unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

RODOLFO PACHECO DA SILVA

Introdução à Identificação de Sistemas utilizando o System Identification Toolbox do MATLAB em conjunto com o Arduino para o Laboratório de Controle Linear

Guaratinguetá – SP
2015

Rodolfo Pacheco da Silva

Introdução à Identificação de Sistemas utilizando o System Identification Toolbox do MATLAB em conjunto com o Arduino para o Laboratório de Controle Linear

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador : Prof. Dr. Francisco A. Lotufo

Guaratinguetá – SP
2015

Silva, Rodolfo Pacheco da
S586i Introdução à identificação de sistemas utilizando o System Identification Toolbox do MATLAB em conjunto com o Arduino para o Laboratório de Controle Linear/ Rodolfo Pacheco da Silva – Guaratinguetá, 2015.
60 f : il.
Bibliografia: f. 49

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2015.
Orientador: Prof. Dr. Francisco Antonio Lotufo

1. Identificação de sistemas 2. MATBLAB (Programa de Computador) 3. Arduino (Controlador programável) I. Título

CDU 681.5.015

unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

Rodolfo Pacheco da Silva

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
"GRADUADO EM ENGENHARIA ELÉTRICA"

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
Prof. Dr. Leoanardo Mesquita
Coordenador

BANCA EXAMINADORA:

Prof. Dr. Francisco Antônio Lotufo
Orientador/UNESP-FEG

Prof. Dr. José Feliciano Adami
UNESP-FEG

Profª. Drª. Paloma Maria Silva Rocha Rizol
UNESP-FEG

Dezembro 2015

De modo especial, ao meu filho Miguel, aos Meus Pais Mário e Luziana e ao meu padrinho Lairson que foram grandes incentivadores para que eu continuasse e concluísse o curso de Engenharia Elétrica.

“Eduquem as crianças, para que não seja necessário punir os adultos.”

Pitágoras

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus pela minha vida, e por tudo que ele me proporcionou e me proporciona até hoje,

Agradeço aos meus pais Mário e Luziana que sempre me apoiaram em todos os momentos da minha vida e sempre estiveram do meu lado me dando conselhos e ajudando nos momentos felizes e difíceis, também minhas irmãs Bárbara e Laura por estarem sempre ao meu lado,

Ao meu filho Miguel, por me proporcionar momentos lindos, e por ajudar inconscientemente na minha formação acadêmica e pessoal,

Ao meu tio e padrinho Lairson Marques e ao meu avô Joaquim por sempre acreditarem no meu potencial, ajudando muito durante todo o período que estive em Guaratinguetá,

Ao meu orientador Prof. Francisco Antônio Lotufo que me auxiliou em todas as dificuldades que tive durante o desenvolvimento desse projeto, me incentivando, motivando e instruindo para levar o trabalho ao melhor resultado possível,

À Republica Jurupinga onde morei durante a minha graduação e fiz grandes amigos.

DA SILVA, R. P. **Introdução à Identificação de Sistemas utilizando o System Identification Toolbox do MATLAB em conjunto com o Arduino para o Laboratório de Controle Linear.** 2015. 61 f. Trabalho de Graduação (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2010.

Resumo

A educação pensada e planejada de forma clara e objetiva é de suma importância para que as universidades possam preparar profissionais competentes para o mercado de trabalho, e, sobretudo, possam atender a população com um trabalho eficiente. Especificamente, com relação à engenharia, a realização de aulas nos laboratórios é muito importante para a aplicação da teoria e desenvolvimento da parte prática do estudante.

O planejamento e preparação dos laboratórios, assim como equipamentos e atividades laboratoriais devem ser elaboradas de forma sucinta e clara, mostrando aos alunos como aplicar na prática o que foi aprendido na teoria e muitas vezes mostrar o porquê e onde poderá ser utilizado, quando eles se tornarem engenheiros.

Este trabalho utiliza a ferramenta MATLAB em conjunto com o *System Identification Toolbox* e o *Arduino* para a identificação de sistemas lineares para o laboratório de Controle Linear. O MATLAB é um programa muito utilizado na área da engenharia para cálculo numérico, processamento de sinais, construção de gráficos, identificação de sistemas, entre outras funções. Desse modo a introdução ao MATLAB e consequentemente à identificação de sistemas utilizando o *System Identification Toolbox* torna-se relevante na formação dos estudantes para que posteriormente quando for necessário realizar a identificação de um sistema a base e o conceito já tenham sido visto. Para esse procedimento a plataforma de código aberto *Arduino* foi utilizada como uma placa de aquisição de dados sendo a mesma também introduzida ao aluno, lhe oferecendo uma gama de *softwares* e *hardwares* para o aprendizado, dando-lhe cada dia mais bagagem para sua formação.

PALAVRAS-CHAVE: Identificação de Sistema, *System Identification Toolbox*, MATLAB, *Arduino*, Controle Linear.

DA SILVA, R. P. **Introdução à Identificação de Sistemas utilizando o System Identification Toolbox do MATLAB em conjunto com o Arduino para o Laboratório de Controle Linear.** 2015. 61 p. Graduate Work (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2010.

ABSTRACT

The education designed and planned in a clear and objective manner is of paramount importance for universities to prepare competent professionals for the labor market, and above all can serve the population with an efficient work. Specifically, in relation to engineering, conducting classes in the laboratories it is very important for the application of theory and development of the practical part of the student.

The planning and preparation of laboratories, as well as laboratory equipment and activities should be developed in a succinct and clear way, showing to students how to apply in practice what has been learned in theory and often shows them why and where it can be used when they become engineers.

This work uses the MATLAB together with the *System Identification Toolbox* and *Arduino* for the identification of linear systems in Linear Control Lab. MATLAB is a widely used program in the engineering area for numerical computation, signal processing, graphing, system identification, among other functions. Thus the introduction to MATLAB and consequently the identification of systems using the System Identification Toolbox becomes relevant in the formation of students to thereafter when necessary to identify a system the base and the concept has been seen. For this procedure the open source platform *Arduino* was used as a data acquisition board being the same also introduced to the student, offering them a range of software and hardware for learning, giving you every day more luggage to their training.

KEYWORDS: Identification System, *System Identification Toolbox*, MATLAB, *Arduino*, Linear Control.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

MATLAB	*Matrix Laboratory*
PWM	*Pulse With Modulation*
ARX	*Autoregressive model with exogenous inputs*
ARMAX	*Autoregressive moving average model with exogenous inputs*
OE	*Output error*
BJ	*Box-Jenkins*

SUMÁRIO

1 INTRODUÇÃO

Representar, através de modelos matemáticos, sistemas e fenômenos observados sempre foi um desafio. Desde a antiguidade, o homem tem procurado descrever matematicamente sistemas reais para ajudá-lo a entendê-los e, assim, resolver problemas relacionados a eles (AGUIRRE, 2000).

A identificação de sistemas é uma área que estuda maneiras de analisar e modelar sistemas a partir dos dados de entrada e saída obtidos do mesmo. Atualmente a identificação de sistemas é desejável para quase todas as áreas de conhecimento, tornando ela muitas vezes necessária para determinados sistemas dinâmicos, já que a coleta de dados hoje em dia é realizada constantemente em todos os setores.

A aquisição de dados é utilizada em grande escala em todo e qualquer processo realizado na engenharia e por mais simples que seja esta aquisição, é necessária a presença de um sensor, ou seja, um leitor de tensão, corrente ou qualquer a grandeza a ser aferida, uma plataforma para receber os dados, sejam eles analógicos ou digitais, e um computador ou interface para analisá-los. Com esses dados em mãos é então realizada a identificação do sistema em questão.

1.1 MOTIVAÇÃO

Assim como a formação de todos profissionais, a formação de um engenheiro deve, com certeza, ser a melhor possível, pois o mesmo poderá atuar em diversas áreas, podendo afetar a população a sua volta e até a ele mesmo, de forma boa e construtiva ou de forma péssima e destrutiva dependendo da sua personalidade e da sua formação profissional, devido esses fatores a graduação deste profissional deve prepará-lo para que ele faça o melhor enquanto engenheiro. A estruturação, elaboração e realização de laboratórios durante este processo tornam-se muito importante, pois ajudará a desenvolver habilidades correlacionadas com o ambiente de trabalho, preparando-o para o futuro.

Em muitos casos, os alunos se queixam de não conseguir visualizar ou até mesmo compreender como e por que utilizarão a parte teórica de uma determinada matéria na pratica e até mesmo no ambiente de trabalho, e a motivação deste projeto é ajudar nessa compreensão e visualização sobre a identificação de sistemas.

1.2 OBJETIVO

O projeto desenvolvido tem como finalidade elaborar uma aula laboratorial com ênfase na identificação de sistemas lineares para o Laboratório de Controle Linear ministrado pelo Prof. Francisco Antônio Lotufo, utilizando a técnica de aquisição de dados simples através da plataforma *Arduino* e de identificação de sistemas através do programa de alto desempenho voltado para calculo numérico, o MATLAB, em conjunto com sua ferramenta auxiliar *System Identification Toolbox*.

As atividades permitirão que o aluno adquira os dados de entrada e saída do sistema desejado, por meio do *Arduino* de forma simples, utilizando a programação em linguagem C++, enviando-os para o MATLAB, através de comandos escritos no próprio MATLAB, e após toda aquisição de dados o aluno poderá através do *Toolbox* estimar um modelo do sistema estudado e determinar a equação matemática que melhor representa tal sistema e assim melhorar a compreensão do mesmo.

2 CONCEITOS

2.1 IDENTIFICAÇÃO DE SISTEMAS

A identificação de sistemas é uma área do conhecimento que estuda maneiras de modelar e analisar sistemas a partir da observação (AGUIRRE 2000). É um termo utilizado para descrever as ferramentas matemáticas e que permitem construir modelos dinâmicos a partir de dados medidos.

Através dos dados obtidos é possível utilizar ferramentas matemáticas e algoritmos que permitem construir modelos que regem o processo, como por exemplo, o fluxo de água entre tanques de um determinado sistema, é possível determinar o modelo utilizando equações matemáticas e aplicando a lei de Bernoulli para que se possa desenvolver o modelo do processo. Infelizmente em poucas situações práticas haverá tempo e conhecimento suficientes para desenvolver um modelo a partir das equações e leis da física que controlam o processo (AGUIRRE 2000), sendo inviável desenvolver um modelo matemático baseado nas leis que regem o processo.

Para efetuar a identificação de um sistema real não conhecido, de forma satisfatória, é necessário seguir algumas etapas tais como: Aquisição de dados e pré-processamento, escolha da representação do sistema e validação do modelo.

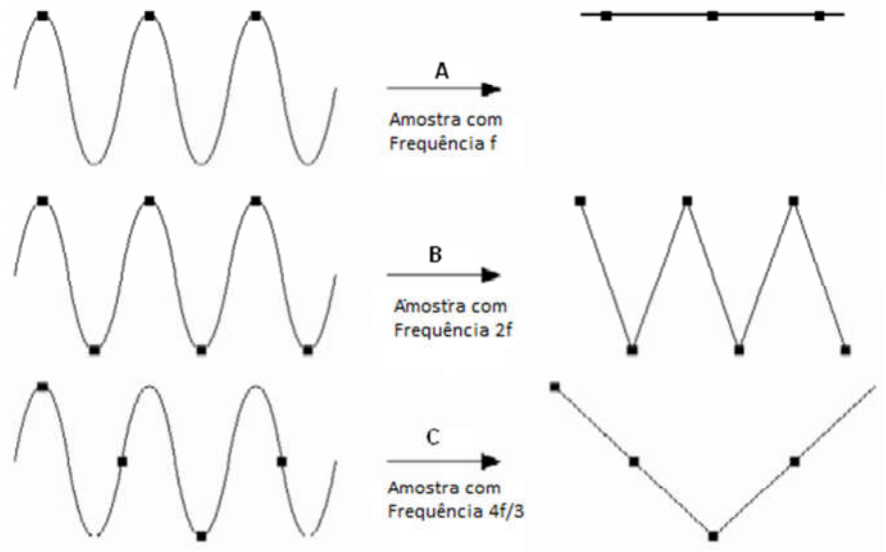
2.1.1 Aquisição de dados e pré-processamento

Uma vez que a identificação se propõe a obter modelo a partir de dados, é necessário coletar tais dados. Muitas vezes os únicos dados disponíveis serão dados de “operação normal”, mas em alguns casos será possível e desejável efetuar testes de forma a extrair informações dinâmicas do sistema (AGUIRRE 2000). Deve-se assegurar de que os dados obtidos estejam representando o processo da melhor maneira possível, para que a identificação não ocorra de forma errônea.

Havendo uma estimativa para a frequência do processo, pode-se dimensionar uma frequência de amostragem alta para garantir o cumprimento do critério de Nyquist, onde a frequência de amostragem deve ser no mínimo duas vezes maior do que a frequência máxima de interesse do sinal medido. A figura 1 mostra uma pequena e breve relação entre o sinal a ser medido e algumas frequências de amostragem, a fim de ratificar o critério de Nyquist, sendo essa etapa muito importante para que se tenha dados corretos e consequentemente um modelo satisfatório.

O pré-processamento dos dados refere-se a um conjunto de operações que podem ser executadas sobre os dados adquiridos, de forma a melhorar os resultados obtidos com a aplicação dos algoritmos para extração do modelo do sistema. Estas operações incluem a remoção da média do sinal, filtragem de componentes ruidosas do espectro, entre outras.

Figura 1 – Efeitos do uso diferente de taxas de amostragem



Fonte: (National Instruments)

2.1.2 Escolha da representação do sistema

Há diversas maneiras de representar o mesmo modelo matemático, ou seja, há varias formas em que as equações que descrevem o comportamento do sistema podem ser escritas (AGUIRRE 2000). Neste capitulo são descritas brevemente algumas representações para sistemas lineares, sendo elas as mais comuns, e deixando de lado também as técnicas e representações de sistemas não-lineares, pois é inviável descrever todas as representações existentes.

2.1.2.1 Funções de transferência

As funções de transferência são funções que modelam o comportamento dinâmico de um par entrada-saída de um sistema, ou seja, descrevem como uma determinada entrada é dinamicamente “transferida” para a saída do sistema (AGUIRRE 2000).

Utilizando a ferramenta matemática da transformada de Laplace pode-se fazer uma estimativa da função de transferência de um sistema, dividindo a transformada de Laplace da saída pela transformada de Laplace da entrada. Deste modo, as funções de transferência são normalmente representadas por dois polinômios em s. A função de transferência *G(s)* da transformada de Laplace da entrada *U(s)* e saída *Y(s)* é dada por:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{N(s)}{D(s)} = \frac{b_0 + b_1 s^1 + b_2 s^2 + \cdots + b_q s^q}{a_0 + a_1 s^1 + a_2 s^2 + \cdots + a_n s^n} \tag{2.1}$$

Onde *N(s)* e *D(s)* são polinômios em s e n > q. Assim definem-se três importantes parâmetros da função de transferência, sendo eles o número de polos de *G(s)*, número de zeros *G(s)* e a ordem do sistema.

- Polos de G(s): Raízes de D(s)
- Zeros de G(s): Raízes de N(s)
- Ordem do Sistema: Grau de D(s)

2.1.2.2 Representações Discretas

Há também modelos amplamente utilizados na identificação de sistemas que utilizam a função de transferência discreta. Esses modelos utilizam a transformada-z para modelagem sendo todos eles regidos praticamente por uma equação de modelo geral dada por:

$$A(z)y(k) = \frac{B(z)}{F(z)}u(k) + \frac{C(z)}{D(z)}e(k) \tag{2.2}$$

Sendo *y(k), u(k)* e *e(k)* saída, entrada e ruído branco respectivamente, e *A(z), B(z), C(z), D(z), F(z)* os polinômios definidos a seguir:

$$A(z) = 1 + a_1 z^{-1} + \ a_2 z^{-2} \ + \cdots + a_{n_y} z^{-n_y};$$
$$B(z) = b_1 z^{-1} + \ b_2 z^{-2} \ + \cdots + b_{n_u} z^{-n_u};$$
$$C(z) = 1 + c_1 z^{-1} + \ c_2 z^{-2} \ + \cdots + a_{n_\xi} z^{-n_\xi};$$
$$D(z) = 1 + d_1 z^{-1} + \ d_2 z^{-2} \ + \cdots + a_{n_d} z^{-n_d};$$
$$F(z) = 1 + f_1 z^{-1} + \ f_2 z^{-2} \ + \cdots + f_{n_f} z^{-n_f};$$

2.1.2.2.1 Modelo Box-Jenkins

O *Box-Jenkins* é o modelo mais genérico possível, onde se consideram funções de transferência independentes para o ruído, e(k), e a entrada u(k). O modelo pode ser obtido através do modelo geral (2.2) tomando-se A(z) = 1 e os demais polinômios arbitrários, resultando em:

$$y(k) = \frac{B(z)}{F(z)}u(k) + \frac{C(z)}{D(z)}e(k) \tag{2.3}$$

ou alternativamente,

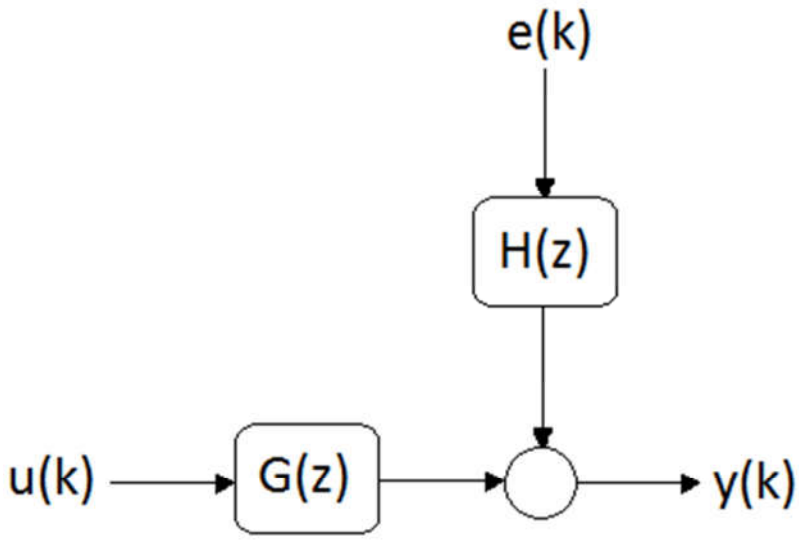
$$y(k) = G(z)u(k) + H(z)e(k) \tag{2.4}$$

Sendo:

$$G(z) = \frac{B(z)}{F(z)} \quad e \quad H(z) = \frac{C(z)}{D(z)} \tag{2.5}$$

A figura 2 apresenta o diagrama de bloco do modelo BJ, que é útil quando há distúrbios que entram no final do processo, como por exemplo, um ruído de medição na saída é uma perturbação no final do processo.

