

PYDAQ + SysIdentPy: integração de bibliotecas para identificação de sistemas em tempo real

Rafaella Faria Araújo Cunha * Samir Angelo Milani Martins *

* GCoM - Grupo de Controle e Modelagem. Universidade Federal de São João del-Rei. Departamento de Engenharia Elétrica. São João del-Rei, MG, Brasil. (e-mails: rafaellafaria135@aluno.ufsj.edu.br, martins@ufsj.edu.br).

Abstract: Mathematical models are fundamental in science as they allow the representation and understanding of real systems. Aiming to simplify their development, a new feature was implemented in *PYDAQ*, a Python library dedicated to empirical data acquisition. This new functionality integrates *SysIdentPy* into *PYDAQ*, a Python library developed for model identification based on the NARMAX representation. It enables parameter estimation, structure definition, and model validation directly from the collected data. This integration allows system identification through real-time data acquisition, optimizing the analysis of dynamic systems. To validate the solution, a representative model of the Datapool® 2208 Servo Mechanism Educational Module was obtained, achieving validation metrics that demonstrate the effectiveness of the library, such as a Root Relative Squared Error (RRSE) of 0.1177 and a Mean Squared Error (MSE) of 0.0779.

Resumo: Modelos matemáticos são fundamentais na ciência por permitirem representar e compreender sistemas reais. Visando simplificar sua obtenção, uma nova funcionalidade foi implementada ao *PYDAQ*, uma biblioteca em Python dedicada a aquisição empírica de dados. A nova funcionalidade integra ao *PYDAQ* o *SysIdentPy*, biblioteca em Python desenvolvida para identificação de modelos baseados na representação NARMAX, possibilitando a estimação de parâmetros, definição de estrutura e validação dos modelos diretamente a partir dos dados coletados. Essa integração permite realizar a identificação de sistemas a partir da aquisição de dados em tempo real, otimizando a análise de sistemas dinâmicos. Para validar a solução, foi obtido um modelo representativo do Módulo Didático 2208 Datapool® de Servomecanismo com métricas de validação que comprovam a eficácia da biblioteca, como um erro quadrático relativo à raiz (RRSE) de 0,1177 e um erro quadrático médio (MSE) de 0,0779.

Keywords: System Identification; Mathematical Models; Python; *PYDAQ*; *SysIdentPy*.

Palavras-chaves: Identificação de Sistemas; Modelos matemáticos; Python; *PYDAQ*; *SysIdentPy*.

1. INTRODUÇÃO

A área de identificação de sistemas visa obter um modelo matemático que explique, pelo menos de forma aproximada, a relação de causa e efeito presente nos dados (Aguirre, 2007). Seu uso na ciência facilita a observação visual de certas características de partes do modelo real, como na análise aerodinâmica em um túnel de vento de um aeromodelo (Young et al., 2002). Para identificar um modelo matemático que represente os comportamentos de um sistema real, é necessário realizar cinco etapas principais (Martins et al., 2011):

- Teste Dinâmico e Coleta de Dados;
- Escolha da Representação Matemática a ser Utilizada;
- Determinação da Estrutura do Modelo;

- Estimação dos Parâmetros;
- Validação do Modelo.

A programação possibilita a realização de todas as etapas necessárias para modelar um sistema real a partir de dados de entrada e saída. No mercado, existem diversas bibliotecas dedicadas à identificação de sistemas, como o *System Identification Toolbox* do Matlab, amplamente reconhecido pela qualidade e confiabilidade dos resultados. Outra ferramenta de destaque é o *N4SID*, também disponível no Matlab, que se destaca pela robustez, apresentando excelente desempenho mesmo na presença de ruídos e em sistemas com múltiplas entradas e saídas. Mas muitas das bibliotecas já desenvolvidas para identificação de sistemas possui limitações, como a necessidade de amplo conhecimento em programação e o custo para uso. No entanto, o *SysIdentPy* é um módulo de código aberto escrito em Python de identificação de sistemas, utilizando modelos NARMAX polinomiais para sua obtenção, e oferece fer-

* Os autores agradecem ao apoio das agências Brasileiras CAPES, FAPEMIG e CNPq.

ramentas para determinação da estrutura, estimação de parâmetros e também métricas para análise de modelos.

