Instituto Tecnológico de Aeronáutica

Pós-Graduação em Engenharia Aeronáutica e Mecânica

Desenvolvimento de Interface Interativa para Simulações de Controle em Unity3D

Relatório Final da disciplina AA-600 - Estágio de Docência.

Autor: João Filipe Renó Peixoto de Azevedo Silva

São José dos Campos, 3 de dezembro de 2020

Informações Gerais do Estágio de Docência

- Título do projeto:

  Desenvolvimento de Interface Interativa para Simulações de Controle em Unity3D

- Nome do autor:

  João Filipe Renó Peixoto de Azevedo Silva

- Instituição sede:

  Pós-Graduação em Engenharia Aeronáutica e Mecânica do Instituto Tecnológico de Aeronáutica

- Curso:

  Pós-Graduação em Engenharia Aeronáutica e Mecânica

- Área:

  EAM-1 - Projeto Aeronáutico, Estrururas e Sistemas Aeroespaciais

- Disciplina:

  MPS43 - Sistemas de Controle

- Período coberto por este relatório:

  Agosto/2020 a dezembro/2020

Resumo

O objetivo deste relatório é descrever as atividades realizadas durante o semestre para o desenvolvimento de uma interface visual que represente um ensaio laboratorial de Sistemas de Controle. Utilizando uma planta de um carro sobre trilho conectado a um pêndulo, simulações ilustram seu comportamento frente a distúrbios. Agrega-se então às simulações uma representação virtual do ensaio, modelada em Unity3D, que visa tornar mais imersiva e interativa a visualização do experimento[1].

[1]Todos os algoritmos e o simulador serão incluídos aos anexos deste relatório e também estarão disponíveis em: `https://github.com/jfilipe33/EstDoc`

Abstract

The goal of this report is to describe the activities carried out throughout the semester for the development of a visual interface that represents a Control Systems laboratory experiment. Employing a plant consisted by a cart on a rail connected to a pendulum, simulations illustrate its behavior in the face of disturbances. A virtual representation of the test is then added to the simulations, modeled in Unity3D, which aims to make the visualization of the experiment more immersive and interactive.

Sumário

1 Introdução

O estudo de teorias de controle, assim como de diversas outras disciplinas, proporciona melhor retenção dos conceitos discutidos em aula quando complementado por experimentos práticos. Por meio de softwares de processamento numérico, como o Matlab, é possível ao pesquisador simular sistemas dinâmicos a partir da representação do seu modelo matemático em um algoritmo computacional. Desta forma, o estudante de controle consegue ver graficamente os efeitos da variação de parâmetros do modelo, bem como avaliar, qualitativa ou quantitativamente, propriedades como estabilidade, tempo de acomodação, *overshoot*, entre outras.

As aulas de laboratório são grandes aliadas no ensino de conceitos teóricos, pois permitem aos discentes o contato, físico ou visual, com as plantas sobre as quais o conteúdo será estudado. Tal contato instiga a curiosidade, incita dúvidas e gera a oportunidade do aluno se aprofundar no assunto. Entretanto, o acesso a tais equipamentos nem sempre é possível. Isso foi agravado, na época de publicação deste trabalho, devido à imposição de isolamento social dada a pandemia de COVID-19. Criou-se então a necessidade de adaptar as aulas, teóricas e práticas, para que os discentes pudessem acompanhar o conteúdo lecionado à distância.

O avanço de tecnologias de processamento digital de imagens é majoritariamente percebido em mídias como jogos e filmes. Tal avanço, mais recentemente, trouxe também a popularização de *game engines* mais acessíveis para o desenvolvimento de projetos independentes, como a Unity. Por meio desta, o usuário pode criar ambientes virtuais bidimensionais ou tridimensionais, para serem usados em jogos, simulações, e outros experimentos. A *engine*, que alia ambientes de desenvolvimento gráfico com rotinas de programação, dá ao desenvolvedor o ferramental para representar visualmente modelos matemáticos convertidos em algoritmos, tornando possível simular um ambiente laboratorial virtual com os equipamentos necessários para fazer ensaios similares aos existentes na instituição.