Figura 2 – Modelo *Box-Jenkins*



Fonte: National Instruments

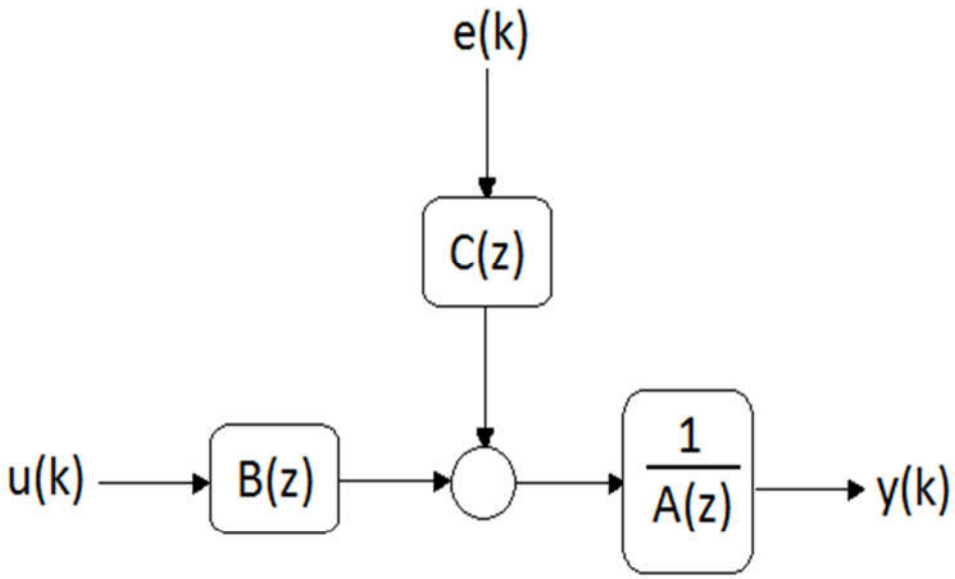
2.1.2.2.2 Modelo ARMAX

O modelo auto regressivo com média móvel e entradas exógenas (ARMAX) pode ser obtido a partir do modelo geral (2.2), tomando-se *D(z) = F(z) = 1* e *A(z), B(z), C(z)* polinômios arbitrários, resultando em:

$$y(k) = \frac{B(z)}{A(z)}u(k) + \frac{C(z)}{A(z)}e(k) \tag{2.6}$$

Os modelos ARMAX são uma extensão dos modelos Box-Jenkins, pois consideram a possibilidade de inclusão de variáveis exógenas como regressores. São formados por uma parte auto regressiva AR, definida como uma regressão linear do valor atual da serie contra valores passados ponderados, uma parte médias móveis MA, uma soma ponderada da observação atual e de observações do passado de um processo ruído branco. Os Modelos ARMAX, como mostra a figura 3, são uteis quando se tem o domínio de distúrbios que entram no inicio do processo, como por exemplo, uma rajada de vento que afeta uma aeronave é um distúrbio dominante no inicio do processo (National Instruments).

Figura 3 – Modelo ARMAX



Fonte: National Instruments

2.1.2.2.3 Modelo ARX

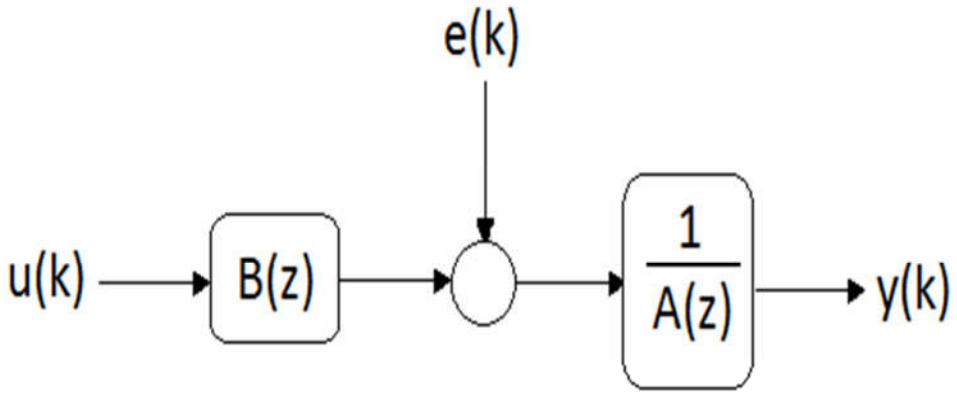
O modelo auto-regressivo com entradas externas pode ser obtido também a partir do modelo geral (2.2), adotando-se *C(z), D(z), F(z)* = 1 e *A(z)* e *B(z)* polinômios arbitrários resultando em:

$$y(k) = \frac{B(z)}{A(z)}u(k) + \frac{1}{A(z)}e(k) \tag{2.7}$$

O modelo ARX, mostrado na figura 4, é o modelo mais simples incorporando o sinal de estímulo. A estimativa do modelo ARX é o mais eficiente dos métodos de estimação polinomiais porque é o resultado de resolução de equações de regressão linear em forma analítica. Além disso, a solução é única. Em outras palavras, a solução sempre satisfaz o mínimo global de perda de função. Por conseguinte, o modelo ARX é preferível, especialmente quando a ordem do modelo é alta (National Instrument).

O ARX é uma simplificação do ARMAX considerando que o efeito do polinómio associado ao ruído não é importante, ou seja, temos ruído aleatório aplicado diretamente à dinâmica do sistema, tornando-o deste modo, assim como ARMAX, um modelo pertencente à classe de modelos de erro na equação.

Figura 4 – Modelo ARX



Fonte: National Instruments

2.1.2.2.4 Modelo *Output Error*

O nome em inglês *Output Error,* que significa erro na saída, justifica o nome desta classe de modelo. Sendo que o ruído branco, *e(k)*, pode ser, por exemplo, apenas um ruído na medição da saída. Este modelo assim como todos os anteriores também pode ser obtido a partir do modelo geral (2.2), tomando-se *A(z), C(z), D(z)* = 1 e *B(z)* e *F(z)* polinômios arbitrários resultando em:

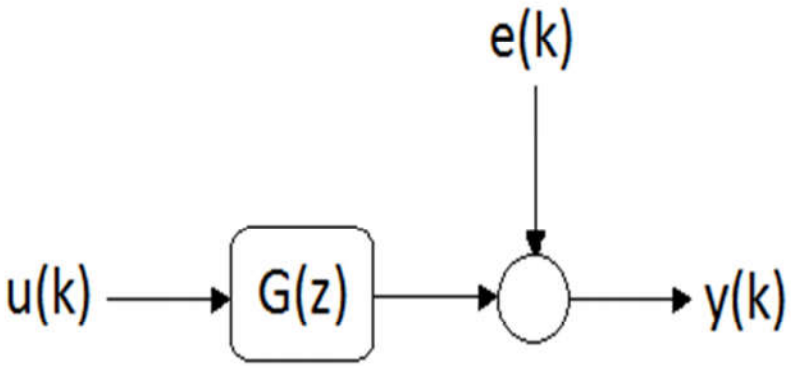
$$y(k) = \frac{B(z)}{F(z)}u(k) + e(k) \tag{2.8}$$

Ou alternativamente utilizando a igualdade da equação (2.5),

$$y(k) = G(z)u(k) + e(k) \tag{2.9}$$

Pode-se notar que o modelo OE é mais simples que o ARX, e com certeza um dos modelos mais descomplicado entre todos aqui apresentados, como mostra a figura 5, com o ruído aleatório aplicado diretamente na saída do sistema.

Figura 5 – Modelo Output Error



Fonte: National Instruments

2.1.2 Validação do Modelo

Tendo obtido uma família de modelos, é necessário verificar se eles incorporam ou não as características de interesse do sistema original (AGUIRRE 2000), ou seja, é preciso verificar se o modelo realmente descreve o que as entradas e as saídas fornecem.

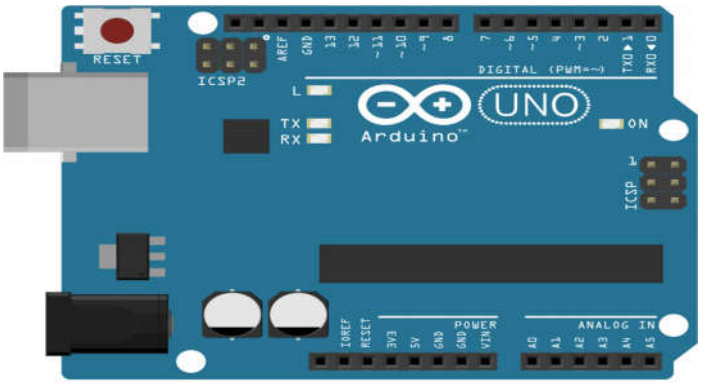
Além disso, é interessante poder comparar os modelos entre si e decidir se há algum significativamente melhor que os demais, sendo esta etapa certamente muito subjetiva, pois o resultado da validação dependerá de diversas variáveis. Como por exemplo, a aplicação pretendida para o modelo, já que nenhum modelo, por definição, irá representar o sistema real em todos os aspectos, o sistema será considerado valido se incorporar aquelas características do sistema que são fundamentais para a aplicação em questão (AGUIRRE 2000). A forma mais utilizada para a validação do modelo é a simulação do mesmo com os dados previamente obtidos. Neste caso, deseja-se saber se o modelo reproduz ao longo do tempo os dados observados.

2.2 *ARDUINO*

Arduino é uma plataforma de prototipagem de código aberto baseado em *hardware* e *software* de fácil utilização. As placas *Arduino* são capazes de ler entradas - a luz em um sensor, um dedo em um botão, ou uma mensagem de *Twitter* - e transformá-lo em uma saída - a ativação de um motor, ligar um LED, publicar algo online. Para fazer isso você usa a linguagem de programação *Arduino* (com base na fiação), e do Software *Arduino* (IDE), com base em Processamento (Arduino.cc).

Atualmente o *Arduino* é utilizado em diversos projetos e processos de engenharia devido a sua fácil utilização e compreensão e pode ser utilizado de forma simples por iniciantes ou até mesmo de forma mais complexa por pessoas experientes. A figura 6 mostra o desenho de um *Arduino UNO*, um dos modelos de *Arduino* mais barato do mercado e muito utilizado devido ao seu preço acessível. Ele possui pinos de entradas e saídas digitais, entradas analógicas, saídas digitais através do PWM, porta USB para comunicação e alimentação, entrada de alimentação e entre outros diversos componentes e pinos.

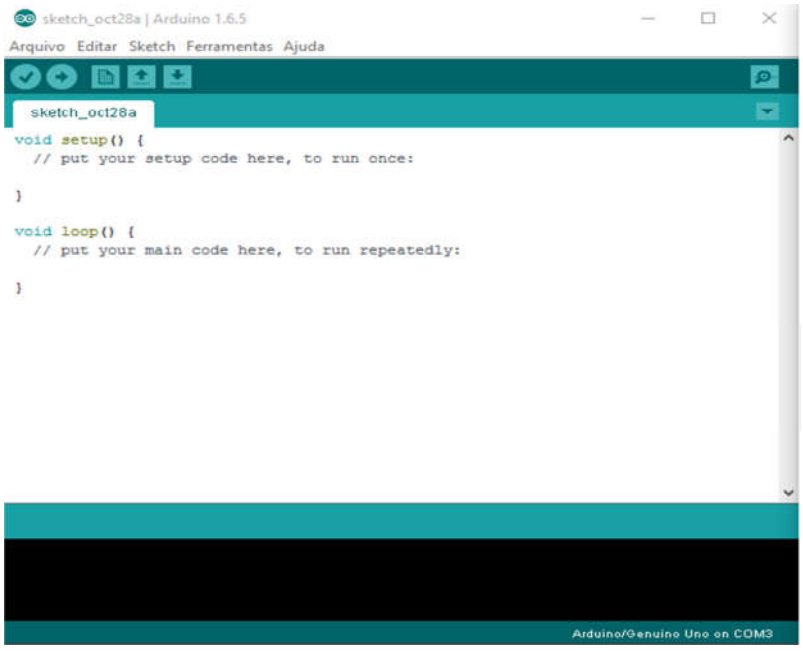
Figura 6 - *Arduino* UNO



Fonte: (Arduino.cc)

O *software* do *Arduino* (IDE) utiliza uma linguagem de programação muito conhecida que é a linguagem C++, e a sua interface de modo simples e compacta permite ao usuário uma fácil e rápida adaptação. A figura 7 mostra o *software* do *Arduino* ao ser inicializado, para a programação no *Arduino* existem duas funções sendo elas *void setup()* onde deve ser digitado o código de inicialização, que será lido pelo programa uma única vez e *void loop()* onde deve ser introduzido o código principal da programação que irá rodar repetitivamente.

Figura 7 – Interface do *Software Arduino* (IDE)



Fonte: Autoria própria

O programa pode ser rodado em todos os sistemas operacionais do mercado como Windows, Mac e Linux. Professores e alunos estão utilizando-o devido ao seu baixo custo para provar princípios físicos e químicos, ou até mesmo para iniciar a pratica de programação e robótica. Os projetistas e arquitetos constroem protótipos interativos, músicos e artistas usam-no para instalar e experimentar novos instrumentos. *Arduino* é uma ferramenta chave para aprender novas coisas (Arduino.cc).

2.3 MATLAB

O MATLAB foi criado na década de setenta por Cleve Moler, nesta época ele era presidente do departamento de ciência da computação da Universidade do Novo México, onde lecionava Analise Numérica e Teoria das Matrizes. Com o intuito de ajudar os alunos em realizar diversas operações matemáticas sem que eles precisassem programar na linguagem Fortran, ele criou então através da linguagem Fortran o MATLAB. Havia apenas 80 funções no inicio, e não havia código M ou *Toolboxes*, caso alguma função tivesse de ser adicionada seria necessário modificar o código fonte na linguagem Fortran e compilá-lo novamente. O comando *HELP* listava apenas as funções disponíveis com suas respectivas abreviações.