Apesar de sua robustez e facilidade de uso, o *SysIdentPy* depende de dados de entrada e saída previamente adquiridos para seu funcionamento. Desta forma, o *PYDAQ* oferece uma solução integrada com sua funcionalidade de aquisição de dados em tempo real, garantindo dados confiáveis para aquisição do modelo matemático.

O *PYDAQ* é uma biblioteca desenvolvida em Python, criada para experimentos empíricos (Martins, 2023), que, utilizando placas de aquisição de dados, permite obter ou enviar dados de forma simples, seja por meio de uma interface gráfica ou por linhas de código. Criada em 2023, a biblioteca vem passando por constantes aprimoramentos, demonstrando seu alto potencial e aplicabilidade.

Para disponibilizar uma ferramenta mais completa, capaz não apenas de coletar dados, mas também de realizar a identificação de sistemas em tempo real, tornou-se necessária a combinação dessas duas bibliotecas. Essa integração permite tanto a aquisição eficiente dos dados experimentais quanto a modelagem precisa dos sistemas, oferecendo uma solução robusta para análise e otimização de sistemas dinâmicos.

Por conseguinte, a integração das bibliotecas *SysIdentPy* e *PYDAQ* resulta em uma ferramenta poderosa para a aquisição de dados e identificação de modelos matemáticos que representem sistemas reais da melhor forma possível. Utilizando placas de aquisição disponíveis no mercado, como as do Arduino e da National Instruments, a integração é capaz de obter um modelo matemático em poucos cliques. Além disso, sua interface gráfica intuitiva torna o uso acessível até mesmo para usuários com pouca experiência, facilitando todo o processo de modelagem e análise de sistemas dinâmicos.

2. CONCEITOS PRELIMINARES

2.1 Identificação de sistemas

A obtenção de um modelo matemático é dividida em cinco etapas principais, pelas quais se passa para obter um modelo representativo.

Obtenção de dados A primeira etapa da identificação de sistemas consiste na coleta de dados a partir de experimentos realizados diretamente no sistema. Isso é feito aplicando um conjunto de entradas conhecidas e registrando o sinal das saídas. Esses dados são essenciais tanto para identificar a estrutura do modelo quanto para ajustar os seus parâmetros.

As entradas utilizadas para excitar o sistema devem ser projetadas para garantir a adequação dos dados obtidos (Billings, 2013). O sinal de entrada deve excitar o sistema de forma a garantir que todas as não linearidades sejam exploradas. Um sinal comum para excitar sistemas é o sinal binário pseudo-aleatório (PRBS), que é periódico e gerado de forma determinística, mas apresenta características de um sinal aleatório, com um espectro de frequências amplo (Aguirre, 2007). Seu uso na área de identificação de sistemas é consolidado, com diversas aplicações práticas,

como na identificação de parâmetros de baterias VRLA usando PRBS (Gomes et al., 2010).

Para definir um sinal PRBS, é necessário especificar sua *seed* e o número de bits (*nbits*). A *seed* determina as condições iniciais do sinal, enquanto o número de bits define a periodicidade do sinal PRBS, que se repete a cada $2^{nbits} - 1$ ciclos. Essa combinação de propriedades torna os sinais PRBS uma ferramenta valiosa para excitar o sistema e explorar suas não linearidades, proporcionando dados que cobrem uma ampla gama de comportamentos dinâmicos.

Escolha da representação matemática A representação matemática de um sistema é um conjunto de equações que descrevem seu comportamento real. Alguns tipos de representações possíveis incluem funções trigonométricas, conjuntos de polinômios ou equações diferenciais.

