
**UNIVERSIDADE ESTADUAL DE CAMPINAS**

**FACULDADE DE ENGENHARIA MECÂNICA**

**DEPARTAMENTO DE MECÂNICA COMPUTACIONAL**

**Relatório Final de Pesquisa - PIBIC**

# Análise de Robustez em Métodos de Identificação de Sistemas

Aluno: Paulo Yoshio Kuga
RA: 204451

Orientador: Prof. Dr. Alberto Luiz Serpa

Campinas
Setembro de 2023

## Resumo

Este projeto avalia alguns métodos clássicos de identificação de sistemas para sistemas dinâmicos, considerando o efeito de ruídos gaussianos na identificação. Para tal, foi mensurada a robustez dos métodos na identificação de um sistema linear (massa-mola-amortecedor) e para um sistema que possua efeitos não lineares (sistema carro-pêndulo). Nos dois casos, os resultados obtidos com a identificação foram avaliados em termos da resposta temporal e da resposta em frequência. Usualmente os métodos obtém como resposta um sistema linearizado, e assim o efeito do ruído e de não linearidades são considerados aspectos desconhecidos que afetam a identificação, permitindo uma análise crítica da robustez dos métodos, visando o emprego em situações da identificação de danos em sistemas. Neste trabalho, são analisados os algoritmos ARX, N4SID e ERA todos implementados em Julia.


**Palavras Chave:** Identificação de Sistemas, Simulação Computacional, Sistemas dinâmicos, Vibrações de Sistemas Mecânicos.

## 1 Introdução

Nos últimos 50 anos, o desenvolvimento do setor aeroespacial levou à necessidade da criação de novos componentes e mecanismos com comportamentos dinâmicos até então pouco conhecidos. Com isso, foi necessário o desenvolvimento de métodos que permitissem obter uma representação do sistema através do conhecimento da resposta a uma entrada específica, obtendo-se assim uma estimativa para os parâmetros do sistema [8].

Desta forma, métodos como o Algoritimo de Realização de Autovalores (ERA) [3], Algoritimos para Identificação no Espaço de Estados por Subespaços (N4SID) [5], Identificação através do Observador de Kalman (OKID), e Mínimos Quadrados com Entradas Exógenas (ARX) foram desenvolvidos no intuito de atender a esta necessidade [4]. Estes diferentes algoritmos possuem conceitos e formulações distintas, de tal forma que, para um mesmo conjunto de dados, são obtidos modelos com certa diferença para representar o sistema [2].

Com isso, é necessário ter um olhar crítico sobre os métodos, compreender o funcionamento e avaliar seus resultados. Avaliar a robustez dos métodos é um aspecto importante, uma vez que efeitos não lineares e a presença de ruído usualmente são presentes na maioria das situações práticas. O entendimento e quantificação desta robustez permite uma melhor compreensão das limitações dos métodos de identificação, particularmente quando são usados em situações mais críticas da presença de danos em sistemas [8].

Neste trabalho, foi investigada a robustez dos métodos ERA, N4SID e ARX em diferentes condições, procurando averiguar qual a tolerância a efeitos como o ruído gaussiano e efeitos não lineares na identificação de sistemas. Com isso, através de uma avaliação crítica dos resultados da robustez obtidos, através de avaliações nas respostas temporais e em frequência, foi possível compreender critérios que auxiliem na decisão de escolha dentre os algoritmos investigados.

## 2 Objetivos

Neste trabalho, propõe-se como objetivo avaliar alguns métodos clássicos, como o N4SID, GRA (ERA) e ARX, avaliando a robustez ao ruído e capacidade de identificação adequada quando são presentes efeitos não lineares considerados como incertezas do problema. Serão avaliadas as respostas no tempo e em frequência obtidas. As técnicas escolhidas fornecem, como resultado, um modelo linear, e com isso é possível avaliar tanto o efeito do ruído, como o causado pela resposta não linear do sistema. Nessas duas situações são considerados o mapeamento entre a entrada e saída através dos respectivos modelos gerados, que são comparados criticamente aos modelos identificados pelos algoritmos.

Serão considerados dois sistemas dinâmicos para avaliar os resultados dos métodos: 1) Sistema Massa-Mola-Amortecedor, intrinsecamente linear, cujo objetivo é de avaliar preferencialmente o efeito de ruídos na identificação; e 2) Sistema Carro-Pêndulo, que possui efeitos não lineares em seu movimento oscilatório, cujo objetivo é avaliar a capacidade de identificação quando os efeitos não-lineares são aumentados. Nestes dois casos, a capacidade dos métodos em obter o sistema identificado é avaliada, de modo que atualmente, é possível estabelecer as diferenças em relação ao esperado.

## 3 Metodologia

Como dito anteriormente, são considerados dois sistemas dinâmicos para avaliar os resultados dos métodos. Desta forma é preciso avaliar os métodos propostos na literatura, revisando-os, de modo que sua implementação seja fundamentada. Desta forma é apresentada a modelagem matemática dos modelos e posteriormente as definições que fundamentam os conceitos dos algoritmos implementados. Posteriormente são apresentados os desenvolvimentos dos algoritmos. Posteriormente, é estabelecida uma métrica, que mensura a discrepância da resposta gerada pelo modelo identificado, gerada pelo sinal de entrada. Desta forma, é possível avaliar a resposta do sistema identificado, comparado com a resposta original, assim medindo a robustez do algoritmo.

Para o sistema linear, um ruído branco gaussiano é adicionado ao sinal que será gerado a partir de uma *engine* (um módulo computacional implementado para a linguagem Julia), baseada no Algoritmo Mersenne Twister, advindo do paper de MATSUMOTO e NISHIMURA [19] e uma *seed* (um número de referência que gera sempre o mesmo sinal). Controlando a amplitude deste sinal de ruído, a rotina implementada na linguagem Julia permite testar a robustez com o aumento gradativo da amplitude, de tal forma a avaliar normas comparativas baseadas na resposta em frequência e no tempo. Com isto, é possível avaliar a robustez do método de identificação.

Em outro aspecto, é necessário avaliar se os algoritmos são robustos às não-linearidades. O sistema carro-pêndulo apresentado é não-linear devido ao movimento pendular. Para pequenos deslocamentos é possível ter um modelo linearizado que ainda assim pode ser considerado representativo, como apresentado em OGATA [9]. É esperado, que para onde seja possível de ser linearizado, o algoritmo o identifique significativamente bem. Entretanto, a robustez será avaliada para uma situação onde a linearização já não possa ser assumida. Tal como foi feito no sistema N-GDL, para este caso não-linear, pode-se aumentar a distância da condição inicial não-linear (no caso, o ângulo $\epsilon$ inicial) e avaliar os

resultados. Para tal, simulamos o sistema carro-pêndulo, utilizando um Método de Runge-Kutta de ordem 4 (também implementado pelo autor) para a determinação da resposta no tempo que é usada na identificação, como apresentado em RUGGIERO [10].

# 4 Desenvolvimento

## 4.1 Conceitos Fundamentais

### 4.1.1 Domínio Discreto

Para a compreensão da matemática no ambiente computacional, é necessário, antes, o estudo do domínio discreto, uma vez que computadores, por definição, não realizam operações de cálculo numérico de uma forma contínua, tal como se sustenta em KNUTH [15]. Supõe-se um espaço discreto $\mathbb{D}$, de tal forma que um elemento genérico de $\mathbb{D}$ seja definido por $D_k = k\delta$. Toma-se $k$ como $k \in \mathbb{Z}^+$ e $\delta$ como sendo um intervalo fixo, onde $\delta \in \mathbb{L}$, onde $\mathbb{L} := \{l \mid l \in \mathbb{R} : 0 < l \leq 1\}$. Com isto, cria-se um subespaço de $\mathbb{R}$ apenas com os elementos que são múltiplos de $\delta$. Portanto, qualquer sinal $S$ que possua uma taxa de amostragem $\delta$, estará contido em $\mathbb{D}$, tal que $\mathbb{D} \subset \mathbb{R}$. Note que se $\delta \to 0$, $\mathbb{D} \subseteq \mathbb{R}$. Desta forma, em uma amostragem de um sinal, cujo intervalo de tempo seja $\delta$, um elemento genérico do tempo $\mathbf{t}$, vetor finito de tempo correspondente aos intervalos da amostragem, pode ser representado como $t_k = k\delta$.

### 4.1.2 Sinal de entrada Y

Supõe-se um sistema que é amostrado discretamente com um intervalo $\delta$ e amostras $k$. Este sinal é caracterizado como tendo suas entradas organizada nas linhas e cada amostra como uma coluna, causando a formação de uma matriz $\boldsymbol{Y}$, onde $n$ é o número de entradas deste sinal, $k$ é o número de amostras de cada entrada:

$$\boldsymbol{Y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \cdots & y_{1k} \\ y_{21} & y_{22} & y_{23} & \cdots & y_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & y_{n3} & \cdots & y_{nk} \end{bmatrix} \tag{1}$$

### 4.1.3 Sistemas Dinâmicos

Como definem BAY [14] e OGATA [9], um sistema dinâmico é descrito como um sistema cujos estados podem ser definidos unicamente por um conjunto de variáveis cujo comportamento é definido por regras previamente estabelecidas. O sistema pode ser descrito de forma contínua ou discreta. Para os sistemas lineares, é possível manipular algebricamente para que a representação destes seja por um sistema de equações diferenciais de primeiro grau, no caso contínuo e equações de recorrência, no caso discreto. Neste sentido, é necessário avaliar que um sistema interage com seus próprios estados, como descrito anteriormente, e também com estímulos exógenos, sendo representados genericamente pelo vetor $\boldsymbol{u}$.

Para definir-se um sistema dinâmico matematicamente, é preciso tomar um conjunto mínimo de variáveis que possibilitem, para um tempo $t_i$ qualquer, conhecer o comporta-

mento do sistema para um tempo $t_i \leq t$ qualquer. Estas variáveis, que definem um sistema dinâmico são chamadas de estados. Neste sentido, é possível agrupar todas as variáveis de estado de um sistema em um vetor $\boldsymbol{e}$ de dimensão $n$, que se dá o nome de vetor de estados. Com isso, assume-se um espaço vetorial $\mathbb{E}^n$ em que $\boldsymbol{e}$ esteja contido, podendo $\mathbb{E}^n$ conter todos os possíveis estados do sistema dinâmico.

Desta forma, tem-se, para o caso contínuo, uma definição genérica de uma equação diferencial:

$$\dot{\boldsymbol{e}} = F(\boldsymbol{e}, \boldsymbol{u}, t) \tag{2}$$

E para o caso discreto podemos estabelecer uma equação genérica de recorrência com o índice $k$:

$$\boldsymbol{e}_{k+1} = F(\boldsymbol{e}_k, \boldsymbol{u}_k, t_k) \tag{3}$$

Reforça-se pelas equações que a variação do vetor de estados é uma função do vetor de estados, dos estímulos exógenos e do tempo. Desta forma, para um sistema linear, podemos escrever a seguinte equação generalizada para um sistema contínuo:

$$\begin{cases} \dot{\boldsymbol{e}} = \boldsymbol{A}_c\boldsymbol{e} + \boldsymbol{B}_c\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}_c\boldsymbol{e} + \boldsymbol{D}_c\boldsymbol{u} \end{cases} \tag{4}$$

E no caso de um sistema discreto, temos uma equação similar:

$$\begin{cases} \boldsymbol{e}_{k+1} = \boldsymbol{A}\boldsymbol{e}_k + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{D}\boldsymbol{u}_k \end{cases} \tag{5}$$

Muitas vezes a descrição dos sistemas é feita de forma contínua, mas no entanto, tal como apresentado na seção 4.1.2, a amostragem dos sinais é feita de forma contínua. Desta maneira, para que se possa comparar a obtenção de um modelo que possa ser adquirido por um algoritmo de identificação de sistema com um modelo teórico, é necessário uma "ponte"entre o domínio discreto e o domínio contínuo. Portanto, para tal, JUANG [1] apresenta que é possível tomar a primeira equação do sistema contínuo (4) e aplicar a Transformada de Laplace, considerando $s$ como a variável do domínio de Laplace, supondo uma condição inicial $\boldsymbol{e}(t_i)$:

$$s\boldsymbol{e}(s) - \boldsymbol{e}(t_i) = \boldsymbol{A}_c\boldsymbol{e}(s) + \boldsymbol{B}\boldsymbol{u}(s) \tag{6}$$

De tal forma que manipulando a expressão, obtém-se:

$$(s\boldsymbol{I} - \boldsymbol{A}_c)\boldsymbol{e}(s) = \boldsymbol{e}(t_i) + \boldsymbol{B}\boldsymbol{u}(s) \tag{7}$$

$$\boldsymbol{e}(s) = (s\boldsymbol{I} - \boldsymbol{A}_c)^{-1}\boldsymbol{e}(t_i) + (s\boldsymbol{I} - \boldsymbol{A}_c)^{-1}\boldsymbol{B}\boldsymbol{u}(s) \tag{8}$$

E por consequência, realizando a anti-transformada[1] é notado como $e$, sem o negrito.:

[1]Importante salientar que os estados são notados como $\boldsymbol{e}$ e o número de Euler

$$\boldsymbol{e}(t) = e^{(\boldsymbol{A}_c(t-t_i))}\boldsymbol{e}(t_i) + \int_{t_i}^{t_f} e^{\boldsymbol{A}_c(t_f-t)}\boldsymbol{B}_c\boldsymbol{u}(t)dt \tag{9}$$

