UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Laboratório de Controle e Automação: uma plataforma de simulação e representações gráficas.

Lucas Faria Antunes

TRABALHO DE CONCLUSÃO DE CURSO
JOÃO MONLEVADE, MG
2017

Universidade Federal de Ouro Preto
Departamento de Engenharia Elétrica

Laboratório de Controle e Automação: uma plataforma de simulação e representações gráficas.

Lucas Faria Antunes

Monografia apresentada ao Departamento de Engenharia Elétrica da Universidade Federal de Ouro Preto como parte dos requisitos exigidos para a obtenção do título de Bacharel em Engenharia Elétrica.

Área de Concentração: Engenharia Elétrica

Orientador: Prof. Dr. Víctor Costa da Silva Campos

João Monlevade, MG
2017

A636l Antunes, Lucas Faria.
Laboratório de controle e automação [manuscrito]: uma plataforma de simulação e representações gráficas / Lucas Faria Antunes. - 2017.

32f.:

Orientador: Prof. Dr. Víctor Costa da Silva Campos.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Engenharia Elétrica.

1. Engenharia de controle automático. 2. Simulação (sistemas de controle). 3. Pêndulo . 4. Computação gráfica. I. Campos, Víctor Costa da Silva. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 681.5

Catalogação: ficha@sisbin.ufop.br

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas
Colegiado do Curso de Engenharia de Elétrica

UFOP

ANEXO II - TERMO DE RESPONSABILIDADE

O texto do trabalho de conclusão de curso intitulado "Laboratório de controle e automação: uma plataforma de simulação e representações gráficas" é de minha inteira responsabilidade. Declaro que não há utilização indevida de texto, material fotográfico ou qualquer outro material pertencente a terceiros sem a devida citação ou consentimento dos referidos autores.

João Monlevade, 07 de abril de 2017.

Lucas Faria Antunes

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas
Colegiado do Curso de Engenharia de Elétrica

UFOP

ANEXO IV - ATA DE DEFESA

Aos 05 dias do mês de abril de 2017, às 17 horas e 30 minutos, no bloco B deste instituto, foi realizada a defesa de monografia pelo (a) formando (a) Lucas Faria Antunes, sendo a comissão examinadora constituída pelos professores: Márcio Feliciano Braga, Rodrigo Augusto Ricco, Víctor Costa da Silva Campos.

O (a) candidato (a) apresentou a monografia intitulada: Laboratório de Controle e Automação: uma plataforma de simulação e representações gráficas. A comissão examinadora deliberou, por unanimidade, pela aprovação do(a) candidato(a), com a nota média 8,5, de acordo com a tabela 1. Na forma regulamentar foi lavrada a presente ata que é assinada pelos membros da comissão examinadora e pelo (a) formando(a).

Tabela 1 – Notas de avaliação da banca examinadora

Banca Examinadora	Nota
Márcio Feliciano Braga	8,5
Rodrigo Augusto Ricco	8,5
Víctor Costa da Silva Campos	8,5
Média	8,5

João Monlevade, 05 de abril de 2017.

Víctor Costa da Silva Campos
Professor(a) Orientador(a)

Lucas Faria Antunes
Aluno (a)

Márcio Feliciano Braga
Professor(a) Convidado(a)

Rodrigo Augusto Ricco
Professor(a) Convidado(a)

AGRADECIMENTOS

Primeiramente agradeço a Deus por me dar saúde, sabedoria e equilíbrio durante toda a minha graduação. Agradeço também meus pais Adilson e Iolanda e ao meu irmão Flávio que sempre estiveram junto comigo nessa árdua jornada, dando apoio e incentivo a todo momento.

Agradeço ao movimento escotista, a ordem DeMolay e a Maçonaria pelos grandes ensinamentos que me deram, moldando meu carácter e índole, e mostrando que o esforço é forma mais eficaz de se conquistar objetivos.

Agradeço também ao meu grande amigo, exemplo de pessoa e orientador Prof. Dr. Víctor Costa da Silva Campos, o qual sem seus esforços, conhecimento e dedicação nunca seria possível a realização desse trabalho.

Resumo

O *software* desenvolvido nesse texto tem como objetivo tem como principal objetivo o melhoramento do aprendizado dos alunos da área de engenharia, com o enfoque na área de controle e automação.

As escolhas dos *softwares* utilizados foram motivadas por serem gratuitos e autorizados para a implementação acadêmica, além de tornar possível a programação dos modelos de forma eficiente e totalmente independente da simulação disponibilizada pelo *software*.

Foi implementado um modelo clássico na área de controle — o pêndulo invertido — utilizando os *softwares* Unity e Visual Studio, realizando o desenvolvimento em malha aberta e em malha fechada.

Para o projeto em malha fechada foram desenvolvidos dois controladores: um que atua sobre a posição do carrinho e um que atua sobre o ângulo da haste do pêndulo. Para esse controle, foram programados algoritmos na linguagem C#. Os controladores foram implementados em cascata a partir de cálculos matemáticos desenvolvidos. O resultado para o projeto de controle pode ser analisado com valores dos parâmetros dos controladores pré-definidos e também com os valores de entradas realizada pelo usuário, para que assim o mesmo possa analisar a influência dos controladores no modelo de acordo com os valores escolhidos.

Apesar de ser desenvolvido na área de engenharia elétrica, ele se expande para outras áreas da engenharia, as quais levam em consideração o modelo simulado. O *software* será utilizado no laboratório de controle e automação da Universidade Federal de Ouro Preto para que possa ser utilizado pelos alunos da instituição para o melhor compreendimento e assimilação do conteúdo estudado em sala das disciplinas que forem passíveis de estudo dos modelos implementados. No desenvolvimento desse programa, a preocupação de realizar de forma mais prática, intuitiva e proveitosa foram levadas em consideração.

A ideia é que outros modelos matemáticos sejam anexados ao *software* futuramente para que possam ser analisados e melhor compreendidos de forma mais prática por discentes de acordo com o interesse de disciplinas da engenharia.

Abstract

The main goal of the software developed in this work is the improvement of engineering students apprenticeship, focusing on control and automation fields of study.

The choice of the software used was motivated, mainly, because the chosen software is free and authorized for academic implementation. Moreover, the software makes it possible to program the models efficiently and completely independent of the simulation available by the software.

A classic model in the control area - the inverted pendulum - was implemented using the software Unity and Visual Studio, performing the development in open and closed loop configurations.

In the closed loop model, two controllers were developed: one that acts on the position of the cart and another one that acts on the angle of the pendulum rod. For this control, algorithms were programmed in the C# language. The controllers were implemented in cascade from developed mathematical calculations. These models can be analyzed with the pre-defined controllers parameters and also with the input values performed by the user (so that it can analyze the influence of the controllers in the model, according to the chosen values).

The software was developed to be used in electrical engineering, but it can still be expanded to other engineering applications (considering the simulated model). The software will be used in the control and automation laboratory of the Federal University of Ouro Preto. It will be used by the students of the institution for the better understanding and assimilation of the content studied in classroom. In the development of this program, the concern to perform in a more practical, intuitive and useful way were considered.

The idea is to include other mathematical models to the software in the future, so that they can be analyzed and better understood in a more practical way by students according to the interest of engineering disciplines.

Lista de ilustrações

Sumário

1 Introdução

Para o desenvolvimento desse *software* foram respeitados os embasamentos e teorias matemáticas da área de controle. O mesmo é dotado de uma interface gráfica da realidade física de forma simplificada, o que permite a melhor assimilação do conteúdo visto em sala de aula com uma possível aplicação prática.

As teorias da área de controle referentes à modelagem matemática em sistemas em malha fechada e em malha aberta foram amplamente utilizadas, realizando-se modelagens para o modelo escolhido com os mesmos parâmetros que usualmente são estudados em sala de aula.

O projeto teve seu desenvolvimento em conjunto com o programa Pró-Ativa no ano de 2016, o qual desenvolveu parte dos algoritmos implementados no *software*. Com essa parceria, foi possível o desenvolvimento com o intuito de realizar práticas de simulação de sistemas dinâmicos de forma independente do *software* Matlab, que normalmente é utilizado, além de possibilitar a realização de parte das práticas em computadores pessoais.

1.1 Motivação

Com a implementação desse projeto, espera-se obter resultados positivos assim como (SHERNOFF; COLLER, 2013), em que "O jogo se mostrou uma ferramenta para transformar a forma de ensino dedutiva para os envolvidos"; não obstante o *software* ainda não foi utilizado por nenhum outro aluno com exceção ao autor.

Acredita-se que com a implementação do *software*, os alunos demonstrem mais interesse na solução dos problemas propostos na disciplina, de forma similar a (COLLER; SCOTT, 2009), que conclui em seu trabalho que "Quando começamos a ensinar no curso métodos numéricos baseados em jogo, sentimos que havia algo incomum e educacionalmente bom acontecendo. Os alunos pareciam mais interessados, mais engajados, e mais interessados em aprender o material.". Vale ressaltar ainda que, como em (COLLER; SCOTT, 2009), o desenvolvimento da parte matemática por parte dos alunos será necessária para aplicar os parâmetros dos controladores.

1.2 Softwares utilizados

Os *softwares* utilizados para a programação foram o *Unity* e o *Visual Studio*, sendo que as principais motivações para a escolha desses dois *softwares* foram pelo fato de serem gratuitos quando utilizados para implementação acadêmica e pelo alto nível de eficiência dos mesmos, uma vez que são comercialmente utilizados globalmente.

1.2.1 Unity

O *software* escolhido se trata de um motor de jogos 3D desenvolvido pela pela Unity Technologies e foi programado em sua versão para *Windows* 64 bits.

Conhecido mundialmente por ser utilizado no desenvolvimento de grandes jogos, ele possui uma comunicação direta com o *software Visual Studio*, possibilitando assim que algoritmos sejam desenvolvidos e aplicados ao *Unity* sem a utilização de *softwares* secundários.