Os modelos temporais discretos NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous Input) descrevem o comportamento de sistemas a partir de valores passados de entrada e saída (Oroski, 2015; Cadenas et al., 2015). Esse tipo de modelo, assim como seus casos particulares (NARX e NAR), é bem documentado e amplamente utilizado, pois a dinâmica da maioria dos sistemas reais pode ser aproximada por uma formulação não linear (Oroski, 2015; Furtado et al., 2002; Rodrigues, 1996; Aguirre and Billings, 1995; Chen and Billings, 1989).

Os modelos NARX podem ser escritos como:

$$y(k) = F^\ell[y(k-1), \dots, y(k-n_y), u(k-1), \dots, u(k-n_u), e(k-1), \dots, e(k-n_e)] + \Xi(k), \quad (1)$$

em que ℓ é o grau de não linearidade do modelo, $y(k)$ é a saída, $u(k)$ é a entrada e $e(k)$ é o sinal de ruído. $\Xi(k)$ representa o erro do processo de predição e também temos n_y , n_u e n_e que são os respectivos atrasos. F é uma função não linear dos regressores, sendo, neste trabalho, restrita a funções polinomiais devido ao foco do pacote *SysIdentPy* nos modelos NARX polinomiais. As principais vantagens dessas representações incluem a facilidade de recuperar as características estáticas de um sistema (Coelho et al., 2002), menor sensibilidade ao ruído, linearidade nos parâmetros (Aguirre, 2007) e a necessidade de uma quantidade menor de dados para sua obtenção.

Determinação da Estrutura do Modelo A determinação da estrutura é crucial para que o comportamento do modelo represente adequadamente a realidade, pois é nesta etapa que os termos da equação são definidos, como os graus do polinômio ou as componentes de frequência em uma série de Fourier. A seleção inadequada de um termo pode alterar significativamente a resposta da saída gerada pelo modelo.

No projeto de integração dos pacotes, adotou-se o critério de informação de Akaike (AIC) (Billings and Chen, 1989), que, junto ao algoritmo de taxa de redução de erro (ERR), quantifica a importância de cada termo e usa a estimativa de máxima verossimilhança para ajustes no modelo. Essa abordagem seleciona os componentes que melhor se ajustam, garantindo uma saída do modelo próxima à real (Corrêa, 1997).

Determinação de Parâmetros Na etapa de estimação de parâmetros, o objetivo é ajustar o modelo aos dados experimentais, garantindo que ele represente com precisão o comportamento dinâmico do sistema.

Uma das técnicas mais utilizadas para essa finalidade é o método de mínimos quadrados, que minimiza a soma dos quadrados das diferenças (erros) entre a saída prevista pelo modelo e os dados reais. Essa abordagem busca reduzir ao máximo o impacto de discrepâncias entre o modelo e o sistema, ajustando os parâmetros de forma a tornar a resposta simulada o mais próxima possível da resposta real (Aguirre, 2007).

Validação do modelo A validação é a etapa final do processo de identificação de sistemas, na qual o modelo matemático gerado é avaliado por meio de métricas e testes para garantir que represente adequadamente o sistema real. Essa fase é fundamental, pois, mesmo que o modelo tenha sido ajustado aos dados experimentais na etapa de identificação, é necessário verificar se ele mantém um desempenho consistente e preciso quando exposto a novos dados, ou seja, aqueles que não foram utilizados durante a modelagem inicial.

Diversos métodos podem ser empregados nessa etapa para avaliar a qualidade do modelo. Um dos mais comuns é a predição livre, na qual o modelo é utilizado para prever o comportamento do sistema ao longo do tempo sem depender diretamente dos dados de entrada. Esse método permite verificar a capacidade do modelo de reproduzir padrões dinâmicos e responder com precisão a variações na entrada. Além disso, pode-se realizar a análise das propriedades topológicas do sistema, examinando se o modelo preserva características estruturais, como estabilidade, comportamento oscilatório e outras dinâmicas específicas.

