

logo, o momento de inércia independe do valor de  $k$ . Portanto, os valores de  $I$  e  $\ell$  permanecem inalterados, sendo  $1,8537 \times 10^{-3} N \cdot m$  e  $0,65m$ , respectivamente.

Em teoria, à medida que a aceleração aumenta e a força permanece constante, logo a massa total deve diminuir. Em termos matemáticos, isso é evidenciado pela Equação 3.28, onde  $m$  é mantido constante em  $0,5kg$  (Massa do pêndulo), resultando em uma diminuição de  $M$  (Massa do carro), conforme descrito a seguir:

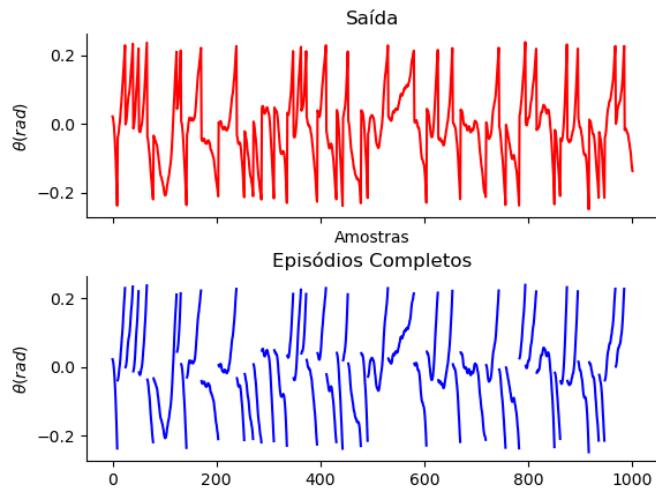
$$M = \frac{u - k m \ddot{\theta} - k m \ddot{x}}{k \ddot{x}}, \quad (3.28)$$

um aspecto crucial a ser considerado é que a massa do carrinho deve ser um valor positivo ( $M > 0$ ). Portanto:

$$0 < \frac{u - k m \ddot{\theta} - k m \ddot{x}}{k \ddot{x}} \Rightarrow 0 < 1 + k \cdot 0,5 \cdot 0,65 \cdot 0,29775 - k \cdot 0,5 \cdot 0,19524, \quad (3.29)$$

ao resolver a Inequação 3.29, chega-se a  $k < 1174,74$ , logo, para determinar o valor de  $k$ , ele será variado de 1 a 1174, considerando 1000 etapas (amostras). Um conjunto de etapas compõe o episódio, sendo que cada episódio é concluído quando o ângulo do polo não está mais na faixa de  $\pm 12^\circ$  ( $\pm 0,2095rad$ ). Sendo como segue:

Figura 3.9 – Episódios inteiros.



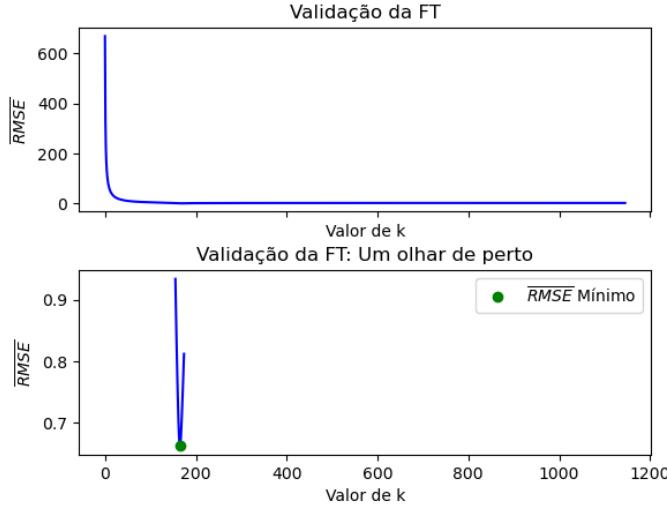
Fonte: Autor.

Ao total, foram realizados 47 episódios completos. Portanto, para cada valor de  $k$ , a simulação é repetida 47 vezes, e o somatório do Erro Quadrático Médio ( $RMSE$ )[10] é calculado. Em seguida, é determinada a média para obter o  $RMSE$  médio associado a  $k$ , da seguinte forma:

$$\overline{RMSE}_k = \frac{1}{47} \sum_{ep=1}^{47} \frac{\sqrt{\sum_{i=1}^N (y_{ep}(i) - \hat{y}_{ep}(i))^2}}{\sqrt{\sum_{i=1}^N (y_{ep}(i) - \bar{y}_{ep})^2}}, \quad (3.30)$$

onde  $\hat{y}_{ep}(i)$  representa a simulação livre para cada episódio (FT) e  $y_k$  é o sinal medido para cada episódio, com a média ( $\bar{y}_{ep}$ ) sendo calculada na janela de identificação. Graficamente:

Figura 3.10 – RMSE médio devido ao fator  $k$ .



Fonte: Autor.

Ao empregar o comando *min* do Python, obtém-se um valor de  $\overline{RMSE}$  igual a 0,66019 para  $k = 165$ .

Ao tomar  $k = 165$ , obtém-se uma FT que responde melhor aos estímulos da entrada. A nova massa a ser:

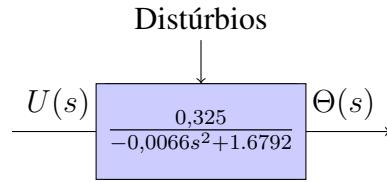
$$M = \frac{1 + 165 \cdot 0,5 \cdot 0,65 \cdot 0,29775 - 165 \cdot 0,5 \cdot 0,19524}{165 \cdot 0,19524} = 0,02668kg, \quad (3.31)$$

com os novos valores dos parâmetros, a FT passa a ser:

$$\frac{\Theta(s)}{U(s)} = \frac{0,325}{-0,00661s^2 + 1,67919}. \quad (3.32)$$

Logo, pode-se representar o diagrama de blocos em malha aberta do sistema:

Figura 3.11 – Diagrama do sistema.



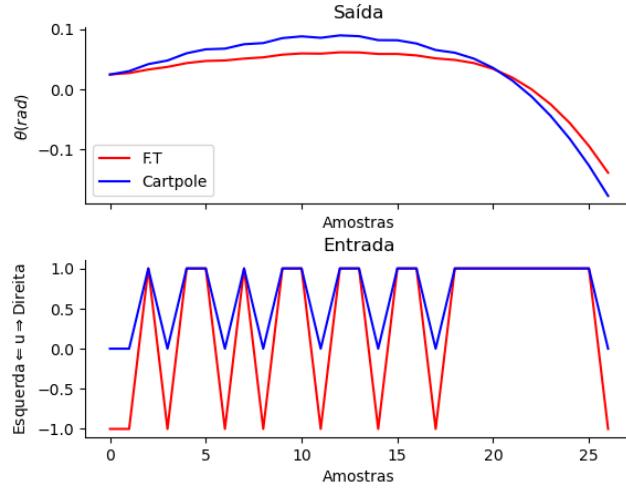
Fonte: Autor.