Apesar de ter sido utilizada a linguagem *C#*, o programa comunica com vários outros tipos de linguagem e *softwares*, os quais podem ser encontrados no *site* da empresa desenvolvedora.

A versão utilizada para a implementação desse trabalho foi a versão *Unity 5*, lançada no ano de 2015.

Figura 1 – Logomarca do *software* Unity.

1.2.2 Visual Studio

O *software Visual Studio* foi desenvolvido pela *Microsoft Visual Studio* e trata-se de uma ferramenta para desenvolvimento de *softwares*. É um ambiente de desenvolvimento integrado (IDE - Integrated Development Environment) que possui recursos para Android, iOS, Windows, Web e nuvem.

Optamos por desligar a física do *Unity* pois a mesma seria de alta complexidade no quesito implementação dos controladores (além de estarmos limitados para certos graus de liberdade e manipulação das variáveis físicas) o que estaria em desacordo com o propósito do trabalho. O tratamento da física foi realizado utilizando algoritmos programados no Visual Studio na linguagem *C#*, tornando possível assim a execução e manipulação de toda a física do sistema.

A versão utilizada para esse trabalho foi a versão community 2015.

Figura 2 – Logomarca do *software* Visual Studio.

1.3 Objetivos

Com o desenvolvimento desse *software*, temos como principais objetivos:

i. Implementação do modelo em um *software* com interface gráfica e interação com o usuário;

ii. Simulação em malha aberta do pêndulo invertido;

iii. Simulação em malha malha fechada do pêndulo invertido;

Simulação do controlador de ângulo;

Simulação do controlador de posição, utilizando controle em cascata com o controlador de ângulo;

iv. Disponibilização para os alunos.

1.4 Estrutura do texto

No Capítulo 1 foram apresentadas as motivações que levaram ao desenvolvimento desse trabalho, assim como os objetivos e os programas utilizados. No Capítulo 2 é apresentada a modelagem do sistema implementado (o pêndulo invertido), sendo demonstrado passo a passo como chegou-se ao modelo matemático tanto na forma linear quanto na forma não linear. No Capítulo 3 é apresentado todo o equacionamento dos controladores que foram implementados, assim como as estratégias utilizadas para o desenvolvimento desses controladores. No Capítulo 4 são citados os métodos matemáticos programados para simular toda a física envolvida no sistema e trechos dos códigos como exemplo das técnicas utilizadas, além dos *scripts* desenvolvidos. No Capítulo 5 é mostrada toda a forma de interação com o usuário, explicando passo a passo de como interagir com o *software* e no Capítulo 6 é apresentada a conclusão final desse projeto.

2 Modelagem do pêndulo invertido

O modelo de um pêndulo invertido é amplamente utilizado para testar diferentes estratégias de controle (OGATA, 2000) (ÅSTRÖM; FURUTA, 2000) (BOUBAKER, 2012) (DEMIRTAS; ALTUN; ISTANBULLU, 2008) (HOVLAND, 2008). Trata-se de um sistema não linear e instável (na posição de pêndulo invertido), pois o pêndulo tende a se afastar de sua posição inicial, sendo necessário assim uma força para equilibrá-lo na sua posição inicial. Para essa demonstração, adaptamos o desenvolvimento apresentado em (OGATA, 2000). Foram consideradas duas dimensões, referenciadas em X e Y num plano real que restringe o movimento do pêndulo, sendo a direção vertical é representada por V enquanto a direção horizontal é representada por H.

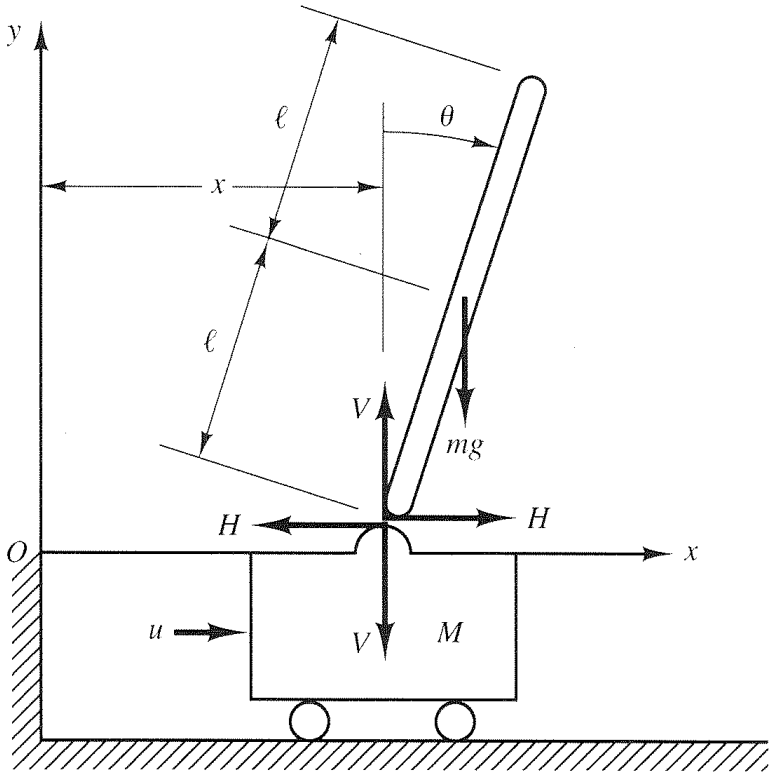

Figura 3 – Sistema pêndulo invertido

Fonte:(OGATA, 2000)

2.1 Considerações físicas para a modelagem

Para a modelagem desse pêndulo, foi considerada uma força u que é aplicada ao suporte móvel e ainda que o centro de gravidade da haste do pêndulo está em seu centro geométrico. Consideramos também que M é a massa do suporte móvel, a massa m é a massa do pêndulo e l corresponde à metade do raio de curvatura do pêndulo. Além disso, θ é o ângulo da haste vertical.

2.2 Equacionamento geral do sistema físico

Com as considerações físicas feitas anteriormente sobre o sistema, temos que essas grandezas se relacionam segundo as seguintes equações:

$$x_g = x + l\,\mathrm{sen}\theta \tag{2.1}$$

$$y_g = l\cos\theta \tag{2.2}$$

em que x_g e y_g são os pontos das coordenadas do centro de gravidade da haste do pêndulo.

O movimento de rotação da haste em torno do seu centro de gravidade é descrito por

$$I\ddot{\theta} = Vl\,\mathrm{sen}\theta - Hl\cos\theta \tag{2.3}$$

em que I é o momento de inércia da haste em relação a seu centro de gravidade.

O movimento horizontal e vertical da haste são dados:

$$m\frac{d^2}{dt^2}(x + l\cos\theta) = H \tag{2.4}$$

$$m\frac{d^2}{dt^2}(l\cos\theta) = V - mg \tag{2.5}$$

O movimento horizontal do suporte móvel é dado por:

$$M\frac{d^2x}{dt^2} = u - H \tag{2.6}$$

2.3 Modelagem do sistema não linear

As equações (2.3) a (2.6) descrevem o movimento do conjunto pêndulo invertido-suporte móvel, e o sistema pode ser representado da seguinte forma:

$$m\ddot{x} + m\frac{d}{dt}(l\dot{\theta}\cos\theta) = H \tag{2.7}$$

$$m\ddot{x} + m(-l\dot{\theta}^2 \operatorname{sen}\theta + l\ddot{\theta}\cos\theta) = H \tag{2.8}$$

$$m\ddot{x} - ml\dot{\theta}^2 \operatorname{sen}\theta + ml\ddot{\theta}\cos\theta = H \tag{2.9}$$

$$M\ddot{x} = u - H \tag{2.10}$$

Substituindo (2.9) em (2.10), temos:

$$(M+m)\ddot{x} - ml\dot{\theta}^2 \operatorname{sen}\theta + ml\ddot{\theta}\cos\theta = u \tag{2.11}$$

Na equação (2.11) temos os parâmentros físicos associados ao modelo. Para encontrarmos uma outra derivação do modelo, manipulamos as variáveis a equação (2.5) da seguinte maneira:

$$m\frac{d}{dt}(-l\dot{\theta}\operatorname{sen}\theta) = V - mg \tag{2.12}$$

$$V = mg + m(-l\dot{\theta^2}\cos\theta - l\ddot{\theta}\operatorname{sen}\theta) \tag{2.13}$$

$$I\ddot{\theta} = Vl\operatorname{sen}\theta - Hl\cos\theta \tag{2.14}$$

Usando a igualdade $I = ml^2$:

$$ml^2\ddot{\theta} = Vl\operatorname{sen}\theta - Hl\cos\theta \tag{2.15}$$

$$ml\ddot{\theta} = V\operatorname{sen}\theta - H\cos\theta \tag{2.16}$$

Substituindo (2.9) e (2.13) em (2.16), temos:

$$ml\ddot{\theta} = (mg - ml\dot{\theta^2}\cos\theta - ml\ddot{\theta}\operatorname{sen}\theta)\operatorname{sen}\theta - (m\ddot{x} - ml\dot{\theta^2}\operatorname{sen}\theta + ml\ddot{\theta}\cos\theta)\cos\theta \tag{2.17}$$

$$ml\ddot{\theta} = mg\operatorname{sen}\theta - ml\ddot{\theta}\operatorname{sen}^2\theta - m\ddot{x}\cos\theta - ml\ddot{\theta}\cos^2\theta \tag{2.18}$$

$$ml\ddot{\theta} = mg\operatorname{sen}\theta - m\ddot{x}\cos\theta - ml\cos\ddot{\theta} \tag{2.19}$$

$$2ml\ddot{\theta} + m\ddot{x}\cos\theta = mg\operatorname{sen}\theta \tag{2.20}$$

$$2l\ddot{\theta} + \ddot{x}\cos\theta = g\,\mathrm{sen}\theta \tag{2.21}$$

As equações (2.11) e (2.21) representam as equações do pêndulo invertido.