Outra métrica para avaliar a precisão de um modelo é o erro quadrático médio relativo (RRSE), no qual os valores reais são comparados com os valores previstos pelo modelo. Esse indicador quantifica o desvio entre a saída estimada e a saída real, normalizando os erros em relação à variabilidade dos dados observados:

$$RRSE = \frac{\sqrt{\sum_{k=1}^n (y_k - \hat{y}_k)^2}}{\sqrt{\sum_{k=1}^n (y_k - \bar{y})^2}}, \quad (2)$$

onde \bar{y} representa a média da série temporal e \hat{y}_k é a saída estimada pelo modelo. Além do RRSE, outras métricas de validação estão disponíveis na documentação do *SysIdentPy* (Lacerda Junior et al., 2020).

2.2 PYDAQ: Aquisição de Dados com Interface Gráfica

A obtenção simplificada de dados é essencial para otimizar processos e viabilizar análises mais eficientes em projetos de engenharia. O *PYDAQ* é uma biblioteca escrita em Python para atender a essa necessidade (Martins, 2023). Inicialmente, o pacote possuía três funções principais: aquisição de dados, envio de dados e resposta ao degrau.

Um dos principais diferenciais do *PYDAQ* é sua interface simples e intuitiva, que permite aos usuários personalizar e executar experimentos de maneira rápida e eficiente, mesmo sem amplo conhecimento técnico em programação.

Além disso, é compatível com uma ampla variedade de dispositivos de aquisição, desde soluções acessíveis, como placas Arduino, até sistemas de maior complexidade e precisão, como as placas NIDAQ (National Instruments Data Acquisition).

Em março de 2025, o *PYDAQ* registra uma média de 547 downloads mensais, sendo amplamente utilizado em cenários de ensino e pesquisa em áreas correlatas. O pacote se mantém em constante evolução e tem sido aprimorado com novas funcionalidades, como a filtragem de sinal e ferramentas de controle. A biblioteca do *PYDAQ* foi publicada na *The Journal of Open Source Software* (Martins, 2023), sendo reconhecida como uma ferramenta de relevante importância científica e como uma solução promissora para a pesquisa e o ensino de Engenharia nos próximos anos.

É possível se obter mais informações sobre a biblioteca no site www.pydaq.org, com detalhes de como utilizar a ferramenta e exemplos práticos.

2.3 SysIdentPy: Identificação de Sistemas com Modelos NARMAX

O *SysIdentPy* (Lacerda Junior et al., 2020) é uma biblioteca desenvolvida em Python para identificação de sistemas utilizando modelos NARMAX Polinomiais (Nonlinear AutoRegressive Moving Average with eXogenous Inputs). Em março de 2025, conta com 1100 downloads mensais, foi publicada na *The Journal of Open Source Software* (Lacerda Junior et al., 2020) e citada na *Nature Communications* (Kent et al., 2024), evidenciando a importância técnica e relevância do projeto.

Dentro do *SysIdentPy*, é possível escolher diferentes métodos para cada etapa do processo de modelagem. Por exemplo, na seleção da estrutura do modelo, a biblioteca implementa algoritmos como a Regressão Progressiva por Mínimos Quadrados Ortogonais (FROLS) e a Seleção da Estrutura do Meta-Modelo (Meta-MSS). Para a estimação de parâmetros, o pacote oferece desde os tradicionais mínimos quadrados até algoritmos mais sofisticados, como os mínimos quadrados recursivos e os mínimos quadrados normalizados. Além disso, o pacote permite a simulação de modelos pré-definidos e a validação por meio de diversas métricas. No site do *SysIdentPy* (www.sysidentpy.org), é possível obter mais informações sobre a biblioteca, incluindo detalhes das funcionalidades e exemplos de aplicação.

Embora seja um pacote bastante completo, o *SysIdentPy* opera exclusivamente *offline*, exigindo que os dados sejam adquiridos previamente em outro ambiente para a obtenção do modelo matemático.

3. METODOLOGIA

A integração das duas bibliotecas tem como foco principal facilitar o acesso à aquisição de modelos matemáticos. Nesse sentido, foi proposta uma nova funcionalidade para o *PYDAQ*: “*Get Model*”, que aplica um sinal de entrada controlado no sistema, obtém a saída correspondente e, usando das etapas de identificação de sistemas do *SysIdentPy* é possível obter um modelo matemático.