Sendo assim, é possível criar um ambiente virtual tridimensional que simula a experiência de estar em um laboratório de controle e, nele, poder realizar ensaios capazes de representar de forma fidedigna o comportamento de sistemas reais. Entretanto, o ferramental fornecido pelo Matlab permite análises mais precisas dos dados dos ensaios e também tem grande valia para o discente. Desta forma, o foco deste trabalho é recriar o ambiente e o equipamento de laboratório virtualmente no Unity, modelar e reproduzir um experimento em Matlab, e criar um canal de comunicação entre as duas plataformas de modo que a representação gerada ilustre ao usuário, de forma imersiva e interativa, o comportamento descrito pela simulação matemática do sistema.

2 Desenvolvimento

Esta seção conterá a descrição da planta de teste, seu modelo matemático e os algoritmos de Matlab que representam seu funcionamento. Contida aqui também estará a descrição das rotinas de programação em Unity, bem como a descrição do desenvolvimento do ambiente virtual criado.

2.1 Planta de Testes

Adotou-se para este estudo uma sistema de Pêndulo Invertido. Tal sistema é composto por um carro sobre um trilho, e uma barra rígida conectada ao carro por um eixo, conforme ilustrado na figura 2.1.

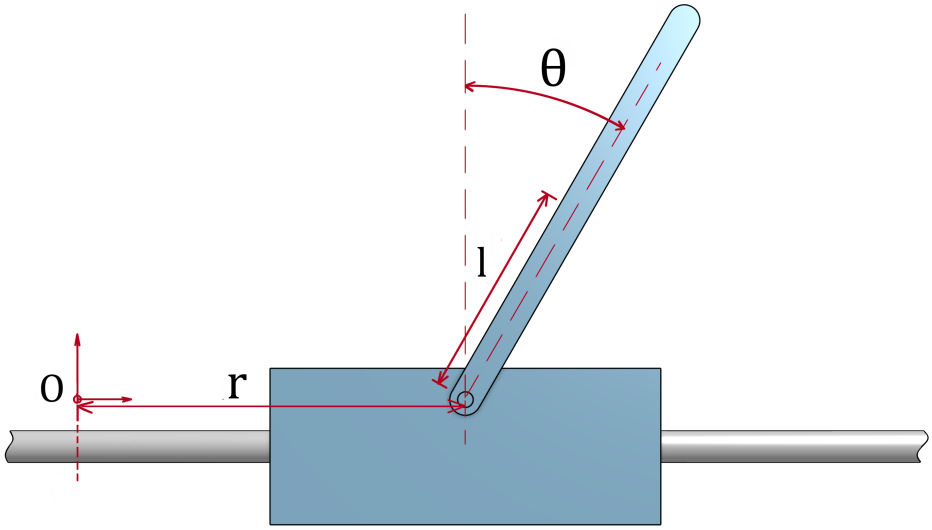


Figura 2.1: Esquema ilustrativo da planta de Pêndulo Invertido.

A planta acima possui dois graus de liberdade. A posição no eixo horizontal, medida entre a origem do sistema O e o centro do carro, é denominada r. Já o ângulo de inclinação do pêndulo, medido a partir de uma reta paralela ao eixo vertical do sistema O, é denominado θ. Importante salientar que foi adotado como $\theta = 0$ o angulo em que o pendulo se encontra equilibrado para cima. Além das variáveis supracitadas, os parâmetros utilizados no equacionamento são descritos na Tabela 2.1.

