Universidade Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Kleber Marcio de Souza

Identificação e Controle Unidirecional em Robôs Móveis Utilizando Kit Robótico Lego Mindstorms

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina para obtenção do Título de Mestre em Engenharia Elétrica.

Londrina, PR
2014

Kleber Marcio de Souza

Identificação e Controle Unidirecional em Robôs Móveis Utilizando Kit Robótico Lego Mindstorms

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina para obtenção do Título de Mestre em Engenharia Elétrica.

Área de concentração: Sistemas de Controle por Realimentação

Orientador:
Prof. Dr. Marcio Roberto Covacic

Londrina, PR
2014

Ficha Catalográfica

de Souza, Kleber Marcio
Identificação e Controle Unidirecional em Robôs Móveis Utilizando Kit Robótico Lego Mindstorms. Londrina, PR, 2014. 80 p.

Dissertação (Mestrado) – Universidade Estadual de Londrina, PR. Centro de Tecnologia e Urbanismo. Programa de Pós-Graduação em Engenharia Elétrica .

1. Identificação de Sistemas. 2. Sistemas de Controle 3. Kit Robótico Lego 4. Estabilidade I. Universidade Estadual de Londrina. Centro de Tecnologia e Urbanismo. Programa de Pós-Graduação em Engenharia Elétrica . II. Identificação e Controle em Robôs Móveis Utilizando Kit robótico Lego Mindstorms.

Kleber Marcio de Souza

Identificação e Controle Unidirecional em Robôs Móveis Utilizando Kit Robótico Lego Mindstorms

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina para obtenção do Título de Mestre em Engenharia Elétrica.

Área de concentração: Sistemas de Controle por Realimentação

Comissão Examinadora

Prof. Dr. Márcio Roberto Covacic
Depto. de Engenharia Elétrica - UEL
Orientador

Prof. Dr. Ruberlei Gaino
Depto. de Engenharia Elétrica - UEL
Co-orientador

Prof.Dr.Paulo Laerte Natti
Depto. Matemática - CCE - UEL
Universidade Estadual de Londrina

17 de maio de 2014

Comece fazendo o que é necessário, depois o que é possível, e de repente você estará fazendo o impossível.

São Francisco de Assis

Agradecimentos

Agradeço à existência de uma instituição pública de ensino superior, a Universidade Estadual de Londrina, que mantém o ensino de excelência e gratuito, sem o qual muitas pessoas não teriam condições de concluir um curso de mestrado ou quiçá um curso de graduação.

Gostaria de expressar a minha gratidão ao meu orientador professor Márcio Roberto Covacic, pela sua extrema paciência comigo, pela sua amizade, ensinamentos e pelas suas correções pontuais no desenvolvimento deste trabalho.

Ao meu colega de laboratório Edno Gentilho Júnior, pelo inestimável auxílio na elaboração desse trabalho com suas ótimas opiniões.

Ao meu co-orientador, professor Ruberlei Gaino, pela sua colaboração técnica e auxílio no trabalho, com suas preciosas contribuições ajudando esse trabalho a se tornar realidade.

À minha esposa Edicléia e aos meus filhos Ives e Nathália, pelo amor, confiança, dedicação e compreensão pelos dias ausentes. Filhos ! Agora com o término deste trabalho poderemos nos divertir, dando aqueles deliciosos passeios de bicicleta.

Resumo

Atualmente robôs móveis estão sendo usados em diversas aplicações como: vigilância e tarefas domésticas. Para execução das aplicações, o controle da locomoção torna-se essencial para evitar obstáculos e colisões. Geralmente os processos de controle são descritos por modelos matemáticos. O objetivo deste trabalho é identificar um modelo matemático do robô *Lego Mindstorms NXT* e desenvolver uma estratégia para o controle de distância unidirecional para este robô. Para isso, é necessário um modelo que relaciona a potência do motor com a distância de um objeto em movimento. Com base nessa premissa, uma identificação do sistema foi desenvolvida para a obtenção de um modelo matemático que represente o robô neste ponto de operação. Para identificação, a caixa de Ferramentas Identificação de Sistemas (*System Identification Toolbox*), que integra o software Matlab foi utilizada. Essa ferramenta tem como utilidade construir modelos matemáticos de sistemas dinâmicos. O modelo matemático desenvolvido, é usado para o desenvolvimento de um controlador PID, que usa como critério de estabilidade o método de Routh-Hurwitz. Para testes reais desse controle, foi desenvolvido um algoritmo de controle que é utilizado juntamente com o Kit Robótico *Lego Mindstorms NXT*, que contém um controlador lógico programável conectado a vários componentes: sensor ultra-sônico, sensor de som, sensor de toque e servo-motores.

Abstract

Nowadays, movable robots are being used in many applications such as: surveillance and home chores. For the execution of such applications, the locomotion control becomes essential to avoid obstacles and collisions. Usually, the processes of controlling are described by mathematical models. The objective of this paper is to identify a mathematical model of the Lego Mindstorms NXT robot and develop a strategy for the unidirectional distance control of this robot. For such thing, it is necessary a model which relates the engine power with the distance of an object in motion. Based on this premise, an identification of the system has been developed to obtain a mathematical model which represents the robot in this point of operation. For the identification, the System Identification Toolbox, which integrates the Matlab software, has been used. This tool has the use of building mathematical models of dynamic systems. The developed mathematical model is used on the development of a PID controller, which uses the Routh-Hurwitz method as a stability criterion. For real tests of this control, a control algorithm has been developed to be used together with the robotic kit Lego Mindstorms NXT, which contains a programmable logic controller connected to several components: ultrasonic sensor, sound sensor, touch sensor and servomotors.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ARARX *AutoRegressive AutoRegressive Exogeneous* - autoregressivo autoregressivo com entrada exógena

ARARMAX *AutoRegressive AutoRegressive Moving Average with Exogenous inputs* - autoregressivo autoregressivo com média móvel e com entrada exógena

ARMAX *AutoRegressive Moving Average with Exogenous inputs* - auto regressivo com média móvel e entrada exógena

ARMA *AutoRegressive Moving Average* - auto regressivo com média móvel

ARX *AutoRegressive Exogeneous* - auto regressivo com entrada exógena

BJ *Box Jenkins*

FIR *Finite Impulse Response* - Resposta Finita ao Impulso

FT *Transfer Function* - Função de transferência

GUI *Graphical User Interface* - Interface Gráfica do Usuário

IDE *Integrated Development Environment* - Ambiente de desenvolvimento integrado

MIMO *Multiple-Input Multi-Output* - Múltiplas entradas e Múltiplas saídas

MISO *Multiple-Input Single-Output* - Múltiplas entradas e uma saída

NBC *Next Byte Codes*

NQC *Not Quite C*

NXC *Not eXactly* C - Não exatamente C

OE *Output Error*

P Proporcional

PD Proporcional Derivativo

PI Proporcional Integral

PID Proporcional Integral Derivativo

PWM *Pulse Width Modulation* - Modulação de Largura de Pulso

RAD *Rapid Application Development* - Desenvolvimento Rápido de Aplicações

RAM *Random Acess Memory* - Memória de acesso randômico

RCX *Robotic Control Explorer*

SIMO *Single-Input Multi-Output* - Uma entrada e mútiplas saídas

SISO *Single Input and Single output* - Uma entrada e Uma saída

SPD Semi-plano Direito

SPE Semi-plano Esquerdo

1 Introdução

Diversas estratégias de controle têm sido propostas e utilizadas com o intuito de melhorar o desempenho dos sistemas industriais; entre essas estratégias encontram-se as que utilizam modelos matemáticos do processo a ser controlado (AGUIRRE, 2007b). A identificação de sistemas é uma área do conhecimento que elabora modelos matemáticos de sistemas dinâmicos baseados em observações das suas entradas e saídas (SODERSTROM; STOICA, 1989). Para o entendimento deste trabalho, cabe ressaltar a definição de sistema segundo (AGUIRRE, 2007b): sistema é um objeto cujo comportamento é determinado por variáveis internas que interagem e produzem sinais observáveis denominados saídas. São exemplos de sistemas: aviões, robôs, automóveis e o corpo humano. Engenheiros e pesquisadores usam os modelos matemáticos para projetar sistemas dinâmicos com o intuito de simular e controlar fenômenos da vida real. Assim, torna-se necessário conhecer técnicas de modelagem matemáticas para resolver problemas em muitas áreas, como aviação e controle.

No caso do controle, especificamente, a identificação é importante para projetos e implementação de um sistema de controle (AGUIRRE, 2007b), onde uma boa identificação gera como produto um controle bem sucedido (LJUNG, 1987). Existem vários métodos para construção de modelos, que podem ser encontrados na literatura, como em (LJUNG, 1987) e (AGUIRRE, 2004), (SODERSTROM; STOICA, 1989). Basicamente, existem duas classificações para métodos de identificação: paramétrico e não paramétrico. Os paramétricos são usados para estimar vetores de parâmetros dados como θ. Dado um conjunto de entrada e saída do sistema, encontram-se os parâmetros do modelo que melhor representa o processo. Por outro lado, os não paramétricos não empregam vetores de parâmetros de forma direta, e são utilizados, por exemplo, na obtenção da resposta em frequência ou na obtenção de sua resposta a uma entrada degrau (AGUIRRE, 2007b). A identificação de parâmetros com base nos dados de entrada e saída é representada por estruturas estatísticas, chamadas de regressores, que podem ser utilizados para predizer o seu comportamento futuro.

Trabalhos que envolvem a identificação de sistemas, geralmente, são estudos em que há uma procura por um modelo matemático que represente um sistema de maneira integral ou parcial. (MALDONADO, 2005) utilizou métodos de identificação paramétricos, tais como ARX, ARMAX, de modo a obter modelos alternativos para uma micro-turbina, para que ela trabalhe com combustível de baixo poder calorífico. Procurando implementar computacionalmente um modelo linear que represente os transientes aos quais um duto está sujeito, (BRANDOLT, 2002) se utiliza de métodos de identificação para a modelagem do sistema. Em (GAINO, 2009), foram realizados estudos para a obtenção de um controle não linear da perna de um paraplégico em modelos Takagi-Sugeno(T-S). Para isso, foi proposto um método de identificação de sistemas para os modelos locais T-S.

Outro exemplo de aplicação de identificação linear e não linear foi proposto por (BIAZETO, 2011), onde foram desenvolvidos algoritmos de identificação para serem aplicados ao modelo não linear do movimento do complexo canela-tornozelo de um paciente paraplégico para que fosse estimada a função de transferência do mesmo. Por fim, (OLIVEIRA, 2013) realiza uma identificação dinâmica do modelo do movimento do paciente paraplégico, utilizando-se do método mínimos quadrados recursivos para desenvolvimento de um controlador do tipo paralelo distribuído com modelos fuzzy (T-S).

Nos últimos anos tem se observado uma intensa investigação mundial a respeito da robótica móvel, existindo cada vez mais aplicações para robôs móveis na prestação de serviços (HOWARD; IAGNEMMA; KELLY, 2010); (SIEGWART, 2006) e (WEINBERG; YU, 2003). Esses robôs são geralmente utilizados em tarefas em que envolve riscos para o homem tais como: inspeção em ambientes com gases tóxicos, desativação de minas e a procura por sobreviventes em zonas de desastres naturais. Da mesma forma, existem robôs que atuam em laboratórios farmacêuticos, em salas de cirurgias ou nas atividades diárias de pessoas com necessidades especiais. Para sua mobilidade, esses robôs são geralmente dotados de controles de locomoção para mover-se em determinados ambientes de maneira segura e sem colisão, apoiando-se nas informações dos sistemas de sensores externos.

Utilizando-se de um robô, o presente trabalho une duas teorias: identificação de sistemas e controle. Seu objetivo é identificar um modelo matemático do robô Lego *Mindstorms NXT* e desenvolver uma estratégia para o controle de distância, em relação a um objeto em movimento seguindo em linha reta, com base na informação proveniente do sensor de ultrassom. Para o estudo da estabilidade em questão utiliza-se da teoria Routh-Hurwitz. Especificamente é desenvolvido um algoritmo de controle PID, que será usado no robô *Lego Mindstorms NXT*. O

Kit robótico *Lego Mindstorms NXT*[1], contém um controlador lógico programável que é conectado a vários componentes: sensor ultrassônico, sensor de som, sensor de toque e servo-motores.

O kit robótico Lego Mindstorms é usado em diversos estudos. Em (MANTOVANI, 2011), o robô é utilizado para simular uma rede neural de *spikes*, na prevenção de colisões com obstáculos em um ambiente desconhecido. Já (ANDRADE; PEDRO FILHO; SILVA, 2013), de uma maneira pedagógica, utilizam o kit para desenvolver uma aplicação para utilização de sensores, no auxílio do ensino da programação. No trabalho de (GONÇALVES et al.,) tem-se uma dificuldade no desenvolvimento de software de controle, localização e navegação para robôs móveis; então, em seu trabalho, propõe-se o uso de simuladores realistas para desenvolvimento, validação e migração de *softwares* para robôs reais. Para isso, usa-se o kit-robótico Lego para uma prévia na modelagem dos simuladores. (KIM, 2011) mostra que o sistema Lego, juntamente com o software *Matlab\Simulink*, podem ser usados para o ensino das teorias de controle clássico e moderno nos cursos de graduação.

1.1 Motivação e Contribuição da Dissertação

A quantidade de pessoas deficientes e idosas no mundo estão em crescente expansão, aumentando assim a necessidade por interfaces homem-máquina com a finalidade de ajudar na mobilidade destas pessoas (FERREIRA; CERVANTES; GERMANOVIX,) . Preocupado com a qualidade de vida de tetraplégicos, o grupo de pesquisa de Controle Avançado, Robótica e Engenharia Biomédica da Universidade Estadual de Londrina desenvolve, desde do ano 2000, estudos no desenvolvimento em uma cadeira de rodas, a qual teve sua cinemática modelada por (MAZO et al., 1995) e adaptada em (FONSECA SOBRINHO et al., 2000) e (FONSECA SOBRINHO et al., 2003), para tetraplégicos, que é controlada por uma interface de sopro e sucção. Testes em pacientes utilizando essa cadeira foram bem sucedidos como mostrado em (LUCIANO, 2014). Por causa disso, outros trabalhos derivados surgiram no departamento como em (SANTOS et al., 2013) que pesquisa a variação do posicionamento do corpo do paciente sobre uma cadeira de rodas a partir de um modelo dinâmico não linear que inclua desvios lateral e longitudinal do centro de massa. Em (ROSSINI, 2013), é desenvolvido um controlador robusto aplicado à cadeira de rodas móveis, utilizando a abordagem por LMIs. Com base em dados coletados, em (ROSA et al., 2013) é proposto um sistema de controle para o

[1]Para maiores detalhes acesse:http://mindstorms.lego.com/en-us/default.aspx

protótipo da cadeira de rodas controlada por sopro e sucção, o qual, utiliza-se da técnica de alocação de polos. Em (GENTILHO JUNIOR, 2013) é implementado um processador digital de sinais (DSP) para o controle da cadeira de rodas utilizando de um controlador PID.

Para contribuir com os esforços do grupo de pesquisa, este trabalho tem como objetivo identificar e desenvolver um controle PID para o robô Lego Mindstorms, no intuito de servir como protótipo e estudo para viabilizar pequenos testes de controle, quando um sistema robótico auxiliado por sensores encontra um obstáculo móvel ou estático. Os conhecimentos adquiridos no projeto do robô podem ser aproveitados em trabalhos futuros para simular mecanismos de controle na cadeira de rodas, com a finalidade de obter segurança durante os deslocamentos da cadeira, evitando assim colisões com os obstáculos. Também, a contribuição desta dissertação destaca-se em um desenvolvimento iterativo de identificação de sistemas e controle, que é auxiliada por ferramentas computacionais no desenvolvimento de um modelo matemático.

1.2 Organização do Trabalho

Esse trabalho divide-se na seguinte forma: no capítulo 2, serão apresentados aspectos relevantes sobre a identificação de sistemas dinâmicos, também é discutido como são representados modelos lineares e, por fim, são explicados os métodos paramétricos. O capítulo 3 mostra o uso da Toolbox para identificação de sistemas e, no capítulo 4, sistemas de controle são discutidos. No capítulo 5, são discutidas especificações técnicas a respeito do Kit Robótico Lego Mindstorms e as ferramentas de programação e captação de dados utilizando este robô. No capítulo 6, o projeto de identificação e controle do robô é apresentado juntamente com os resultados obtidos. No capítulo 7, é apresentada a conclusão desse trabalho e algumas propostas para trabalhos futuros. Por fim, o apêndice A mostra a execução do algoritmo para encontrar faixa de estabilidade. O apêndice B é apresentado o algoritmo do controle da distância do robô Lego e o apêndice C mostra todos os modelos matemáticos estimados graficamente.

1.2.1 Disseminação

- SOUZA, Kleber Marcio; COVACIC, Márcio Roberto; GAINO, Ruberlei; GENTILHO JUNIOR, Edno; *Identificação e Controle em Robôs Móveis Utilizando Kit Robótico Lego Mindstorms.* In: Anais do XI Simpósio Bra-

sileiro de Automação Inteligente, 2013, Fortaleza. Disponível em: `http://www.sbai2013.ufc.br/pdfs/5655.pdf`. Acessado em 13 DEZ 2013.

- SOUZA, Kleber Marcio; COVACIC, Márcio Roberto; GAINO, Ruberlei; GENTILHO JUNIOR, Edno; *IDENTIFICAÇÃO E CONTROLE DISCRETO UTILIZADO EM ROBÔS MÓVEIS*. Submetido a Revista Semina: Ciências Exatas e Tecnológicas (2014) - UEL. Este trabalho discute o desenvolvimento de um modelo matemático, o qual, é usado para o desenvolvimento de um controlador proporcional discreto, que usa como critério de estabilidade o método de Routh-Hurwitz.

2 Aspectos da Identificação de Sistemas

Com uma complexidade cada vez maior nos processos, é fundamental ter conhecimento dos modelos matemáticos, para análise, compreensão e controle. A identificação de sistemas tem como objetivo criar técnicas alternativas de modelagem matemática que descrevam o comportamento de um sistema dinâmico a partir das medições das suas entradas e saídas.

Normalmente os sistemas são descritos por modelos matemáticos, que posteriormente se transformam em sistemas reais. O projeto e a modelagem matemática dos fenômenos que constituem um sistema são, às vezes, extremamente complexos, por isso, um modelo gerado nunca reproduz o comportamento do sistema original (LJUNG, 1987).

No aspecto de identificação de sistemas, o processo de modelagem consiste na criação de equações matemáticas que melhor descrevam o sistema. Assim, a modelagem tem uma especial importância na engenharia, em particular para projetos de sistemas de controle. Utilizando o modelo matemático obtido, pode-se fazer análise e predizer o comportamento de um sistema em determinadas condições e, posteriormente, ajustar seu desempenho, caso ele não seja satisfatório.

Basicamente, existem duas maneiras de construir modelos matemáticos (SODERSTROM; STOICA, 1989):

- Modelagem matemática: Quando as relações físicas e matemáticas do fenômeno envolvido são conhecidas e esse conhecimento é utilizado para construção do modelo. A Figura 2.1 mostra as equações que regem o circuito RLC . Os modelos matemáticos podem auxiliar no entendimento do comportamento do sistema, porém, a construção de modelos pode ser bastante complicada quando o sistema a ser modelado é muito grande e complexo (AGUIRRE, 2004).

- Identificação de sistemas: É uma técnica que permite construir modelos ma-

$$L\frac{di(t)}{dt} + Ri(t) + v_c(t) = v(t),$$

$$C\frac{dv_c(t)}{dt} = i(t).$$

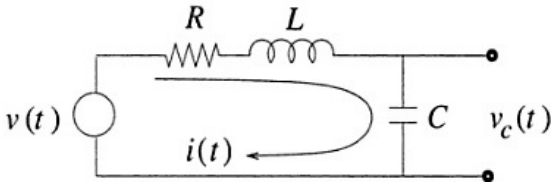


Figura 2.1: Circuito RLC

Fonte: (AGUIRRE, 2004)

temáticos de sistemas dinâmicos a partir de dados medidos. Nesse modo, não existe conhecimento matemático do fenômeno. O que existe de fato são dados observados dos fenômenos que servem para construir o modelo. Assim, modelos gerados a partir dessa técnica são utilizados para inferir propriedades dinâmicas e estatísticas do sistema original (RODRIGUES, 1996) e (LJUNG, 1987).

A metodologia para identificação é feita de duas maneiras, conforme (COELHO; COELHO, 2004):

1. Off-line: Nesse modo, os dados primeiramente são armazenados e posteriormente são transferidos para um computador, onde são tratados e processados.

2. On-line: Conhecida como identificação em tempo real, os dados são processados de forma recursiva à medida que um novo conjunto de dados é disponibilizado.

Então, de uma maneira geral, uma identificação pode ser realizada excitando-se um sistema com algum tipo de sinal de entrada e, após isso, observa-se a sua saída em um intervalo de tempo, armazenando essas informações. Mais especificamente, o processo de identificação baseia-se em dados amostrados para selecionar a estrutura de um modelo. Após essa etapa, e seguindo algum critério de qualidade, faz-se uma estimativa dos parâmetros envolvidos no processo e, por fim, valida-se o modelo. Assim, pode-se notar a existência de quatro etapas típicas para a construção de um modelo, observe a figura 2.2. Essas etapas são segundo (AGUIRRE, 2007b):

- *Experimento:* O experimento ou coleta de informações tem como objetivo capturar informações relevantes ao processo. Essa fase é importante no

processo de identificação, visto que, sem dados, não se pode obter um modelo. Assim, os experimentos deverão ser realizados em condições as mais semelhantes possíveis àquelas em que o modelo será empregado.

Para (AGUIRRE, 2004), quando não existe a possibilidade de uma experiência específica para a identificação, os dados devem ser adquiridos da operação normal do sistema, em outras, entretanto, testes devem ser feitos para extrair informações relevantes da dinâmica do processo.

- *Modelagem:* Com os dados amostrados, nesta etapa é definido se é utilizada uma representação linear ou não linear e se o modelo será do tipo paramétrico ou não paramétrico. No caso, se o objetivo for obter um modelo linear, o problema se resume na escolha do número de polos, zeros e o atraso de tempo.

  Ainda, é preciso decidir sobre a representação matemática que será usada, por exemplo, se for modelos lineares paramétricos, eles podem ser representados por funções de transferência ou modelos em espaços de estados.