O desenvolvimento da estrutura dessa funcionalidade foi realizado em etapas, visando manter a biblioteca otimizada, sem comprometer a simplicidade e a intuitividade no uso. O algoritmo a seguir descreve as etapas de funcionamento do código desenvolvido.

Algorithm 1 Desenvolvimento do código da funcionalidade *Get Model*

- 1: Configuração dos parâmetros de aquisição de dados.
 - 1.1: Configuração do sinal PRBS.
- 2: Geração do sinal de entrada.
 - 2.1: O sinal é gerado de acordo com o número de ciclos necessários, determinado pela duração da sessão e o tempo de amostragem.
- 3: Configuração da comunicação serial.
 - 3.1: Abre a porta serial e inicializa a comunicação com o microcontrolador.
 - 3.2: Garante a limpeza dos buffers antes do início da transmissão.
- 4: Aquisição dos dados.
 - 4.1: Envia cada amostra do sinal PRBS ao microcontrolador.
 - 4.2: Lê a resposta do sistema (saída) via porta serial.
 - 4.3: Armazena os dados de entrada, saída e o tempo.
 - 4.4: Apresenta os dados em um gráfico dinâmico durante a aquisição.
- 5: Salva os dados adquiridos em um arquivo .txt.
- 6: Preparação dos dados para modelagem
 - 6.1: Desconsidera os dados iniciais, conforme o tempo de espera escolhido.
 - 6.2: Divide os dados em conjuntos de treinamento e validação com base no percentual escolhido.
- 7: Uso da biblioteca SysIdentPy para identificação do modelo matemático.
- 8: Exibição dos resultados.
- 9: Encerramento da comunicação serial.

Por conseguinte, no Fluxograma 1, está demonstrado o fluxo de utilização da funcionalidade *Get Model*.

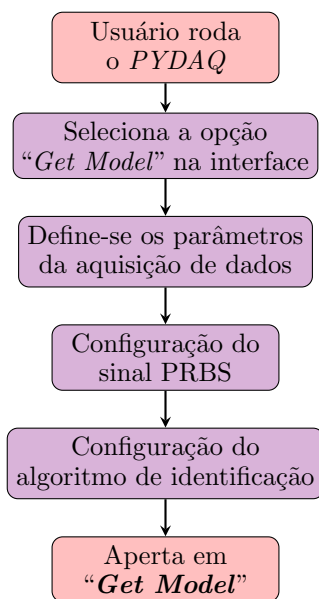


Figura 1. Fluxo de utilização da funcionalidade *Get Model* no PYDAQ.

Outro ponto importante considerado foi a utilização de valores padrão, de modo que a configuração do sinal PRBS e do algoritmo de identificação fosse opcional. Ou seja, o usuário pode simplesmente conectar sua placa de aquisição e executar o *PYDAQ* para obter um modelo inicial sem precisar fazer ajustes. Caso necessário, esses valores padrão oferecem um bom ponto de partida para aprimorar a modelagem do sistema em questão.

4. RESULTADOS

Foram criadas duas novas janelas para a aquisição de modelos matemáticos: uma para placas Arduino e outra para placas da National Instruments (NIDAQ). Cada uma delas possui opções de personalização adaptadas ao respectivo tipo de *hardware*. A Figura 2 apresenta a interface desenvolvida para a funcionalidade *Get Model* na opção de placas Arduino. A interface para placas NIDAQ é similar, diferindo apenas em algumas configurações específicas para esse tipo de dispositivo.

