UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

KLARISSA CARVALHO DE SOUZA

IDENTIFICAÇÃO NÃO-LINEAR NO ESPAÇO DE ESTADOS POR REGRESSÃO ESPARSA DE BANCADA MOTOR-GERADOR

TUCURUÍ-PA
2023

KLARISSA CARVALHO DE SOUZA

IDENTIFICAÇÃO NÃO-LINEAR NO ESPAÇO DE ESTADOS POR REGRESSÃO ESPARSA DE BANCADA MOTOR-GERADOR

TUCURUÍ-PA
2023

KLARISSA CARVALHO DE SOUZA

IDENTIFICAÇÃO NÃO-LINEAR NO ESPAÇO DE ESTADOS POR REGRESSÃO ESPARSA DE BANCADA MOTOR-GERADOR

DATA DE APROVAÇÃO: 25/01/2023

CONCEITO:

Prof. Dr. Raphael Barros Teixeira
Orientador: UFPA - CAMTUC - FEE

Prof. Dr. Cleison Daniel Silva
Membro: UFPA - CAMTUC - FEE

Prof. Dr. Rafael Suzuki Bayma
Membro: UFPA - CAMTUC - FEE

TUCURUÍ-PA
2023

RESUMO

No decorrer dos anos, cresce a busca pela identificação de sistemas dinâmicos utilizando aprendizagem de máquina. Essas ferramentas auxiliam, por exemplo, na compreensão e extração de informações de dados experimentais e na descoberta de estruturas de modelos matemáticos. Desta forma, este trabalho tem por objetivo apresentar os modelos obtidos de uma bancada motor-gerador utilizando o método de identificação esparsa de dinâmica não-linear (SINDy), este método tem-se mostrado útil na identificação da dinâmica não-linear, ao assumir que as equações possuem apenas alguns termos importantes que regem a dinâmica. A abordagem SINDy resolve um problema de regressão esparsa, eliminando os termos cujos coeficientes são menores que um limite. Os componentes usados no processo de identificação são baseadas no Otimizador Threshold Least Square (STLSq). São apresentados os modelos obtidos no experimento linear e não linear por meio do pacote PySINDy. A raiz do erro quadrático é calculada pela métrica NRMSE para cada um dos modelos. Ao final do trabalho é visto que o método SINDy é capaz d identificar o modelo da bancada motor-gerador.

Palavras-chave: 1. Otimizador 2. SINDy 3. Identificação não-linear 4.Motor-gerador

ABSTRACT

Over the years, there has been an interest in identifying dynamic systems using machine learning. These tools help, for example, in understanding and extracting information from experimental data and discovering model structures kill éaticos. Thus, this work aims to present the models obtained of the motor-generator bench using the sparse identification method of non-generator dynamics linear (SINDy), this method has been shown to be useful in the identification of non-linear dynamics, by assuming that the equations have only a few important terms that guide the dynamics. The SINDy approach solves a sparse regression problem, eliminating the terms whose coefficients are less than a limit. The components used in the Identification processes are based on the Threshold Least Square Optimizer (STLSq). Are The models obtained in the linear and non-linear experiment through the package are presented. PySINDy. The root squared error is calculated by the NRMSE metric for each of the models. At the end of the work it is seen that the SINDy method is able to identify the model of the motor-generator bench. Keywords: 1. Optimizer 2. SINDy 3. Non-linear identification 4. Motor-generator

Keywords: 1.Optimizer 2. SINDy 3. Non-linear identification 4.Motor-generator

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$y(k)$	Saída do sistema
$u(k)$	Entrada do sistema
$\mathcal{L}(.)$	Operador da transformada de Laplace
ξ	Vetor de parâmetros a se determinar
$\boldsymbol{e}$	Vetor de erros
$\boldsymbol{A}$	Matriz de regressão
$\|\| \cdot \|\|_2$	Norma quadrática
$J(\boldsymbol{\xi})$	Custo
$\boldsymbol{\Theta}_x$	Termos polinomiais entre as variáveis de estado
$\boldsymbol{\Theta}_u$	Termos associados exclusivamente às entradas
$\boldsymbol{\Theta}_{xu}$	Termos com fatores de produtos cruzados entre as variáveis de estado e as entradas
$\boldsymbol{\Xi}$	Matriz de coeficientes de funções candidatas para SINDy

1 INTRODUÇÃO

1.1 Motivação e Contexto

A modelagem matemática é a área do conhecimento que estuda maneiras de construir e implementar modelos matemáticos de sistemas reais. Sendo assim, a identificação é uma área de modelagem matemática que estuda técnicas que necessitam de pouco ou nenhum conhecimento prévio do sistema. A finalidade é descrever, com esses modelos, as relações de causa e efeito entre as variáveis de entrada e saída.

Com o avanços dos estudos de técnicas de identificação em conjunto com computadores mais robustos o uso de modelos matemáticos tem aumentado em praticamente todas as áreas do conhecimento. Além disso, houve um crescente interesse por representações não-lineares para caracterizar sistemas e fenômenos reais. Na medida em que as representações lineares são substituídas em algumas aplicações por seus correspondentes não lineares, torna-se possível analisar e reproduzir certos fenômenos e comportamentos dinâmicos mais complexos (AGUIRRE; RODRIGUES; JÁCOME, 1998).

Essas técnicas possuem a capacidade de permitir que os cientistas extraiam informações de conjuntos de dados numerosos e complexos. Eles geralmente são capazes de detectar relacionamentos e estruturas entre variáveis que seriam impossíveis para um ser humano identificar devido a natureza complexa destas relações. Pressuposto a isso, os sistemas dinâmicos e suas soluções oferecem uma imensa gama de técnicas para estudo teórico, dessa forma esses métodos de identificação podem ser vistos como complementares aos técnicas analíticas. Uma vez conhecidas as equações, essas podem ser aproveitadas para fazer previsões sobre o comportamento desses sistemas(SILVA, 2020).

A técnica de identificação é bastante utilizada para encontrar modelos matemáticos à partir de dados de entrada-saída. Esses modelos matemáticos identificados permitem obter uma boa aproximação do comportamento dinâmico do sistema de interesse. Não há um único paradigma que reproduza de forma exata o comportamento de um sistema de forma que não existe apenas um modelo para um sistema determinado, mas sim várias possibilidades de modelos cuja capacidade de representação variam. A escolha do modelo deve ser feita dependendo da aplicação, um modelo mais simples pode conseguir atender e ser mais adequado.

Uma técnica recente proposta para esta abordagem é a identificação esparsa de sistemas dinâmicos não lineares (SINDy)[1]. Esta abordagem se destaca pela possibilidade de identificação de modelos lineares e não lineares, contínuos e discretos, no espaço de estados. A técnica realiza tanto a seleção de um conjunto de termos candidatos, buscando aqueles que melhor se ajustem ao conjunto de dados, quanto a determinação dos parâmetros destes

[1]Sparse Identification of Nonlinear Dynamics

termos. A abordagem SINDy envolve a formulação de um problema de otimização que estabelece uma relação de compromisso entre estes dois objetivos.

Neste trabalho, é realizado o estudo da técnica SINDy visando a identificação de modelos dinâmicos lineares e não lineares no espaço de estados. O SINDy é usado inicialmente para identificar modelos de sistemas dinâmicos clássicos como o Lorenz, Duffing e Van der Pol, a partir de dados gerados computacionalmente. Em seguida a estratégia é então voltada para um sistema real da bancada motor-gerador do Laboratório de Sistemas de Controle, da Faculdade de Engenharia Elétrica da UFPA-Tucuruí. Este sistema exibe não linearidades do tipo zona-morta e ganho variável, e oferece uma possibilidade para o estudo das capacidades do SINDy quanto a identificação e seleção de modelos.

O trabalho faz uso extensivo do pacote PySINDy, um conjunto de métodos em Python que implementa funções para a criação de estruturas de modelos, a identificação e a simulação no espaço de estados. Para a bancada do motor-gerador foram colhidos dados da saída para realizar a identificação linear e não-linear e apresentados os resultados. Para a identificação não-linear foram simulados o modelo linear, e os modelos não-lineares com não linearidades polinomiais de segunda à quinta ordem. Ao final, ao analisar todos os resultados, foi possível mostrar que o método conseguiu identificar o com um bom grau de ajuste ao conjunto de dados medidos.

1.2 Justificativa

A modelagem utilizando leis físicas é limitada a situações onde se tem um bom conhecimento das regras gerais do sistema em estudo. Assim, a identificação se apresenta como uma boa estratégia para se criar modelos baseadas em dados, sem conhecimento prévio das relações dinâmicas entre as variáveis do sistema. Tipicamente, a identificação baseada em modelos lineares é bem conduzida utilizando estratégias clássicas de regressão, a exemplo da regressão por mínimos quadrados. No contexto de modelagem de sistemas lineares e invariantes no tempo (LIT) as estratégias de identificação apontam tradicionalmente para a determinação dos parâmetros de uma função de transferência.

Embora altamente útil em problemas de controle e processamento de sinais, as funções de transferência buscam determinar relações, no tempo e na frequência, entre uma única variável de entrada e uma outra de saída. Além disso, esta estrutura é utilizada apenas para se modelar sistemas LIT. Nesse contexto, a estratégia desenvolvida neste trabalho aponta para a identificação de modelos no espaço de estados, sejam eles lineares ou não lineares, o que abre uma possibilidade de aplicações em situações que demandem análise e controle não-linear.

Uma vantagem dos modelos aqui identificados é que a estrutura em esparso de estados permite a representação não linear, que é explorada a neste trabalho visando a identificação de modelos esparsos no espaço de estados. Isto por que, tipicamente, os modelos no espaço de estados costumam ter um número pequeno de termos, dai a necessidade de o

método de identificação embutir a propriedade da esparsidade dos parâmetros a serem determinados. Uma estratégia esparsa visa, portanto, encontrar modelos com bons ajustes e que sejam compactos em sua estrutura.

A otimização por regressão esparsa aparece como uma solução, quando são propostas uma grande quantidade de relações possíveis entre as variáveis de estado, prevendo termos lineares e não lineares, o que compõe uma biblioteca de termos candidatos. A estratégia de identificação atua determinando quais destes termos melhor explicam a dinâmica do sistema, e ainda, qual o valor do parâmetro de cada um dos termos residuais.

A esparsidade é atingida eliminando-se termos acima de um determinado limiar. Assim, espera-se que muitos dos termos iniciais na biblioteca de termos candidatos sejam eliminados, restando apenas aqueles mais representativos e que ofereçam melhor desempenho ao modelo, de acordo com o estabelecido na função objetivo do problema de otimização.

1.3 Metodologia

Esta estratégia será desenvolvida, inicialmente, pela sua formulação matemática, visando apresentar e definir o problema de otimização do SINDy. Isto será realizado a partir do estudo e revisão teórica da metodologia (BRUNTON; PROCTOR; KUTZ, 2016).

Assim, esta técnica é avaliada computacionalmente em linguagem Python para a identificação de um sistema não-linear a partir de dados gerados computacionalmente. O escolhido foi o clássico sistema de Lorenz que exibe não linearidades polinomiais e dinâmica caótica. O resultado demonstrou uma excelente capacidade do método SINDy em selecionar os termos mais adequados do sistema e também em determinar os valores dos coeficientes de forma bastante precisa. Embora, pequenos erros para este sistema desencadeiem erros grandes com o passar do tempo, o que é gerado devido a sensibilidade as condições iniciais ou ao valor dos parâmetros, propriedade bem conhecida de sistemas caóticos.

O trabalho prosseguiu com o estudo do pacote PySINDy, um conjunto de métodos computacionais em linguagem Python criado pelos desenvolvedores do SINDy (BRUNTON; PROCTOR; KUTZ, 2016). Neste caso, buscou-se estudar as funcionalidades do pacote por meio da reprodução da identificação de modelos de alguns sistemas dinâmicos clássicos, a exemplo do sistema autônomo de Duffing e do de Van der Pool.

A principal etapa deste trabalho diz respeito a avaliação do SINDy na identificação de modelos, lineares e não-lineares no espaço de estados, de uma bancada didática do Laboratório de Sistemas de Controle da Faculdade de engenharia elétrica da Universidade Federal do Pará-Campus Tucuruí. Trata-se de um sistema real chamado de motor-gerador por acoplar dois motores de corrente contínua pelo eixo de rotação. A tensão na armadura em um dos motores é tomada como o sinal de entrada, enquanto que a tensão induzida na armadura do segundo motor (gerador) representa a variável de saída.

Este sistema apresenta alguns desafios de modelagem que são estudados e apresentados no trabalho, como é o caso das não-linearidades por zona morta e o ganho variável. Foram realizados experimentos de bancada para a coleta de dados, buscando registrar sinais amostrados de entrada e saída, adquiridos por meio de uma interface de hardware com o microcontrolador Arduíno. Estes dados foram, então, repassados ao SINDy para a composição de termos candidatos estritamente polinomiais, buscando assim aplicar a estratégia de identificação. Foram realizados experimentos de identificação SINDy onde se demonstrou a boa representatividade de modelos lineares sobre pontos particulares de operação do sistema, identificados a partir de dados gerados por uma excitação com um sinal binário pseudo-aleatório (PRBS). Estes modelos apresentaram porém uma degradação do comportamento a medida que se afastavam deste ponto. Buscando recuperar modelos com representação mais abrangente do sistema, foram formulados então experimentos de identificação não linear no espaço de estados, a partir de dados gerados com um sinal multi-nível pseudo aleatório, visando atingir uma ampla faixa de excitação do sistema. Neste caso abriu-se um grande espaço de busca, no que diz respeito tanto a ordem dos modelos testados quanto ao limiar dos seus parâmetros.

Foram então identificados modelos polinomiais com não-linearidades de segunda até quinta ordem, e avaliados sobre uma faixa de valores para o limiar. De uma forma geral, modelos de ordem mais elevadas demonstraram melhor desempenho, exigindo porém uma quantidade muito maior de termos. Da mesma forma, modelos com limiares pequenos de valores paramétricos apresentaram melhor desempenho. Foi realizado então uma avaliação de compromisso entre a ordem e o limiar mais adequados para a identificação do modelo não-linear do sistema motor gerador. Pode-se dizer que o método SINDy, implementado pelo pacote PySINDy, ofereceu uma boa abordagem de identificação de modelos não lineares no espaço de estados.

1.4 Objetivos

Este trabalho tem como principal objetivo apresentar o método SINDy para identificação de modelos não-lineares esparsos no espaço de estados, por meio da avaliação em um sistema real de laboratório.

1.5 Objetivos específicos

- Formular a estratégia de identificação SINDy por meio de um problema de otimização;
- Apresentar a implementação computacional desta estratégia em linguagem python;
- Apresentar o pacote pySINDy, suas funcionalidades e desempenho na identificação de modelos, com dados computacionais, de sistemas dinâmicos clássicos no espaço de estados;
- Fazer o estudo experimental das não-linearidades do sistema real motor-gerador;

- Identificar modelos LIT locais, com dados reais, para o sistema motor-gerador;
- Identificar modelos não-lineares globais para a representação do comportamento dinâmico do sistema motor gerador.

1.6 Apresentação do Trabalho

O trabalho foi organizado em três capítulos. No Capítulo 2, serão apresentados tópicos sobre identificação de sistemas, apresentando as principais etapas para identificação de modelos matemáticos. Será apresentada, também, a metodologia utilizada para realizar a identificação pelo método SINDy proposta neste trabalho. Será apresentada também a identificação, utilizando o pacote PySINDy, de dois sistemas conhecidos, o sistema de Van der Pol e Duffing. No Capítulo 3 será apresentada a identificação linear e não linear do sistema real da bancada motor-gerador, utilizando a metodologia apresentada nesse trabalho. Serão apresentados e discutidos os resultados obtidos. Por fim, o Capítulo 4 será dedicado às considerações gerais a cerca do trabalho realizado e dos resultados obtidos.

2 IDENTIFICAÇÃO ESPARSA DE MODELOS NÃO-LINEARES

2.1 Modelagem e Identificação de Sistemas

O modelo matemático de um sistema é um conjunto de equações que descrevem o seu comportamento dinâmico ou estático, no tempo ou na frequência, de natureza contínua ou discreta. A obtenção de um modelo matemático deve estabelecer um compromisso entre a simplicidade do modelo e a precisão e na representação do fenômeno físico (OGATA, 2010). Os modelos ditos lineares e invariantes no tempo (LIT) possuem uma descrição matemática bem estabelecida, enquanto os modelos não-lineares fazem referência a uma ampla gama de comportamentos dinâmicos que buscam tratar das muitas possibilidades de comportamento da natureza.

Historicamente, modelos matemáticos têm sido desenvolvidos a partir do conhecimento da física do sistema, o que é conhecido como modelagem fenomenológica (AGUIRRE; RODRIGUES; JÁCOME, 1998). Esta abordagem exige o conhecimento detalhado do comportamento das diferentes partes constituintes de um sistema e das suas relações. Como exemplo, e sem perda de generalidade, considera-se um sistema mecânico clássico, o massa mola e amortecedor, descrito na Figura 2.1:

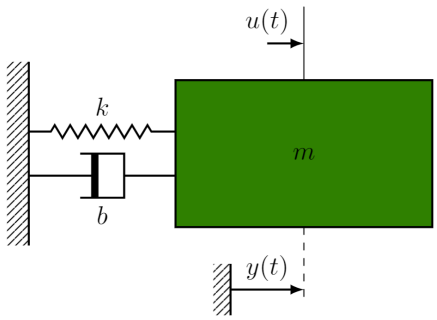

Figura 2.1 – Sistema massa mola amortecedor.

Neste sistema busca-se explicar o comportamento da posição $y(t)$ de uma massa m, que é ligada a uma parede fixa por meio de uma mola com constante elástica k e um amortecedor com coeficiente de amortecimento b, deviso a aplicação de uma força externa $u(t)$, sendo t o tempo contínuo. A aplicação da segunda lei de Newton, descrevendo as relações das forças sobre a massa resulta na equação diferencial ordinária (EDO) dada por:

$$m\frac{d^2y}{dt^2} + b\frac{dy}{dt} + ky(t) = u(t) \tag{2.1}$$

Esta é uma equação LIT cuja solução $y(t)$ resulta em uma função analítica que descreve o comportamento da massa.

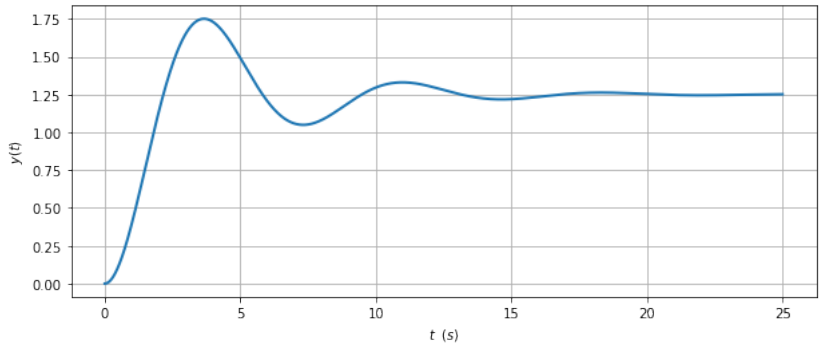

Figura 2.2 – Resposta subamortecida ao degrau do sistema massa, mola e amortecedor.