- *Estimativa:* Essa etapa se resume em escolher o algoritmo de estimação, em que, na maioria dos casos, é adotado o método clássico de mínimos quadrados.

- *Validação do modelo:* Na validação, um modelo matemático é testado e verificado se é uma representação adequada do sistema real. Se não o for, uma estrutura mais complexa deve ser considerada, seus parâmetros estimados e um novo modelo validado (SODERSTROM; STOICA, 1989).

Existem algumas formas de classificar o sistema a ser identificado, conforme o conhecimento que se tem a respeito deste sistema. Pode-se determinar a classe de modelo a ser utilizada de acordo com o nível de conhecimento disponível sobre o sistema (SJÖBERG et al., 1995). Dessa forma, podemos classificar as classes de modelos em:

- Modelos "Caixa Branca": Esse modelo indica um conhecimento total sobre o processo, como as leis da física que o regem, ou seja, conhece-se a relação entre as variáveis envolvidas no comportamento dinâmico do sistema. Para (AGUIRRE, 2004), essa modelagem também é conhecida como *modelagem pela física* ou *natureza do processo*. Esse sistema nem sempre é viável, levando-se em conta o tempo e o conhecimento necessário para a modelagem desses sistemas.

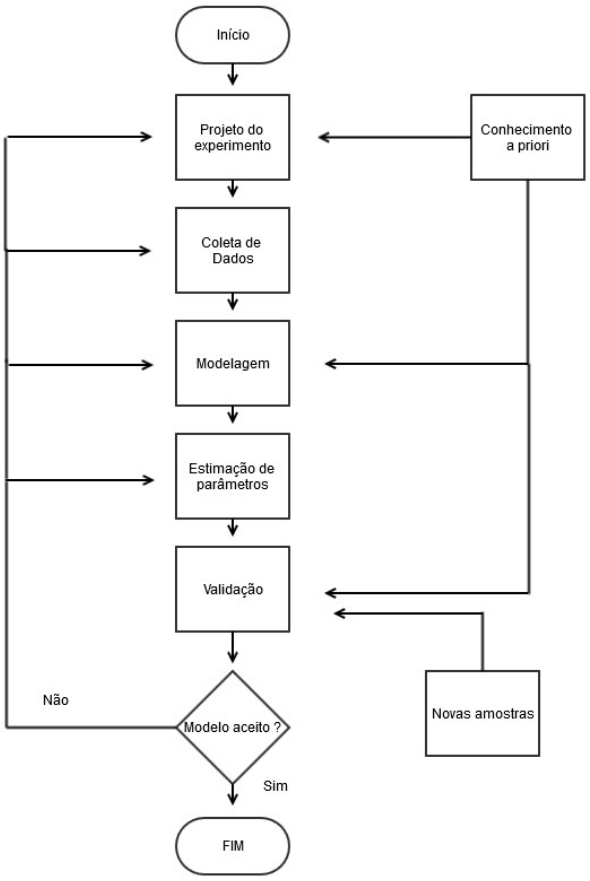


Figura 2.2: Fluxograma do processo de identificação.

Fonte: Adaptado de (SODERSTROM; STOICA, 1989)

- Modelos "Caixa Cinza": Esse modelo pode ser colocado entre a identificação caixa branca e a identificação caixa preta. As técnicas aqui utilizadas indicam que se tem algum conhecimento sobre o processo, porém não são conhecidos alguns parâmetros ou relações das variáveis envolvidas. Nesse caso, utilizam-se também as informações de entrada e saída.

- Modelos "Caixa Preta": Conhecido como modelagem empírica, esse modelo necessita de pouco ou nenhum conhecimento prévio do sistema. Esse modelo se baseia apenas nos dados de entrada e saída e é objeto de estudo neste trabalho.

2.1 Conceitos de Modelos

O conhecimento científico teve sua evolução com base na construção e análise de modelos para os sistemas em estudo. Em sua grande maioria, os modelos procuravam reproduzir alguns padrões encontrados na natureza. Então, historicamente, a modelagem de sistemas permitiu estudar sistemas minúsculos e complexos como

os átomos, e sistemas imensos, como o sistema solar (RODRIGUES, 1996). Em (AGUIRRE, 2004), o funcionamento do ser humano é baseado em modelos e a maioria desses modelos é mental.

Modelos, então, podem ser regras ou um conjuntos delas, que descrevem o comportamento do sistema fornecendo informações temporais das variáveis envolvidas (AGUIRRE, 2007b). Os modelos podem ser: mentais, gráficos, matemáticos.

Normalmente utilizamos os modelos mentais no dia a dia; esses modelos indicam uma sequência de ações para atingir um objetivo. A habilidade de dirigir um carro para deslocamento é um exemplo de modelo mental. O modelo gráfico, é descrito por tabelas que relacionam as diferentes variáveis do sistema. Como exemplo de modelo gráfico tem-se a curva característica tensão-corrente de um dispositivo eletrônico (RODRIGUES, 1996).

Por fim, modelos matemáticos são estruturas mais utilizadas para representar o comportamento dos sistemas. Em diversas aplicações, além de utilizar o modelo mental, é necessário o uso de modelos que possam ser representados por relações matemáticas. Os modelos matemáticos servem para descrever os sistemas de uma forma quantitativa. Dessa forma, modelos matemáticos são constituídos de equações diferenciais (tempo contínuo) ou equações de diferenças (tempo discreto). Com o objetivo de obter um controle para o robô Lego, o modelo matemático é o que mais interessa a este trabalho. Os modelos matemáticos mais comuns, segundo (AGUIRRE, 2004).

Modelos estáticos e dinâmicos: Um modelo é estático quando a saída, num determinado instante, depende apenas da entrada no mesmo instante, sendo assim, podemos falar que não existem realimentações. O modelo dinâmico é considerado assim, quando num instante de tempo as saídas dependem daquela entrada no mesmo instante e também de dados colhidos em instantes de tempos anteriores (das entradas e saídas). Modelos estáticos são descritos por equações algébricas, já os modelos dinâmicos são descritos por equações diferenciais (ou a diferenças para o caso discreto).

Modelos discretos e contínuos: Modelos contínuos são os valores de saída conhecidos em todos os instantes num intervalo de tempo, ou seja, representam a evolução do sistema continuamente no tempo. Esses tipos de modelo são descritos por equações diferenciais. Por outro lado, têm-se modelos discretos quando se conhece valores de saídas apenas em instantes de tempo, e são descritos por equações a diferenças.

Modelos monovariáveis e multivariáveis: Basicamente, modelos multivariáveis são os que têm múltiplas entradas ou múltiplas saídas (MIMO, MISO e SIMO). Representando a relação causa e efeito, os modelos monovariáveis têm apenas um par de variáveis, ou seja, de uma entrada para uma saída. Na literatura, são conhecidos como SISO.

Modelos determinísticos e estocásticos: Modelos determinísticos não consideram o ruído ou parâmetros incertos presentes nos dados, mesmo que se aceite a presença destes. Por outro lado, métodos estocásticos, além de levar em conta a presença de ruídos e incertezas em seus dados, também utilizam recursos para tratá-los.

2.2 Representação de Modelos Lineares

A palavra sistema pode ser considerada muito abrangente, podendo ser usada em uma grande variedade de contextos e, muitas vezes, pode até se referir a sistemas dinâmicos abstratos. No âmbito da identificação de processos, conforme (AGUIRRE, 2007b), sistema é um componente que tem seu comportamento regulado por variáveis internas que interagem e produzem sinais observáveis que são denominados saídas.

Um sistema é dito linear quando segue o princípio da superposição. Considere um sistema que, quando excitado pela entrada $u_1(t)$, produz como saída $y_1(t)$ e quando excitado pela uma entrada $u_2(t)$, produz também a saída $y_2(t)$. Se esse sistema seguir o princípio da superposição, então, quando excitado por $a.u_1(t) + b.u_2(t)$, sua saída será $a.y_1(t) + b.y_2(t)$, sendo a e b constantes reais. Assim, por definição, um sistema é linear se satisfizer o princípio da superposição (DORF; BISHOP, 2001).

Existem várias formas de representar as equações que descrevem um sistema. Entre as mais importantes representações matemáticas de modelos lineares estão: Função de transferência (FT) e a espaço de estados. Será dado um enfoque especial para as funções de transferência, pela importante relevância para neste trabalho.

2.2.1 Funções de Transferência

Funções de transferência são modelos matemáticos que modelam o comportamento dinâmico de um par de entrada e saída de um sistema, ou seja, descrevem

como a entrada é dinamicamente transferida para a saída do sistema. Essa função é definida como a relação entre a transformada de Laplace da saída e a transformada de Laplace da entrada, ou seja, dois polinômios em s com todas as condições iniciais nulas (AGUIRRE, 2004).

A transformada de Laplace é um método utilizado para solucionar equações diferenciais lineares. Assim, pode-se converter funções como seno, cosseno e exponenciais em funções algébricas no domínio de uma variável complexa s. Uma das características desse método é utilizar técnicas gráficas para a previsão de desempenho de um sistema, sem a necessidade de solucionar sistemas de equações diferenciais (OGATA, 2003). Por isso, (DORF; BISHOP, 2001), entendem que FT são importantes porque dotam o analista e o projetista de um modelo matemático útil dos elementos do sistema.

A obtenção da função de transferência pode ser considerada um problema típico em modelagem de sistemas lineares. Dessa forma, utiliza-se de duas formas para a obtenção dessa estrutura linear (AGUIRRE, 2004):

1. Aplicando diretamente a transformada de Laplace à equação diferencial que representa o sistema. Por sua vez, a equação diferencial é obtida conhecendo as leis que descrevem os fenômenos envolvidos.

2. Pode-se também obter a função de transferência a partir de dados gerados pelo sistema utilizando métodos de identificação de sistemas discutido no capítulo 2.

Considere o sistema definido pela seguinte equação diferencial:

$$a_n\frac{d^n y}{dt^n}+a_{n-1}\frac{d^{n-1}y}{dt^{n-1}}+...+a_1\frac{dy}{dt}+a_0 y = b_m\frac{d^m x}{dt^m}+b_{m-1}\frac{d^{m-1}x}{dt^{m-1}}+...+b_1\frac{dx}{dt}+b_0 x, \quad (2.1)$$

sendo y á saída do sistema, x a entrada e $n \geq m$. A obtenção da função de transferência desse sistema é obtida tomando-se a transformada de Laplace de ambos os membros da equação.

$$G(s) = \frac{\mathcal{L}\left[saída\right]}{\mathcal{L}\left[entrada\right]}|\text{condições iniciais nulas, ou seja,}$$

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + ... + b_{m-1}s + b_m}{a_0 s^n + a_1 s^{n-1} + ... + a_{n-1}s + a_n}. \quad (2.2)$$

A ordem do sistema será determinada, conforme potência de s no denominador da função de transferência. Se a máxima potência no denominador for igual

a n, o sistema será de ordem n. Com o conceito de função de transferência, a dinâmica do processo do sistema é representada pelas equações algébricas em s (OGATA, 2003).

Embora a FT seja limitada a sistemas de equações diferenciais lineares invariantes no tempo, ela fornece informações mais intuitivas do que as próprias equações diferenciais. Então, usando as funções de transferência podemos promover mudanças nos parâmetros do sistema e, de uma maneira rápida, notam-se essas mudanças na resposta do sistema (NISE, 2010). Ainda com base em (OGATA, 2003), mais alguns pontos relevantes a respeito da FT podem ser destacados:

1. Funções de transferência não fornecem aspectos a respeito da estrutura física do sistema.
2. Independentemente da magnitude e natureza da entrada, sua propriedade é inerente ao sistema.
3. Sistemas físicos diferentes podem ter funções de transferência idênticas.
4. Se em um determinado sistema a sua função de transferência for conhecida, a sua resposta pode ser estudada para várias maneiras de entrada. De uma maneira contrária, se não se conhece a função de transferência, ela pode ser determinada estabelecendo experimentalmente entradas conhecidas para um estudo de suas saídas.

Existem diversas representações matemáticas para modelos lineares. A função de transferência é a representação matemática mais utilizada. Porém, uma outra maneira útil de representar um modelo dinâmico linear é utilizar a representação espaço de estados. Para mais conceitos dessas representações sugere-se a leitura de (LJUNG, 1994)(OGATA, 2003) e (DORF; BISHOP, 2001).

2.2.2 Polos e Zeros do Sistema

Para projetos de sistemas de controle, a análise dos polos e zeros da FT torna-se fundamental, pois simplifica a análise do comportamento qualitativo de funções de transferência (AGUIRRE, 2004). Considere a seguinte função de transferência:

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_0 + b_1 s + ... + b_q s^q}{a_0 + a_1 s + ... + a_n s^n}. \tag{2.3}$$

Os zeros de $H(s)$ são os valores da variável de Laplace (s) que tornam a FT nula, ou seja, tornam o numerador igual a zero. Já os polos de $H(s)$ são os valores

da variável de Laplace que tornam a FT infinita, isto é, tornam o denominador da FT igual a zero. Outra definição importante é a de polinômio característico. Este polinômio identifica o comportamento do sistema no domínio frequência, recebendo o nome de polinômio característico, que na equação (2.3) é dado por $D(s)$.

Por fim, os polos e zeros de uma função de transferência podem ser reais ou complexos (AGUIRRE, 2004). Nas funções de transferência reais, quando os polos e zeros se apresentarem complexos, eles aparecerão em pares conjugados da seguinte forma: $\alpha + j\beta$, se for um polo ou zero. Seguindo esse mesmo raciocínio $\alpha - j\beta$ também será um polo ou zero do sistema. Então um par complexo pode ser representado por $\alpha \pm j\beta$.

2.3 Modelos Paramétricos

Os métodos desenvolvidos para a identificação de sistemas possuem duas classificações importantes: métodos paramétricos e não paramétricos. São denominados paramétricos aqueles que utilizam estruturas matemáticas parametrizadas para descrever o comportamento dinâmico original. Os parâmetros são caracterizados por números ou coeficientes e são ajustados por algoritmos de estimação a partir de seus dados medidos. Em contrapartida, métodos não-paramétricos são caracterizados por gráficos tais como: resposta ao impulso e a resposta em frequência (AGUIRRE, 2007b). Nesse tópico, trataremos em especial do método paramétrico, haja vista que é um tópico de relevância para esse trabalho.

Os métodos paramétricos são utilizados quando se tem um conhecimento *a priori* sobre o sistema, tais como: leis físicas que o descrevam ou outras relações que permitam uma escolha de famílias de modelos mais adequada para o sistema em estudo (AGUIRRE, 2007b). Os modelos paramétricos são usados para estimar vetores de parâmetros aqui designado como θ, para um certo modelo. Como exemplo observe a seguinte equação:

$$y(t) + ay(t-1) = bu(t-1) + e(t). \tag{2.4}$$

Analisando a equação (2.4), nota-se uma equação típica de um modelo discreto de primeira ordem, sendo assim, seu vetor pode ser definido como (SODERSTROM; STOICA, 1989):

$$\theta = \frac{a}{b} \tag{2.5}$$

Para (COELHO; COELHO, 2004), a identificação paramétrica é a observação de

variáveis do sistema com algum critério predefinido, gerando assim modelos matemáticos. Essa análise se baseia num modelo de regressão linear.

Na literatura sobre identificação de sistemas existem diversas estruturas paramétricas para identificação (SODERSTROM; STOICA, 1989). Na maioria das vezes são usadas em modelos discretos (AGUIRRE, 2004). Dentre os diferentes modelos utilizados temos: Box-Jenkins (BJ), output error (OE), auto regressivo com entrada externa (ARX), auto regressivo com média móvel e entradas exógenas (ARMAX). Na seção seguinte, será apresentada a estrutura geral desses sistemas e detalharemos o modelo ARX por ser uma estrutura mais simples e mais utilizada na literatura sobre identificação e também importante para o desenvolvimento deste trabalho.

2.4 Estruturas de Modelos Paramétricos

Na estatística, existem ferramentas que auxiliam na previsão de uma determinada variável a partir da observação do conjunto de outras variáveis. A auto-regressão é uma representação matemática do comportamento de um processo através de um modelo que pode ser utilizado para predizer o seu comportamento futuro da saída $y(k)$ em relação à entrada $u(k)$. Neste trabalho, são utilizadas essas representações para a modelagem matemática. Os modelos são aplicados sobre determinada sequência de observações de uma ou mais variáveis em estudo. Dessa forma, o conjunto de variáveis observadas pode ser representado por um sinal $y(t) = [y_t, y_{t-1}, \ldots, y_{t-k}]$, onde t é o instante referente a um dado valor de y e k é o valor do atraso considerado para o sinal (CARVALHO, 2008). O modelo geral segundo (AGUIRRE, 2004) é dado por:

$$A(q)y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}e(k), \tag{2.6}$$

$$y(k) = \frac{B(q)}{F(q)A(q)}u(k) + \frac{C(q)}{D(q)A(q)}e(k),$$

$$y(k) = H(q)u(k) + G(q)e(k),$$

sendo que q^{-1} representa o operador de atraso, isto é, $y(k)q^{-1} = y(k-1)$, e $e(k)$ um ruído branco com A, B, C, D, F polinômios definidos a seguir:

$$A(q) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a} \tag{2.7}$$

$$B(q) = b_1 q^{-1} + ... + b_{n_b} q^{-n_b} \tag{2.8}$$

$$C(q) = 1 + c_1 q^{-1} + ... + c_{n_c} q^{-n_c} \tag{2.9}$$

$$D(q) = 1 + d_1 q^{-1} + ... + d_{n_d} q^{-n_d} \tag{2.10}$$

$$F(q) = 1 + f_1 q^{-1} + ... + f_{n_f} q^{-n_f}. \tag{2.11}$$

As funções $H(q)$ e $G(q)$ são referidas como funções de transferência do processo e ruído, respectivamente.

Conforme os polinômios utilizados na equação (2.6) , a estrutura do modelo recebe um nome específico, conforme a tabela 2.1 (FERNANDEZ, 2009):

Polinômios utilizados	**Nome Estrutura do Modelo**
B	FIR
A,B	ARX
A,B,C	ARMAX
A,C	ARMA
A,B,D	ARARX
A,B,C,D	ARARMAX
B,F	OE(output error)
B,C,D,F	BJ(Box Jenkins)

Tabela 2.1: Nomenclatura das estruturas de modelos polinomiais.

Fonte: (FERNANDEZ, 2009)

2.4.1 Modelo ARX

O modelo AR (auto-regressivo) é considerado o mais comum regressor paramétrico, e é dado pela fórmula (AGUIRRE, 2004):

$$A(q^{-1})y(t) = e(t). \tag{2.12}$$

Como dito, o operador q^{-n_a} indica o valor da função $y(t)$ em um instante anterior $y(t - n_a)$. Então, ao selecionar a ordem n_a do operador determina-se o número de valores atrasados de $y(t)$, que serão utilizados para determinar o valor no instante atual (CARVALHO, 2008). O erro cometido ao se tentar modelar $y(t)$ em função dos seus valores atrasados é $e(t)$. Expandido a equação (2.12), tem-se

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t - n_a) = e(t). \tag{2.13}$$

Ao acrescentar ao modelo AR (Auto-regressivo) uma entrada externa $u(t)$, que também é uma série temporal de uma variável que é admitida como parte explicativa do comportamento de $y(t)$, teremos modelo ARX conforme a equação

(2.14).

$$A(q)y(k) = B(q)u(k) + e(k). \tag{2.14}$$

Então, pode-se obter o modelo ARX a partir da equação (2.6) fazendo $C(q) = D(q) = F(q) = 1$, sendo $A(q)$ e $B(q)$ polinômios arbitrários e, considerando que $A(q)y(t)$ é a parte regressiva e $B(q)u(k)$ a entrada externa. Obtém-se, assim, a equação (2.14).

Aparecendo o erro $e(k)$ diretamente na equação, o modelo ARX pode se classificado como um modelo de erro da equação (AGUIRRE, 2004).

A equação (2.14) pode ser reescrita da seguinte forma:

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k), \tag{2.15}$$

o que coloca em evidência as funções de transferência do sistema $H(q) = B(q)/A(q)$ e de ruído $C(q)/[D(q)A(q)] = 1/A(q)$ (LJUNG, 1987). A representação ARX pode ser representada pela Figura 2.3.

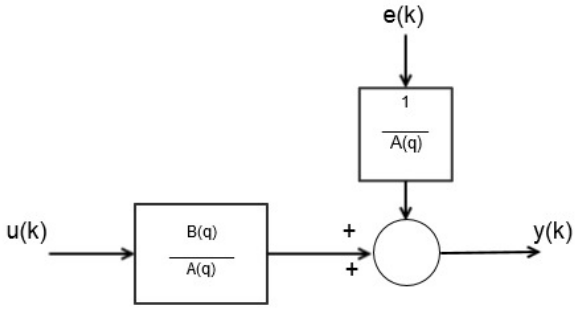


Figura 2.3: Representação do modelo ARX

Fonte: Adaptado de (AGUIRRE, 2004)

A Equação (2.14) pode ser reescrita em termos dos regressores e parâmetros dos polinômios $A(q)$ e $B(q)$ da seguinte forma:

$$y(k) = \psi^T\theta + e(k), \tag{2.16}$$

sendo $\psi^T = [y(k-1)\ y(k-2)\cdots y(k-n_a)\ u(k-1)\ u(k-2)\cdots u(k-n_b)]$ e $\theta^T = [a_1\ a_2\cdots a_{n_a}\ \ b_1\ b_2\cdots b_{n_b}]$. Dessa maneira, $y(k)$ é a saída, ψ e θ são denominados vetores de medidas e de parâmetros respectivamente, $e(k)$ os ruídos (AGUIRRE, 2007b). Na seção 2.4.2 será visto o estimador de mínimos quadrados, um dos mais utilizados na identificação de sistemas.

2.4.2 O Estimador de Minímos Quadrados

Após a determinação da estrutura de um modelo, deve-se estimar seus parâmetros para que se aproxime do comportamento do sistema. Um dos métodos mais conhecidos para obtenção de parâmetros a partir de dados experimentais é o método dos mínimos quadrados (AGUIRRE, 2004).

Levando-se em conta um conjunto de dados, o método de mínimos quadrados procura um melhor ajustamento para esses dados a fim de minimizar a soma dos quadrados das diferenças entre o valor estimado e os dados observados; essas diferenças são consideradas resíduos.