Tabela 2.1: Parâmetros da Planta

Denominação	**Descrição**	**Valor**
m	Massa do Pêndulo	$0.150\ kg$
M	Massa do Carro	$0.8\ kg$
l	Comprimento do Pêndulo	$0.3\ m$
g	Aceleração da Gravidade	$9.81\ m/s^2$
μ_s	Coef. de Atrito Estático	0.08
μ_d	Coef. de Atrito Dinâmico	0.04
ε	Coef. de Atrito Viscoso	$2.5\ kg/s$
I	Momento de Inércia do Pêndulo	$0.0045\ kgm^2$

A dinâmica da planta é descrita pelo conjunto de equações (2.1).

$$\begin{aligned}&(I+ml^2)\ddot{\theta}+mlcos(\theta)\ddot{r}-mglsen(\theta)=0\\&(M+m)\ddot{r}+ml(cos(\theta)-\mu sen(\theta))\ddot{\theta}-ml(sen(\theta)+\mu cos(\theta))\dot{\theta}^2=Fc-Fa+Fd\end{aligned} \tag{2.1}$$

A fim de adaptar as equações de movimento para o algoritmo em Matlab, estas foram modificadas para a forma matricial em (2.2).

$$\begin{bmatrix}\ddot{r}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}M+m & ml(cos(\theta)-\mu sen(\theta))\\ lmcos(\theta) & I+ml^2\end{bmatrix}^{-1}\begin{bmatrix}ml(sen(\theta)+\mu cos(\theta))\dot{\theta}^2+Fa+Fc+Fd\\ mlgsen(\theta)\end{bmatrix} \tag{2.2}$$

onde Fc é a força de comando, Fa a força de atrito, e Fd uma força de distúrbio. Para mais informações, o equacionamento completo, incluindo a representação em espaço de estados, será incluído nos anexos.

Para o cálculo da força de atrito, tomou-se como base o trabalho de Campbell *et al.* [1]. Nele, as autoras equacionam o modelo do atrito entre o carro e o trilho como:

$$Fa=\begin{cases}F_{static} & \text{se } \dot{r}=0,\\ F_{coulomb}+F_{viscoso} & \text{se } \dot{r}\neq 0.\end{cases} \tag{2.3}$$

Sabendo que F_N é a força de reação normal à força peso, os componentes da equação (2.3) são definidos por:

$$F_{estatico}=\begin{cases}-Fc & \text{se } |Fc|<\mu_s F_N\\ -\mu_s F_N sign(Fc) & \text{se } |Fc|\geq\mu_s F_N.\end{cases}, \tag{2.4}$$

$$F_{coulomb}=-\mu_c F_N sign(\dot{r}), \tag{2.5}$$

$$F_{viscoso}=-\varepsilon\dot{r}. \tag{2.6}$$

A fim de testar o modelo da planta, este também foi simulado com valores de atrito nulos e aproximados a valores puramente proporcionais à velocidade do carro.

Definido o equacionamento do modelo, é necessário reproduzí-lo em Matlab para dar início às simulações.

2.2 Algoritmos em Matlab

A rotina de programação se inicia atribuindo aos parâmetros do sistema seus respectivos valores descritos na Tabela 2.1. O sistema parte de condições iniciais nulas, com exceção do ângulo do pêndulo. Este partirá de $\theta = 0.05\pi$, de modo a tirá-lo da estabilidade.

O algoritmo é fechado em um ciclo temporal que itera a cada período de $T_s = 0.02s$ por 15 segundos. Neste intervalo, simula-se o comportamento do modelo descrito pelas equações diferenciais em (2.2). Para uma força de comando $Fc = 0$ durante todo o tempo do ciclo, espera-se que o pêndulo caia. Caso não haja força de atrito, o ângulo θ deverá variar senoidalmente sem perda de energia. Consequentemente, a ação do movimento giratório da haste acarretará em um deslocamento do carro sobre o trilho.

Já quando aplicada tal força, apesar de o pêndulo e o carro se deslocarem de modo similar, a ação do atrito reduz a amplitude das variações consecutivas da haste, até que esta se estabilize em seu ponto de equilíbrio em $\theta = \pi$ e o carro pare.