Equações LIT como estas são vastamente empregadas em ciências e em engenharia para descrever sistemas Lineares. Uma outra perspectiva deste modelo é tratá-lo algebricamente por uma função de transferência, definida a partir da razão entre transformada de Laplace $\mathcal{L}\{\cdot\}$ dos sinais de saída e entrada do sistema. Para o massa mola amortecedor isto resulta:

$$G(s) = \frac{1}{ms^2 + bs + k} \tag{2.2}$$

Funções de transferência oferecem uma possibilidade de análise de sistemas LIT no domínio da frequência, sendo no entanto restrita a sistemas com uma entrada e uma saída (SISO).

O código em linguagem Python abaixo apresenta a definição desta função utilizando o pacote *scipy* e implementação da resposta a uma entrada tipo degrau, para uma escolha particular dos parâmetros, visando um comportamento subamortecido. A Figura 2.2 apresenta a solução numérica da resposta.

```
1    from scipy import signal as sg
2    m, b, k = 1.0, 0.5, 1.2
3    G = sg.TransferFunction(1,[m, b, k])
4    t, y = sg.step(G)
```

Código 2.1 – Função de transferência e resposta ao degrau do sistema massa, mola amortecedor.

Esta abordagem de modelagem de um sistema linear pelo fenômeno encontra limitação justamente pela dificuldade de se conhecer as leis que governam sistemas mais complexos (a exemplo de sistemas econômicos, sociais, ambientais, ...) e mesmo sistemas de engenharia com uma estrutura que depende de muitas partes constitutivas, como um avião ou uma usina hidrelétrica, por exemplo.

Neste caso existe uma tendência de fugir desta abordagem tradicional e se lançar mão das técnicas de identificação de sistemas. A identificação de sistema trata do problema de construção de modelos matemáticos de sistemas dinâmicos com base em dados medidos do sistema (LJUNG, 1999). O objetivo da identificação é obter um modelo matemático que melhor descreve a relação entre os dados de entrada e saída do sistema de interesse, com a

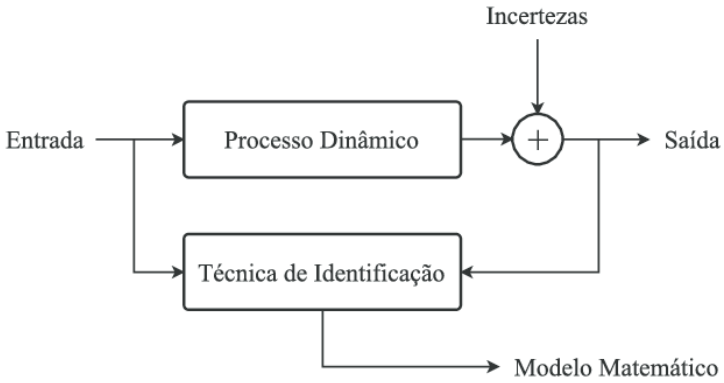

Figura 2.3 – Procedimentos para identificação de processos (COELHO, 2004)

precisão necessária, quanto mais precisas estas condições, mais próximo da realidade será o modelo. A Figura 2.3 ilustra no formato de um diagrama a ideia geral da determinação de um modelo matemático a partir do uso de alguma técnica de identificação sobre um processo dinâmico, neste caso os dados coletados dizem respeito à entrada e à saída.

Os procedimentos para a identificação de sistemas podem seguir cinco etapas:

1. Testes dinâmicos e coletas de dados;
2. Escolha da representação matemática a ser utilizada;
3. Determinação da estrutura do modelo;
4. Estimação dos parâmetros por regressão e otimização;
5. Validação do modelo.

Estas etapa podem ser observados no fluxograma na Figura 2.4, (AGUIRRE, 2004). O fluxograma aponta para o fato de que o processo de identificação é cíclico, podendo ser repetido buscando se determinar um modelo mais apropriado dentro de algum critério de escolha.

Os dados de entrada-saída são coletados na forma de forma amostrada, visando preservar o máximo de informações representativas do sistema, já que os resultados dos processos seguintes dependerão da qualidade e quantidade das amostras. Neste caso, existe uma preocupação na escolha na natureza do sinal de excitação, sua amplitude de excursão, a banda de frequência, o período de amostragem T_s da coleta (LJUNG, 1999).

No caso, sinais fundamentais como degraus, senoides ou exponenciais podem ser utilizados apenas em casos muito restritos, dada o baixo nível de excitação que geram no sistema. A preferência é por sinais que possam garantir a persistência de excitação, ou seja a possibilidade de o sistema ser ativado em diferentes faixas de amplitude e de frequência, de modo que todos os seus modos característicos se expressem no sinal de saída (LJUNG, 1999).

As sequências binárias pseudo-aleatórios - PRBS [1] e as sequências multi-níveis

[1] Pseudo Random Binary Sequence

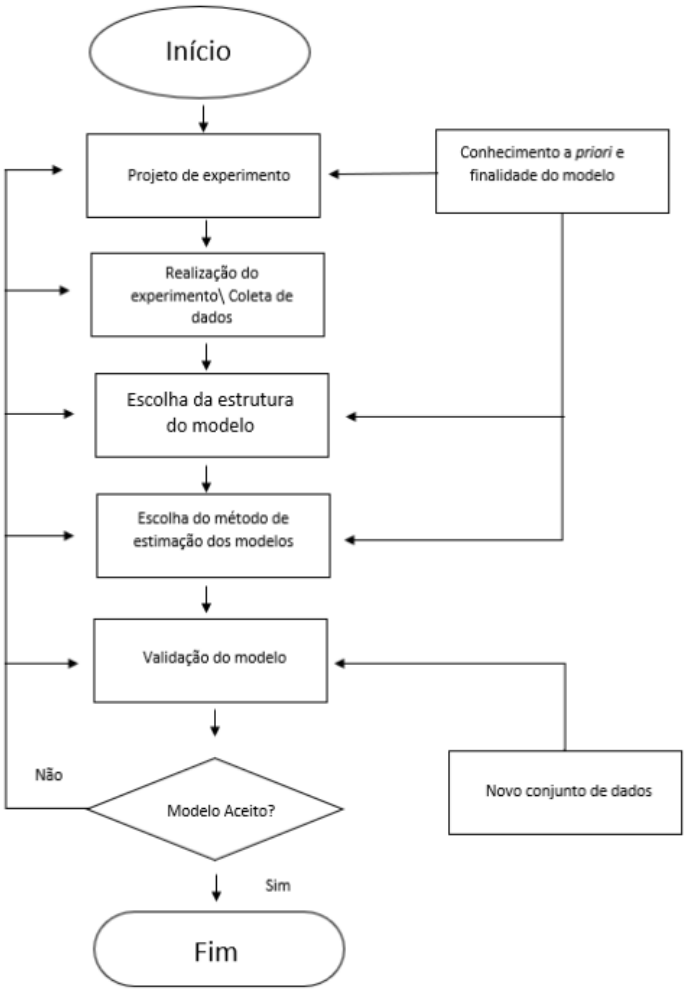

Figura 2.4 – Fluxograma de decisão para identificação (SÖDERSTRÖM; STOICA, 1989)

pseudo-aleatórios - PRMLS [2] são sinais que possibilitam a persistência de excitação. Para sistemas lineares, o PRBS é frequentemente usado (SÖDERSTRÖM; STOICA, 1989) dada a sua alternância entre duas amplitudes fixas. Para sistemas não lineares a alternância entre duas amplitudes pode não capturar o comportamento não linear (NELLES, 2001), particularmente em um sistema com não linearidade de ganho variável, dessa forma é necessário utilizar o sinal PRMLS onde a amplitude de um PRBS convencional é variada aleatoriamente, capturando assim as características não lineares do sistema.

A determinação da taxa de amostragem é um aspecto importante no desenvolvimento de sinais para identificação de sistemas, uma vez que é uma técnica de modelagem para controle de processos na qual modelos dinâmicos são construídos através de dados observados do sistema (SÖDERSTRÖM; STOICA, 1989). Para se obter um modelo que descreva o comportamento do sistema adequadamente é necessário que o sistema seja amostrado com uma taxa de amostragem apropriada e dessa forma gere resultados permitam a construção do modelo. Outro aspecto esta relacionado a escolha do tempo de amostragem já que tempos de amostragem diferentes produzem modelos diferentes.

[2] Pseudorandom Multilevel Sequence

Nesta etapa busca-se determinar os parâmetros para a estrutura escolhida de modo a encontrar a forma mais aproximada de representar o comportamento dinâmico do sistema. A etapa de estimação resulta na determinação dos valores numéricos dos parâmetros (AGUIRRE, 2004). Em sistemas lineares o método mais utilizado é o dos mínimos quadrados.

2.2 Identificação pelo Método dos Mínimos Quadrados

Proposta inicialmente por Karl Friedrich Gauss em 1795, a teoria dos mínimos quadrados foi utilizada para estimar o movimento orbital dos planetas a partir de medições telescópicas (WELLS D. E. KRAKIWSKY, 1997). A partir disso este método tornou-se a principal ferramenta para a obtenção de parâmetros a partir de dados experimentais, sendo o mais conhecido e mais utilizado entre engenheiros e cientistas (BRANDOLTA, 2002).

Na formulação do método para o contexto de identificação de sistemas dinâmicos parte-se de um conjunto de N amostras dos dados de entrada e saída[3]:

$$\boldsymbol{u} = [u_1 \; u_2 \; \ldots \; u_N]^T$$
$$\boldsymbol{y} = [y_1 \; y_2 \; \ldots \; y_N]^T$$

Presume-se que exista uma relação funcional e dinâmica entre as duas grandezas. Entre as muitas formas possíveis será considerada, sem perda de generalidade, a forma LIT auto-regressiva com entrada (ARX), com a equação de diferença $y(n) = \mathcal{H}\{\boldsymbol{u}, \boldsymbol{y}\}$, ou seja, o operador $\mathcal{H}$ expressa combinações lineares de atrasos de u e de y, o que pode ser representado pela equação de diferença:

$$y(n) = \mathcal{H}\{\boldsymbol{u}, \boldsymbol{y}\} = \sum_{l=1}^{l_y} a_l y(n-l) + \sum_{l=0}^{l_u} b_l u(n-l) \tag{2.3}$$

Sendo $\boldsymbol{\xi} = [a_1 \;\; a_2 \;\; ... \;\; a_{l_y} \;\; b_0 \;\; b_1 \;\; ... \;\; b_{l_u}]^T$ os coeficientes que parametrizam $\mathcal{H}$.

A identificação visa então determinar $\boldsymbol{\xi}$ de modo que $\mathcal{H}$ represente com alguma precisão a relação entre os dados do sistema. A escolha de $\boldsymbol{\xi}$ deve ser baseada então em algum critério de ajuste dos dados. Isso é determinado ao se incorporar os dados a estrutura do modelo com o fato de que para a $i-$ésima amostra existirá um erro e_i entre o modelo e o conjunto de dados, o que pode ser representado por:

$$y_i + e_i = \mathcal{H}\{\boldsymbol{u}_i, \boldsymbol{y}_i\} \tag{2.4}$$

[3]O sobre-escrito T indica transposição matricial.

Agora, aplicando N amostra de dados nesta estrutura resulta :

$$\begin{aligned} a_1y_0 + a_2y_{-1} + \ldots + b_0u_1 + b_1u_0 + \ldots &= y_1 + e_1 \\ a_1y_1 + a_2y_0 \ + \ldots + b_0u_2 + b_1u_1 + \ldots &= y_2 + e_2 \\ \vdots \qquad\qquad = \quad \vdots & \\ a_1y_{N-1} + a_2y_{N-2} + \ldots + b_0u_N + b_1u_{N-1} + \ldots &= y_N + e_N \end{aligned}$$

O resultado é um sistema de equações lineares, que pode ser expresso na forma vetorial matricial:

$$\begin{bmatrix} y_0 & y_{-1} & \ldots & u_1 & u_0 & \ldots \\ y_1 & y_1 & \ldots & u_2 & u_1 & \ldots \\ . & . & \vdots & . & . & \vdots \\ y_{N-1} & y_{N-2} & \ldots & u_N & u_{N-1} & \ldots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ b_0 \\ b_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} \tag{2.5}$$

Ou seja:

$$\boldsymbol{A\xi} = \boldsymbol{y} + \boldsymbol{e}, \tag{2.6}$$

sendo

$$\boldsymbol{A} = \begin{bmatrix} y_0 & y_{-1} & \ldots & u_1 & u_0 & \ldots \\ y_1 & y_1 & \ldots & u_2 & u_1 & \ldots \\ . & . & \vdots & . & . & \vdots \\ y_{N-1} & y_{N-2} & \ldots & u_N & u_{N-1} & \ldots \end{bmatrix} \tag{2.7}$$

a matriz de regressão, cujas colunas são compostas unicamente a partir de medições deslocadas dos sinais de entrada e saída; $\boldsymbol{e}$ é o vetor de erros e $\boldsymbol{\xi}$ o vetor de parâmetros a se determinar.

O problema de identificação pede então que se determine $\boldsymbol{\xi}$ que gere o menor erro possível sobre o conjunto de dados. O erro é dado a partir de 2.6 como:

$$\boldsymbol{e} = \boldsymbol{A\xi} - \boldsymbol{y} \tag{2.8}$$

No caso, é conveniente estabelecer uma métrica sobre $\boldsymbol{e}$. Dentre as muitas possibilidades, uma das mais comuns na estatística é a medida do erro quadrático dada por:

$$J_{mq}(\boldsymbol{\xi}) = \sum_{i=1}^{N} {e_i}^2 = \boldsymbol{e}^T\boldsymbol{e} \tag{2.9}$$

Um escalar dado pela soma dos quadrados dos erros. Ou ainda, aplicando 2.8 resulta:

$$J_{mq}(\boldsymbol{\xi}) = (\boldsymbol{A\xi} - \boldsymbol{y})^T(\boldsymbol{A\xi} - \boldsymbol{y}) \tag{2.10}$$

A medida do erro $J_{mq}(\cdot)$ está associada então a uma função, denotada por função custo, que depende dos parâmetros do sistema, de modo que cada escolha distinta do conjunto de parâmetros possui um erro diferente. Pode-se buscar por $\boldsymbol{\xi}$ utilizando a ideia de se determinar o mínimo desta função, ou seja:

$$\frac{\partial J_{mq}}{\partial \boldsymbol{\xi}} = 0 \tag{2.11}$$

Isto pode ser conduzido expandindo 2.10, o que resulta:

$$J_{mq}(\boldsymbol{\xi}) = \boldsymbol{\xi}^T\boldsymbol{A}^T\boldsymbol{A\xi} - \boldsymbol{\xi}^T\boldsymbol{A}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{A\xi} - \boldsymbol{y}^T\boldsymbol{y} \tag{2.12}$$

Aplicando o gradiente, resulta:

$$\frac{\partial J_{mq}}{\partial \boldsymbol{\xi}} = 2\boldsymbol{A}^T\boldsymbol{A\xi} - 2\boldsymbol{A}^T\boldsymbol{y} = 0 \tag{2.13}$$

É possível isolar $\boldsymbol{\xi}$ considerando a pré-multiplicação pela pseudo-inversa $(\boldsymbol{A}^T\boldsymbol{A})^{-1}$ de A, o que resulta no clássico resultado analítico de minimização quadrática:

$$\boldsymbol{\xi} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y} \tag{2.14}$$

Esta solução analítica pode ser bem conduzida, porém a abordagem por otimização permite lançar um outro olhar sobre o problema. Neste caso, retoma-se a função custo, e redefine-se o problema na forma

$$\boldsymbol{\xi} = \underset{\boldsymbol{\xi'}}{\operatorname{argmin}} \ ||\boldsymbol{A\xi'} - \boldsymbol{y}||_2 \tag{2.15}$$

O problema de otimização é assim definido tendo como argumento $\boldsymbol{\xi'}$, a busca numérica é pelo $\boldsymbol{\xi}$ ótimo que minimiza a norma quadrática $||\cdot||_2$ do custo $J(\boldsymbol{\xi})$.

2.2.1 Exemplo computacional de identificação LIT por função de transferência.

O exemplo a seguir ilustra os procedimentos computacionais para a identificação de sistemas por função de transferência para o sistema massa mola amortecedor.

Inicialmente foi realizada a simulação do sistema visando a geração de dados para uma entrada PRBS. O código a seguir apresenta a geração destes sinais utilizando métodos do pacote Python-Scipy:

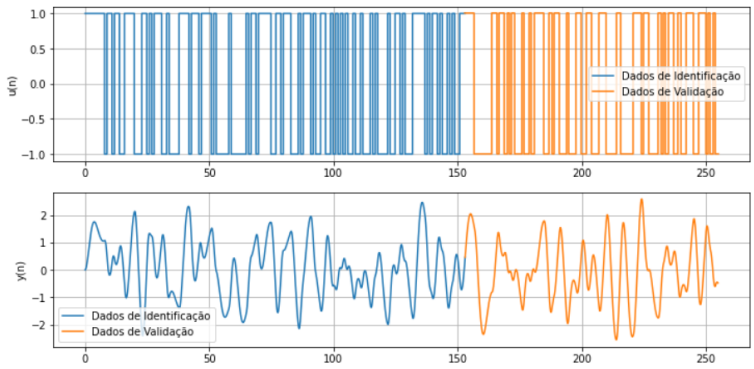

Figura 2.5 – Dados de entrada e saída do sistema massa mola amortecedor, divididos em dados de identificação (azul) e validação (amarela).

```
u = 2*(sg.max_len_seq(8)[0]-0.5)
Ts = 0.2
t = Ts*np.arange(0,N)
y = np.squeeze(sg.lsim(G, u, T = t)[1])
```

Código 2.2 – Geração de dados de entrada $u(n)$ e saída $y(n)$ para o sistema massa mola amortecedor com entrada PRBS.

A Figura 2.5 apresenta os dados gerados, os quais foram divididos em duas partes, 60% para identificação e 40% para validação.

O passo seguinte pede que se estabeleça uma estrutura para o modelo. Considerando a ordem do modelo contínuo, foi adotada uma representação ARX de segunda ordem, ou seja $l_y = l_u = 2$, a equação de diferença recai:

$$y(n) = a_1 y(n-1) + a_2 y(n-2) + b_0 u(n) + b_1 u(n-1) + b_2 u(n-2) \tag{2.16}$$

Neste caso, aplicando a transformada $\mathcal{Z}\{\cdot\}$ e isolando a razão $Y(z)/U(z)$ determina-se a função de transferência discreta para este modelo, na forma:

$$G(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{2.17}$$

O que permite a representação e análise do sistema no domínio da frequência.