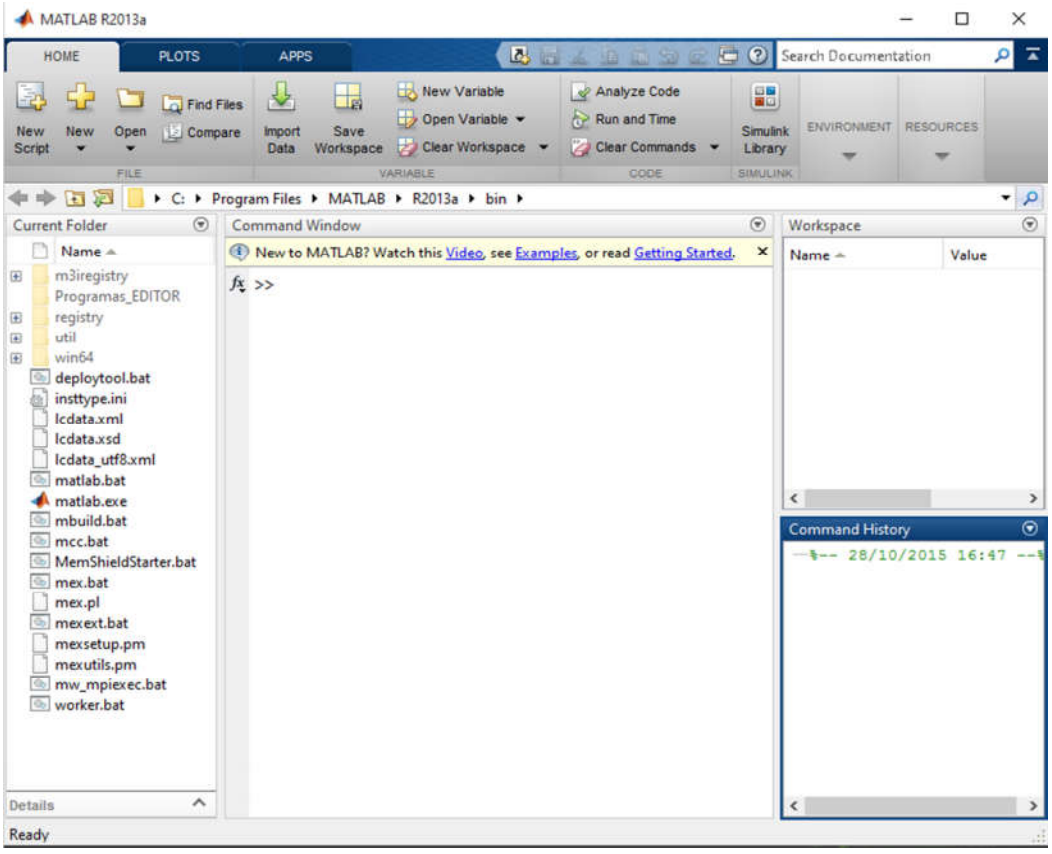
Moler visitou a Universidade de Stanford em 1979 para lecionar algumas aulas de Análise Numérica, neste curso alguns alunos utilizavam o MATALB para alguns exercícios e exemplos, porém por esses alunos serem metade da área de matemática e ciência da computação, eles não estavam muitos interessados no MATLAB, pois se tratava apenas de um programa escrito em Fortran, e não representava uma grande utilidade na área de analises numérica. Mas a outra metade era de engenheiros e eles adoraram o MATLAB, eles estudavam outras matérias como análise, controle e processamento de sinais e a ênfase em matrizes do MATLAB mostrou ser muito útil a eles.

Jack Little, um experiente engenheiro de controle, da Universidade de Stanford foi o principal desenvolvedor de um dos primeiros produtos comerciais baseados em MATLAB Fortran. Quando a IBM anunciou o seu primeiro computador pessoal em agosto de 1981 Jack rapidamente antecipou a possibilidade de utilizar o MATLAB e o computador para técnicas de computação. Ele e seu colega Steve Bangert reprogramaram o MATLAB na linguagem C e adicionaram o código M, Toolboxes e gráficos mais poderosos, que utilizamos até hoje no MATLAB.

Atualmente utiliza-se o MATLAB com o seu código M, uma linguagem matemática, tornando mais fácil o seu entendimento no decorrer da programação, e a sua interface é de forma simples como mostra a figura 8, permitindo que o usuário tenha acesso as informações principais de sua programação, tais como: *Command Window*, *Workspace* e *Command History*.

A *Command Window* é o local onde as operações são feitas diretamente mostrando os resultados após tal operação desejada, onde também são atribuídos valores a variáveis entre outros comandos, já o *Workspace* é o espaço destinado às variáveis que estão salvas na memória, onde é possível visualizar o nome, classe e valor da mesma e o *Command History* é destinado para operações passadas, onde são organizados por data de execução, permitindo o usuário checar o que já foi realizado e se necessário utilizar operações passadas com um duplo clique.

Figura 8 – Interface do MATLAB



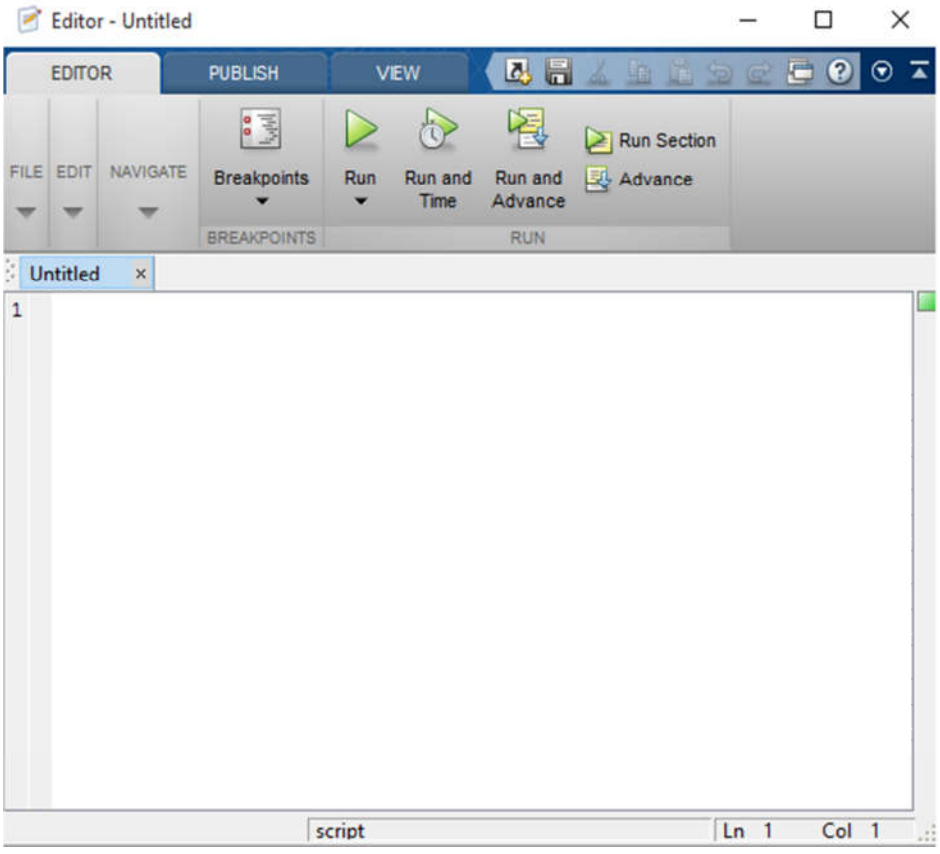
Fonte: Autoria própria.

Como quase todo programa que está disponível hoje em dia no mercado, o MATLAB possui vídeos e exemplos para iniciantes se familiarizarem melhor e mais rápido com o mesmo, além de possuir também uma pagina na internet para ajudar o usuário em mais tópicos caso necessário, que pode ser acessada através do link http://www.mathworks.com/help/matlab/.

Há também a possibilidade de criar arquivos .m, arquivos criados no MATLAB que são chamados de M-files, acessando a guia New>>Script o MATLAB abrirá uma nova janela com o nome "*Editor-Untitled*" como pode-se ver na figura 9, onde é possível criar procedimentos para criar uma comunicação entre o programa e o usuário e também comunicação com outras plataforma como neste caso com o *Arduino.*

O Procedimento gerado é salvo na extensão 'nomedoaqurivo.m' que pode então posteriormente ser acessado e compilado no MATLAB, mostrando na *Command Window* a função realizada sem mostrar todo o procedimento nela contido, mas na interface no campo do *Workspace* aparecerá valores das variáveis criadas no processos, caso houver variáveis.

Figura 9 – Tela inicial do *Script-Editor*



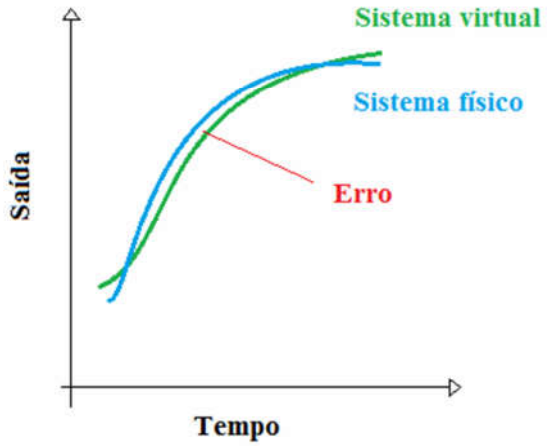
Fonte: Autoria própria.

2.3.1 *System Identification Toolbox*

Toolbox ™ Identificação de Sistema é uma ferramenta disponível no MATLAB para a construção de modelos matemáticos de sistemas dinâmicos a partir de dados de entrada e saída adquiridos de um sistema, criada por Lennart Ljung Professor Doutor da Universidade de Linköping na Suécia. A ferramenta permite criar e usar modelos de sistemas dinâmicos que não são facilmente modelados a partir de princípios físicos. Podem ser usados dados no domínio do tempo e no domínio da frequência de entrada-saída para identificar em tempo discreto modelos de processos na forma de função de transferência, espaço de estado, polinômios entre outros.

Esta ferramenta ajuda a obter de forma rápida e simples modelos de certo sistema, e assim permite comparar os resultados da simulação dos modelos com os resultados reais coletados e então verificar qual possui o menor erro em relação ao sistema real, como mostra a figura 10.

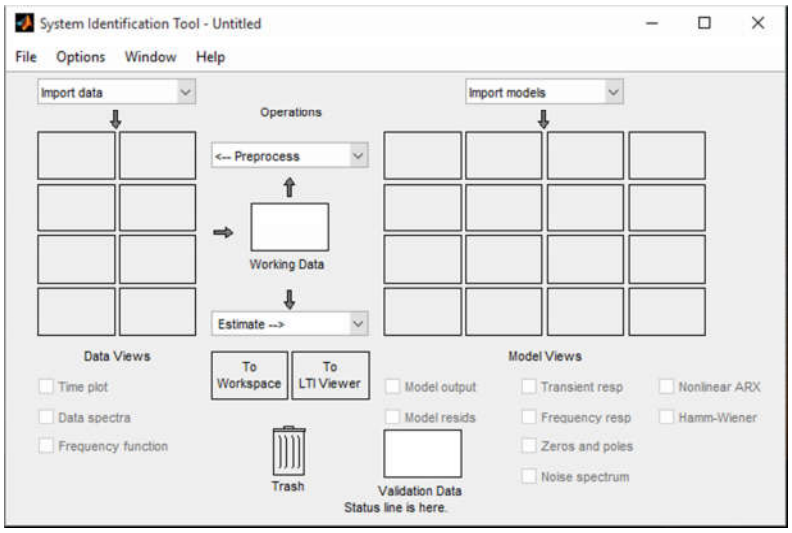
Figura 10 – Gráfico de comparação do Sistema físico com virtual



Fonte: Autoria própria.

Ao digitar *'ident'* na janela *Command Window* do MATLAB o *Toolbox* é aberto em uma janela à parte como mostra a figura 11, e então é possível visualizar previamente as funções principais e elementares da ferramenta, sendo elas: *Import data* e *Estimate.*

Figura 11 – Interface do *System Identification Toolbox*

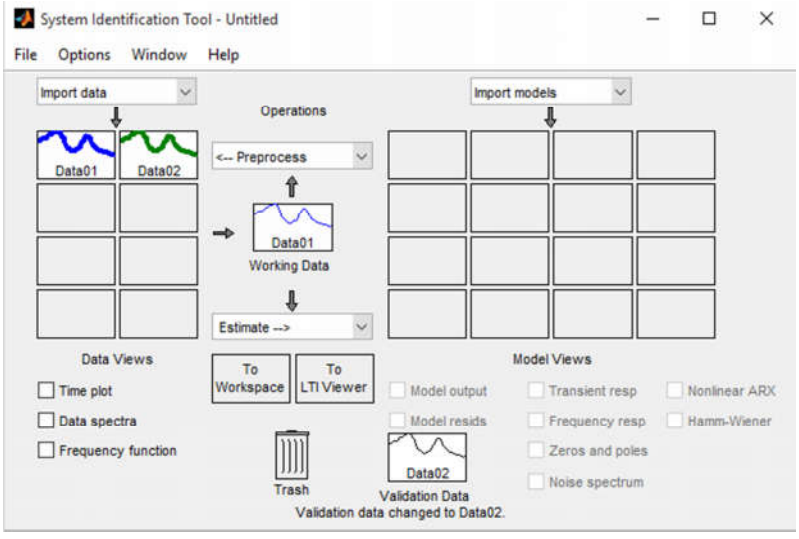


Fonte: Autoria própria.

A função *Import Data*, como o próprio nome diz, irá importar os dados de entrada e saída do sistema a ser modelado, sendo ela a primeira tarefa a ser executada ao utilizar a ferramenta, já que sem tais dados nada pode ser feito com a mesma. *Estimate* permite escolher entre diversos modelos tais como função transferência, ARX, ARMAX entre outros para estimar o modelo do sistema desejado.

Ao importar os dados do sistema real o mesmo é alocado em uma das caixas destinada à dados importados como mostra a figura 12.

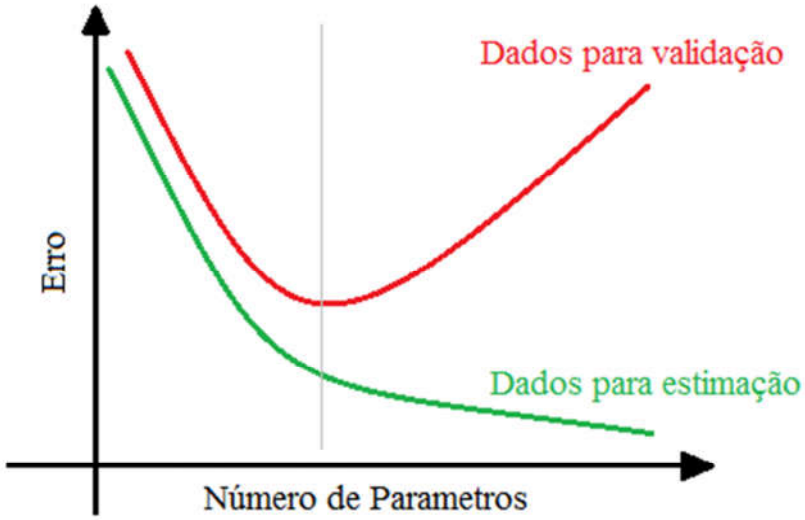
Figura 12 – Dados importados e alocados no Toolbox



Fonte: Autoria própria.

É muito importante obter duas aquisições de dados, pois uma será utilizada para estabelecer o modelo sendo alocada no quadro '*Working Data*' e a outra utilizada para a validação do mesmo sendo colocada no quadro '*Validation Data*'. Pois um modelo com muitos parâmetros pode ajustar bem qualquer conjunto de dados, sendo assim necessário verificar se o modelo consegue ajustar outro conjunto de dados, diferente dos que foram usados para a estimação, e claro ambos obtidos do mesmo processo em observação. Isto é feito para minimizar o erro, como mostra o gráfico da figura 13 o erro diminui com o aumento do número de parâmetros para dados de estimação, mas o mesmo não ocorre para os dados de validação, a partir de certo ponto o erro começa a aumentar, com relação ao aumento de parâmetros, chamado de *over fitting*. Assim os dados de validação devem ser coletados com um número determinado de parâmetros menor do que os dados de estimação, visando obter um erro menor.

Figura 13 – Gráfico de Erro em relação ao Número de Parâmetros.



Fonte: Autoria própria.

Com os dados preparados deve-se então selecionar um ou mais modelos a serem estimados e então verificar qual irá representar da melhor forma possível, dentro dos aspectos desejados, o modelo real.

3 DESENVOLVIMENTO

Nesse capitulo é apresentado de forma detalhada toda a parte de planejamento para a execução do laboratório e as atividades em suas diferentes etapas de desenvolvimento.

3.1 ELABORAÇÃO DOS EXPERIMENTOS

Neste projeto foram elaboradas duas atividades diferentes, ambas pensadas e projetadas de forma simples e básica com o intuito de serem apresentadas de um modo mais didático possível. As atividades possuem alguns objetivos em comum e algumas diferenças entre si, descritos nesse capítulo.

A primeira atividade foi elaborada com o objetivo de introduzir aos alunos novos programas e *hardwares* disponíveis atualmente no mercado, muito utilizados em universidades e no mercado de trabalho. Pois a princípio a plataforma *Arduino* e a ferramenta *System Identification Toolbox* do MATLAB nunca foram vistos e utilizados por alunos do terceiro ano de engenharia elétrica da UNESP de Guaratinguetá. Mesmo sendo um experimento considerado simples, o primeiro experimento é muito importante para o aprendizado, pois além de introduzir os conceitos básicos em relação ao *Arduino* e aquisição de dados através do mesmo, ele irá também iniciar o processo de introdução à identificação de sistemas, tema principal deste trabalho.

A segunda atividade foi preparada com a função de estimar novos modelos com dados de outro sistema, concretizar o aprendizado realizado anteriormente e mostrar ao aluno uma vasta opção de ações que a ferramenta disponibiliza. Esta atividade também apresenta ao aluno a importância da identificação de sistemas no mundo real de trabalho, trazendo dados de um sistema de aquecimento que a própria ferramenta disponibiliza, onde a potência de um secador de cabelo, ou um aquecedor é a entrada e temperatura do ar é saída do sistema, fazendo com que o aluno enxergue que processos mais complicados podem ser estimados.