Foi definida também uma lei de controle de realimentação de estados da forma

$$Fc = g_1 r + g_2 \theta + g_3 \dot{r} + g_4 \dot{\theta}, \tag{2.7}$$

cujos ganhos sugeridos para diferentes respostas estão listados na Tabela 2.2.

Tabela 2.2: Ganhos de Realimentação

Denominação	**Resposta Lenta**	**Resposta Rápida**
g_1	50	70
g_2	100	140
g_3	20	40
g_4	26	26

Quando sob a ação da lei de controle (2.7), a força Fc é exercida sobre o carro com o objetivo de estabilizar ambos r e θ em zero. Para colocar à prova a eficácia da compensação, forças de distúrbio podem ser aplicadas pontualmente durante a simulação. A lei de controle deverá ser o suficiente para reestabilizar o sistema perturbado, desde que $|Fd| \leq 300N$.

Além da conversão do equacionamento supracitado para algoritmo, é necessário utilizar um método numérico para solucionar as equações diferenciais em (2.2). O método escolhido foi o de Runge-Kutta, ou comumente conhecido como RK4 [2]. O algoritmo completo, bem como todas as subrotinas de funções, serão adicionadas à seção de anexos deste relatório.

Ao final de cada iteração, deseja-se enviar para a plataforma Unity os estados r e θ, para que este possa reproduzir virtualmente o funcionamento da planta. Para tal, criou-se um canal de comunicação via TCP/IP. A cada iteração do loop, o Matlab, que exerce a função de cliente, abre o canal, acondiciona os dados em uma sequência de caracteres de texto, envia estes dados e fecha a comunicação. Foi adicionado também um comando

de pausa, com duração de $0.02s$, para garantir que haja tempo hábil para o Unity receber e processar os dados adequadamente.

2.3 Laboratório Virtual em Unity

A tela inicial do Unity apresenta ao usuário um espaço tridimensional vazio, uma fonte de iluminação e uma câmera, através da qual o experimento será observado. O ambiente permite a inserção de objetos de diferentes formas geométricas, a partir dos quais foi possível montar a sala do laboratório, a mesa e a planta.

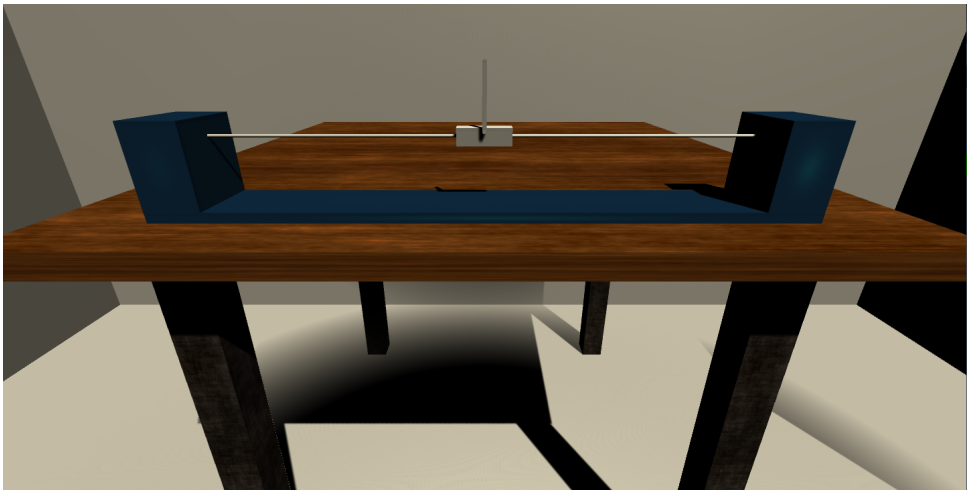


Figura 2.2: Representação virtual do laboratório e da planta de Pêndulo Invertido.