Procedeu-se então com a identificação criando-se a partir dos dados a matriz de regressão para o modelo proposto, a qual terá 5 colunas, cada uma associada a um parâmetro do modelo:

```

na, nb = 2, 3
ni = np.arange(na,Ni+na)
A = np.zeros((Ni,na+nb))

# Para regressores de y:
for l in np.arange(0,na):
```

```
8       A[:,l] = y[ni-l-1]
9
10    # Para regressores de u:
11    for l in np.arange(0,nb):
12      M[:,na+l] = u[ni-l]
```

Código 2.3 – Geração da matriz de regressão.

A implementação analítica do algoritmo de mínimos quadrados pode ser implementada com uma única linha de código em Python-numpy, que determina o vetor $\boldsymbol{\xi}_a$ analítico.

```
1     xia = np.linalg.inv(A.T@A)@A.T@y
```

Código 2.4 – Aplicação do algoritmo de mínimos quadrados.

De modo equivalente foi implementada a solução por otimização. Neste caso inicialmente foi definida a função custo, dada por:

```
1     def Jmq(xi,dados):
2         A, y = dados
3         J = np.sum(np.power(np.abs(A@xi-y),2))
4         return J
```

Código 2.5 – Função custo $Jmq(\xi)$ definida em Python.

A qual foi chamada pelo método de otimização fmin, do pacote scipy.optimize, visando a determinação do ponto ótimo. O código abaixo apresenta a implementação da estratégia, que parte de um ponto inicial para determinar o vetor $\boldsymbol{\xi}_o$ por otimização:

```
1     dados = (A,y)
2     xi0 = np.array([1,1,1,1,1]) # Ponto inicial.
3     xio = scipy.optimize.fmin(Jmq,xi0,args=(dados,));
```

Código 2.6 – Implementação da função fmin() para minimização da função custo $J_{mq}()$.

A solução dos dois métodos é apresentada a seguir:

```
1     print('xia: ', xia)
2     print('xio: ', xio)
3
4   xia:  [ 1.8744665  -0.90483742  0.00649297  0.02529453  0.00617615]
5   xio:  [ 1.87446567 -0.90483691  0.00649324  0.02529457  0.0061762 ]
```

Código 2.7 – Resultado de identificação minimo quadrática analítica e por otimização.

As soluções mostram que os dois métodos conduzem a resultados numéricos praticamente idênticos, com diferenças se expressando a partir da quinta casa decimal, que pode ser ajustado ao se impor menores tolerâncias ao método de otimização.

Como etapa final do processo foi realizada a validação deste modelo, inicialmente tomando-se os parâmetros do sistema a partir do vetor e então definindo a função de transferência discreta do modelo, o que é dado por:

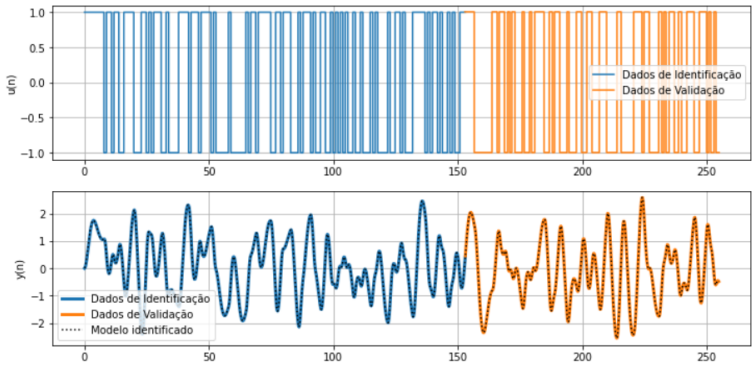

Figura 2.6 – Resultado de simulação do modelo identificado, confrontado com os dados de identificação e validação.

```
    a1, a2, b0, b1, b2 = xia
    Ba = [b0 , b1, b2]
    Aa = [1, -a1, -a2]

    Gi = sg.TransferFunction(Ba,Aa, dt = Ts)

    # Resposta do modelo identificado:
    yp = np.squeeze(sg.dlsim(Gi, u, t = t)[1])
```

Código 2.8 – Função de transferência a partir do modelo identificado e simulação.

Em seguida o modelo foi simulado para todo o conjunto de dados, tanto de identificação quanto de validação. O resultado é apresentado na Figura 2.6

Como não foi aplicado nenhum ruído aos dados e o erro é nulo, o resultado do modelo identificado é de grande exatidão numérica.

A estratégia de minimização quadrática mostra-se assim extremamente apropriada para os propósitos de identificação, sendo aplicada a diferentes tipos de modelos que são lineares nos parâmetros. A estratégia encontra valores paramétricos precisos. Porém destaca-se que o método encontra uma solução que prevê valores para todos os termos candidatos do modelo. Ou seja, o método não inclui nenhuma estratégia para eliminar termos que porventura não contribuam para a representação dinâmica.

Este fato é relevante quando se tem em perspectiva a identificação de modelos que partem de estruturas com um grande número de termos, que é o caso particular de um modelo não linear no espaço de estados. Neste caso é desejável incluir na estratégia a possibilidade de eliminação de termos, visando um modelo final mais compacto, ou de modo equivalente, com estrutura esparsa.

2.3 Identificação de modelos no espaço de estados - Abordagem SINDy

A representação de interesse deste trabalho é em espaço de estados. Esta classe de modelos no domínio do tempo baseia-se em um sistema de equações dinâmicas de primeira ordem de um conjunto de variáveis básicas para explicar o comportamento de um sistema, os estados. Em sua formulação mais genérica, um modelo em espaço de estados pode ser expresso como:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \tag{2.18}$$

Onde $\boldsymbol{x} = [\ x_1\ x_2\ \ldots\ x_n\ ]^T$ é o vetor de variáveis de estado e $\boldsymbol{u} = [\ u_1\ u_2\ \ldots\ u_m\ ]^T$ é o vetor das entradas. O conjunto $\boldsymbol{f} = [\ f_1\ f_2\ \ldots\ f_n\ ]^T$ agrega funções de estado, uma para cada variável, n é a ordem do modelo.

A determinação deste modelos costuma ocorrer sob um processo baseado em modelagem fenomenológica, onde se parte das leis físicas do sistema sob estudo, determina-se uma equação diferencial de ordem n, e então esta equação é quebrada em n outras de primeira ordem, cada uma associada a uma variável de estado, gerando um sistema de equações.

A estrutura do modelo permite que a equação de cada uma das variáveis, seja expressa em função do vetor de estados (todas as variáveis de estado) e das entradas, não havendo restrições para as possibilidades destas relações que podem ser lineares ou não lineares, dependentes ou independentes do tempo. Além disso, o modelo pode ser de múltiplas entradas e de múltiplas saídas, sendo as saídas alguma função estática das saídas e das entradas.

A título de exemplo, considera-se a EDO do sistema massa-mola-amortecedor, dada por 2.1, onde é adota-se como variáveis de estado a posição $x_1(t) = y(t)$ e a velocidade $x_2(t) = y'(t)$ da massa. Assim, pode-se escrever as equações das derivadas de primeira ordem destas variáveis, respeitando a estrutura do modelo, o que recai na forma:

$$\dot{x}_1(t) = x_2(t) \tag{2.19}$$

$$\dot{x}_2(t) = -\frac{k}{m}x_1(t) - \frac{b}{m}x_2(t) + \frac{1}{m}u(t) \tag{2.20}$$

Esta estrutura LIT permite que as variáveis de estado sejam isoladas em um vetor de estados $\boldsymbol{x} = [x_1\ x_2]^T$, o que permite escrever 2.19 na forma canônica geral de um modelo LIT em espaço de estados:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{2.21}$$

sendo $\boldsymbol{A}$ e $\boldsymbol{B}$ as matrizes de estado e de entrada do sistema. A dependência de t está implícita.

Para o massa mola amortecedor 2.19 recai em:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \boldsymbol{u} \tag{2.22}$$

Estes modelos são amplamente utilizados para a análise, simulação e projeto de controladores. De fato, a dita teoria de controle moderno é fortemente baseada em estruturas no espaço de estados. No escopo deste trabalho busca-se identificar modelos diretamente no espaço de estados. Isto é relevante pois aponta para outras possibilidades dos métodos de identificação, para além dos utilizados para funções de transferência. Além disso, como a representação de estados admite termos não lineares busca-se uma formulação que se abra a esta possibilidade.

Deve-se destacar que tipicamente, os modelos de espaço de estados obtidos a partir da modelagem fenomenológica costumam ser compactos em sua estrutura, o que pode ser relatado como uma representação esparsa. Uma estratégia de identificação que busque determinar estes modelos a partir dos dados das variáveis deve então buscar determinar: (i) os termos mais relevantes que relacionam as variáveis, (ii) os parâmetros que ponderam estes termos e (iii) uma estratégia de eliminação de termos que não agregam na precisão do modelo. Uma proposição de solução recente para estas restrições é dada por (BRUTON; KUTZ, 2019) com o método SINDy.

2.3.1 Identificação SINDy

A estratégia SINDy busca obter uma aproximação para cada função de estado a partir de medições reais das entradas, dos estados e possivelmente de suas derivadas. O pressuposto é que estas funções possuem uma quantidade pequena de termos ativos, dai a técnica ser denominada esparsa. A forma geral para a aproximação de cada função proposta na estrategia é (BRUTON; KUTZ, 2019):

$$f(\boldsymbol{x},\boldsymbol{u}) \approx \sum_k \theta_k(\boldsymbol{x},\boldsymbol{u})\xi_k, \tag{2.23}$$

ou seja, um somatório de termos $\theta_k(\boldsymbol{x},\boldsymbol{u})$, cada um ponderado por um parâmetro ξ_k. Expressando esta relação na forma vetorial têm-se

$$f(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{\Theta}(\boldsymbol{x},\boldsymbol{u})\boldsymbol{\xi} \tag{2.24}$$

, sendo $\boldsymbol{\xi} = [\ \xi_1,\ \xi_2,\ \ldots]^T$ o vetor de parâmetros da função a ser determinado por uma estratégia de regressão. Estes parâmetros ponderam os termos candidatos $\boldsymbol{\Theta}(\boldsymbol{x},\boldsymbol{u}) = [\ \theta_1,\ \theta_2,\ \ldots]$ que são dados por relações lineares ou não lineares entre as variáveis de estado e as entradas.

Visando compor um cenário mais genérico, com um conjunto de funções de estado que se busca determinar, pode-se tomar as medições dos estados e das entradas, na forma.

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_n \\ | & | & & | \end{bmatrix} \tag{2.25}$$

e

$$\boldsymbol{U} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_m \\ | & | & & | \end{bmatrix} \tag{2.26}$$

para então se compor uma biblioteca de termos candidatos $\boldsymbol{\Theta}(\boldsymbol{X},\boldsymbol{U})$[4]. Estes termos podem ser compostos por fatores trigonométricos (Fourier), exponenciais e outros, as possibilidades são ilimitadas. No escopo deste trabalho serão tomados termos exclusivamente polinomiais, visando aproveitar a simplicidade estrutural e a grande versatilidade de representação de formas expressa pelos polinômios. Assim, a biblioteca de termos candidatos adotada é dada por:

$$\boldsymbol{\Theta}(\boldsymbol{X},\boldsymbol{U}) = \begin{bmatrix} \boldsymbol{\Theta}_x & \boldsymbol{\Theta}_u & \boldsymbol{\Theta}_{xu} \end{bmatrix}, \tag{2.27}$$

onde, $\boldsymbol{\Theta}_x$ representa os termos polinomiais entre as variáveis de estado:

$$\boldsymbol{\Theta}_x = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{X} & \boldsymbol{X}^2 & \dots & \boldsymbol{X}_x^d, \end{bmatrix} \tag{2.28}$$

$\boldsymbol{\Theta}_u$ representa os termos associados exclusivamente às entradas:

$$\boldsymbol{\Theta}_u = \begin{bmatrix} \boldsymbol{U} & \boldsymbol{U}^2 & \dots & \boldsymbol{U}^d \end{bmatrix} \tag{2.29}$$

e $\boldsymbol{\Theta}_{xu}$ a matriz de termos com fatores de produtos cruzados entre as variáveis de estado e as entradas:

$$\boldsymbol{\Theta}_{xu} = \begin{bmatrix} \boldsymbol{XU} & \boldsymbol{X}^2\boldsymbol{U} & \boldsymbol{XU}^2 & \boldsymbol{X}^2\boldsymbol{U}^2 & \dots & \boldsymbol{X}^{d_x}\boldsymbol{U}^{d_u} \end{bmatrix}. \tag{2.30}$$

Atente-se que o grau máximo adotado para os termos polinomiais é d e que, portanto, $d_x + d_u \leq d$. Outro detalhe é que a notação de relação polinomial entre as matrizes expressa um produto cruzado entre as suas colunas, por exemplo $\boldsymbol{XU}$ é uma matriz onde cada coluna é o produto cruzado entre as colunas de $\boldsymbol{X}$ com as de $\boldsymbol{U}$ e não o produto *matricial* de $\boldsymbol{X}$ com $\boldsymbol{U}$.

O conjunto completo de equações necessita arranjar também todos os parâmetros dos termos candidatos, neste caso será denotado que a i-ésima função será parametrizada por $\boldsymbol{\xi}_i$, de modo que todos os parâmetros possam ser agrupados como:

[4]Cada coluna de $\boldsymbol{X}$ ou $\boldsymbol{U}$ é uma medição temporal com N amostras de cada variável

$$\boldsymbol{\Xi} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\xi}_1 & \boldsymbol{\xi}_2 & \dots & \boldsymbol{\xi}_n \\ | & | & & | \end{bmatrix} \tag{2.31}$$

A estratégia de regressão necessita criar um sistema de equações para se buscar $\boldsymbol{\Xi}$, o qual é baseado no conjunto de dados coletados da dinâmica do sistema. Para que se possa estruturar o sistema na forma vetorial matricial é necessário ainda compor o lado esquerdo das equações, ou seja, o conjunto de medidas das derivadas das variáveis de estado, as quais são dadas por:

$$\dot{\boldsymbol{X}} = \begin{bmatrix} | & | & & | \\ \dot{\boldsymbol{x}}_1 & \dot{\boldsymbol{x}}_2 & \dots & \dot{\boldsymbol{x}}_m \\ | & | & & | \end{bmatrix} \tag{2.32}$$

Esta última necessidade, dependência das derivadas dos sinais, pode constituir a priori uma limitação. Contudo é possível determinar $\dot{X}$ diretamente a partir das medições X por métodos numéricos apropriados, de modo que isto não limita a abordagem.

Pode-se então se valer da estrutura geral para escrever:

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{\Xi} \tag{2.33}$$

Trata-se de um sistema de equações lineares em $\boldsymbol{\Xi}$, onde cada uma das suas colunas constitui um vetor de coeficientes dos termos candidatos em $\boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})$ para gerar $\dot{\boldsymbol{X}}$. Esta estrutura pode ser compreendida como um arranjo de sistemas de equações lineares em que cada coluna $\dot{\boldsymbol{x}}_k$ de $\dot{\boldsymbol{X}}$ é dada pelas por uma combinação linear $\boldsymbol{\xi}_k$ das colunas de $\boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})$. O desejável é encontrar uma solução em que apenas alguns termos de $\boldsymbol{\xi}_k$ sejam não nulos, o que implicaria em uma solução esparsa para cada função.

A necessidade de eliminar alguns termos entre os candidatos pode ser considerada a partir de uma medida dos próprios valores absolutos dos valores dos coeficientes destes termos, ou simplesmente $||\boldsymbol{\xi}'_k||_1$ onde o subíndice $_1$ refere-se a norma 1, que mede um vetor a partir do valor absoluto de seus termos.

A perspectiva aqui é claramente a de se propor um problema de otimização onde a função custo consiga expressar as duas demandas: (i) A determinação do vetor de parâmetros $\boldsymbol{\xi}$ o que é atingido, como se mostrou, incorporando na função custo uma medida do erro quadrático do modelo; (ii) Um segundo termo associado a medida da magnitude do vetor de pesos.

Busca-se então determinar $\boldsymbol{\Xi}$ por estratégias de regressão que levem a modelos com poucos termos e que conduzam a um bom ajuste entre os dados. Este é o tipico

cenário de regressão esparsa regularizada pela norma l_1, que penaliza a magnitude dos elementos de $\boldsymbol{\Xi}$, cujo problema de otimização pode ser formulado por:

$$\boldsymbol{\Xi} = \underset{\boldsymbol{\Xi}'}{\operatorname{argmin}} \ ||\dot{\boldsymbol{X}} - \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{\Xi}'||_2 \ + \ \lambda||\boldsymbol{\Xi}'||_1 \tag{2.34}$$

o parâmetro λ funciona como um parâmetro de regularização do problema. Esta função custo é bem conhecida no contexto de estatística, compondo um problema de regressão conhecido como LASSO[5].

Uma variação deste problema de otimização para o caso proposto é o de substituir a determinação de $\boldsymbol{\Xi}$ pela determinação de suas colunas individualmente, o que é conduzida por algoritmos como o algoritmo de mínimos quadrados com limite sequencial - STLS[6], para o qual a função custo pode ser então definida como:

$$\boldsymbol{\xi}_k = \underset{\boldsymbol{\xi}'_k}{\operatorname{argmin}} \ ||\dot{\boldsymbol{x}}_k - \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{\xi}'_k||_2 \ + \ \lambda||\boldsymbol{\xi}'_k||_1 \tag{2.35}$$

Este algoritmo inclui na sua rotina um limiar τ (threshold) para os parâmetros do sistema como uma restrição adicional, de modo que o resultado da otimização $\boldsymbol{\xi} > \tau$. Este procedimento tem se mostrado importante como um elemento adicional visando a esparsidade de modelos no espaço de estados determinado por SINDy, de modo que o STLSQ é seu algoritmo de otimização padrão.

A implementação computacional desta estratégia exige portanto a presença de um laço de repetição, onde o problema 2.34 é resolvido partindo da solução de n problemas na forma 2.35, um para cada variável de estado. Esta estratégia tem conduzido a resultados interessantes no problema de identificação de modelos por espaço de estados, conforme será demonstrado.

Um aspecto importante sobre o desenvolvimento aqui apresentado para o SINDy é fato de se incluir, no contexto deste trabalho, termos lineares e não-lineares dependentes da entrada, uma extensão no desenvolvimento apresentado por (BRUTON; KUTZ, 2019) que trata apenas de sistemas autônomos, e portando dependentes apenas dos estados. A principal motivação para o desenvolvimento aqui realizado é a perspectiva da utilização destes modelos em trabalhos futuros no contexto de sistemas de controle.