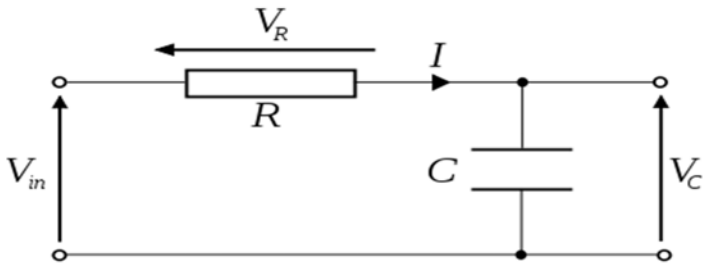
3.2 PRIMEIRA ATIVIDADE

De forma simples e didática a atividade utiliza como exemplo o experimento da descomplicada carga de um capacitor, um circuito simples e conhecido na engenharia elétrica,

o circuito RC. Este experimento visa mostrar a prática em conjunto com a teoria apresentando também ao aluno as facilidades e a importância da ferramenta utilizada.

O circuito utilizado é um circuito simples, porém de fácil implementação, e reúne as condições básicas para explicar a metodologia de identificação e modelagem de sistemas, que é um sistema de primeira ordem. O circuito de carga de um capacitor como mostra a figura 14, é composto por uma fonte de tensão, uma resistência e um capacitor. Neste caso a fonte de tensão é obtida através da placa do *Arduino* com o valor de 5 volts, os valores do resistor e capacitor utilizados foram 2,2k Ohms e 100µ Farads, respectivamente. Esses valores foram selecionados devido ao tempo de carga ($t_c \approx 5*RC$) de um capacitor em serie com um resistor, sendo assim possível realizar uma ótima aquisição de dados, tanto para a transmissão de dados entre *Arduino* e MATLAB através da porta serial, quanto para a visualização do gráfico de carga do capacitor, porém os mesmos podem ser alterados para que outros exemplos sejam modelados.

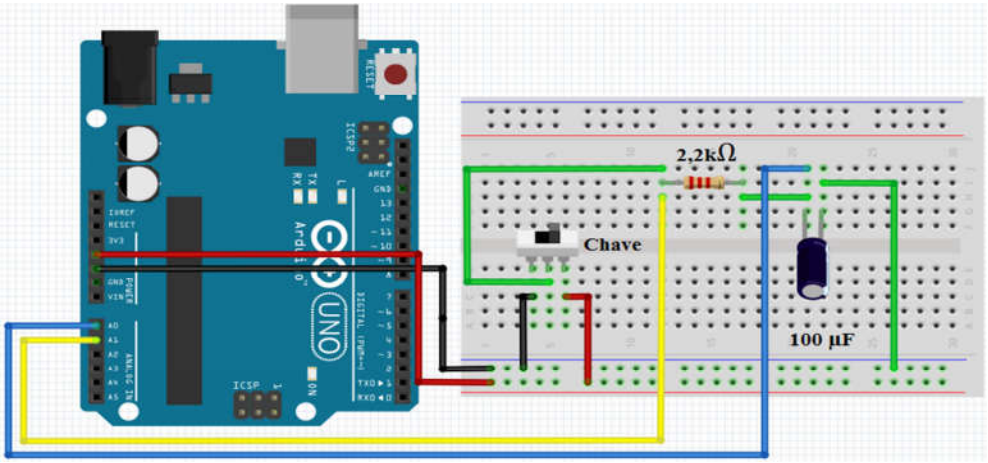
Figura 14 – Circuito RC



Fonte: Autoria própria.

Para iniciar o experimento a primeira tarefa a ser feita é montar o circuito em um *protoboard* com suas devidas conexões com o *Arduino*, tais como, a fonte para alimentar o circuito, a chave para realizar a modificação do valor de tensão de entrada entre 0 e 5 volts, e posteriormente os cabos que irão fazer a leitura do sinal entrada (Fonte de Tensão) e saída (Tensão do capacitor), sendo esses cabos de leitura conectados aos pinos analógicos do *Arduino* de acordo com a programação realizada no *software* IDE, como mostra a figura 15. O apêndice A contém um exemplo detalhado de programação que pode ser utilizado pelo aluno, podendo o mesmo também ser alterado em algumas partes de modo que a leitura e transmissão de dados ocorram corretamente.

Figura 15 – Esquema de ligação do circuito RC conectado ao *Arduino.*



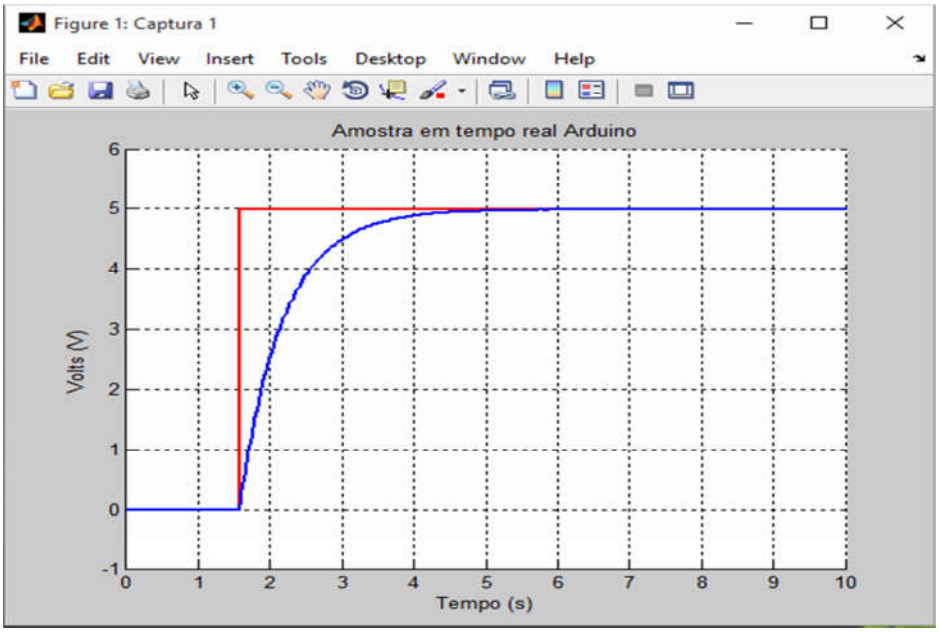
Fonte: Autoria própria.

Após realizar a montagem e a programação, antes de conectar o *Arduino* ao MATLAB, é possível visualizar na janela de comunicação serial do IDE os valores apresentados para verificar se a leitura das tensões está ocorrendo corretamente, podendo assim evitar já previamente erros de leitura. Com as leituras ocorrendo normalmente e corretamente é necessário fechar o software IDE para que a porta serial fique livre para comunicação com o MATLAB.

Ao iniciar o MATLAB é preciso criar um novo *Script* onde é digitado o programa que vai se comunicar com o *Arduino*, porém no apêndice B há um exemplo que pode ser utilizado pelo aluno, onde há comentários detalhando-o para melhor entendimento. Com o programa digitado ou copiado no *Editor* do MATLAB é possível então iniciar a aquisição dos dados entre o *Arduino* e MATLAB clicando no botão 

. Ao aparecer a janela do gráfico no computador o aluno deve então mudar o estado da chave para que a tensão de entrada seja alterada de 0V para 5V e o capacitor então comece a carregar. A princípio os primeiros dados adquiridos são posteriormente utilizados como dados de estimação dentro do *System Identification Toolbox*, então o aluno deve se atentar para a variável '*tmax*', tempo de aquisição, para que este tempo seja maior que o tempo de aquisição para os dados de validação. A figura 16 mostra uma aquisição de dados com a variável '*tmax*' igual a 10 segundos, sendo que neste exemplo os dados de validação foram coletados com '*tmax*' igual a 5 segundos. Os dados coletados são transformados em matrizes no MATLAB, sendo 'v1' a matriz para a tensão de entrada, e 'v2' a matriz para a tensão de saída.

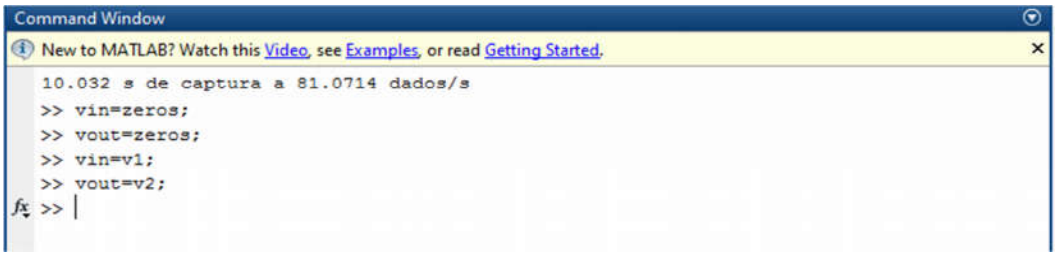
Figura 16 – Exemplo de Aquisição de Dados para “Dados de Estimação”



Fonte: Autoria própria.

Para realizar a próxima aquisição de dados o aluno deve salvar os dados já obtidos e guardados nas matrizes v1 e v2 em outras novas matrizes, pois para tornar a segunda aquisição fácil e rápida como a primeira, é utilizado o mesmo programa, alterando apenas o valor de *tmax,* onde consequentemente são utilizadas as mesmas variáveis de matrizes v1 e v2 para armazenar os próximos dados. Portanto é preciso utilizar a função ‘zeros’, onde a mesma retorna uma matriz com todos os valores em zero. Duas novas matrizes são criadas através de comandos na janela ‘*Command Window’*, como mostra a Figura 17, onde as mesmas recebem os valores de v1 e v2, não esquecendo que neste caso v1 são os dados de entrada (tensão da fonte) e v2 são os dados de saídas (tensão do capacitor). É importante também observar a informação apresentada na primeira linha após a leitura dos dados, onde a mesma informa o tempo de leitura, e a quantidade de dados lidos por segundo, esse número é importante para definir o intervalo de amostragem (*Sampling Interval*) posteriormente.

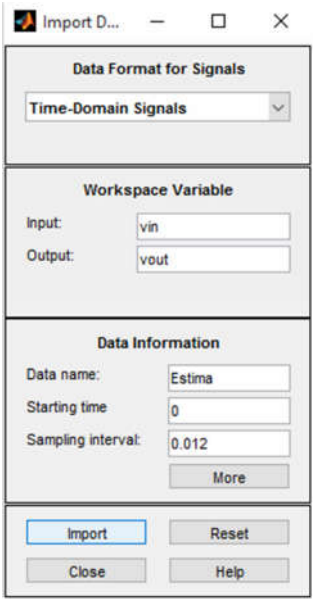
Figura 17 – Exemplo de realocação das matrizes



Fonte: Autoria própria.

Após realizar a segunda aquisição de dados, sendo eles dados de validação, deve-se então abrir o *Toolbox* e iniciar o procedimento de importação de dados de estimação e validação, como mostra a figura 18, a importação de dados de estimação, nomeado 'Estima'.

Figura 18 – Janela para importação de Dados

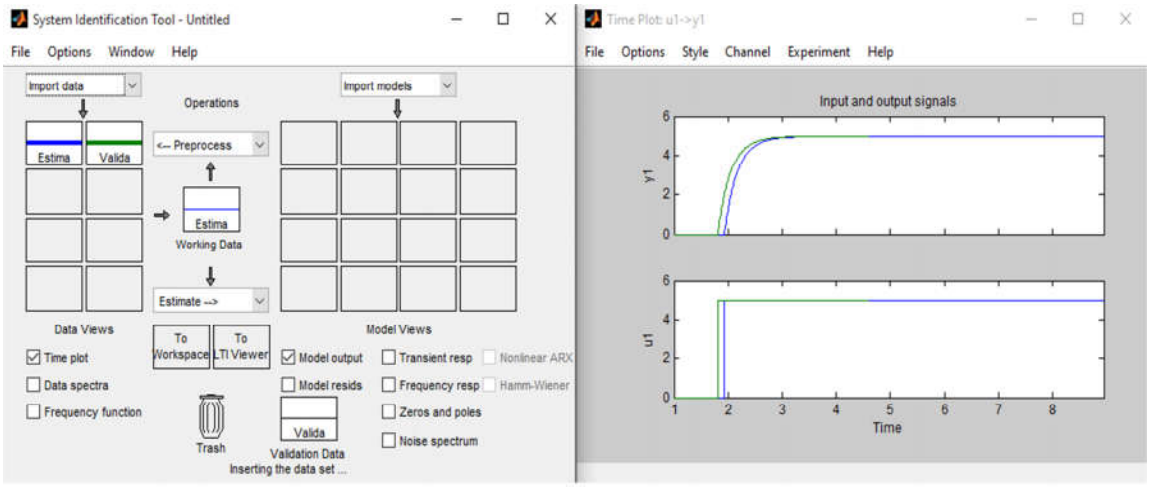


Fonte: Autoria própria.

O intervalo de amostragem é definido através do número de leituras ocorridas por segundo (dado fornecido na primeira linha da *Command Window* após cada aquisição, como mostra a figura17), sendo o intervalo de amostragem igual ao inverso do número de leituras ocorridas neste um segundo, para esse exemplo obtêm-se o valor de aproximadamente 0,012 segundos por amostragem.

O gráfico dos dados importados pode ser visualizado ao marcar a opção '*Time plot*', atentando-se que para estimar um modelo correto os dados de estimação devem ser colocados no quadro '*Working Data*' e os dados de validação no quadro '*Validation Data*', para coloca-los nas respectivas caixas, basta clicar e arrastar o dado até o local desejado. A figura 19 apresenta onde devem estar alocados cada dado coletado além de apresenta-los no gráfico ao lado.

Figura 19 - Exibição de dados importados no *System Identification Toolbox*



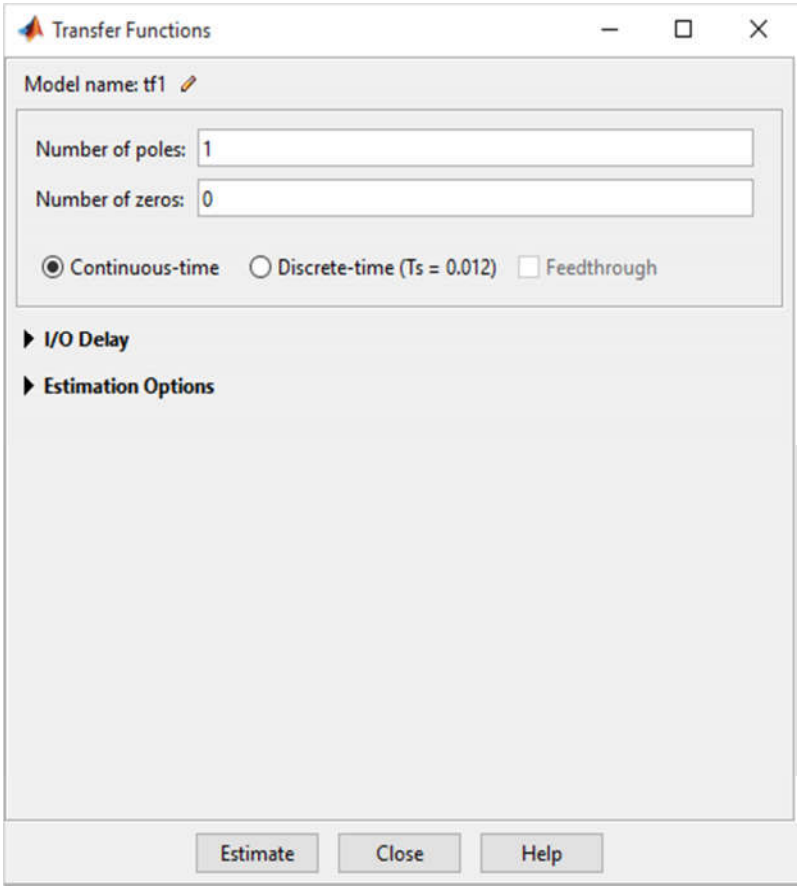
Fonte: Autoria própria.

Após realizar todo o procedimento de aquisição e preparação dos dados, pode-se começar a escolher os modelos para o processo estudado. A ferramenta possui diversos modelos que podem ser utilizados, desde os mais simples como uma função de transferência até os mais complexos como modelos não lineares, porém nesse experimento é utilizado e estimado apenas alguns mais simples, com o objetivo de facilitar o entendimento e a introdução à ferramenta.

Ao clicar na aba '*Estimate*' o aluno tem diversas opções de modelos que podem ser estimados. O usuário deve procurar, de forma iterativa, um modelo que melhor representa o processo e que pode satisfazê-lo, pois em diversos casos há modelos diferentes que podem de forma satisfatória representar o mesmo processo, permitindo o usuário escolher aquele que melhor o satisfaz, devido a esse fator, geralmente o usuário deve utilizar modelos com os quais já se tenha um pouco mais conhecimento do mesmo, para que a compreensão e a análise ocorram facilmente. Este projeto foca apenas nos modelos de funções de transferências e os de polinômios.