Tomando uma taxa de amostragem $\delta$ de tal maneira que para $\boldsymbol{u}$, entre o intervalo $[k\delta, (k+1)\delta)$ os valores sejam constantes e iguais a $\boldsymbol{u}(k\delta)$, é possível tirar $\boldsymbol{u}$ da integral. Não obstante, tomando $t_i = k\delta$ e $t_f = (k+1)\delta$, obtém-se:

$$\boldsymbol{e}((k+1)\delta) = e^{\boldsymbol{A}_c\delta}\boldsymbol{e}(k\delta) + \left(\int_{k\delta}^{(k+1)\delta} e^{\boldsymbol{A}_c\delta}\boldsymbol{B}_c dt\right)\boldsymbol{u}(k\delta) \tag{10}$$

Note que o termo associado ao tempo na exponencial é dirimido como $(k+1)\delta - k\delta = \delta$. Posteriormente, podemos estabelecer uma comparação entre este termo e a primeira linha da equação 5. Desta maneira, analisando e comparando ambas as equações concomitantemente, tem-se:

$$\begin{cases} \boldsymbol{e}_{k+1} = \boldsymbol{A}\boldsymbol{e}_k + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{e}((k+1)\delta) = e^{\boldsymbol{A}_c\delta}\boldsymbol{e}(k\delta) + \left(\int_{k\delta}^{(k+1)\delta} e^{\boldsymbol{A}_c\delta}\boldsymbol{B}_c dt\right)\boldsymbol{u}(k\delta) \end{cases} \tag{11}$$

Observar-se-á o resultado da comparação, um dos fundamentos do comando c2d, presente em bibliotecas de linguagens como MATLAB e Julia.:

$$\begin{cases} \boldsymbol{e}_{k+1} = \boldsymbol{e}((k+1)\delta) \\ \boldsymbol{A} = e^{\boldsymbol{A}_c\delta} \\ \boldsymbol{e}_k = \boldsymbol{e}(k\delta) \\ \boldsymbol{B} = \int_{k\delta}^{(k+1)\delta} e^{\boldsymbol{A}_c\delta}\boldsymbol{B}_c dt \\ \boldsymbol{u}_k = \boldsymbol{u}(k\delta) \end{cases} \tag{12}$$

### 4.1.4 Controlabilidade

Como apresentado em BAY [14], uma das propriedades necessárias na avaliação dos sistemas dinâmicos, é a consideração se a partir de um esforço, aquele sistema poderá ser controlado. Desta forma, considerando a equação 9, é possível observar, que termos de exponenciais de matrizes, podem ser recomputados através do Teorema de Cayley-Hamilton:

$$e^{\boldsymbol{A}_c t} = \sum_{k=0}^{\infty} \boldsymbol{A}_c^k \frac{t^k}{k!} = \sum_{k=0}^{n-1} \boldsymbol{A}_c^k \frac{t^k}{k!} = \sum_{k=0}^{n-1} \boldsymbol{A}_c^k \Phi_i(t) \tag{13}$$

Desta forma, pode-se escrever a equação citada como:

$$\boldsymbol{e}(t) = \left(\sum_{k=0}^{n-1} \boldsymbol{A}_c^k \Phi_i(t-t_i)\right)\boldsymbol{e}(t_i) + \int_{t_i}^{t_f} \sum_{k=0}^{n-1} \boldsymbol{A}_c^k \Phi_i(t_f - t)\boldsymbol{B}_c\boldsymbol{u}(t)dt \tag{14}$$

Uma vez que $\Phi_k(t)$ são funções integráveis, o problema torna avaliar se as condições das matrizes formadas permitirão que, a partir do estado inicial em $t_i$, o estado em $t_f$ seja alcançado. Desta forma, a nível de simplificação, sem perda de generalidade, toma-se $\boldsymbol{e}(t_i) = \boldsymbol{0}$, tendo portanto:

$$\boldsymbol{e}(t) = \int_{t_i}^{t_f} \sum_{k=0}^{n-1} \boldsymbol{A}_c^i \Phi_k(t_f - t) \boldsymbol{B}_c \boldsymbol{u}(t) dt \tag{15}$$

onde manipulando a expressão obtém-se:

$$\boldsymbol{e}(t) = \sum_{k=0}^{n-1} \boldsymbol{A}_c^k \boldsymbol{B}_c \int_{t_i}^{t_f} \Phi_k(t_f - t) \boldsymbol{u}(t) dt \tag{16}$$

$$\begin{bmatrix} \boldsymbol{B}_c & \boldsymbol{A}_c\boldsymbol{B}_c & \boldsymbol{A}_c^2\boldsymbol{B}_c & \boldsymbol{A}_c^3\boldsymbol{B}_c & \dots & \boldsymbol{A}_c^{n-1}\boldsymbol{B}_c \end{bmatrix} \begin{bmatrix} \int_{t_i}^{t_f} \Phi_1(t_f - t)\boldsymbol{u}(t)dt \\ \int_{t_i}^{t_f} \Phi_2(t_f - t)\boldsymbol{u}(t)dt \\ \int_{t_i}^{t_f} \Phi_3(t_f - t)\boldsymbol{u}(t)dt \\ \dots \\ \int_{t_i}^{t_f} \Phi_{n-1}(t_f - t)\boldsymbol{u}(t)dt \end{bmatrix} \tag{17}$$

Desta forma, é possível definir $\mathcal{C} = \begin{bmatrix} \boldsymbol{B}_c & \boldsymbol{A}_c\boldsymbol{B}_c & \boldsymbol{A}_c^2\boldsymbol{B}_c & \boldsymbol{A}_c^3\boldsymbol{B}_c & \dots & \boldsymbol{A}_c^{n-1}\boldsymbol{B}_c \end{bmatrix}$ como sendo a matriz Controlabilidade dos sistemas.

Para o caso discreto, deve-se analisar a equação 5, obtendo-se:

$$\begin{cases} \boldsymbol{e}_1 = \boldsymbol{A}\boldsymbol{e}_0 + \boldsymbol{B}\boldsymbol{u}_0 \\ \boldsymbol{e}_2 = \boldsymbol{A}\boldsymbol{e}_1 + \boldsymbol{B}\boldsymbol{u}_1 \\ \boldsymbol{e}_3 = \boldsymbol{A}\boldsymbol{e}_2 + \boldsymbol{B}\boldsymbol{u}_2 \\ \boldsymbol{e}_4 = \boldsymbol{A}\boldsymbol{e}_3 + \boldsymbol{B}\boldsymbol{u}_3 \\ \vdots \\ \boldsymbol{e}_k = \boldsymbol{A}\boldsymbol{e}_k - 1 + \boldsymbol{B}\boldsymbol{u}_{k+1} \end{cases} \tag{18}$$

Portanto, analisando e considerando $\boldsymbol{e}_0 = \boldsymbol{0}$, é possível depreender que:

$$\begin{cases} \boldsymbol{e}_2 = \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{B}\boldsymbol{u}_1 \\ \boldsymbol{e}_3 = \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{B}\boldsymbol{u}_2 \\ \boldsymbol{e}_4 = \boldsymbol{A}^3\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_2 + \boldsymbol{B}\boldsymbol{u}_3 \\ \vdots \\ \boldsymbol{e}_k = \sum_{i=0}^{k-1} \boldsymbol{A}^i\boldsymbol{B}\boldsymbol{u}_{k-1-i} \end{cases} \tag{19}$$

$$\boldsymbol{e}_k = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{A}^3\boldsymbol{B} & \dots & \boldsymbol{A}^{k-1}\boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \vdots \\ \boldsymbol{u}_{k-1} \end{bmatrix} \tag{20}$$

Portanto, para o caso discreto, a Controlabilidade é $\mathcal{C} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{A}^3\boldsymbol{B} & \dots & \boldsymbol{A}^{k-1}\boldsymbol{B} \end{bmatrix}$

### 4.1.5 Observabilidade

Além de avaliar se é possível controlar o sistema, também é imprescindível analisar se é possível observar a saída do sistema. Desta forma, BAY [14] considera as equações 4 e 9 e novamente o Teorema de Cayley-Hamilton da equação 13, assumindo $\boldsymbol{u} = \mathbf{0}$ sem perda de generalidade, para avaliar a generalização na saída:

$$\boldsymbol{y}(t) = \boldsymbol{C}_c \left( \sum_{k=0}^{n-1} \boldsymbol{A}_c^k \Phi_i(t - t_i) \right) \boldsymbol{e}(t_i) \tag{21}$$

onde manipulando a expressão, tem-se:

$$\boldsymbol{y}(t) = \begin{bmatrix} \boldsymbol{C}_c \\ \boldsymbol{C}_c\boldsymbol{A}_c \\ \boldsymbol{C}_c\boldsymbol{A}_c^2 \\ \boldsymbol{C}_c\boldsymbol{A}_c^3 \\ \vdots \\ \boldsymbol{C}_c\boldsymbol{A}_c^{k-1} \end{bmatrix} \begin{bmatrix} \Phi_1(t - t_i) & \Phi_2(t - t_i) & \Phi_3(t - t_i) & \dots & \Phi_{k-1}(t - t_i) \end{bmatrix} \boldsymbol{e}(t_i) \tag{22}$$

Desta forma, atribui-se o símbolo $\mathcal{O}$ para a matriz em questão:

$$\boldsymbol{y}(t) = \mathcal{O}\boldsymbol{\Phi}(t)\boldsymbol{e}(t_i) \tag{23}$$

Para o caso discreto, deve-se analisar a equação 5, obtendo-se:

$$\begin{cases} \boldsymbol{y}_0 = \boldsymbol{C}\boldsymbol{e}_0 \\ \boldsymbol{y}_1 = \boldsymbol{C}\boldsymbol{A}\boldsymbol{e}_1 \\ \boldsymbol{y}_2 = \boldsymbol{C}\boldsymbol{A}\boldsymbol{e}_2 \\ \boldsymbol{y}_3 = \boldsymbol{C}\boldsymbol{A}\boldsymbol{e}_3 \\ \vdots \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{A}\boldsymbol{e}_{k-1} \end{cases} \tag{24}$$

Portanto, analisando e considerando $\boldsymbol{e} = \mathbf{0}$, é possível depreender que:

$$\begin{cases} \boldsymbol{y}_0 = \boldsymbol{C}\boldsymbol{e}_0 \\ \boldsymbol{y}_1 = \boldsymbol{C}\boldsymbol{A}\boldsymbol{e}_0 \\ \boldsymbol{y}_2 = \boldsymbol{C}\boldsymbol{A}^2\boldsymbol{e}_0 \\ \boldsymbol{y}_3 = \boldsymbol{C}\boldsymbol{A}^3\boldsymbol{e}_0 \\ \vdots \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{e}_0 \end{cases} \tag{25}$$

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \boldsymbol{CA}^3 \\ \vdots \\ \boldsymbol{CA}^{k-1} \end{bmatrix}^T \boldsymbol{e}(0) \tag{26}$$

$$\boldsymbol{y} = \mathcal{O}^T \boldsymbol{e}(0) \tag{27}$$

Posteriormente, no desenvolvimento deste trabalho, estes conceitos serão utilizados para a consolidação dos algoritmos de identificação. Neste trabalho, são utilizados três algoritmos de identificação: ARX, N4SID e GRA. O N4SID utiliza uma formulação que envolve a observabilidade, para a partir do estado inicial, conseguir realizar a estimativa dos estados do sistema. Posteriormente, o GRA, utiliza a matriz de Hankel, que pode ser representada como $\boldsymbol{H} = \mathcal{OC}$. Neste trabalho, não há um aprofundamento no desenvolvimento do N4SID, entretanto, nas próximas seções haverá um detalhamento de como o GRA é implementado.

# 5 Modelos Matemáticos dos Sistemas

## 5.1 Modelo Linear

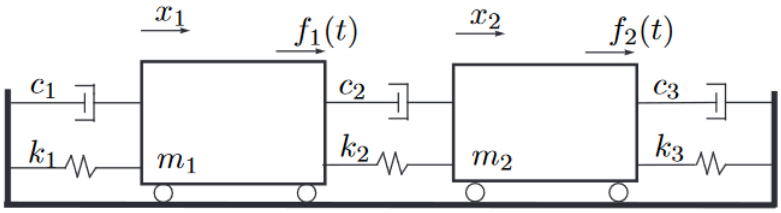


Figura 1: Sistema Massa-Mola-Amortecedor de 2 graus de liberdade e seus parâmetros típicos.

Neste trabalho, o primeiro sistema dinâmico a ser identificado é um Sistema Massa-Mola-Amortecedor com dois graus de liberdade. Na literatura, este sistema é uma abstração bastante utilizada para absorvedores de vibrações em turbinas eólicas, tal como no artigo de HAO [16], onde uma massa representa a turbina eólica e a outra a massa absorvedora. Este sistema será utilizado para testar a robustez dos algoritmos com relação ao ruído no sinal de saída. Este sistema pode ser descrito por um sistema de equações diferenciais, da seguinte forma:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{28}$$

Um detalhamento maior do desenvolvimento deste modelo é fornecido no Apêndice A1.1. Nomeando as matrizes, é possível obter uma equação

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{C}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \boldsymbol{f} \tag{29}$$

Nesta equação, $\boldsymbol{M}$ é a matriz de inércia, $\boldsymbol{C}$ é a matriz relativa aos amortecimentos do sistema, $\boldsymbol{K}$ é a matriz de rigidez do sistema e $\boldsymbol{x}$ é o vetor dos graus de liberdade. Com a obtenção deste sistema de equações de segundo grau, a formulação do modelo de estados é construída através do "empilhamento"da variável e de sua derivada, tal que:

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} \tag{30}$$

E com isto, é possível readequar a equação 28, tomando $\boldsymbol{u} = \boldsymbol{f}$:

$$\dot{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C} \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1}\boldsymbol{f} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C} \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1} \end{bmatrix} \boldsymbol{u} \tag{31}$$

Tendo então:

$$\begin{cases} \boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C} \end{bmatrix} \\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1} \end{bmatrix} \end{cases} \tag{32}$$

Para a implementação, na biblioteca de Controle da linguagem Julia, é necessário aplicar esta forma de estados e escolher saídas do sistema. Neste trabalho, as escolhidas para testar os algoritmos foram as variáveis de posição, levando à definição de $\boldsymbol{C}$ e $\boldsymbol{D}$ como:

$$\begin{cases} \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \\ \boldsymbol{D} = \boldsymbol{0} \end{cases} \tag{33}$$

## 5.2 Modelo Não-Linear

Este problema, da figura 2, pode ser descrito como se segue na equação 34, tendo seu desenvolvimento detalhado no Apêndice A1.2:

$$\begin{cases} (m_1 + m_2)\ddot{x} + m_2 l(sin(\theta)\dot{\theta}^2 - cos(\theta)\ddot{\theta}) + c_1\dot{x} + kx = u \\ (J + m_2 l^2)\ddot{\theta} - m_2 l\ddot{x}cos(\theta) + \beta\dot{\theta} + m_2 glsin(\theta) = 0 \end{cases} \tag{34}$$

Observando as equações, constata-se que este é um problema não linear. Existem duas abordagens para usuais para resolver o problema numericamente. Em torno de um ponto de operação, é possível tomar uma linearização das equações, ou simplesmente implementar as equações não lineares para serem resolvidas pelo Método de Runge-Kutta.