Outro ponto a se considerar é que a formulação desenvolvida tratou exclusivamente de modelos contínuos, que requerem informações dos estados e de suas derivadas. A formulação para tempo discreto necessita, como se sabe, apenas de valores deslocados de uma amostra dos sinais medidos dos estados. Considera-se que a formulação de tempo discreto pode ser diretamente conduzida a partir das ideias aqui apresentadas.

[5]Least Absolute Shrinkage and Selection Operator
[6]Sequential thresholded Least-Squares

2.4 Implementação computacional do método SINDy: O Sistema de Lorenz

Esta seção irá implementar, a titulo de exemplificação, o método SINDy em um sistema dinâmico não linear, com uma estrutura simples, e amplamente conhecido, o sistema de Lorenz. O modelo matemático de Lorenz exibe como efeito da não linearidade um comportamento caótico. É composto por três equações diferenciais ordinárias autônomas de primeira ordem, com três variáveis $x(t)$, $y(t)$ e $z(t)$ e três parâmetros σ, ρ e β, que são números reais positivos, sendo:

$$\dot{x} = -\sigma(x + y) \tag{2.36}$$

$$\dot{y} = x(\rho - z) - y \tag{2.37}$$

$$\dot{z} = xy - \beta z \tag{2.38}$$

O caos é atingido para alguns valores de parâmetros, como é o caso de $\sigma = 10$, $\rho = 28$ e $\beta = 8/3$. Em Python o modelo é simulado da seguinte forma:

```
# Parametros do sistema Lorenz
  dt, T = 0.01, 50
  t = np.arange(dt,T+dt,dt)
  beta = 8/3
  sigma = 10
  rho = 28

# Dinamica do sistema Lorenz:
def Lorenz(v, t0, sigma=sigma, beta=beta, rho=rho):
  x, y, z = v
  dx = sigma*(y - x)
  dy =  x * (rho - z) - y
  dz = x*y - beta*z
return [dx,dy,dz]
```

Código 2.9 – Codificação do modelo de Lorenz em Python.

Para obter a solução de Lorenz é realizada a integração de 3.1 3.2 e 3.3,utilizando a função scipy.integrate.odeint esta função integra numericamente um sistema de equações diferenciais ordinárias. Os dados foram gerados considerando a identificação do modelo contínuo, portanto houve a necessidade de se medir os estados e as suas derivadas. O Código 2.10 implementa a ideia.

```
  x0 = (-8,8,27)
  x = integrate.odeint(Lorenz, x0, t)
  # Derivada
  dx = np.zeros_like(x)
  for j in range(len(t)):
    dx[j,:] = Lorenz(x[j,:],0,sigma,beta,rho)
```

Código 2.10 – Integração do modelo de Lorenz visando gerar dados para identificação SINDy.

De posse dos dados medidos para os estados e suas derivadas procedeu-se então com a formulação da biblioteca de termos candidatos, do tipo polinomiais de até terceira ordem entre estas variáveis, o que gerou a matriz *Theta* no código abaixo. Este código apresenta então a implementação do algoritmo STLSQ, em uma rotina de repetição a que faz uso do algoritmo de mínimização quadrática da biblioteca numpy.

```
def sparsifyDynamics(Theta,dXdt,tau,n):
 # Determina condicao inicial para Xi
 Xi = np.linalg.lstsq(Theta,dXdt,rcond=None)[0]

 for k in range(10):
     indiceP = np.abs(Xi) < tau # Valores menores que o limiar
     Xi[indiceP] = 0             # Zera valores abaixo do limiar

      for ind in range(n):         # Um problema para cada variavel
        indiceG = indiceP[:,ind] == 0
        # Regressao epenas com indices nao nulos
        Xi[indiceG,ind] = np.linalg.lstsq(Theta[:,indiceG],dXdt[:,ind
],rcond=None)[0]
 return Xi
```

Código 2.11 – Algoritmo de minimos quadrados sequencialmente limitado.

Para aplicação do método SINDy, primeiro é realizada uma implementação simplificada do algoritmo de mínimos quadrados sequencialmente limitado, implementado em Python. Ao construir um conjunto de equações candidatas, o SINDy procura resolver o problema de otimização apresentado na Equação 2.34, onde $\mathbf{X}$ é uma matriz de medidas, $\dot{\mathbf{X}}$ é uma matriz de derivadas de $\mathbf{x}$, $\boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})$ é uma matriz de biblioteca cujas colunas consistem em funções potenciais do lado direito avaliadas nos dados de medição, $\boldsymbol{\xi}$ é uma matriz de coeficientes e λ é um termo de regularização que leva a esparsidade. Uma versão do algoritmo de mínimos quadrados sequencialmente limitado, que é usada para resolver a Equação 2.34, pode ser implementado em Python como:

Esta implementação resolve todas as colunas dos coeficientes de $\boldsymbol{\xi}$, produzindo as equações candidatas cada variável. Gerando a seguinte saída:

$$\dot{x}_0 = -10x_0 + 10x_1 \tag{2.39}$$
$$\dot{x}_1 = -27.998x_0 - x_1 - x_0x_2 \tag{2.40}$$
$$\dot{x}_2 = -2.667x_2 + x_0x_1 \tag{2.41}$$

$$\dot{x}_0 = -9.999x_0 + 9.999x_1 \tag{2.42}$$
$$\dot{x}_1 = 27.992x_0 - 0.999x_1 - 0.999x_0x_2 \tag{2.43}$$
$$\dot{x}_2 = -2.666x_2 + 1.000x_0x_1 \tag{2.44}$$

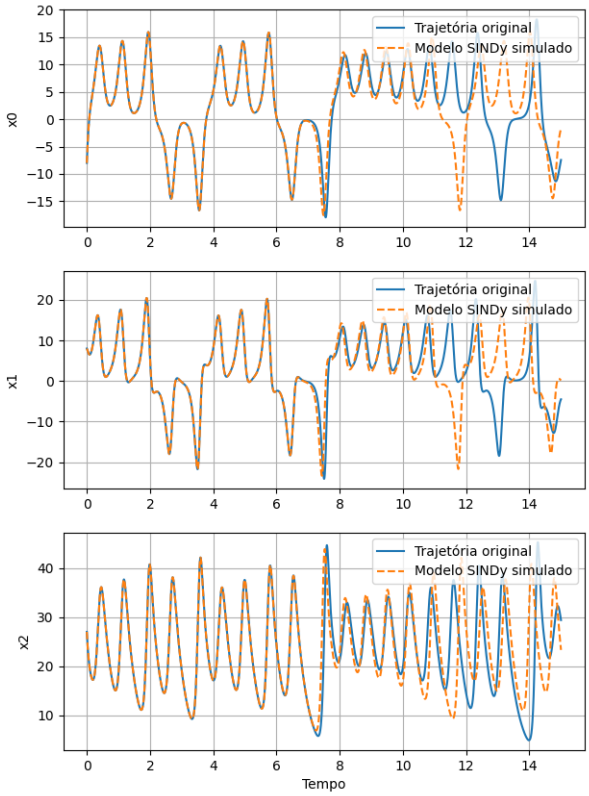

Figura 2.7 – Variáveis de estado do sistema de Lorenz em função do tempo. São apresentadas a trajetória original e o o modelo obtido através do método SINDy, para uma função polinomial de ordem 5 e limiar 0.05.

Observa-se que o sistema gerado por este algoritmo é semelhante aos parâmetros definidos para o sistema de Lorenz tanto nos termos identificados quanto nos valores de seus coeficientes, que diferem muito pouco dos valores originais. A Figura 2.7 apresenta a comparação dos dados gerados computacionalmente com aqueles gerados para a identificação. Nota-se que no começo das amostras a correspodência dos sinais é grande, o erro aumenta a medida que o tempo passa. Este fenômeno está associado muito mais a propriedade de sensibilidade a parâmetros (ou a condições iniciais) que são bem conhecidas de sistemas com dinâmica caótica.

A Figura 2.8 mostra a ideia geral da aplicação do SINDy para o sistema de Lorenz, desde como os dados são coletados e empilhados em duas grandes matrizes de dados $\boldsymbol{X}$ e $\dot{\boldsymbol{X}}$. Onde cada linha de $\boldsymbol{X}$ é um instante do estado $\boldsymbol{x}$ no tempo. Aqui, as dinâmicas do lado direito são identificadas em um espaço de termos polinomiais $\Theta(\dot{\mathbf{X}})$ em x, y, z até quinta ordem.

A Figura mostra ainda a ideia de quebrar o problema LASSO em 3 outros problemas parciais (que possuem termos de regressão comum), destacando que de todos os

parâmetros dos termos candidatos nos vetores $\boldsymbol{\xi}$ apenas alguns poucos (os corretos!) são selecionados em cada equação. Mostra-se ainda que o valor dos parâmetros selecionado está em grande acordo com os utilizados na simulação do sistema.

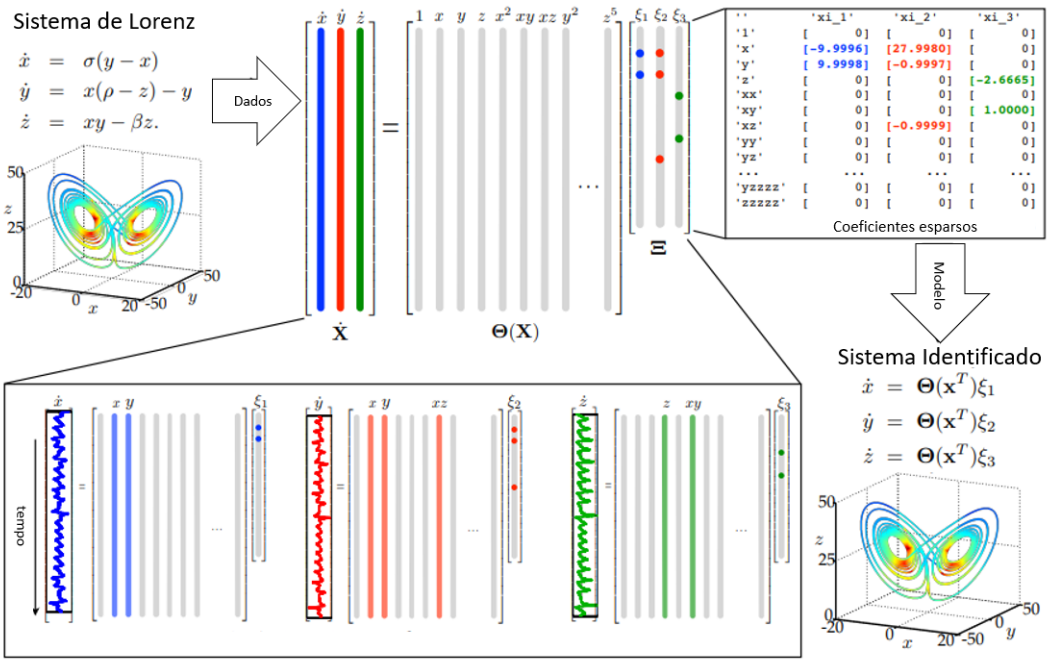

Figura 2.8 – Esquema do algoritmo de identificação esparsa de dinâmica não linear (SINDy). Modelos selecionados de uma biblioteca de termos não lineares candidatos usando regressão esparsa. Esta biblioteca X pode ser construída a partir de dados de medição. Modificado de (BRUNTON; PROCTOR; KUTZ, 2016)

.

2.5 O pacote PySINDy

Os métodos utilizados para identificação de modelos necessitam de ferramentas para torná-los amplamente acessíveis. Uma dessas ferramentas, o PySINDy, é um pacote Python usado para a descoberta de modelos de sistemas dinâmicos à partir de dados. Em particular, PySINDy fornece ferramentas para aplicação da identificação esparsa de sistemas dinâmicos não lineares (SINDy) para descoberta de modelos (S.L.Brunton,J.L.Proctor,and J.N. Kutz,2016). O pySINDy realiza a identificação do modelo como um problema de regressão esparso, onde os termos relevantes nas equações governantes são selecionados de uma biblioteca de funções candidatas. Esta abordagem pode ser personalizada usando diferentes algoritmos de regressão esparsa ou funções codificados na biblioteca. O pacote PySINDy destina-se a pesquisadores que precisam aplicar diretamente o algoritmo SINDy, sem a necessidade de codificar toda a estratégia, permitindo que o trabalho de modelagem seja focado no acesso a dados de medição para a criação de modelos. O pacote permite, também, incluir opções que permitem a usuários com conhecimento mais avançados o personalizem de acordo com suas necessidades.

O pacote PySINDy é construído em torno do método SINDy, que agrupa todas as etapas necessário identificar o modelo um sistema dinâmico com SINDy. Para aplicação do pacote PySINDy, implementa-se um sistema dinâmico a partir dos dados e identifica-se o modelo resultante utilizando o SINDy e o método *fit*, Conforme Código 2.12.

```
1    # Construcao do pacote PySINDy
2    model=ps.SINDy()
3    model.fit (x,t=dt)
```

Código 2.12 – Implementação do pacote PySINDy

O pacote PySINDy possui vários métodos alternativos para *construção da biblioteca* e *otimização*. Essas opções são selecionadas passando os termos correspondentes do PySINDy para o SINDy, como pode ser visto no Código 2.13

```
1     # Selecao da biblioteca  PolynomialLibrary
2     PySINDyfeature_library=ps.PolynomialLibrary(degree=3)
3     optimizer=ps.STLQS(threshold=0.1) # Limiar
4     # Omitmizador escolhido STLQS
5     model=ps.SINDy(
6     feature_library=feature_library,
7     optimizer=optimizer,
8     model.fit(x,t=dt)
```

Código 2.13 – Implementação do pacote PySINDy

Normalmente, pouco conhecimento prévio do sistema de interesse e sua dinâmica deve ser usado para fazer a escolha das funções para seleção da biblioteca. Quando essas informações não estiverem disponíveis, a *PolynomialLibrary* é uma boa escolha para iniciar.

O PySINDy usa vários otimizadores para resolver um problema de regressão esparsa. O otimizador padrão em PySINDy é o algoritmo de mínimos quadrados com limite sequencial(STLSQ) [7]. Além de ser o método originalmente proposto para uso com o SINDy, envolve apenas um hiperparâmetro e exibe bom desempenho em vários tipos de problemas. O parâmetro *Limiar* ou *thresholder* controla o tipo de regularização que é aplicado.

2.5.1 Exemplo: Equação de Van der Pol

O oscilador de Van der Pol é uma equação diferencial de segunda ordem com terceiro grau de não linearidade representa um sistema caótico autônomo , dado por:

$$\dot{x} = y \tag{2.45}$$

$$\dot{y} = -\mu(1 - y^2)\dot{y} + y \tag{2.46}$$

No Código 2.14 é realizada a implementação das equações 3.7 e 3.8. São definidos alguns parâmetros para realizar a identificação e utilizada a função *solve_ivp* para realizar

[7] Sequentially-thresholded least-squares algorithm

a integração. No Código 2.15 é aplicado o pacote PySINDy para identificar a equação de Van der Pol.

É definido um valor de 0.1 para μ. As variáveis de estado de Van der Pol são apresentadas na 2.9

```
# Parametros do sistema de Van der Pol
 Ts = 0.001
 t = np.arange(0, 15, Ts)
 x0=[0,1]
# Dinamica do sistema de Van der Pol
 def vanderpol(t,X):
    x = X[0]
    y = X[1]
 return[y,1*(1-x*x)*x-x]
 opcoes_integracao = {}
 opcoes_integracao['rtol'] = 1e-12
 opcoes_integracao['method'] = 'LSODA'
 opcoes_integracao['atol'] = 1e-12
 sol = solve_ivp(vanderpol, (0,T), x0, t_eval = t,**
 opcoes_integracao).y.T
```

Código 2.14 – Implementação da equação de Van der Pol em Python.

```
model = ps.SINDy(sol=solve_ivp,
optimizer=ps.STLSQ(threshold=0.05), # Limiar
feature_library=ps.PolynomialLibrary(degree=3),)
```

Código 2.15 – Implemnatação PySINDy para encontrar a equação de Van der Pol.

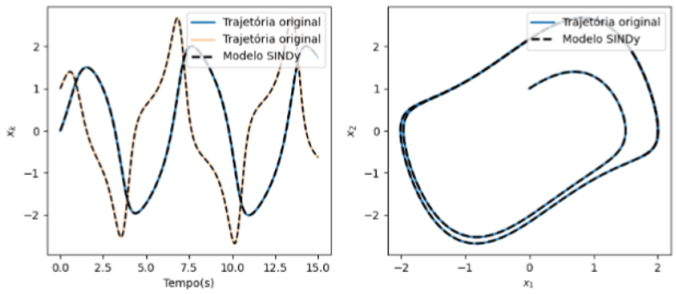

Figura 2.9 – Variáveis de estado do sistema de Van der Pol em função do tempo. São apresentadas a trajetória original e o o modelo obtido através do método SINDy, para uma função polinomial de ordem 3,Threshold=0.05 e x0=[0,1].