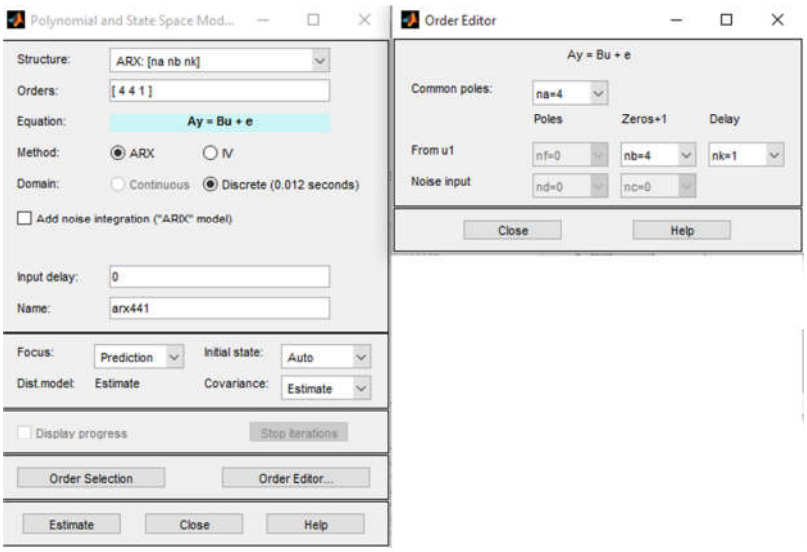
Ao escolher determinado modelo a ser estimado, o aluno deve então pré-determinar iterativamente, ou com algum conhecimento prévio do processo, ordens e parâmetros do modelo, como por exemplo, em um modelo de função de transferência ele deve escolher o número de polos e zeros como mostra a figura 20, e em modelos polinomiais o usuário tem que determinar a ordem dos polinômios arbitrários *A(z), B(z) C(z), D(z) e F(z)*, sendo eles representados como na, nb, nc, nd e nf respectivamente na ferramenta e o aluno deve também determinar o nk, sendo este o parâmetro de atraso de resposta (*delay*). Em caso de dúvidas e incertezas em determinar as ordens de cada polinômio o usuário pode utilizar o botão '*Order Editor*' que abre uma nova janela com a equação do modelo e os parâmetros nomeados, ajudando a relembrar quais são parâmetros de ruídos, polos, zeros+1 e atraso como mostra a figura 21 um exemplo com o modelo ARX.

Figura 20 - Janela de estimação de uma Função de Transferência.



Fonte: Autoria própria.

Figura 21 - Janelas de estimação de polinômios e '*Order Editor*'



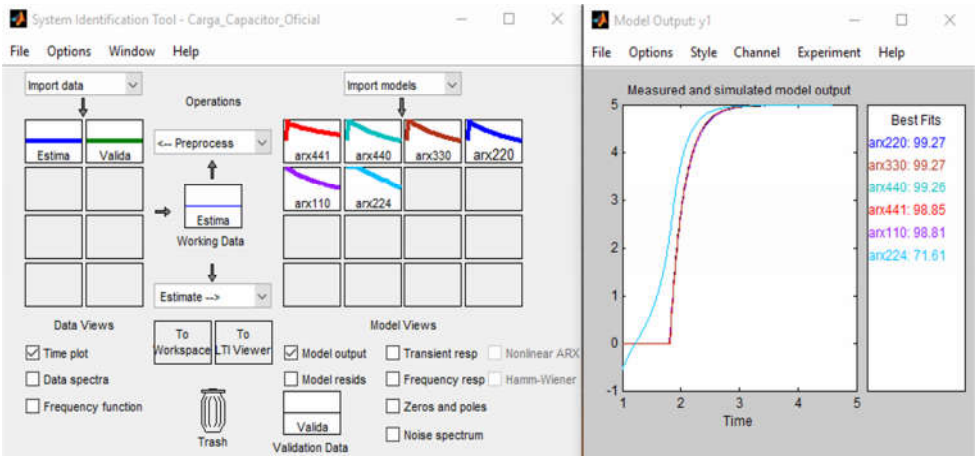
Fonte: Autoria própria

Nos casos de modelos polinomiais o aluno deve estimar diversos modelos de diferentes ordens para conseguir encontrar o que irá identificar o processo de forma satisfatória e com uma complexidade aceitável, já que é possível estimar polinômios com ordem superior a 4, podendo esse modelo até ser satisfatório, porém o número da ordem elevada do modelo pode dificultar o entendimento e utilização do mesmo (RAMÍREZ, 2012). Então é necessário trabalhar em cima dos parâmetros para que se possa obter um modelo satisfatório com a menor ordem possível, sabendo que os polinômios de ordens superiores também podem ser satisfatórios dependendo do sistema estudado, porém só é útil de acordo com a familiaridade do usuário com tal modelo e ordem.

Um parâmetro que ajuda a diminuir a ordem de um modelo é o de atraso (nk), ao determinar o atraso correto de um processo que está sendo estimado, a ordem do mesmo deve com certeza diminuir em relação ao modelo estimado com um atraso errado. No processo estudado sabe-se que em uma carga de capacitor não há um atraso de resposta na saída, pois ao ocorrer uma variação na entrada a saída varia instantaneamente. A figura 22 mostra alguns exemplos de estimação do modelo ARX e suas respectivas porcentagens de adequação da saída do modelo em relação à saída do sistema real (Janela *Model Output*). Ao alterar o parâmetro de atraso para zero (valor correto de atrasado desse sistema) é possível diminuir a ordem dos parâmetros na e nb e melhorando a porcentagem de adequação como apresenta, por exemplo, os modelos arx220 e arx441 sendo os números que seguem o nome arx os parâmetros na, nb e nk respectivamente. A figura 22 mostra também o oposto, pois ao alterar

o atraso para um valor muito alto mantendo a mesma ordem a porcentagem de adequação diminui como mostra o modelo arx224, tornando-o insatisfatório, apesar de também ser um modelo de baixa ordem.

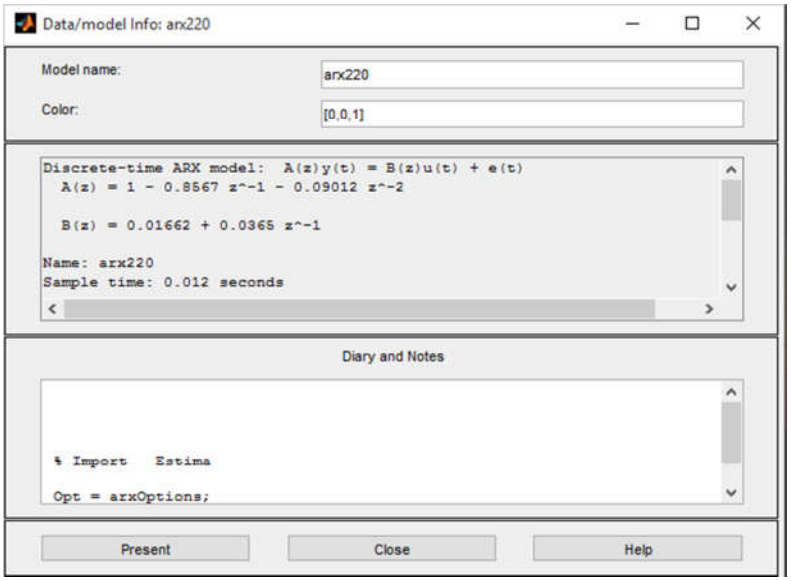
Figura 22- Exemplo de Estimações ARX alterando ordens e atraso.



Fonte: Autoria própria.

Após realizar algumas estimações de sua preferência o aluno pode então compará-las de acordo com o erro, para então escolher aquela que irá atendê-lo melhor. Com um duplo clique em cima do modelo considerado satisfatório o aluno pode obter mais informações sobre o modelo, como apresenta a figura 23, utilizando como exemplo o modelo arx220.

Figura 23 - Janela de informação do Modelo ARX220



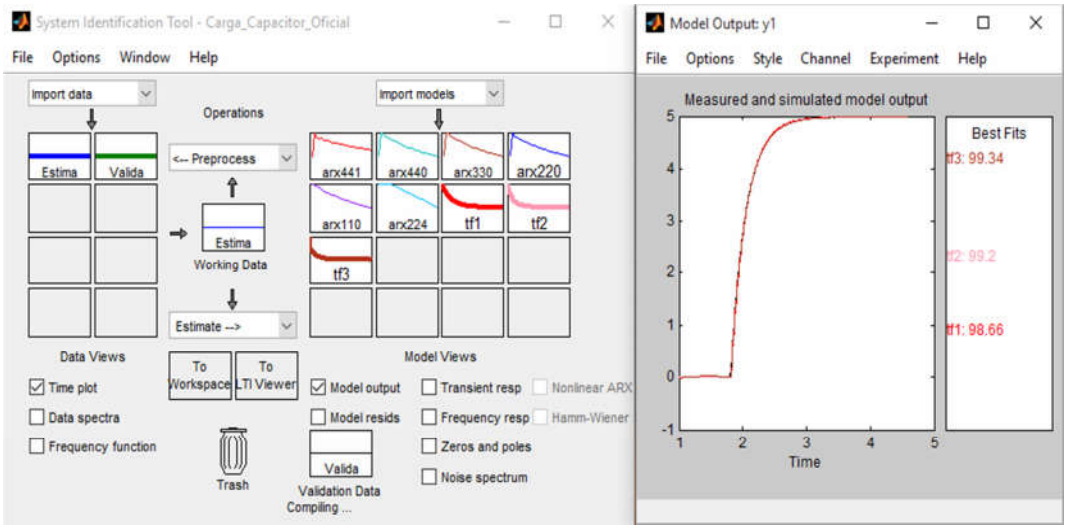
Fonte: Autoria própria.

Na janela de informação do modelo o aluno obtém a equação do modelo ARX em tempo discreto, sendo essa equação uma variação simples da equação 2.7, e obtém então os polinômios arbitrários *A(z)* e *B(z)*, obtendo assim a equação de relação entre saída e entrada do modelo, dada neste caso por:

$$y(k) = \frac{0{,}01662 + 0{,}0365z^{-1}}{1 - 0{,}8567z^{-1} - 0{,}09012z^{-2}}u(k) + \frac{1}{1 - 0{,}8567z^{-1} - 0{,}09012z^{-2}}e(k) \quad (2.10)$$

É possível realizar o mesmo processo com os modelos de função de transferência, porém nos modelos de função de transferência é necessário apenas determinar o número de polos e zeros, e posteriormente decidir qual função de transferência representa satisfatoriamente o sistema. A figura 24 apresenta algumas funções de transferências estimadas no quadro de modelos, além dos modelos ARX estimados anteriormente, mostrando a porcentagem de adequação de cada uma delas, sendo excluídos do gráfico os modelos ARX com o único clique sobre cada um deles para melhor visualizar a adequação das funções de transferências.

Figura 24 - Exemplo de Estimações de Função de Transferência.



Fonte: Autoria própria.

As funções de transferências nomeadas como tf1, tf2, tf3 foram parametrizadas como sendo de primeira, segunda e terceira ordem respectivamente. É possível visualizar que as três possuem uma porcentagem de adequação satisfatória e muito próxima uma a outra. Por se tratar de um experimento de primeira ordem, as funções de transferência de segunda ordem

também conseguem representar o processo, porém como discutido anteriormente no caso dos polinômios, é preferível utilizar funções com a menor ordem possível. Através da teoria tem-se que a função de transferência de primeira ordem de um circuito RC é dada por:

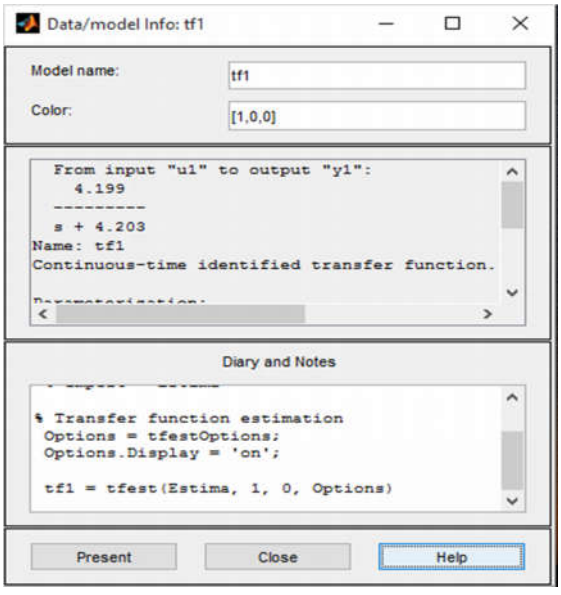
$$\frac{Y(s)}{U(s)} = G(s) = \frac{1}{RCs + 1} \tag{2.11}$$

E então substituindo os valores de R e de C pré-estabelecidos neste exemplo, tem-se a equação da função de transferência teórica para esse modelo que é dado por:

$$\frac{Y(s)}{U(s)} = G(s) = \frac{1}{0{,}22s + 1} \tag{2.11}$$

Já foi dito diversas vezes que o conhecimento prévio do processo é de suma importância para ajudar na estimativa de um modelo para o processo, considerando que se trata de um processo de primeira ordem, iremos nos atentar ao modelo tf1, que é um modelo satisfatório de primeira ordem o qual é mais fácil o entendimento e a comparação com a teoria. Utilizando a equação teórica 2.11 é possível compará-la com a equação do modelo estimado. A figura 25 apresenta a janela de informações de tf1 para obter a função de transferência do modelo estimado e compará-la com a teórica.

Figura 25 - Janela de informações da função de transferência tf1



Fonte: Autoria própria.

A equação da função de transferência tf1 é dada por:

$$\frac{Y(s)}{U(s)} = G(s) = \frac{4{,}199}{s + 4{,}203} \tag{2.12}$$

Para poder comparar a equação estimada através da ferramenta com a equação teórica, é necessário que o valor do numerador seja igual a 1, para isso é necessário dividir equação 2.12 pelo próprio valor do numerador (4,199), obtendo então a equação 2.13.

$$\frac{Y(s)}{U(s)} = G(s) = \frac{1}{0{,}23s + 1} \tag{2.13}$$

Deste modo o aluno pode então comparar a equação da função de transferência estimada com a teórica e concluir que as duas são praticamente iguais, tornando-o o modelo estimado totalmente satisfatório para o processo analisado. É importante observar que as equações não são exatamente iguais, pois na função de transferência teórica não são levados em consideração alguns aspectos do circuito real, tais como: resistência dos cabos, ruídos e entre outros.

O apêndice C fornece alguns possíveis modelos e informações sobre os mesmos que o aluno pode encontrar para esse sistema. É possível repetir todo procedimento, para ratificar todos os resultados obtidos, pois em muitos casos para uma melhor estimação de modelos a aquisição de dados deve ser feita novamente. O aluno pode o com o mesmo circuito e programa realizar a aquisição de dados da descarga do capacitor e então estimar novos modelos e compará-los com os modelos anteriores.

3.3 SEGUNDA ATIVIDADE

A segunda atividade visa aprofundar o conhecimento do *Toolbox*, concretizar a introdução realizada na atividade anterior, utilizar e explicar mais funções disponíveis na ferramenta, apresentar ao aluno um exemplo com dados considerados mais complexos do que a primeira atividade expondo a utilidade que a ferramenta tem em realizar qualquer tipo de estudo que seja necessário entre duas grandezas correlacionadas e apresentar a importância que a ferramenta possa ter no futuro ao adquirir o conhecimento de certo processo no ambiente de trabalho.

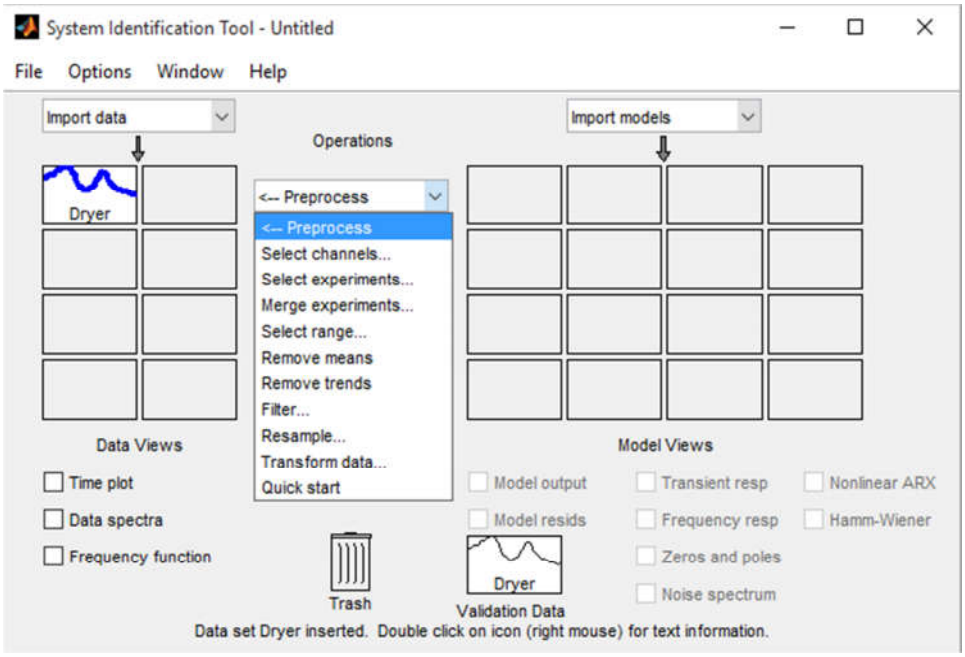
Nesse procedimento os dados utilizados são disponibilizados pela própria ferramenta, não havendo a necessidade de uma coleta prévia de dados. Esses dados foram utilizados com o objetivo de manter o foco do projeto na introdução de mais funções e ações disponíveis do *Toolbox*, pois a implementação de um sistema mais complexo e aquisição de dados do mesmo iria desviar o foco do tema principal do trabalho. São dados de um sistema onde a entrada é a potência de um secador e a saída é a temperatura do ar onde o secador atua. A utilização desses dados é feita para mostrar ao aluno que dados de um sistema real também podem ser

estimados, pois o objetivo do projeto é abrir a mente do aluno com relação ao programa utilizado, deixando claro que o programa pode, com certeza, ser utilizado em diversas áreas.