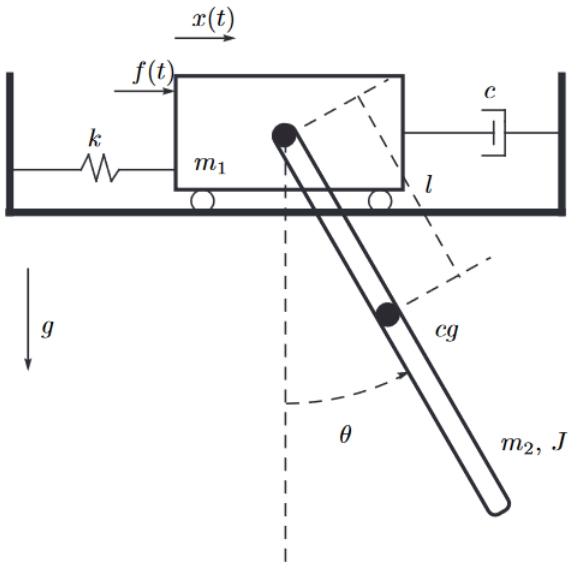


Figura 2: Sistema carro-pêndulo e seus parâmetros típicos.

### 5.2.1 Linearização

Fazendo a linearização, considerando o ponto de operação como $\theta = 0$, assume-se:

$$\begin{cases} \dot{\theta}^2 \approx 0 \\ \sin\theta \approx \theta \\ \cos\theta \approx 1 \end{cases} \tag{35}$$

Com estas aproximações, as equações de movimento tornam-se:

$$\begin{cases} (m_1 + m_2)\ddot{x} - m_2 l\ddot{\theta} + c_1\dot{x} + kx = u \\ (J + m_2 l^2)\ddot{\theta} - m_2 l\ddot{x} + \beta\dot{\theta} + m_2 g l\theta = 0 \end{cases} \tag{36}$$

Estas equações podem ser descritas como:

$$\begin{bmatrix} m_1 + m_2 & -m_2 l \\ -m_2 l & J + m_2 l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} c_1 & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} k & 0 \\ 0 & m_2 g l \end{bmatrix} \begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{37}$$

Similar a formulação da seção 9, pode ser considerada uma formulação de estados igual a (30), considerando $\boldsymbol{x} = (x, \theta)$ e se propõe a resolução tal como a descrição do sistema dinâmico contínuo.

### 5.2.2 Não-Linearização

O segundo sistema a ser tratado neste trabalho é o Sistema Carro-Pêndulo. Este sistema, como será modelado, apresenta um comportamento não-linear em sua formulação,

de modo que será utilizado para a avaliação de robustez de não-linearidade dos algoritmos. Para tal, a modelagem do sistema carro-pêndulo será feita a partir da Mecânica Newtoniana.

Como variáveis do problema, é importante salientar que $x$ é a coordenada de translação do carrinho e $\theta$ é o ângulo de inclinação do pêndulo. $m_1$ e $m_2$ são as massas do carro e pêndulo, respectivamente. $J$ é o momento de inércia do pêndulo no centro de gravidade. $c$ é o coeficiente de amortecimento do amortecedor, $l$ é o comprimento do centro de gravidade ($cg$) do pêndulo, até o carro. $g$ é a gravidade e $f(t)$ é a força aplicada ao carro, onde no desenvolvimento abaixo, já será considerada como $u$, o esforço atuante sobre o sistema.

Se a equação não for linearizada, ela pode ser escrita como:

$$\begin{bmatrix} m_1+m_2 & -m_2 l\cos\theta \\ -m_2 l\cos\theta & J+m_2 l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} c_1 & m_2 l\sin\theta\dot{\theta} \\ 0 & \beta \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} kx \\ m_2 g l\sin\theta \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{38}$$

Isolando os termos relativos às maiores derivadas, tem-se:

$$\begin{bmatrix} m_1+m_2 & -m_2 l\cos\theta \\ -m_2 l\cos\theta & J+m_2 l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} c_1\dot{x}+m_2 l\sin\theta\dot{\theta}^2+kx-u \\ \beta\dot{\theta}+m_2 g l\sin\theta \end{bmatrix} = \mathbf{0} \tag{39}$$

Posteriormente, é possível definir:

$$\begin{bmatrix} I_1 & -I_2(\theta) \\ -I_2(\theta) & I_3 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} f_1(x,\dot{x},\theta,\dot{\theta})-u \\ f_2(\theta,\dot{\theta}) \end{bmatrix} = \mathbf{0} \tag{40}$$

onde tem-se:

$$\begin{cases} I_1 = m_1+m_2 \\ I_2 = -m_2 l\cos\theta \\ I_3 = J+m_2 l^2 \\ f_1(x,\dot{x},\theta,\dot{\theta}) = c_1\dot{x}+m_2 l\sin\theta\dot{\theta}^2+kx \\ f_2(\theta,\dot{\theta}) = \beta\dot{\theta}+m_2 g l\sin\theta \end{cases} \tag{41}$$

Denominando a matriz de inércia de $\boldsymbol{J}$, define-se o par $\boldsymbol{x}=(x,\theta)$ e avalia-se:

$$\boldsymbol{J}(\theta)\ddot{\boldsymbol{x}} = -\begin{bmatrix} f_1(x,\dot{x},\theta,\dot{\theta})-u \\ f_2(\theta,\dot{\theta}) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u - \boldsymbol{f}(x,\dot{x},\theta,\dot{\theta}) \tag{42}$$

A saber, a inversa da matriz de inércia é:

$$\boldsymbol{J}^{-1}(\theta) = \frac{1}{I_1 I_3 - I_2(\theta)^2} \begin{bmatrix} I_3 & I_2(\theta) \\ I_2(\theta) & I_1 \end{bmatrix} \tag{43}$$

Separando os termos compostos com $u$, compostos apenas das forças, é possível reescrever a equação como:

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}^{-1}(\theta)\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} u - \boldsymbol{f}(x,\dot{x},\theta,\dot{\theta})\right) \tag{44}$$

Assume-se agora um modelo de estados similar ao da formulação anterior. Tem-se então:

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} \tag{45}$$

Desta forma, é escrito $\boldsymbol{e}_i$ o i-ésimo elemento do vetor $\boldsymbol{e}$:

$$\dot{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{e}_3 \\ \boldsymbol{e}_4 \\ -\boldsymbol{J}^{-1}(\theta)\vec{F}(\boldsymbol{e}) + \boldsymbol{J}^{-1}(\theta) \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \end{bmatrix} \tag{46}$$

Em essência $\dot{\boldsymbol{e}} = F(\boldsymbol{e}, \boldsymbol{u}, t)$, tal como o modelo contínuo genérico apresentado na seção 4.1.3. Esta equação diferencial é possível de ser resolvida de forma numérica, utilizando, o método de Runge-Kutta de ordem 4 (RK4). De forma genérica, o método é utilizado quando há um Problema de valor inicial, com condição inicial conhecida e uma função que descreve uma derivada de primeira ordem das variáveis a serem conhecidas. Matematicamente:

$$\begin{cases} \boldsymbol{e}(0) = \boldsymbol{e}_0 \\ \dot{\boldsymbol{e}} = F(\boldsymbol{e}, \boldsymbol{u}, t) \end{cases} \tag{47}$$

Na formulação do RK4, proposta em RUGGIERO [10], a função $\dot{\boldsymbol{e}}$ nos reais ($\mathbb{R}$), se torna uma equação em que se pode determinar o estado de um instante $k+1$ conhecendo o instante $k$:

$$\dot{\boldsymbol{e}} = \frac{d\boldsymbol{e}}{dt} = \boldsymbol{F} \rightarrow \boldsymbol{e}_{k+1} = \boldsymbol{e}_k + \frac{\Delta t}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{48}$$

De tal forma que o RK4 é numericamente definido como:

$$\begin{cases} \boldsymbol{k}_1 = \boldsymbol{F}(\boldsymbol{e}_k, t_k, u_k) \\ \boldsymbol{k}_2 = \boldsymbol{F}(\boldsymbol{e}_k + 0.5\boldsymbol{k}_1\Delta t, t_k + 0.5\Delta t, u_k) \\ \boldsymbol{k}_3 = \boldsymbol{F}(\boldsymbol{e}_k + 0.5\boldsymbol{k}_2\Delta t, t_k + 0.5\Delta t, u_k) \\ \boldsymbol{k}_4 = \boldsymbol{F}(\boldsymbol{e}_k + \boldsymbol{k}_3\Delta t, t_k + \Delta t, u_k) \end{cases} \tag{49}$$

Com isto, de fato a resposta do sistema é obtida no domínio $\mathbb{D}$, de tal forma que é possível implementá-la computacionalmente e plotar seus respectivos gráficos de saída, de maneira que seja possível compreender o comportamento do sistema com relação as suas coordenadas.

# 6 Elaboração dos Algoritmos

## 6.1 GRA (ERA)

### 6.1.1 GRA

O *General Realization Algorithm* é um algoritmo criado por DE GRAFFON et al. [7], que é uma generalização do ERA de JUANG e PAPPA [2]. O algoritmo permite que

através de uma entrada qualquer, a matriz $\boldsymbol{H}$ seja obtida e aplicada no ERA, que apenas consegue identificar sistemas a partir de entradas impulso. Como dito anteriormente, a Matriz de Hankel é formada a partir dos sinais de impulso $\boldsymbol{h}$. Desta forma, o ponto central do algoritmo é a avaliação da matriz $\boldsymbol{H}$ sem necessariamente excitar o sistema a partir de um sinal de impulso. Desta forma, sintéticamente, DE GRAFFON apresenta que, em um caso discreto, tem-se uma equação para a saída no seguinte sentido:

$$\boldsymbol{y}_k = \boldsymbol{D}\boldsymbol{u}_k + \sum_{i=1}^{k} \boldsymbol{h}_i u_{k-i} \tag{50}$$

Para a situação discreta, o que se tem é o fato de que o sinal obtido é consequência da excitação e a entrada impulso, de tal forma que pode ser sintetizada, para todos os pontos, manipulando a matriz de Toeplitz, como:

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{U} \tag{51}$$

onde podemos aplicar a Pseudotransformada Inversa de Moore-Penrose:

$$\boldsymbol{H} = \boldsymbol{Y}\boldsymbol{U}^T \left(\boldsymbol{U}\boldsymbol{U}^T\right)^{-1} \tag{52}$$

Com isso, obtemos $\boldsymbol{H}$ e aplicamos o algoritmo do ERA para encontrarmos os parâmetros.