Após a criação do cenário e do equipamento, foram inseridas rotinas de programação em C++ para alterar os objetos de acordo com os dados recebidos do Matlab.

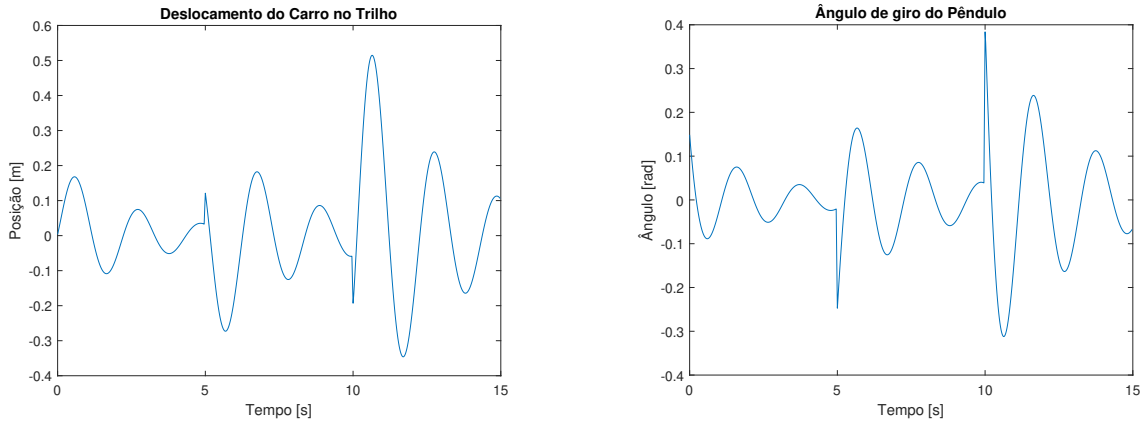
A rotina principal, denominada *cartControl* é responsável por criar a outra ponta do canal de comunicação. O Unity, que exerce a função do servidor, recebe a mensagem enviada, decodifica a sequencia de caracteres de texto novamente em dois valores numéricos, envia o valor de θ para a rotina responsável pelo controle do pêndulo, e altera a posição do objeto que representa o carro. Já a rotina secundária *pendControl* é mais simples, e exerce a função de alterar o ângulo do objeto que representa a haste em relação ao eixo vertical.

Apesar de fornecer uma vasta gama de efeitos físicos, como gravidade, detecção de colisões, arrasto, entre outras, os objetos criados no Unity somente reproduzem o comportamento dos estados que são enviados do Matlab e servem apenas de auxílio visual imersivo. Desta forma, evita-se a possibilidade do processamento de dados do Matlab conflitar com o da *game engine*.

3 Resultados e Discussão

Este capítulo conterá os resultados das simulações de Matlab. Quanto aos resultados no ambiente virtual criado, o leitor é convidado a assistir ao vídeo que contém os testes feitos[1].

Para a primeira simulação, serão usados os valores de ganho de realimentação referentes à resposta lenta, descritos na tabela 2.2. Para os testes com a planta controlada, foram inseridos um distúrbio impulsivo de amplitude $Fd = 200N$ no instante $t = 5s$, e outro distúrbio impulsivo de amplitude $Fd = -300N$ no instante $t = 10s$. As respostas obtidas estão ilustradas na figura 3.1.



(a) Posição r do Carro

(b) Ângulo θ da haste

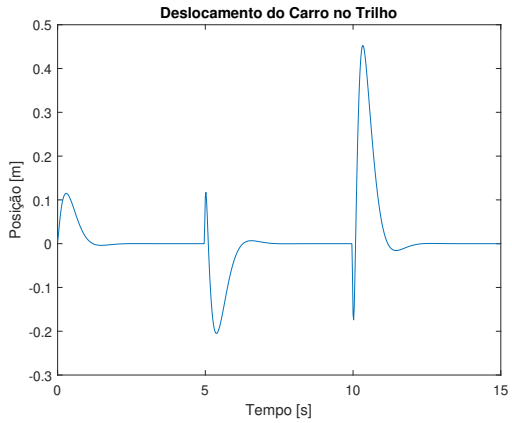
Figura 3.1: Comportamento da planta de Pêndulo Invertido controlada por realimentação de estados com ganhos baixos.