Para iniciar a segunda atividade o *System Identification Toolbox* deve ser inicializado normalmente assim como na primeira atividade. Ao abrir aba de importação de dados o aluno deve escolher a opção '*Example*' onde abre automaticamente a janela de importação com o dado já nomeado de '*Dryer*' que deve ser importado para iniciar o aprendizado de novas funções.

Lembrando que é preciso utilizar dados diferentes para estimação e validação, deve-se então realizar um pré-processamento com o dado importado. A ferramenta oferece diversas opções para processar o dado antes de utilizá-lo como mostra a figura 26. Porém nessa atividade utilizam-se apenas duas opções de pré-processamento para obtermos dados com valores diferentes e com um número menor de amostras para serem utilizados como dado de validação, sendo elas a remoção de tendências (*Remove trends*) e a seleção de extensão (*Select Range*).

Figura 26 – Opções de Pré-processamento de dados



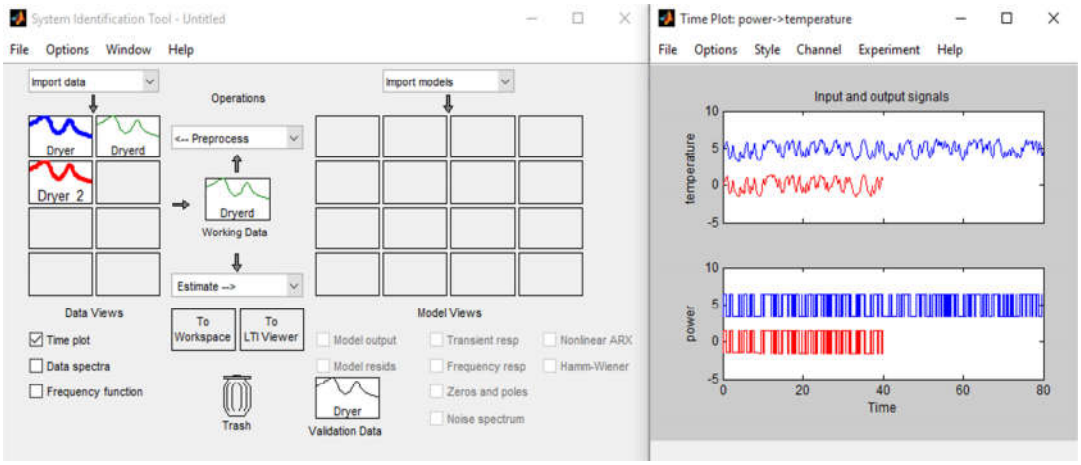
Fonte: Autoria própria.

O dado a ser processado deve estar no quadro *Esimation Data*, pois somente o mesmo é retrabalhado. O aluno deve então remover as tendências do dado *Dryer* e posteriormente

alocar o novo dado com as tendências removidas no quadro *Estimation Data* e então selecionar uma extensão com um número de amostras menores para esse novo dado.

A figura 27 mostra a interface da ferramenta com os dois novos dados criados através do pré-processamento e o gráfico de entrada e saída dos dados *Dryer* e *Dryer_2*, sendo o *Dryerd* o dado com as tendências removidas, e *Dryer_2* o dado obtido através do *Dryerd* com uma extensão menor de 0 até 40 segundos.

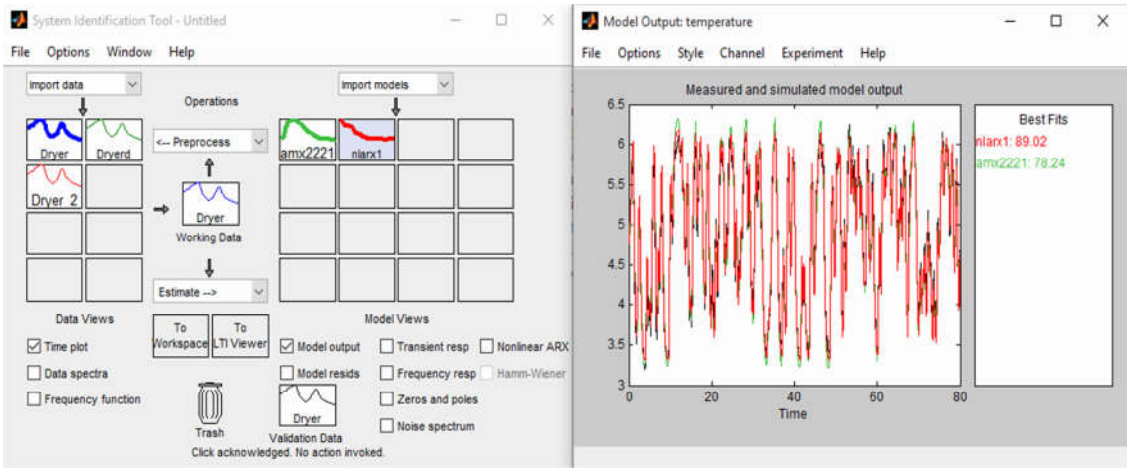
Figura 27- Interface do *Toolbox* com os dados pré-processados



Fonte: Autoria própria.

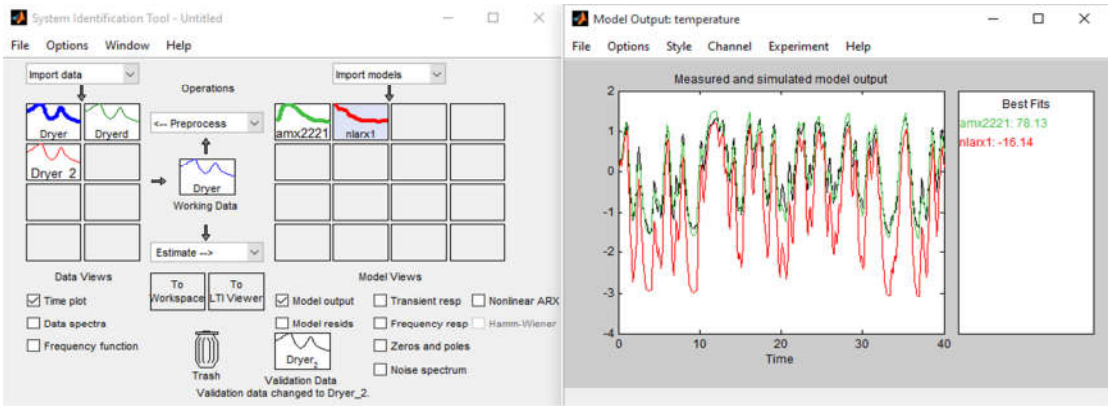
Para comprovar e mostrar ao estudante que é realmente necessário utilizar dados de estimação e validação diferentes para estimar um modelo satisfatório, um modelo linear e um modelo não linear são estimados com o dado *Dryer* no quadro de estimação e validação. Observa-se então que deste modo que o modelo não linear ARX (nlarx1) possui um melhor ajuste a saída real com um valor de 89,02% do que o modelo linear amx2221 com um valor de 78,24%, como mostra a figura 28, porém ao utilizar o dado pré-processado *Dryer_2* como dado de validação, o ajuste do modelo não linear que era superior ao modelo linear diminui drasticamente, chegando a ser negativo, enquanto o modelo linear mantém o valor de ajuste próximo ao valor anterior, como mostra a figura 29.

Figura 28 – Modelos estimados com o mesmo dado para Estimação e Validação



Fonte: Autoria própria.

Figura 29 - Modelos estimados com dados diferentes para Estimação e Validação

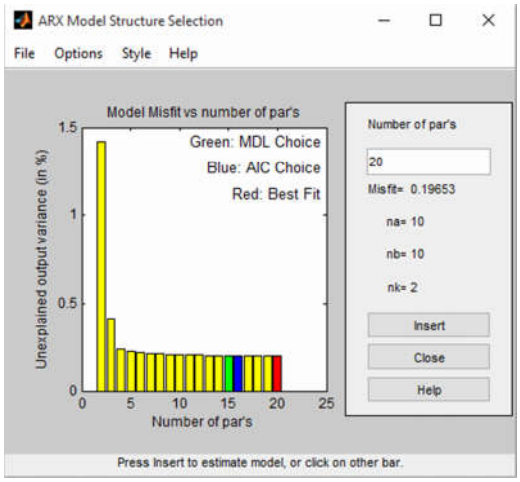


Fonte: Autoria própria.

Com os dados corretamente alocados o usuário pode iniciar a estimação de modelos com mais confiança. Porém ao estimar um modelo com pouco conhecimento prévio sobre o sistema, tal como o atraso de resposta, é difícil saber por onde iniciar. Mas a função *Order Selection* disponível na janela do modelo ARX é possível encontrar um ponto de início para as estimações. A função realiza o cálculo de diversas ordens e atrasos de modelos ARX que melhor se ajustam ao sistema estudado, mostrando em um gráfico cada modelo através de barras, onde o gráfico apresenta variações inexplicáveis na saída por número de pares, sendo o melhor modelo àquele que apresenta a menor variação inexplicável na saída, como mostra a figura 30. O usuário pode utilizar o melhor modelo através dessa função, porém geralmente a ordem dos parâmetros são altas, sendo a utilização dessa função recomendada para ter uma

noção próxima de qual o melhor valor de atraso a ser usado, sendo nesse caso o melhor atraso igual a 2 e assim encontrar modelos com esse atraso porém com ordens menores.

Figura 30 – Seleção de estrutura do modelo ARX



Fonte: Autoria própria.

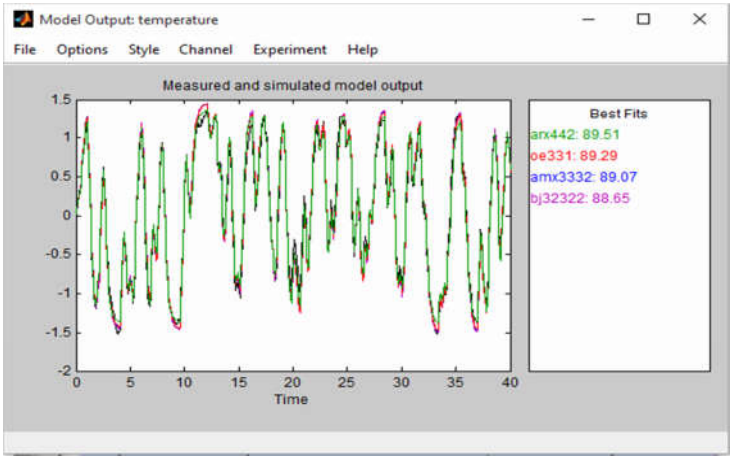
Após estimar alguns modelos considerados satisfatórios e antes de utilizar algum deles como sendo um modelo que melhor se ajusta ao modelo real é preciso fazer a analise residual dos mesmos.

> Resíduos são diferenças entre um passo da saída previsto do modelo e a saída do dado de validação. A análise residual consiste em dois testes: o teste de brancura (*whiteness test*) e o teste de independência (*independesse test*). O critério do teste de brancura afirma que um bom modelo tem a função de auto correlação residual dentro do intervalo de confiança das estimativas correspondentes, indicando que os resíduos não estão correlacionados. O critério do teste de independência assegura que um bom modelo tem resíduos não correlacionados com entradas passadas, a evidência de correlação indica que o modelo não descreve como parte da saída relaciona-se com a entrada correspondente (LJUNG, 2015, p.16-23).

Desse modo um bom modelo deve passar em ambas as análises, ou seja, estar em ambos os teste dentro da faixa de confiança. Apenas para modelos OE e função de transferência não se deve atentar para os resultados do teste de brancura, apenas para o teste de independência.

A figura 31 mostra alguns exemplos de modelos estimados, onde todos possuem uma porcentagem de ajuste entre 88,65% e 89,51%, sendo o modelo arx442 com a maior porcentagem de ajuste, sendo ele a princípio o melhor modelo que representa o sistema.

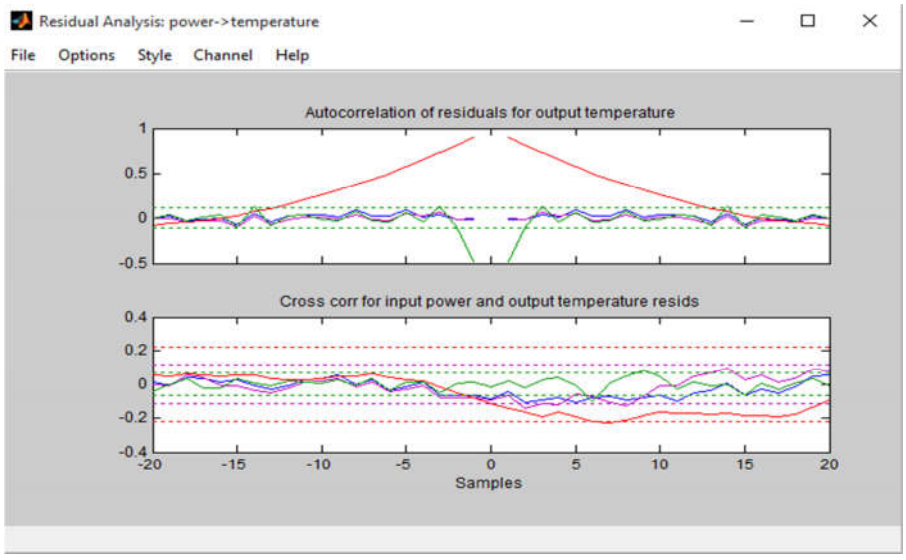
Figura 31 – Exemplo de modelos e suas porcentagens de ajuste



Fonte: Autoria própria.

Porém ao verificar as análises de resíduos, através da opção *Model resids* onde o gráfico superior é destinado ao teste de brancura e o inferior ao teste de independência (LJUNG, 2015), é possível verificar que o modelo arx442 de cor verde, apesar de possuir o maior ajuste em porcentagem entre os modelos, o mesmo não se encontra dentro do intervalo de confiança no teste de brancura como apresenta a figura 32, tornando-o insatisfatório para a representação do sistema. Vale ressaltar que para modelo oe333 a análise do teste de brancura deve ser desconsiderada e que o modelo bj32322 possui um pequeno pico fora da faixa da confiança para o teste de independência podendo ser descartado.

Figura 32 – Análise Residual de 4 modelos exemplos



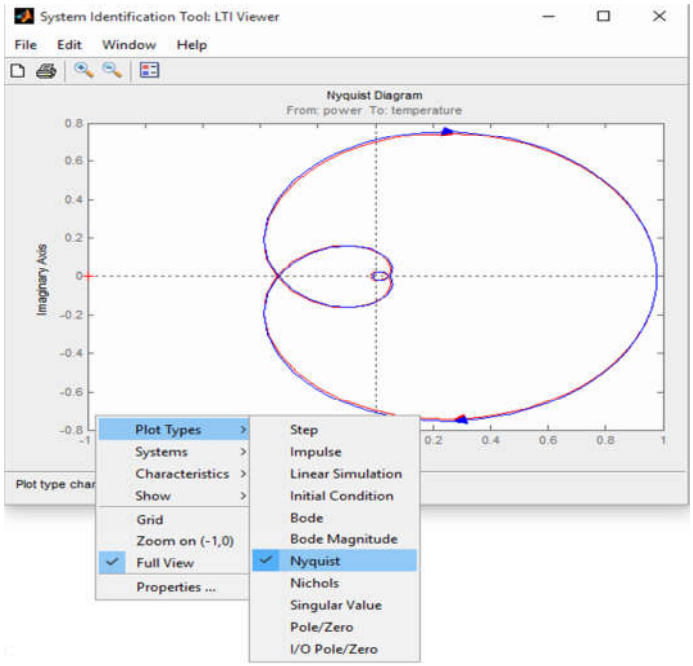
Fonte: Autoria própria.

Feito a análise residual o usuário pode escolher qual o melhor modelo para representar o sistema (LJUNG, 2015), preferindo sempre escolher um modelo com uma maior intimidade e de baixa ordem, sendo nesse caso o modelo oe331. Ao executar o duplo clique sobre o modelo o usuário pode então obter a equação do modelo escolhido que melhor descreve o sistema como descrito na primeira atividade.

Em ambas as atividades o aluno pode também utilizar o recurso de transportar um ou mais modelos para o quadro *LTI view* para visualizar mais gráficos correspondentes ao modelo. Dependendo do conhecimento que se queira obter sobre o modelo. O *LTI view* pode trazer informações úteis e interessantes (LJUNG, 2015), como diagramas e gráficos apresentados durante o curso de Controle Linear, tais como: diagrama de Bode, diagrama de Nyquist e o gráfico de polos e zeros.