Repetindo o procedimento utilizado para equação de Lorenz,agora é definido um polinômio de ordem 3, a saída para este sistema é

$$\dot{x_0} = x_1 \tag{2.47}$$

$$\dot{x_1} = -x_0 + 0.100x_1 - 0.100x_0^2x_1 \tag{2.48}$$

Integração numérica das equações 3.7 e 3.8 mostra que para toda condição inicial (exceto x = 0, 'x = 0) se aproxima de um único movimento periódico. Isso depende do valor de μ para pequenos valores de μ o movimento é quase senoidal, enquanto que para grandes valores de μ é uma oscilação de relaxamento, similar a uma série de funções de passo,indo entre valores positivos e negativos duas vezes por ciclo. A integração numérica mostra que o ciclo limite é uma curva fechada envolvendo a origem no plano de fase x-y, conforme (ROZMAN, 2020)

2.5.2 Exemplo: Equação de Duffing

A equação de Duffing que é uma equação diferencial não linear (não linearidade no termo x^3) de segunda ordem usada para modelar osciladores, sendo também um exemplo de sistema dinâmico que apresenta comportamento caótico. Considerando o caso sem forçamento externo $\gamma = 0$:

$$\ddot{x} + \delta\dot{x} - \alpha x + \beta x^3 = 0 \tag{2.49}$$

Transformando em um sistema de primeira ordem, obtém-se:

$$\dot{u} = v \tag{2.50}$$

$$\dot{v} = \alpha u - \beta u^3 - \delta v \tag{2.51}$$

Os parâmetros para esse sistema são $\beta = 0.2$, $\sigma = 0.05$ e $\rho = 1$ para condição inicial de x0=[0,1]. No Código 2.16 é realizada a implementação da da equação de Duffing. Como no exemplo do sistema de van der Pol, foram estabelecidos e utilizado a função *solve_ivp* para realizar a integração. No Código 2.17, é utilizado o pacote PySINDy para identificar a equação de Duffing. Para este modelo foi verificado que um polinômio de ordem 3 satisfaz o modelo SINDy.

```
    # Parametros do sistema de Duffing
    Ts = 0.001
    t = np.arange(0,20 , Ts)
    x0=[0,1]
    # Dinamica do sistema de Duffing
    def Duffing(t, x, p=[0.2, 0.05, 1]):
    return [x[1], -p[0] * x[1] - p[1] * x[0] - p[2] * x[0] ** 3]
    opcoes_integracao = {}
    opcoes_integracao['rtol'] = 1e-12
    opcoes_integracao['method'] = 'LSODA'
    opcoes_integracao['atol'] = 1e-12
    sol = solve_ivp(duffing, (0,T), x0, t_eval = t,**
    opcoes_integracao).y.T
```

Código 2.16 – Implementação da equação de Duffing em Python.

```
1
2     model = ps.SINDy(
3     optimizer=ps.STLSQ(threshold=0.05), # Limiar
4     feature_library=ps.PolynomialLibrary(degree=3),)
```

Código 2.17 – Implemnatação PySINDy para encontrar a equação de Duffing.

$$\dot{x}_0 = x_1 \tag{2.52}$$

$$\dot{x}_1 = -0.05x_0 - 0.2x_1 - x_0^3 \tag{2.53}$$

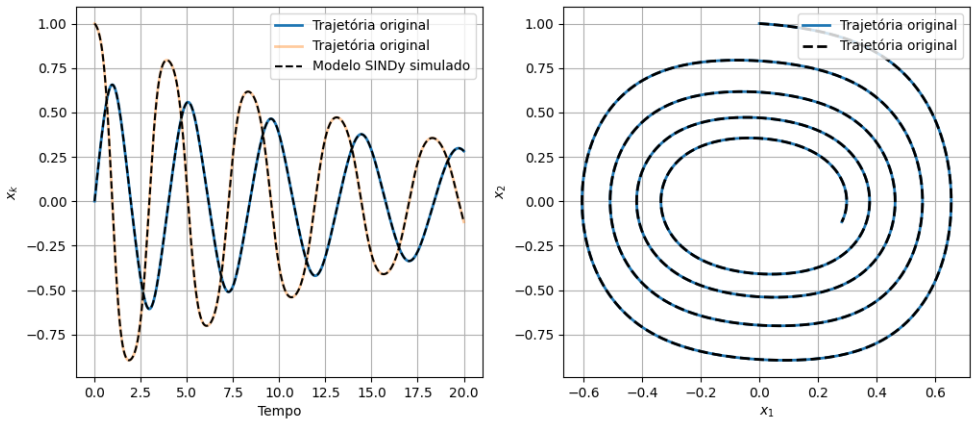

Figura 2.10 – Variáveis de estado do sistema de Duffing em função do tempo. São apresentadas a trajetória original e o o modelo obtido através do método SINDy, para uma função polinomial de ordem 3,Threshold=0.05 e x0=[0,1]

3 Aplicação do Método SINDy para Identificação do Sistema Motor Gerador

Neste capítulo será abordado a utilização do método SINDy aplicado a um sistema real, o sistema motor-gerador (MG), uma planta didática para experimentação em sistemas dinâmicos e controle. O capítulo apresenta uma descrição do sistema e suas funcionalidades. Em seguida serão realizados ensaios que permitirão verificar a não linearidade, a partir da diferença de comportamento para a resposta em degrau em diferentes pontos de operação. Serão então levantados modelos lineares em cinco diferentes pontos de operação do sistema e do modelo não linear, sendo que para este, será necessário apenas um conjunto de dados. O dados reais de entrada-saída serão coletados e tratados e, após isso, será realizada a aplicação do algoritmo SINDy para identificar os modelos..

3.1 Sistema Motor-Gerador

Neste trabalho foi usada uma bancada de motor-gerador disponibilizada pelo Laboratório de Sistemas de Controle, da Universidade Federal do Pará - CAMTUC. Esta é formada por dois motores DC, um Drive L298N, uma matriz de contato e uma placa de Arduíno, conforme pode ser visto na Figura 3.1. A bancada é alimentada por uma fonte de tensão DC de 15V, a tensão gerada nos terminais do gerador é levada para um filtro RC que tem a finalidade de minimizar o ruído na medição. Após isso a tensão filtrada é entregue ao Arduíno. A placa Arduíno funciona como um elo de ligação entre a bancada e o computador. O algoritmo de comando encontra-se no computador e nele é definido qual a tensão o motor DC deve ser alimentado para que, no terminal do gerador, seja observada a tensão desejada.

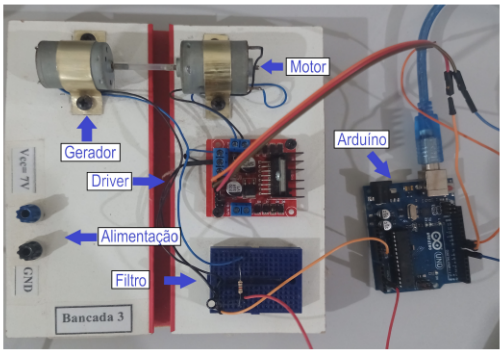

Figura 3.1 – Bancada motor-gerador DC.

O diagrama de blocos da bancada motor-gerador, Figura 3.2 é uma forma de mostrar a rede de comunicação entre os dispositivos na sua forma mais simples e direta.

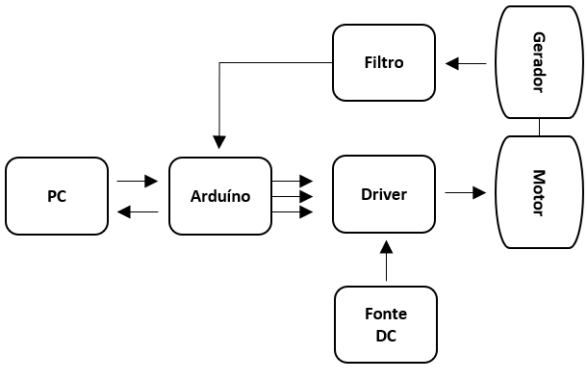

Figura 3.2 – Diagrama de blocos da bancada motor-gerador DC

3.2 Caracterização dinâmica do motor-gerador

Para a caracterização dinâmica do sistema motor-gerador foi aplicado um sinal degrau em diferentes pontos. Este sinal foi gerado através de um código Python e aplicado ao motor-gerador. Os pontos de interesse para levantamento da curva foram desde a zona morta do até o ponto onde o motor começa entrar em regime permanente. À partir do dados levantados foi plotado a curva $V_a \times V_\omega$, sendo V_a o sinal de tensão aplicado na armadura do motor e V_ω o sinal de tensão medido no gerador, o qual possui uma correspondência com a velocidade angular do motor. Os valores de regime permanente destes sinais são apresentados na Figura 3.3.

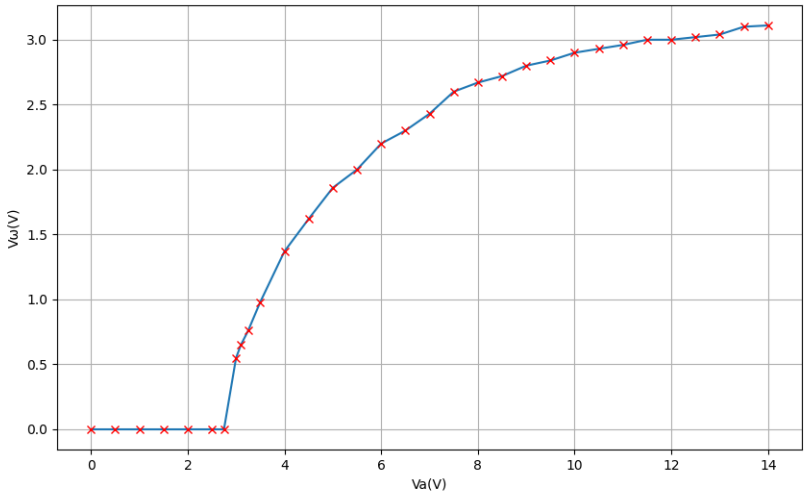

Figura 3.3 – Curva da relação $V_a \times V_\omega$ do motor gerador, exibindo as não-linearidades de zona-morta e ganho variável.

3.3 Validação Cruzada

A validação cruzada é um método estatístico de avaliação e comparação de algoritmos de aprendizado de forma a avaliar a qualidade de predição do modelo.É feita validando o modelo com um conjunto de dados diferente do usado para estimar os parâmetros. O procedimento é feito dividindo os dados em dois grupos: um usado para treinar o e o outro utilizado para validar o modelo. Na validação cruzada, os conjuntos de treinamento e validação devem ser cruzados em rodadas sucessivas, de modo que cada dado ponto tem uma chance de ser validado. Para determinar o grau de ajuste foi utilizada a métrica do complemento da normalização da NRMSE [1], que é definida por:

$$NRMSE = \left(1 - \frac{\sqrt{(y - y_p)^2}}{\sqrt{(y - y_m)^2}}\right) 100 \tag{3.1}$$

Onde

y: são os valores das variáveis controladas do conjunto de dados de validação;
y_m: é a média dos valores de cada y;
y_p: é o valor predito para cada y.

Para validação dos modelos foi utilizada a validação cruzada. Foram adicionados os valores médios dos dados experimentais de validação de saída aos dados de saída obtidos a partir dos modelos estimados e dessa forma realizada a comparação entre ambos.

3.4 Experimento: Modelo Linear

A consideração da linearidade normalmente simplifica encontrar o modelo a ser desenvolvido pela facilidade de obtenção e por uma ampla disponibilidade de ferramentas matemáticas na engenharia. Contudo,estes modelos comportam-se de forma aproximadamente linear, isso é verificado observando-se o comportamento do sistema numa faixa estreita de operação. Assim, os modelos são precisos somente na vizinhança dos pontos de operação escolhidos para o processo de identificação.

Como o sistema exibe não linearidades, serão levantados modelos lineares em diferentes pontos de operação. Para implementar este sinal de entrada são definidos cinco pontos de tensão V_a para coleta de dados.

3.4.1 Geração do Sinal PRBS

Neste experimento foi utilizada uma bancada de motor-gerador de 15v. Para identificação do modelo linear em regime permanente em diferentes pontos de operação, utilizados um sinal de entrada PRBS. Foi implementado um código em Python conforme Apêndice A para implementar este sinal de entrada e são definidos cinco pontos de operação

[1] Normalized Root Mean Square Error - Raiz do erro médio quadrático normalizado

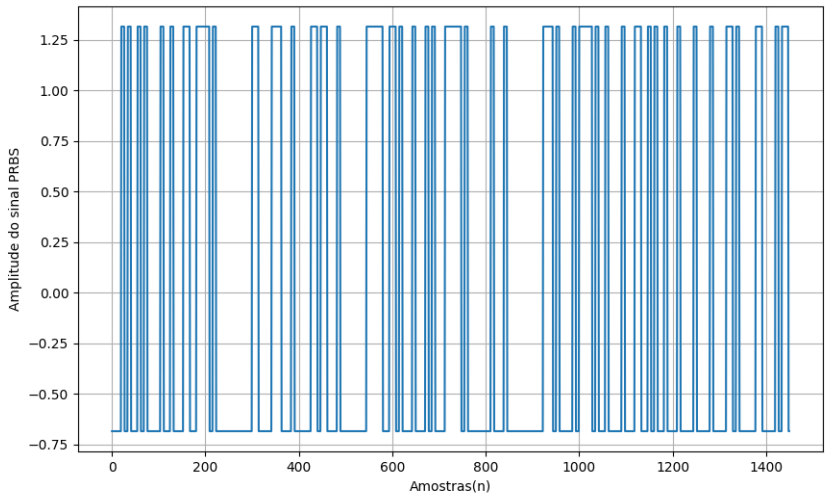

Figura 3.4 – Sinal PRBS implementado através de código Python e gerado para identificação linear do motor-gerador.

para coleta de dados, sendo 5, 6.5, 8, 9, 12.5. Para a criação do sinal PRBS foi utilizada a função $PRBS(size\_min\_seq, size)$, o sinal pode ser visto na Figura 3.4. É utilizada também a função *Random.seed* da biblioteca *Numpy* para definir um padrão para a sequência gerada.

3.4.2 Identificação de modelos lineares

Após a coleta de dados, é realizado o pré-processamento dos dados coletados. É definida uma variável N_t que será o números de amostras retiradas do transitório, dessa forma definimos $N_t = 50$. Após isso é removida uma parcela que será utilizada para validação cruzada, para obter modelos iniciais do sistema. N_i serão os dados de teste, é definido 70% destes dados para teste e 30% para validação.

O método PySINDy é utilizado para realizar a identificação. Foram testados para os cinco conjuntos de dados e obtidos os modelos para estes. Para este experimento foram consideradas funções polinomiais candidatas de ordem 1.

```
1    modelo = ps.SINDy(t_default=Ts,
2        discrete_time=True,
3        optimizer=ps.STLSQ(threshold=0.01),
4        feature_library=ps.PolynomialLibrary(degree=1))
5    model.fit(Xi, u = ui, t = Ts)
6    model.print()
```

Código 3.1 – Código de identificação SINDy para o modelo linear do motor-gerador.

Para o modelo linear, realizamos o experimento considerando modelos de ordem 1 e *limiar* de 0.01. Foram simulados cinco modelos utilizando cinco conjuntos de dados obtidos em pontos de operação diferentes1. Cada modelo foi testado para todos os conjuntos de

dados coletados em pontos de operação diferentes. O modelo y_{p1} foi identificado para os dados y_{r1}, y_{r2},y_{r3},y_{r4} e y_{r5}, como pode ser visto na Figura 3.5. Isso foi feito, também, para y_{p2}, y_{p3}, y_{p4} e y_{p5}, apresentados nas Figuras 3.6, 3.7,3.8 e 3.9. As equações para cada modelo identificados são apresentadas a seguir

Modelos lineares para $Limiar$=0.01,com n_t=4:

fit=97.6, para tensão de 5V

$$\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.258x_0[k] + 1.114x_1[k] + 0.064u_0[k] \end{cases}$$

fit=97, para tensão de 6.5V

$$\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.108x_0[k] + 0.893x_1[k] + 0.054u_0[k] \end{cases}$$

fit=94.05,para tensão de 8V

$$\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.103x_0[k] + 0.847x_1[k] + 0.042u_0[k] \end{cases}$$

fit=88.08,para tensão de 9V

$$\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = 0.042x_0[k] + 0.636x_1[k] + 0.035u_0[k] \end{cases}$$

fit=70.05,para tensão de 12.5V

$$\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = 0.016x_0[k] + 0.558x_1[k] + 0.024u_0[k] \end{cases}$$

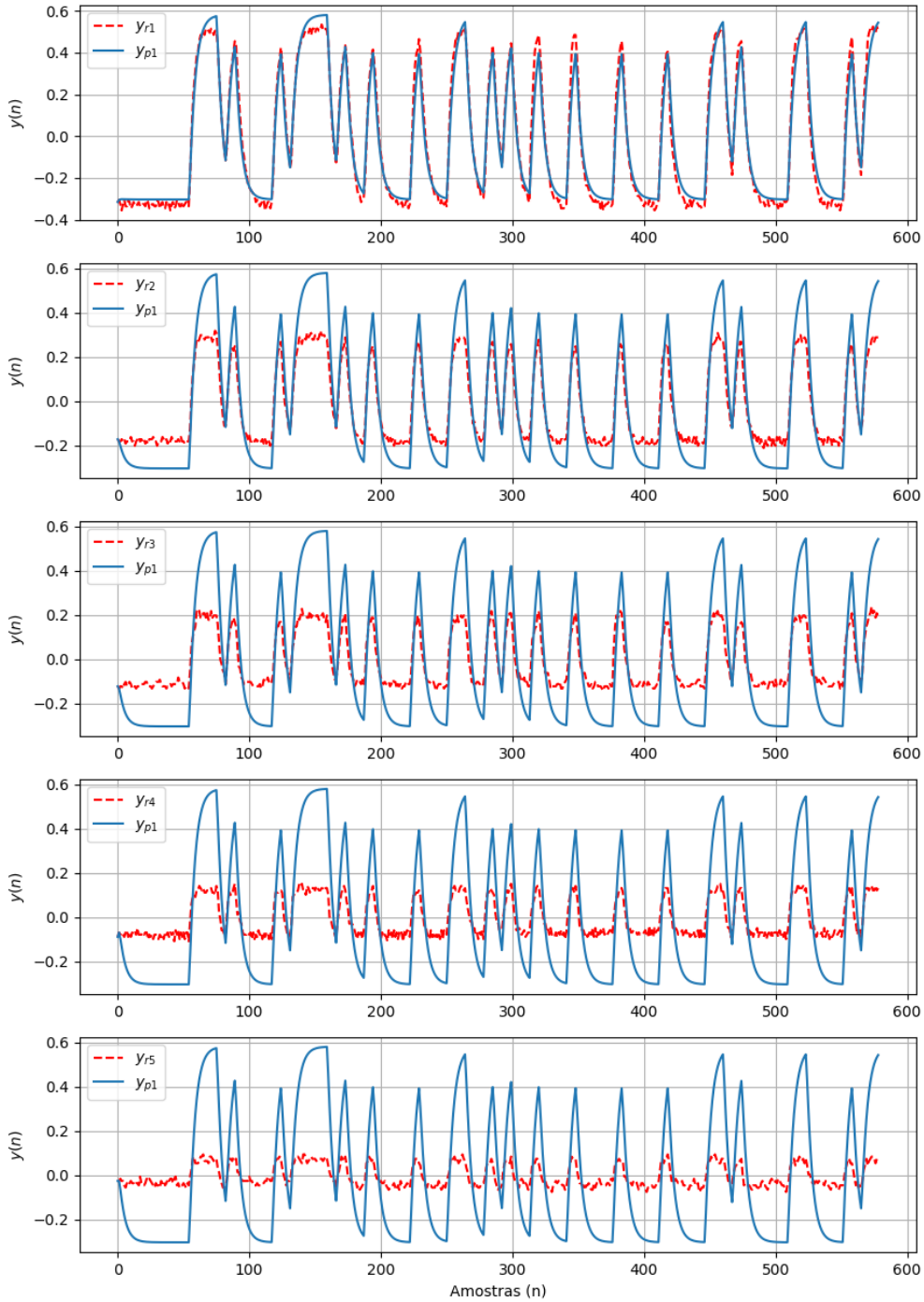

Figura 3.5 – Comparação entre os dados observados y_{r1},y_{r2},y_{r3}, y_{r4}, y_{r5} e o modelo obtido pelo método SINDy y_{p1}, para dados coletados para tensão de 5V. Os modelos identificados foram produzidos utilizando o Threshold de 0.01.