É possível perceber que o sistema caminha à estabilidade, mas não a atinge no tempo de simulação, o que comprova que os ganhos escolhidos produzem uma resposta lenta.

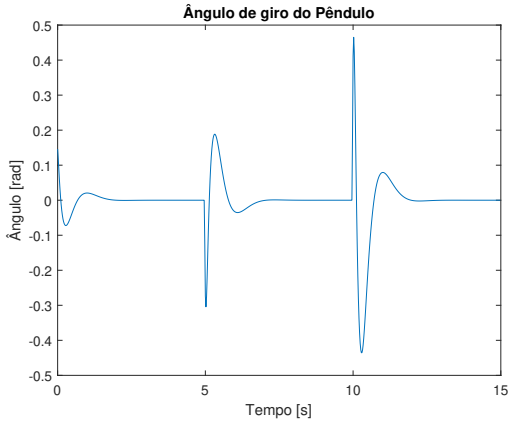
No segundo teste serão usados os valores de ganho de realimentação referentes à resposta rápida, descritos na tabela 2.2. De forma análoga ao anterior, os mesmos distúrbios serão aplicados ao sistema. As respostas obtidas estão ilustradas na figura 3.2.

Desta vez, devido aos altos ganhos do compensador, a estabilidade foi atingida rapidamente após todas as vezes em que o sistema foi perturbado. É importante ressaltar ao leitor que os ganhos sugeridos foram obtidos empiricamente e não necessariamente condizem com os valores máximo e mínimo que podem ser atribuídom à cada g. É visível que o teste 2 apresenta menor tempo de acomodação. Entretanto, os altos ganhos também causam maiores picos de variação no ângulo da haste. Logo, aumentar ainda mais os

[1]Disponível em: `https://youtu.be/ZO8njoCMWm4`



(a) Posição r do Carro

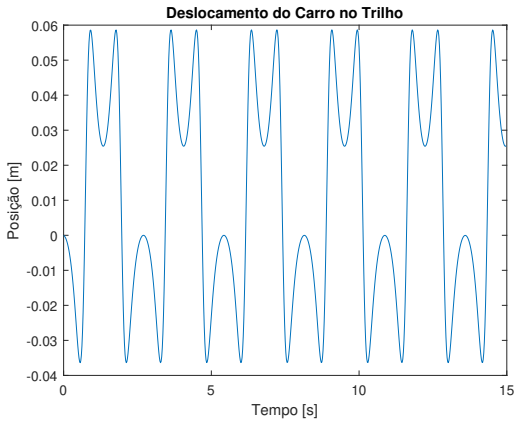


(b) Ângulo θ da haste

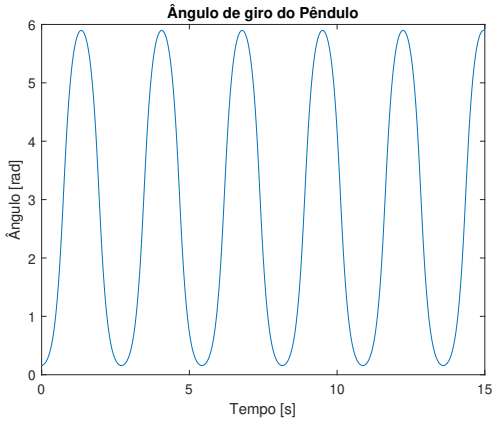
Figura 3.2: Comportamento da planta de Pêndulo Invertido controlada por realimentação de estados com ganhos altos.

ganhos para tornar o sistema mais rápido pode também fazer com que a inclinação atingida seja muito acentuada e a força Fc não seja suficiente para recuperar o movimento de queda do pêndulo.