### 6.1.2 Matriz de Hankel

No contexto de identificação de sistemas, a matriz de Hankel, em suma, é uma matriz que permite a organização dos resultados de uma função impulso do sistema, de tal forma, que ao realizar sua Decomposição em Valores Singulares, podemos obter as matrizes $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ e $\boldsymbol{D}$. Definindo $\boldsymbol{H}$ como sendo o sinal do sistema com relação a entrada impulso:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}_0 & \boldsymbol{h}_1 & \boldsymbol{h}_2 & \dots & \boldsymbol{h}_n \end{bmatrix} \tag{53}$$

A matriz de Hankel $\boldsymbol{H}_n$ pode ser sintetizada como:

$$\boldsymbol{H}_n = \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \boldsymbol{h}_3 & \dots & \boldsymbol{h}_{n-p} \\ \boldsymbol{h}_2 & \boldsymbol{h}_3 & \boldsymbol{h}_4 & \dots & \boldsymbol{h}_{n-p+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{h}_p & \boldsymbol{h}_{p+1} & \boldsymbol{h}_{p+2} & \dots & \boldsymbol{h}_n \end{bmatrix} \tag{54}$$

onde $p$ é um parâmetro a ser controlado no problema de identificação de sistemas, tal como mostrado no artigo de SOARES e SERPA [13]. Desta forma, também é importante notar, que neste sentido temos:

$$\boldsymbol{H}_n = \begin{bmatrix} \boldsymbol{C}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^2\boldsymbol{B} & \dots & \boldsymbol{C}\boldsymbol{A}^{n-p}\boldsymbol{B} \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^3\boldsymbol{B} & \dots & \boldsymbol{C}\boldsymbol{A}^{n-p+1}\boldsymbol{B} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}\boldsymbol{A}^p\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{p+1}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{p+2}\boldsymbol{B} & \dots & \boldsymbol{C}\boldsymbol{A}^n\boldsymbol{B} \end{bmatrix} \tag{55}$$

Portanto, é possível de observar que a matriz pode ser escrita como sendo:

$$\boldsymbol{H}_n = \mathcal{O}\mathcal{C} \tag{56}$$

Posteriormente, podemos definir $\boldsymbol{H}_1$ e $\boldsymbol{H}_2$ como:

$$\boldsymbol{H}_1 = \begin{bmatrix} \boldsymbol{CB} & \boldsymbol{CAB} & \boldsymbol{CA^2B} & \dots & \boldsymbol{CA^{n-p-1}B} \\ \boldsymbol{CAB} & \boldsymbol{CA^2B} & \boldsymbol{CA^3B} & \dots & \boldsymbol{CA^{n-p}B} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA^pB} & \boldsymbol{CA^{p+1}B} & \boldsymbol{CA^{p+2}B} & \dots & \boldsymbol{CA^{n-1}B} \end{bmatrix} \tag{57}$$

$$\boldsymbol{H}_2 = \begin{bmatrix} \boldsymbol{CAB} & \boldsymbol{CA^2B} & \boldsymbol{CA^3B} & \dots & \boldsymbol{CA^{n-p}B} \\ \boldsymbol{CA^2B} & \boldsymbol{CA^3B} & \boldsymbol{CA^4B} & \dots & \boldsymbol{CA^{n-p+1}B} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA^p} + 1\boldsymbol{B} & \boldsymbol{CA^{p+2}B} & \boldsymbol{CA^{p+3}B} & \dots & \boldsymbol{CA^nB} \end{bmatrix} \tag{58}$$

### 6.1.3 ERA

A descrição do algoritmo neste trecho é uma síntese do que é apresentado no livro de JUANG [1] e no artigo de JUANG e PAPPA [2]. Os autores iniciam o algoritmo separando a Matriz de Hankel em duas, tal como apresentado anteriormente. A primeira parte correspondente da primeira até a penúltima coluna ($\boldsymbol{H}_1$) e a segunda parte, correspondente da segunda até a última coluna ($\boldsymbol{H}_2$). Para possibilitar a realização de manipulações algébricas com relação a mesma, é possível executar uma Decomposição em Valores Singulares (SVD), de tal maneira que se obtenha matrizes que possam ser analogamente manipuladas como uma decomposição em auto-valores. Dessa decomposição são considerados apenas os valores singulares significativos, representados pelo índice $r$, que também é um parâmetro de escolha. No caso, $r$ acaba sendo a ordem do modelo. Para a matriz $\boldsymbol{H}_1$ é possível analisar o que se segue:

$$\boldsymbol{H_1} = \boldsymbol{U}_t \boldsymbol{S}_t {\boldsymbol{V}_t}^T = \begin{bmatrix} \boldsymbol{U}_r & \boldsymbol{U}_z \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_z \end{bmatrix} \begin{bmatrix} {\boldsymbol{U}_r}^T \\ {\boldsymbol{U}_z}^T \end{bmatrix} \tag{59}$$

Como $\boldsymbol{S}_z$ é praticamente $\boldsymbol{0}$, é possível reescrever a SVD como:

$$\boldsymbol{H_1} = \boldsymbol{U}_r \boldsymbol{S}_r {\boldsymbol{V}_r}^T \tag{60}$$

onde, desta forma, pode-se observar que:

$$\boldsymbol{H_1} = \boldsymbol{U}_r \boldsymbol{S}_r {\boldsymbol{V}_r}^T = \boldsymbol{U}_r \boldsymbol{G}^2 {\boldsymbol{V}_r}^T = (\boldsymbol{U}_r \boldsymbol{G})(\boldsymbol{G}{\boldsymbol{V}_r}^T) \tag{61}$$

Porém, sabe-se que:

$$\boldsymbol{H_1} = \mathcal{O}_r \mathcal{C}_r \tag{62}$$

Portanto:

$$\begin{cases} \mathcal{O} = \boldsymbol{U}_r \boldsymbol{G} \\ \mathcal{C} = \boldsymbol{G}{\boldsymbol{V}_r}^T \end{cases} \tag{63}$$

Entretanto, nota-se que $\boldsymbol{H}_2$ também pode ser escrita em função de $\mathcal{O}$ e $\mathcal{C}$. Logo:

$$\boldsymbol{H_2} = \mathcal{O}\boldsymbol{A}\mathcal{C} \tag{64}$$

Desta forma, é possível reescrever $\boldsymbol{H}_2$ como sendo:

$$\boldsymbol{H_2} = \boldsymbol{U}_r\boldsymbol{G}\boldsymbol{A}\boldsymbol{G}\boldsymbol{V}_r{}^T \tag{65}$$

Então, isolando $\boldsymbol{A}$, é possível obter:

$$\boldsymbol{A} = \boldsymbol{G}^{-1}\boldsymbol{U}_r{}^T\boldsymbol{H_2}\boldsymbol{V}_r\boldsymbol{G}^{-1} \tag{66}$$

Desta maneira, é possível depreender que, devido às definições anteriormente apresentadas de Controlabilidade e Observabilidade, onde o termo $i \times j$ representa o número de elementos a serem considerados a partir do primeiro elemento da matriz e $l$ o número de entradas de $\boldsymbol{u}$, as matrizes $\boldsymbol{B}$ e $\boldsymbol{C}$ podem ser definidas como:

$$\begin{cases} \boldsymbol{C} = (\boldsymbol{U}_r\boldsymbol{G})_{k\times r} \\ \boldsymbol{B} = (\boldsymbol{G}\boldsymbol{V}_r{}^T)_{r\times l} \end{cases} \tag{67}$$

Não obstante, é possível depreender do próprio sinal, uma vez que $\boldsymbol{Y}(0) = \boldsymbol{D}$.

## 6.2 ARX

O acrônimo ARX significa *Autorregressive model with Exogenous Inputs*, que em português seria "Modelo Autoregressivo com Entradas Exógenas". Em suma, isto significa dizer que se elabora um modelo que, para predizer a $s = k + 1$ saída do sistema, avalia-se as saídas e entradas presentes e anteriores ($k$, $k-1$, etc.). Neste sentido, a implementação de KURKA [11] propõe que seja possível escrever um modelo ARX da seguinte forma:

$$\sum_{i=s-p_d}^{s} \alpha_i \boldsymbol{y}_i = \sum_{i=s-p_n}^{k} \boldsymbol{B}_i \boldsymbol{u}_i \tag{68}$$

onde $\alpha_i$ são os coeficientes associados a $\boldsymbol{y}_i$ e $\boldsymbol{B}_i$ são as matrizes de coeficientes que correlacionam as entradas $\boldsymbol{u}_i$ com as saídas. $p_d$ e $p_n$ são parâmetros que indicam a ordem do denominador e numerador do modelo. Entende-se por denominador o número de saídas anteriores que serão consideradas e numerador o número de entradas que serão consideradas no modelo. De forma geral, para se encontrar os coeficientes, a ferramenta proposta é a consolidação de um sistema linear cujas variáveis possam ser resolvidas. Neste sentido, manipulando a expressão para isolar o termo avançado $\boldsymbol{y}_{k+1}$, obtém-se:

$$\alpha_{k+1}\boldsymbol{y}_{k+1} = \sum_{i=s-p_n}^{k} \boldsymbol{B}_i \boldsymbol{u}_i - \sum_{i=s-p_d}^{k} \alpha_i \boldsymbol{y}_i \tag{69}$$

Com isso, assume-se $\alpha_{k+1} = 1$. Um caso onde $\alpha_{k+1} = 0$ não faria sentido, de tal forma que então $\alpha_k$ seria considerado o termo avançado. Portanto, para se encontrar os coeficientes, pode-se manipular as expressões, de tal forma que se encontre um sistema matricial. De acordo com AXLER [18], pode-se reescrever o somatório com as saídas tal como um produto interno:

$$\sum_{i=s-p_d}^{k} \alpha_i \boldsymbol{y}_i = \begin{bmatrix} \boldsymbol{y}_{s-p_d} & \boldsymbol{y}_{s-p_d+1} & \boldsymbol{y}_{s-p_d+2} & \cdots & \boldsymbol{y}_k \end{bmatrix} \begin{bmatrix} \alpha_{s-p_d} \\ \alpha_{s-p_d+1} \\ \alpha_{s-p_d+2} \\ \vdots \\ \alpha_k \end{bmatrix} = \boldsymbol{y}^k \boldsymbol{\alpha} \tag{70}$$

Não obstante, para tal, também considera-se a matriz $\boldsymbol{\Phi}$ que permite organizar as saídas $\boldsymbol{y}$ de uma forma matricial:

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{y}^{p_d-1} \\ \boldsymbol{y}^{p_d} \\ \vdots \\ \boldsymbol{y}^{n-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_0 & \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_{p_d-1} \\ \boldsymbol{y}_1 & \boldsymbol{y}_2 & \boldsymbol{y}_3 & \cdots & \boldsymbol{y}_{p_d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{y}_{n-2-p_d} & \boldsymbol{y}_{n-1-p_d} & \boldsymbol{y}_{n-p_d} & \cdots & \boldsymbol{y}_{n-1} \end{bmatrix} \tag{71}$$

Também, neste sentido, é importante salientar que se pode chamar $\boldsymbol{y_n}$ como sendo o vetor de vetores com todas as respostas de saída avançadas:

$$\boldsymbol{y_n} = \begin{bmatrix} \boldsymbol{y}_{p_d} \\ \boldsymbol{y}_{p_d+1} \\ \vdots \\ \boldsymbol{y}_n \end{bmatrix} \tag{72}$$

Com relação ao somatório de matrizes de entrada, é necessário avaliar que as incógnitas das matrizes não estão como vetores. Desta forma, é necessário uma manipulação algébrica em todos os termos do somatório:

$$\sum_{i=s-p_n}^{k} \boldsymbol{B}_i \boldsymbol{u}_i = \boldsymbol{B}_{s-p_n} \boldsymbol{u}_{s-p_n} + \ldots + \boldsymbol{B}_k \boldsymbol{u}_k \tag{73}$$

Neste sentido, o objetivo torna-se transformar cada coluna da matriz como um vetor, de tal forma, que ocorra o empilhamento dos coeficientes para o sistema linear. Seja $l$ e o número de linhas de $\boldsymbol{u}$ e $j$ a dimensão de $\boldsymbol{y}$:

$$\boldsymbol{B}_i \boldsymbol{u}_i = \begin{bmatrix} \beta_{1,1}^i & \cdots & \beta_{1,l}^i \\ \vdots & \ddots & \vdots \\ \beta_{j,1}^i & \cdots & \beta_{j,l}^i \end{bmatrix} \begin{pmatrix} u_1^i \\ u_2^i \\ \vdots \\ u_l^i \end{pmatrix} = \begin{bmatrix} \beta_{1,1}^i \\ \vdots \\ \beta_{j,1}^i \end{bmatrix} u_1^i + \ldots \begin{bmatrix} \beta_{1,l}^i \\ \vdots \\ \beta_{j,l}^i \end{bmatrix} u_l^i \tag{74}$$

Seja $\mathbf{I}_p$ a matriz identidade de dimensão $j$ e seja $\mathbf{b}_k$ a k-ésima coluna de $\boldsymbol{B}$ . Pode-se reorganizar a matriz e avaliar que o resultado final é um Produto de Kronecker:

$$\boldsymbol{B}_i \boldsymbol{u}_i = u_1^i \boldsymbol{I}_p \begin{bmatrix} \beta_{1,1}^i \\ \vdots \\ \beta_{j,1}^i \end{bmatrix} + \ldots + u_l^i \boldsymbol{I}_p \begin{bmatrix} \beta_{1,l}^i \\ \vdots \\ \beta_{j,l}^i \end{bmatrix} = \begin{bmatrix} u_1^i \boldsymbol{I}_p & \ldots & u_s^i \boldsymbol{I}_p \end{bmatrix} \begin{bmatrix} \mathbf{b}_1^i \\ \vdots \\ \mathbf{b}_l^i \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_i^T \otimes \boldsymbol{I}_p \end{bmatrix} \begin{bmatrix} \mathbf{b}_1^i \\ \vdots \\ \mathbf{b}_l^i \end{bmatrix} \tag{75}$$

Desta forma, pode-se reconsiderar novamente o somatório escrito como:

$$\sum_{i=s-p_n}^{k} \boldsymbol{B}_i \boldsymbol{u}_i = \sum_{i=s-p_n}^{k} \left[\boldsymbol{u}_i^T \otimes \boldsymbol{I}_p\right] \begin{bmatrix} \mathbf{b}_1^i \\ \vdots \\ \mathbf{b}_s^i \end{bmatrix} = \sum_{i=s-p_n}^{k} \left[\boldsymbol{u}_i^T \otimes \boldsymbol{I}_p\right] \mathbf{b}^i = \left[\left(\boldsymbol{u}^k\right)^T \otimes \boldsymbol{I}_p\right] \mathbf{b} \tag{76}$$

Para todas as entradas do sinal, é possível considerar a matriz $\boldsymbol{U}$, que consolida todas as entradas do sistema e o vetor $\mathbf{b}$, descritos como:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}^{p_n-1} \\ \boldsymbol{u}^{p_n} \\ \vdots \\ \boldsymbol{u}^{n-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \dots & \boldsymbol{u}_{p_n-1} \\ \boldsymbol{u}_1 & \boldsymbol{u}_3 & \boldsymbol{u}_4 & \dots & \boldsymbol{u}_{p_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{u}_{n-p_n} & \boldsymbol{u}_{n+1-p_n} & \boldsymbol{u}_{n+2-p_n} & \dots & \boldsymbol{u}_{n-1} \end{bmatrix} \tag{77}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{s-p_n} \\ \mathbf{b}^{s-p_n+1} \\ \vdots \\ \mathbf{b}^{k} \end{bmatrix} \tag{78}$$