Onde  $U(s)$  representa a direção da força com módulo de 1N aplicada (com 1 indicando para a direita e -1 para a esquerda), enquanto  $\Theta$  refere-se ao ângulo do pêndulo.

Uma abordagem adicional envolve a identificação das raízes do polinômio no denominador da Função de Transferência, que são 15,9351 e  $-15,9351$ . É crucial notar que uma destas raízes está localizada no semiplano direito do eixo real, indicando a presença de um sistema instável.

Como exemplo do desempenho da FT, uma entrada será aplicada ao sistema para analisar como ele se comporta:

Figura 3.12 – Entrada e saída - Validação da FT.

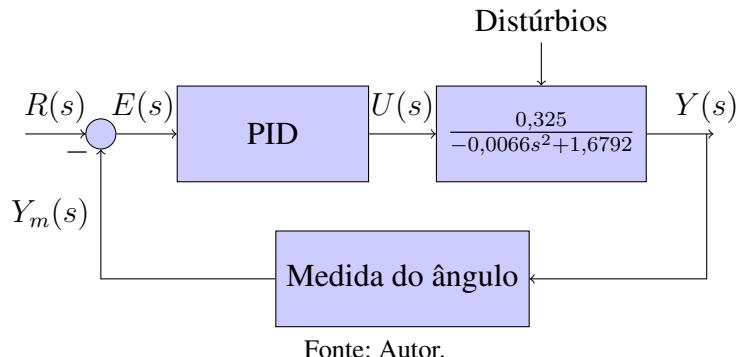


Fonte: Autor.

### 3.2.2 Sintonia do PID

Para estabilizar o sistema, será utilizado um controlador PID:

Figura 3.13 – Malha de controle para o *Cart Pole*.



Fonte: Autor.

No âmbito temporal, a formulação da expressão para um controlador PID é delineada da seguinte maneira:

$$g(t) = K_C \cdot e(t) + \frac{K_C}{\tau_I} \cdot \int_0^t e(t)dt + K_C \cdot \tau_D \cdot \frac{de(t)}{dt}. \quad (3.33)$$

Para a determinar dos valores de  $K_P$ ,  $K_I$  e  $K_D$ , será utilizado o pacote GEKKO[11]. Este pacote, uma ferramenta em Python dedicada ao aprendizado de máquina e otimização de inteiros mistos, bem como a equações algébricas diferenciais usada para resolver a Equação 3.33. Nessa equação,  $\tau_i$  é fixado em 2 e  $\tau_d$  em 0,25. Em seguida, o pacote fornecerá o valor de  $K_C$ . A partir

desse ponto, os parâmetros podem ser facilmente determinados. O ponto de referência será estabelecido em  $\theta = 0$  ( $r(t)$ ) mesmo após um distúrbio, será usada uma função impulso:

$$e(t) = r(t) - y_m(t), \quad (3.34)$$

o GEKKO, ao buscar a minimização do erro, simplifica o processo, necessitando apenas da introdução da FT e da aplicação do impulso à entrada. Os valores estimados estão vinculados ao  $K_C$  determinado, o qual é 0,5, os valores de  $\tau_i$  e  $\tau_d$  foram previamente estabelecidos como 2 e 0,25, respectivamente:

Tabela 3.2 – Parâmetros do PID.

Termos	Relação com $K_C$	Valor
Proporcional	$K_C$	0,5
Integral	$\frac{K_C}{\tau_I}$	0,25
Derivativo	$K_C \cdot \tau_D$	0,125

Fonte: Autor.

Os valores dos parâmetros  $K_P$ ,  $K_I$  e  $K_D$  serão incorporados à Equação 3.33, resultando em:

$$g(t) = 0,5 \cdot e(t) + 0,25 \cdot \int_0^t e(t)dt + 0,125 \cdot \frac{de(t)}{dt}. \quad (3.35)$$

Ainda é necessário determinar o valor do erro (proporcional), sua integral e derivada no contexto do controle de um pêndulo invertido. Dado que o *setpoint* almejado (representado por  $r(t)$ ) é manter o pêndulo em pé, idealmente, esse *setpoint* é zero. Assim, o erro torna-se o próprio ângulo do *Cart Pole*, enquanto a derivada é a velocidade angular, ambas informações fornecidas pela biblioteca Gymnasium.

Além disso, é crucial estimar a integral do erro. Fisicamente, a integral pode ser interpretada como a soma acumulativa do ângulo ao longo do tempo, já que o *setpoint* é zero. Este componente integral no controle é valioso para corrigir o erro acumulado ao longo do tempo, contribuindo para a estabilidade do sistema e a redução de desvios significativos. Matematicamente a integral é dada por:

$$\int_0^t e(t)dt = \sum_{i=0}^n \theta_i, \quad (3.36)$$

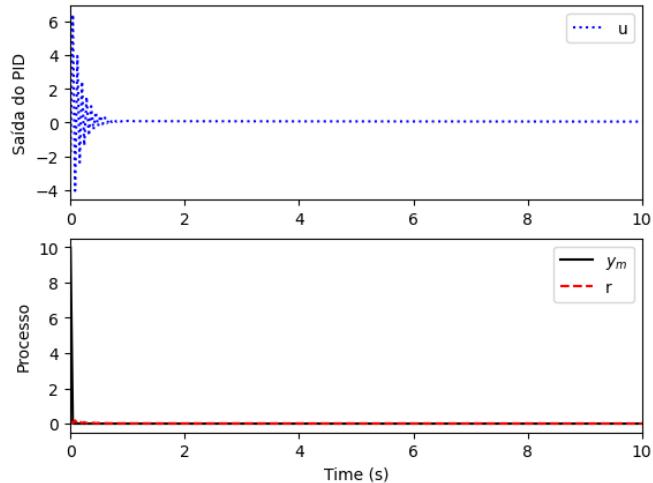
aqui,  $\theta_i$  denota o ângulo do pêndulo invertido, enquanto  $n$  representa o número atual da etapa.

Assim, ao levar em conta o ângulo, a velocidade angular e a integral do erro, é viável construir um conjunto abrangente de informações para a implementação do PID no ambiente do *Cart Pole* (pêndulo invertido) da seguinte forma:

$$u_n = 0,5 \cdot \theta_n + 0,25 \cdot \sum_{i=0}^n \theta_i + 0,125 \cdot \omega_n. \quad (3.37)$$

Antes de prosseguir com as tarefas, é fundamental realizar uma análise gráfica da saída do GEKKO. Essa visualização desempenhará um papel crucial como base para as etapas subsequentes deste trabalho:

Figura 3.14 – Resposta ao impulso para obtenção dos parâmetros do PID.



Fonte: Autor.

Pela Figura 3.14, observe que a saída  $u$  assume valores contínuos, mas o ambiente *Cart Pole* utilizado é discreto (1 para força à direita e 0 para força à esquerda). Para resolver esse problema, será implementada a seguinte estratégia, se  $u > 0$ , será aplicada uma força à direita; caso contrário, será aplicada uma força à esquerda.

### 3.3 *Cart Pole*: Controle Preditivo Generalizado Aplicado

Na presente seção, será exposto o processo para derivar a trajetória de referência, a função custo e obter o modelo ARX. Os princípios delineados na Seção 2.3.1 serão aplicados em um algoritmo para a realização dessas tarefas.