Para o terceiro teste, a força de comando e os distúrbios foram fixados em zero para avaliar o comportamento do sistema sem atuação externa. Neste teste também, desprezou-se o efeito da força de atrito.



(a) Posição r do Carro

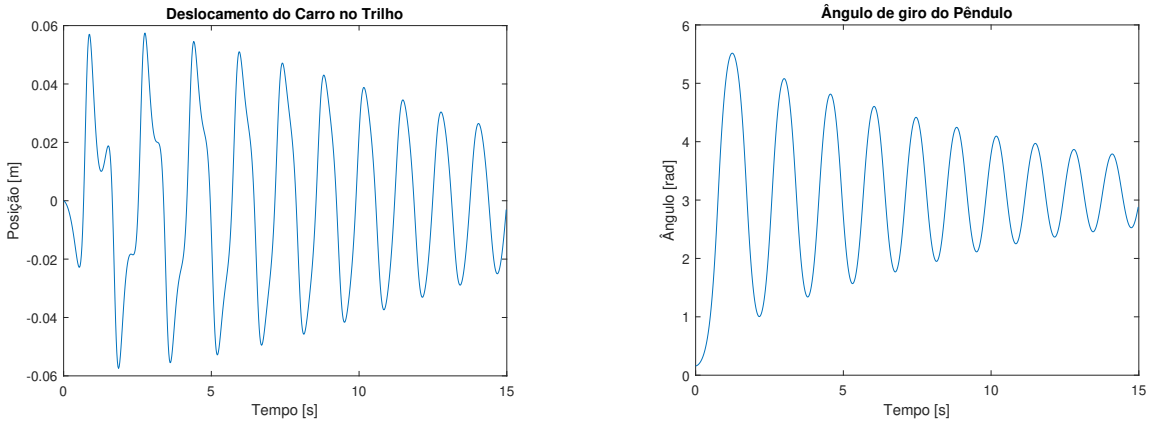


(b) Ângulo θ da haste

Figura 3.3: Comportamento da planta de Pêndulo Invertido sem atuação de forças externas e sem atrito.

Observa-se, pelos gráficos e pelo vídeo, que o ângulo da haste alterna senoidalmente em torno do ponto de equilíbrio $\theta = \pi$. Sem ações externas, os esforços internos, originários do deslocamento angular do centro de massa do pêndulo, provocam o deslocamento do carro sobre o trilho.

O último teste contemplado neste relatório ilustra o comportamento da planta sem a influência de esforços externos, porém sob efeito da força de atrito.



(a) Posição r do Carro

(b) Ângulo θ da haste

Figura 3.4: Comportamento da planta de Pêndulo Invertido sem atuação de forças externas e com atrito.

Conforme esperado, a força de atrito age como um dissipador de energia cinética. A queda inicial da haste novamente causa deslocamento do carro. Entretanto, o atrito do carro sobre o trilho faz com que a amplitude desse movimento diminua ao longo do tempo de simulação. Consequentemente, os esforços internos gerados reduzem também o ângulo de giro do pêndulo.