Essa teoria foi proposta por Karl Gauss para condução de seu trabalho sobre a predição das órbitas dos planetas (SORENSON, 1970). O objetivo principal desse método é determinar o melhor ajuste do modelo aos dados experimentais a partir da minimização do erro. Esse método é encontrado em diversos trabalhos sobre identificação de sistemas (AGUIRRE, 2004). Com base em (MOREIRA, 2008), (AGUIRRE, 2004), (GUERRA; GAINO, 2011) é mostrado o sistema de equação com solução única e posteriormente o caso sobredeterminado, para o qual há mais equações do que incógnitas.

2.4.2.1 Mínimos Quadrados

Considera-se uma função escalar $y = f(x)$ aplicada a N valores de x, de forma que:

$$\begin{aligned} y_1 &= f(x_1) \\ y_2 &= f(x_2) \\ \vdots &= \vdots \\ y_N &= f(x_N) \end{aligned} \tag{2.17}$$

No caso vetorial $f(x) : \mathbb{R}^n \to \mathbb{R}$ depende de um vetor θ de n parâmetros. Dessa forma, a função $f(x)$ é parametrizada por $\theta \in \mathbb{R}^n$ e pode ser representada como:

$$y = f(x, \theta). \tag{2.18}$$

Assim, tem-se um conjunto de equações a partir de várias observações do escalar y (variável dependente) e do vetor de variáveis independentes, da seguinte forma:

$$\begin{aligned} y_1 &= f(x_1, \theta) \\ y_2 &= f(x_2, \theta) \\ \vdots &= \vdots \\ y_N &= f(x_N, \theta), \end{aligned} \tag{2.19}$$

sendo que y_i é a i-ésima observação de y, e $x_i = [x_{1i,} x_{2i,} \cdots, x_{ni}]^T$ são as i-ésimas observações dos n elementos de vetor x. A função definida na equação (2.18) define uma família de equações, sendo que N membros dessa família estão representados na equação (2.19). Assim, cada membro será denominado *restrição*, ou seja, a equação (2.19) é um conjunto de N restrições da função descrita na equação (2.18).

Caso sejam conhecidos x_i e y_i $i \in \{1, 2, \cdots, N\}$, deseja-se determinar f e θ. Para isso, serão feitas as seguintes considerações.

1. A função f e o vetor θ não variam de uma restrição para outra, ou seja, todas as restrições são, de fato, da mesma equação.

2. A equação (2.18) pode ser escrita como

$$y = x^T \theta. \tag{2.20}$$

3. São consideradas n restrições, a fim de se ter n equações para determinar os n de θ, de forma que $N = n$.

Da consideração 1 fica claro que em problemas de identificação de sistemas dinâmicos, normalmente supõe-se que o sistema seja invariante no tempo e que os sinais medidos sejam estacionários. A consideração 2 implica que f seja linear nos parâmetros. A partir das considerações acima, pode-se escrever a equação (2.18) da seguinte forma:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix},$$

$$y = X\theta, \tag{2.21}$$

sendo $X \in \mathbb{R}^{n \times n}$ e x_i a i-ésima coluna de X (deve-se notar que x_i é um vetor coluna de n linhas, ou seja, $x_i \in \mathbb{R}^n$ que é diferente de $x_i \in \mathbb{R}$).

Na equação (2.21), y é a variável dependente, pois depende dos regressores $x_1, x_2, \cdots x_n$, que são também chamados de variáveis independentes. θ é o vetor de parâmetros a determinar. Pode-se determinar o vetor de parâmetros invertendo X (desde que X seja não singular), ou seja:

$$\theta = X^{-1}y \tag{2.22}$$

2.4.2.2 Sistemas Sobredeterminados

Caso houver $N > n$ restrições da equação (2.18), tem-se um sistema sobredeterminado, de forma que X $\in \mathbb{R}^{N\times n}$, y $\in \mathbb{R}^{N\times 1}$ e $\theta \in \mathbb{R}^{n\times 1}$. Como a matriz X não é quadrada, ela não pode ser invertida. Entretanto, pré-multiplicando a equação (2.21) por X^T em ambos os lados tem-se:

$$X^T y = X^T X\theta. \tag{2.23}$$

Dessa forma, $X^T y \in \mathbb{R}^{n\times 1}$ e $X^T X \in \mathbb{R}^{n\times n}$. Se $X^T X$ for não singular, pode ser invertida chegando-se a:

$$\theta = \left[X^T X\right]^{-1} X^T y, \tag{2.24}$$

sendo $\left[X^T X\right]^{-1} X^T$ chamada de matriz pseudo inversa. A equação (2.24) resolve o problema de determinação de θ quando se tem maior número de restrições do que parâmetros.

2.4.2.3 O Método de Mínimos Quadrados

Supondo que se conhece o valor estimado do vetor de parâmetros $\hat{\theta}$ e que é cometido um erro ξ ao se tentar explicar o valor observado y a partir dos regressores de x e de $\hat{\theta}$, ou seja:

$$y = x^T\hat{\theta} + \xi \tag{2.25}$$

Escrevendo de forma matricial, quando se tem $N > n$ medições da equação (2.25):

$$y = X\hat{\theta} + \xi. \tag{2.26}$$

Deseja-se encontrar $\hat{\theta}$ que minimize o valor do vetor de erros ξ. Este valor

será dado pelo somatório do quadrado dos erros ξ, ou seja:

$$J = \sum_{i=1}^{N} \xi(i)^2 = \xi^T \xi = \|\xi\|^2. \tag{2.27}$$

O custo J é uma quantia que mostra o quanto o vetor $\hat{\theta}$ se ajusta ás medidas de y e X. Quanto menor for J, melhor será esse ajuste. Isolando ξ na equação (2.26) e substituindo na equação (2.27):

$$J = (y - X\hat{\theta})^T (y - X\hat{\theta}) = y^T y - y^T X\hat{\theta} - \hat{\theta}^T X^T y + \hat{\theta}^T X^T X\hat{\theta}. \tag{2.28}$$

Para encontrar o vetor de parâmetros $\hat{\theta}$ que minimiza o valor de J, deve-se resolver $\frac{\partial J}{\partial \hat{\theta}} = 0$. Fazendo isso, tem-se

$$\frac{\partial J}{\partial \hat{\theta}} = -(y^T X)^T - X^T y + (X^T X + X^T X)\hat{\theta},$$

$$\begin{aligned} \frac{\partial J}{\partial \hat{\theta}} &= -(X^T y) - X^T y + 2X^T X\hat{\theta}, \\ &= -2(X^T y) + 2X^T X\hat{\theta}. \end{aligned} \tag{2.29}$$

Igualando a equação (2.29) a 0 tem-se:

$$\hat{\theta} = \left[X^T X\right]^{-1} X^T y. \tag{2.30}$$

Para que $\hat{\theta}$ corresponda ao mínimo de J, é necessário verificar que

$$\frac{\partial J^2}{\partial \hat{\theta^2}} = 2X^T X > 0,$$

o que é verdadeiro, pois $2X^T X$ é positiva definida por construção (AGUIRRE, 2004). Então conclui-se que as equações (2.24) e (2.30) são iguais.

2.4.3 Exemplo de Estimação de Parâmetros

Nesta seção, é discutido, com um exemplo, o processo de estimação utilizando o método mínimos quadrados. Para isso, utiliza-se um exemplo prático, que pode ser encontrado em (PHILLIPS; NAGLE, 1995). Suponha que um sistema de primeira ordem produza os dados mostrados na tabela 2.2:

k	u(k)	y(k)
0	1,0	0
1	0,75	0,3
2	0,50	0,225

Tabela 2.2: Tabela com dados para estimação.

Fonte: (PHILLIPS; NAGLE, 1995).

Assume-se a função de transferência

$$G(z) = \frac{b_1}{z - a_1},$$

cuja equação diferencial é:

$$y(k) = a_1 y(k-1) + b_1 u(k-1) = [y(k-1)\ u(k-1)] \begin{bmatrix} a_1 \\ b_1 \end{bmatrix}.$$

O objetivo, então, é encontrar o vetor de coeficientes

$$\Theta = \begin{bmatrix} a_1 \\ \overline{b_1} \end{bmatrix}.$$

Para esse exemplo, a forma matricial tem-se

$$\begin{bmatrix} y(1) \\ y(2) \end{bmatrix} = \begin{bmatrix} y(0) & u(0) \\ y(1) & u(1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0,3 & 0,75 \end{bmatrix}$$

então

$$X^T X = \begin{bmatrix} 0 & 0,3 \\ 1 & 0,75 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0,3 & 0,75 \end{bmatrix} = \begin{bmatrix} 0,09 & 0,225 \\ 0,225 & 1,5625 \end{bmatrix},$$

a inversa da matriz é calculada da seguinte forma:

$$[X^T X]^{-1} = \begin{bmatrix} 17,361 & -2,5 \\ -2,5 & 1 \end{bmatrix}.$$

A estimação de Θ pelo mínimos quadrados é dada por

$$\Theta = \left[X^T X\right]^{-1} X^T Y = \begin{bmatrix} 17,361 & -2,5 \\ -2,5 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0,3 \\ 1 & 0,75 \end{bmatrix} \begin{bmatrix} 0,3 \\ 0,225 \end{bmatrix}$$

$$= \begin{bmatrix} -2,5 & 3,333 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0,3 \\ 0,225 \end{bmatrix} = \begin{bmatrix} 0 \\ 0,3 \end{bmatrix} = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix}$$

Assim, a função de transferência identificada é

$$G(z) = \frac{0,3}{z},$$

e com sua equação diferencial

$$y(k) = 0,3u(k-1).$$

Neste capítulo, foram discutidos e apresentados conceitos importantes sobre a identificação de sistemas. Nesse trabalho, optou-se por trabalhar com a representação linear função de transferência. Também foram apresentadas as representações matemáticas paramétricas, em especial o ARX (*Autoregressive with Exogeneous inputs*) que são utilizadas em identificação de sistemas, e que usam para a estimação de parâmetros o algoritmo de mínimos quadrados.

3 *Toolbox* de Identificação de Sistemas para Uso com Matlab

Como discutido no capítulo 2, a identificação de sistemas é de grande importância para modelar sistemas que não são facilmente representados pelos seus termos físicos. A *Toolbox* de identificação de sistemas que integra o software Matlab permite ao usuário criar modelos matemáticos de um sistema dinâmico com base em suas entradas e saídas (LJUNG, 2007).

3.1 Utilização da *Toolbox* de Identificação de Sistemas

A *Toolbox* de identificação de sistemas contém uma grande variedade de técnicas para ajustar os parâmetros de todos os tipos de modelos lineares. Essa ferramenta facilita tarefas e reduz esforços, devido à facilidade de seu uso. Também ajuda examinar todas as propriedades dos modelos e ainda verifica se estão de acordo com o modelo desejado (validação), dessa forma, chega-se a modelos de processo sem mesmo conhecer profundamente os métodos de identificação.

Essa ferramenta possui uma GUI (*Graphical User Interface*) bastante interativa, a Figura 3.1 mostra a GUI. As funções estão, na sua maioria, contidas dentro da GUI e fornecem de uma maneira fácil acesso às variáveis envolvidas na sessão, tornando o processo de identificação e modelagem muito simples de ser executado (LJUNG, 2007). Assim, podemos encontrar um modelo caixa-preta, que é um dos objetivos desse trabalho.

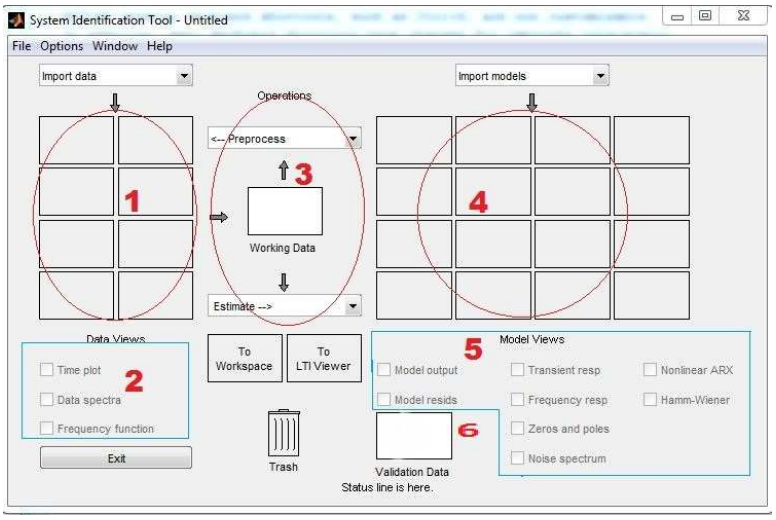


Figura 3.1: Divisão da GUI.

Fonte: (MATHWORKS2, 2014)

Para um melhor entendimento do uso desta ferramenta, a GUI foi dividida em seis regiões que estão mostradas numericamente, como mostra a Figura 3.1.

Na primeira região, indicada pelo número 1, ficam organizados os dados importados. A importação dos dados só é possível se os dados estiverem no *workspace* do software Matlab. Para importar os dados, usa-se a *pop-menu Import Data*. A GUI aceita apenas os seguintes dados (LJUNG, 2007):

- Dados no domínio do tempo;
- Dados no domínio da frequência;
- Dados estruturados em um objeto criado pelo Matlab.

Na região indicada pelo número 2, encontram-se opções para visualização dos dados que estão apenas na região 1. Os dados que o usuário quiser utilizar para estimar um modelo devem ser colocados no *working Data*, situado na região 3. Nessa região, existem duas caixas *pop-menu*: a primeira (*Preprocess*) condensa as funções de pré-processamentos de dados antes destes serem utilizados para estimar um modelo e a *pop-menu Estimate* que possui um conjunto de diferentes modelos para estimação, como por exemplo, modelos lineares e não lineares.

Na região 4, ficam os modelos que foram importados e posteriormente estimados. Já na região 5, tem-se um conjunto de *check-boxes* que permite ao usuário comparar e visualizar determinadas características de desempenho dos modelos estimados, entre as opções estão: *Model Resids*, onde é feita análise residual do modelo estimado; *transient resp*, para obter a resposta transitória; *frequency resp*, que obtém resposta em frequência e o diagrama de polos e zeros.

Por fim, são arrastados para a região 6 os dados que são utilizados para validar os modelos estimados. Esses dados geralmente não são os mesmos utilizados para a estimação do modelo.

Como exemplo, a Figura 3.2 mostra a GUI com dados importados do *workspace* do *Matlab* originais adquiridos de um secador de cabelos, e os dados resultantes do processo de tratamento desses sinais para obtenção de um modelo matemático para o sistema do secador. Os dados originais do secador são fornecidos juntamente com o a *Toolbox* de identificação para simulações e aprendizado. O modelo de um sistema real, o robô Lego, é estimado usando essa ferramenta.

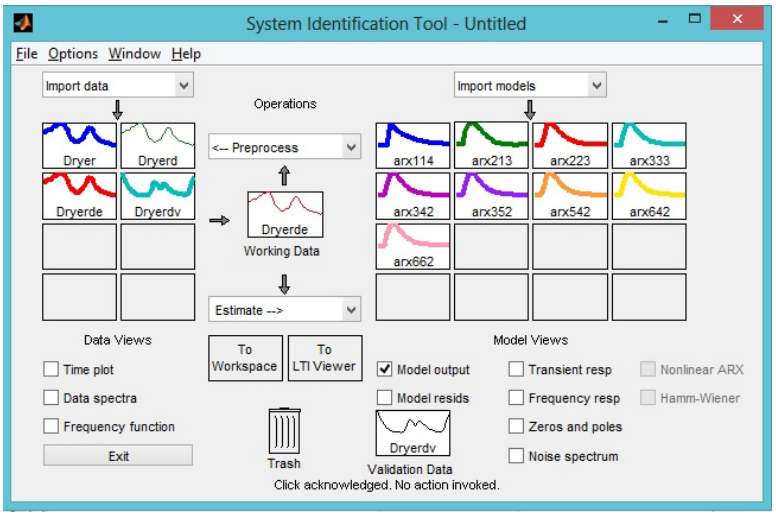


Figura 3.2: *Toolbox* com dados inseridos.

Fonte: Adaptado de (MATHWORKS2, 2014)

3.2 Escolha de um Modelo Paramétrico

Um dos passos importantes na identificação de sistemas é a escolha de uma família de modelos. A *Toolbox* de identificação suporta várias estruturas de modelos lineares. Nesta seção é mostrado os passos para a escolha de uma família de modelos ARX.

Para estimar um modelo linear na ferramenta *ident*, clica-se em *Estimate* > *Linear Parametric Models* situado na região 3. Assim, aparecerá a tela para escolha de modelos, como mostra a Figura 3.3.

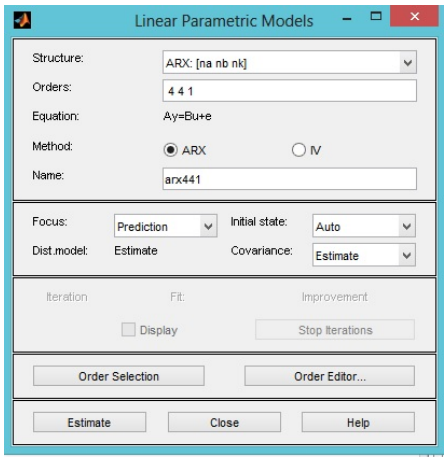
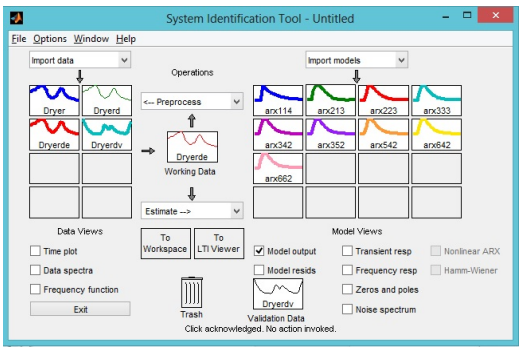


Figura 3.3: Escolha de modelos paramétricos

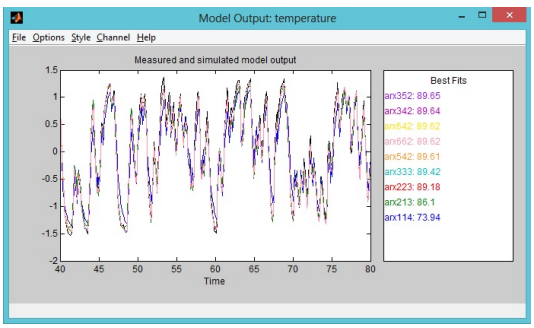
Fonte: Adaptado de (MATHWORKS2, 2014)

Na janela apresentada na Figura 3.3, observa-se que o modelo ARX vem selecionado por padrão no campo *Structure*. Nesse campo, têm-se outros tipos de modelos lineares tais como: ARMAX, OE, BJ. Após a escolha do modelo, a ordem do modelo deve ser colocada no campo *Orders* no formato n_a, n_b, n_c, se for um modelo ARX, em que n_a representa o números de polos, n_b os números de zeros e n_k representa o atraso que é o tempo necessário para que a entrada tenha efeito sobre a saída. No *check-box Method* escolhe-se a opção ARX. Fazendo isso, é escolhido o estimador de mínimos quadrados, que foi discutido na seção 2.4.2 . Para finalizar o processo, clica-se em *Estimate*.

Após o procedimento anterior, os modelos gerados são mostrados na GUI para análise do projetista. Para comparar graficamente a saída dos modelos estimados do sistema com sua saída real, deve-se selecionar a opção *Model Output* na janela GUI. Dessa forma, as curvas são geradas com um índice que qualifica em termos percentuais o modelo estimado na tabela *Best Fits*, Figura 3.4(a) e 3.4(b).



(a) Modelos gerados



(b) Modelos mostrados graficamente

Fonte: (MATHWORKS2, 2014)

Os modelos criados dentro da *GUI* não ficam disponíveis no *Workspace* do *Matlab*. Entretanto, qualquer modelo pode ser exportado. Essa operação é feita

facilmente apenas arrastando o modelo escolhido para o ícone *To Workspace* contida na *GUI*. Uma vez exportado o modelo, existem vários comandos de manipulação e conversão que podem ser usados nos modelos.

Em determinadas situações, o uso de ferramentas computacionais ajudam na compreensão e solução de problemas com sistemas dinâmicos. Nesse capítulo, foi abordada a utilização da *Toolbox* de identificação do Matlab em relação a sua manipulação. Essa ferramenta mostra-se importante para esse trabalho pela sua praticidade em desenvolver aplicações em modelagem e identificação de sistemas que será discutido em capítulos posteriores.

4 Sistemas de Controle

Sistemas de controle fazem parte da sociedade moderna. Existem inúmeras aplicações ao nosso redor: elevadores, braços mecânicos em processos industriais, lançamentos de foguetes, satélites e automóveis, entre outras. Existem também os processos naturais: no próprio corpo humano existem vários sistemas de controle, como o pâncreas que regula o açúcar no sangue e a adrenalina que aumenta os batimentos cardíacos, aumentando o oxigênio nas células (NISE, 2010). Neste capítulo discute-se vários conceitos de controle e o conceito de estabilidade.

4.1 Definição do Problema

Na área de controle, o primeiro trabalho importante de controle automático foi o regulador centrífugo construído por James Watt para o controle de velocidade de uma máquina a vapor, no século XVIII. Nos primeiros estágios do desenvolvimento da teoria de controle muitos trabalhos importantes foram desenvolvidos graças a Minorsky, Hazen e Nyquist (OGATA, 2003). Minorsky em 1922, por exemplo, demonstrou como a estabilidade poderia ser obtida a partir de equações diferenciais que descrevem o sistema e o autor também trabalhou com controladores automáticos para pilotagem de embarcações. Com a disponibilidade de computadores digitais, na década de 60, viu-se o desenvolvimento de uma abordagem em espaços de estados para lidar com a crescente complexidade dos sistemas modernos (OGATA, 2003). Atualmente, têm-se usado os computadores digitais como parte dos sistemas de controle. Por exemplo, o uso de computadores nos robôs industriais e em naves espaciais (NISE, 2010).

Um sistema de controle realimentado é um sistema que estabelece uma relação de comparação entre uma saída e uma entrada de referência, utilizando a diferença entre as duas como um meio de controle (OGATA, 2003). Para (DORF; BISHOP, 2001), um sistema de controle apresenta componentes conectados formando uma configuração de sistema que produzirá uma resposta desejada do sistema.