### 3.3.1 Controle Preditivo Generalizado: Formulação para o *Cart Pole*

A trajetória de referência ( $w$ ) representa o comportamento do sinal desejado para a saída futura, sendo o primeiro passo na aplicação do controle preditivo generalizado (GPC). No caso do pêndulo invertido, ela será 0, pois a ideia é manter o pêndulo estável, o que implica em minimizar o  $\theta$ , de maneira:

$$w = [0, 0, 0, \dots, n_{etapas}], \quad (3.38)$$

considerando que  $n_{etapas}$  representa o número total de etapas.

O próximo passo consiste em estabelecer uma função de custo, considerando as recompensas do ambiente *Cart Pole*. Nesse contexto, uma recompensa é atribuída por cada passo dado, abrangendo inclusive a etapa de encerramento, uma vez que o objetivo é manter o poste ereto pelo maior tempo possível. O limite estabelecido para as recompensas é de 500[17]. Logo,

o esforço de controle não será penalizado, apenas o erro em relação à predição da saída e à referência:

$$J(k) = \sqrt{\left( \sum_{j=d}^{h_p} [\hat{y}(j+k|k) - w(j+k)] \right)^2}, \quad (3.39)$$

a função de custo utilizada enfatiza a capacidade do parâmetro  $\theta$  de mudar de sinal, possibilitando sua proximidade ao limiar de 0 radianos.

No entanto, ainda é necessário identificar um modelo capaz de prever a saída ( $\hat{y}$ ) em função de  $\Delta u$ , seguindo a metodologia do GPC. Nesse contexto, a saída é representada por 1 para movimento à direita e -1 para movimento à esquerda, a fim de manter consistência com a codificação utilizada na FT. Isso resulta em  $\Delta u$  assumindo três valores inteiros, o que se justifica por:

$$\begin{aligned} u_{k-1} = 1 \text{ e } u_k = -1 &\Rightarrow \Delta u = -2 \\ u_{k-1} = 1 \text{ e } u_k = 1 &\Rightarrow \Delta u = 0 \\ u_{k-1} = -1 \text{ e } u_k = -1 &\Rightarrow \Delta u = 0 \\ u_{k-1} = -1 \text{ e } u_k = 1 &\Rightarrow \Delta u = 2 \end{aligned} \quad (3.40)$$

A definição do problema de otimização é a seguinte:

$$\begin{aligned} \text{Minimizar} \quad J(k) &= \sqrt{\left( \sum_{j=d}^{h_p} [\hat{y}(j+k|k) - w(j+k)] \right)^2}, \\ \text{sujeto a:} \quad \Delta u &= \{-2, 0, 2\} \end{aligned} \quad (3.41)$$

para minimizar essa função custo, emprega-se o Método de Pesquisa em Grade, o qual é detalhado na Seção 2.5. Esse método consiste em uma abordagem sistemática que varre diferentes combinações de parâmetros dentro de um intervalo predefinido (as combinações da entrada), buscando encontrar o conjunto ótimo que resulta na minimização da função custo.

### 3.3.2 Estimação do Modelo ARX

Agora será realizado o processo de identificação do modelo ARX do sistema, utilizando um sinal PRBS (do inglês, *Pseudo-Random Binary Sequence*) para excitar o *Cart Pole*. O PRBS [10] assume apenas dois valores possíveis,  $+V$  e  $-V$ . Além de ser fácil de implementar, é replicável, o que o torna bastante popular na identificação de sistemas.

O menor intervalo no qual o nível do sinal é mantido é denominado  $T_b$ . Seu período pode ser determinado por  $T = NT_b$ , sendo  $N$  um número ímpar, de modo que  $T_b = T_s$ . Um resultado heurístico para a escolha de  $T_b$  é:

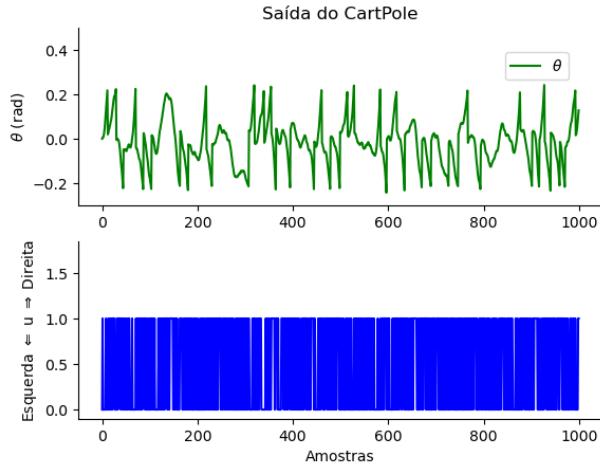
$$\text{Lineares} \Leftrightarrow \frac{\tau_{min}}{10} \leq T_b \leq \frac{\tau_{min}}{3} \Rightarrow \text{Não lineares}, \quad (3.42)$$

no trecho, onde  $\tau_{min}$  representa a menor constante de tempo de interesse, sugere-se, de acordo com Nelles (2001)[21], que, para sistemas lineares,  $T_b$  seja escolhido próximo ao valor do

intervalo de amostragem. Para sistemas não lineares, por outro lado, recomenda-se que  $T_b$  seja aproximadamente igual a  $\tau_{max}$ , onde  $\tau_{max}$  é a constante de tempo do sistema.

Trabalhando com um modelo ARX, será considerado o tempo de amostragem  $T_b$ . Logo, são obtidos:

Figura 3.15 – Aplicando o PRBS.



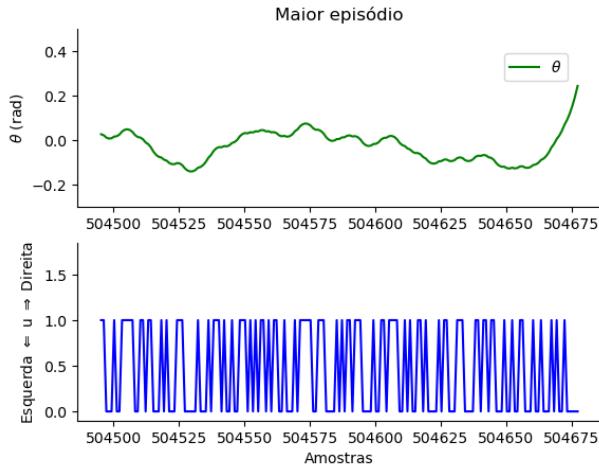
Fonte: Autor.

É importante observar que o ambiente *Cart Pole* reinicia automaticamente sempre que atinge um desvio de aproximadamente  $\pm 12^\circ$  (equivalente a  $\pm 0,209$  radianos). Essa característica representa um desafio, pois resulta em episódios muito curtos, dificultando a estimativa do modelo ARX. Inicialmente, uma abordagem para solucionar esse problema poderia envolver a redução do intervalo de tempo  $T_b$ . Isso implicaria em uma diminuição no tempo de amostragem do sistema, permitindo que as ações sejam executadas em períodos mais curtos e, assim, estendendo a duração dos episódios.