Portanto, representa-se $\boldsymbol{y}_{k+1}$ como sendo

$$\boldsymbol{y}_{k+1} = \left[\left(\boldsymbol{u}^k\right)^T \otimes \boldsymbol{I}_p\right] \mathbf{b} - \boldsymbol{y}^k \boldsymbol{\alpha} \tag{79}$$

E então, sintetiza-se $\mathbf{b}$ e $\boldsymbol{\alpha}$ como sendo $\boldsymbol{\xi}$:

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{b} \end{bmatrix} \tag{80}$$

De tal forma, que se manipule o termo inteiro como:

$$\boldsymbol{y}_{k+1} = \begin{bmatrix} -\boldsymbol{y}^k & \left(\boldsymbol{u}^k\right)^T \otimes \boldsymbol{I}_p \end{bmatrix} \boldsymbol{\xi} \tag{81}$$

Como foi assumido que, para todo $k$, $\boldsymbol{\xi}$ é sempre o mesmo, devido a hipótese do sistema ser linear ser tomada, generaliza-se a equação para $\boldsymbol{y_n}$:

$$\boldsymbol{y_n} = \boldsymbol{M}\boldsymbol{\xi} \tag{82}$$

onde $\boldsymbol{M}$ é definido por:

$$\boldsymbol{M} = \begin{bmatrix} -\boldsymbol{\Phi} & \boldsymbol{U}^T \otimes \boldsymbol{I}_p \end{bmatrix} \tag{83}$$

Entretanto, a equação 82 é um sistema possível indeterminado, de tal forma que é possível aplicar o Método dos Mínimos Quadrados para resolvê-lo e encontrar $\boldsymbol{\xi}$.

$$\boldsymbol{M}\boldsymbol{\xi} = \boldsymbol{y} \longrightarrow \boldsymbol{M}^T\boldsymbol{M}\boldsymbol{\xi} = \boldsymbol{M}^T\boldsymbol{y} \longrightarrow \boldsymbol{\xi} = \left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T\boldsymbol{y} \tag{84}$$

Encontrando-o, pode-se montar o modelo com os coeficientes. Para a implementação e simulação em Julia, pode-se utilizar a transformada-z, que permite que os coeficientes sejam interpretados como parte de uma função de transferência no domínio discreto para

a biblioteca de controle em Julia, de CARLSEN et al [6]. Desta forma, entende-se a transformada-z como:

$$\boldsymbol{y}_{k+1} = z\boldsymbol{y}_k \tag{85}$$

Relembrando a equação inicial que fundamenta a hipótese do modelo:

$$\sum_{i=s-p_d}^{s} \alpha_i \boldsymbol{y}_i = \sum_{i=s-p_n}^{k} \boldsymbol{B}_i \boldsymbol{u}_i \tag{86}$$

Através da transformada, é possível escrever que:

$$\boldsymbol{y}_s \left( \sum_{i=s-p_d}^{s} \alpha_i z^{s-i} \right) = \left( \sum_{i=s-p_n}^{k} \boldsymbol{B}_i z^{k-i} \right) \boldsymbol{u}_k \tag{87}$$

De tal forma, que considerando $\alpha_s = 1$, pode-se obter a função de transferência do modelo ARX:

$$\boldsymbol{y}_{k+1} = \frac{\left( \sum_{i=s-p_n}^{k} \boldsymbol{B}_i z^{k-i} \right)}{\left( 1 + \sum_{i=s-p_d}^{k} \alpha_i z^{s-i} \right)} \boldsymbol{u}_k \tag{88}$$

Desta maneira, a função de transferência pode ser implementada na linguagem Julia e ser a representação do sistema identificado pelo método.

## 6.3 N4SID

O algoritmo N4SID, é um acrônimo geral para *Numerical Algorithms For Subspace State System Identification*.[2] Estes algoritmos se utilizam, essencialmente, da técnica de realizar projeções ortogonais nos subespaços de estados. Em termos de manipulações algébricas para soluções computacionais, decomposições QR e SVD são realizadas, tal como explica VAN OVERSCHEE e de MOOR [5].

Não obstante, o artigo aqui citado destes dois autores são referência para a implementação de CARLSON e FÄLT [6] para a biblioteca de Identificação de Sistemas da linguagem Julia, que é utilizada neste trabalho. A implementação de CARLSON e FÄLT foi preferida em virtude do cronograma do projeto e que a implementação, pois teria um impacto de consumo de tempo muito alto, podendo atrasar a geração de resultados.

## 6.4 Métrica de Erro

Como métrica de erro, adota-se neste trabalho a métrica da norma 2. Primeiro, avalia-se que a norma 2 de uma matriz qualquer $\boldsymbol{B}_{l\times k}$ é definida como:

$$||\boldsymbol{B}||_2 = \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{l} B_{ij}^2} \tag{89}$$

[2] O som do número 4 na língua inglesa é semelhante a palavra *for*.

Agora, define-se uma comparação, que é a discrepância de uma matriz estimada ($\boldsymbol{B_e}$) com relação a matriz original ($\boldsymbol{B_o}$). Esta diferença é dividida pela norma da matriz original, de tal forma que obtemos um resultado percentual. Desta forma, definimos o parâmetro $\epsilon$ que estabelece esta relação.

$$\epsilon = \frac{||\boldsymbol{B_e} - \boldsymbol{B_o}||_2}{||\boldsymbol{B_o}||_2} \tag{90}$$

Esta métrica será utilizada para avaliar cada simulação com um percentual de ruído somado ao sinal do sistema linear ou efeitos não-lineares mais pronunciados, no sistema não-linear. Com essa métrica, elaborar-se-á um gráfico da propriedade analisada versus a métrica, de tal forma que se gere uma curva de robustez, que mensurará o quanto de erro existe na identificação. Esta curva será melhor explicada na seção de Resultados.

# 7 Resultados

## 7.1 Resultados das simulações

### 7.1.1 Verificação do Sistema

Como dito anteriormente, é necessário simular o sistema, de modo que se possa gerar uma resposta para aplicarmos o ruído, no caso do Massa-Mola-Amortecedor, ou avaliar a não-linearidade, no caso do Carro-Pêndulo.

Nestes sistemas, conforme a formulação apresentada, utiliza-se os seguintes parâmetros [3] para o sistema Massa-Mola-Amortecedor:

$$\begin{cases} m_1 = 0.77 & kg \\ m_2 = 0.59 & kg \\ c_1 = 2.1 & Ns/m^2 \\ c_2 = 1.2 & Ns/m^2 \\ c_3 = 9 & Ns/m^2 \\ k_1 = 200 & N/m \\ k_2 = 200 & N/m \\ k_3 = 200 & N/m \end{cases} \tag{91}$$

e para o Carro-Pêndulo:

[3] Demais condições numéricas podem ser encontradas no apêndice A2, nos arquivos *main.jl* e *main_nl.jl*

$$\begin{cases} g = 9.81 & m/s^2 \\ l = 0.3302 & m \\ m_1 = 0.75 & kg \\ m_2 = 0.23 & kg \\ J = 0.00788 & kgm^3 \\ c = 5.4 & Ns/m^2 \\ \beta = 0.0024 & Nms/rad \\ k = 142 & N/m \end{cases} \tag{92}$$

Devemos verificar a coerência dos resultados com o que seria esperado pela literatura. Como sabemos, para este problema, estabelece-se uma entrada de pulso quadrado para excitação de diversas frequências. De forma geral, pode-se entender esta entrada como uma sequência de duas entradas degrau unitário, uma com amplitude contrária a outra, tal como mostrado no exemplo da figura 3, que é utilizada como fonte excitadora nos problemas.

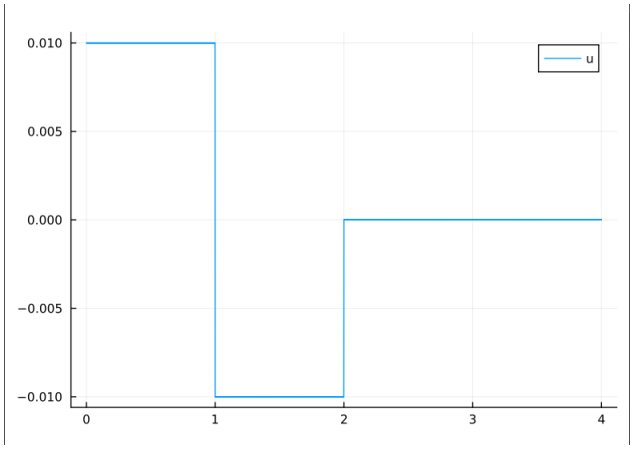


Figura 3: Exemplo de excitação aplicada ao sistema

Desta forma, usando ou o comando *lsim* da biblioteca de controle da linguagem Julia, para o caso linear e o Método de Runge-Kutta 4, para o caso não-linear, é possível obter a resposta e avaliá-la. Para efeito de verificação da coerência das repostas, simula-se o sistema utilizando da mesma força que será aplicada ao longo das análises, com condições iniciais zeradas. Posteriormente, é avaliado qualitativamente o sinal, considerando aspectos como estabilidade da resposta e coerência com a força aplicada. Desta forma, obteve-se as figuras 5 e 4.


Figura 4: Simulação para o sistema Massa-Mola-Amortecedor

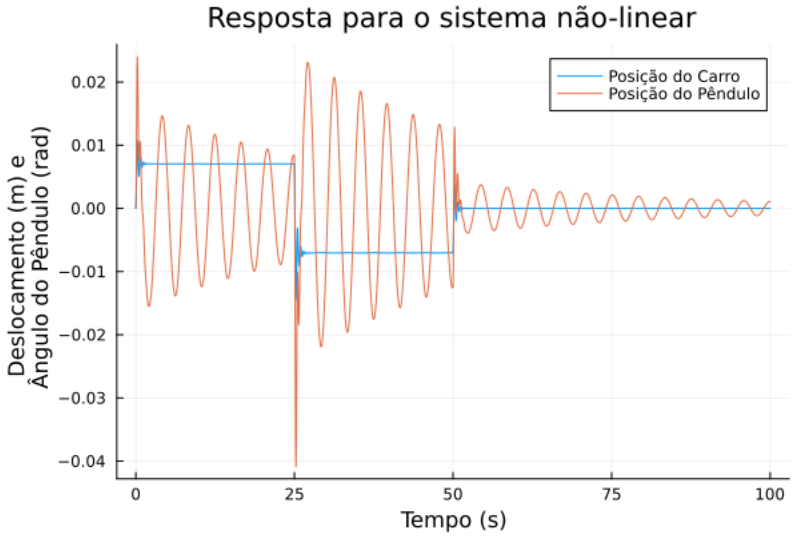


Figura 5: Simulação para o sistema Carro-Pêndulo

De acordo com RAO [11], a resposta esperada de sistemas massa-mola-amortecedor é uma oscilação em torno do um ponto da entrada imputada, corrigido pelas constantes da equação. Desta forma, como mostrado na figura, os sistemas respondem da maneira esperada.

## 7.2 Comparativos empregados

Para efetuar as análises neste trabalho, foram utilizadas duas metodologias para cada um dos problemas. Para o problema linear (PL), foi utilizada o Diagrama de Bode e para o problema não-linear (PNL), foi utilizado a Transformada de Fourier - como uma *Fast Fourier Transform* (FFT). Nesta subseção, será realizada uma exemplificação dos comparativos que serão empregados para mensurar a robustez.

### 7.2.1 Massa-Mola-Amortecedor - Problema Linear (PL)

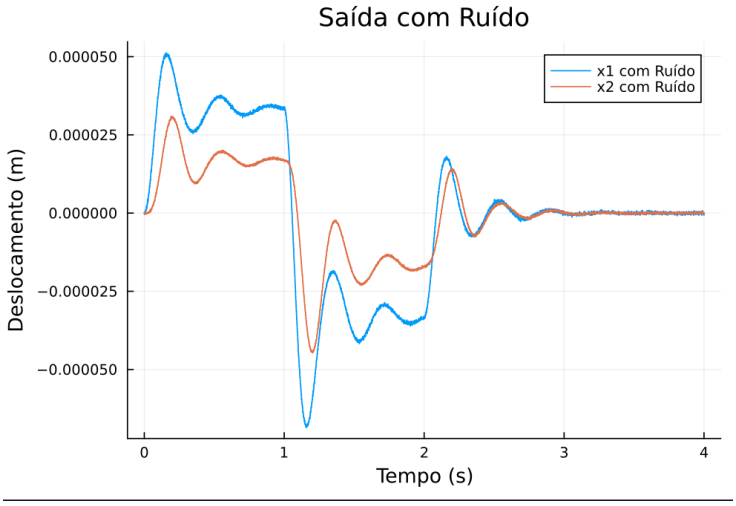


Figura 6: Simulação do PL com 2% de ruído

Para o PL, assume-se 2% de ruído com relação a maior amplitude do sinal temporal. Com isso, obtém-se a figura 6. Usando os três algoritmos para a identificação dos sistemas, é possível analisar, para cada um dos sistemas identificados, a saída $x_1$, tal como mostra a figura 7. Posteriormente, com as saídas dos sistemas identificados, é possível de se gerar as componentes do Diagrama de Bode. Na figura 8 mostram-se a amplitude da magnitude da transformada para a mesma saída. Para o Diagrama de Bode, as componentes são a amplitude da magnitude, o ganho fasorial e a frequência. Nesta análise, para gerar a frequência, foi utilizado o comando *bode* no sistema original, retornando estas três componentes, e reaplicando a frequência como argumento para as análises nos sistemas identificados.