Para (NISE, 2010) sistema de controle consiste em subsistemas e processos

construídos para obter uma saída desejada, com um desempenho desejado, com uma entrada específica fornecida.

De certa forma, a maior preocupação de um sistema de controle é com a saída do sistema, ou seja, a partir de uma entrada específica podemos controlar as variáveis do sistema a fim de obter uma resposta desejada. Então, o uso de controle deve ser constituído de estratégias de projeto de controle para o aprimoramento de sistemas (DORF; BISHOP, 2001).

É necessária a definição de algumas terminologias para um melhor entendimento na discussão de sistemas de controles: *Variável controlada* é normalmente a variável de saída do sistema, ou a variável a ser medida e controlada no processo. Já a *Variável manipulada* é a variável que afeta a variável controlada, assim essa variável será manipulada pelo controlador. Na especificação de um projeto de controle temos o conceito de *planta ou sistema a controlar*, esse conceito pode ser definido como qualquer objeto ou equipamento que sofre ação de ser controlado. Exemplo: Um forno e um motor. A palavra *processo*, especificamente no campo da engenharia de controle, entende-se como qualquer operação a ser controlada. Por fim, um item importante em controle é a definição de *sistema*. Sistema é um objeto em que suas variáveis internas interagem entre si, produzindo sinais observáveis denominados saídas (OGATA, 2003),(NISE, 2010).

4.2 Sistemas de Controle de Malha Fechada e Aberta

Como definido anteriormente, o objetivo de um sistema de controle é obter uma saída com um desempenho desejado. Com a obtenção da saída, pode-se realimentar o sistema com um sinal, que irá mudar a saída final. Existe na literatura um vasto conteúdo a respeito do assunto tratado nessa seção:(ZHANG; SHIEH; DUNN, 2004), (NISE, 2010),(DORF; BISHOP, 2001), (ASTRÖM; HÄGGLUND, 2001).

Sistemas de controle de malha fechada ou sistema com retroação é um sistema que utiliza a resposta real, para compará-la com a resposta desejada. Por isso, para melhorar um processo, quase sempre é necessário introduzir uma retroação (BHATTACHARYYA; CHAPELLAT; KEEL, 1995).

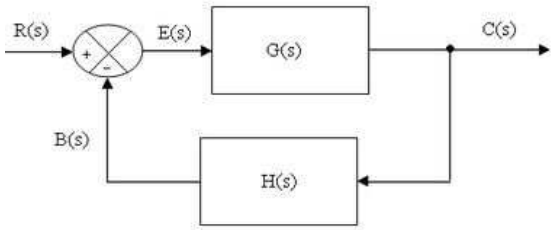


Figura 4.1: Diagrama de blocos de um sistema de malha fechada.

Fonte:(OGATA, 2003)

A Figura 4.1 mostra a representação clássica de um sistema de controle em malha fechada, onde a saída $C(s)$ é realimentada e comparada com $R(s)$. Então, $C(s)$ é obtida pela multiplicação da função de transferência $G(s)$ pela entrada do bloco $E(s)$. $H(s)$ pode ser considerado como um elemento de realimentação, cuja função, é fazer modificação no sinal de $C(s)$ antes de ser comparada com $R(s)$ (OGATA, 2003). Nota-se que $B(s)$ é o sinal que vai em direção ao ponto de soma. Com base no sistema da Figura 4.1 a função de transferência em malha fechada que relaciona C(s) com R(s) é assim obtida:

$$
\begin{aligned}
C(s) &= G(s)E(s) \\
C(s) &= G(s)(R(s) - B(s)) \\
C(s) &= G(s)(R(s) - H(s)C(s)) \\
C(s) &= G(s)R(s) - G(s)H(s)C(s) \\
C(s) + G(s)H(s)C(s) &= G(s)R(s) \\
C(s)(1 + G(s)H(s)) &= G(s)R(s).
\end{aligned}
$$

Como resultado temos,

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

Por outro lado, o sistema de malha aberta é aquele em que a saída não exerce influência no controle do sistema, ou seja, o sinal de saída não é medido e nem realimentado para uma comparação com a entrada. A equação da função de transferência em malha aberta é:

$$\frac{B(s)}{E(s)} = G(s)H(s)$$

Um diagrama que representa os sistemas em malha fechada é mostrado na Figura 4.2, apresentando os principais componentes do sistema (SILVA, 2012):

- **Referência**: Valor desejável da variável controlada.
- **Comparador**: Item que calcula o erro entre o valor desejado e o obtido.

- **Controlador**: Item que manipula o sinal de erro, gerando um sinal de controle que será usado no sistema, para corrigir a variável controlada.
- **Atuador**: Item que recebe o sinal do controlador e gera um sinal com potência suficiente para atuar sobre o sistema.

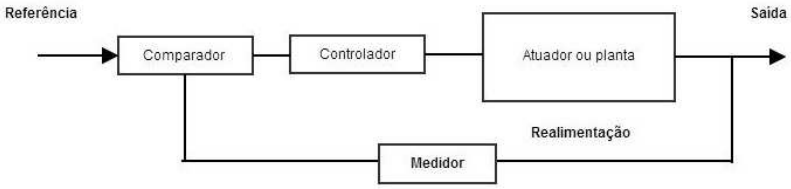


Figura 4.2: Diagrama de um sistema de controle em malha fechada.

Fonte: (SILVA, 2012)

Precisão é algo importante para controle. Sistemas em malha fechada têm mais precisão do que sistemas em malha aberta, porque a resposta da malha fechada é menos sensível a distúrbios, variações e mudanças no sistema. Por isso, é a estratégia mais utilizada em sistemas industriais. Além disso, a resposta transitória e a resposta estacionária desses sistemas podem ser facilmente controladas, muitas vezes, apenas ajustando o ganho do sistema ou até mesmo reprojetando um novo controle (OGATA, 2003), (NISE, 2010).

Então, um projeto de controle começa pelas especificações de desempenho. As especificações geralmente são medidas aplicando uma entrada degrau no sistema e fazendo sua análise transitória (OGATA, 2003). A Figura 4.3, mostra a curva de resposta a uma entrada em degrau de um sistema de segunda ordem. Com isso, podemos retirar alguns parâmetros e analisar algumas propriedades, relevantes da dinâmica do sistema dinâmico. Porém, essas propriedades valem somente para sistemas com 2 polos e nenhum zero.

- **Máximo valor de ultrapassagem** M_p: O máximo desvio da referência observado após a aplicação do degrau.
- **Tempo de subida**, t_r: É o tempo transcorrido para resposta ir de 10% a 90%, ou de 5% a 95% ou de 0% a 100% do seu sinal. Nos sistemas de segunda ordem subamortecidos, por exemplo, o tempo de subida utilizado é 0% a 100%. Esse é indicativo de quão é rápido o sistema a aplicação ao degrau.
- **Tempo de assentamento**, t_s: É o tempo necessário para que a resposta permaneça dentro de uma faixa percentual entre 2% e 5% em torno do valor de regime permanente.

- **Tempo do pico**, t_p: É o tempo para que a resposta atinja o primeiro pico de sobre-sinal.

- **Tempo de atraso**, t_d: É o tempo requerido para que a resposta alcance metade de seu valor final pela primeira vez.

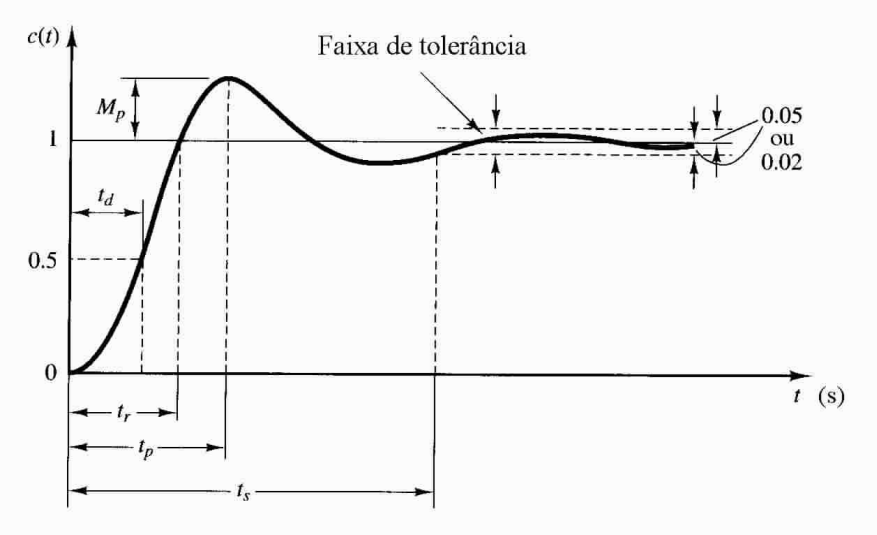


Figura 4.3: Indicativos à resposta degrau, para um sistema de segunda ordem.

Fonte: (OGATA, 2003)

4.2.1 Controladores PID

O Controlador Proporcional-Integral-Derivativo (PID) é uma estrutura de controle usada mundialmente em controle industrial. Este controlador tem como característica a simplicidade e um desempenho robusto no controle de muitas aplicações (ZHANG; SHIEH; DUNN, 2004). Apesar de seu grande uso comercial, essa estrutura é amplamente estudada desde a sua criação, tendo diversos métodos e ajustes pesquisados, como se verifica, por exemplo, em (ALIA; YOUNES; SUBAH, 2011) e (ASTRÖM; HÄGGLUND, 2001). O controlador PID pode ser ajustado para obter um desempenho satisfatório, com base em um nível modesto de informação (AGUIRRE, 2007a). Apesar da existência de vários métodos de ajuste existentes, o ajuste manual ainda é utilizado. Porém, sistemas complexos exigem métodos mais sistemáticos para um bom ajuste dos parâmetros, diversos métodos para esse problema podem ser verificados em (O'DWYER, 2009).

A estrutura PID é composta por três coeficientes: *Proporcional*, *Integral*, *Derivativo*, que podem ser variados a fim de obter a resposta ideal. Considere a malha fechada do sistema SISO na Figura 4.4:

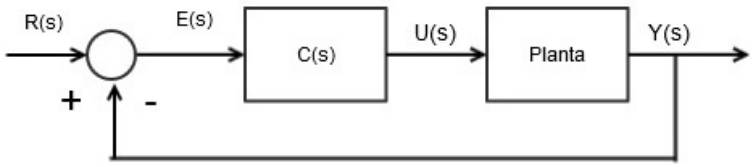


Figura 4.4: Malha Fechada -SISO.

Fonte: Elaborado pelo autor

As expressões para os controladores Proporcional (P), Proporcional-Integral (PI), Proporcional-Derivativo (PD) e Proporcional-Integral-Derivativo (PID) podem ser descritas pela relação da saída do controlador *U(s)* e o sinal de erro *E(s)=R(s)-Y(s)*. A seguir, as respectivas funções de transferências dos controladores (OGATA, 2003):

- Ação de controle proporcional (P):

$$\frac{U(s)}{E(s)} = K_p,$$

- Ação de controle proporcional-integral (PI):

$$\frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s}\right),$$

- Ação de controle proporcional-derivativo (PD):

$$\frac{U(s)}{E(s)} = K_p\left(1 + T_d s\right),$$

- Ação de controle proporcional-integral-derivativo (PID):

$$\frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right), \tag{4.1}$$

onde T_i é chamado de tempo integrativo, T_d é chamado de tempo derivativo e K_p é denominado ganho proporcional. Como observado nas ações de controle, os membros dessas estruturas podem ser combinados. Essa combinação tem a vantagem de usufruir das características individuais de cada uma. Analisemos agora os efeitos individuais:

- *Ação proporcional*: Essa ação depende do valor do erro *E(s)=R(s)-Y(s)*. Intuitivamente, essa ação leva a pensar que um sinal de controle maior deve ser aplicado ao processo quando esse se encontra distante do valor de referência, ou, aplicar um sinal de controle menor quando o valor de

referência estiver menor. Aumentando-se o ganho proporcional aumenta-se a velocidade de resposta do sistema de controle (AGUIRRE, 2007a).

- *Ação Integral*: Para (KIM; SCHAEFER, 2005), a ação integral fornece uma saída no controle proporcional ao erro acumulado. Assim, essa ação irá acumular o erro ao longo do tempo. A sua principal característica prática é conduzir o erro estacionário para zero.

- *Ação Derivativa*: Essa ação fornece a aplicação de um sinal de controle proporcional à derivada do sinal de erro (AGUIRRE, 2007a). A derivada de uma função se relaciona com uma tendência de variação dessa função em algum instante de tempo. Assim, essa ação tem característica preditiva, e tende a fazer o sistema reagir mais rapidamente.

4.2.2 Aplicação do Critério de Estabilidade de Routh-Hurwitz à Análise de Sistemas de Controle

No projeto de um sistema de controle em malha fechada, a estabilidade torna-se um assunto de muita importância. Um controlador instável não tem aplicação prática. Para (DORF; BISHOP, 2001), existem dois tipos de estabilidade: absoluta e relativa. Seguindo essas definições, um sistema que possua estabilidade absoluta é um sistema estável; por outro lado, se um sistema é estável, é possível caracterizar o grau de estabilidade desse sistema, pois, assim, é referido como uma estabilidade relativa. Para se determinar se um sistema é estável (no sentido absoluto), os polos da função de transferência desse sistema devem estar no semiplano s da esquerda. Com todos os polos na esquerda do semiplano, a estabilidade relativa é dada examinando-se a localização relativa dos polos (DORF; BISHOP, 2001) (NISE, 2010). A Figura 4.5 mostra um exemplo do plano s correspondente a um sistema instável, pelo fato de existir um polo no semiplano da direita.

Atualmente, para descobrir as raízes de uma equação de ordem qualquer, dispomos de poderosas ferramentas que facilitam muito o trabalho. Porém, nem sempre foi fácil assim. Para obter raízes de ordem superior a 2, era muito trabalhoso. Assim, em 1905, Routh-Hurwitz apresentou um trabalho que permitia conhecer a região de alocação das raízes de um sistema, sem conhecer a sua exata posição. Esse método indica a quantidade de polos no semi-plano direito (SPD), no semi-plano esquerdo (SPE) e sobre o eixo jw, sem definir as coordenadas dos

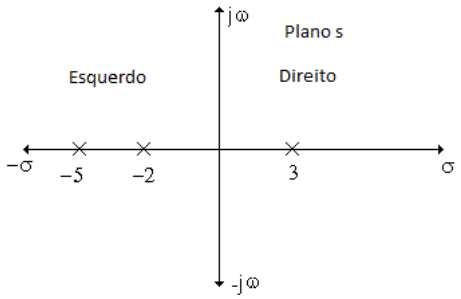


Figura 4.5: Plano *s*.

Fonte: Adaptado de (NISE, 2010)

polos. O critério de Routh-Hurwitz estabelece que a existência de um polo, ou mais, no SPD torna o sistema instável.

Segundo (NISE, 2010) o método de Routh requer dois passos:

1. Gerar o arranjo ou tabela de Routh-Hurwitz.
2. Interpretar os valores obtidos, identificando a localização dos polos e dizendo quantos estão no SPD, SPE e *jw*.

Na literatura de controle, esse assunto é tratado, detalhadamente, como em (PARKS, 1962), (NISE, 2010), (GIL et al., 2004), (PENA, 2004). Com base na equação do polinômio (4.2), a definição para montagem do arranjo de Routh é mostrada a seguir, segundo (OGATA, 2003). Considere o seguinte polinômio:

$$D(s) = a_0 s^n + a_1 s^{n-1} + ... + a_{n-1} s + a_n = 0 \tag{4.2}$$

Fazendo o arranjo fica:

$$\begin{array}{cccccc}
s^n & a_0 & a_2 & a_4 & a_6 & \cdots \\
s^{n-1} & a_1 & a_3 & a_5 & a_7 & \cdots \\
s^{n-2} & b_1 & b_2 & b_3 & b_4 & \cdots \\
s^{n-3} & c_1 & c_2 & c_3 & c_4 & \cdots \\
s^{n-4} & d_1 & d_2 & d_3 & d_4 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
s^2 & e_1 & e_2 & & & \\
s^1 & f_1 & & & & \\
s^0 & g_1 & & & & .
\end{array}$$

Os elementos a_0, a_1,$\cdots$,a_n são coeficientes de $D(s)$ e os elementos b_1, b_2, b_3 são

calculados como segue:

$$b_1 = \frac{a_1 a_2 - a_0 a_3}{a_1}$$
$$b_2 = \frac{a_1 a_4 - a_0 a_5}{a_1}$$
$$b_3 = \frac{a_1 a_6 - a_0 a_7}{a_1}.$$
$$\vdots$$

Dessa forma, o cálculo dos elementos da terceira linha continua até que os elementos restantes sejam todos zeros. O mesmo padrão de multiplicação em cruz dos coeficientes das duas linhas anteriores é seguido para o cálculo dos elementos das outras linhas, etc;

$$c_1 = \frac{b_1 a_3 - a_1 b_2}{b_1}$$
$$\vdots$$

O processo de formação das linhas continua até que se esgotem todos os elementos. Consideremos um exemplo prático. Admitamos a seguinte função de transferência:

$$G(s) = \frac{20}{s^3 + 10s^2 + 31s + 1030} \tag{4.3}$$

Admitindo o polinômio característico da função de transferência da equação 4.3, monta-se a tabela de Routh.

s^3	1	31
s^2	10	1030
s^1	-72	0
s^0	1030	0

Tabela 4.1: Exemplo de tabela de Routh.

Fonte: (GOODWIN; GRAEBE; SALGADO, 2001)

Analisando a tabela 4.1 podemos fazer algumas interpretações:

- O critério de Routh-Hurwitz estabelece que o número de raízes de um polinômio que está no SPD é igual ao número de mudanças de sinal que ocorrem nas células da 1 coluna. Então, de acordo a tabela 4.1, existem dois polos no SPD, deixando o sistema instável. Por outro lado, se não houvesse mudanças de sinal, significaria que o sistema seria estável.

- A cada mudança de (+) para (-) ou (-) para (+) significa que um polo

alocou-se no SPD.

Atualmente, o uso do computador é de essencial importância no auxílio e análises de projetos de controle de sistemas. O *software Matlab* pode auxiliar na análise de estabilidade fornecendo métodos em que se pode calcular os polos do polinômio característico. Matlab é uma linguagem de alto nível que possui um ambiente interativo para computação numérica, visualização e programação. Com essa ferramenta é possível também analisar dados, desenvolver algoritmos e criar modelos (MATHWORKS, 2014).

Com uso do *software* Matlab, podemos verificar, de uma maneira fácil o resultado de Routh-Hurwitz, fazendo o cálculo diretamente das raízes do polinômio característico, utilizando a função *roots*. Considerando como exemplo a tabela 4.1, nota-se que houve duas trocas de sinais na primeira coluna, indicando, assim, que existem dois polos no SPD. Portanto, o sistema é instável. Utilizando o software Matlab, digita-se o código mostrado abaixo. Ao final da execução do código(ans), nota-se de fato a existência de raízes instáveis.

$$
\begin{aligned}
&>> \text{numg} = [20]; \text{deng} = \begin{bmatrix} 1 & 10 & 31 & 1030 \end{bmatrix}; \\
&>> \text{roots(den)} \\
&\text{ans} = \\
&-13,4792 \\
&1,7396 + 8,6528\text{i} \\
&1,7396 - 8,6528\text{i}
\end{aligned}
$$

A tabela de Routh mostra a estabilidade e instabilidade. Porém, em algumas situações, queremos determinar um intervalo de valores que o sistema permite impor sem perder a estabilidade. Com a variação do ganho de um sistema, suas raízes vão se deslocando no plano s. Seguindo esse raciocínio, fica evidente dizer que é muito importante conhecer os limites de ganho a ser impostos em um sistema.

A partir de um exemplo extraído em (DORF; BISHOP, 2001), considere o sistema da Figura 4.6. Assim, dada uma variável K, o problema é achar uma faixa de ganho dessa variável, que o sistema permite impor, sem afetar a estabilidade. Observe o sistema abaixo:

A função transferência em malha fechada da Figura 4.6 é dada por:

$$T(s) = \frac{K}{s^3 + 2s^2 + 4s + K}. \tag{4.4}$$

Observando-se o polinômio característico da equação (4.4), podemos, então, mon-

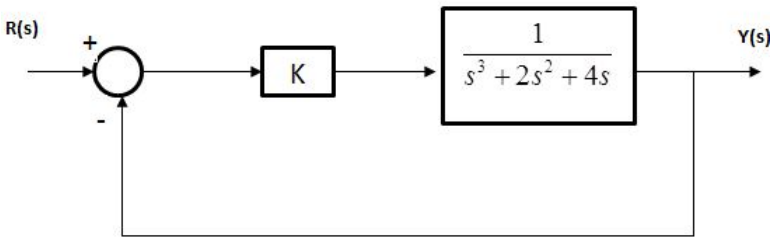


Figura 4.6: Controle de malha fechada

Fonte: Adaptado de (DORF; BISHOP, 2001)

tar a tabela de Routh:

s^3	1	4
s^2	2	K
s^1	(8-K)/2	0
s^0	K	0

Tabela 4.2: Tabela de Routh da função de transferência(4.4)

Analisando a tabela 4.2, nota-se que se K for positivo, toda a primeira coluna será também, exceto a célula da primeira coluna e linha s^1. Essa célula pode ser negativa, positiva e zero, dependendo do valor que K assumir. Se $K < 8$, todos os termos da primeira coluna serão positivos. Dessa forma, não haverá troca de sinal, e o sistema terá os polos no SPE e ser estável. No caso de $K > 8$, a célula da linha s^1 da primeira coluna é negativo. Sendo assim, haverá duas trocas de sinais, ou seja, o sistema tem dois polos no SPD portanto é instável. Por fim, se $K = 0$ significa que os polos estão sobre o eixo real jw. Para ter um sistema estável é necessário que:

$$0 < K < 8.$$

Para um melhor entendimento desse assunto, os resultados para K podem ser visualizados graficamente, mostrando a localização das raízes no plano s, utilizando-se o software Matlab. Dessa forma, observa-se na Figura 4.7 que, à medida que K aumenta, as raízes complexas do polinômio característico movem-se em direção ao SPD.