No entanto, é crucial observar que essa abordagem não é viável no contexto do ambiente *Cart Pole* no Gymnasium. Isso se deve ao fato de que a taxa de amostragem no ambiente *Cart Pole* não é diretamente ajustável. O ambiente *Cart Pole* foi projetado para simular a física real associada ao problema de um pêndulo invertido sobre um carro, e a taxa de amostragem é intrínseca a essa simulação. Portanto, modificar diretamente a taxa de amostragem do ambiente não é uma opção disponível. Esse aspecto limita a capacidade de ajuste do tempo de amostragem para atender às necessidades específicas do modelo ARX no contexto do problema *Cart Pole*.

Para enfrentar essa tarefa complexa, serão realizadas duas milhões de iterações no ambiente de simulação. Dentre essas iterações, o episódio mais longo, com o maior número de passos ou amostras, será cuidadosamente selecionado para a estimativa do modelo ARX. Este episódio em particular se estende por 183 amostras, fornecendo assim uma base robusta e abrangente para a análise e modelagem do sistema. A escolha deste episódio mais extenso assegura que tenhamos dados significativos e suficiente para capturar as nuances e complexidades do sistema, permitindo uma estimativa precisa do modelo ARX:

Figura 3.16 – Maior episódio.



Fonte: Autor.

Será utilizado 158 amostras para o treinamento do modelo, sendo as 25 restantes empregadas para validação. O modelo ARX encontrado pelo SysIdentPy, foi:

$$y(k) = 2 \cdot y(k-1) - 0,99367 \cdot y(k-2) - 0,00583 \cdot u(k-1), \quad (3.43)$$

sendo  $RMSE = 0,0057$ .

Para lidar com a instabilidade do *Cart Pole*, adotou-se um horizonte de controle igual de previsão ( $H_p = H_c$ ), ajustando  $H_c$  com foco na eficiência do controle e tempo de execução, considere o tempo de execução  $t$  segundos para  $H_c = 2$ :

Tabela 3.3 – Horizonte de Controle em relação ao Tempo.

Horizonte de Conrole ( $H_c$ )	2	3	4	5	6	7
Tempo de Execução [s]	$t$	$1,071t$	$1,218t$	$1,8164t$	$3,58t$	$9,18t$

Fonte: Autor.

A escolha do horizonte de controle de 4 foi baseada na análise de que, apesar do aumento modesto no tempo de execução ao passar de 3 para 4, a transição de 4 para 5 representa um intervalo mais significativo. A lógica subjacente é buscar o maior horizonte de controle viável, minimizando simultaneamente o tempo de execução.

### 3.4 *Lunar Lander*: Implementação e Sintonia do Controlador PID

Nesta seção, será abordada a implementação do controlador PID passo a passo, com o objetivo de controlar a *Lunar Lander*. No entanto, a abordagem será mais empírica em comparação com o *Cart Pole*.

#### 3.4.1 *Lunar Lander*: Aplicando o controle PID

O controlador PID foi retirado do repositório Lunar Lander\_OpenAIGym [22] com algumas modificações. A ideia de modelar o sistema por meio de um modelo caixa branca não foi bem-sucedida, uma vez que se trata de um sistema MIMO, o que eleva o grau de complexidade.

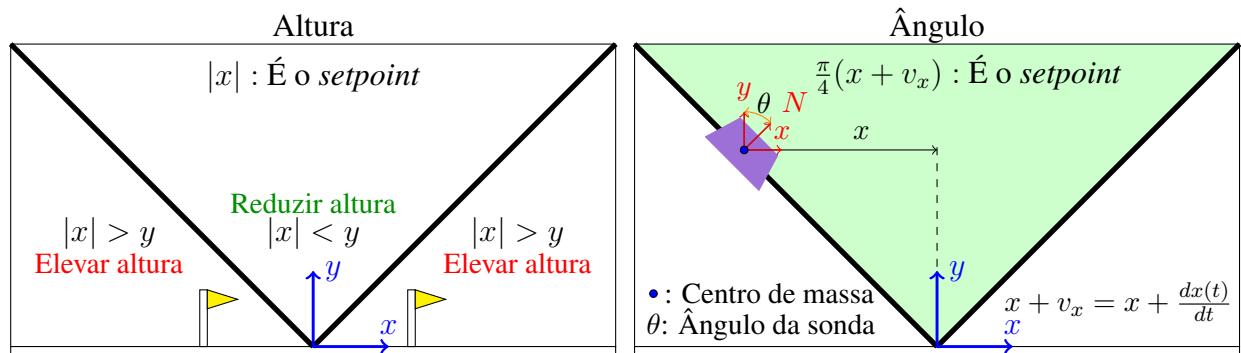
Seria necessário uma função de transferência para cada entrada e saída, tornando complicada a determinação dos parâmetros intrínsecos ao sistema. Assim, este repositório apresenta uma abordagem mais empírica, semelhante ao que ocorre no meio industrial, onde o PID é amplamente utilizado.

A plataforma de aterrissagem permanece fixa nas coordenadas (0,0), sendo permitido a possibilidade de pouso fora da área designada. Além disso, é importante ressaltar que não há restrições quanto ao combustível, sendo considerado infinito para a execução da tarefa.

O primeiro passo é determinar as variáveis de processo a serem controladas, sendo o *boosters* do motor principal e secundário, usados respectivamente para controlar a altitude e o ângulo e/ou posição horizontal da nave.

A sonda ajusta-se com base nos sensores para minimizar erros ao longo do tempo, utilizando controle proporcional-derivativo (PD). Altitude, ângulo e velocidades são conhecidos em cada etapa. O erro é calculado como a diferença entre os *setpoints* e as medições atuais, permitindo controle proporcional. As velocidades são usadas para o controle derivativo. O passo subsequente é definir os *setpoints* para a implementação do controle PD:

Figura 3.17 – *Setpoints* para o controle PD.



Fonte: Autor.

O primeiro *setpoint* considerado é a altura, com a posição  $x$  como *setpoint*, conforme a Figura 3.17. Se a sonda estiver dentro do cone, deve descer, caso contrário, deve subir. Isso define o termo proporcional. Para o termo derivativo, utiliza-se a velocidade linear em  $y$ :

$$y_{PD} = k_{p2} \cdot (|x| - y) + k_{d2} \cdot v_y. \quad (3.44)$$

É crucial manter uma inclinação constante da nave em direção ao seu objetivo, pois isso determina a direção do impulso do propulsor principal. Para implementar essa abordagem, utiliza-se o *setpoint*  $x + v_x$ , onde  $v_x$  é a taxa de variação em  $x$ , entendendo essa expressão como  $x_{t+1}$  para minimizar  $\theta$ . Quando a sonda está nas bordas extremas do triângulo, ela deve se inclinar 45° em direção a plataforma, com essa inclinação diminuindo à medida que a sonda se aproxima do alvo (0,0), de acordo com o *setpoint*  $x_{t+1}$ . O controle proporcional é aplicado, onde a posição  $x$  diminui à medida que a sonda se aproxima do alvo, enquanto o termo derivativo utiliza a velocidade angular:

$$\theta_{PD} = k_{p2} \cdot \left[ \frac{\pi}{4} \cdot (x + v_x) - \theta \right] + k_{d2} \cdot v_\theta. \quad (3.45)$$