Para a simulação desse sistema, devemos representá-lo no espaço de estados. Para isso, consideramos os seguintes estados:

$$\begin{aligned} x_1 &= x \\ x_2 &= \theta \\ x_3 &= \dot{x} \\ x_4 &= \dot{\theta} \end{aligned} \tag{2.22}$$

Pelos estados escolhidos, temos as seguintes relações:

$$\begin{bmatrix} M+m & ml\cos\theta \\ \cos\theta & 2l \end{bmatrix} \begin{bmatrix} \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} ml\dot{\theta}^2\,\mathrm{sen}\theta + u \\ g\,\mathrm{sen}\theta \end{bmatrix} \tag{2.23}$$

que por manipulação matricial pode ser escrita como

$$\begin{bmatrix} \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \frac{1}{2(M+m) - m\cos^2\theta} \begin{bmatrix} 2 & -m\cos\theta \\ \frac{-\cos\theta}{l} & \frac{M+m}{l} \end{bmatrix} \begin{bmatrix} ml\dot{\theta}^2\,\mathrm{sen}\theta + u \\ g\,\mathrm{sen}\theta \end{bmatrix} \tag{2.24}$$

Realizando as multiplicações matriciais acima, temos que as equações que descrevem o comportamento do sistema, em conjunto com (2.22), são dadas por:

$$\dot{x_3} = \frac{2mlx_4^2\,\mathrm{sen}x_2 + 2u - mg\,\mathrm{sen}x_2\cos x_2}{2(M+m) - m\cos^2 x_2} \tag{2.25}$$

$$\dot{x_4} = \frac{-mlx_4^2\,\mathrm{sen}x_2\cos x_2 - cosx_2u + (M+m)g\,\mathrm{sen}x_2}{2(M+m)l - ml\cos^2 x_2} \tag{2.26}$$

2.4 Linearizando o sistema

Podemos assumir ainda que o ângulo θ seja muito pequeno nas equações (2.3) a (2.6) e linearizar as equações da seguinte forma:

$$I\ddot{\theta} = Vl\theta - Hl \tag{2.27}$$

$$m(\ddot{x} + l\ddot{\theta}) = H \tag{2.28}$$

$$0 = V - mg \tag{2.29}$$

$$M\ddot{x} = u - H \tag{2.30}$$

Manipulando as equações (2.28) e (2.30), temos:

$$(M + m)\ddot{x} + ml\ddot{\theta} = u \tag{2.31}$$

Manipulando as equações (2.27) e (2.29), temos:

$$I\ddot{\theta} = mgl\theta - Hl \tag{2.32}$$

$$I\ddot{\theta} = mgl\theta - l(m\ddot{x} + ml\ddot{\theta}) \tag{2.33}$$

$$2ml^2\ddot{\theta} + ml\ddot{x} = mgl\theta \tag{2.34}$$

Portanto, as equações (2.31) e (2.34) descrevem o comportamento do sistema pêndulo invertido-suporte móvel em sua forma linearizada.

3 Controladores

Foram sintonizados dois controladores em malha fechada para o sistema, um que realiza o controle de ângulo e outro que utiliza o controlador de ângulo para controlar a posição do carrinho. O que atua realizando o controle do deslocamento do ângulo foi sintonizado pelo método do lugar geométrico das raízes, e o que controla a posição no carrinho foi sintonizado pelo método de síntese direta.

Para a sintonia, consideramos as equações (2.31) e (2.34) e foi omitido o argumento jw em todas as equações. Considerando condições iniciais nulas, temos:

$$(M+m)s^2X + mls^2\theta = U \tag{3.1}$$

$$2ml^2s^2\theta + mls^2X = mgl\theta \tag{3.2}$$

Manipulando a equação (3.2), temos:

$$\begin{gathered} mls^2X = mgl\theta - 2ml^2s^2\theta \\ X = \frac{mgl\theta - 2ml^2s^2\theta}{mls^2} \end{gathered} \tag{3.3}$$

Aplicando a equação (3.3) em (3.1):

$$(M+m)g\theta - \frac{(M+m)(l+2ml^2)s^2\theta}{ml} + mls^2\theta = U$$

$$(ml - 2ml^2(m+M))s^2\theta + (M+m)g\theta = U \tag{3.4}$$

$$\frac{\theta(s)}{U(s)} = \frac{-1}{(2ml^2(m+M) - ml)s^2 - (M+m)g}$$

$$\frac{X(s)}{\theta(s)} = \frac{-2ml^2s^2 + mgl}{mls^2} \tag{3.5}$$

3.1 Controlador de ângulo

Foram feitas duas análises para o controlador para essa situação, sendo o controlador PI e o controlador PID. Essa análise foi feita decompondo os parâmetros P, I e D, encontrando um modelo para cada parâmetro antes de realizar o agrupamento.

3.1.1 Controlador proporcional-integral para controle do ângulo

A equação geral do controlador é dada por:

$$C(s) = K_p + \frac{K_r}{s} = \frac{U(s)}{E(s)} \tag{3.6}$$

De acordo com as teorias de controle, a análise considerando o erro do sistema é dada por:

$$U(s) = K_p E(s) + \frac{K_r}{s} E(s) \tag{3.7}$$

De forma geral, podemos considerar que:

$$u = K_p e + K_i \int e dt \tag{3.8}$$

Portanto, os estados do controlador são os mostrados no conjuntos de equações (3.9).

$$\begin{cases} \dot{x_i} = e \\ u = K_i x_i + K_p e \end{cases} \tag{3.9}$$

3.1.2 Controlador proporcional-integral-derivativo para controle do ângulo

Para o desenvolvimento desse controlador, foi utilizada a seguinte equação característica do mesmo:

$$C(s) = K_p + \frac{K_i}{s} + K_d s \frac{b}{s+b} \tag{3.10}$$

Para o controlador, foram considerados as seguintes equivalências e os seguintes estados:

$$\begin{gathered} K_p e \\ \frac{K_i}{s} = K_i x_i \\ \dot{x_i} = e \\ \frac{K_d b s}{s+b} = K_d b - \frac{K_d b^2}{s+b} = K_d b e + x_d \\ -\frac{K_d b^2}{s+b} = \frac{x_d}{e} \end{gathered} \tag{3.11}$$

$$-K_d b^2 e = \dot{x_d} + b x_d \tag{3.12}$$

$$\dot{x_d} = -b x_d - K_d b^2 e$$

Portanto, o controlador encontrado com os estados considerados foram os seguintes:

$$\begin{cases} \dot{x_i} = e \\ \dot{x_d} = -bx_d - K_d b^2 e \\ u = (K_p + K_d b)e + K_i x_i + x_d \end{cases} \tag{3.13}$$

3.2 Controlador para posição do carrinho

Para a sintonia desse controlador, foi utilizada a técnica de síntese direta pois é um método simples que nos permite especificar o desempenho desejado do sistema em malha fechada. Não foi possível utilizá-lo na seção anterior porque ele não pode ser aplicado em sistemas instáveis.

Para simplificar a síntese do controlador de posição e conseguir uma estratégia de controle de posição do carrinho que garanta que o pêndulo continuará controlado, fez-se uso da estratégia de controle em cascata. Nesse caso, utilizamos a referência do controlador de ângulo desenvolvido na seção anterior como a entrada de controle do sistema e assumimos que a resposta do controlador é perfeita (sem erro instânteneo). Dessa forma, para a síntese desse controlador, utilizamos a função de transferência do ângulo para a posição (3.6)

A equação (3.14) nos mostra a equação geral da síntese direta utilizada

$$\begin{aligned} G_{mf}(s) &= \frac{C(s)G(s)}{1 + C(s)G(s)} \\ C(s) &= \frac{1}{G(s)} \frac{G_{mf}(s)}{1 - G_{mf}(s)} \end{aligned} \tag{3.14}$$

Portanto, manipulando a equação como mostrado na equação (3.15), chegamos a equação caracteristica mostrada na equação (3.16).

$$C(s) = \frac{\dfrac{mls^2}{(-2ml^2)s^2 + mgl} \dfrac{-(2ml^2)s^2 + mgl}{s^2 + 2\zeta\omega_n s + \omega_n^2}}{\dfrac{s^2 + 2\zeta\omega_n s + \omega_n^2 - mgl + 2ml^2 s^2}{s^2 + 2\zeta\omega_n s + w_n^2}} \tag{3.15}$$

Fazendo $\omega_n^2 = mgl$, temos:

$$C(s) = \frac{mls^2}{(1 + 2ml^2)s^2 + 2\zeta\omega_n s} \tag{3.16}$$

Foi escolhido o tempo de acomodação (T_s) de 10 segundos para o critério de 2% de forma empírica, alterando esse valor e analisando o comportamento do sistema para o melhor entendimento do usuário.

Portanto, para essa escolha, foram obtidos os seguintes parâmetros:

- $T_s = 10$ s
- $\omega_n = 0{,}57$ rad/s
- $\zeta = 0{,}407$

Aplicando os valores de encontrados em (3.16), chegamos ao controlador mostrado em (3.17)

$$-0,0412s\frac{0,7676}{s+0,7676} \tag{3.17}$$

Por meio do gráfico de lugar das raízes da ferramenta sisotool do MATLAB, e considerando os parâmetros do sistema como comprimento da haste de 1 m, gravidade 9,78 m/s^2, massa do carro de 3 Kg, foram encontrados os seguintes parâmetros:

- $K_p = 514$
- $K_i = 710$
- $K_d = 92$
- $b = 1000$

3.3 Anti Wind-up aplicado nos controladores

Devido à limitação dos atuadores, foi necessário o tratamento do *wind-up* para que a saturação do sinal de controle fosse tratado.

O *wind-up* ocorre devido a limitações do sistema físico, as quais acabam afetando a ação do controlador PI, uma vez que devido a essas limitações ocorrem não linearidades no sistema. O que ocorre nessa não linearidade é que a ação integral vai acumulando o erro e gerando um sinal de controle num nível acima do que o atuador suporta (ou seja, ocorre uma saturação).