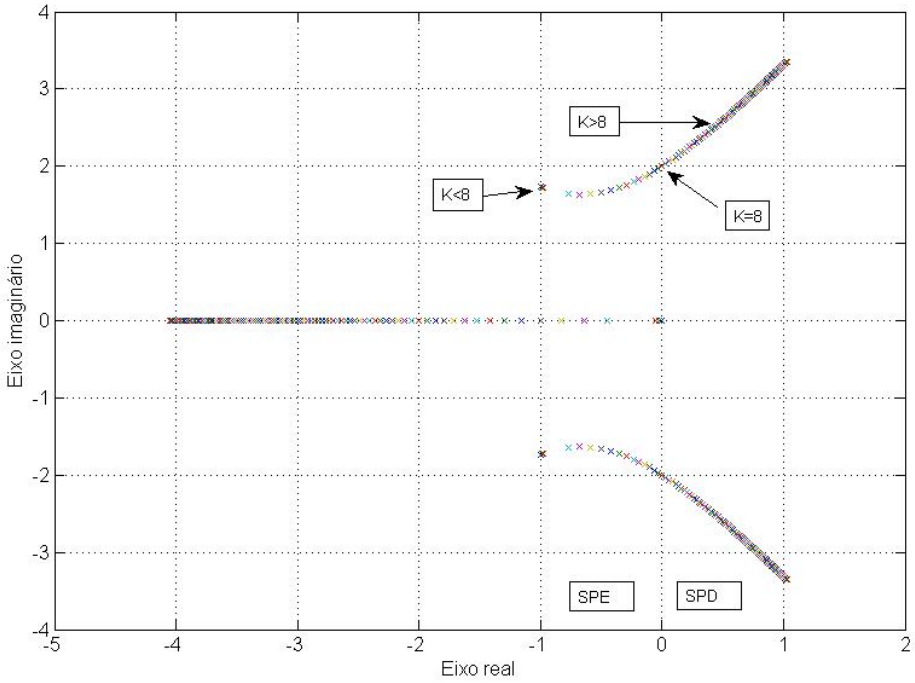


Figura 4.7: Gráfico da localização das raízes

Fonte: (DORF; BISHOP, 2001)

A estabilidade de um sistema é de grande importância em projetos de controle em malha fechada. Nesse capítulo, discutimos conceitos de estabilidade utilizando critério de Routh-Hurwitz, que é uma ferramenta útil para determinar a estabilidade do sistema. Esse estudo é importante pelo fato de que nesse trabalho é desenvolvido um controle de distância para o robô Lego Mindstorms, o qual será mostrado no próximo capítulo.

5 Robô *Lego Mindstorms NXT*

A automatização é cada vez mais empregada nas indústrias dando a possibilidade de substituição do trabalho braçal por máquinas automatizadas e de alta precisão (robôs). Porém, a Robótica vai além do sistema produtivo industrial, sendo utilizada também na educação e pesquisa (GUEDES; KERBER, 2011)(WEINBERG; YU, 2003).

O Robô Lego é uma ferramenta de pesquisa e ensino muito utilizada atualmente (SIQUEIRA; CONFORTO; VALLIM, 2012) e (GONÇALVES et al.,). Lego surgiu como brinquedo na década de 1930 e foi criado pelo dinamarquês *Ole Kirk Christiansen*. O criador fundiu duas palavras em dinamarquês para se obter o nome Lego: *leg godt* que significa "brincar bem". O produto mais importante do Grupo Lego é o *Lego brick*. Esse produto é composto de peças que se encaixam umas nas outras, permitindo assim várias combinações. Atualmente, o grupo Lego é o terceiro maior fabricante mundial de materiais lúdicos (MORTENSEN, 2013). A empresa Lego lançou o primeiro kit Mindstorms em 1998: um conjunto de peças especiais para montagem de robôs chamado de *Robotic Invention System*(RIS) juntamente com o *Robotic Control Explorer* (RCX), que é um módulo programável. A Figura 5.1 mostra o RCX. Essa invenção foi fruto da parceria

Figura 5.1: Robô RCX.

Fonte: (HURBAIN; GASPERI, 2007)

da *Lego Group* com o *Massachusetts Institute of Technology* (MIT). Além da sua função lúdica, o kit tem também uma função didática em instituições de ensino tecnológico, abordando teoria e prática de conteúdos relacionados à Robótica, permitindo o desenvolvimento de projetos (HURBAIN; GASPERI, 2007). O RCX é constituído de um processador de 8 bits, três entradas para conexão dos sensores, três saídas onde se conectam os motores, comunicação via infravermelho, um alto falante, e um display LCD de quatro dígitos.

Em 2006, foi lançado o Lego Mindstorms NXT, agora em uma versão mais avançada em relação ao seu antecessor. Com um processador mais potente e com sensores de luz, som, toque, servomotores, o kit Lego Mindstorms NXT permitiu a programação e criação de robôs com noção de distância, cores, movimentos, com um razoável grau de precisão (HURBAIN; GASPERI, 2007). Esse robô é composto por um sistema embarcado conhecido como *Intelligent Brick*(Tijolo inteligente), conforme mostra a Figura 5.2.

Figura 5.2: Tijolo inteligente.

Fonte: (LEGO, 2013)

Para ajudar o usuário do robô Lego em determinadas configurações, o *Intelligent Brick* contém 4 botões de interface e um monitor LCD preto e branco com matriz de pontos de 100 x 64 pixels com área de visualização 26 x 40,6 mm, conforme mostra a Figura 5.2.

Para o processamento das informações o *Intelligent Brick* é composto por um coprocessador e um processador principal, como mostra a Figura 5.3. O processador é o *Atmel* 32 *bits* ARM7 - AT91SAM7S256, que possui 256 *Kbytes*(KB) de memória *flash* que executa na frequência de 48 *Megahertz* (MHz), e que pode armazenar uma quantidade limitada de programas. Conta também com 64 *Kbytes*(KB) de memória *Ram*(Random Acess Memory). A função principal deste processador é controlar as funcionalidades específicas utilizadas pelos usuários

como: portas *USB*, *Display*, sons e *Bluetooth*.

O processador principal conta com o auxílio do coprocessador ATmega48 de 8 *bits*, que contém memória *flash* de 4 Kbytes(KB), 512 Bytes (B) de memória *Ram* que trabalha na frequência de 8 Megahertz(MHz) (LEGO GROUP, 2013). O coprocessador lida com as seguintes tarefas de baixo nível para auxiliar o processador principal (LEGO GROUP, 2013):

- **Gerenciamento de energia:** Controla o acionamento de ligar e desligar o robô quando pressionado o botão laranja, mostrado na Figura 5.2. Também monitora uso da bateria, informando o processador principal.
- **Geração PWM** (*Pulse-Width Modulation*): Gera pulsos para as três portas de saída, com uma frequência de 8 KHz, com o ciclo especificado pelo processador principal.
- **Conversor A/D:** Executa a conversão Analógica/Digital de 10 bits das portas de entradas.
- **Decodificador de botões:** Controla o uso do acionamento dos botões, informando ao processador principal quais botões estão sendo pressionados ou não.

Para lidar com todas as informações das funcionalidades do Lego, é necessário que o processador e o coprocessador troquem informações periodicamente. A comunicação entre os dois processadores é feita por duas memórias alocadas no *firmware*[1]do robô Lego, que utilizam a interface de hardware I^2C para comunicação, tendo o processador *ARM7* funcionando como mestre, como mostra a Figura 5.3.

O *firmware* usado pelo Lego é de código aberto, com o propósito de permitir o desenvolvimento de plataformas por terceiros, bem como propor melhoramentos do *firmware* adicionando novas funcionalidades, ou modificando as existentes (PESTANA, 2008). Por essa razão, a programação de aplicativos para esse robô pode ser feita em diversas linguagens tais como C, Java, Matlab, Python e NXC. Com isso, aumenta a liberdade de programação para esse equipamento criando, dessa forma, inúmeras possibilidades de desenvolvimento.

[1]Conjunto de instruções programadas em circuitos integrados, para controle do hardware.

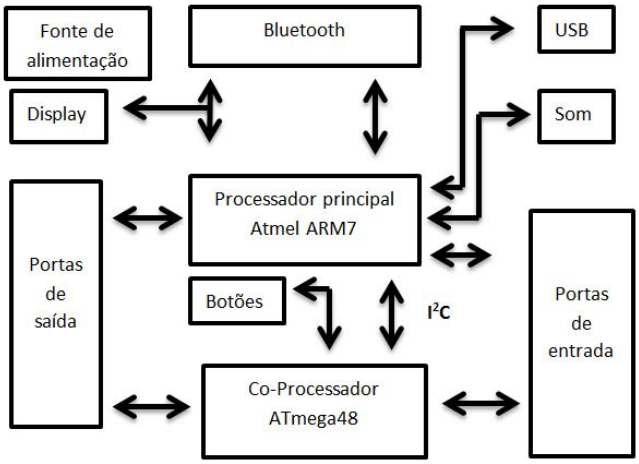


Figura 5.3: Diagrama do *Intelligent Brick*

Fonte: Traduzido de (LEGO GROUP, 2013)

5.1 Sensores e Motores

O Lego Mindstorms NXT possui 4 diferentes tipos de sensores, como pode ser observado na Figura 5.4 : um sensor de toque, um sensor de som, um sensor de luz e um ultrassônico. O sensor de toque detecta quando ele é pressionado por algo, já o ultrassônico mede a distância de um obstáculo. O sensor de luz pode medir a intensidade de luz em uma sala, por exemplo. Por fim, o sensor de som pode medir a intensidade do som em um ambiente. A seguir, são discutidas as especificações mais importantes desses sensores.

Figura 5.4: Sensores do Lego Mindstorms NXT

Fonte: (SOUSA, 2013)

- **Sensor de som:** Este sensor é capaz de medir os níveis de ruído de um ambiente e retornar uma medida em porcentagem de 0 a 100%. Pode ser

configurado em dB e dBA: dBA são sons que a audição humana está apta a ouvir entre 16Hz e 20kHz, ao contrário do dB que capta frequências de som que o ser humano não pode ouvir, mas que o sensor reconhece.

- **Sensor de luz:** O sensor de luz apenas habilita o robô a reconhecer entre o claro e o escuro. Também faz a leitura da intensidade de luz em um ambiente, e faz a medição da intensidade da luz em superfícies coloridas. A Figura 5.5 ilustra a visão do robô em relação à visão humana. Os blocos coloridos, mostrados na Figura 5.5, revelam como o olho humano enxerga, já a leitura feita pelo sensor de luz está representada pelos blocos em tons de cinza:

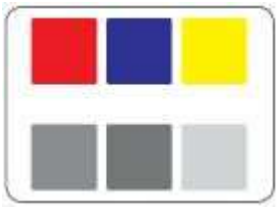


Figura 5.5: Leitura de sensor de luz.

Fonte: (SOUSA, 2013)

- **Sensor de toque**: Assim como os interruptores mecânicos simples, o sensor de toque do Lego Mindstorms pode ser pressionado ou liberado. Dentro de uma lógica, o valor zero significa que o sensor de toque não está pressionado, caso esteja pressionado seu valor será 1.

- **Sensor ultrassônico:** Uma das formas mais comuns de medição de distância utilizado em robótica são os sensores ultrassônicos (BORENSTEIN; EVERETT; FENG,)(YANG; BORENSTEIN; WEHE, 2000). Além de medir a distância, esses sensores facilitam reconhecimentos de objetos e evitam obstáculos. O princípio científico de funcionamento é o mesmo adotado por morcegos: enviam-se sinais curtos de ultrassom a 40 kHz e mede-se a distância calculando o tempo que uma onda de som demora para atingir um objeto e voltar (HURBAIN; GASPERI, 2007).

  No robô Lego, a distância é dada em centímetros ou em polegadas, e tem alcance de 0 a 2,5 metros com uma precisão de +/- 3 cm. Por fim, quanto maior a superfície melhor é a precisão do sensor, entretanto objetos curvos, finos e pequenos dificultam a leitura do sensor. O funcionamento do sensor ultrassônico é ilustrado na Figura 5.6.

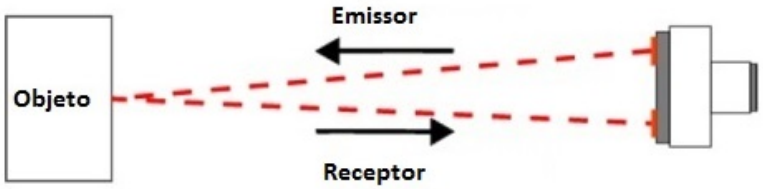


Figura 5.6: Ilustração do sensor de ultrassom

Fonte: Elaborado pelo autor

- **Motores:** Além de fazer os robôs se locomoverem, os servomotores são interativos, o que significa que eles contêm um sensor de rotação que permite ao usuário controlar interativamente a posição do eixo com um grau de precisão. O sensor é capaz de medir a rotação (uma rotação equivale a 360^o) em graus positivos (sentido horário) e graus negativos (sentido anti-horário). Desse modo, pode haver o controle dos movimentos de uma forma bem precisa. A Figura 5.7 mostra a estrutura interna do servomotor.

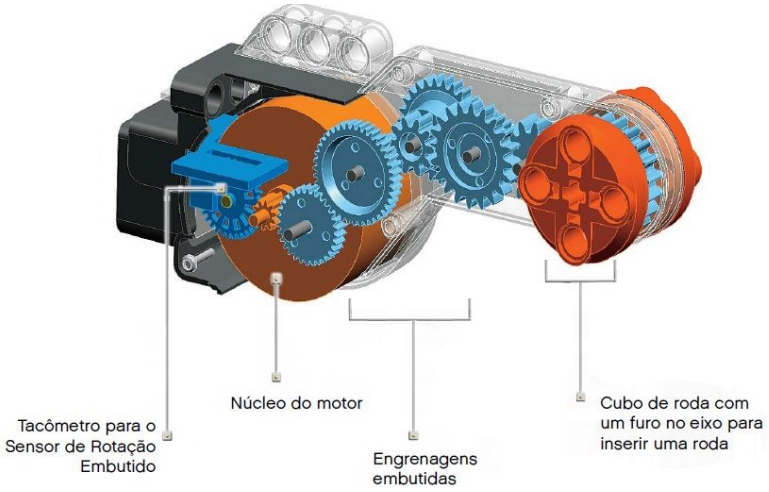


Figura 5.7: Servomotor do robô Lego

Fonte: (LEGO, 2013)

5.2 Programação e Captação de Dados em Robôs Lego

O uso de robôs envolve a criação de algoritmos que permitem executar determinadas tarefas como: controle, utilização de sensores e trajetórias. Por isso, o uso de ferramentas de programação é de essencial importância no desenvolvimento de projeto de robôs (SCHOLZ, 2007). Nessa seção, serão discutidas as principais características dos softwares utilizados nesse trabalho.

5.2.1 Bricx Command Center - BricxCC

Um ambiente de desenvolvimento integrado (*IDE - Integrated Development Environment*) é uma ferramenta de apoio que auxilia no processo de desenvolvimento de *softwares* com intuito de diminuir esforços dos programadores. Essas IDEs ajudam a utilizar as técnicas de RAD(*Rapid Application Development*) que são técnicas que enfatizam o desenvolvimento rápido de aplicações, aumentando assim, a produtividade. Os itens básicos que são encontrados nessas IDEs são (DEITEL, 2005):

- **Editor**: Tem a função de editar o código-fonte do programa suportado pela IDE.
- **Compilador**: É um programa de computador que, a partir do código-fonte, transforma uma linguagem de alto nível para uma linguagem de baixo nível, por exemplo, o código de máquina.
- **Depurador**: São programas que ajudam a encontrar erros de lógica e falhas em programas.

A IDE *Bricx Command Center* (BricxCC) é um programa que é suportado para todas as versões do sistema operacional *Windows*, fornecendo facilidades no desenvolvimento de programas e diversos controles de comunicação com robô Lego. Entre as facilidades oferecidas podemos citar: *Download* de programas para o robô, navegação pela memória *flash* e comunicação via *bluetooh*. O código dessa ferramenta é distribuído pela licença de *software* livre *Mozilla Public License*. *Software* livre é qualquer programa de computador que pode ser copiado, estudado e compartilhado seu código-fonte, de acordo com as necessidades de cada usuário (FREE SOFTWARE FOUNDATION, 2013). Sendo assim, o uso dessa ferramenta torna-se interessante para a diminuição de custos com licenças de *softwares* proprietários utilizados em projetos.

Essa ferramenta inclui suporte para a programação para os robôs Lego Mindstorms, utilizando diversas linguagens de programação. Entre as principais linguagens podemos citar: *Next Byte Codes* (NBC), *Not Quite C* (NQC), *Not eXactly C* (NXC) e Java. A configuração dessas linguagens é mostrada na Figura 5.8 (HANSEN, 2013).

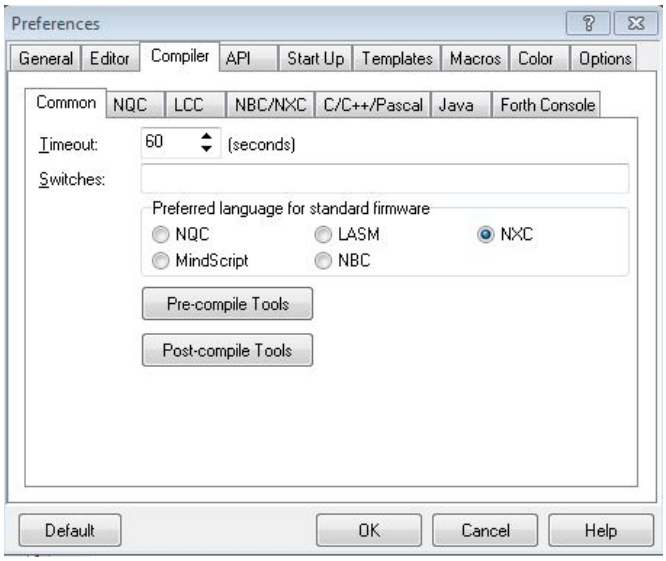


Figura 5.8: Linguagens usadas na IDE.

Fonte: (HANSEN, 2013)

A IDE BricxCC é composta por conjuntos de barras com o intuito de facilitar o seu uso, conforme mostra a Figura 5.9. Aqui são discutidos apenas alguns itens relevantes para esse trabalho: Barra de menu, Opção *Tools* e o Editor.

- **Barra de menu:** Nessa barra apresentam-se menus para prover funcionalidades a aplicação, tais como, abrir e fechar arquivos, edição de comandos e configurações do sistema.

- **Opção** *Tools*: Essa opção encontra-se na barra de menu, e é mais importante pelo fato de que ela provê funcionalidades específicas de manipulação do robô Lego, como: Controle remoto, gerenciamento de arquivos, compilação e comunicação da IDE com o robô.

- **Editor**: O editor pode ser dividido em duas partes, conforme revela a Figura 5.9, em que do lado esquerdo localiza-se uma lista que contém os elementos da linguagem utilizada na programação; já do seu lado direito está localizado o editor de algoritmos.

Com base nas características explicadas acima, esse trabalho utiliza ferramenta BricxCC e a linguagem NXC, que executará o algoritmo de controle utilizado nesse trabalho. Programas escritos em NXC são compilados para serem executados no *firmware* do robô Lego. A compilação do algoritmo gera *NXT bytecodes*, que são uma forma intermediária de código. Sendo assim, esses códigos podem ser lidos por um interpretador *NXT Bytecode* contido no robô Lego (SCHOLZ, 2007). Essa é uma configuração interessante dessa ferramenta

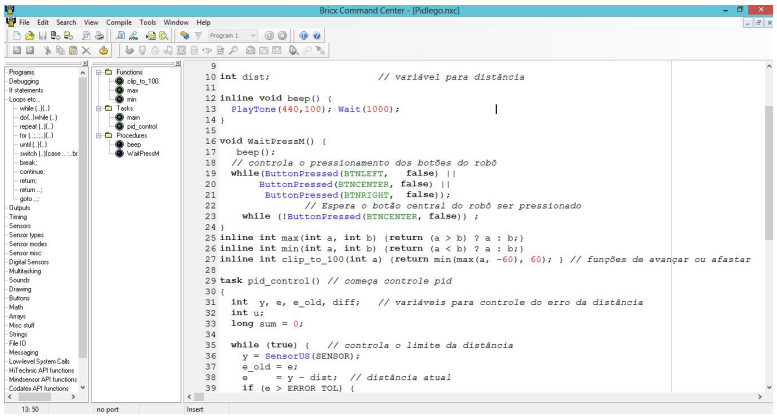


Figura 5.9: Editor do BricxCC.

Fonte:(HANSEN, 2013).

porque, ao contrário da linguagem de máquina pura que é dependente de uma plataforma de hardware, os *bytecodes* podem ser executados em qualquer plataforma mostrando assim sua portabilidade. A figura 5.10 ilustra o processo de execução do algoritmo na IDE BricxCC.

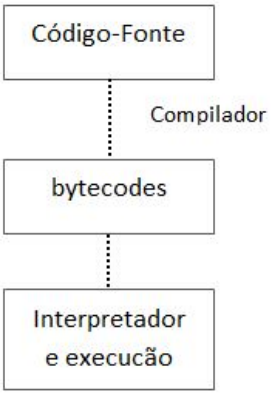


Figura 5.10: Execução do algoritmo na IDE.

Fonte:Elaborado pelo autor.

5.2.2 *Labview* para *Lego Mindstorms*

Labview é uma linguagem de programação gráfica que utiliza ícones e linhas, em vez de linhas de comando para criar aplicações. Essa ferramenta computacional tem como principal função criar programas direcionados para aquisição, geração e tratamento de dados, utilizando-se de placas de aquisição de dados.

É um software utilizado mundialmente por professores e cientistas para criação de sistemas que integram *hardware* e *software* (NATIONAL INSTRUMENTS , 2013). As ferramentas básicas que pertencem às bibliotecas de programação são representadas por ícones, que são conhecidas por VIs (*virtual instruments*) ou instru-

mentos virtuais. Esses instrumentos simulam ferramentas físicas ou operadores de dados (operações matemáticas, funções booleanas, plotagem de gráficos, gerador de funções). Essas funções são apresentadas através de formato de arquivo " vi " (PISANI, 2012).

Para a programação no *Labview* utilizam-se duas janelas principais: *Front Panel*(Painel Frontal), que se encontra do lado esquerdo da Figura 5.11 e *Block Diagram*(Blocos de Diagramas), que se encontra do lado direito da Figura 5.11.