Todos os elementos essenciais foram identificados, exceto pelos valores apropriados dos quatro parâmetros  $k_{p1}$ ,  $k_{d1}$ ,  $k_{p2}$  e  $k_{d2}$ . Para determinar esses pesos, será adotada a técnica de Otimização por Subida de Encosta. Ao que tudo indica, diferentemente do que é feito no repositório LunarLander\_OpenAIGym [22], onde  $k_p$  e  $k_d$  são iguais no PD referente ao ângulo e à altura, o que leva a crer que foi sintonizado manualmente. Essa metodologia inicia com a premissa de que todos os parâmetros são inicialmente nulos (sem controle). Após cada tentativa de aterrissagem da sonda e avaliação da pontuação, os parâmetros são ajustados com pequenas variações aleatórias. Se a pontuação da sonda melhorar, os novos valores são mantidos e o processo é repetido. Caso contrário, os novos valores são descartados e tenta-se adicionar ruído aleatório novamente. Os valores encontrados foram:  $k_{p1} = 9,0565$ ,  $k_{d1} = -9,9488$ ,  $k_{p2} = 11,9271$  e  $k_{d2} = -5,0963$ .

### 3.5 *Lunar Lander: Controle Preditivo Generalizado Aplicado*

Nesta seção, será apresentado como se chega às trajetórias de referência, à função custo e como o modelo ARX é obtido. A descrição presente na Seção 2.3.1 será aplicada em um algoritmo.

#### 3.5.1 Controle Preditivo Generalizado: Formulação para o *Lunar Lander*

As trajetórias de referência ( $w$ ) do *Lunar Lander* são um pouco mais complexa, na qual serão adotadas algumas estratégias.

Sabe-se que o ambiente no qual a sonda interage é traduzido por meio de um plano cartesiano. Essa abordagem permite conhecer valores como posição e direção, aspectos de suma importância na navegação da sonda.

Com intuito de implementar o GPC, será empregado uma matriz rotação para rotacionar o plano cartesiano em  $45^\circ$  (o motivo será explicado adiante), por ser bidimensional, tem-se:

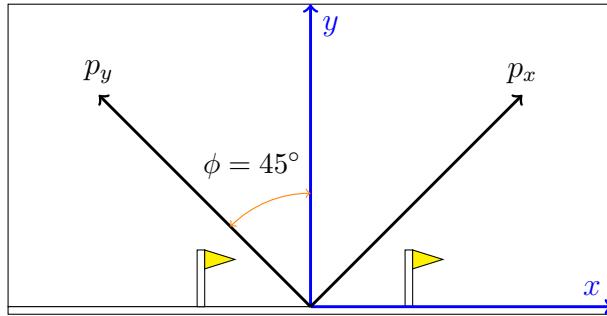
$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3.46)$$

se considerar  $\phi = 45^\circ$  e respectivos valores fornecidos pelo ambiente de observação:

$$\begin{aligned} p_x &= x \cdot \cos 45^\circ + y \cdot \sin 45^\circ \\ p_y &= -x \cdot \sin 45^\circ + y \cdot \cos 45^\circ, \end{aligned} \quad (3.47)$$

a ideia é rotacionar o plano cartesiano, da seguinte maneira:

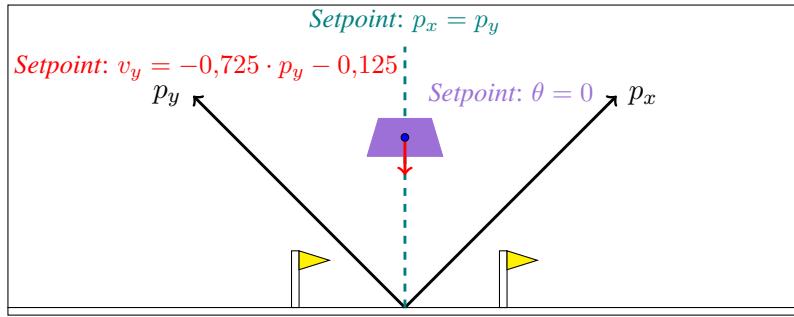
Figura 3.18 – Plano cartesiano do *Lunar Lander*,



Fonte: Autor.

Com o plano cartesiano rotacionado, é possível estimar três trajetórias de referência. A primeira, em ciano, é definida por  $p_x = p_y$ . Similarmente ao *Cart Pole*, o ângulo  $\theta$  do *Lunar Lander* deve estar próximo de zero, representando a segunda trajetória de referência. O próximo passo envolve determinar a terceira trajetória de referência para a velocidade linear em  $p_y$ , expressa por  $v_y = -0,725 \cdot p_y - 0,125$ . Esta ideia será ilustrada.

Figura 3.19 – *Setpoints* para o GPC,



Fonte: Autor.

Para estimar o modelo ARX e facilitar a compreensão, os propulsores auxiliares de orientação direita e esquerda serão considerados como um só ( $u_2$ ), assumindo 1 para direita e -1 para esquerda, com 0 representando a condição desligada. Para o propulsor principal ( $u_1$ ), 1 indica o acionamento e 0 representa que está desligado.

Em relação ao incremento de controle ( $\Delta u$ ), uma metodologia alternativa está sendo considerada para o ambiente *Lunar Lander*. Enquanto o MPC tradicionalmente penaliza o  $\Delta u$  com base em sua magnitude, neste contexto lunar específico, a penalização varia com os propulsores: auxiliares têm penalização de 0,03, o principal de 0,3 e nenhum acionamento não é penalizado. Dessa forma, a proposta é focar no controle absoluto  $u$  em vez do incremento  $\Delta u$ . Quando o horizonte de controle ( $H_c$ ) é menor que o de previsão ( $H_p$ ),  $u$  é mantido em 0, permitindo respostas que maximizam a recompensa do ambiente.

Conforme mencionado anteriormente, a metodologia de aplicação do modelo define a função de custo conforme a Equação 3.48:

$$J(k) = \sum_{j=d}^{h_p} [\{\alpha \cdot (\hat{p}_x(j+k|k) - \hat{p}_y(j+k|k)) - \beta \cdot \hat{\theta}(j+k|k) + \delta \cdot (\hat{v}_y(j+k|k) - 0,725 \cdot \hat{p}_y(j+k|k) - 0,125)\}^2]^{\frac{1}{2}} \quad (3.48)$$

sujeito a:

$$Propulsor = \begin{cases} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{cases} u_1 \Rightarrow \text{Propulsor principal} \\ \begin{cases} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{cases} u_2 \Rightarrow \text{Propulsor auxiliar}$$

as entradas devem ser intercaladas ou completamente desligadas. Em outras palavras, se uma estiver ativa, a outra deve permanecer desligada, ou ambas devem estar desligadas simultaneamente. O horizonte de controle foi definido como igual a 2 ( $H_c = 2$ ) e o horizonte de predição como 4 ( $H_p = 4$ ). Esses valores foram definidos experimentalmente e, apesar de serem pequenos, apresentaram uma boa resposta com um excelente tempo de execução.