A estratégia de anti *windup* utilizada gera uma malha interna no controlador que só está ativa quando ocorre a saturação, garantindo assim que os valores do ganho integral não cresçam sem limites e consequentemente que o valor atual de saída do controlador reflita o que é realmente aplicado sobre o sistema.

Na Figura 4, observamos (como mostra a parte destacada) que no controlador da posição, controlador $C_e(s)$ em cascata com a planta $G_e(s)$ apresenta um parâmetro de controle praticamente unitário, e para que o controlador atuasse corretamente sobre a planta, o ângulo não poderia ultrapassar o limite escolhido.

O tratamento foi realizado utilizando um integrador anti-*wind-up*, como mostrado no diagrama de blocos na Figura 5 (parte destacada). Esse controle atua impondo um

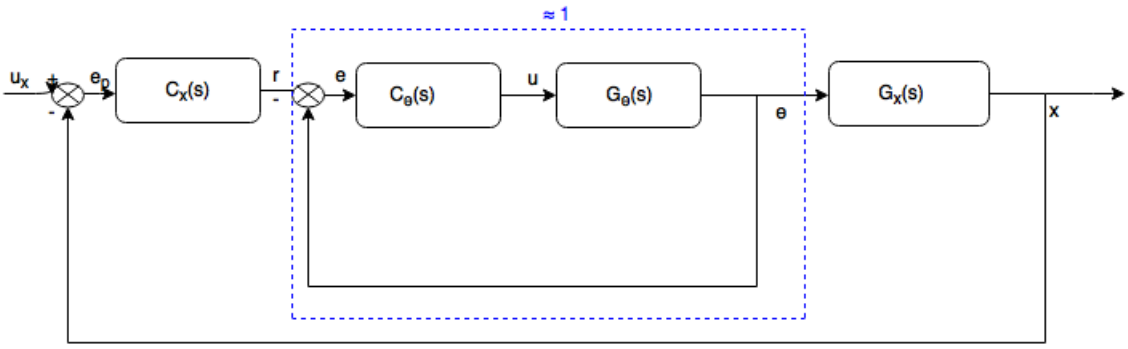

Figura 4 – Diagrama de blocos em cascata do controlador.

limite máximo para os possíveis valores do integrador, limitando assim o acúmulo de erro integral e evitando a saturação do atuador da planta.

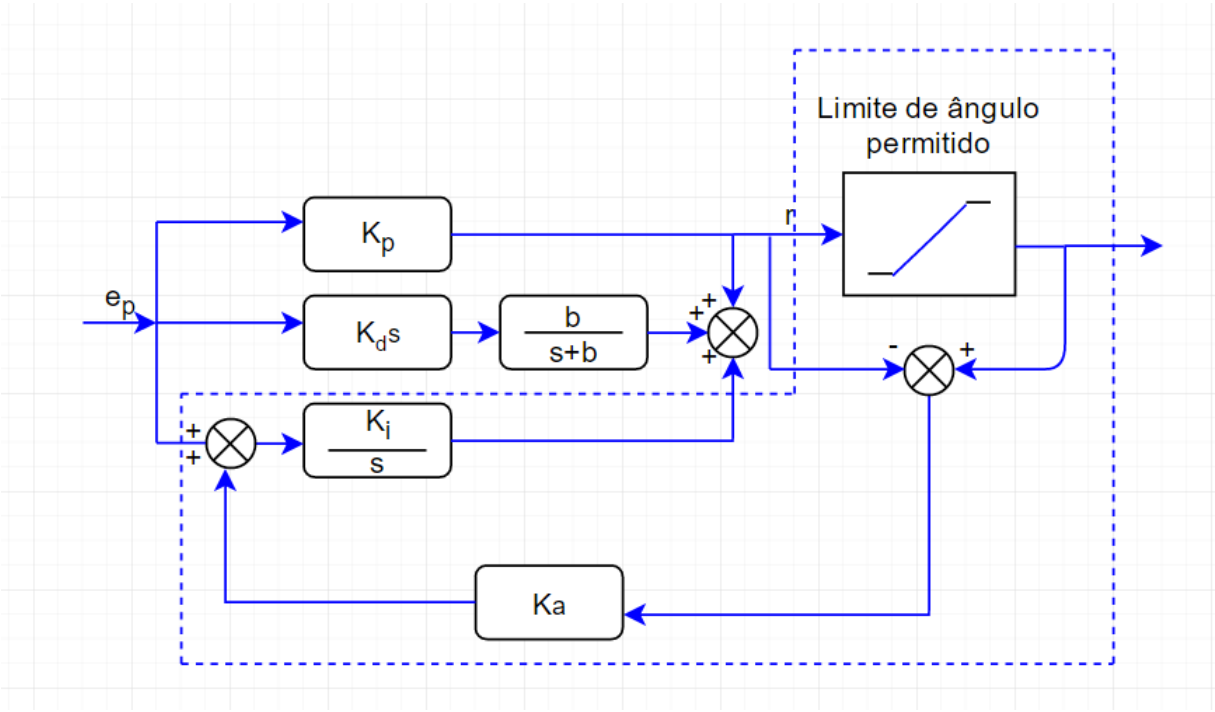

Figura 5 – Tratamendo anti *wind-up* utilizado representado em diagrama de blocos.

O tratamento do *wind-up* neste texto atua efetivamente somente na malha em que o sistema é controlado pela força, caso em que o controlador é do tipo proporcional integrativo. O controlador encontrado para atuar sobre o ângulo é do tipo proporcional derivativo, ou seja, não apresenta a parte integrativa. Apesar de não estar sendo utilizado o tratamento para a malha em que o controlador age sobre o ângulo, ele também foi implementado, mas a sua ação não afeta no processo em si atualmente. Futuramente, caso alguém continue a implementação desse *software*, existe a possibilidade da necessidade do tratamento anti *wind-up*, sendo necessário apenas que configure os parâmetros do controlador já implementado no algoritmo.

4 Desenvolvimento

Para a implementação do *software*, houve uma parceria com um projeto do programa Pró-Ativa da Universidade Federal de Ouro Preto, no qual foram implementados métodos de solução numérica de equações diferenciais ordinárias e que foram utilizados para simular o comportamento dos sistemas modelados. Vale ressaltar que o Unity, por ser uma plataforma de desenvolvimento de jogos, já possui métodos de simulação física, mas a utilização de tais métodos dificultaria a implementação das leis de controle.

4.1 Métodos matemáticos programados

Como citado anteriormente, houve o desenvolvimento de algoritmos com o intuito de termos mais liberdade com a simulação do que teríamos se fosse utilizada a física do Unity. Os métodos implementados foram:

i. Euler para frente (primeira ordem)

ii. Euler para trás (primeira ordem)

iii. Trapezoidal (segunda ordem)

iv. Adams-Moulton de terceira e quarta ordem

v. Runge-Kutta de quarta ordem

Para maiores informações sobre os métodos numéricos implementados, consulte (BUTCHER, 2008)

4.2 Algoritmos utilizados no Unity

Para o entendimento da vinculação do algoritmo ao *software*, deve-se ter conhecimento que o Unity trabalha com cenas (ou seja, cada tela possui uma configuração específica) e que o programa nos dá a possibilidade de acrescentar algoritmos diretamente nas cenas. Além disso, por questões de limitação do programa e lógica de programação, a referência utilizada para movimentação do carrinho (graficamente falando) foi feita na haste e não no carrinho. Devido a estratégia escolhida, fez-se necessário a vinculação direta da haste com o carrinho, fazendo com que assim o movimento gráfico fosse feito em cima do ponto fixo da haste (centro da circunferência descrita pelo movimento na parte inferior da haste) e não sobre o carrinho.

4.2.1 Algoritmo aplicado ao sistema em malha aberta

O algoritmo para a movimentação do carrinho em malha fechada é mostrado abaixo.

```
using System;
using UnityEngine;
using ODE;
using MathNet.Numerics.LinearAlgebra;
using UnityEngine.UI;

public class mudancaangulohaste : MonoBehaviour
{
private Transform carrinho;
private Transform pendulo;
private SistemaMA carrinhoMA;
private float aux;

public int ode_op;
public double dt;
public double m_carro, m_haste, g, meio_comprimento, ganho_forca;
public float ganho_x;
public float lim_esq;
public float lim_dir;

void Start()
{
carrinho = transform.parent.GetComponent<Transform>();
pendulo = GetComponent<Transform>();
carrinho.position = new Vector3(0, 178, 0);
pendulo.Rotate(Vector3.back, 360);
carrinhoMA = new SistemaMA(dt, ode_op, f);
carrinhoMA.x = new double[4];
carrinhoMA.t = 0;
}

public Vector<double> f(Vector<double> x, double t, Vector<double>
   u)
{
double[] xdot = new double[4];
xdot[0] = x[2];
xdot[1] = x[3];
xdot[2] = (2 * m_haste * meio_comprimento * Math.Sin(x[1]) * x[3] *
    x[3] + 2 * u[0] + m_haste * g * Math.Sin(x[1]) * Math.Cos(x[1])
```

```
    ) / (2 * (m_carro + m_haste) - m_haste * Math.Cos(x[1]) * Math.
    Cos(x[1]));
xdot[3] = (m_haste * meio_comprimento * Math.Sin(x[1]) * Math.Cos(x
    [1]) * x[3] * x[3] + Math.Cos(x[1]) * u[0] + (m_haste + m_carro)
     * g * Math.Sin(x[1])) / (2 * (m_carro + m_haste) - m_haste *
    Math.Cos(x[1]) * Math.Cos(x[1]));

Vector<double> x_dot = Vector<double>.Build.DenseOfArray(xdot);
return x_dot;
}

{
double T = Time.fixedDeltaTime;
double fx = Input.GetAxis("Horizontal");
double[] u = { ganho_forca * fx };

carrinhoMA.avancaTempo(T, u);
if (carrinhoMA.x[0] < lim_esq)
{
carrinhoMA.x[0] = lim_esq;
carrinhoMA.x[2] = -0.2 * carrinhoMA.x[2];
}

if (carrinhoMA.x[0] > lim_dir)
{
carrinhoMA.x[0] = lim_dir;
carrinhoMA.x[2] = -0.2 * carrinhoMA.x[2];
}
}

void Update()
{
carrinho.position = new Vector3(ganho_x * (float)carrinhoMA.x[0],
    178, 0);
pendulo.Rotate(Vector3.back, -aux);
double ang = -180 * carrinhoMA.x[1] / Math.PI;
pendulo.Rotate(Vector3.back, (float)ang);
aux = (float)ang;
}
}
```

O algoritmo aplicado para o movimento do carrinho é mostrado na Figura6. Como já mencionado anteriormente, todo o movimento e controle é feito sobre a haste do carrinho.