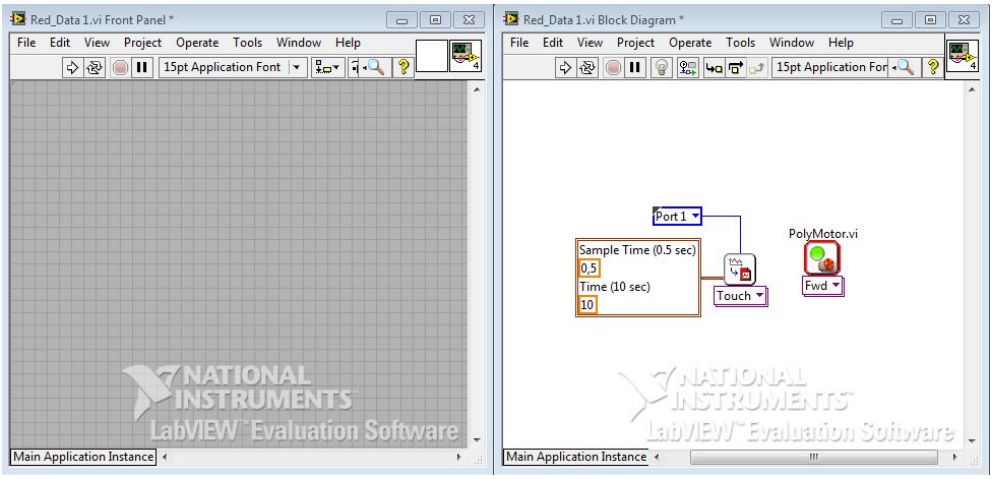


Figura 5.11: Interface *Front Panel* e *Block Diagram*.

Fonte: Adaptado de (NATIONAL INSTRUMENTS , 2013).

O *Front Panel* é a interface que o usuário utiliza para manipular os controles. Esse *Panel* permite introduzir controles que podem ser botões virtuais, dados gráficos e controles numéricos que mostram o estado atual do programa quando este está sendo executado. A cada controle inserido no *Front Panel* é associado um ícone no diagrama de blocos.

O diagrama de blocos é construído utilizando-se a linguagem gráfica de programação G. Então, pode ser entendido que o diagrama de blocos é o próprio código-fonte do programa. Esse diagrama é composto de estruturas de programação, como: funções aritméticas, laços de repetição e desvios condicionais. Ainda essas estruturas são ligadas por linhas e conectores que representam os fluxos de dados. Na Figura 5.12, são mostrados controles inseridos no painel frontal com seus respectivos ícones no diagrama de blocos.

Apesar de o *Labview* ter uma variedade de funções, é possível adicionar *toolkits* para desenvolver códigos para necessidades especiais. Nesse trabalho, é discutido *toolkit* para desenvolvimento para robôs Lego Mindstorms. Esse *kit* contém ferramentas que auxiliam na captação de dados dos sensores do robô Lego, que é relevante para esse trabalho.

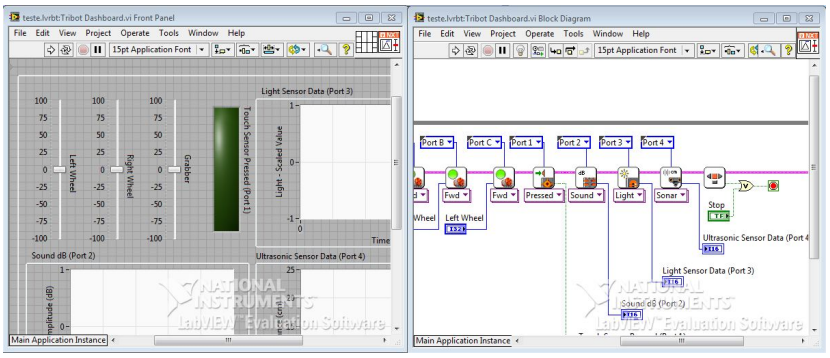


Figura 5.12: Execução do *Labview*.

Fonte: Adaptado de (NATIONAL INSTRUMENTS , 2013).

A captação de dados em robôs Lego Mindstorms é feita pelos componentes *Log Red Dataset*, *Log Green Dataset*, *Log Blue Dataset*; os dados captados são armazenados respectivamente na memória do robô Lego com os seguintes nomes: *RedData.dat, GreenData.dat, BlueData.dat*. A Figura 5.13 mostra os componentes de captação de dados.

Figura 5.13: Componentes de captação de dados do *Labview*.

Fonte: Adaptado de (NATIONAL INSTRUMENTS , 2013)

Como visto, no capítulo 2, a modelagem matemática baseia-se nas coletas de dados das características de entrada e saída do sistema e utilização destes para obtenção do modelo matemático.

Em (MOURA SILVA, 2012), foi projetado um *software* em *Labview* para captação de sinais proveniente de motores com objetivo de identificar e desenvolver controle digital e analógico com o kit DSP da Texas *instruments*. Com base em (MOURA SILVA, 2012), é desenvolvido neste trabalho um algoritmo para captação e armazenamento de dados utilizando-se de componentes contidos no *software Labview*. Na Figura 5.14, observam-se os principais componentes utilizados na captação de dados no robô Lego.

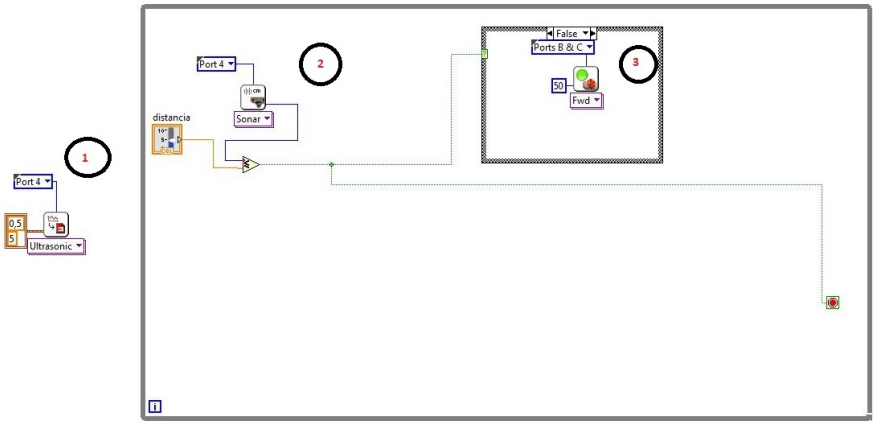


Figura 5.14: Coleta de dados pelo Labview.

Fonte: Adaptado de (NATIONAL INSTRUMENTS , 2013)

A Figura 5.14 está numerada com os números -1, 2, 3- para um melhor entendimento dos componentes utilizados para a captação de dados.

1. *Log Red Dataset*: Esse componente tem a função de captar dados provenientes da porta especificada do robô. No caso, mostrado na Figura 5.14 como *port* 4, representa a porta utilizada pelo robô para sinais de dados do sensor de ultrassom. Pode-se ainda configurar nesse componente o tempo total da captura de dados e o intervalo de tempo entre cada medida, dada em segundos. Os dados captados são armazenados na memória do robô Lego com o nome de *RedData.dat*. Procurando uma melhor otimização do espaço da memória do Robô Lego, os dados capturados são armazenados em formato binário.

2. *Read Ultrasound* : A leitura do sensor do ultrassom é feita por esse componente na porta especificada, mostrando a distância lida em centímetros.

3. Motor *on*: Esse componente faz o robô se locomover utilizando a porta especificada, com uma potência estabelecida pelo projetista.

Como dito anteriormente, sinais são captados e posteriormente armazenados na memória do robô Lego. Para a visualização e manipulação dos dados, a *Toolkit* para robôs Lego fornece o aplicativo *Data Viewer*. Pelo *Front Panel* a ferramenta é acessada pelos seguintes comandos *Tools/NXT application/Browser/Data Viewer*. A Figura 5.15 ilustra a ferramenta.

As principais funcionalidades da ferramenta *Data Viewer* estão em visualizar arquivos de dados armazenados no robô e também poder exportar esses dados em formato texto que estão delimitados por tabulação, de maneira que se possa

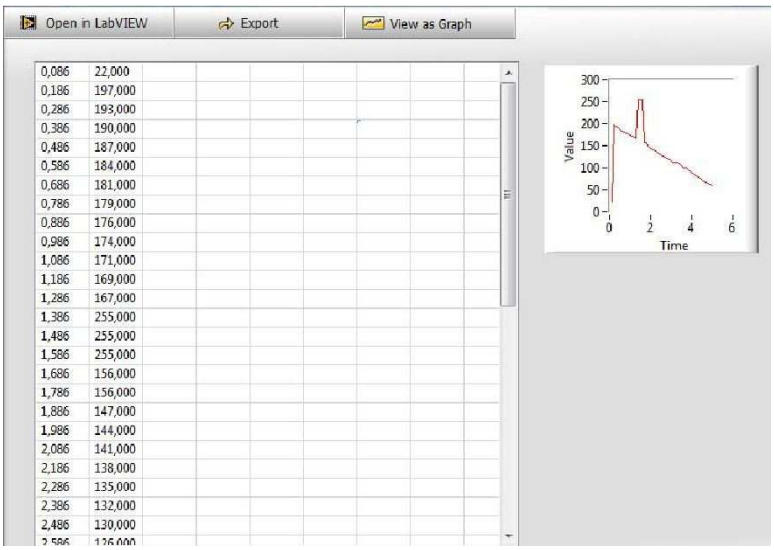


Figura 5.15: Ferramenta Data Viewer.

Fonte: Adaptado de (NATIONAL INSTRUMENTS , 2013)

abrir em planilhas eletrônicas, facilitando para o projetista uma análise de dados rápida.

Foram apresentadas, nesse capítulo a origem e as especificações técnicas do robô Lego Mindstorms, juntamente com as ferramentas necessárias para programação e captação de dados utilizando-se dessa plataforma. No próximo capítulo será descrito o desenvolvimento de um controlador PID, para que o robô mantenha uma distância determinada de um objeto em movimento.

6 Projeto de Identificação e Estabilidade do Robô *Lego Mindstorms NXT*

Atualmente, robôs já fazem parte do processo de automatização industrial. São cada vez mais essenciais para que as tarefas industriais possam ser executadas de maneira eficiente e rápida (ANDRADE; PEDRO FILHO; SILVA, 2013).

Robôs Lego são bastante utilizados como ferramenta de pesquisa e ensino atualmente (GONÇALVES et al.,). Esses robôs se locomovem em linha reta ou desviando-se de obstáculos, e param conforme algum critério estabelecido. Nessa pesquisa, procura-se desenvolver um controlador PID para auxiliar o robô Lego a manter-se a uma distância constante de um objeto em movimento em linha reta, tendo um grau de liberdade, conforme Figura 6.1.

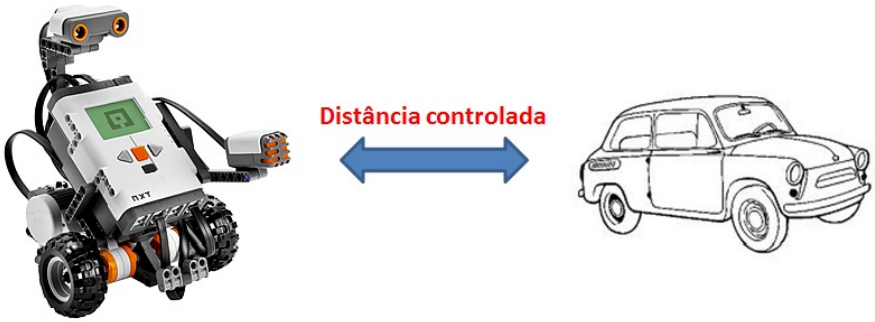


Figura 6.1: Distância do robô de um objeto.

Fonte: Elaborado pelo autor

Primeiramente é escolhida uma distância pré-definida a ser mantida, ou seja, é definido o *setpoint* do controle. Um processo de retroação então é estabelecido, no qual se usa o sensor de ultrassom do robô para emitir sinais contra o objeto em movimento. Esses sinais retornam e são comparados com o *setpoint*. A diferença obtida nessa comparação (erro) é utilizada como parâmetro de entrada de controle, que então atua na planta com o objetivo de diminuir o erro.

Para a identificação, nesse projeto, é adotada a estrutura de modelo de iden-

tificação caixa-preta, pelo fato de que não se tem nenhum conhecimento prévio do sistema. Essa estrutura pode ser considerada mais flexível e com uma boa aproximação do sistema real. Assim, é necessário fazer uma identificação do robô para que se obtenha a função de transferência que representa a relação matemática entre as entradas e saídas do sistema. Para isso, serão captados dados que relacionem a potência do motor e a distância de um objeto. Por fim, executando as etapas de um processo de identificação, um modelo matemático é obtido para o robô.

No projeto, é utilizado um controlador PID como descrito na equação (4.1). Esses controladores se baseiam em obter valores para K_p, K_d, K_i que mantenham o sistema de controle em malha fechada estável. O processo para obter os ganhos ideal para um controlador é chamado de ajuste. Nesse trabalho, a obtenção dos ganhos utiliza o polinômio característico do sistema e o critério de Routh-Hurwitz para estabilização. De posse desses ganhos, esses serão inseridos em um algoritmo de controle que será executado pelo robô.

Por fim, para uma simulação real desse projeto, o algoritmo de controle foi carregado no Kit-robótico *Lego Mindstorms NXT*, que contém um controlador lógico programável conectado a vários componentes: sensor ultrassônico, sensor de som, sensor de toque e servo-motores. Para o desenvolvimento desse projeto as seguintes etapas são desenvolvidas, conforme discutido no capítulo 2: Coleta de dados, obtenção do modelo do sistema, estimação de parâmetros, validação do modelo.

6.1 Coleta de dados

Como discutido no capítulo 2, a identificação constrói modelos matemáticos a partir de dados medidos. Segundo (AGUIRRE, 2004), às vezes, os dados obtidos são apenas da execução normal do sistema, entretanto, em outras situações, os dados são obtidos com a excitação do sistema com sinais adequados. Para esse projeto, os dados captados foram da operação normal de locomoção do robô. Para locomover-se, o robô Lego utiliza uma faixa de potência que pode variar de 0 a 100%. Dessa forma, uma série de valores de potência foi aplicada a fim de obter dados.

Para a identificação dinâmica do processo, foram coletados dados referentes à potência aplicada nos motores do robô e à distância percorrida pelo mesmo. Assim, foi aplicada aos motores do robô *NXT* uma potência fazendo o robô

locomover-se e, simultaneamente, coletar dados através do sensor ultrassônico da distância percorrida que, posteriormente, foram gravados na memória do robô através do componente *Log Red Dataset*. Posteriormente, foram carregados a partir da memória do robô usando-se a ferramenta *Data Viewer*, ambos pertencentes ao software *Labview*, uma plataforma de programação gráfica que fornece uma interface para robôs *Lego Mindstorms NXT*, como discutido na seção 5.2.2. A Figura 6.2 mostra a ferramenta de captura, e os primeiros dados coletados do processo.

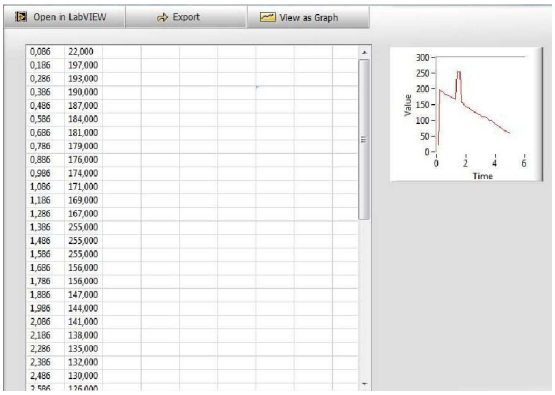


Figura 6.2: Dados adquiridos do *Labview*.

Fonte: Adaptado de (NATIONAL INSTRUMENTS , 2013)

6.2 Modelagem Matemática e Estimação

Na identificação, a escolha de uma estrutura que representa o comportamento do sistema dinâmico desejado é importante. Então, primeiramente, define-se se a representação é linear ou não linear e também se escolhem os métodos paramétricos e não paramétricos que serão utilizados para estimar os parâmetros do modelo (COELHO; COELHO, 2004), (AGUIRRE, 2004). Em se tratando de sistemas de controle que se baseiam em modelos, percebe-se a preferência por modelos lineares pelo fato do grande número de técnicas de controle baseadas nessas estruturas (FERNANDES, 2006).

Assim, essa etapa da identificação é considerada a mais difícil do processo pelo fato de que existe uma grande variedade de possibilidades que podem ser aplicadas, e um bom resultado depende diretamente da escolha da estrutura do modelo. Muitas vezes, um conhecimento prévio do sistema ou alguma característica ajuda na escolha da estrutura do modelo. Ainda na seleção de uma melhor estrutura, pode-se definir alguns parâmetros que possam melhorá-la, como por exemplo: ordem do modelo e o atrasos.

Definida a escolha de representação que será usada, o próximo passo é escolher a estrutura do modelo. Para (AGUIRRE, 2004), a escolha da estrutura do modelo é subjetiva por depender de vários aspectos: do uso pretendido, tipo de problema, da disponibilidade e qualidade dos dados utilizados, experiência do usuário e a complexidade do sistema. Além disso, a escolha de estrutura do modelo é uma tarefa normalmente feita por tentativa e erro.

De acordo com (LJUNG, 1994), na prática, diferentes estruturas devem ser testadas e comparadas para a obtenção de um modelo. (LJUNG, 1994), sugere ainda que primeiramente a escolha de uma estrutura deve ser iniciada por modelos mais simples, como o ARX(*Autoregressive with exogenous inputs*), que costuma ser a estrutura mais facilmente estimada. Assim, a utilização de uma estrutura mais complexa apenas se justifica se a mais simples não resolver o problema. Diante das ideias expostas anteriormente, no trabalho em questão é escolhida a representação paramétrica ARX.

Com os dados obtidos na coleta de dados, pode-se obter uma estrutura do modelo do robô usando a *Toolbox* de identificação de sistemas a qual possui uma interface gráfica (GUI) de fácil manipulação pelo usuário usada juntamente com o software *Matlab*. Essa ferramenta constrói modelos matemáticos de sistemas dinâmicos utilizando os dados obtidos da entrada e saída do sistema (LJUNG, 2007). A figura 6.3 mostra os modelos testados nesse trabalho.

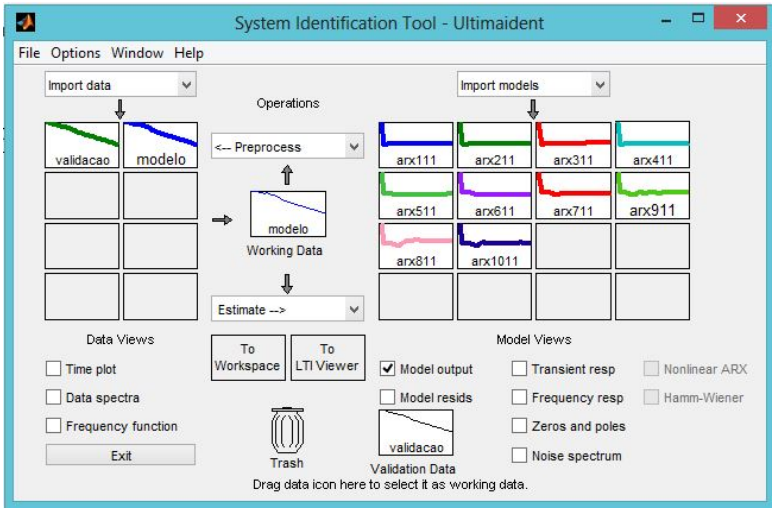


Figura 6.3: Modelos testados.

Fonte: Adaptado de (MATHWORKS2, 2014)

Utilizando a opção *Estimate* > *Linear parametric models* desta ferramenta, pode-se estimar modelos paramétricos regressivos ARX, como discutido na seção 3.1. Na Figura 6.4, observam-se os principais modelos ARX(na= números de polos, nb= números de zero, nk= atraso de tempo) estimados nesse trabalho.

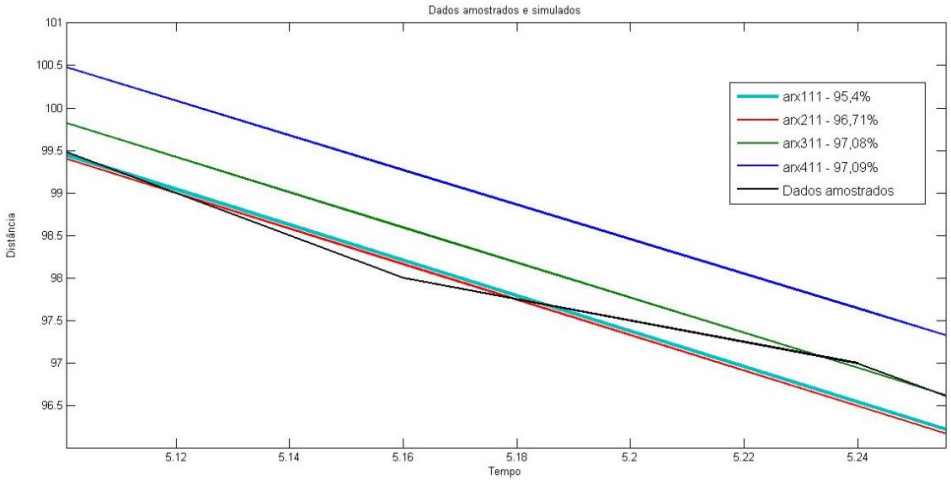


Figura 6.4: Principais modelos estimados usando *Toolbox*.

Fonte: Elaborado pelo autor

Como discutido anteriormente, a escolha de uma estrutura depende da sua complexidade. Um dos aspectos importantes na escolha da estrutura de modelos é a escolha da ordem do modelo. O que se procura é a menor ordem possível, que consiga se adequar satisfatoriamente aos dados. E que a escolha da ordem errada pode não representar a sua complexidade estrutural ou torná-lo mal condicionado (AGUIRRE, 2007b).

Para uma melhor observação a Figura 6.4 mostra apenas os quatro principais modelos obtidos. Os demais modelos testados são mostrados no Apêndice C. As estruturas obtidas pela *Toolbox* de Identificação são selecionadas com um índice que qualifica em termos percentuais os modelos estimados em relação aos dados amostrados, conforme mostra a Figura 6.4. Os modelos são estimados pelo estimador de Mínimos Quadrados discutido na seção 2.4.2, se for escolhida a estrutura estatística ARX.