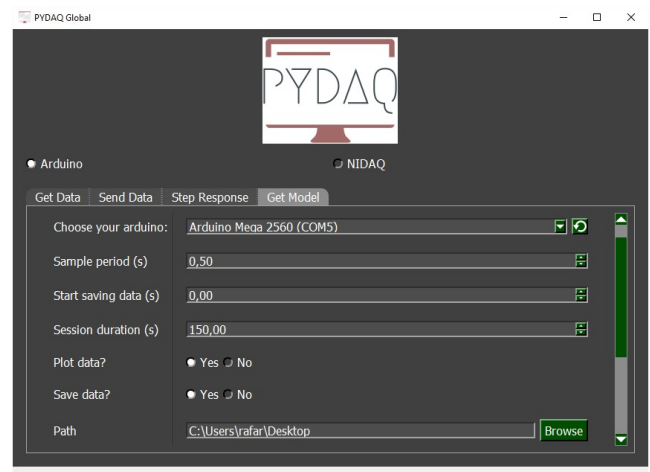


Figura 2. Interface do *Get Model* Arduino.

Na tela inicial, é possível personalizar os parâmetros de aquisição de dados, como o tempo total da sessão, o período de amostragem (intervalo entre duas leituras consecutivas) e o tempo para início da aquisição, caso seja necessário eliminar o transitório inicial do sistema. Além disso, o usuário pode visualizar o processo de aquisição em tempo real e salvar os dados coletados.

Na seção “Input Signal”, é possível configurar o sinal PRBS utilizado para obter o modelo matemático, permitindo alterar o número de bits, a *seed* e o tempo entre bits. No botão **Advanced Settings**, o usuário pode personalizar as características do algoritmo de identificação do *SysIdentPy* (Figura 3), como o grau do polinômio, o atraso dos regressores de entrada e saída, o estimador de parâmetros e a porcentagem dos dados a serem utilizados para validação do modelo (esses dados não serão utilizados no processo de treinamento do modelo, mas apenas para o cálculo das métricas e outras análises).

System Identification

Degree: 2

Output Lag: 3

Input Lag: 3

Number of information values: 6

Estimator: Least squares

Extended least squares algorithm: ☐ True ☒ False

Percentage of data for validation: 30

Figura 3. Tela de personalização dos parâmetros de aquisição do modelo.

Quando o usuário pressionar o botão **Get Model** no final da tela, os dados do sistema serão adquiridos, e, ao término, o modelo matemático será exibido, acompanhado dos gráficos de validação e da tabela de métricas.

A fim de demonstrar a funcionalidade **Get Model**, foi realizado um exemplo prático para a identificação de um sistema real. Para isso, foi utilizada uma placa da National Instruments (NIDAQ 6009) na obtenção do modelo matemático de um módulo didático 2208 Datapool de Servomecanismo (Figura 4), em tempo real.



Figura 4. Módulo didático 2208 Datapool de Servomecanismo (Datapool, 2025).

Este sistema contém um servomotor que é acionado por um sinal de tensão (entrada do sistema), e que tem a sua posição angular monitorada por um sensor do tipo tacômetro de velocidade angular (saída do sistema), os dados obtidos do ensaio podem ser observados na Figura 5.

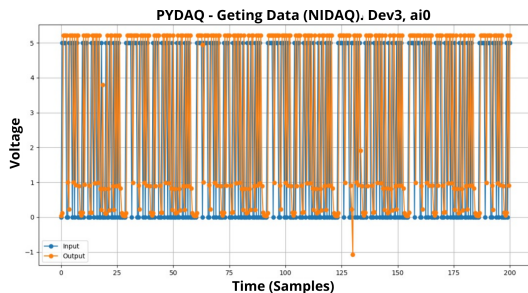


Figura 5. Dados de entrada e saída, ampliado, do Módulo didático 2208 Datapool de Servomecanismo usando a placa NIDAQ 6009.

Após o período de aquisição, o algoritmo calcula o modelo e apresenta ao usuário as janelas com os resultados: o gráfico comparativo entre os dados reais e os dados preditos pelo modelo (sendo que esses dados de validação não foram usados no treinamento), uma tabela com as 10 métricas disponíveis no *SysIdentPy* (mais informações no livro Lacerda Junior (2024)), e os gráficos de autocorrelação dos resíduos (inferior esquerdo) e correlação cruzada dos resíduos (inferior direito).