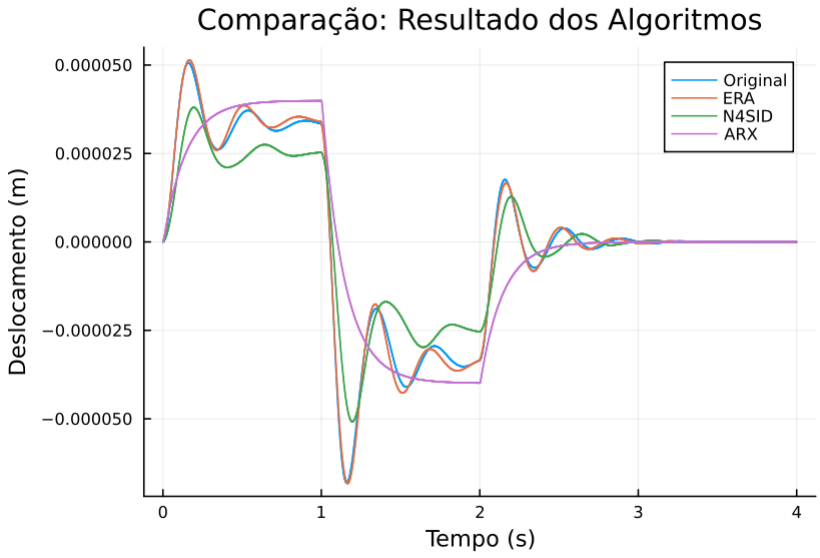


Figura 7: Simulação do PL com 2% de ruído

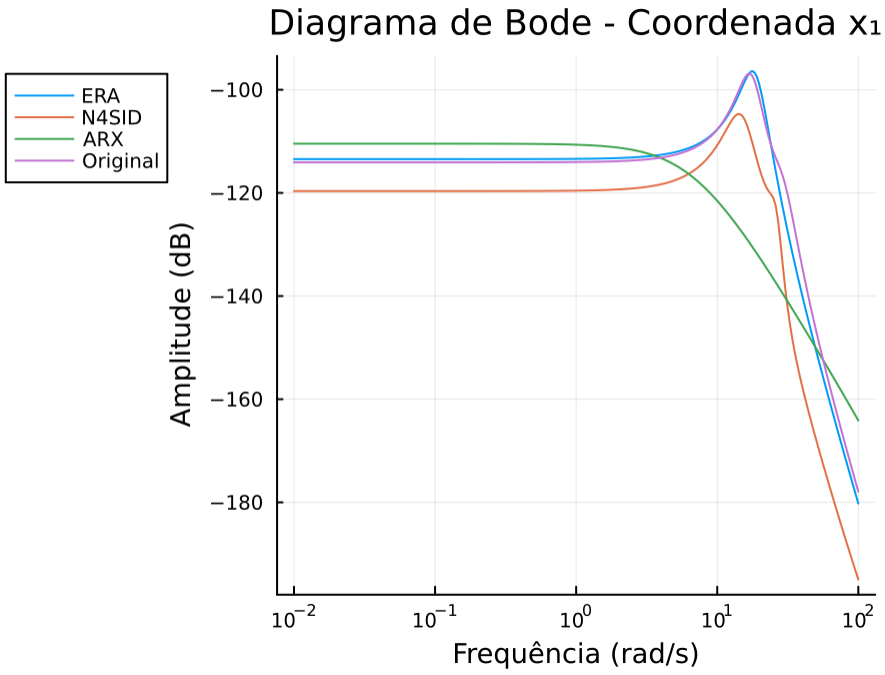


Figura 8: Diagrama de Bode da amplitude da magnitude para a coordenada $x_1$ do PL

### 7.2.2 Carro-Pêndulo - Problema Não-Linear (PNL)

Para o PNL assume-se uma condição inicial de 2.7º. Desta forma, as saídas do problema podem ser vistas na figura 9.

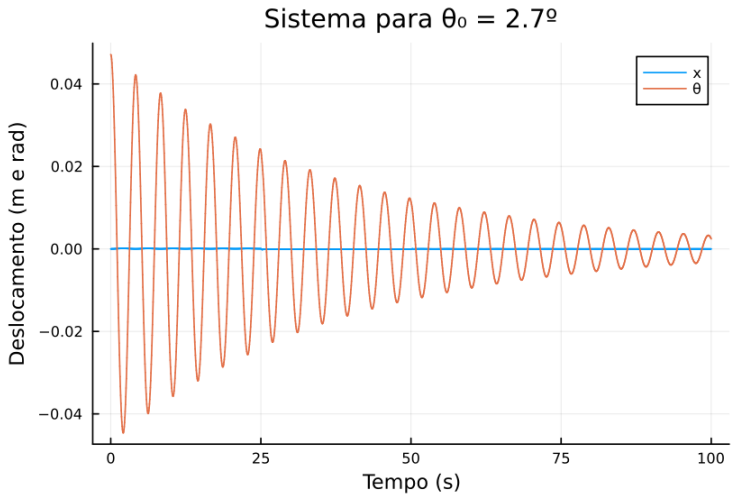


Figura 9: Simulaçãodo PNL com condição inicial no ângulo $\theta = 2.7$

Novamente, utilizando os algoritmos, obtém-se os sistemas identificados. Desta forma, é possível analisar a saída $x$ (posição do carro), tal como mostra a figura 10.

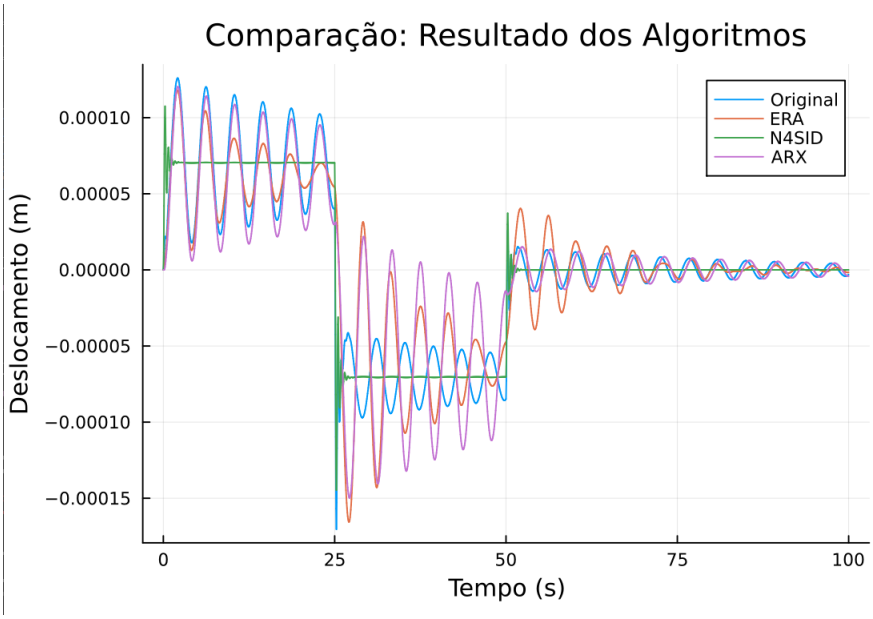


Figura 10: Simulaçãodo PNL com condição inicial no ângulo $\theta = 2.7$

Posteriormente, aplica-se a FFT nos sinais obtidos, de tal forma a obter a amplitude destes. Na figura 11 é mostrada a FFT para a coordenada $x$ do carro no PNL.

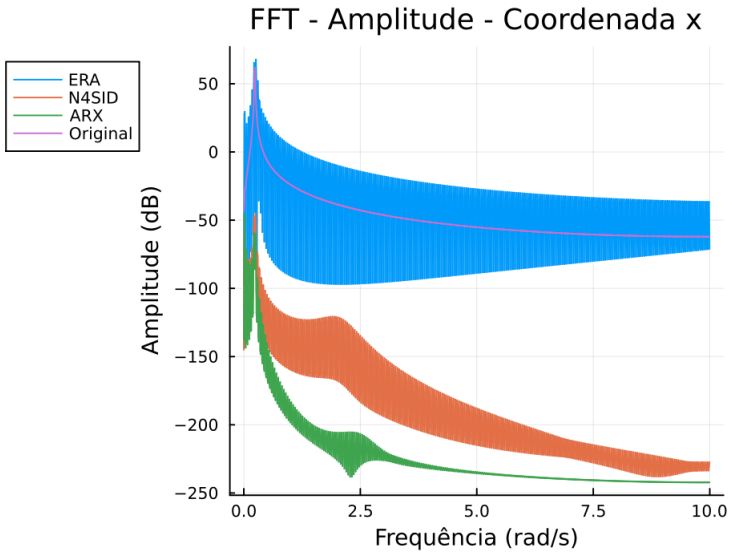


Figura 11: Amplitude resultado da FFT da posição do carro no PNL

Para a FFT, as componentes são a amplitude e fase do sinal. No caso, aplica-se o comando *fftshift*, que ordena as componentes de forma cardinal. Não obstante, representa-se apenas as componentes positivas da frequência (uma vez que as negativas não são do interesse desta análise). Desta forma, para melhorar a acurácia da representação, pela FFT ser simétrico, soma-se as componentes negativas as positivas.

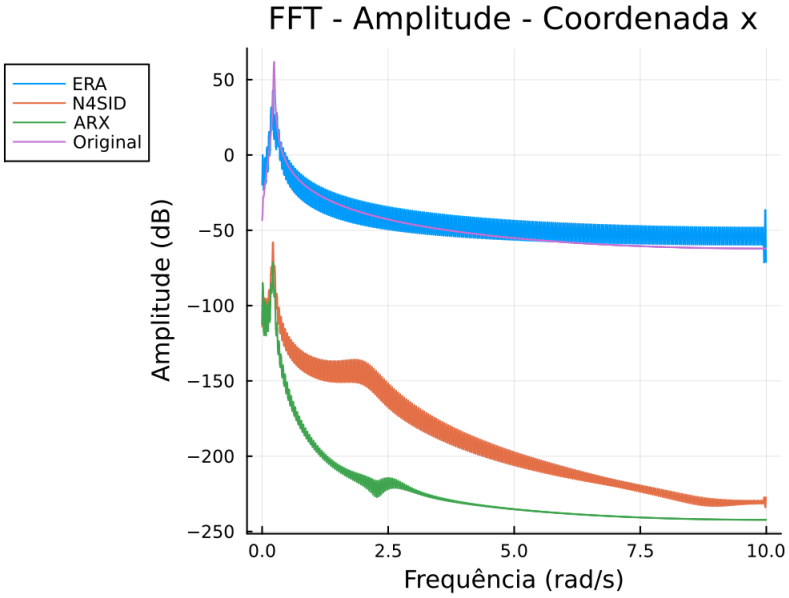


Figura 12: Amplitude resultado da FFT da posição do carro no PNL

No caso, nem todas as frequências dos sinais gerados pelo comando *lsim* possuem

amplitudes relevantes na FFT, gerando valores próximos de zero. Desta forma, estas, que não são aparentes na FFT, acabam se tornando picos, gerando o ruído apresentado no diagrama da Transformada. Para uma visualização mais clara, na figura 12, é possível analisar os sinais, caso fossem aplicados uma média móvel simples, considerando as 5 próximas frequências, com relação a frequência local. Entretanto, não foi considerado o erro em média, uma vez que o interesse deste trabalho é analisar o sistema obtido pelo algoritmo, e por consequência, sua simulação.

### 7.2.3 Resultados dos Comparativos

Nesta exemplificação, houve apenas a plotagem das amplitudes de cada processo. Entretanto, é importante salientar que, na avaliação da robustez, também são comparadas comparada as componentes em radianos.

Desta forma, obtém-se uma pontuação de erro, baseada na métrica apresentada na seção 6.4, onde comparando os resultados do sistema original, com os resultados dos sistemas simulados, é possível gerar um conjunto de dados. Desta forma, obtém-se os valores apresentados nas tabelas 1 e 2.

| **Algoritmo** | **Erro em Amplitude (Bode)** | **Erro em Fase (Bode)** |
|---|---|---|
| GRA | 0.074 | 0.21 |
| N4SID | 0.286 | 0.404 |
| ARX | 0.478 | 0.636 |

Tabela 1: Erros obtidos para 2% de ruído no PL

| **Algoritmo** | **Erro em Amplitude (FFT)** | **Erro em Fase (FFT)** |
|---|---|---|
| GRA | 1.236 | 1.429 |
| N4SID | 0.996 | 1.607 |
| ARX | 0.998 | 1.224 |

Tabela 2: Erros obtidos para $\theta_0 = 2.7^{\circ}$ no PNL

## 7.3 Resultados das análises para o Massa-Mola-Amortecedor e o Carro-Pêndulo

Utilizando dos códigos implementados na seção"Apêndices", é possível simular e identificar os sistemas várias vezes, com diferentes parâmetros. Desta forma, pode-se utilizar a métrica apresentada em (90) para comparar respostas, que para cada parâmetro numérico da simulação, como ruído ou condição inicial de não linearidade, acabam gerando uma resposta com relação a métrica, de tal forma que podem ser estabelecidos pontos cartesianos, que podem ser visualizados em uma curva. Esta curva, chamamos aqui de curva de robustez.