3.1 Discussões Gerais

A maior dificuldade encontrada no desenvolvimento deste trabalho foi fazer com que o ambiente virtual se comportasse como o modelo descrito em Matlab. Não só a definição do protocolo de comunicação, mas também em garantir que todos os objetos fossem apenas representações visuais da simulação gerada por um *software* externo ao Unity. A *game engine* dá ao usuário um ferramental suficiente para a produção de diferentes experimentos, jogos e simulações. Inclusive, seria possível simular o modelo matemático puramente em rotinas C++ internas ao Unity. Porém, as funções e ferramentas de análise fornecidas pelo Matlab, além de serem de maior familiaridade ao aluno de engenharia, são de grande relevância ao pesquisador que deseja entender os pormenores do comportamento do sistema simulado. A possibilidade de, por meio de comandos simples, gerar gráficos de desempenho do sistema ao fim da simulação, a fim de estudá-los ou disponibilizá-los em artigos ou relatórios, é um dos fatores que tanto atrai a atenção da comunidade acadêmica para esta plataforma. Logo, optou-se pela integração destes dois programas, a fim de proporcionar ao usuário tanto a imersão no ambiente virtual, quanto o acesso aos recursos computacionais de simulação e análise.

Na elaboração da comunicação TCP/IP entre as plataformas, a complexidade encontrada foi em como acondicionar dois dados a serem transmitidos em uma só mensagem, para então ser decodificada novamente no servidor em dois valores numéricos. Para este trabalho, os valores de r e θ foram envelopados em uma *string* de caracteres, e separados por um caractér de vírgula. No lado do receptor, esta *string* então foi subdividida em duas, com o auxílio do caractér separador incluído no passo anterior. Cada uma das sequências foi então convertida para um valor numérico para poder ser utilizada como parâmetro nos comandos de transformação de posição e rotação dos objetos virtuais. Como a conversão de texto para valores numéricos incorre na perda do separador decimal, foi necessário realizar uma divisão do valor convertido por uma potência de base dez.

Como citado anteriormente, um dos focos do trabalho foi poder proporcionar a possibilidade de obter os gráficos do comportamento do sistema ao fim da simulação, o que torna necessário que a simulação ocorra em tempo finito. A fim de aumentar a interação do usuário com o ambiente virtual, uma das ideias seria a de substituir as forças de distúrbio inseridas em instantes pré-definidos por comandos dados pelo usuário. Mas para isso, a simulação teria que estar interna à um *loop* de tempo infinito, sempre aguardando um comando para poder reagir à ele. Dentre as duas opções, foi dada prioridade ao acesso aos gráficos de desempenho por acreditar serem de maior valia aos discentes do que a interatividade.

4 Conclusão

O objetivo do trabalho foi estudar a integração do Matlab com o Unity para representar visualmente um experimento laboratorial básico de Sistemas de Controle. O algoritmo de Matlab foi capaz de gerar uma reprodução digital do funcionamento do ensaio, dados os modelos do sistema. A fim de tornar a experiência mais imersiva, foi gerada uma representação virtual e interativa deste ensaio em Unity, que se comunica com o Matlab por meio de um *socket* TCP-IP. Espera-se que tais representações se tornem parte do ferramental didático no ensido de Sistemas de Controle e que tragam aos discentes maior entendimento da disciplina e suas aplicações, bem como dos resultados dos experimentos, indo além da análise matemática e de gráficos de desempenho.

Como trabalhos futuros, espera-se criar uma nova versão do programa que contenha a interatividade citada na seção 3.1. Assim, o aluno poderá optar entre atuar diretamente sobre o sistema, ou obter os gráficos de desempenho.

Salienta-se que este estudo abre portas para que uma gama de experimentos laboratoriais sejam representados virtualmente, como controle de nível em tanques, levitadores magnéticos, entre outros. Assim, as aulas práticas de Sistemas de Controle podem abrangir um maior acervo de conceitos teóricos e viabilizar o acesso virtual à equipamentos que possam não estar disponíveis fisicamente.

Referências Bibliográficas

[1] CAMPBELL, S. A.; CRAWFORD, S.; MORRIS, K. Friction and the inverted pendulum stabilization problem. *Journal of Dynamic Systems, Measurement, and Control*, v. 130, n. 5, 2008.

[2] PRESS, W. H. et al. *Numerical Recipes in C: The Art of Scientific Computing*. USA: Cambridge University Press, 1988. ISBN 052135465X.