### 3.5.2 Estimação do Modelo ARX

Nesse processo, optou-se por empregar uma entrada aleatória, com o objetivo de obter um modelo para as coordenadas  $p_x$  e  $p_y$ ,  $\theta$  e  $v_y$  (velocidade linear em y). Durante a obtenção do modelo, observou-se uma correlação de cada uma das saídas com uma entrada específica, conforme segue:

Tabela 3.4 – Relação entre entradas e saídas.

Saída	$p_x$	$\theta$	$p_y$	$v_y$
Entrada	$u_2$	$u_2$	$u_1$	$u_1$

Fonte: Autor.

Dessa forma, a metodologia começou a investigar várias alternativas para cada entrada. A cada iteração, um valor era selecionado aleatoriamente dentre as opções disponíveis. Para  $u_1$ , o intervalo considerado foi  $[0, 2]$  do ambiente e  $[0, 1]$  para a identificação, enquanto  $u_2$  apresentava três possíveis respostas  $[0, 1, 3]$  e  $[0, 1, -1]$  para a identificação.

Novamente, será usado o SysIdentPy para estimar os modelos ARX para os itens já mencionados:

$$p_x(k) = 1,9986 \cdot p_x(k-1) - 0,9986 \cdot p_x(k-2) \\ - 7,3642 \cdot 10^{-5} \cdot u_2(k-1) - 4,7979 \cdot 10^{-4} \Rightarrow RMSE = 0,0009, \quad (3.49)$$

$$p_y(k) = 1,9928 \cdot p_y(k-1) - 0,99279 \cdot p_y(k-2) \\ + 8,1745 \cdot 10^{-5} \cdot u_1(k-1) - 3,9457 \cdot 10^{-4} \Rightarrow RMSE = 0,0355, \quad (3.50)$$

$$\theta(k) = -1,9891 \cdot \theta(k-1) + 0,989 \cdot \theta(k-2) \\ + 1,7201 \cdot 10^{-4} \cdot u_2(k-1) \Rightarrow RMSE = 0,0007, \quad (3.51)$$

$$v_y(k) = 1,2503 \cdot v_y(k-1) - 0,25137 \cdot v_y(k-2) \\ - 1,3791 \cdot 10^{-2} \cdot u_1(k-1) + 2,3644 \cdot 10^{-3} \Rightarrow RMSE = 0,0549. \quad (3.52)$$

Mesmo com o uso de apenas 70 amostras para a identificação do modelo, o SysIdentPy demonstrou um desempenho notável ao capturar um modelo de alta qualidade para o *Lunar Lander*. Esse desempenho ressalta a eficácia e a robustez do pacote SysIdentPy, que conseguiu

extrair informações precisas o suficiente das amostras disponíveis para formar um modelo que adequadamente representa a complexa dinâmica do *Lunar Lander*.

### 3.5.3 Determinação dos Pesos da Função Custo

Inicialmente, a estimativa dos pesos foi feita por tentativa e erro, onde foram identificados alguns pontos importantes posteriormente utilizados. O principal foi a separação do ambiente lunar em dois estágios. O primeiro estágio consiste em verificar se a velocidade da sonda é inferior à trajetória de referência. Nesse caso, o peso  $\delta$  na Equação 3.48, sendo zero, ajuda o modelo a não tomar decisões equivocadas, como inclinar em um ângulo  $\theta$  grande e utilizar os motores auxiliares para acelerar e alcançar a referência. Dessa maneira, a sintonização manual se tornou mais tranquila, onde no primeiro estágio o algoritmo se preocupa apenas com os *setpoints* de posição e inclinação angular. No segundo estágio, a velocidade linear em  $y$  passa a fazer parte dos *setpoints*.

Apesar de muitas tentativas, o GPC apresentou resultados semelhantes ao PD do repositório LunarLander\_OpenAIGym. No entanto, ao adotar um algoritmo de Subida de Encosta para estimar os parâmetros proporcionais e derivativos do PD deste trabalho, houve uma melhoria significativa no desempenho do PD. Assim, um ajuste manual do GPC para se equiparar ao PD tornou-se complicado. A proposta consistiu em aplicar o algoritmo de Subida de Encosta para estimar os pesos do segundo estágio ( $\alpha_2$ ,  $\beta_2$ , e  $\delta_2$ ), enquanto os pesos do primeiro estágio foram previamente definidos manualmente como  $\alpha = \beta = 1$  e  $\delta = 0$ , mantidos devido ao seu desempenho satisfatório. Os pesos empregados foram:

Figura 3.20 – Pesos da Função de Custo para o *Lunar Lander*.

1º Estágio	$\beta_1 = 1$ $\alpha_1 = 1$ $\delta_1 = 0$
$\hat{y}_v[0] = -0,725\hat{p}_y[0] - 0,125$	
2º Estágio	$\beta_2 = 23,1039$ $\alpha_2 = 3,6923$ $\delta_2 = 58,5223$

Fonte: Autor.

Os pesos ( $\alpha_2$ ,  $\beta_2$  e  $\delta_2$ ) derivados pelo algoritmo de subida de Encosta resultaram em uma notável melhoria no desempenho do Controle Preditivo Generalizado (GPC), em comparação com os ajustes manuais realizados por tentativa e erro. Essa otimização automática permitiu uma sintonia mais precisa e eficiente dos parâmetros do controlador, levando a um funcionamento mais eficaz do sistema de controle.

## 4 RESULTADOS

### 4.0.1 Cart Pole

A seguir, encontra-se um GIF que ilustra o funcionamento do Controle PID e GPC:

Figura 4.1 – PID: *Cart Pole* GIF.

Figura 4.2 – GPC: *Cart Pole* GIF.

Fonte: Autor.

Para facilitar a compreensão, é útil representar o comportamento do *Cart Pole* de maneira gráfica:

Figura 4.3 – PID: *Cart Pole*.

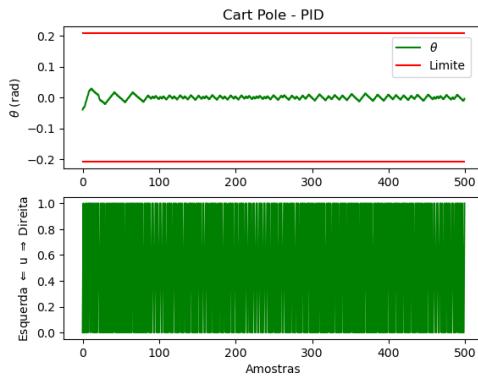
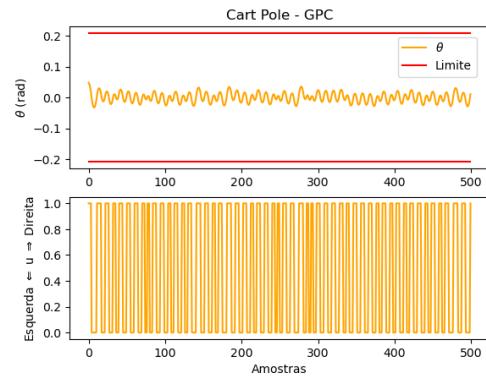


Figura 4.4 – GPC: *Cart Pole*.



Fonte: Autor.