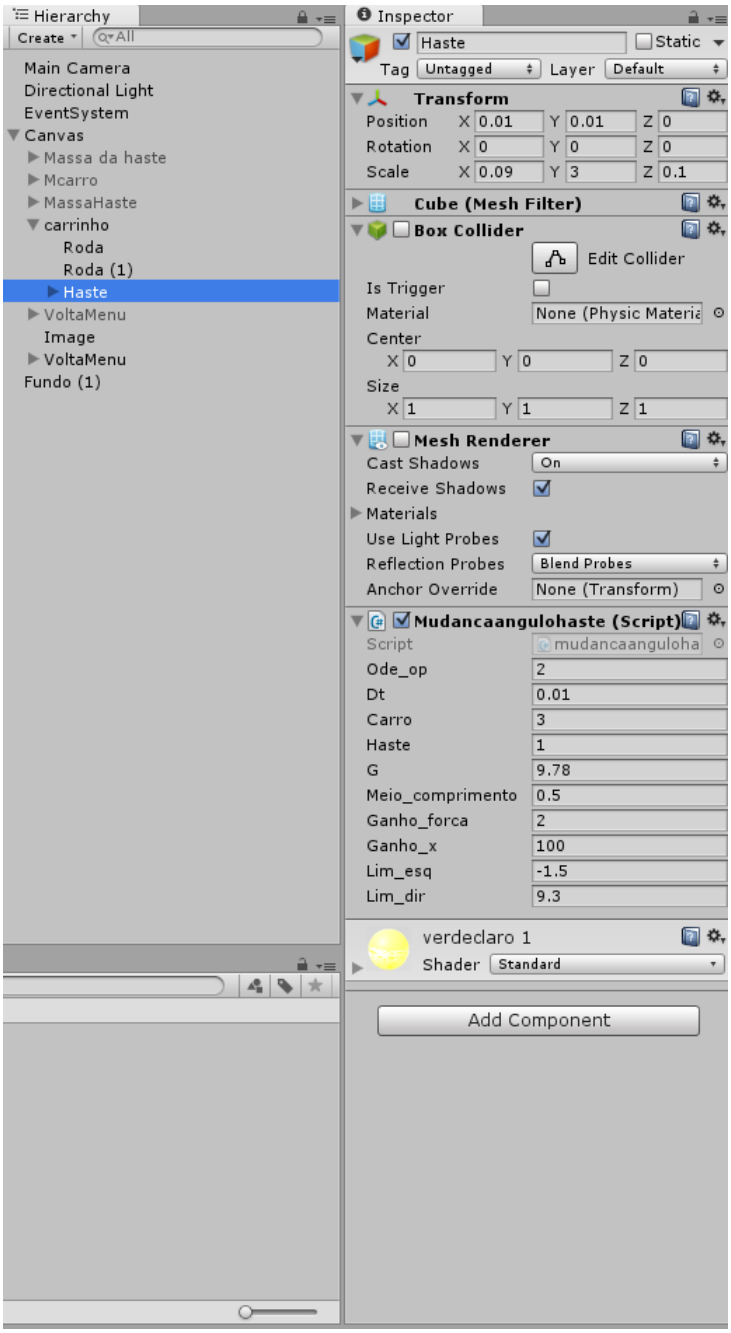

Figura 6 – Script aplicado ao Unity - sistema em malha aberta.

4.2.2 Algoritmo aplicado ao sistema em malha fechada - controlador de posição

O algorítimo utilizado para que o controle de posição fosse aplicado ao carrinho é mostrado abaixo.

```
1 using System;
2 using UnityEngine;
```

```
3  using ODE;
4  using MathNet.Numerics.LinearAlgebra;
5  using UnityEngine.UI;
6
7  public class Pendulomf : MonoBehaviour {
8
9  private Transform carrinho;
10 private Transform pendulo;
11 private SistemaMA carrinhoMA;
12 private float aux;
13 public int ode_op;
14 public double dt;
15 public double m_carro, m_haste, g, meio_comprimento, ganho_forca;
16 public float ganho_x;
17 public double ki;
18 public double kp;
19 public double kd;
20 public double b;
21 public float lim_esq;
22 public float lim_dir;
23
24 void Start()
25 {
26 carrinho = transform.parent.GetComponent<Transform>();
27 pendulo = GetComponent<Transform>();
28 carrinho.position = new Vector3(0, 178, 0);
29 pendulo.Rotate(Vector3.back, 360);
30 carrinhoMA = new SistemaMA(dt, ode_op, f);
31 carrinhoMA.x = new double[6];
32 carrinhoMA.t = 0;
33 }
34
35 public Vector<double> f(Vector<double> x, double t, Vector<double>
      r)
36 {
37 double e = r[0] - x[1];
38 double u = (kp + kd * b) * e + ki * x[4] + x[5];
39 double[] xdot = new double[6];
40 xdot[0] = x[2];
41 xdot[1] = x[3];
42 xdot[2] = (2 * m_haste * meio_comprimento * Math.Sin(x[1]) * x[3] *
       x[3] + 2 * u + m_haste * g * Math.Sin(x[1]) * Math.Cos(x[1])) /
```

```
    (2 * (m_carro + m_haste) - m_haste * Math.Cos(x[1]) * Math.Cos(
    x[1]));
xdot[3] = (m_haste * meio_comprimento * Math.Sin(x[1]) * Math.Cos(x
    [1]) * x[3] * x[3] + Math.Cos(x[1]) * u + (m_haste + m_carro) *
    g * Math.Sin(x[1])) / (2 * (m_carro + m_haste) - m_haste * Math.
    Cos(x[1]) * Math.Cos(x[1]));
xdot[4] = e;
xdot[5] = -b * x[5] - kd * b * b * e;
Vector<double> x_dot = Vector<double>.Build.DenseOfArray(xdot);
return x_dot;
}

void FixedUpdate()
{
double T = Time.fixedDeltaTime;
double fx = Input.GetAxis("Horizontal");
double[] r = { ganho_forca * fx };
carrinhoMA.avancaTempo(T, r);

if (carrinhoMA.x[0] < lim_esq)
{
carrinhoMA.x[0] = lim_esq;
carrinhoMA.x[2] = -0.2 * carrinhoMA.x[2];
}

if (carrinhoMA.x[0] > lim_dir)
{
carrinhoMA.x[0] = lim_dir;
carrinhoMA.x[2] = -0.2 * carrinhoMA.x[2];
}
}

void Update()
{
carrinho.position = new Vector3(ganho_x* (float)carrinhoMA.x[0],
    178, 0);
pendulo.Rotate(Vector3.back, -aux);
double ang = -180 * carrinhoMA.x[1] / Math.PI;
pendulo.Rotate(Vector3.back, (float)ang);
aux = (float)ang;
}
}
```

O algoritmo aplicado para esse controlador é mostrado na Figura7. Como já mencionado anteriormente, todo o movimento e controle é feito sobre a haste do carrinho.

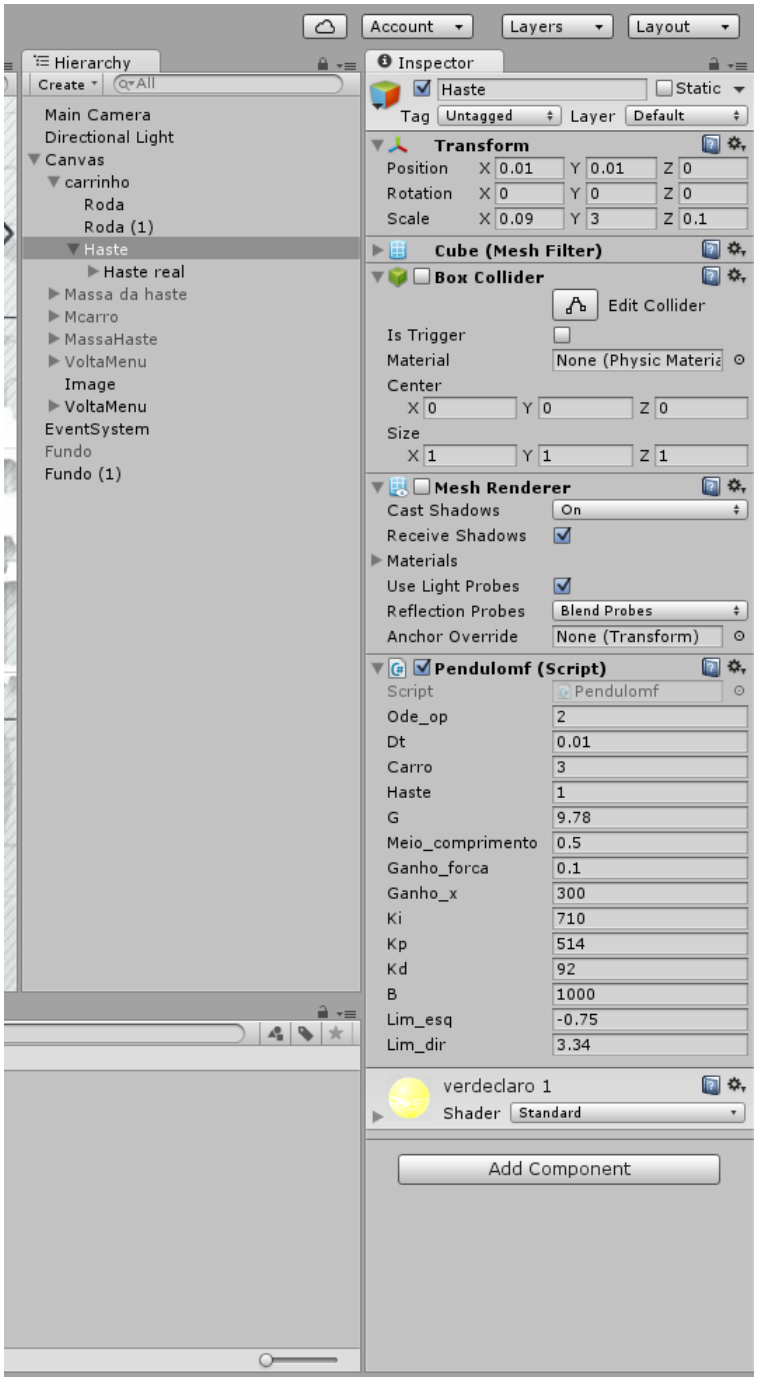

Figura 7 – Script aplicado ao Unity - controlador atuando sobre a posição.