Na Figura 6, é apresentada a janela de resultados do modelo do sistema de Servomecanismo do módulo didático Datapool. Um intervalo de confiança de 95% é mostrado em azul claro, o que indica a correlação, ou a falta dela, entre as variáveis e o desempenho do modelo obtido. De acordo com a teoria de identificação de sistemas, a autocorrelação dos resíduos deve se aproximar de um impulso unitário, enquanto a correlação cruzada entre os resíduos e a entrada deve ser estatisticamente nula para qualquer atraso analisado, caso o modelo seja representativo.

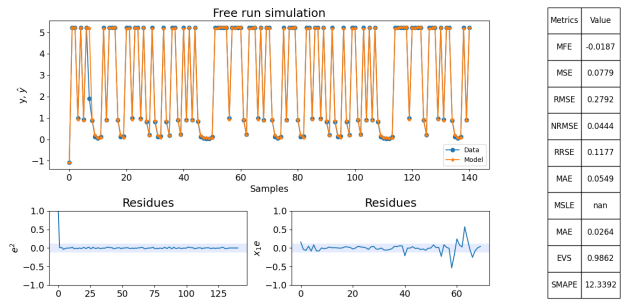


Figura 6. Exemplo de Gráficos de Validação e Resíduos do modelo matemático obtido utilizando a placa NIDAQ 6009.

Em outra janela, é exibida a equação do modelo matemático retornado pelo *SysIdentPy* (Figura 7), permitindo uma análise detalhada de suas características, onde, x_y é a entrada (sinal de tensão aplicado) e y_k é a saída (velocidade angular do mecanismo).

Mathematical Model

$$y_k = 1.0224x_{k-1} + 0.1595y_{k-1} - 0.0217x_{k-1}y_{k-1} + 0.0545 + 0.0015x_{k-2}y_{k-2} - 0.0096x_{k-2}x_{k-1}$$

Figura 7. Exemplo de equação do modelo matemático obtido utilizando a placa NIDAQ 6009.

Neste ponto, a obtenção do modelo matemático está concluída. Após todas as etapas de identificação de sistemas, é possível observar que o modelo obtido do exemplo prático é válido e robusto. A partir dos valores obtidos nas métricas, pode-se perceber uma boa correlação entre o modelo gerado e o sistema real.

Com esta nova funcionalidade, até mesmo usuários sem experiência em modelagem matemática podem, com apenas

alguns cliques, obter modelos complexos para representar sistemas reais. De forma simplificada, também é possível personalizar as configurações do algoritmo para refinar a identificação e ajustar o modelo conforme as necessidades específicas. Esse avanço representa um ganho considerável no ensino, pesquisa e extensão em áreas que demandam o conhecimento de modelagem de sistemas. Exemplos de aplicações vão desde experimentos em Engenharia Elétrica, ensaios em campo (plantas industriais ou laboratórios), até áreas como biologia, robótica e aeronáutica.

5. CONCLUSÃO

A área de identificação de sistemas é suma importância para a ciência, auxiliando no desenvolvimento de métodos que facilitam processos de predição e controle de cenários reais. Este trabalho aproxima a área de identificação de sistemas para estudantes e pesquisadores iniciantes na área, facilitando o acesso à aquisição de modelos matemáticos.

Ao utilizar a função *Get Model*, que combina as funcionalidades de identificação de sistemas com modelos polinomiais NARMAX do pacote *SysIdentPy* no *PYDAQ*, é possível obter modelos matemáticos representativos de alta confiabilidade.

Um sistema de servomecanismo foi identificado por meio da nova funcionalidade do *PYDAQ*, apresentando um erro quadrático relativo à raiz (RRSE) de 0,1177 e um erro quadrático médio (MSE) de 0,0779, indicando que o modelo matemático obtido é de alta qualidade e representa adequadamente o sistema real.

Para trabalhos futuros, é possível expandir a funcionalidade *Get Model*, incorporando novos métodos de seleção de estrutura disponíveis no *SysIdentPy*, identificação de sistemas com múltiplas saídas (SIMO), e a apresentação da transformada Z de modelos lineares. Além disso, seria interessante implementar novos sinais de excitação dos dados, bem como controladores preditivos baseados em modelos.