Ambas as estratégias de controle (GPC e PID) demonstraram ser capazes de controlar o sistema, entretanto, o PID foi mais eficaz em manter o *Cart Pole* mais estável:

Tabela 4.1 – PID x CPC - *Cart Pole*.

Controle	$\theta$	$\theta_{max}$	Etapas	Tempo de Execução [s]
PID	$1,1053 \cdot 10^{-5}$	$4,6395 \cdot 10^{-2}$	5000	9,0421
GPC	$-9,7546 \cdot 10^{-5}$	$7,3337 \cdot 10^{-2}$	5000	15,1423

Fonte: Autor.

A capacidade antecipativa do GPC se fundamenta na consideração não apenas do estado presente do sistema, mas também na previsão de seu comportamento futuro. No entanto, devido à adoção de um horizonte de previsão e controle relativamente curto neste caso específico, o GPC tem uma visão limitada do futuro. Isso resulta na persistência de uma ação de controle específica ( $\Delta u = 0$ ), exacerbando a aceleração angular do *Cart Pole* (ver Figura 4.1). A consequente

ampliação da inércia do sistema implica uma força contrária prolongada para corrigir  $\theta$  quando há inversão de sinal. Esse processo se repete em um ciclo contínuo.

Esse horizonte de controle e previsão ( $H_c = H_p$ ) curto se deve ao problema de otimização inteira. Para sua resolução, foi utilizado o método de pesquisa em grade, que é custoso computacionalmente. Mesmo após um estudo indicar que  $H_c = H_p = 4$  seria a melhor opção, o GPC ainda demonstra ser mais lento que o PID. Se houver um aumento de uma unidade nos horizontes, o ganho não é tão significativo em termos de desempenho, no entanto, o tempo de resposta aumenta consideravelmente.

#### 4.0.2 *Lunar Lander*

A seguir, encontra-se um GIF que ilustra o funcionamento do Controle PD e GPC:

Figura 4.5 – PD: *Lunar Lander* GIF.

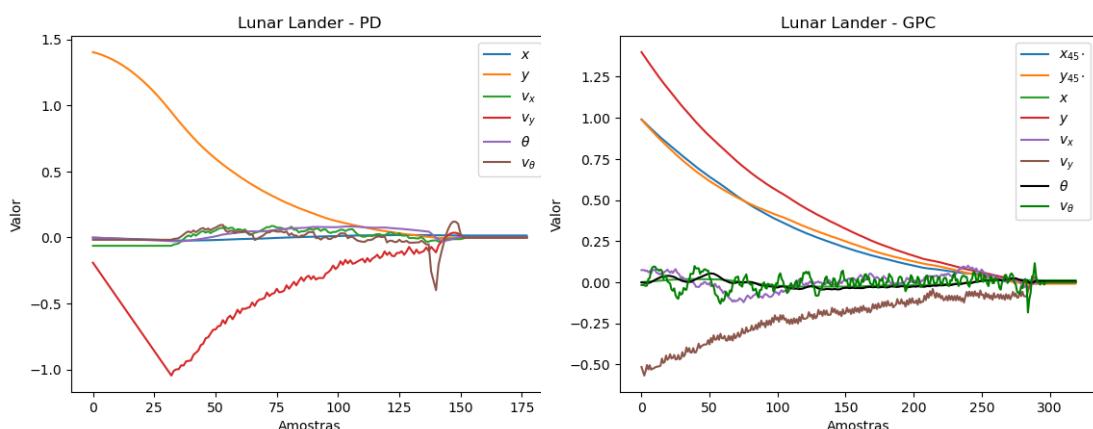
Figura 4.6 – GPC: *Lunar Lander* GIF.

Fonte: Autor.

É útil representar o comportamento da *Lunar Lander* de maneira gráfica:

Figura 4.7 – PD: *Lunar Lander*.

Figura 4.8 – GPC: *Lunar Lander*.



Fonte: Autor.

O GPC e o PD conseguiram realizar a aterrissagem da sonda lunar (*Lunar Lander*). Observando o GIF dos três poucos (Figura 4.5 e 4.6) e o gráfico apresentado na Figura 4.7, é evidente que o GPC adota uma abordagem mais cautelosa em comparação com o PD. Isso é perceptível pelo fato de o GPC iniciar a desaceleração da sonda antes do PD, resultando em um pouso mais suave, embora com mais etapas. Para validar será considerada 1000 tentativas de pouso:

Tabela 4.2 – PID x GPC - *Lunar Lander*.

Controle	Pouso	Pouso com +200 pontos	Etapas	Tempo de execução [s]	Pontuação média
PD	765	764	177577	1086,6976	217,4862
GPC	870	808	340369	1017,0346	209,1881

Fonte: Autor.

De maneira geral, ambos os controladores obtiveram resultados satisfatórios, porém o GPC demonstrou superioridade em relação ao PD, apresentando uma resposta mais consistente em termos de episódios resolvidos (+200 pontos). Por exemplo, o GPC alcançou sucesso em 87% das tentativas, indicando que a sonda conseguiu atingir o local de pouso designado e parou sem movimentos adicionais. Além disso, em 80,8% das vezes, o GPC alcançou 200 pontos, indicando não apenas um pouso bem-sucedido, mas também a satisfação de outros critérios estabelecidos pelo ambiente para resolver completamente o desafio. Ao considerar um horizonte de predição igual a 4 ( $H_p = 4$ ) e um horizonte de controle igual a 2 ( $H_c = 2$ ), o GPC apresentou um tempo de execução melhor entre as amostras (0,6145 ms). No entanto, os PD's mostraram uma resposta de pontuação média melhor, pois, por serem mais arrojados em seus *setpoints*, conseguiram pousar com menos amostras e acionamentos dos propulsores (principal e secundário).

Entretanto, neste caso, tratava-se de um sistema MIMO, o que exigia dois PD's aplicados de forma empírica, sem a utilização de um modelo matemático de caixa branca, como no caso do *Cart Pole*. Tentou-se aplicar tal abordagem, mas a complexidade em determinar as relações entre as saídas e entradas, bem como estimar os parâmetros físicos da sonda por meio de simulações, revelou-se muito grande. Também foi observada uma dificuldade em definir um *setpoint* mais elaborado para os PD's, optando-se por técnicas que simplificam os *setpoints* para torná-los mais compatíveis com o controlador. Além disso, sua resposta não é aplicável, pois o *Lunar Lander* opera com entradas inteiras, requerendo ajustes na saída dos PD's.

Por outro lado, o GPC utiliza um modelo ARX. Embora seja caixa preta, é mais flexível do ponto de vista da implementação de *setpoints*, permitindo a inserção intuitiva de vários *setpoints*. Outra vantagem é que ele fornece diretamente a ação de controle para o sistema.

## 5 CONCLUSÕES

Os resultados encontrados neste trabalho evidenciaram que ambas as técnicas de controle, tanto o Controle Proporcional-Integral-Derivativo quanto o Controle Preditivo Generalizado, demonstraram ser totalmente capazes de controlar sistemas complexos. É importante ressaltar que ambas as técnicas apresentaram desafios durante a implementação. Por exemplo, no caso do PID no *Cart Pole* os parâmetros intrínsecos do sistema, como a massa do carro, do pêndulo e a altura da haste do pêndulo, utilizados na Função de Transferência, foram determinados empiricamente. No entanto, após a sintonização pelo GEKKO, o sistema controlado apresentou resultados altamente promissores.