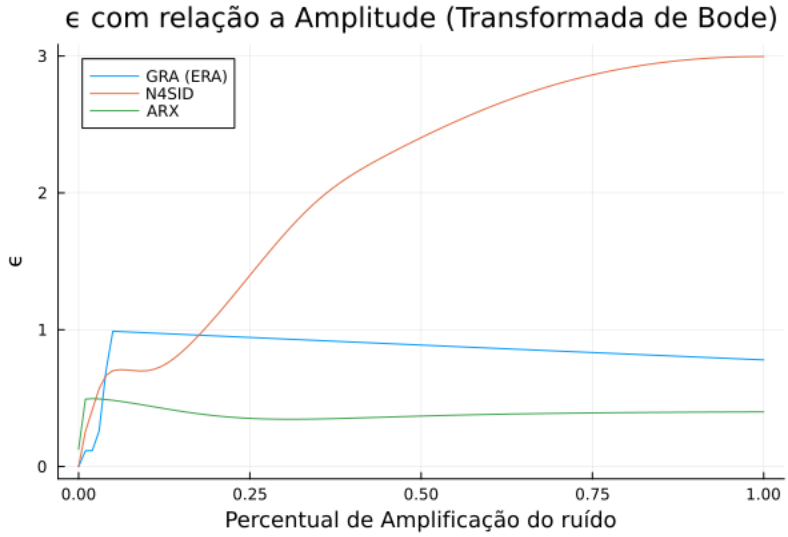


Figura 13: $\epsilon$ para a Amplitude do Massa-Mola-Amortecedor

Para o sistema Massa-Mola-Amortecedor, foi aplicado um ruído gaussiano sobre o sinal simulado, e os valores que geram o Diagrama de Bode foram avaliados. Este diagrama estabelece uma representação visual da amplitude e da fase do sistema para diferentes frequências do sistema estimado. Entretanto, estes dados são gerados computacionalmente, de tal maneira que podem ser armazenados em um vetor e ter a métrica em (90) computada.

Com isso, realizando as identificações obtemos as figuras 13 e 14. Na figura 13 temos a curva de robustez em amplitude, que em, aproximadamente 5%, mostra um comportamento aproximadamente linear dos algoritmos. Depois deste intervalo, o ARX e o ERA acabam apresentando um comportamento de decaimento, e entre de 10% e 15% o N4SID assume um comportamento não linear, caracterizado por uma curva que atesta um valor de erro grande comparado com outros algoritmos no que tange a amplitude.

Para a fase, representada na figura 14 o comportamento acaba tendo certas semelhanças, de tal forma que até 5% poderia ser realizada uma aproximação linear e a partir deste valor o comportamento dos algoritmos varia. Curiosamente, a curva de robustez de fase do ARX e do ERA possuem um formato de curva similar a “curva”de robustez em amplitude, enquanto a curva de robustez do N4SID acaba tendo um formato menos reto, tal como a sua curva de amplitude.

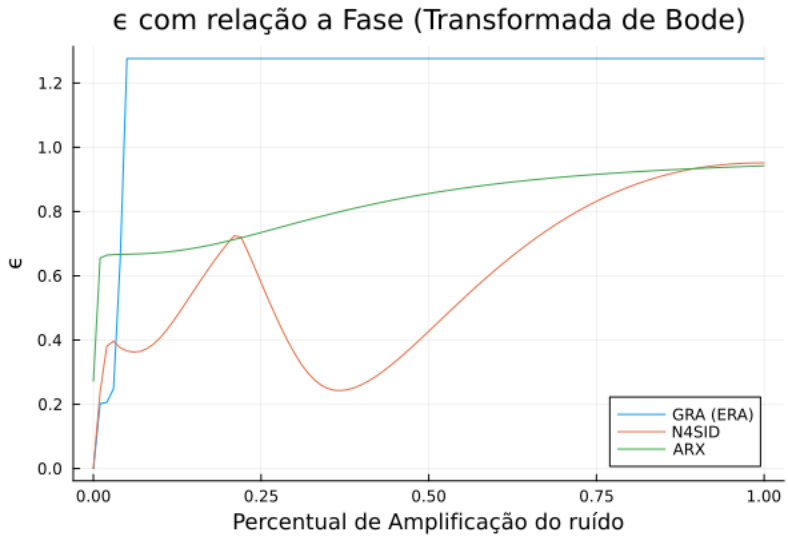


Figura 14: $\epsilon$ para a Fase do Massa-Mola-Amortecedor

Posteriormente, para o sistema não-linear do Carro-Pêndulo, toma-se a Transformada Discreta de Fourier, através do algoritmo *Fast Fourier Transform* (FFT). A FFT possibilita que um sinal discreto seja caracterizado em função das frequências contidas nele. Desta forma, ao comparar a amplitude e a fase para cada frequência, pode-se caracterizar o sinal e comparar com seu original. Com isso, obtém-se os resultados descritos nas figuras 15 e 16.

Para a curva de robustez da amplitude da FFT no caso não-linear, representada em 15, temos um comportamento aproximadamente linear até aproximadamente o ângulo de $3^{o}$. Posteriormente, temos um comportamento estável dos algoritmos, de tal forma que o erro é sempre constante, a exceção do ERA, entre aproximadamente $55^{o}$ e $75^{o}$, tendo um crescimento da curva de robustez neste intervalo, com um pico em aproximadamente $65^{o}$.

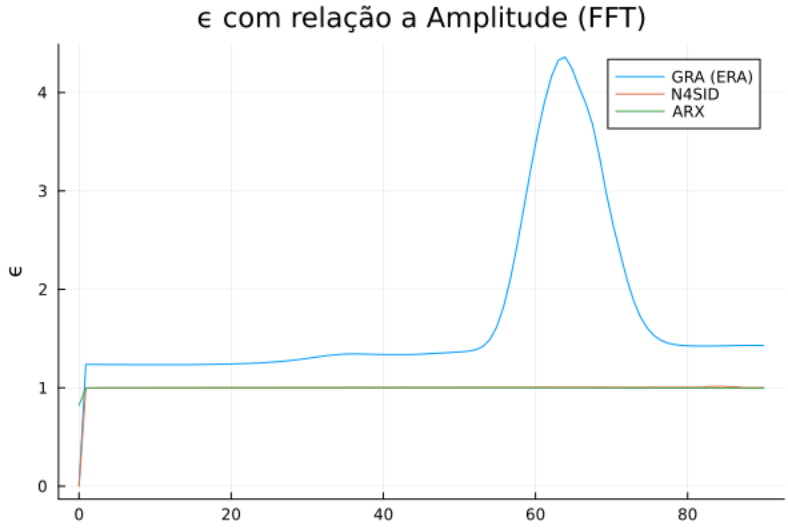


Figura 15: $\epsilon$ para a Amplitude do Carro-Pêndulo

Com relação à fase representada na figura 16, temos um comportamento mais difícil de ser descrito com relação ao ERA e ao ARX, de tal forma que os algoritmos têm uma oscilação muito grande da curva de robustez da fase. Era esperado, considerando que os algoritmos assumem modelos lineares, que haveria distorção no sinal do modelo identificado, porém, esperava-se que o comportamento seria mais previsível, tal como no sistema linear. O único algoritmo que teve uma estabilidade maior foi o N4SID que possuiu um erro constante ao longo do intervalo para frente de 5º, aproximadamente.

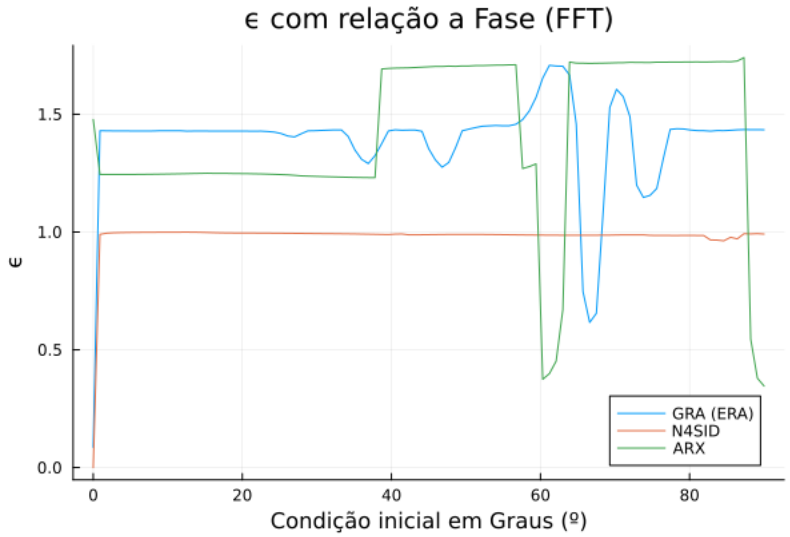


Figura 16: $\epsilon$ para a Fase do Carro-Pêndulo

# 8 Discussões

Analisando os algoritmos, podemos notar que a formulação deles é diferente. Em suma, o ARX é uma forma do Método dos Mínimos Quadrados aplicados aos coeficientes do polinômio na transformada-z, o ERA é baseado em uma SVD (*Singular Value Decomposition*), onde através de manipulações algébricas pode-se obter as matrizes de estado e o N4SID é um algoritmo que, acaba combinando as duas técnicas. Uma suma da sequência deste algoritmo é que, primeiro realiza-se uma técnica de Mínimos Quadrados aplicado aos subespaços encontrados, e posteriormente realiza-se uma SVD para encontrar os parâmetros do sistema. Portanto, é importante notar que os algoritmos possuem conceitos diferentes e possuem formas diferentes de serem implementados. Reforçar esta diferença é importante para ser notado que a comparação está ocorrendo entre algoritmos de natureza diferentes, e não evoluções ou modificações de um mesmo algoritmo.

Desta forma, observando os resultados, pode-se avaliar que os algoritmos possuem um comportamento diferente nas duas situações. De forma geral, o ARX possuiu menos erro em ambas. O algoritmo, em suma, realiza um procedimento que, a partir das informações de saída do sistema, produz sua interpretação criando um modelo linear e descobrindo os coeficientes do mesmo pelo Método dos Mínimos Quadrados. Entretanto, mesmo para a situação não-linear, o algoritmo produziu valores baixos de erro, em comparação com os outros, mas, para a situação onde não há ruído, nem condição não-linear aplicada,

ele produziu um erro diferente de zero. Este resultado não era esperado, uma vez que o algoritmo deveria retornar a saída idêntica ao sistema.

Várias hipóteses foram traçadas para esse problema, entretanto, a causa-raiz não foi encontrada. Considerando que, para situação sem ruído, nem condição não-linear aplicada, existe um valor de erro, uma das hipóteses que pode ser cogitada é o fato de que na modelagem apresentada, não há um termo de *offset*. Outras formulações, como por exemplo a de HAMILTON [17], apresenta não um único coeficiente linear para as saídas, mas sim, matrizes similares as das entradas, que correlacionam os termos para cada canal do vetor. Desta forma, a interpolação é feita para matrizes, e uma técnica similar a apresentada na seção 6.2 utilizando o Produto de Kronecker para encontrar os coeficientes da matriz.

Outro aspecto a destacar é a diferença do comportamento do N4SID para ambos os cenários. Para o cenário de ruído, o algoritmo possui um aumento do erro proporcional ao aumento do ruído, no que tange a amplitude, e para a fase, possui um comportamento significativamente melhor que o do ARX, porém, com variações significativas. Entretanto, para o cenário de não-linearidade, o algoritmo é o que tem a maior constância e o menor erro entre todos. Mesmo sobre condições severas de não-linearidade, o algoritmo consegue preservar seu erro com relação a fase e amplitude de maneira praticamente constante, possuindo melhor desempenho, dentre os três escolhidos, para lidar com fenômenos de não-linearidade, com relação ao sistema utilizado neste trabalho.

Por fim, para o GRA, para os resultados com ruído, verifica-se que ele possui o efeito inverso do N4SID, possuindo um grau de erro na amplitude, menor que o do supracitado. Entretanto, para a fase, o algoritmo teve um erro maior, porém, na sua curva de erro, possui um comportamento mais constante, independente da situação apresentada. Mesmo que o ruído aumente, o erro com relação a fase permanece o mesmo, o que possibilita que o sistema possa ser avaliado com a mesma taxa de erro, independentemente da condição a qual ele é submetida. Desta forma, entende-se que a constância do comportamento também é algo que deve ser avaliado como propriedade do algoritmo na sua utilização.

Ao longo deste trabalho, também foi notado que fatores como a amplitude do sinal de entrada e a discretização do problema são significativos na obtenção dos resultados. Entretanto, para o estudo, estes parâmetros foram fixados para uma abordagem inicial do problema. Foram considerados parâmetros que atendiam a limitação do hardware disponível para confecção dos resultados e também foi considerada uma amplitude numérica que estivesse condizente com as dimensões do problema.

Este trabalho, não cobriu a questão do ruído e da não-linearidade combinados. Entretanto, podemos inferir que utilizar apenas um algoritmo de identificação para um problema genérico não é conveniente, uma vez que cada um possui propriedades diferentes para ambos os efeitos. Como dito anteriormente, os algoritmos possuem formulações diferentes, que fazem com que cada um possua propriedades e características diferentes. Neste sentido, para tentar mitigar vieses de cada modelo, o ideal é utiliza-los em conjunto. Não somente, considerando que a maioria dos fenômenos, quando medidos, possuem ruídos, devido aos seus sistemas de medição e não-linearidades. Neste sentido, ambos podem ser relevantes, dependendo da análise que o usuário pretende fazer. Desta forma, novamente, reforça-se a utilização de diferentes algoritmos em um mesmo problema, de tal forma que um algoritmo compense o que o outro acaba não sendo suficientemente robusto.

# 9 Conclusões

Avaliando o desenvolvimento dos algoritmos implementados neste trabalho, foi possível analisar que os fundamentos que os assentam são distintos. Esta distinção faz com que, neste trabalho, a comparação seja entre algoritmos que não tem o mesmo fundamento e nem sejam evoluções ou desenvolvimentos de um mesmo algoritmo. Desta forma, como uma proposição de trabalho futuro, seria fazer a comparação entre algoritmos distintos, de maneira a providenciar uma comparação entre estruturas algorítmicas similares.

Neste sentido, para elaborar a comparação, utilizou-se do conceito da curva de robustez, que é o erro pela propriedade a ser avaliada. No caso deste trabalho, foi avaliado ruído e não-linearidade. Desta forma, para cada alteração de parâmetro, uma identificação foi realizada e seus resultados medidos, de tal forma, que foi possível gerar a curva de robustez.