Ainda analisando a Figura 6.4, os modelos estimados para os valores de na = entre 1 e 4, apresentaram resultados satisfatórios. Em contrapartida, os modelos de ordem na= entre 5 e 10 não apresentaram bons resultados, tendo um decaimento percentual, observe a Figura C.1 no apêndice C. Por causa disso, a escolha dos melhores modelos se restringe-se apenas a 4.

Então surge a questão: qual modelo escolher entre os que foram estimados satisfatoriamente? Para essa questão especificamente, em (SODERSTROM; STOICA, 1989) e (BOX; JENKINS; REINSEL, 2008) sugerem a utilização do princípio da parcimônia. Esse princípio é uma regra útil para a determinação da ordem de um modelo apropriado, ou seja, diante de dois ou mais modelos com ajustes igualmente bons, escolhe-se aquele com o menor número de parâmetros. As

justificativas para isso são: *a)* simplicidade na estrutura identificada e *b)* não usar parâmetros extras para descrever um sistema dinâmico, se eles não são necessários. Com base no exposto acima, a estrutura escolhida para representar o robô Lego é a ARX($na = 1$, $nb = 1$,$nk = 1$) por ter uma pequena complexidade e uma satisfatória aproximação com o modelo real.

6.3 Validação

A última etapa no procedimento de identificação é a validação. A validação ajuda a avaliar a capacidade de generalização do modelo, ou seja, se o modelo consegue captar a relação entre os dados de entrada e saída (LJUNG, 1994). Conforme (FERNANDES, 2006), geralmente utilizam-se métodos clássicos estatísticos de validação para avaliar a qualidade de um modelo, como por exemplo, a análise de autocorrelação dos erros de predição, análise da correlação cruzada entre entrada e resíduos, esses e outros métodos podem ser verificados em (AGUIRRE, 2004) e (SODERSTROM; STOICA, 1989). Porém, o fato de ser considerado estatisticamente válido não qualifica o modelo como a melhor opção para uma determinada aplicação. A qualidade de um modelo está relacionada fortemente com a finalidade pela qual foi obtido (FERNANDES, 2006). Dessa maneira, se o objetivo é identificar um modelo para o projeto de um controlador, o que importa será o desempenho encontrado nesse controlador quando for implementado em uma planta.

A maneira mais usual de validação é comparar a simulação do modelo obtido com os dados medidos. Por ser um procedimento simples, o seu uso é muito comum (AGUIRRE, 2004). Entretanto, alguns cuidados precisam ser observados.

Primeiramente, não se deve usar os dados utilizados na estimação para obter o modelo na validação. Esse cuidado torna-se necessário para verificar o resultado que ele consegue reproduzir para um conjunto de dados diferente dos usados para a obtenção do modelo, porém obtido do mesmo processo de observação (LJUNG, 1987). Esse processo é conhecido como validação cruzada.

Por isso, o ideal é efetuar dois testes independentes na observação do sistema gerando, dessa maneira, dois conjuntos de dados diferentes. Um deles é usado para a identificação do sistema e o outro para a validação. Em alguns casos, se os dois testes não puderem ser realizados, então poderá dividir o conjunto de dados em duas partes, sendo a primeira para a identificação e a segunda para validação (AGUIRRE, 2004) (LJUNG, 1987). Para facilitar o trabalho do usuário, a *Toolbox*

de identificação, discutida no capítulo 3, disponibiliza o recurso de separar o conjunto de dados em dois, acessando a *pop-up Preprocess > Select range*. .

Nesse trabalho, foram coletados dados tanto para a identificação quanto para a validação, ambos operando em condições semelhantes. Após isso, esses dados foram importados para *Toolbox* de Identificação e arrastados para o campo *Validation Data*. Com isso os modelos foram ajustados para o novo conjunto de dados e são mostrados conforme Figura C.1. Por fim, notou-se que não houve alterações nos modelos obtidos anteriormente, mostrando assim uma boa relação no conjunto de dados de estimação e validação.

6.4 Obtenção dos Ganhos do Controlador PID

Controlador PID é composto por um termo proporcional K_p, um termo integral $\frac{K_i}{s}$, e um termo derivativo $K_D s$ conforme (DORF; BISHOP, 2001). Sua função de transferência é representada pela equação (6.1).

$$C_{PID} = \left(K_p + \frac{K_i}{s} + K_D s. \right) \tag{6.1}$$

O processo em que os parâmetros do controlador são selecionados a fim de melhorar um índice de desempenho é conhecido como sintonia de controlador. A função dessa sintonia é ajustar os valores de K_p, K_i, K_d tomando como base a resposta experimental ao degrau e o valor de K_p (OGATA, 2003). Esse processo de sintonia pode também ser desenvolvido de forma empírica, porém pode tornar-se um trabalho difícil.

Primeiramente, o robô Lego utilizado neste trabalho necessita de que no seu algoritmo de controle sejam inseridos os parâmetros do controlador PID que lhe proporcionarão a devida estabilidade. Por isso, saber os parâmetros do controlador que tornarão o sistema estável é importante. A dificuldade em encontrar ganhos para o controlador usando o método de Routh-Hurwtiz aumenta conforme a ordem da planta. Sendo essas plantas de ordem elevada, fica difícil a investigação das mudanças de sinal na primeira coluna de tabela de Routh (TEIXEIRA; ASSUNÇÃO; COVACIC, 2007).

Em (TEIXEIRA; ASSUNÇÃO; COVACIC, 2007), é desenvolvido um algoritmo para obter a região de estabilidade para os valores dos ganhos do controlador, facilitando assim a pesquisa em sistema de alta ordem. Dessa forma, a determinação da região de estabilidade para os ganhos do controlador PID, como mostrado na Figura 4.4, utilizou o algoritmo em questão, o qual se baseia no polinômio ca-

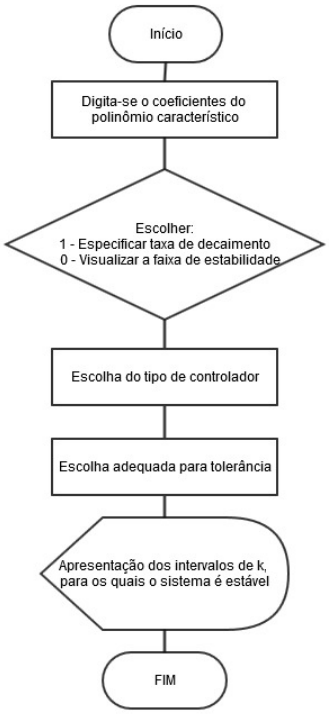


Figura 6.5: Fluxograma do Algoritmo de Ganho PID.

Fonte: Elaborado pelo autor

racterístico e determina um conjunto de ganhos do controlador que garante a estabilidade do sistema; o funcionamento deste algoritmo pode ser observado na Figura 6.5.

Para o uso desse algoritmo, digitam-se os coeficientes do polinômio característico e obtêm-se valores para os parâmetros K_p, K_i , K_d respectivamente. A execução do algoritmo começa digitando-se os coeficientes do numerador e denominador da função de transferência. No caso do robô, sua função de transferência é dada por

$$G(s) = \frac{0,5989s + 14,97}{s + 0,001097}.$$

Então, o numerador e o denominador devem ser digitados da seguinte forma [0.5989 14.97] e [1 0.001097], respectivamente. Após isso, escolhem-se mais duas especificações: taxa de decaimento (BOYD et al., 1994) ou apenas a estabilidade. Por fim, escolhe-se o tipo de controlador. Esse algoritmo contêm um pequeno erro de aproximação quando se encontra as raízes reais dos termos da primeira coluna, assim, algumas raízes repetidas podem ser vistas como raízes diferentes. Dessa forma, para resolver esse problema, duas raízes reais adjacentes z_i e z_{i+1}, são consideradas diferentes se somente se $|z_i - z_{i+1}| > \varepsilon$, onde $\varepsilon = 10^{-6}$ é a margem de tolerância (TEIXEIRA; ASSUNÇÃO; COVACIC, 2007).

Para o caso da escolha do PID, o programa solicita valores finais e iniciais que

devem ser digitados para K_d e K_i respectivamente. Assim, obtém-se para cada par (K_d, K_i) a faixa de estabilidade de K_p. A seguir, é mostrado o resultado da execução do algoritmo em questão:

```
Type the coefficients of N(s): [0.5989 14.97]
Type the coefficients of D(s): [1 0.001097]

Type "1" if you want to specify a decay rate or
"0" if you want to see the stability range: 0

Choose the type of the controller:
 1 - Proportional (P);
 2 - Integral (I);
 3 - Derivative (D);
 4 - Proportional-Integral (PI);
 5 - Proportional-Derivative (PD);
 6 - Integral-Derivative (ID);
 7 - Proportional-Integral-Derivative (PID);
 0 - Cancel.
Option: 7

Default tolerance is 10^(-6).
Is this value suitable for you (Y/N)? y

Type the initial value of kd: 5
Type the final value of kd: 10
Type the step of kd: 1
Type the initial value of ki: 5
Type the final value of ki: 10
Type the step of ki: 1
```

Assim, digitando os coeficientes obtém-se a região de valores para K_p, K_i, K_d. No caso do robô, alguns resultados obtidos são mostrados na tabela 6.1. Todas as soluções encontradas desse algoritmo, podem ser vistas no Apêndice A.

For kd = 5 and ki = 5:	Solution: kp > -0.0027063
For kd = 5 and ki = 6:	Solution: kp > -0.0032319
For kd = 5 and ki = 7:	Solution: kp > -0.0037572
For kd = 5 and ki = 8:	Solution: kp > -0.0042822
⋮	⋮

Tabela 6.1: Faixas de valores para o PID para alguns casos.

Fonte: Elaborada pelo autor

Por fim, considerando-se a escolha do controlador PID, é mostrada graficamente a região de estabilidade no final da execução do algoritmo como mostra a Figura 6.6.

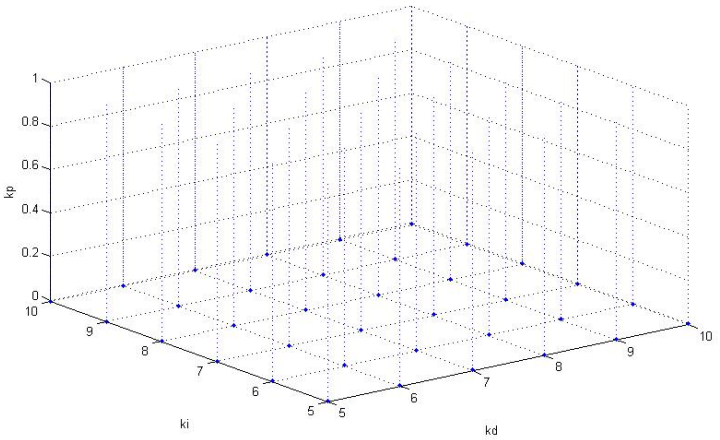


Figura 6.6: Faixa de estabilidade para o controlador PID.

Fonte: (TEIXEIRA; ASSUNÇÃO; COVACIC, 2007)

6.5 Resultados Obtidos e Discussões

Para simulação de sistemas, geralmente os modelos mais adequados são os que tiveram um melhor ajuste paramétrico. Então, após as etapas de identificação, chegou-se a alguns modelos que podem caracterizar, de forma adequada, o robô Lego como mostra a Figura 6.4 na seção anterior.

Os resultados obtidos mostram que o sistema pode ser representado por um modelo de baixa ordem (ARX111), apesar de que modelos de ordem superior também o podem. Porém, em testes realizados no robô, nota-se que o sistema de primeira ordem representa de maneira satisfatória a dinâmica desse sistema, haja vista que atingir 100% de um modelo pode não ser a solução do problema, pelo fato de que sistemas de ordem elevada podem ajustar-se bem a qualquer

conjunto de dados e que modelos complexos trazem um custo computacional elevado (LJUNG, 1987). Com a estrutura escolhida, podemos obter a função de transferência que representa o robô Lego Mindstorms, objetivo principal deste trabalho. Por fim, utilizando-se a função do Matlab *d2c*, transforma-se o modelo ARX, que é discreto, para um modelo contínuo, mostrando a sua função de transferência conforme equação (6.2).

$$G(s) = \frac{0,5989s + 14,97}{s + 0,001097} \tag{6.2}$$

Após a identificação, é testado o controle real no robô. A partir da tabela 6.1, foram definidos os ganhos $K_p = 10$, $K_i = 8$, $K_p = 5$ após o uso do algoritmo desenvolvido em (TEIXEIRA; ASSUNÇÃO; COVACIC, 2007), de forma a manter os valores distantes do limite da instabilidade.

Assim, o conjunto de valores inseridos no algoritmo de controle, que se encontra comentado no Apêndice B deste trabalho, está dentro da região de estabilidade. Por fim, o algoritmo é carregado na memória do robô utilizando-se a função *Download e Run* do *Bricx Command Center*. Para se implementar um controle PID discreto no Lego Mindstorms, usou-se a linguagem chamada NXC, que é criada especialmente para construir aplicações para Lego Mindstorms e, que é suportada por um ambiente de programação(IDE) chamado *Bricx Command Center*. Essa IDE auxilia na escrita de programas, no download dos programas para o robô, e nas ações de começar ou parar a execução do código, conforme discutido na seção 5.2.1.

Após a definição dos ganhos do controlador, torna-se importante testar o controlador para verificar se ele se mantém a uma distância pré-definida. Especificamente para este teste do controlador, estipulou-se que o robô permanecesse a uma distância de 21 cm de um objeto estático. Com base em dados reais coletados da memória do robô a execução em questão pode ser visualizada graficamente na Figura 6.7.

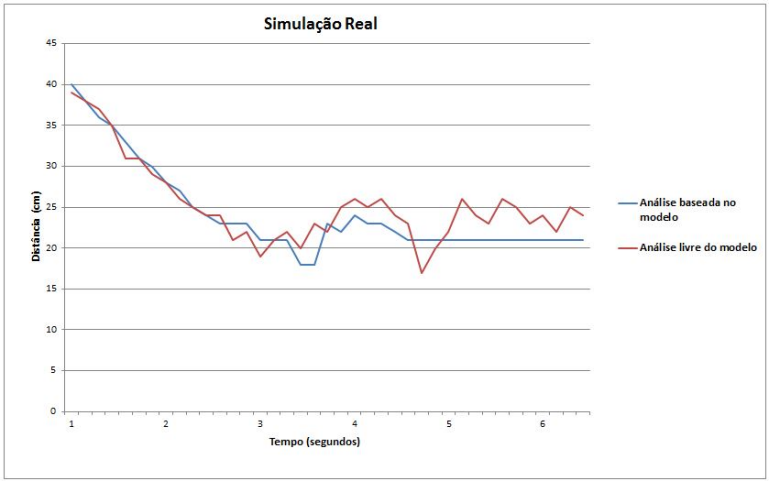


Figura 6.7: Simulação do controlador.

Fonte: Elaborado pelo autor

Nessa figura, observa-se uma análise livre do modelo, na qual digitam-se valores aleatórios para $K_p = 12$, $K_i = 11$, $K_d = -0,002831$, obtendo, assim, um sistema instável. O outro teste usando os valores obtidos do modelo mostra a estabilização do sistema na distância pretendida.

Como resultado, observa-se, na Figura 6.7, que a linha azul (análise baseada no modelo) parte aproximadamente da distância 40 cm e vai se estabilizando até chegar à distância pré-definida 21 cm, por outro lado, nota-se que a linha vermelha (análise livre do modelo), partindo do mesmo ponto da linha azul, não se estabiliza na distância de 21 cm. Diante disso, o projeto de identificação e estabilidade do robô mostrou-se eficiente em testes reais[1].

[1]O teste real do controlador pode ser visto em: `http://www.youtube.com/watch?v=W_2lkgNizyY`

7 Conclusão

Nesse trabalho, foi desenvolvida uma identificação de sistemas para a obtenção de um modelo matemático que representasse o robô Lego Mindstorms NXT em um ponto de operação. Com esse modelo, foi desenvolvido um controle PID para que o robô Lego se mantivesse a distância de um objeto em movimento em linha reta, ou seja, tendo um grau de liberdade.

Para esse projeto, técnicas de identificação paramétricas foram discutidas a partir de dados de entrada e saídas. Essas técnicas, utilizam-se de estruturas matemáticas parametrizadas para descrever o comportamento dinâmico original. Os parâmetros destas estruturas matemáticas são ajustados por algoritmos de estimação a partir dos dados medidos. Também, foi observado que a identificação de um sistema é desenvolvida com base em dois princípios: modelagem matemática e identificação de sistemas.

Para identificar o robô Lego, foi necessário captar dados que representasse a dinâmica do robô em movimento. Para essa ação, foi desenvolvido um algoritmo em *Labview* para que captasse e armazenasse os dados referentes a distância percorrida e a potência utilizada no motor em movimento seguindo em linha reta. Várias coletas de dados foram feitas, a fim de desenvolver a modelagem matemática e suas devidas estimações. Para o modelamento matemático especificamente, utilizou-se a *Toolbox* de identificação de sistemas do Matlab que contém ferramentas para ajuste de parâmetros, criação de modelos matemáticos, comparação de modelos, entre outros. O procedimento de identificação gerou vários modelos que puderam ser comparados, a fim de se obter, o melhor modelo que representasse o robô.

Na dinâmica do robô Lego, para manter-se a uma distância de um objeto, a estabilidade se tornou importante para que não houvesse oscilação de distância do robô. Para estabilizar esse sistema, o conceito de estabilidade de Routh-Hurwitz foi utilizado. Com esse conceito e o polinômio característico do robô adquirido na modelagem, parâmetros para o controlador P, I, D foram encontrados de maneira que deixasse o sistema estável. Então, esses parâmetros foram inseridos no algo-

ritmo de controle desenvolvido neste trabalho e executado pelo robô. Em diversos testes reais com locomoção, conclui-se que a identificação realizada aproximou-se do sistema dinâmico do robô, gerando assim, um controle que pode ser utilizado no robô com um grau de liberdade.

Portanto, esse trabalho mostrou técnicas alternativas de modelagem de um determinado sistema quando não se tem conhecimento matemático do mesmo, e ainda, desenvolveu um controle como resultado dessas técnicas. Contribuindo dessa forma, para trabalhos na área de identificação e controle em sistemas robóticos.

7.1 Sugestões para Trabalhos Futuros

Esse trabalho apresentou a técnica de identificação de sistemas e controle utilizando o kit robótico Lego Mindstorms, obtendo um resultado satisfatório. Após a leitura de vários trabalhos, testes e manipulações desse robô no decorrer do desenvolvimento desse trabalho, abriu-se um grande horizonte de ideias que podem ser colocadas como propostas para trabalhos futuros, porém citamos apenas algumas:

1. Desenvolvimento de um controlador, que dê ao robô dois graus de liberdade para se movimentar. E, ainda, podem ser utilizados de métodos de controle avançado.

2. Aplicação de controles remotos do robô utilizando a tecnologia *Blueto-oth*. Com isso, várias ideias podem ser implementadas na área de controle: veículos controlados remotamente, aquisição de dados em tempo real, analisar remotamente o desempenho de sistemas de controle desenvolvidos.

3. Implementação do kit robótico Lego nas aulas da disciplina de controle, para que o aluno entenda as técnicas apresentadas em aulas teóricas aplicadas a um modelo real. Dessa forma, os seguintes temas podem ser explorados: estudos de sinais, projeto de controladores, estudo de sistemas não lineares.

Referências

AGUIRRE, L. A. *Introdução à identificação de sistemas–Técnicas lineares e não-lineares aplicadas a sistemas reais*. Belo Horizonte: editora UFMG, 2004. ISBN 9788570415844.

AGUIRRE, L. A. *Enciclopédia de automática: Controle e automação volume II*. São Paulo: Blucher, 2007. ISBN 9788521204091.

AGUIRRE, L. A. *Enciclopédia de automática: Controle e automação volume III*. São Paulo: Blucher, 2007. ISBN 9788521204091.

ALIA, M. A.; YOUNES, T. M.; SUBAH, S. A. A design of a PID self-tuning controller using Labview. *Journal of Software Engineering and Applications*, v. 4, n. 3, p. 161–171, 2011.

ANDRADE, P.; PEDRO FILHO; SILVA, H. Sensores do Lego Mindstorms e robótica educacional. *Revista Eletrônica Científica Inovação e Tecnologia*, v. 1, n. 1, 2013. ISSN 2175-1846.

ASTRÖM, K. J.; HÄGGLUND, T. The future of pid control. *Control engineering practice*, Elsevier, v. 9, n. 11, p. 1163–1175, 2001.

BHATTACHARYYA, S. P.; CHAPELLAT, H.; KEEL, L. H. *Robust control*. [S.l.]: Prentice-Hall Upper Saddle River, New Jersey, 1995.

BIAZETO, A. R. Contribuições de identificação de sistemas linear e não-lineares para o caso do paciente paraplégico. *Trabalho de conclusão de curso apresentado na Universidade Estadual de Londrina*, 2011.

BORENSTEIN, J.; EVERETT, H.; FENG, L. *Where am I? Sensors and methods for mobile robot positioning*. Disponível em: <`http://ftp.eecs.umich.edu/people/johannb/pos96rep.pdf`>. Acesso em: 02/02/2014.

BOX, G. E.; JENKINS, G. M.; REINSEL, G. C. *Time series analysis: forecasting and control - quarta edição*. [S.l.]: John Wiley & Sons, Inc., 2008. ISBN-978-0-470-27284-8.

BOYD, S. P.; GHAOUI, L. E.; FERON, E.; BALAKRISHNAN, V. *Linear matrix inequalities in system and control theory*. [S.l.]: SIAM, 1994.

BRANDOLT, H. G. *Simulação de Escoamento em Dutos por Caracterização de Eventos*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina Centro Tecnológico Programa de Pós-Graduação em Engenharia Química, 2002.

CARVALHO, A. S. *Modelagem de Colunas de Destilação Através de Modelos Auto-Regresivos*. Dissertação (Mestrado) — Universidade Estadual do Norte Fluminense, 2008.

COELHO, A. A. R.; COELHO, L. dos S. *Identificação de sistemas dinâmicos lineares*. Florianópolis: Editora da UFSC, 2004. (Didáctica). ISBN 9788532802804.

DEITEL, H. M. D. e P. *Java como programar*. 6. ed. [S.l.]: Prentice Hall, 2005.