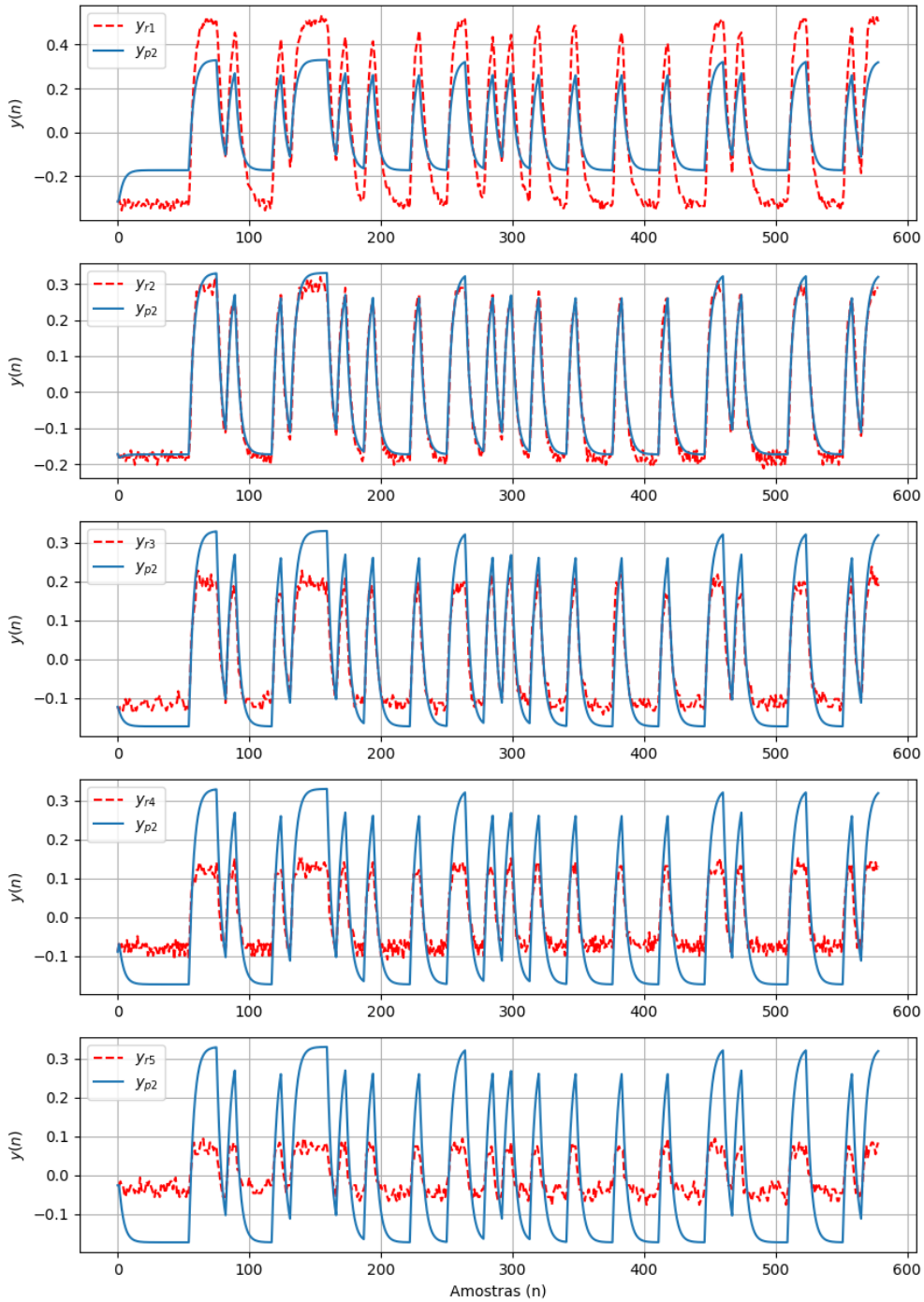

Figura 3.6 – Comparação entre os dados observados y_{r1},y_{r2},y_{r3}, y_{r4}, y_{r5} e o modelo obtido pelo metodo SINDy y_{p2}, para dados coletados para tensão de 5V. Os modelos identificados foram produzidos utilizando o Threshold de 0.01.

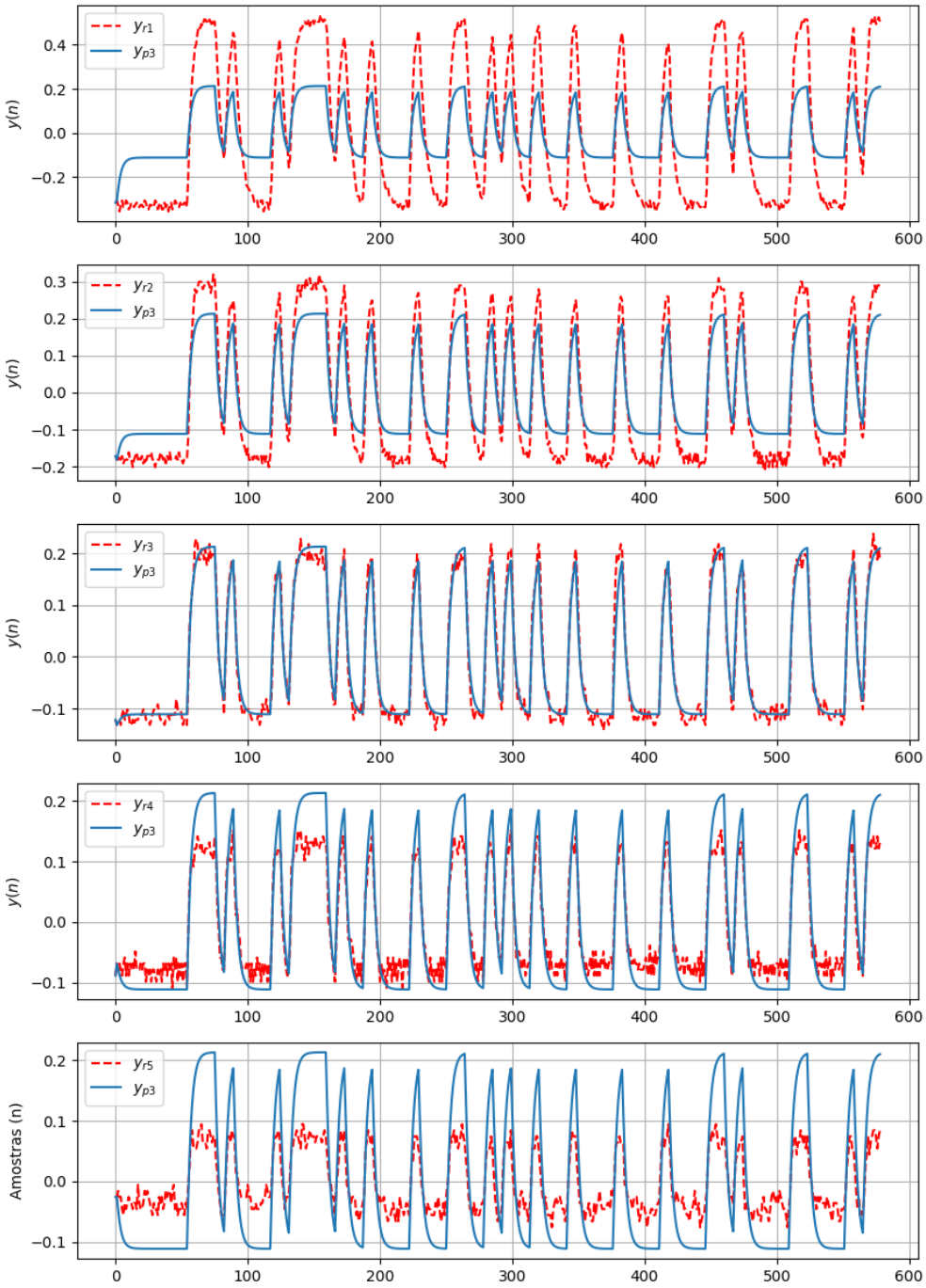

Figura 3.7 – Comparação entre os dados observados y_{r1},y_{r2},y_{r3}, y_{r4}, y_{r5} e o modelo obtido pelo metodo SINDy y_{p3}, para dados coletados para tensão de 5V. Os modelos identificados foram produzidos utilizando o Threshold de 0.01.

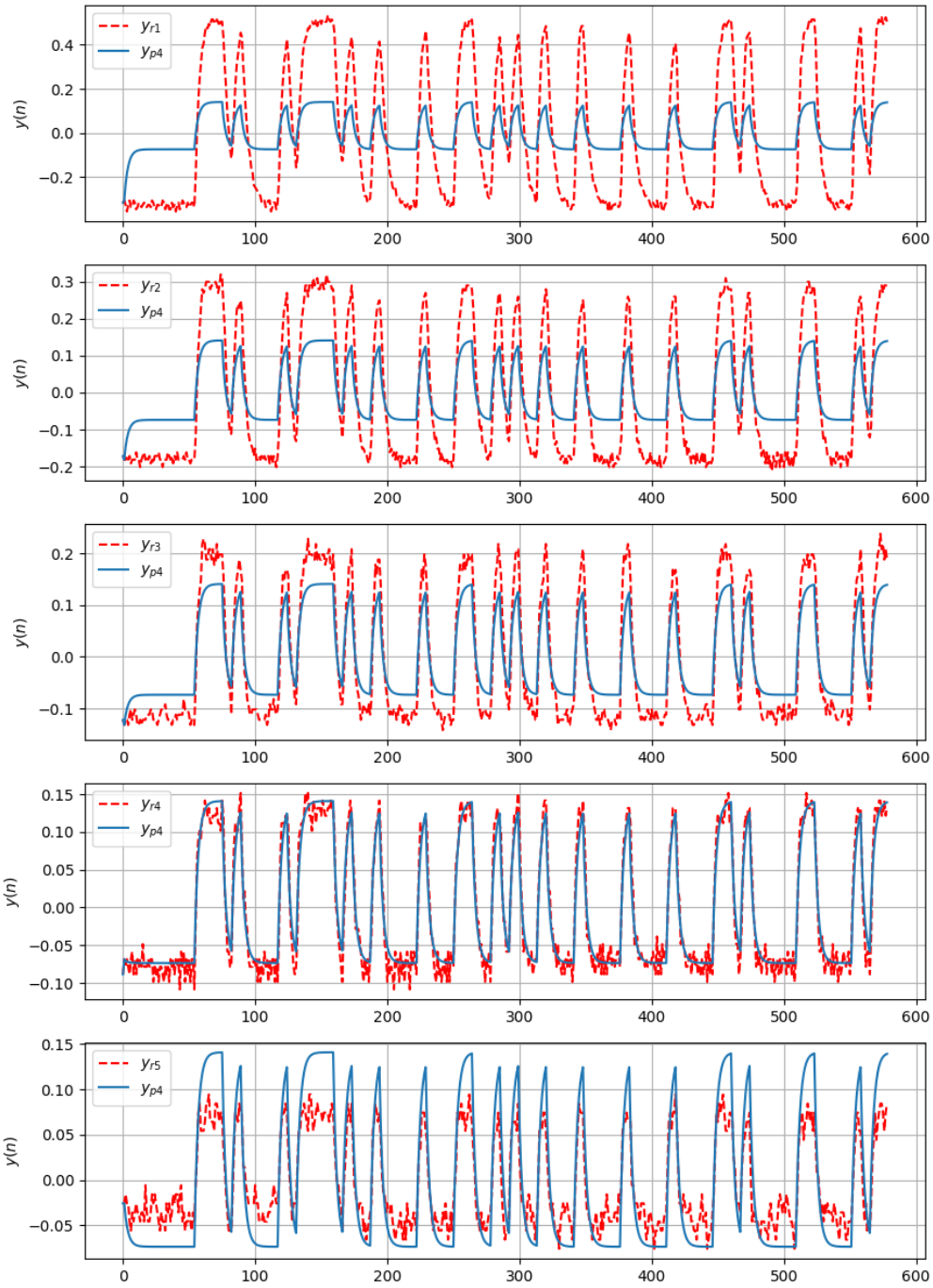

Figura 3.8 – Comparação entre os dados observados y_{r1},y_{r2},y_{r3}, y_{r4}, y_{r5} e o modelo obtido pelo metodo SINDy y_{p4}, para dados coletados para tensão de 5V. Os modelos identificados foram produzidos utilizando o Threshold de 0.01.

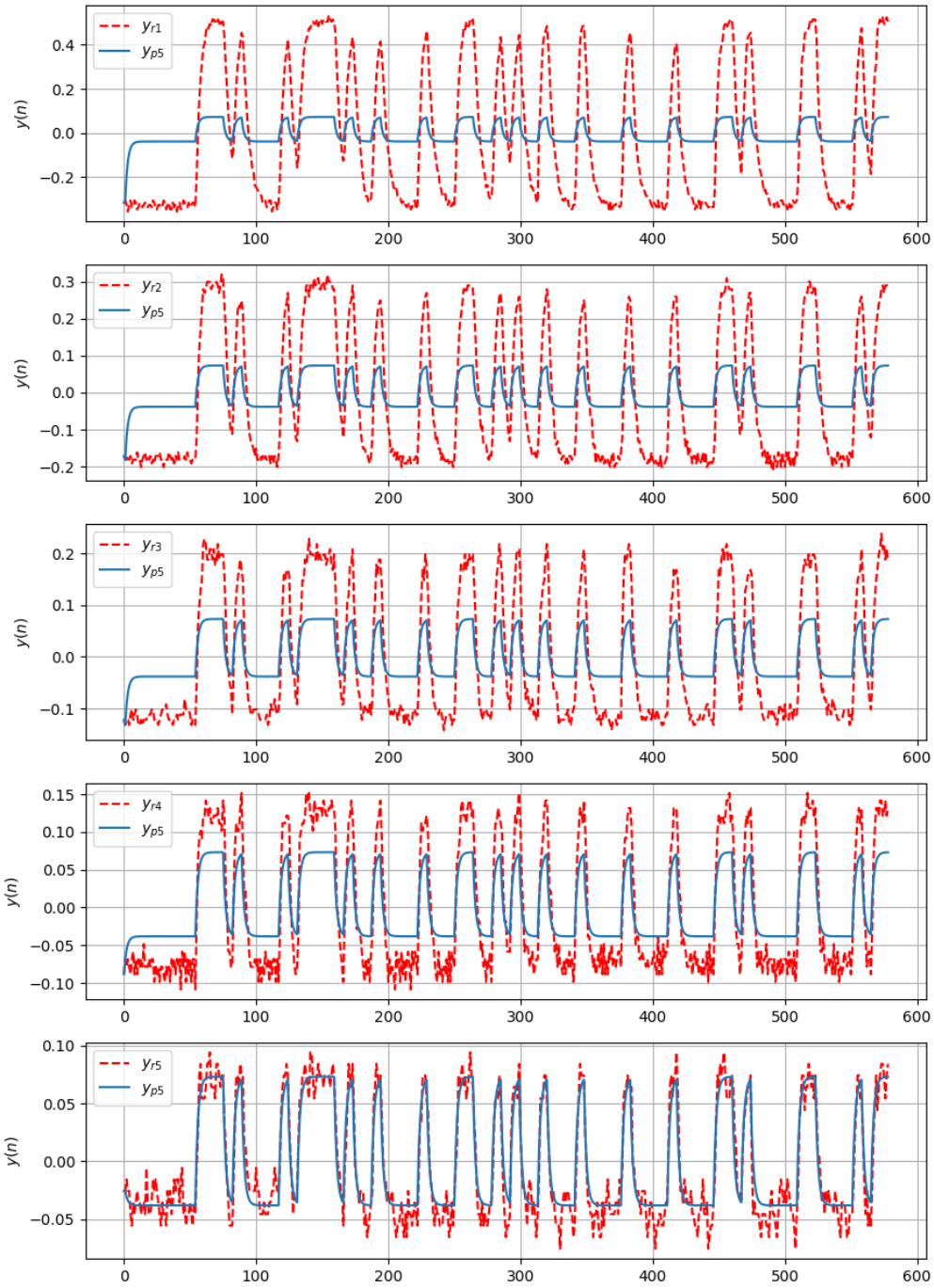

Figura 3.9 – Comparação entre os dados observados y_{r1},y_{r2},y_{r3}, y_{r4}, y_{r5} e o modelo obtido pelo metodo SINDy y_{p5}, para dados coletados para tensão de 5V. Os modelos identificados foram produzidos utilizando o *limiar* de 0.01.

3.4.3 Análise dos resultados dos modelos lineares

A Figura 3.10 apresenta o resultado para cada modelo fazendo a validação cruzada com cada conjunto de dados. A diagonal em azul escuro apresenta os modelos com melhor fit, esses correspondem aos modelos testados para seu próprio conjunto de dados de validação. Na Figura 3.11, temos a comparação entre os modelos lineares obtidos para os diferentes conjuntos de dados coletados e ajuste NRMSE, na forma de um gráfico de linhas. Novamente, podemos observar que para seu próprio conjunto de dados os modelos apresentaram melhor *limiar*.

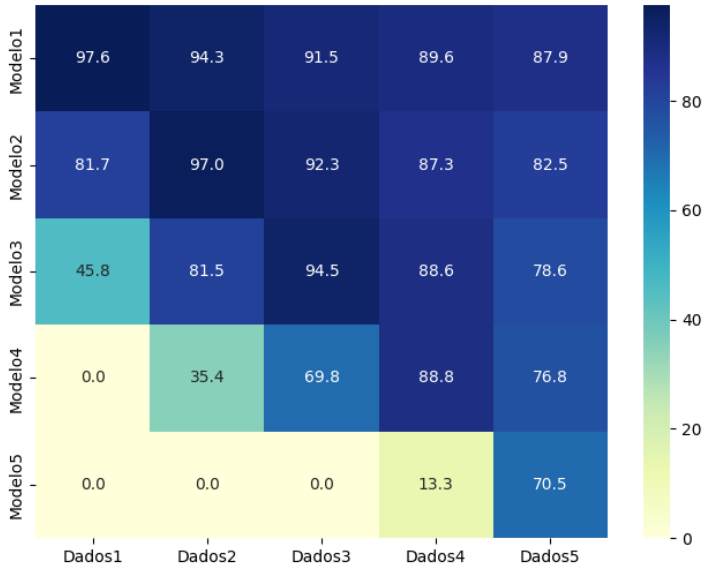

Figura 3.10 – Mapa de calor para todas as funções candidatas do sistema dinâmico em cada iteração do método pelo método SINDy. cores mais claras indica termos a serem eliminados;A cor branca indica termos eliminados;o azul mais escuro indica os termos mais importantes.