Ao arrastar um modelo para o *LTI view* uma janela abre automaticamente apresentando o gráfico de resposta ao degrau do modelo (LJUNG, 2015). Para obter outros gráficos, como por exemplo, o diagrama de Nyquist basta o aluno clicar com o botão direito sobre o gráfico e então selecionar *Plot Types,* como mostra a figura 33, e selecionar o diagrama ou gráfico a ser exibido.

Figura 33: Diagrama de Nyquist dos modelos amx3332 e oe 331.



Fonte: Autoria própria.

Todos os procedimentos e funções realizados nas atividades do projeto são importantes para que a estimação de modelos seja executada de uma maneira simples e clara.

4 CONCLUSÃO

A ótima formação de um engenheiro é muito importante não só para si mesmo como também para a sociedade que irá receber os seus serviços direta ou indiretamente. Os estudos de princípios básicos assim como a introdução de novas tecnologias, novos programas são muito importantes para ser um engenheiro com uma base de educação sólida e ao mesmo tempo atualizada.

O trabalho teve como objetivo introduzir ao aluno a ferramenta *System Identification Toolbox* disponibilizada por meio do MATLAB, um programa muito utilizado em diversas áreas da engenharia, e salientar a importância que a identificação de sistema pode ter em seu ambiente de trabalho. Este objetivo foi claramente atingido, pois diversas funções e explicações foram realizadas e comprovadas mostrando ao aluno como utilizar corretamente a ferramenta e a importância de identificar um sistema. Na primeira atividade o aluno tem acesso aos princípios básicos e importantes sobre ferramenta, além de poder comparar equações teóricas com as equações obtidas por meio da ferramenta, comparando assim a prática com a teoria. E na segunda atividade foi utilizado um conjunto de dados de um sistema real mais complexo apresentando novas funções e ações para uma estimação satisfatória e mostrando também que a ferramenta pode claramente ser aplicada em processos considerados mais intricados. Além do objetivo alcançado foi possível também apresentar ao aluno conceitos básicos de aquisição de dados e introduzi-lo ao *Arduino,* uma plataforma de código aberto muito utilizada ultimamente. Conceitos que irão ser utilizados e ensinados com um maior aprofundamento em outras matérias, como por exemplo, a plataforma do *Arduino* utilizada na disciplina de Sistemas Micro Computadorizados.

A Faculdade de Engenharia de Guaratinguetá disponibiliza todos os materiais utilizados no projeto para a realização das atividades, não sendo necessário qualquer investimento da faculdade para a implementação das atividades em Laboratórios de Controle Linear, podendo o mesmo ser implementado assim que desejado.

Portanto, o projeto mostrou ser uma boa opção para ser utilizado no laboratório de controle linear para iniciar a imersão dos alunos na área de identificação de sistema e mostra-los o quão importante é a identificação de sistemas no ambiente profissional.

4.1 TRABALHOS FUTUROS

Como o *System Identification Toolbox* é uma ferramenta com muitas funções o aluno pode se aprofundar e pesquisar mais sobre todas elas no site da MathWorks. Algumas sugestões e outras formas de utilização da ferramenta são descritas a seguir.

- Pode-se criar um sistema de blocos no SIMULINK e transportar os dados entrada e saída desse sistema para o *Workspace* do MATLAB para os mesmo serem para dentro da ferramenta.
- É possível enviar um determinado modelo estimado na ferramenta para o SIMULINK e então realizar mais testes ou acrescentar mais blocos e operações em conjunto com o modelo.
- Dados de entrada e saída dispostos em uma tabela do EXCEL podem ser importados para o MATLAB e posteriormente importados para a ferramenta.

REFERÊNCIA

ARDUINO Website. Disponível em: < http://www.arduino.cc/en/Guide/Introduction> Acesso em 28 out. 2015.

MOLER, C. **The Origens of MATLAB.** Disponível em: < http://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html> Acesso em 28 out. 2015.

MATHWORKS Website. Disponível em: <http://www.mathworks.com/help/ident/> Acesso em 28 out. 2015.

LJUNG, L. **System identification**. 2.ed. Upper saddle river: Prentice-Hall PTR, 1999. 609p.

LJUNG, L. **System identification toolbox**: user's guide. Disponível em: <http://www.mathworks.com/help/pdf_doc/ident/ident.pdf> Acesso em: 14 nov. 2015.

RAMÍREZ, N.B, **Métodos de identificação de processos**. Disponível em: <http://www.professores.uff.br/controledeprocessos-eq/images/stories/Control_Aula14_Met-Ident.pdf> Acesso em 21 nov.2015.

MATHWORKS Website, **Input-Output Polynomial Models**. Disponível em: <http://www.mathworks.com/help/ident/input-output-polynomial-models.html> Acesso em 21 nov 2015

BIBLIOGRAFIA CONSULTADA

AGUIRRE, L A. **Introdução à identificação de sistemas**: técnicas lineares e não lineares aplicadas a sistemas reais, Belo Horizonte: UFMG, 2000, 729p.

MATHWORKS Website. Disponível em:
<http://www.mathworks.com/help/matlab/> Acesso em 28 out. 2015.

OGATA, Katsuhiko. **Engenharia de controle moderno**. 4.ed. São Paulo: Pearson Prentice Hall, c2003. 788p.

NATIONAL INSTRUMENT – **Conceitos básicos da amostra analógica**. Disponível em: <http://www.ni.com/white-paper/3016/pt/> Acesso em 03 nov. 2015.

LJUNG, L. **System identification**. 2. Ed, Upper Saddle River: Prentice-Hall PTR, 1999. 609p.

MATHWORKS Website. Disponível em:
<http://www.mathworks.com/help/ident/input-output-polynomial-models.html> Acesso em 21 nov 2015

APÊNDICE A – Programação no arduino para carga e descarga do capacitor

```
int out1 =0;          //Definição da primeira Leitura (Tensão da Fonte)
int out2 = 0;         // Definicção da segunda leitura (Tensão do Capacitor)

void setup() {

Serial.begin(9600);       // Inicia-se a comunicação serial
}

void loop() {

  out1 = analogRead(A0);      // Leitura da Tensão da Fonte
  out2 = analogRead(A1);      // Leitura da Tensão do Capacitor

  Serial.print(out1);       // Envio da Leitura da Tensão Fonte atraves da porta serial
  Serial.print(",");
  Serial.println(out2);    // Envio da Leitura da Tensão do Capacitor atraves da porta serial
  delay(10);                // Tempo de espera para que a comunicação entre Arduino e Matlab //ocorra corretamente,

}
```

APÊNDICE B – Programação no editor para carga e descarga do capacitor

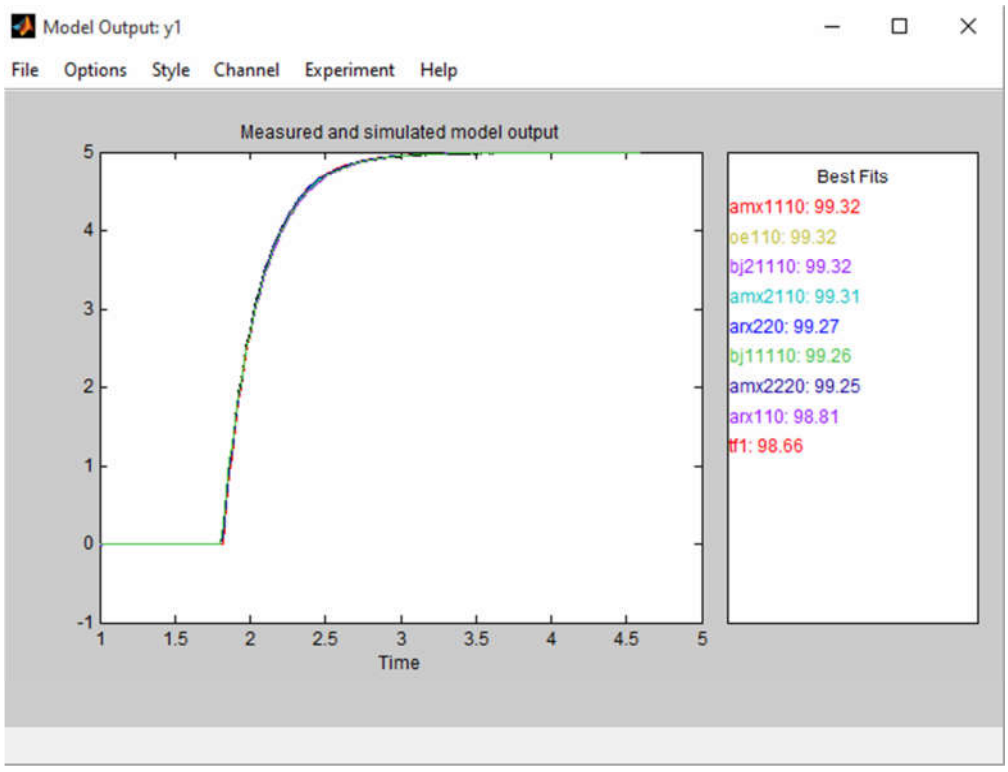
```
delete(instrfind({'Port'},{'COM3'}));  %Limpa a porta serial
s = serial ('COM3','BaudRate',9600,'Terminator','CR/LF'); %Estalebece conexão com a Porta Serial (Neste caso a COM3)
warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
fopen(s);%Inicia a comunicação com o Arduino

%parametros de medidas
tmax = 10; %Tempo de aquisição
rate = 33;

%Preparando o grafico
f=figure('Name','Captura 1'); %Nomeia a Janela do Grafico
a = axes ('Xlim',[0 tmax],'Ylim',[-1.0 6.0]); %Determina os valores dos eixos
l1 = line(nan,nan,'Color','r','LineWidth',2); %Determina a cor da linhas
l2 = line(nan,nan,'Color','b','LineWidth',2);

xlabel('Tempo (s)') %Nomeia o Eixo X
ylabel('Volts (V)') %Nomeia o Eixo Y
title('Amostra em tempo real Arduino') %Nomeia o Titulo do Grafico
grid on
hold on

v1=zeros(1,tmax*rate); % matrizes V1 e V2 para receber os dados.
v2=zeros(1,tmax*rate);
x2=zeros(1,tmax*rate);
i=1;
t=0;

tic

while t<tmax %Realiza a Leitura atraves do Arduino por um tempo de tmax
t=toc;
```

```
a=fscanf(s,'%d,%d')'; %Realiza a leitura das tensões.
v1(i)=a(1)*5/1024;     %Equaliza os valores recebidos para que sejam exibidos entre 0 e 5 Volts.
v2(i)=a(2)*5/1024;

x2 = linspace(0,tmax,tmax*rate);
x = linspace(0,i/rate,i);
set(l1,'YData',v1(1:i),'XData',x); %Plota a primeira leitura (Tensão na Fonte)
set(l2,'YData',v2(1:i),'XData',x); %Plota a segunda leitura (Tensão no Capacitor)
drawnow
v1;
datox=[x,v1(1:i)]
i=i+1

end
clc;
fprintf('%g s de captura a %g dados/s \n',t,i/t); %Retorna quantos segundos e quantos leitura de dados ocorreram por segundos
```

APÊNDICE C – Modelos possíveis para a carga do capacitor

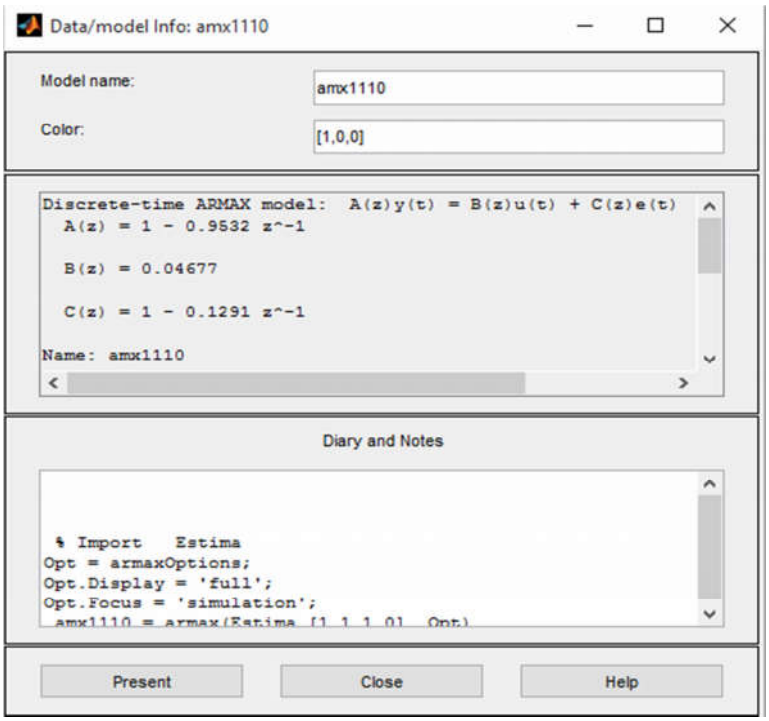
A figura 34 apresenta o gráfico de ajuste e suas respectivas porcentagens de diversos possíveis modelos que poderá ser encontrado tais como ARX, ARMAX, BJ e OE e as figuras 34 a 44 apresentam as informações de equação dos possíveis modelos que podem ser estimados na primeira atividade, além de trazer posteriormente a analise de resíduo (apresentada na segunda atividade) de todos eles para mostrar que nem todos esses modelos podem representar o sistema corretamente. A análise de resíduo da primeira atividade pode ser proposta como uma tarefa, com o intuito de mostra o aluno que talvez ele possa ter escolhido um modelo previamente satisfatório sendo o mesmo insatisfatório devido aos resíduos.

Figura 34 – Gráfico de ajuste de alguns modelos possíveis.



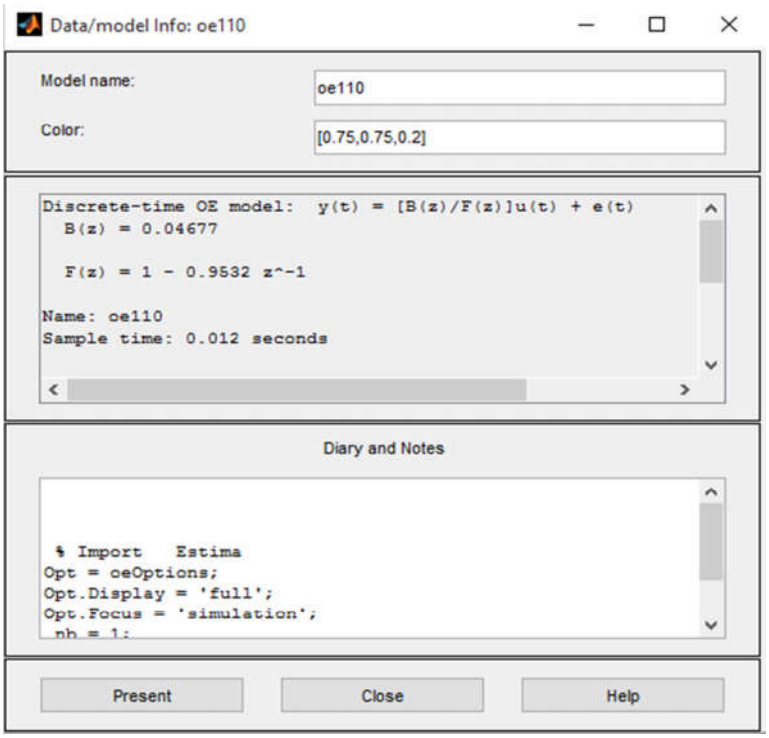
Fonte: Autoria própria.

Figura 35 – Informações do modelo amx1110



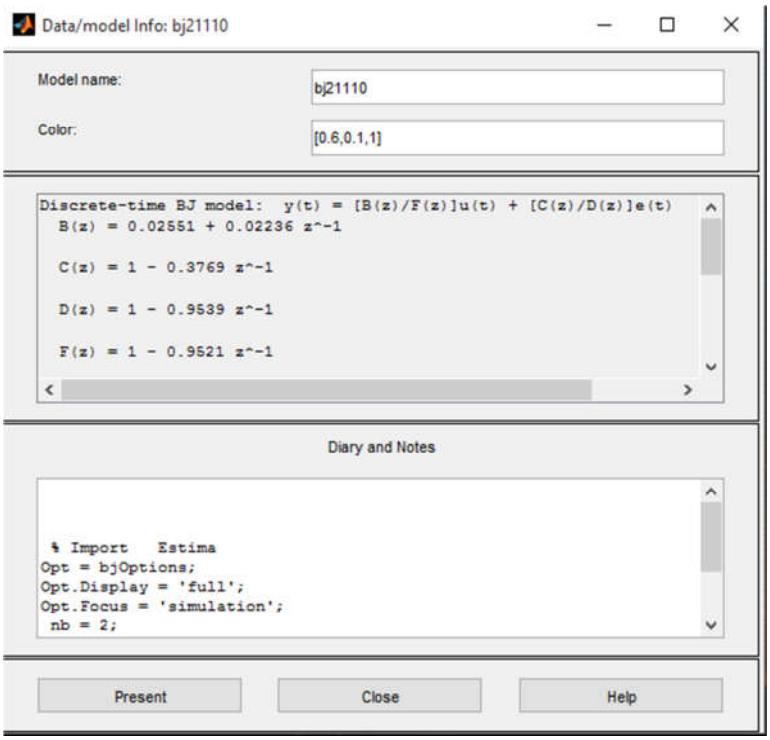
Fonte: Autoria própria.