DORF, R. C.; BISHOP, R. H. *Sistemas de Controle Modernos, oitava edição*. Rio de Janeiro: LTC - Livros Técnicos e Cientifícos Editora S.A, 2001.

FERNANDES, F. de S. *Identificação por predição de erro e síntese de controladores robustos*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais - Pós Graduação em Engenharia Elétrica, 2006.

FERNANDEZ, E. M. F. *Controle preditivo com enfoque em subespaços*. Dissertação (Mestrado) — Escola Politécnica, Universidade de São Paulo, 2009.

FERREIRA, C. L. L.; CERVANTES, S. G. de S.; GERMANOVIX, W. *Cadeira de rodas controlada por sopro e sucção*. Disponível em: <`http://www.labplan.ufsc.br/congressos/CBA2008/textos/CBA_2008_Artigos/40057.pdf`>. Acesso em: 02/02/2014.

FONSECA SOBRINHO, A. S.; FELIZARDO, K. R.; GERMANOVIX, W.; GAINO, R. Sistemas de controle para cadeira de rodas comandados por sopro e sucção. *In:Simpósio Brasileiro de automação Inteligente - Bauru*, p. Sociedade Brasileira de Automática p. 1–6, 2003.

FONSECA SOBRINHO, A. S.; FELIZARDO, K. R.; SILVA, M. A. da; OLIVEIRA, H. P.; LONE, L. P.; GERMANOVIX, W.; GAINO, R. Cadeira de rodas controlada por sopros e sucções. *Semina: Ciências Exatas e Tecnológicas*, v. 21, n. 4, p. 3–7, 2000.

FREE SOFTWARE FOUNDATION. *Filosofia do Projeto GNU*. 2013. Disponível em: <`http://www.gnu.org/philosophy/free-sw.pt-br.html`>. Acesso em: 11/01/2014.

GAINO, R. *Controle de Movimentos de Pacientes Paraplégicos Utilizando Modelos Fuzzy TS*. Tese (Doutorado) — Universidade Estadual Paulista - Campus de Ilha Solteira, 2009.

GENTILHO JUNIOR, E. Controle implementado em dsp para cadeira de rodas acionada por sopro e sucção. In: *XI SBAI - Simpósio Brasileiro de Automação Inteligente*. [s.n.], 2013. Disponível em: <http://www.sbai2013.ufc.br/pdfs/8244.pdf>. Acesso em: 02/03/2014.

GIL, J. J.; AVELLO, A.; RUBIO, A.; FLOREZ, J. Stability analysis of a 1 dof haptic interface using the routh-hurwitz criterion. *Control Systems Technology, IEEE Transactions on*, IEEE, v. 12, n. 4, p. 583–588, 2004.

GONÇALVES, J.; LIMA, J.; MALHEIROS, P.; COSTA, P. *Sensor and actuator stochastic modeling of the Lego Mindstorms NXT educational Kit*. Disponível em: <`http://hdl.handle.net/10198/2262`>. Acesso em: 23/08/2013.

GOODWIN, G. C.; GRAEBE, S. F.; SALGADO, M. E. *Control system design*. [S.l.]: Prentice Hall Upper Saddle River, 2001.

GUEDES, A. L.; KERBER, F. M. Usando a robótica como meio educativo. *Unoesc & Ciência-ACET*, v. 1, n. 2, p. 199–208, 2011.

GUERRA, L.; GAINO, R. Identificação paramétrica de dados antropométricos de paciente paraplégicos através do método dos mínimos quadrados. *Revista Júnior de Iniciação Científica em Ciências Exatas e Engenharia*, ICCEEG, v. 1, n. 1, p. 15–23, 2011.

HANSEN, J. *Bricx Command Center 3.3*. 2013. Página Oficial. Disponível em: <http://bricxcc.sourceforge.net/>. Acesso em: 23/08/2013.

HOWARD, A.; IAGNEMMA, K.; KELLY, A. *Field and Service Robotics - Results of the 7th International Conference*. [S.l.]: Springer-Verlag Berlin Heidelber, 2010. ISBN: 978-3-642-13407-4.

HURBAIN, P. E.; GASPERI, M. *Extreme NXT: Extending the LEGO Mindstorms NXT to the Next Level*. 2. ed. New York: Apress, 2007.

KIM, K.; SCHAEFER, R. C. Tuning a PID controller for a digital excitation control system. *Industry Applications, IEEE Transactions on*, IEEE, v. 41, n. 2, p. 485–492, 2005.

KIM, Y. Control systems lab using a Lego Mindstorms NXT motor system. *Education, IEEE Transactions on*, IEEE, v. 54, n. 3, p. 452–461, 2011.

LEGO. *Building Instruction*. 2013. Disponível em: <http://mindstorms.lego.com/en-us/support/buildinginstructions/8547/8547>. Acesso em: 23/07/2013.

LEGO GROUP. *Lego Mindstorms Nxt - Hardware Developer Kit*. 2013. Disponível em: <http://zeus.nyf.hu/~simona/NXT-Hardware.pdf>. Acesso em: 02/02/2014.

LJUNG, L. *System identification: theory for the user*. [S.l.]: Prentice-Hall, 1987. (Prentice-Hall information and system sciences series). ISBN 9780138816407.

LJUNG, L. *Modeling of dynamic systems*. [S.l.]: Prentice Hall, 1994. (Prentice Hall information and system sciences series). ISBN 0-13-597097-0.

LJUNG, L. *System Identification Toolbox 7: User's Guide*. [S.l.]: MathWorks, Incorporated, 2007.

LUCIANO, A. *UEL desenvolve Cadeira Movida a Sopro*. 2014. Seção - Vida na Universidade. Disponível em: <http://www.gazetadopovo.com.br/vida-universidade/pesquisaetecnologia/conteudo.phtml?id=1450925>. Acesso em: 03/05/2014.

MALDONADO, M. A. R. *Modelagem e Simulação do Sistema de Controle de uma Micro-turbina a Gás*. Dissertação (Mestrado) — Universidade Federal de Itajubá, 2005.

MANTOVANI, R. G. *Utilização de redes neurais de spikes para tarefas de navegação de agentes robóticos autonômos*. Dissertação (Mestrado) — Universidade Estadual de Londrina, 2011.

MATHWORKS. *Matlab - The Language of Technical Computing*. 2014. Disponível em: <http://www.mathworks.com/products/matlab/index.html>. Acesso em: 23/07/2013.

MATHWORKS2. *System Identification Toolbox*. 2014. Disponível em: <http://www.mathworks.com/products/sysid/description1.html>. Acesso em: 23/04/2014.

MAZO, M.; RODRÍGUEZ, F.; LÁZARO, J.; UREÑA, J.; GARCÍA, J. C.; SANTISO; REVENGA, P. Electronic control of a wheelchair guided by voice commands. *Control Engineering Practice*, Elsevier, v. 3, n. 5, p. 665–674, 1995.

MOREIRA, W. J. *Identificação Linear a Parâmetros Variantes no Tempo de Sistemas Não-Lineares*. Dissertação (Mestrado) — Instituto Militar de Engenharia, 2008.

MORTENSEN, T. F. *Lego History Timeline*. 2013. Disponível em: <http://aboutus.lego.com/en-us/lego-group/the_lego_history>. Acesso em: 02/02/2014.

MOURA SILVA, R. C. de. Técnicas de identificação e controle analógico e digital com labview e um kit dsp da texas instruments. *Trabalho de conclusão de curso apresentado na Universidade Estadual de Londrina*, 2012.

NATIONAL INSTRUMENTS . *Ambiente gráfico de desenvolvimento de sistemas Labview*. 2013. Disponível em: <http://www.ni.com/labview/pt/>. Acesso em: 20/12/13.

NISE, N. S. *Control Systems Engineering, 6th Edition*. [S.l.]: Wiley, 2010. 926 pages p.

O'DWYER, A. *Handbook of PI and PID controller tuning rules*. [S.l.]: Imperial College Press London, 2009.

OGATA, K. *Engenharia de controle moderno*. [S.l.]: Pearson Prentice Hall, 2003.

OLIVEIRA, T. C. de. *Identificação Fuzzy Takagi-Sugeno e Projeto de Controle Adaptativo da Articulação do Joelho*. Dissertação (Mestrado) — Universidade Estadual de Londrina - Área de Concentração: Automação, Agosto 2013.

PARKS, P. C. A new proof of the routh-hurwitz stability criterion using the second method of lyapunov. In: CAMBRIDGE UNIV PRESS. *Proc. Cambridge Philos. Soc.* [S.l.], 1962. v. 58, n. 4, p. 694–702.

PENA, J. M. Characterizations and stable tests for the routh–hurwitz conditions and for total positivity. *Linear algebra and its applications*, Elsevier, v. 393, p. 319–332, 2004.

PESTANA, H. G. *DROIDE M.L.P - NXT Software Development Kit*. Dissertação (Mestrado) — Universidade da Madeira - Portugal, 2008.

PHILLIPS, C. L.; NAGLE, H. T. *Digital Control System Analysis and Design*. 3th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 1995. ISBN 0-13-309832-x.

PISANI, R. L. Controle de motor de passo empregado em fermentação semi-sólida para amostragem de gases utilizando as ferramentas computacionais labview. *Trabalho de conclusão de curso apresentado na Escola de Engenharia de São Carlos - Universidade de São Paulo*, p. 33, 2012.

RODRIGUES, G. G. *Identificação de Sistemas Dinâmicos Não-Lineares Utilizando Modelos NARMAX Polinomiais- Aplicação a Sistemas Reais*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, 1996.

ROSA, J. E. A.; JUNIOR, E. G.; COVACIC, M. R.; GAINO, R. Técnica de controle otimizado aplicada a uma cadeira de rodas controlada por sopro e sucção. In: *XI SBAI - Simpósio Brasileiro de Automação Inteligente*. [s.n.], 2013. Disponível em: <http://www.sbai2013.ufc.br/pdfs/5475.pdf>. Acesso em: 08/05/2014.

ROSSINI, F. L. *Projeto de Controlador Robusto Aplicado a Cadeiras de Rodas Móveis via Abordagem por LMIs*. Dissertação (Mestrado) — Universidade Estadual de Londrina - Programa de Pós-Graduação em Engenharia Elétrica, 2013.

SANTOS, N. D.; ROSSINI, F. L.; GAINO, R.; COVACIC, M. R. Controle robusto de veículo sobre plataforma com rodas e tração diferencialutilizando lmis. In: *XI-SBAI Simpósio Brasileiro de Automação Inteligente*. [s.n.], 2013. Disponível em: <http://www.sbai2013.ufc.br/pdfs/4737.pdf>. Acesso em: 08/05/2014.

SCHOLZ, M. P. *Advanced NXT: The Da Vinci Inventions Book (Technology in Action)*. Berkely, CA, USA: Apress, 2007. ISBN 1590598431.

SIEGWART, R. Mobile robots facing the real world. In: YUTA, S.; ASAMA, H.; PRASSLER, E.; TSUBOUCHI, T.; THRUN, S. (Ed.). *Field and Service Robotics*. Springer Berlin Heidelberg, 2006, (Springer Tracts in Advanced Robotics, v. 24). p. 21–30. ISBN 978-3-540-32801-8. Disponível em: <http://link.springer.com/chapter/10.1007%2F10991459_3#page-1>. Acesso em: 21/08/2013.

SILVA, A. S. E. *Fundamentos de Controle Clássico (Apostila)*. 2012. Departamento de engenharia elétrica - EEL- CTC - UFSC. Disponível em: <http://www.labspot.ufsc.br/~aguinald/ensino/eel7063/controle.pdf>. Acesso em: 04/04/2013.

SIQUEIRA, M. D.; CONFORTO, J.; VALLIM, M. B. An FPGA-based robotics platform for educational and research applications. In: IEEE. *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*. [S.l.], 2012. p. 313–318.

SJÖBERG, J.; ZHANG, Q.; LJUNG, L.; BENVENISTE, A.; DELYON, B.; GLORENNEC, P.-Y.; HJALMARSSON, H.; JUDITSKY, A. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, Elsevier, v. 31, n. 12, p. 1691–1724, 1995.

SODERSTROM, T.; STOICA, P. *System Identification*. [S.l.]: Prentice Hall, 1989. (Prentice Hall International Series In Systems And Control Engineering). ISBN 9780138812362.

SORENSON, H. W. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, v. 7, p. 63–68, July 1970.

SOUSA, L. F. de. *Lego Mindstorms NXT 2.0*. 2013. Disponível em: <http://blog.computero.com.br/lego-mindstorms-nxt-2-0/>. Acesso em: 01/09/2013.

TEIXEIRA, M. C.; ASSUNÇÃO, E.; COVACIC, M. R. Proportional controllers: direct method for stability analysis and Matlab implementation. *Education, IEEE Transactions on*, IEEE, v. 50, n. 1, p. 74–78, 2007.

WEINBERG, J. B.; YU, X. Robotics in education: Low-cost platforms for teaching integrated systems. *Robotics & Automation Magazine, IEEE*, IEEE, v. 10, n. 2, p. 4–6, 2003.

YANG, H.; BORENSTEIN, J.; WEHE, D. Sonar-based obstacle avoidance for a large, non-point, omni-directional mobile robot. In: *Omni-directional Mobile Robot, Spectrum 2000 International Conference on Nuclear and Hazardous Waste Management*. [S.l.: s.n.], 2000. p. 24–28.

ZHANG, Y.; SHIEH, L.; DUNN, A. Pid controller design for disturbed multivariable systems. In: IET. *Control Theory and Applications, IEE Proceedings-*. [S.l.], 2004. v. 151, n. 5, p. 567–576.

Apêndice A – Execução do Algoritmo para Encontrar Faixa de Estabilidade

```
Type the coefficients of N(s): [0.5989 14.97]
Type the coefficients of D(s): [1 0.001097]

Type "1" if you want to specify a decay rate or
"0" if you want to see the stability range: 0

Choose the type of the controller:
 1 - Proportional (P);
 2 - Integral (I);
 3 - Derivative (D);
 4 - Proportional-Integral (PI);
 5 - Proportional-Derivative (PD);
 6 - Integral-Derivative (ID);
 7 - Proportional-Integral-Derivative (PID);
 0 - Cancel.
Option: 7

Default tolerance is 10^(-6). Is this value suitable for you (Y/N)? y

Type the initial value of kd: 5
Type the final value of kd: 10
Type the step of kd: 1
Type the initial value of ki: 5
Type the final value of ki: 10
Type the step of ki: 1

For kd = 5  and  ki = 5:  Solution: kp > -0.0027063
```

```
For kd = 5  and  ki = 6:  Solution: kp > -0.0032319

For kd = 5  and  ki = 7:  Solution: kp > -0.0037572

For kd = 5  and  ki = 8:  Solution: kp > -0.0042822

For kd = 5  and  ki = 9:  Solution: kp > -0.0048068

For kd = 5  and  ki = 10:  Solution: kp > -0.0053311

For kd = 6  and  ki = 5:  Solution: kp > -0.0022728

For kd = 6  and  ki = 6:  Solution: kp > -0.0027121

For kd = 6  and  ki = 7:  Solution: kp > -0.0031511

For kd = 6  and  ki = 8:  Solution: kp > -0.0035898

For kd = 6  and  ki = 9:  Solution: kp > -0.0040284

For kd = 6  and  ki = 10:  Solution: kp > -0.0044667

For kd = 7  and  ki = 5:  Solution: kp > -0.0019619

For kd = 7  and  ki = 6:  Solution: kp > -0.0023392

For kd = 7  and  ki = 7:  Solution: kp > -0.0027162

For kd = 7  and  ki = 8:  Solution: kp > -0.0030931

For kd = 7  and  ki = 9:  Solution: kp > -0.0034698

For kd = 7  and  ki = 10:  Solution: kp > -0.0038464

For kd = 8  and  ki = 5:  Solution: kp > -0.001728

For kd = 8  and  ki = 6:  Solution: kp > -0.0020586
```

```
For kd = 8  and  ki = 7:  Solution: kp > -0.002389

For kd = 8  and  ki = 8:  Solution: kp > -0.0027193

For kd = 8  and  ki = 9:  Solution: kp > -0.0030495

For kd = 8  and  ki = 10:  Solution: kp > -0.0033795

For kd = 9  and  ki = 5:  Solution: kp > -0.0015457

For kd = 9  and  ki = 6:  Solution: kp > -0.0018399

For kd = 9  and  ki = 7:  Solution: kp > -0.0021339

For kd = 9  and  ki = 8:  Solution: kp > -0.0024279

For kd = 9  and  ki = 9:  Solution: kp > -0.0027218

For kd = 9  and  ki = 10:  Solution: kp > -0.0030155

For kd = 10  and  ki = 5:  Solution: kp > -0.0013995

For kd = 10  and  ki = 6:  Solution: kp > -0.0016645

For kd = 10  and  ki = 7:  Solution: kp > -0.0019295

For kd = 10  and  ki = 8:  Solution: kp > -0.0021943

For kd = 10  and  ki = 9:  Solution: kp > -0.002459

For kd = 10  and  ki = 10:  Solution: kp > -0.0027237

The bounds  of the intervals,  for each value of  kd and ki,  are
stored in the matrix  "sol3v", where each  row corresponds to one
interval.
The first and the second columns contain the values of kd and ki,
respectively. The third and  the fourth columns contain the lower
and the upper bounds of the intervals of kp, respectively.
```

Apêndice B – Algoritmo do controle da distância para o Robô *Lego Mindstorm NXT*

```
#define KP 0 // Ganho proporcional
#define KD 0 // Ganho derivativo
#define KI 0 // Ganho integral
#define ERROR_TOL 50        // define limite de erro

#define MOTORS OUT_BC       // conexão com o motor
#define SENSOR IN_4         // conexão com o sensor


int dist;                   // variável para distância

inline void beep() {
  PlayTone(440,100); Wait(1000);
}

void WaitPressM() {
   beep();
  // controla o pressionamento dos botões do robô
  while(ButtonPressed(BTNLEFT,   false) ||
       ButtonPressed(BTNCENTER, false) ||
        ButtonPressed(BTNRIGHT,  false));
               // Espera o botão central do robô ser pressionado
    while (!ButtonPressed(BTNCENTER, false)) ;
}
// funções de avanço ou retroação do robô
inline int max(int a, int b) {return (a > b) ? a : b;}
inline int min(int a, int b) {return (a < b) ? a : b;}
```

```
inline int clip_to_100(int a) {return min(max(a, -100), 100); }

task pid_control() // começa controle pid
{
  int  y, e, e_old, diff; // variáveis para controle do erro da distância
  int u;
  long sum = 0;

  while (true) { // controla o limite da distância
    y = SensorUS(SENSOR);
    e_old = e;
    e     = y - dist;  // distância atual
    if (e > ERROR_TOL) {
      TextOut(0, LCD_LINE1, "Distância perdida" );
      break;
    }
    sum  = e + 0.95 * sum;  // soma do erro
    diff = e - e_old;   // diferença do erro
    //cálculo do controlador PID
    u = clip_to_100(KP * e + KI * sum + KD * diff);
    OnFwd(MOTORS, u);  // comando para potência do motor
    TextOut(0,  LCD_LINE4, "Y =         " ); // exibe distância atual
    TextOut(10, LCD_LINE4, NumToStr(y));
    TextOut(0,  LCD_LINE5, "E =         " ); // exibe o erro atual
    TextOut(10, LCD_LINE5, NumToStr(e));
    TextOut(0,  LCD_LINE6, "U =         " ); // exibe a potência aplicada
    TextOut(10, LCD_LINE6, NumToStr(u));
  }
}
task main() {
  SetSensorLowspeed(SENSOR);
  if (!false) { // PID control
    TextOut(0, LCD_LINE1, "Medir a distância" );
    WaitPressM();  // espera do pressionamento do botão
    dist = SensorUS(SENSOR);  // Faz a leitura inicial da distância
    ClearScreen();
// exibe distância na tela do robô
    TextOut(0, LCD_LINE3, StrCat("DISTÂNCIA = " , NumToStr(dist)));
```

```
    TextOut(0, LCD_LINE1, "INICIAR" );
    WaitPressM();
    TextOut(0, LCD_LINE1, "Executando" );
    start pid_control; Inicia a função para o cálculo do PID
   }
}
```

Apêndice C – Modelos obtidos na identificação

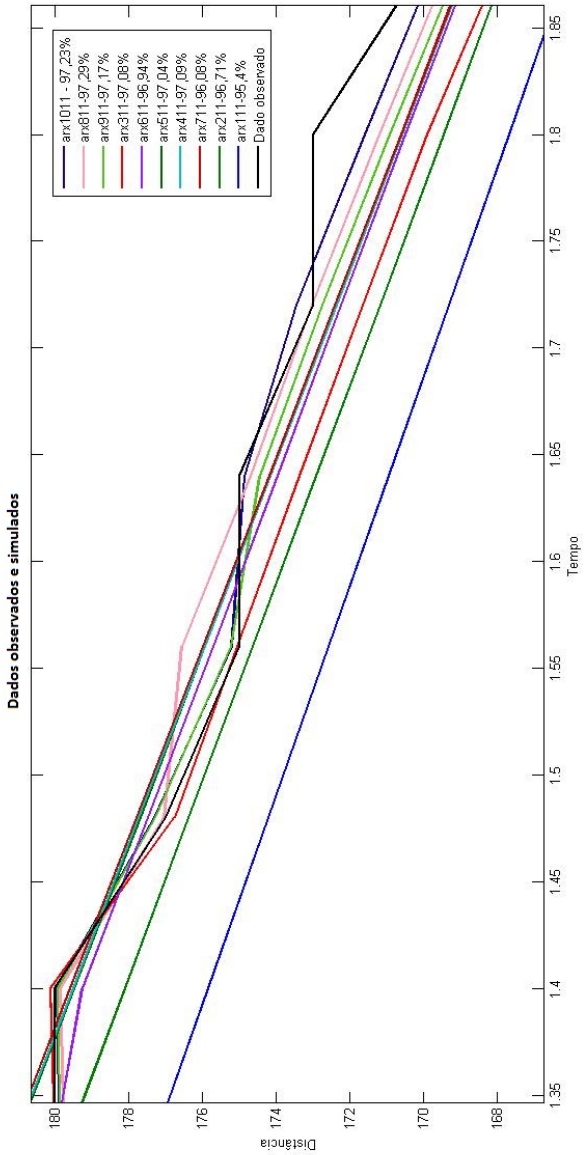


Figura C.1: Modelos obtidos através da *Toolbox*
Fonte: Elaborado pelo autor.