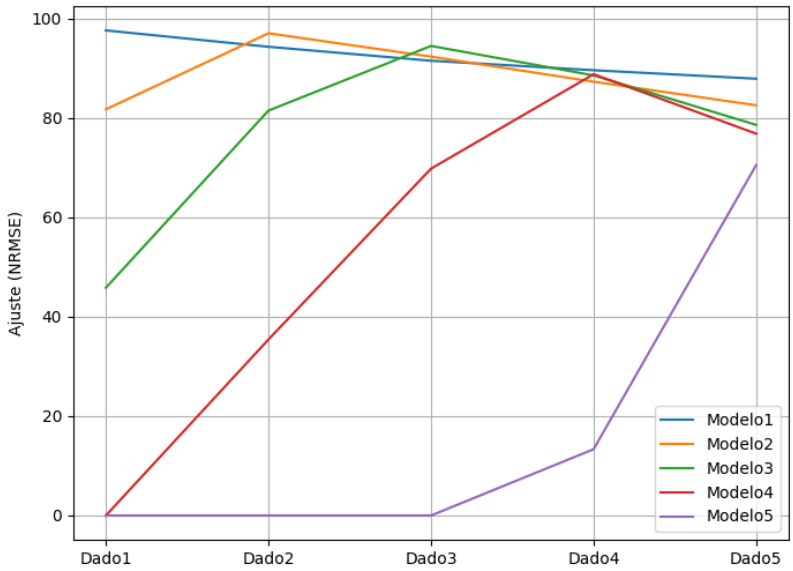

Figura 3.11 – Comparação entre os modelos lineares obtidos para os diferentes conjuntos de dados.

3.5 Experimento: Modelo Não-Linear

Os modelos lineares possuem as estruturas mais simples, e na grande maioria das aplicações são suficientes, entretanto em algumas situações essa consideração não é adequada, como para sistemas de dinâmica bilinear(que não podem ser descritos adequadamente por um único modelo linear) e em casos em que se deseja estudar características dinâmicas não lineares do sistema.

3.5.1 Geração do sinal PRMLS

O sinal PRMLS é preferível para identificação de modelos não lineares, já que alternância entre duas amplitudes pode não capturar o comportamento não-linear. Dessa forma, para o experimento não linear, foi implementado e aplicado um sinal PRMLS. Na Figura 3.12 é apresentado o sinal PRMLS utilizado para identificação do motor-gerador. O código utilizado para implementação deste sinal se encontra no Apêndice B.

3.5.2 Identificação de modelos não-lineares

A identificação é realizada usando um conjunto de dados de coletados do motor-gerador. Um segundo conjunto de dados de 70% amostras é utilizado para validação. Para o ensaio não linear foi realizada,também, a identificação do modelo linear. Pode-se observar pela comparação entre a saída real (y_r) e a saída do modelo linear identificado SINDy

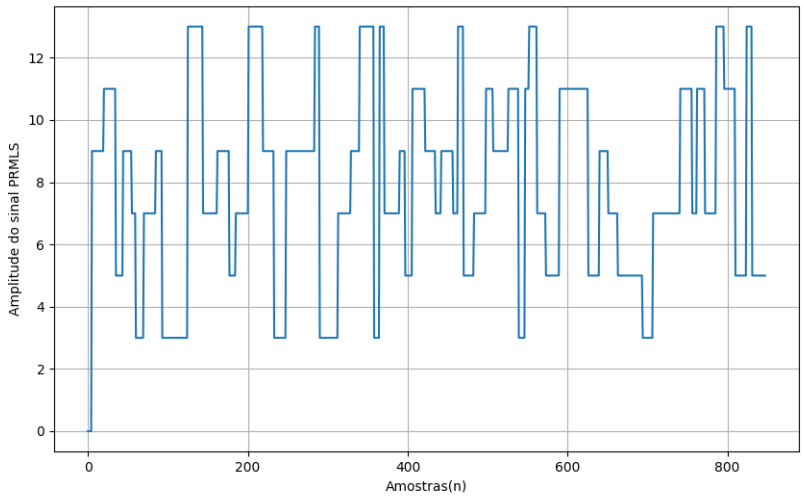

Figura 3.12 – Sinal PRMLS implementado através de código Python e gerado para identificação não linear do motor-gerador.

(y_{m1}), Figura 3.13, que o comportamento do sistema não é bem representado. Para este modelo foi escolhido o $\tau = 0.0005$ e obtido um fit de 96.93%.

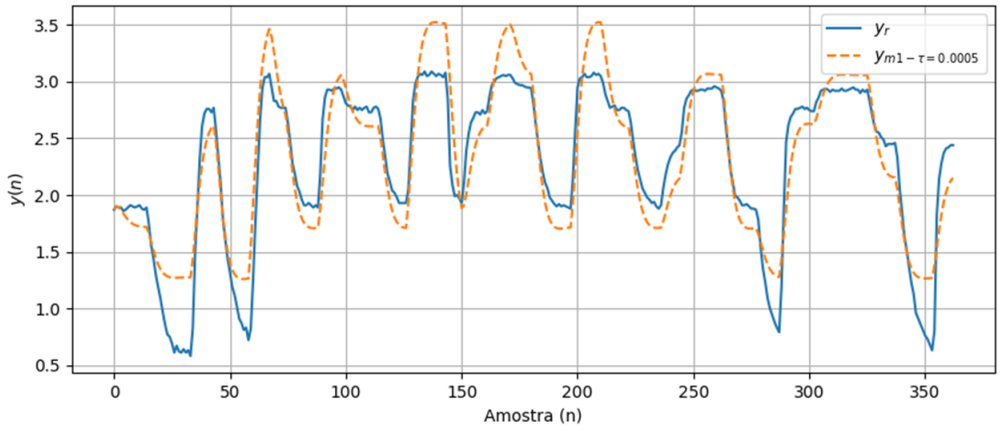

Figura 3.13 – Comparação entre a saída real e saída do modelo linear identificado utilizando o ensaio para identificação não linear. Sendo y_r a saída real e y_{m1} a saída do modelo identificado.

Modelo não-linear y_{m1} para $Limiar$=0.0005, com n_t=4:
fit=96.93

$$\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.513x_0[k] + 1.391x_1[k] + 0.027u_0[k] \end{cases}$$

Na Figura 3.14, temos o modelo de ordem 2 . Foram escolhidos 3 valores de *limiar* para observar o comportamento da resposta de saída do sistema identificado (y_{m2}).É

possível notar que conforme diminuímos o valor de *limiar*, o modelo identificado pelo método SINDy se aproxima da saída real, contudo, o número de termos para esse modelo aumenta. Essa relação pode ser observada, também, no modelos obtidos apresentados a seguir, onde são apresentadas a equações identificadas do modelo não linear de ordem 2.

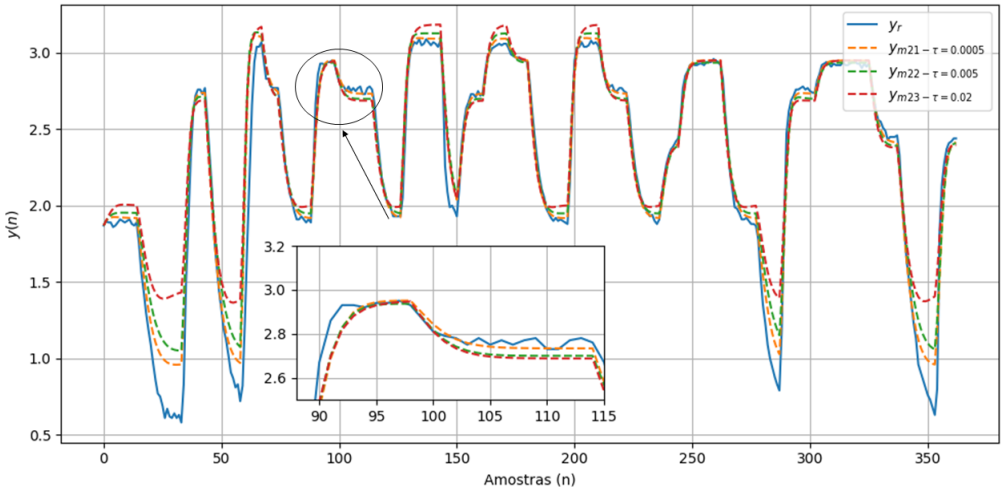

Figura 3.14 – Comparação entre a saída real y_r e as saídas do modelo não linear identificado de ordem 2 (y_{m21},y_{m22},y_{m23}) utilizando o pacote PySINDy.

Modelos não-lineares de ordem 2:

fit=98.49, *Limiar*=0.0005, n_t=11

$$y_{m21}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.1791 - 0.802x_0[k] + 1.812x_1[k] + 0.079u_0[k] - 0.041x_0[k] \\ +0.323x_0[k]x_1[k] - 0.015x_0[k]u_0[k] - 0.317x_1[k]^2 + 0.008x_1[k]u_0[k] \\ -0.002u_0[k]^2 \end{cases}$$

fit=98.09, *Limiar*=0.005, n_t=8

$$y_{m22}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.1661 - 0.906x_0[k] + 1.961x_1[k] + 0.061u_0[k] + 0.229x_0[k]x1[k] \\ -0.267x_1[k]^2 - 0.011x_1[k]u_0[k] \end{cases}$$

fit=97.28, *Limiar*=0.02, n_t=7

$$y_{m23}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.1201 - 1.149x_0[k] + 2.253x_1[k] + 0.037u[k] + 0.336x_0[k]x_1[k] \\ -0.404x_1[k]^2] \end{cases}$$

O modelo de ordem 3 tem uma boa representação dos sistema. Em algumas regiões, conforme aumentamos *limiar*, os modelos apresentam variação pequena. Esse conjunto de

modelos apresentam dinâmica próximo ao sinal de saída real, conforme Figura 3.15. A seguir são apresentadas as as equações do modelo de ordem 3, com o aumento de *limiar* podemos notar que o número de termos das equações aumenta.

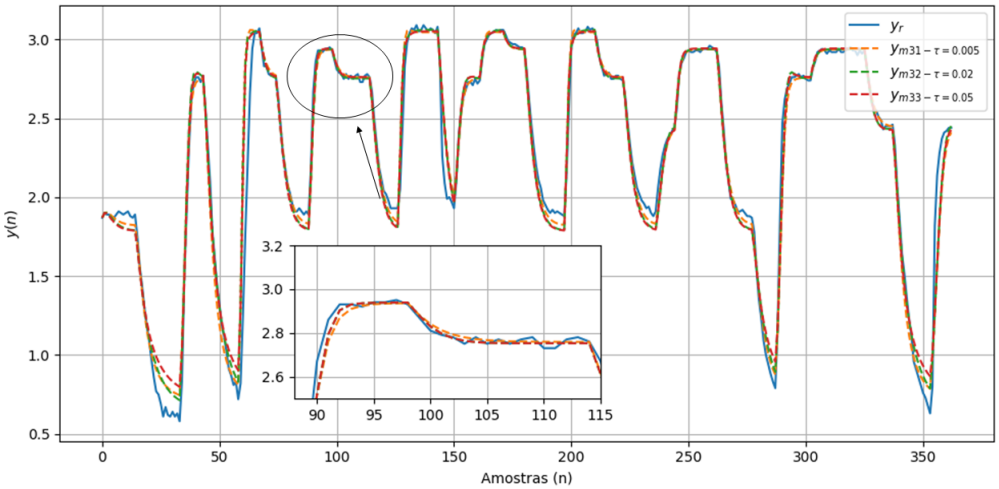

Figura 3.15 – Comparação entre a saída real y_r e as saídas do modelo não linear identificado de ordem 2 (y_{m31},y_{m32},y_{m33}) utilizando o pacote PySINDy.

Modelos não-lineares de ordem 3:

fit=98.56, *Limiar*=0.005, n_t=19

$$y_{m31}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.0531 - 0.139x_0[k] + 0.960x_1[k] + 0.042u_0[k] + 0.683x_0[k]^2 \\ -2.023x_0[k]x_1[k] + 0.080x_0[k]u_0[k] + 1.378x_1[k]^2 - 0.056x_1[k]u_0[k] \\ -0.111x_0[k]^3 + 0.590x_0[k]^2x_1[k] - 0.140x_0[k]^2u_0[k] - 0.804x_0[k]x_1[k]^2 \\ +0.367x_0[k]x_1[k]u_0[k] - 0.017x_0[k]u_0[k]^2 + 0.321x_1[k]^3 - 0.236x_1[k]^2u_0[k] \\ +0.017x_1[k]u_0[k]^2 \end{cases}$$

fit=98.59, *Limiar*=0.02, n_t=15

$$y_{m32}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.0381 + 0.390x_0[k] + 0.397x_1[k] + 0.038u_0[k] - 0.430x_0[k]^2 \\ +0.433x_0[k]x_1[k] - 0.155x_0[k]u_0[k] + 0.056x_1[k]^2 + 0.182x_1[k]u_0[k] + 0.223x_0[k]^3 \\ -0.418x_0[k]^2x_1[k] + 0.188x_0[k]x_1[k]^2 + 0.072x_0[k]x_1[k]u_0[k] - 0.082x_1[k]^2u_0[k] \end{cases}$$

fit=98.54, *Limiar*=0.05, n_t=11

$$y_{m33}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = 0.050x_0[k] + 0.820x_1[k] - 0.056x_0[k]^2 - 0.137x_0[k]u_0[k] + 0.203x_1[k]u_0[k] \\ +0.158x_0[k]^3 - 0.386x_0[k]^2x_1[k] + 0.250x_0[k]x_1[k]^2 \\ +0.066x_0[k]x_1[k]u_0[k] - 0.085x_1[k]^2u_0[k] \end{cases}$$

Para o modelo de ordem 4, as saídas identificadas apresentam comportamento bem próximo ao da saída real. Nos pontos mais baixos os modelos apresentam desempenho semelhantes. Conforme diminuímos *limiar* o número de termos nas equações aumenta, contudo mesmo variando este parâmetro de 0.0005 à 0.005 só há perda de dois termos entre a as duas equações. Não havendo, também, mudanças significativas no fit. Para o modelo y_{m33} apesar de diminuir o *limiar*, este modelo simulou bem o sistema real da mesma forma que os modelos y_{m31} e y_{m32}.

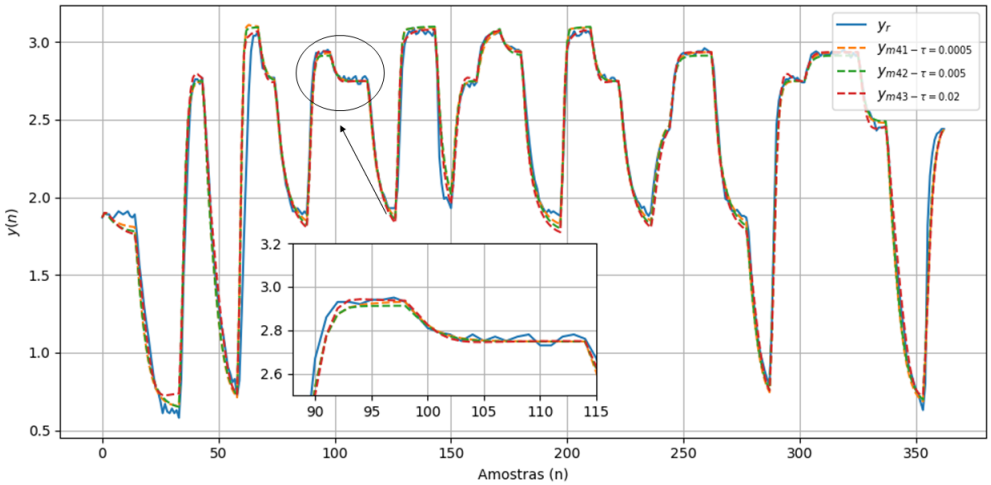

Figura 3.16 – Comparação entre a saída real y_r e as saídas do modelo não linear identificado de ordem 4 (y_{m41},y_{m42},y_{m43}) utilizando o pacote PySINDy.

Modelos não-lineares de ordem 4:
fit=98.59, $Limiar$=0.0005, n_t=32

$$
y_{m41}\begin{cases}
x_0[k+1] = x_1[k] \\
x_1[k+1] = -0.0171 + 0.888x_0[k] - 0.719x_1[k] + 0.114u_0[k] + 0.511x_0[k]^2 \\
-1.289x_0[k]x_1[k] - 0.281x_0[k]u_0[k] + 0.954x_1[k]^2 + 0.497x_1[k]u_0[k] \\
-0.022u_0[k]^2 - 3.301x_0[k]^3 + 10.177x_0[k]^2x_1[k] - 0.474x_0[k]^2u_0[k] \\
-10.082x_0[k]x_1[k]^2 + 0.868x_0[k]x_1[k]u_0[k] + 0.020x_0[k]u_0[k]^2 + 3.284x_1[k]^3 \\
-0.506x_1[k]^2u_0[k] - 0.022x_1[k]u_0[k]^2 + 0.001u_0[k]^3 - 0.084x_0[k]^4 + 2.249x_0[k]^3x_1[k] \\
-0.100x_0[k]^3u_0[k] - 6.559x_0[k]^2x_1[k]^2 + 0.575x_0[k]^2x_1[k]u_0[k] - 0.013x_0[k]^2u_0[k]^2 \\
+6.492x_0[k]x_1[k]^3 - 0.756x_0[k]x_1[k]^2u_0[k] + 0.017x_0[k]x_1[k]u_0[k]^2 - 0.001x_0[k]u_0[k]^3 \\
-2.113x_1[k]^4 + 0.286x_1[k]^3u_0[k]
\end{cases}
$$

fit=98.64, *Limiar*=0.005, n_t=30

$$
y_{m42}\begin{cases}
x_0[k+1] = x_1[k] \\
x_1[k+1] = -0.0181 + 0.854x_0[k] - 0.309x_1[k] + 0.031u_0[k] + 0.760x_0[k]^2 \\
-2.542x_0[k]x_1[k] - 0.102x_0[k]u_0[k] + 1.733x_1[k]^2 + 0.270x_1[k]u_0[k] - 3.673x_0[k]^3 \\
+11.489x_0[k]^2x_1[k] - 0.619x_0[k]^2u_0[k] - 11.181x_0[k]x_1[k]^2 + 1.144x_0[k]x_1[k]u_0[k] \\
+3.495x_1[k]^3 - 0.606x_1[k]^2u_0[k] - 0.005x_1[k]u_0[k]^2 - 0.122x_0[k]^4 + 2.658x_0[k]^3x_1[k] \\
-0.121x_0[k]^3u_0 - 7.610x_0[k]^2x_1[k]^2 + 0.655x_0[k]^2x_1[k]u_0[k] - 0.007x_0[k]^2u_0[k] \\
+7.460x_0[k]x_1[k]^3 - 0.862x_0[k]x_1[k]^2u_0[k] + 0.008x_0[k]x_1[k]u_0[k]^2 - 2.411x_1[k]^4 \\
+0.336x_1[k]^3u_0[k]
\end{cases}
$$

fit=98.60, *Limiar*=0.02, n_t=19

$$
y_{m43}\begin{cases}
x_0[k+1] = x_1[k] \\
x_1[k+1] = -0.346x_0[k] + 1.064x_1[k] + 0.035u_0[k] - 0.568x_0[k]x_1[k] \\
+0.122x_0[k]u_0[k] + 0.559x_1[k]^2 - 0.056x_1[k]u_0[k] - 0.425x_0[k]^3 + 1.136x_0[k]^2x_1[k] \\
-0.138x_0[k]^2u_0[k] - 0.061x_0[k]x_1[k]^2 - 0.564x_1[k]^3 + 0.094x_1[k]^2u_0[k] + 0.255x_0[k]^4 \\
-0.522x_0[k]^3x_1[k] + 0.087x_0[k]^2x1[k]u_0[k] + 0.246x_0[k]x_1[k]^3 - 0.080x_0[k]x_1[k]^2u_0[k]
\end{cases}
$$

Os modelos de ordem 5, assim como os modelos de ordem 3 e 4, representam bem o sistema do motor-gerador. Porém houve um aumento no número de termos das equações identificadas. A Figura 3.17, apresenta as saídas dos modelos identificados para os parâmetros de *limiares* 0.0005, 0.005 e 0.02.