Figura 36 – Informações do modelo oe110



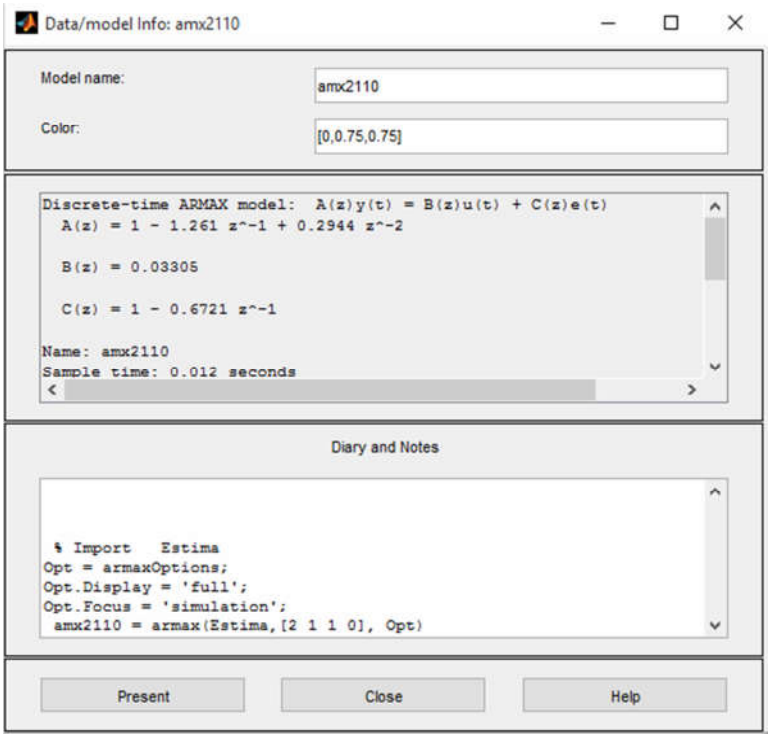
Fonte: Autoria própria.

Figura 37 – Informações do modelo bj21110



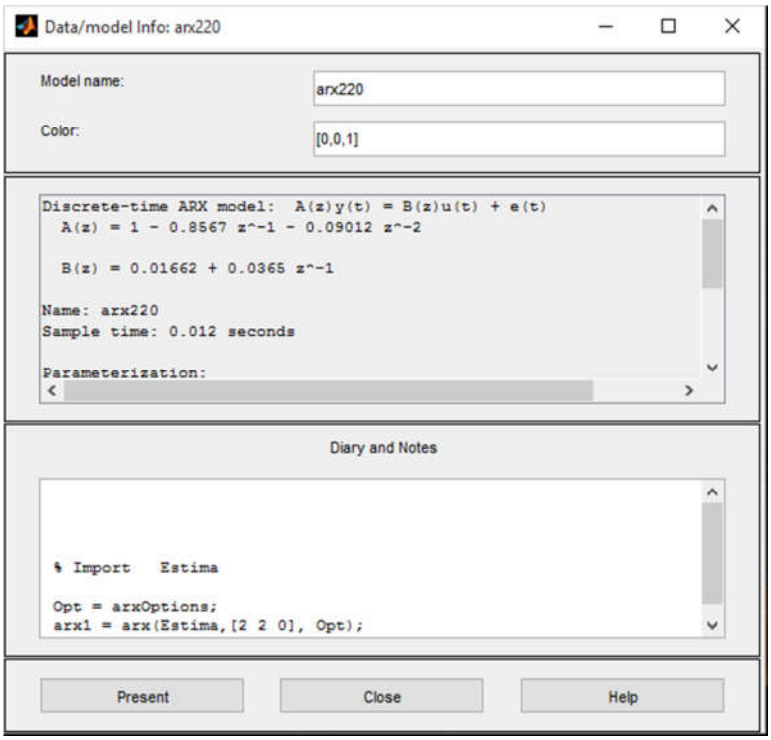
Fonte: Autoria própria.

Figura 38 – Informações do modelo amx2110



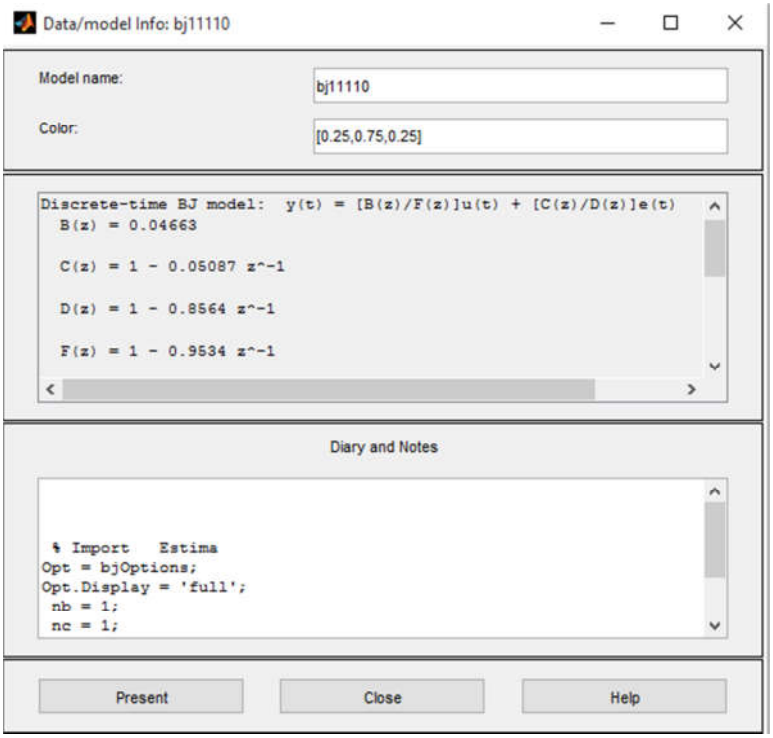
Fonte: Autoria própria.

Figura 39 – Informações do modelo arx220



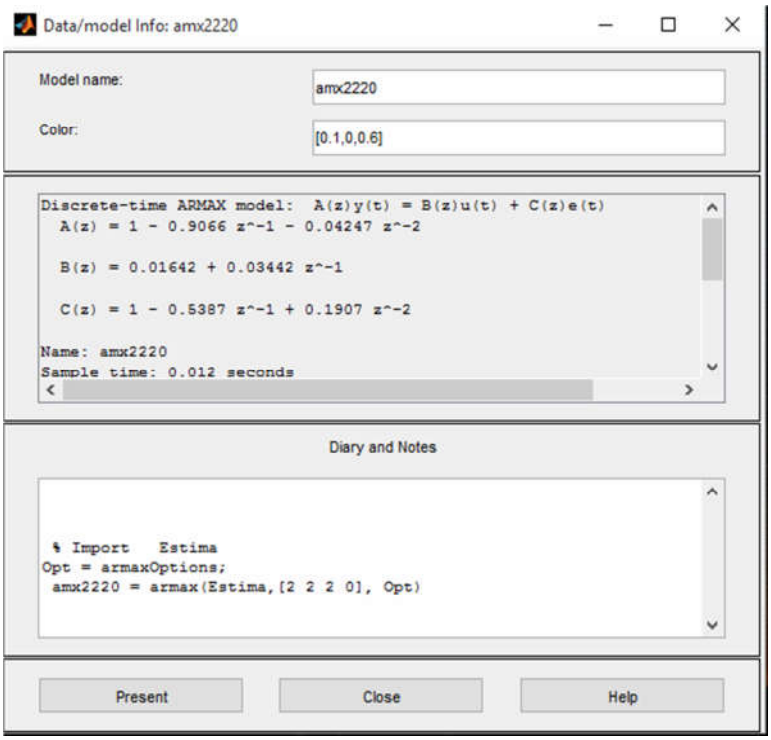
Fonte: Autoria própria.

Figura 40 – Informações do modelo bj11110



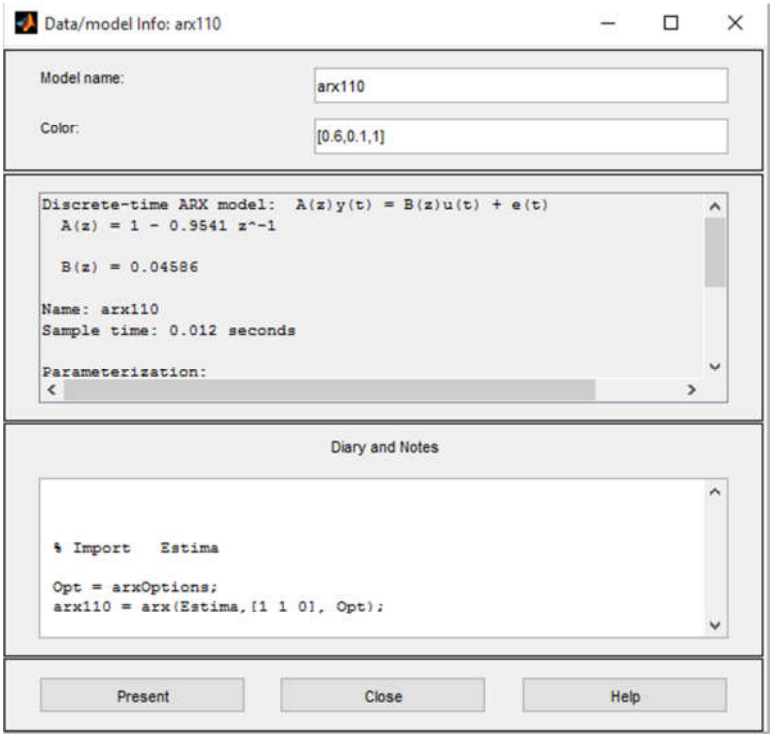
Fonte: Autoria própria.

Figura 41 – Informações do modelo amx2220



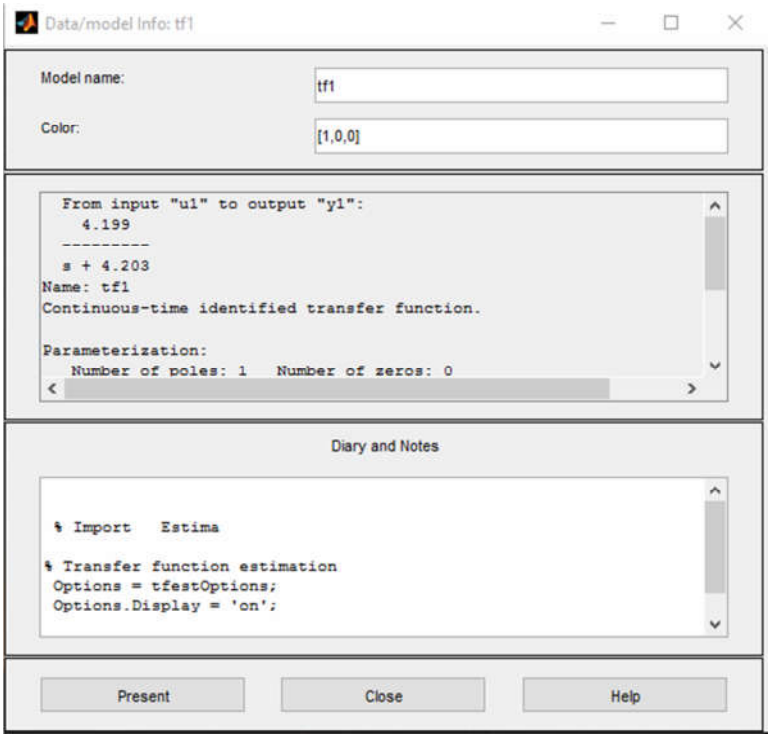
Fonte: Autoria própria.

Figura 42 – Informações do modelo arx110



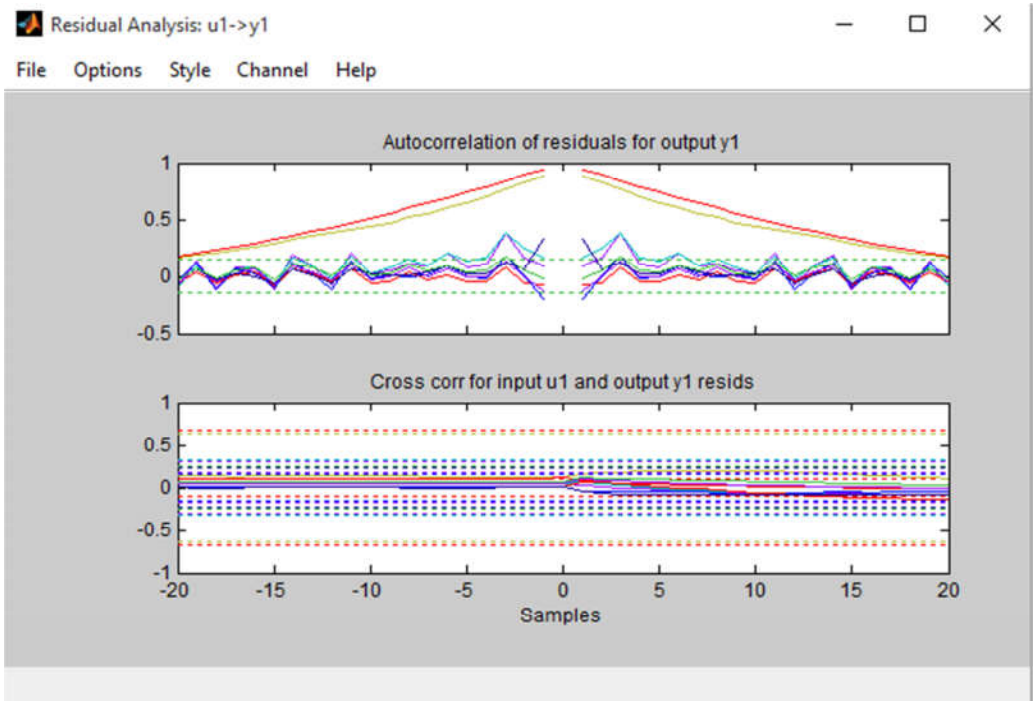
Fonte: Autoria própria.

Figura 43 – Informações do modelo tf1 de primeira ordem.



Fonte: Autoria própria.

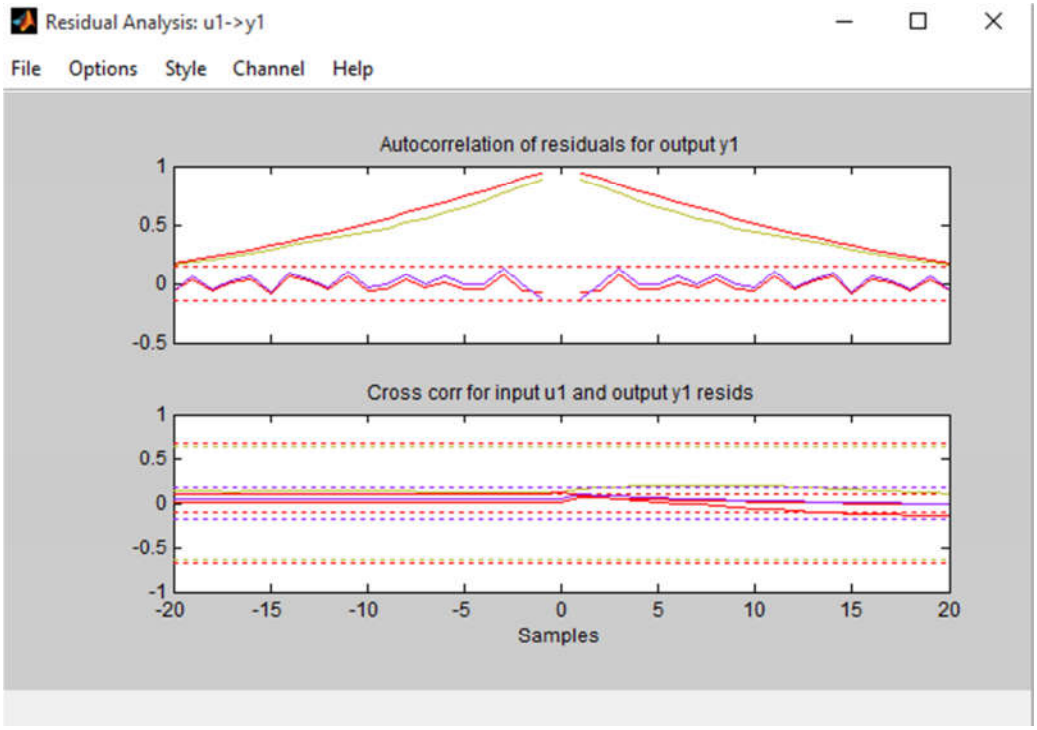
Figura 44 – Analise de Residual de todos modelos juntos



Fonte: Autoria própria.

Devido ao teste de brancura, para esse exemplo, os seguintes modelos foram considerados insatisfatórios: arx220, amx2220, amx2110, bj11110 e bj21110. A figura 45 apresenta a análise residual dos modelos satisfatórios mostrando que todos estão dentro do intervalo de confiança.

Figura 45 – Analise de Residual de todos modelos juntos



Fonte: Autoria própria.