4.2.3 Algoritmo aplicado ao sistema em malha fechada - controlador de ângulo

O algorítimo utilizado para que o controle de ângulo fosse aplicado ao carrinho é mostrado abaixo.

```
using System;
using UnityEngine;
using ODE;
using MathNet.Numerics.LinearAlgebra;
using UnityEngine.UI;

public class mfpendulo : MonoBehaviour {

private Transform carrinho;
private Transform pendulo;
private SistemaMA carrinhoMA;
private float aux;
public int ode_op;
public double dt;
public double m_carro, m_haste, g, meio_comprimento;
public float ganho_x;
public double ki;
public double kp;
public double kd;
public double b;
public double kip;
public double kpp;
public double kdp;
public double bp;
public double ka;
private double refere;
public float posiy;
public float posix;
public float lim_esq;
public float lim_dir;

void Start()
{
carrinho = transform.parent.GetComponent<Transform>();
pendulo = GetComponent<Transform>();
carrinho.position = new Vector3(0, 178, 0);
pendulo.Rotate(Vector3.back, 360);
carrinhoMA = new SistemaMA(dt, ode_op, f);
carrinhoMA.x = new double[8];
carrinhoMA.x[0] = posix / ganho_x;
refere = carrinhoMA.x[0];
carrinhoMA.t = 0;
```

```
43 }
44
45 public Vector<double> f(Vector<double> x, double t, Vector<double>
      rp)
46 {
47 double ep = rp[0] - x[0];
48 double r = (kpp + kdp * bp) * ep + kip * x[6] + x[7];
49 double raux = r;
50
51 if (r > Math.PI / 6.0)
52 r = Math.PI / 6.0;
53
54 if (r < -Math.PI / 6.0)
55 r = -Math.PI / 6.0;
56
57 double er = r - raux;
58 double e = r - x[1];
59 double u = (kp + kd * b) * e + ki * x[4] + x[5];
60 double[] xdot = new double[8];
61
62 xdot[0] = x[2];
63 xdot[1] = x[3];
64 xdot[2] = (2 * m_haste * meio_comprimento * Math.Sin(x[1]) * x[3] *
       x[3] + 2 * u + m_haste * g * Math.Sin(x[1]) * Math.Cos(x[1])) /
       (2 * (m_carro + m_haste) - m_haste * Math.Cos(x[1]) * Math.Cos(
      x[1]));
65 xdot[3] = (m_haste * meio_comprimento * Math.Sin(x[1]) * Math.Cos(x
      [1]) * x[3] * x[3] + Math.Cos(x[1]) * u + (m_haste + m_carro) *
      g * Math.Sin(x[1])) / (2 * (m_carro + m_haste) - m_haste * Math.
      Cos(x[1]) * Math.Cos(x[1]));
66 xdot[4] = e;
67 xdot[5] = -b * x[5] - kd * b * b * e;
68 xdot[6] = ep + ka*er;
69 xdot[7] = -bp * x[7] - kdp * bp * bp * ep;
70
71 Vector<double> x_dot = Vector<double>.Build.DenseOfArray(xdot);
72
73 return x_dot;
74 }
75 void FixedUpdate()
76 {
77 double T = Time.fixedDeltaTime;
```

```
78  double fx = Input.GetAxis("Horizontal");
79  double[] rp = { refere };
80  carrinhoMA.avancaTempo(T, rp);
81  if (carrinhoMA.x[0] < lim_esq)
82  {
83  carrinhoMA.x[0] = lim_esq;
84  carrinhoMA.x[2] = -0.2*carrinhoMA.x[2];
85  }
86
87  if (carrinhoMA.x[0] > lim_dir)
88  {
89  carrinhoMA.x[0] = lim_dir;
90  carrinhoMA.x[2] = -0.2*carrinhoMA.x[2];
91  }
92  }
93  void Update()
94  {
95  carrinho.position = new Vector3(ganho_x* (float)carrinhoMA.x[0],
       posiy, 0);
96  pendulo.Rotate(Vector3.back, -aux);
97  double ang = -180 * carrinhoMA.x[1] / Math.PI;
98  pendulo.Rotate(Vector3.back, (float)ang);
99  aux = (float)ang;
100
101 if (Input.GetMouseButton(0))
102 {
103 refere = Input.mousePosition.x/ganho_x;
104 }
105 }
106 }
```

O algoritmo aplicado para esse controlador é mostrado na Figura 8. Como já mencionado anteriormente, todo o movimento e controle é feito sobre a haste do carrinho.

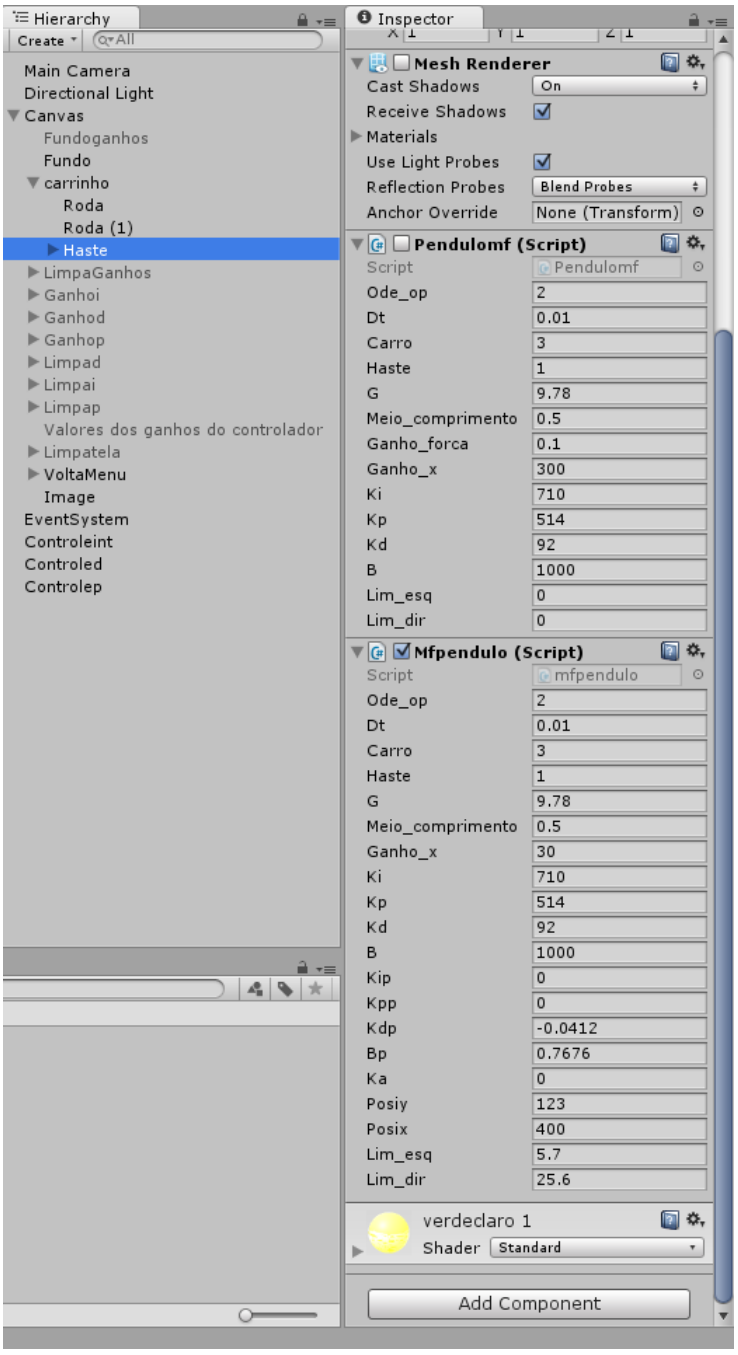

Figura 8 – Script aplicado ao Unity - controlador atuando sobre o ângulo.

5 Utilização do *software*

Essa seção é voltada para o aprendizado da manipulação do *software*, explicada de forma detalhada e simples. O *software* foi desenvolvido para ser executado no sistema operacional Windows.

5.1 Inicializando o *software*

Há várias possibilidades para inicializar a execução do *software*. Porém duas serão exemplificadas aqui. Você pode abrir o programa clicando duas vezes no ícone do programa, que é mostrado na Figura 9 ou clicando com o botão direito e, em seguida, selecionando a opção abrir, como mostrado na Figura 10

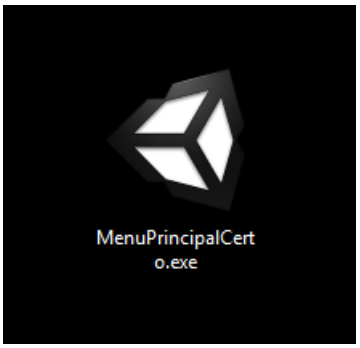

Figura 9 – Ícone do *software*.