Para implementar os PD's no *Lunar Lander*, inicialmente planejou-se seguir os passos adotados no *Cart Pole*. No entanto, a obtenção da Função de Transferência revelou-se custosa devido

ao sistema ser MIMO (Múltiplas Entradas e Múltiplas Saídas). Além disso, a determinação dos parâmetros intrínsecos do sistema por experimentação mostrou-se inviável. Assim, optou-se por uma abordagem mais empírica, semelhante à aplicação industrial de controladores PID, onde muitos deles vêm com funcionalidades de ajuste automático incorporadas. No entanto, neste trabalho, o algoritmo de subida de encosta foi utilizado para ajustar os parâmetros conforme a recompensa, resultando em um bom controle.

A implementação do controle preditivo generalizado foi auxiliada pelo pacote SysIdentPy, uma ferramenta poderosa para identificação de sistemas. Um dos maiores desafios na identificação foi em relação ao *Cart Pole*, onde mesmo com o auxílio do PRBS, foi necessário fixar o número de passos em dois milhões para obter o modelo com o maior episódio. Este problema foi atribuído à natureza instável do sistema, no entanto, o controle do *Cart Pole* foi efetuado com sucesso. No *Lunar Lander*, o SysIdentPy obteve ótimos modelos com poucas amostras, e os ajustes dos pesos na função custo foram realizados por meio de tentativa e erro e com auxílio do algoritmo de Subida de Encosta. O resultado se mostrou muito bom.

Um dos principais obstáculos na implementação do GPC foi lidar com um problema de otimização inteira, que foi resolvido pelo Método de Pesquisa em Grade, um algoritmo com complexidade  $O(n!)$ , ou seja, o tempo de execução aumenta fatorialmente em relação à entrada. De maneira geral, no *Cart Pole*, o PID, além de mais estável, foi mais rápido. Pois um dos seus problemas foi por ser um sistema instável, adotou-se um horizonte de controle igual ao de predição ( $H_c = H_p = 4$ ), o que aumenta a complexidade do algoritmo. Já no *Lunar Lander*, a adoção de uma metodologia em termos de  $u$  ao invés de  $\Delta u$ , permitiu otimizar a resposta em relação a pontuação, e trabalhar com um horizonte de controle menor que o de predição ( $H_c = 2$  e  $H_p = 4$ ), o que por sua vez permitiu uma resposta mais rápida do controlador GPC, até mesmo que os controles PD's empregados.

Em suma, ambos os métodos se mostraram capazes de controlar sistemas complexos, mas o PID se saiu melhor com o *Cart Pole* e o GPC com o *Lunar Lander*. Este trabalho contribui para o entendimento e aplicação prática dessas técnicas em contextos reais, abrindo portas para futuras investigações e desenvolvimentos nesse campo promissor.

## 6 PROPOSTAS DE TRABALHOS FUTUROS

Segue abaixo uma lista de sugestões para os próximos trabalhos:

- Trabalhar com bibliotecas de otimização inteira em Python em conjunto com o Controle Preditivo Generalizado, como a Python-MIP, e comparar os resultados obtidos através do Método de Pesquisa em Grade em relação ao tempo de execução e pontuação;
- Investigar e comparar o desempenho de diferentes algoritmos de otimização para ajuste de parâmetros em controladores PID e GPC. Isso pode incluir algoritmos genéticos, algoritmos de enxame de partículas, entre outros.

## REFERÊNCIAS BIBLIOGRÁFICAS

- [1] A. S. of Naval Engineers, *Journal of the American Society of Naval Engineers, Inc*, vol. 42. American Society of Naval Engineers., 1922.
- [2] P. Deulkar and S. Hanwate, “Analysis of pso-pid controller for cstr temperature control,” in *2020 IEEE First International Conference on Smart Technologies for Power, Energy and Control (STPEC)*, pp. 1–6, IEEE, 2020.
- [3] C. B. Alba and I. C. d. E. en Automatica, “Control predictivo: metodología, tecnología y nuevas perspectivas,” *Universidad de Sevilla, Aguadulce, Almeria*, 2000.
- [4] W. R. Lacerda, L. P. C. da Andrade, S. C. P. Oliveira, and S. A. M. Martins, “Sysidentpy: A python package for system identification using narmax models,” *Journal of Open Source Software*, vol. 5, no. 54, p. 2384, 2020.
- [5] Ogata and Katsuhiko, *Engenharia de controle moderno*. Prentice Hall do Brasil, 5 ed., 2010.
- [6] P. B. d. L. Castrucci, A. Bittar, and R. M. Sales, *Controle Automático*. LTC, 1 ed., 2011.
- [7] I. Kafetzis and L. Moysis, “Inverted pendulum: A system with innumerable applications,” *School of Mathematical Sciences*, 2017.
- [8] T. C. Prata, “Controle preditivo baseado em modelo (mpc) aplicado a uma planta didática,” Master’s thesis, Instituto Federal de São Paulo, 2020.
- [9] E. Taketa *et al.*, “Aplicação do controle preditivo generalizado em sistemas mecânicos,” Master’s thesis, Universidade Tecnológica Federal do Paraná, 2018.
- [10] L. A. Aguirre, *Introdução à identificação de sistemas—Técnicas lineares e não-lineares aplicadas a sistemas reais*. Editora UFMG, 2004.
- [11] L. Beal, D. Hill, R. Martin, and J. Hedengren, “Gekko optimization suite,” *Processes*, vol. 6, no. 8, p. 106, 2018.
- [12] E. F. Camacho, C. Bordons, E. F. Camacho, and C. Bordons, *Model Predictive Control*. Springer, 2004.
- [13] M. Alhajeri and M. Soroush, “Tuning guidelines for model-predictive control,” *Industrial & Engineering Chemistry Research*, vol. 59, no. 10, pp. 4177–4191, 2020.
- [14] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, “Generalized predictive control—part i. the basic algorithm,” *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

- [15] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, “Generalized predictive control—part ii extensions and interpretations,” *Automatica*, vol. 23, no. 2, pp. 149–160, 1987.
- [16] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.
- [17] Barto, Sutton e Anderson, “Cartpole.” Online, 2023. [Acesso em setembro 2023].
- [18] Oleg Klimov, “Lunarlander.” Online, 2023. [Acesso em novembro 2023].
- [19] Alvaro Leandro Cavalcante Carneiro, “Algoritmos de otimização: Hill climbing e simulated annealing.” Online, 2020. [Acesso em dezembro 2023].
- [20] Ethan Roberts, “Using pid to cheat an openai challenge.” Online, 2021. [Acesso em setembro 2023].
- [21] N. Oliver, “Nonlinear system identification: from classical approaches to neural networks and fuzzy models,” *Ch*, vol. 11, no. 4, pp. 294–296, 2001.
- [22] Piyush Makhija, “Lunarlander\_openaigym.” Online, 2017. [Acesso em novembro 2023].