AGRADECIMENTOS

Os autores agradecem à UFSJ, a FAPEMIG, a CNPq, a CAPES pelo apoio financeiro para a realização deste trabalho.

REFERÊNCIAS

- Aguirre, L.A. (2007). *Introdução à Identificação de Sistemas – Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. Editora UFMG, Belo Horizonte.
- Aguirre, L.A. and Billings, S.A. (1995). Retrieving dynamical invariants from chaotic data using narmax models. *International Journal of Bifurcation and Chaos*, 5(2), 449–474.
- Billings, S.A. (2013). *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, Nashville, TN, USA.
- Billings, S. and Chen, S. (1989). The determination of multivariable nonlinear models for dynamic systems. *International Journal of Control*, 49(3), 689–710.
- Cadenas, E., Rivera, W., Campos-Amezcu, R., and Cadenas, R. (2015). Wind speed forecasting using the narx model, case: La mata, oaxaca, méxico. *Neural Computing and Applications*, 27(8), 2417–2428.
- Chen, S. and Billings, S.A. (1989). Representations of nonlinear systems: the narmax model. *International Journal of Control*, 49(3), 1013–1032.
- Coelho, M., Aguirre, L., and Cor, M. (2002). Metodologia para representação de modelos narx polinomiais na forma de hammerstein e wiener. *TEMA - Tendências em Matemática Aplicada e Computacional*, 3.
- Corrêa, M.V. (1997). *Identificação de sistemas dinâmicos não-lineares utilizando modelos NARMAX racionais: aplicação a sistemas reais*. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil.
- Datapool (2025). Módulo 2208: Servomecanismo. URL <https://eletronica.datapool.com.br/produtos/controle-e-automacao/modulo-2208-servomecanismo/>. Accessed: 2025-02-19.
- Furtado, E.C., Nepomuceno, E.G., Mendes, E.M.A.M., and Silva, V.V.R. (2002). Identificação de sistemas dinâmicos não-lineares contínuos utilizando modelos narmax: Estudo de caso de um forno a arco elétrico. *XIV Congresso Brasileiro de Automática*, 2150–2155.
- Gomes, J.C.C., de Oliveira, J.C., and de Oliveira, J.C. (2010). Vrla battery parameter identification using pseudo random binary sequences. *IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 1165–1169.
- Kent, R., Barbosa, W., and Gauthier, D. (2024). Controlling chaos using edge computing hardware. *Nature Communications*, 15, 3886.
- Lacerda Junior, W.R., Andrade, L.P.C., Oliveira, S.C.P., and Martins, S.A.M. (2020). Sysidentpy: A python package for system identification using narmax models. *Journal of Open Source Software*, 5(54), 2384.
- Lacerda Junior, W. (2024). *Nonlinear System Identification and Forecasting: Theory and Practice With SysIdentPy*. SysIdentPy. URL <https://sysidentpy.org>. Web version.
- Martins, S.A.M. (2023). Pydaq: Data acquisition and experimental analysis with python. *Journal of Open Source Software*. Em revisão.
- Martins, S.A.M., Nepomuceno, E.G., and M., F.J.P. (2011). Detecção de estruturas de modelos narmax polinomiais: uma abordagem inteligente multi-objetivo. In *Anais do X Simpósio Brasileiro de Automação Inteligente*, 320–325.
- Oroski, E. (2015). *Identificação de sistemas não lineares utilizando modelos NARX, funções ortonormais e otimização heurística*. Ph.D. thesis, Universidade de Brasília (UnB), Brasília, DF, Brasil.
- Rodrigues, G.G. (1996). Identificação de sistemas dinâmicos não-lineares utilizando modelos narmax polinomiais - aplicação a sistemas reais. Dissertação, Universidade Federal de Minas Gerais.
- Young, L.A., Lillie, D., McCluer, M., Yamauchi, G.K., and Derby, M.R. (2002). Insights into airframe aerodynamics and rotor-on-wing interactions from a 0.25-scale tiltrotor wind tunnel model. *NASA Ames Research Center*.