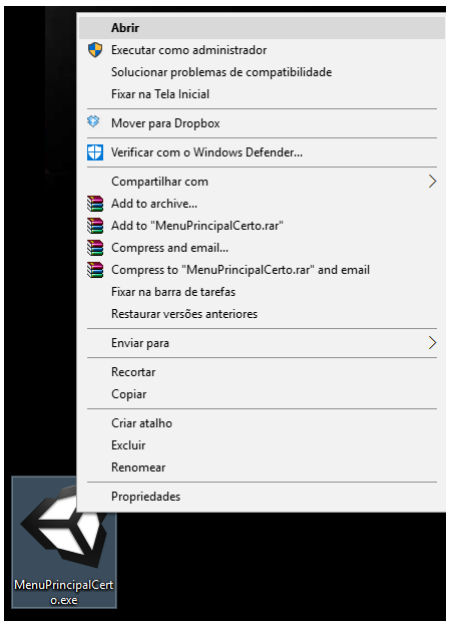

Figura 10 – Segunda forma de abrir o *software*.

Feito isso, será apresentada a tela mostrada na Figura 11. A resolução da tela vem com o valor de 1024×768 pixels. Essa resolução é a indicada para execução, uma vez que o *software* foi todo desenvolvido em cima dessa resolução, para que assim ele executasse de forma correta (sem bugs visuais) num maior número de computadores. Também aconselha-se deixar a caixa "Windowed "selecionada, pois o *software* faz ajustes de cores e *frames* de acordo com o processamento do computador de forma mais eficiente. Além disso, é aconselhado deixar a qualidade dos gráficos em "fantástico ", pois o programa não exige um processamento muito elevado da máquina. Caso o computador utilizado possua mais de um monitor para exibição da imagem, o usuário pode escolher em qual monitor exibir a tela do *software*.

O botão "Quit "encerra a janela e não executa o programa e o botão "Play! "inicializa o *software*.

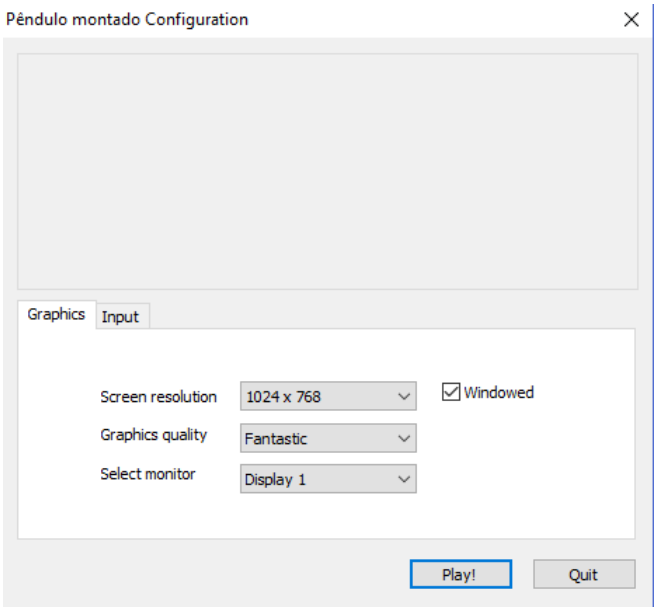

Figura 11 – Tela para a configuração da resolução.

5.2 Operando no menu principal

A tela principal do programa é mostrada na Figura 12. Nessa tela, podemos observar que há dois botões de acesso: o botão "Pêndulo Invertido "e o botão "Sair do programa ". O segundo botão encerra o programa em geral, sendo necessário novamente a abertura do mesmo caso queira analisar algum dos modelos.

Ao clicar no botão Pêndulo invertido, a tela mostrada na Figura 13 é apresentada ao usuário.

Figura 12 – Menu principal do *software*.

5.3 Tela de escolha do tipo de malha a se analisar

Nessa tela, observamos a presença de 3 botões. O botão "Voltar para o menu "está presente em todas as telas (exceto na tela do menu principal) e sua função é retornar para o menu principal de forma direta, seja com o intuito de executar a análise de outro modelo ou de encerrar o programa.

O botão "Pêndulo MA ", abre a tela para a execução do pêndulo invertido em malha aberta, que é explicado na seção 5.4. O botão "Pêndulo MF "abre ao usuário uma tela de configuração dos parâmetros dos controladores para que eles sejam utilizados nos controladores, como explicado na seção 5.5.

Figura 13 – Menu para a escolha entre análise em malha aberta ou em malha fechada.

5.4 Pêndulo em malha aberta

Ao escolher a opção para a análise em malha aberta mencionado na seção 5.3 , a tela mostrada na Figura 14 é exibida.

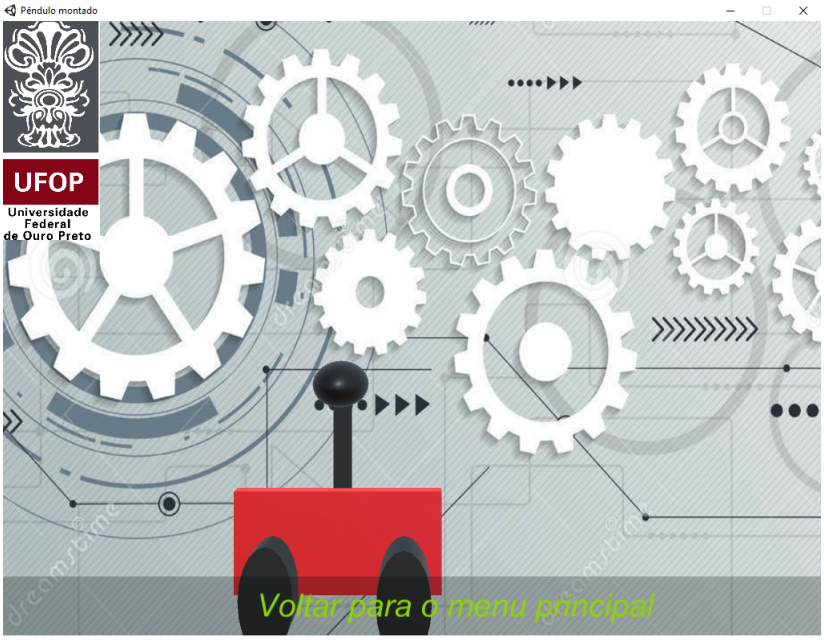

Figura 14 – Tela do pêndulo invertido em malha aberta.

O intuito dessa análise é que o aluno tente controlar o sistema em malha aberta utilizando as setas direcionais do teclado. Dificilmente isso será possível, uma vez que a dinâmica do sistema é muito complexa e sensível (em se tratando no tempo de reação) ao cérebro humano e também aos comandos impostos pelo usuário. Esse processo é extremamente importante para que o raciocínio empírico em cima dos modelos seja desenvolvido pelos alunos, para que ao fim do processo, vejam o poder e a importância de se desenvolver controladores.

5.5 Pêndulo em malha fechada

Ao escolher a opção para a análise em malha fechada mencionado na seção 5.3 , a tela mostrada na Figura 15 é exibida. Nessa tela, há duas opções para o usuário: inserir os ganhos dos controladores e analisar o resultado ou analisar os modelos com os ganhos calculados no Capítulo 3.

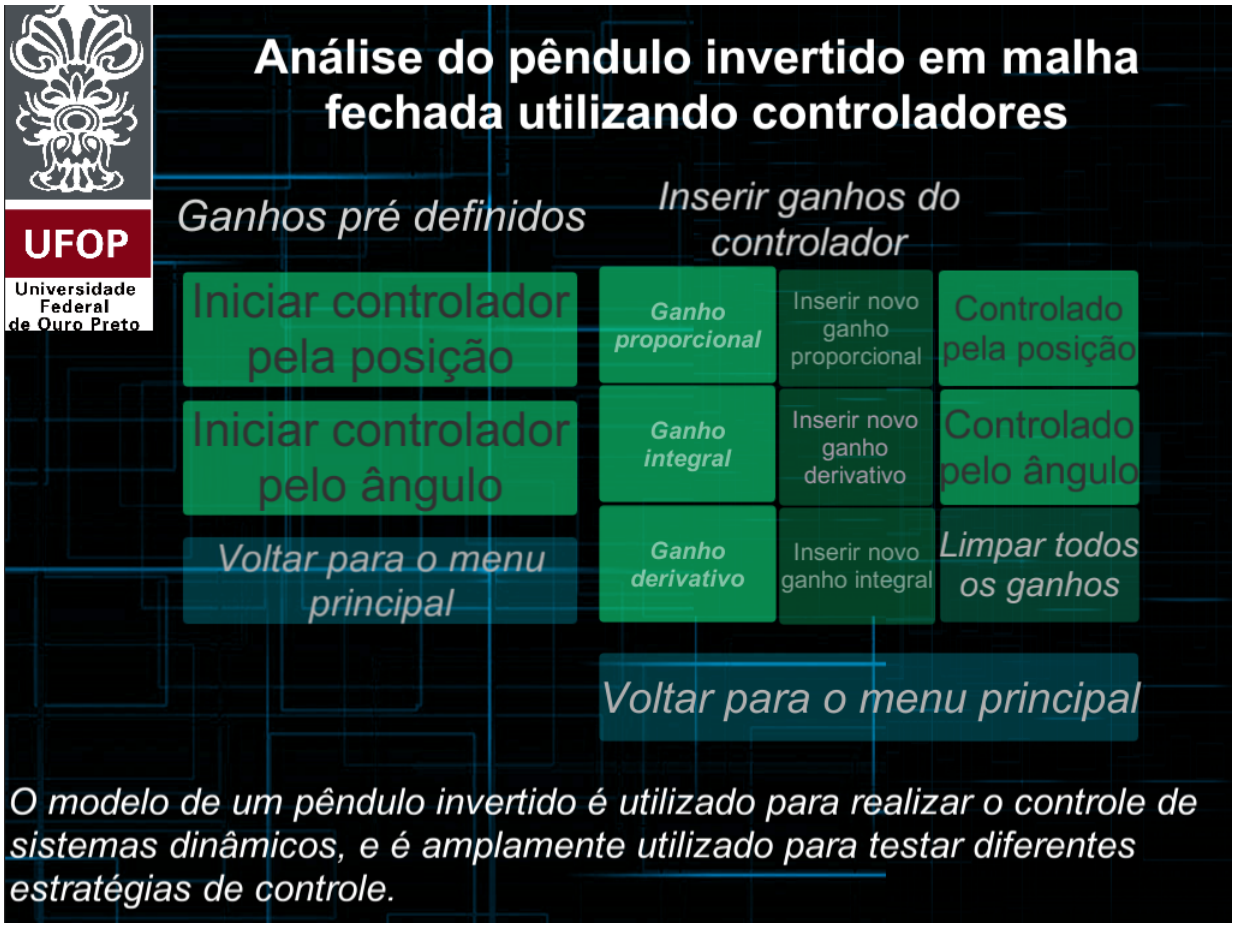

Figura 15 – Tela de configuração dos controladores.