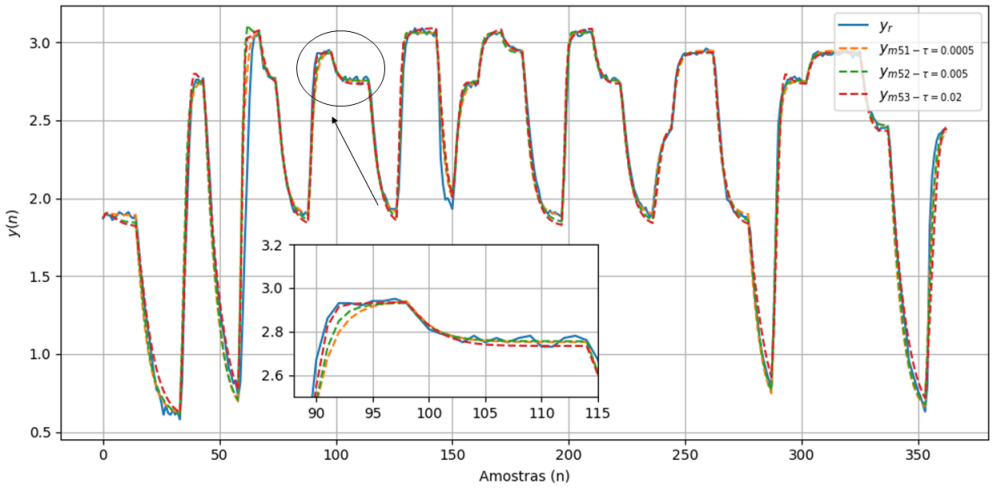

Figura 3.17 – Comparação entre a saída real y_r e as saídas do modelo não linear identificado de ordem 5 (y_{m51},y_{m52},y_{m53}) utilizando o pacote PySINDy.

Modelos não-lineares de ordem 5:

fit=98.74, *Limiar*=0.0005, n_t=49

$$y_{m51}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = 0.0011 + 5.720x_0[k] - 6.641x_1[k] + 0.169u_0[k] - 2.592x_0[k]^2 \\ +4.549x_0[k]x_1[k] - 2.697x_0[k]u_0[k] - 0.660x_1[k]^2 + 3.187x_1[k]u_0[k] - 0.030u_0[k]^2 \\ +1.581x_0[k]^3 - 0.639x_0[k]^2x_1[k] - 1.236x_0[k]^2u_0[k] - 5.353x_0[k]x_1[k]^2 \\ +3.813x_0[k]x_1[k]u_0[k] + 0.223x_0[k]u_0[k]^2 + 3.992x_1[k]^3 - 2.916x_1[k]^2u_0[k] \\ -0.243x_1[k]u_0[k]^2 + 0.002u_0[k]^3 + 5.230x_0[k]^4 - 23.627x_0[k]^3x_1[k] \\ +0.326x_0[k]^3u_0[k] + 36.435x_0[k]^2x_1[k]^2 - 0.284x_0[k]^2x_1[k]u_0[k] + 0.096x_0[k]^2u_0[k]^2 \\ -21.912x_0[k]x_1[k]^3 - 0.553x_0[k]x_1[k]^2u_0[k] - 0.328x_0[k]x_1[k]u_0[k]^2 + 3.963x_1[k]^4 \\ +0.585x_1[k]^3u_0[k] + 0.246x_1[k]^2u_0[k]^2 - 1.378x_0[k]^5 + 7.323x_0[k]^4x_1[k] \\ -0.932x_0[k]^4u_0[k] - 14.193x_0[k]^3x_1[k]^2 + 3.720x_0[k]^3x_1[k]u_0[k] - 0.080x_0[k]^3u_0[k]^2 \\ +12.526x_0[k]^2x_1[k]^3 - 5.263x_0[k]^2x_1[k]^2u_0[k] + 0.143x_0[k]^2x_1[k]u_0[k]^2 \\ -5.000x_0[k]x_1[k]^4 + 3.020x_0[k]x_1[k]^3u_0[k] - 0.003x_0[k]x_1[k]u_0[k]^3 + 0.714x_1[k]^5 \\ -0.554x_1[k]^4u_0[k] - 0.064x_1[k]^3u_0[k]^2 + 0.003x_1[k]^2u_0[k]^3 \end{cases}$$

fit=98.64, *Limiar*=0.005, n_t=30

$$y_{m52}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = 1.319x_0[k] - 1.004x_1[k] - 1.319x_0[k]x_1[k] \\ -0.143x_0[k]u_0[k] + 1.403x_1[k]^2 + 0.487x_1[k]u_0[k] + 0.189x_0[k]^3 - 0.461x_0[k]^2u_0[k] \\ -0.117x_0[k]x_1[k]^2 + 0.814x_0[k]x_1[k]u_0[k] + 0.251x_1[k]^3 - 0.664x_1[k]^2u_0[k] \\ -0.006x_1[k]u_0[k]^2 - 0.151x_0[k]^4 + 0.274x_0[k]^3x_1[k] - 0.155x_0[k]^3u_0[k] \\ +0.694x_0[k]^2x_1[k]u_0[k] - 0.006x_0[k]^2u_0[k]^2 - 0.291x_0[k]x_1[k]^3 \\ -0.787x_0[k]x_1[k]^2u_0[k] + 0.008x_0[k]x_1[k]u_0[k]^2 - 0.011x_1[k]^4 + 0.360x_1[k]^3u_0[k] \\ -0.161x_0[k]^4x_1[k] + 1.031x_0[k]^3x_1[k]^2 - 2.197x_0[k]^2x_1[k]^3 + 1.924x_0[k]x_1[k]^4 \\ -0.016x_0[k]x_1[k]^3u_0[k] - 0.569x_1[k]^5 \end{cases}$$

fit=98.61, *Limiar*=0.02, n_t=14

$$y_{m53}\begin{cases} x_0[k+1] = x_1[k] \\ x_1[k+1] = -0.087x_0[k] + 0.688x_1[k] + 0.038u[k] + 0.135x_0[k]x_1[k] \\ -0.039x_0[k]u[k] + 0.075x_1[k]^2 + 0.086x_1[k]u[k] - 0.038x_0[k]^3 - 0.029x_0[k]^2u_0[k] \\ +0.049x_0[k]^3u_0[k] - 0.111x_0[k]^2x_1[k]u_0[k] + 0.106x_0[k]x_1[k]^2u0[k] - 0.040x_1[k]^3u_0[k] \end{cases}$$

3.5.3 Análise dos resultados do modelo não-linear

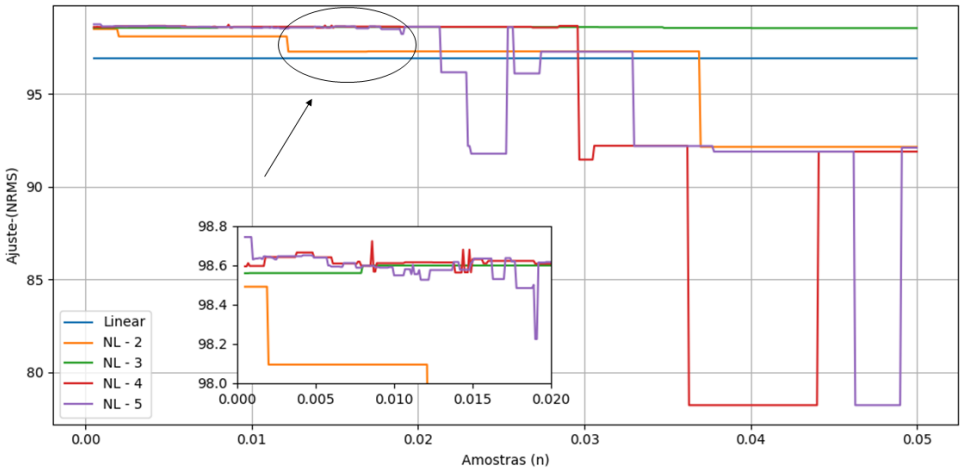

Figura 3.18 – Mapeamento dos valores de *limiar* para os modelos SINDy identificados.

Na Figura 3.18 é apresentado o mapeamento dos valores *limiar* para os modelos linear, de ordem 2, 3, 4 e 5. Com isso podemos verificar a interferência desse parâmetro nos modelos identificados. Para os modelos de ordem 2, 4 e 5, conforme aumentamos o valor de *limiar* fit diminui.Desta forma, podemos notar que para certos valores *limiar*, o modelo de ordem 3, por exemplo, representa melhor o sistema do que o de ordem 5. Em algumas regiões esse valor permanece constante, o que significa a existência de um limite para esse parâmetro que não acrescenta nem diminui termos das equações identificadas.

A Figura 3.19 apresenta o modelo linear e modelos selecionados que possuem os melhores *fit*. Os modelos y_{m22}, y_{m33}, y_{m43} e y_{m53} são os que conseguem melhor representar o sistema tanto nas partes superiores quanto nas inferiores. Assim, a escolha do modelo fica a critério do tipo de aplicação que se deseja ter ou do custo para o sistema que se esteja disposto a empregar.

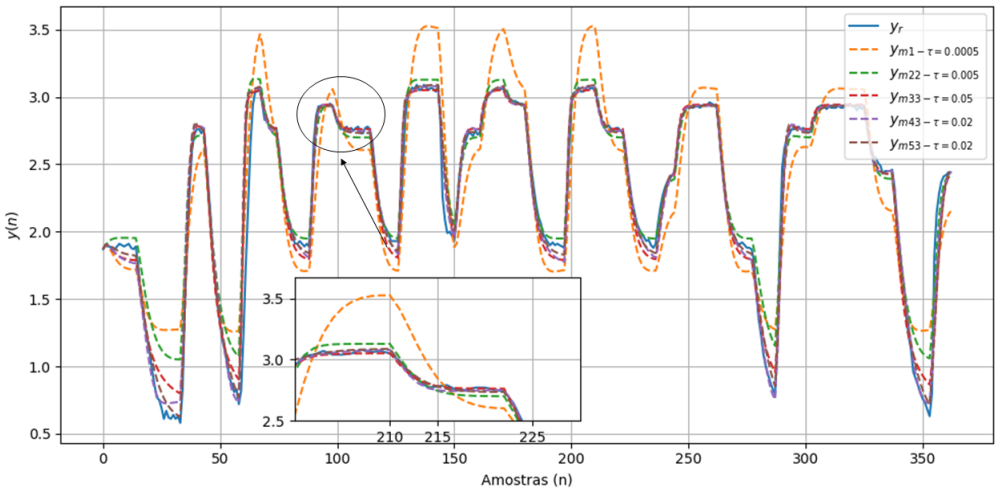

Figura 3.19 – Comparação entre o modelo linear e os melhores modelos identificados selecionados.

4 CONCLUSÃO

Neste trabalho foi apresentado um método baseado em dados para descobrir modelos dinâmicos esparsos.O método foi aplicado em três sistemas clássicos : O sistema de Lorenz, de Van der Pol e Duffing. Foi mostrando que o método de identificação esparsa de dinâmica não-linear é capaz de identificar esses modelos conhecidos. Após mostrar a eficácia do metodo na identificação aplicados a esses modelos. O SINDy foi usado para identificar o modelo de um sistema dinâmico real, a bancada motor-gerador.

Primeiro foi realizado o ensaio linear para esse sistema real e apresentada na Seção 3.2 e levantada a curva que mostrou o comportamento não-linear do sistema, reforçando a escolha deste para aplicação do método SINDy. Após isso, foram mostrados e discutidos os resultados obtidos para o experimento linear. O modelo linear foi avaliado para demonstrar que o comportamento do sistema só é linear próximo a vizinhança dos pontos de operação escolhidos para identificação. Foram identificados cinco modelos e realizada a validação cruzada por meio do mapa de calor para avaliar o desempenho desses modelos quando testados para os dados de validação coletados em diferentes pontos de operação, conforme explicado na Subseção 3.4.2, cada modelo foi testado individualmente para cada conjunto de dados.

Avaliando o parâmetro *fit* para os modelos lineares identificados e testados para conjuntos de dados diferentes que o gerou (diferentes pontos de operação), e embora os modelos encontrados tendam a ajustar os dados relativamente bem, foi observado que estes não apresentaram desempenho tão bom quando comparados e testados para seu próprio conjunto de dados(em seu próprio ponto de operação). Ainda assim, foi avaliado que os modelos com melhores ajustes poderiam ser encontrados, afinal, o modelo é linear apenas nessa região e possui limitação em suas curvas ao tentar se ajustar ao modelo.

Assim, buscou-se encontrar modelos que fossem capaz de identificar esse sistema e capturar sei comportamento de forma mais flexível e ampla.

Dessa modo, na Subseção 3.5.2 foi apresentado os resultados da identificação de modelos não lineares. Delimitamos o SINDy encontrar modelos 2 à quinta ordem para diferentes *limiar*. Avaliando os resultados apresentados, o metodo de identificação esparsa de dinâmica não linear (SINDy) é eficaz para identificar modelos não lineares à partir de dados de um sistema real. Permitindo um espaço de busca para modelos de ordem maiores, utilizando otimização e assim fornecendo representações melhores e com termos significativos que conseguem explicar de forma esparsa sistemas complexos como a bancada motor-gerador.

A Apêndice 1

```
def PRBS(size_min_seq, size):
  np.random.seed(12)
  rand = np.random.randint(0,10,size=size)
  prbs = []
  for i in range(size):
  if rand[i] > 5:
  prbs.append(np.ones(size_min_seq))
  else:
  prbs.append(np.zeros(size_min_seq))
  prbs = np.array(prbs).reshape(size*size_min_seq,)[0:size]
return prbs
u = 1*(2*PRBS(size_min_seq=7, size=1500)-1)+setpoint
numAmostras = len(u)
tempo = np.zeros(numAmostras)
y = np.zeros(numAmostras)
toc = np.zeros(numAmostras)
conexao = serial.Serial(port='COM4', baudrate=9600, timeout=0.005)
t.sleep(1)
  for n in range(numAmostras):
  tic = t.time()
  if (conexao.inWaiting() > 0):
  y[n] = conexao.readline().decode()
  conexao.write(str(round(u[n]*255/15)).encode())
  t.sleep(0.03)
  if (n > 0):
  tempo[n] = tempo[n-1] + Ts
  toc[n] = t.time() - tic
  conexao.write('0'.encode())
  conexao.close()
```

Código A.1 – Implementação em Python do sinal de excitação PRBS para coleta de dados do experimento linear.

B Apêndice 2

```
def PRMLS(nl):
  np.random.seed(20)
  u = np.zeros((5,1))[:,0]
  niveis = [3, 5, 7, 9, 11, 13]
  repeticoes = np.arange(5, 20)
  for n in np.arange(0,nl):
  L = np.random.choice(niveis,size=1)
  rL = np.random.choice(repeticoe,size=1)
  un = L*np.ones(rL)
  u = np.concatenate((u,un))
return u

N=100
u=PRMLS(N)
numAmostras = len(u)
tempo = np.zeros(numAmostras)
y = np.zeros(numAmostras)
toc = np.zeros(numAmostras)
conexao = serial.Serial(port='COM4', baudrate=9600, timeout=0.005)
t.sleep(1)
  for n in range(numAmostras):
  tic = t.time()
  if (conexao.inWaiting() > 0):
  y[n] = conexao.readline().decode()
  conexao.write(str(round(u[n]*255/15)).encode())
  t.sleep(0.03)
  if (n > 0):
  tempo[n] = tempo[n-1] + Ts
  toc[n] = t.time() - tic
  conexao.write('0'.encode())
  conexao.close()
```

Código B.1 – Implementação em Python do sinal de excitação PRMLS para coleta de dados do experimento linear.

REFERÊNCIAS

AGUIRRE, L. A. **Introdução à identificação de sistemas–Técnicas lineares e não-lineares aplicadas a sistemas reais**. [S.l.]: Editora UFMG, 2004. Citado 2 vezes nas páginas 9 e 11.

AGUIRRE, L. A.; RODRIGUES, G. G.; JÁCOME, C. R. Identificação de sistemas não lineares utilizando modelos narmax polinomiais–uma revisão e novos resultados. **SBA Controle e Automação**, v. 9, n. 2, p. 90–106, 1998. Citado 2 vezes nas páginas 2 e 7.

BRANDOLTA, H. G. **SIMULAÇÃO DE ESCOAMENTO EM DUTOS POR CARACTERIZAÇÃO DE EVENTOS**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2002. Citado na página 11.

BRUNTON, S. L.; PROCTOR, J. L.; KUTZ, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 113, n. 15, p. 3932–3937, 2016. Citado 3 vezes nas páginas 5, 4 e 25.

BRUTON, S. L.; KUTZ, J. N. **Data-Driven Science and Engineering Machine Learning, Dynamical Systems, and Control**. [S.l.: s.n.], 2019. Citado 2 vezes nas páginas 18 e 21.

COELHO, A. A. R. **Identificação de sistemas dinâmicos lineares, 1a. edição**. [S.l.]: Florianópolis: Ed. da UFSC, 2004. Citado 2 vezes nas páginas 5 e 9.

LJUNG, L. **System identification, 2a. edição**. [S.l.]: Prentice Hall, 1999. Citado 2 vezes nas páginas 8 e 9.

NELLES, O. Nonlinear dynamic system identification. In: **Nonlinear System Identification**. [S.l.]: Springer, 2001. p. 547–577. Citado na página 10.

OGATA, K. **Engenharia de Controle Moderno, 5a. edição**. [S.l.]: São Paulo : Pearson Prentice Hall, 2010. Citado na página 7.

ROZMAN. **nonlinear oscillators. method of averaging**. [S.l.: s.n.], 2020. Citado na página 28.

SILVA, B. **Data-driven discovery and model reduction of complex systems**. Dissertação (Mestrado) — University of Washington, 2020. Citado na página 2.

SÖDERSTRÖM, T.; STOICA, P. **System identification , 2a. edição**. [S.l.]: Prentice Hall, 1989. Citado 2 vezes nas páginas 5 e 10.

WELLS D. E. KRAKIWSKY, E. J. **THE METHOD OF LEAST SQUARES**. [S.l.: s.n.], 1997. Citado na página 11.