5.5.1 Análise com os ganhos pré-definidos

Caso o usuário escolha essa opção, ele poderá escolher entre o controlador de ângulo e controlador de posição. Ambos os sistemas apresentam a tela mostrada na Figura 16.

Caso o usuário escolha o controlador de posição, o mesmo deve selecionar um ponto na tela (na direção horizontal) e clicar com o botão esquerdo do mouse. Instantaneamente o carro irá se mover em direção ao ponto clicado (a força resultante com direção e sentido ao ponto) e o controlador irá atuar sobre o mesmo, até que o equilíbrio seja alcançado.

Caso o usuário escolha o controlador que atua sobre o ângulo da haste, o carrinho deve ser movimentado pelas setas direcionais do teclado que determinam a referência de ângulo para o controlador. Da mesma forma ao caso anterior, os valores parâmetros utilizados no controlador são os calculados no Capítulo 3.

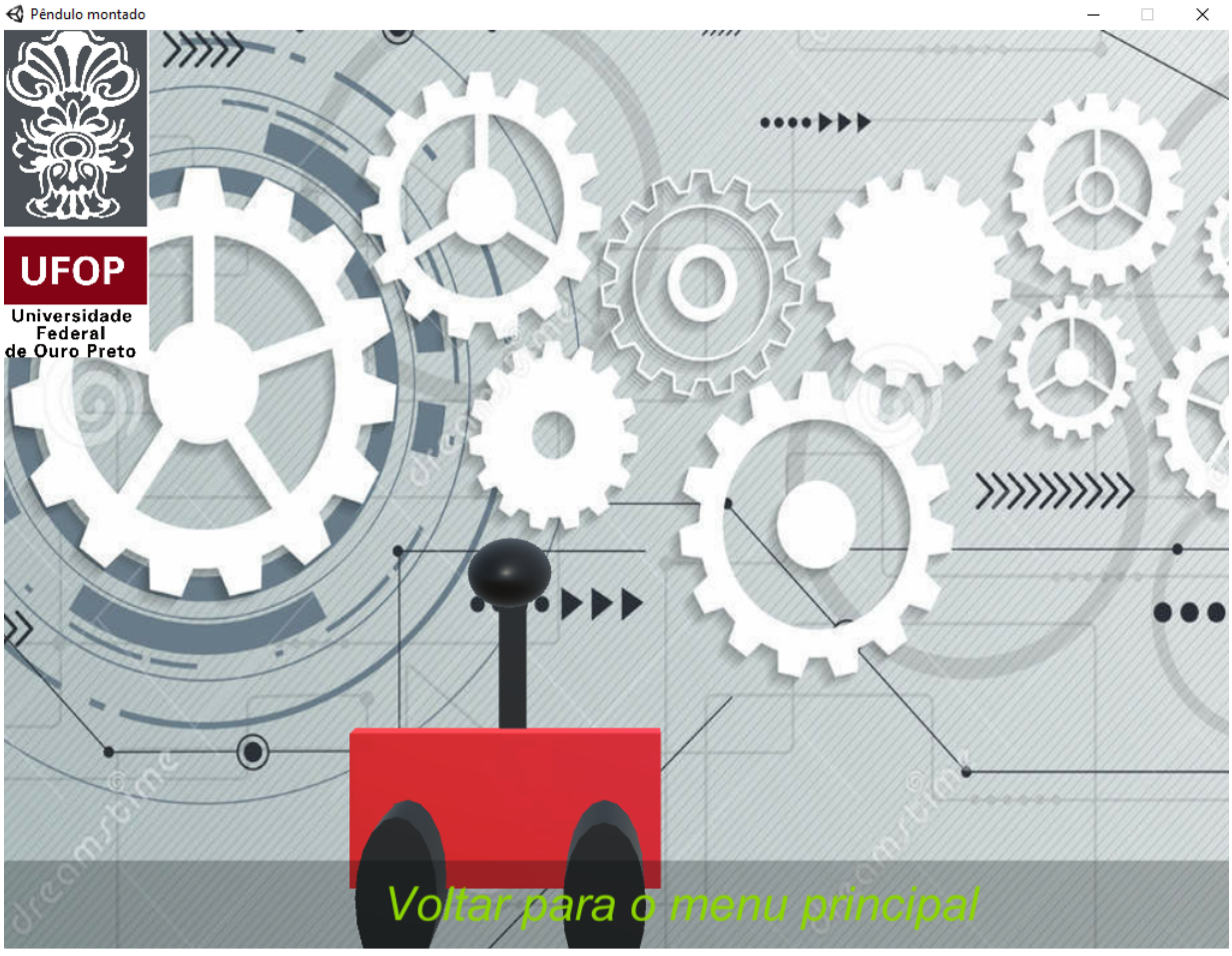

Figura 16 – Tela do controle em malha fechada - controlador atuando pela força.

5.5.1.1 Análise com os ganhos definidos pelo usuário

Nessa opção, o usuário entra com os valores dos controladores, podendo também escolher se o controlador atua sobre o ângulo ou sobre a posição.

Na Figura 15, observamos que temos três campos para serem preenchidos: um para o ganho proporcional, uma para o ganho derivativo e um para o ganho integral. Caso o usuário erre na hora de digitar, ele tem a opção de limpar o campo para inserir o ganho corretamente, clicando em inserir novo ganho no tipo em que ele desejar. Após serem inseridos os ganhos, o usuário escolhe em qual controlador atuar, pelos botões "Controlado pela força "e "Controlado pelo ângulo ". Caso o usuário optar por entrar com os parâmetros referentes ao controlador de posição, o usuário deverá especificar os ganhos do PID de ângulo e os ganhos do PID de posição.

É nesse ponto que o o raciocínio empírico é finalizado, pois ele passa por todos os casos possíveis, e entende a importância de se ter os parâmetros corretos do controlador.

6 Conclusão

A implementação desse *software* mostrou o quão eficaz e simples se torna a teoria de controle aliada as técnicas computacionais. O desenvolvimento de modelos matemáticos na área de controle podem ser facilmente implementados em algoritmos considerando as grandezas físicas reais, as quais podem ser facilmente manipuladas. Além disso, a escolha de utilizar um motor de jogos 3D trouxe as representações dos modelos mais próximos a realidade.

Os *softwares* escolhidos para o desenvolvimento se mostraram de alta confiabilidade, não deixando nada a desejar nos quesitos desempenho e eficiência. Além disso, devido à experiência adquirida com a utilização dos *softwares*, conclui-se também que outros modelos podem ser implementados futuramente e até mesmo o aprimoramento do já implementado. A independência de *softwares* pagos se mostrou possível e de forma eficaz, uma vez que a implementação dos modelos pode ser realizada de forma específica de acordo com o interesse do desenvolvedor.

A expectativa é que os alunos adquiram um ganho de conhecimento que não seria possível sem a utilização desse *software* na universidade. O aperfeiçoamento do modelo desenvolvido e programado poderá ser realizado em cima das opiniões dos alunos que utilizarem o mesmo, além de solucionar possíveis erros e falhas não encontrados até o presente momento.

Referências

ÅSTRÖM, K. J.; FURUTA, K. Swinging up a pendulum by energy control. *Automatica*, Elsevier, v. 36, n. 2, p. 287–295, 2000.

BOUBAKER, O. The inverted pendulum: A fundamental benchmark in control theory and robotics. In: IEEE. *Education and e-Learning Innovations (ICEELI), 2012 international conference on*. [S.l.], 2012. p. 1–6.

BUTCHER, J. C. *Numerical Methods for Ordinary Differential Equation*. second edition. [S.l.]: John Wiley & Sons, Ltd, 2008.

COLLER, B. D.; SCOTT, M. J. Effectiveness of using a video game to teach a course in mechanical engineering. *Computers & Education*, Elsevier, v. 53, n. 3, p. 900–912, 2009.

DEMIRTAS, M.; ALTUN, Y.; ISTANBULLU, A. An educational virtual laboratory for sliding mode and pid control of inverted pendulum. In: IEEE. *Optimization of Electrical and Electronic Equipment, 2008. OPTIM 2008. 11th International Conference on*. [S.l.], 2008. p. 149–156.

HOVLAND, G. Evaluation of an online inverted pendulum control experiment. *IEEE Transactions on Education*, IEEE, v. 51, n. 1, p. 114–122, 2008.

OGATA, K. *Engenharia de Controle Moderno*. third edition. [S.l.]: LTC - Livros Técnicos e Científicos Editora S.A., 2000.

SHERNOFF, D. J.; COLLER, B. D. A quasi-experimental comparison of learning and performance in engineering education via video game versus traditional methods. 2013.