Os resultados obtidos, mostram que, para a questão do ruído, pode-se observar que de forma geral, o ARX possuiu melhor performance para ambas as situações, pois apresentou, qualitativamente, menor índice de erro, porém tendo problemas na interpolação de resultados para condições nulas. Já, o N4SID possui boa robustez à não linearidade, porém na análise do sinal com o ruído de entrada, entre os algoritmos comparados, ele possui o pior desempenho, uma vez que seu erro aumenta proporcionalmente ao aumento do ruído. Por fim, o GRA, com relação as suas curvas de erro, possui menor variação para os casos, porém, possuindo maior erro com relação aos outros algoritmos.

Com estes resultados, pode-se concluir que os algoritmos possuem comportamentos distintos e suas robustez variam de acordo com os parâmetros avaliados. De forma geral, não pode se concluir, que cada algoritmo possui maior robustez para parâmetros diferentes, tal como o ARX sendo menos suscetível ao ruído na amplitude, porém sendo mais sensível ao ruído na não-linearidade. Neste sentido, avalia-se que, para situações reais, os algoritmos devem ser utilizados de forma combinada, de modo a evitar vieses.

# Referências


[1] JUANG, Jer-Nan; **Applied System Identification**. 1 ed. Prentice Hall PTR, 1994.

[2] LJUNG, Lennart; **Perspectives on System Identification**. Disponível em: **http://users.isy.liu.se/rt/ljung/seoul2dvinew/plenary2.pdf**

[3] JUANG, Jer-nan; PAPPA, Richard. S.; **An eigensystem realization algorithm for modal parameter identification and model reduction**. *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 5., 1982

[4] JUANG, Jer-nan; PAPPA, Richard. S.; **Galileo spacecraft modal identification using an eigensystem realization algorithm**. *25th Structures, Structural Dynamics and Materials Conference*, 1984.

[5] VAN OVERSCHEE, P.; DE MOOR, B..; **Subspace Identification of Linear Systems: Theory, Implementation, Applications**. Springer Publishing, 1996.

[6] CARLSON, F. B., FÄLT, M. ; **Julia Control System Identification - Package Documentation**. Disponível em: https://docs.juliahub.com/ControlSystemIdentification/FH1SZ/2.0.2/

[7] De CALLAFON, R. A. ; MOAVENI B., CONTE J. P., et al. **General realization algorithm for modal identification of linear dynamic systems**. *Journal of Engineering Mechanics* Vol. 134-9, 2008

[8] STASZEWSKI, W.J. BOLLER, C., TOMLINSON, G. R.; **Health Monitoring of Aerospace Structures**. John Willey & Sons, 2004.

[9] OGATA, K.; **Engenharia de Controle Moderno**. 5 ed. Pearson Prentice Hall.

[10] RUGGIERO, M. A. G.; LOPES, V. L. R. **Cálculo Numérico: Aspectos Teóricos e Computacionais**. 2 ed., Pearson Prentice Hall.

[11] RAO, S. S. **Mechanical Vibrations**, 5 ed. Prentice Hall, 2010.

[12] KURKA, P. R. G. **Vibrações de Sistemas Dinâmicos - Análise e Síntese**, 2 ed., Elsevier

[13] SOARES Jr., D, SERPA, A. L. **An evaluation of the influence of Eigensystem Realization Algorithm settings on multiple input multiple output system identification**, Journal of Vibration and Control, 8 (21-22), 2022, 3286-3301.

[14] BAY, J. S. **Fundamentals of Linear State Space Systems**, 1ed., McGraw-Hill.

[15] KNUTH, D. E. **The Art of Computer Programming**, Vol. 1, Addison-Wesley Professional, 2011.

[16] ZUO, H., KAIMING B., HONG, H.; **A state-of-the-art review on the vibration mitigation of wind turbines**, *Renewable and Sustainable Energy Reviews*, Vol. 121, 2020.

[17] HAMILTON, J.D.; **Time Series Analysis. Princeton University**, Princeton Univeristy Press, 1994.

[18] AXLER, S. J.; **Linear Algebra Done Right**, Springer, 2004.

[19] MATSUMOTO, M. e NISHIMURA T., **Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator**, ACM Transactions on Modeling and Computer Simulation.

# Apêndice 1 - Desenvolvimento matemático dos modelos físicos

## A1.1 - Modelagem Matemática - Sistema Massa-Mola-Amortecedor com dois graus de liberdade

É possível efetuar a modelagem do sistema via Mecânica Lagrangiana, considerando uma coordenada generalizada do sistema $q$, tal como apresentado em RAO [11]. A equação generalizada de movimento proposta por Lagrange é dada por:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q_{nc} \tag{93}$$

Observa-se que a ação do sistema é descrita por $T$ e $V$, energias cinéticas e potenciais do sistema, respectivamente:

$$L = T - V \tag{94}$$

Observemos que:

$$\begin{cases} T = \frac{m_1 \dot{x}_1^2}{2} + \frac{m_2 \dot{x}_2^2}{2} \\ V = \frac{k_1 {x_1}^2}{2} + \frac{k_2 (x_1 - x_2)^2}{2} + \frac{k_3 {x_2}^2}{2} \end{cases} \tag{95}$$

Desta formulação, observa-se que $T(\dot{q}, t)$ e $V(q, t)$ são funções de $\dot{q}$ e $q$ respectivamente, portanto:

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}}(T(\dot{q}, t) - V(q, t)) - \frac{\partial}{\partial q}(T(\dot{q}, t) - V(q, t)) = Q_{nc} \tag{96}$$

Tornando-se

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}}T(\dot{q}, t) + \frac{\partial}{\partial q}V(q, t) = Q_{nc} \tag{97}$$

Neste trabalho, considera-se a hipótese de que o trabalho dissipado por um amortecedor qualquer possa ser escrito como:

$$Wd_k = -C_k \dot{q} q \tag{98}$$

Podendo avaliar o trabalho total dissipado pelo amortecedor como:

$$D = \sum_{k=1}^{g} Wd_k = \sum_{k=1}^{g} -C_k \dot{q} q \tag{99}$$

O que para o sistema em questão, temos $g = 3$, onde especificamente, para o amortecedor de número 2, a coordenada $q$ é definida como $x_1 - x_2$, pois uma vez que não possui referencial fixo, seu deslocamento total depende das duas coordenadas. Desta forma, expandindo o somatório, temos:

$$D = -(C_1 \dot{x}_1 x_1 + C_2 (\dot{x}_1 - \dot{x}_2)(x_1 - x_2) + C_3 \dot{x}_2 x_2) \tag{100}$$

Para o cômputo do que não é conservativo, leva-se em consideração que existem forças $f_k$ onde $k = 1, 2$ atuando no sistema. Portanto, o trabalho pode ser descrito como:

$$Wf_k = f_k q_k \rightarrow Wf = f_1 x_1 + f_2 x_2 \tag{101}$$

Considerando:

$$Q_{nc} = \frac{\partial Wf}{\partial q} + \frac{\partial D}{\partial q} \tag{102}$$

Obtém-se:

$$\begin{cases} q = x_1 \rightarrow Q_{nc} = -(C_1\dot{x}_1 + C_2(\dot{x}_1 - \dot{x}_2)) + f_1 \\ q = x_2 \rightarrow Q_{nc} = -(C_3\dot{x}_2 + C_2(\dot{x}_2 - \dot{x}_1)) + f_2 \end{cases} \tag{103}$$

Com $Q_{nc}$ avaliado, a avaliação dos termos continua como:

$$\begin{cases} q = x_1 \rightarrow \frac{d}{dt}\frac{\partial T(\dot{q},t)}{\partial \dot{q}} = m_1\ddot{x}_1 \\ q = x_2 \rightarrow \frac{d}{dt}\frac{\partial T(\dot{q},t)}{\partial \dot{q}} = m_2\ddot{x}_2 \end{cases} \tag{104}$$

$$\begin{cases} q = x_1 \rightarrow \frac{\partial V(q,t)}{\partial q} = k_1 x_1 + k_2(x_1 - x_2) \\ q = x_2 \rightarrow \frac{\partial V(q,t)}{\partial q} = k_3 x_2 + k_2(x_2 - x_1) \end{cases} \tag{105}$$

Desta forma, é obtido um sistema de equações diferenciais, possível de ser escrito da seguinte forma:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

(106)

## A1.2 - Modelagem Matemática - Sistema Carro-Pêndulo

Para avaliar o comportamento dinâmico do sistema nas duas coordenadas do plano cartesiano, define-se $R_x$ e $R_y$ como as reações incógnitas do sistema, $N$ como a força normal aplicada ao Carro. Entende-se $cg$ como sendo o centro de gravidade do pêndulo e assume-se $u = f$ de antemão. Com isto, é possível avaliar:

$$\sum F_x = m_1\ddot{x} = -c_1\dot{x} - kx - R_x + u \tag{107}$$

$$\sum F_y = 0 = -R_y - m_1 g + N \tag{108}$$

E do pêndulo tem-se:

$$\sum \vec{F} = m_2 \begin{bmatrix} \ddot{x}_{cg} \\ \ddot{y}_{cg} \end{bmatrix} = \begin{bmatrix} Rx \\ -m_2 g + R_y \end{bmatrix} \tag{109}$$

$$\sum M_o = -\beta\dot{\theta} - m_2 g l sin(\theta) \tag{110}$$

Devido ao referencial do pêndulo ser o carro, deve-se avaliar que o sistema não possui referencial inercial. Com isto, é preciso desenvolver as acelerações angulares no problema. Para tal, primeiramente avalia-se que o corpo de massa $m_2$ que é o pêndulo em si, tem um comportamento dinâmico dependente do carrinho com massa $m_1$. Desta forma, denota-se que $\boldsymbol{r}_{cg\to IN}$ como posição do centro de gravidade do pêndulo com relação ao referencial inercial $IN$ (que está no chão). Esta é composição da posição $\boldsymbol{r}_o$ do carro e da posição do $cg$ com relação ao carro, nomeada como $\boldsymbol{r}_{cg\to o}$ . Portanto, podemos definir $\boldsymbol{r}_{cg\to IN}$ como:

$$\boldsymbol{r}_{cg\to IN} = \boldsymbol{r}_o + \boldsymbol{r}_{cg\to o} \tag{111}$$

$$\boldsymbol{r}_{cg\to IN} = \begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -lsin(\theta) \\ lcos(\theta) \\ 0 \end{bmatrix} \tag{112}$$

Portanto, para encontrarmos a aceleração, precisamos derivar duas vezes no tempo:

$$\dot{\boldsymbol{r}}_{cg\to IN} = \begin{bmatrix} \dot{x} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -lcos(\theta) \\ -lsin(\theta) \\ 0 \end{bmatrix} \dot{\theta} \tag{113}$$

$$\ddot{\boldsymbol{r}}_{cg\to IN} = \begin{bmatrix} \ddot{x} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} lsin(\theta) \\ -lcos(\theta) \\ 0 \end{bmatrix} \dot{\theta}^2 - \begin{bmatrix} lcos(\theta) \\ lsin(\theta) \\ 0 \end{bmatrix} \ddot{\theta} \tag{114}$$

Desta forma, podemos avaliar que o somatório dos momentos presentes no ponto $o$, onde o pêndulo está fixado é definido por:

$$\sum \boldsymbol{M}_o = \begin{bmatrix} 0 \\ 0 \\ J\ddot{\theta} \end{bmatrix} + \begin{bmatrix} lsin(\theta) \\ lcos(\theta) \\ 0 \end{bmatrix} \times m_2 \ddot{\boldsymbol{r}}_{cg\to IN} \tag{115}$$

Desenvolvendo a equação, obtém-se duas equações, tais que:

$$\sum \boldsymbol{M}_o = \begin{bmatrix} 0 \\ 0 \\ (J + m_2 l^2)\ddot{\theta} - m_2 l\ddot{x}cos(\theta) \end{bmatrix} \tag{116}$$

Com isto, escreve-se

$$(J + m_2 l^2)\ddot{\theta} - m_2 l\ddot{x}cos(\theta) = -\beta\dot{\theta} - m_2 glsin(\theta) \tag{117}$$

$$(J + m_2 l^2)\ddot{\theta} - m_2 l\ddot{x}cos(\theta) + \beta\dot{\theta} + m_2 glsin(\theta) = 0 \tag{118}$$

Quanto as forças em $cg$, é possível analisar que:

$$m_2 \ddot{\boldsymbol{r}}_{cg\to IN} = \begin{bmatrix} Rx \\ -m_2 g + R_y \end{bmatrix} \tag{119}$$

Então, tem-se:

$$\begin{bmatrix} R_x \\ R_y \end{bmatrix} = m_2 \begin{bmatrix} \ddot{x} + l\sin\theta\dot{\theta}^2 - l\cos\theta \\ g - l(cos(\theta)\dot{\theta}^2 - sin(\theta)\ddot{\theta}) \end{bmatrix} \tag{120}$$

Uma vez que no eixo $y$ não há movimento por parte do carrinho, é possível simplesmente analisar apenas a equação para $x$, de tal forma a obter:

$$(m_1 + m_2)\ddot{x} + m_2 l(sin(\theta)\dot{\theta}^2 - cos(\theta)\ddot{\theta}) + c_1\dot{x} + kx = u \tag{121}$$

Implicando no conjunto de equações:

$$\begin{cases} (m_1 + m_2)\ddot{x} + m_2 l(sin(\theta)\dot{\theta}^2 - cos(\theta)\ddot{\theta}) + c_1\dot{x} + kx = u \\ (J + m_2 l^2)\ddot{\theta} - m_2 l\ddot{x}cos(\theta) + \beta\dot{\theta} + m_2 glsin(\theta) = 0 \end{cases} \tag{122}$$

# 10 Apêndice 2 - Códigos Implementados

## KugaPack.jl

Este código possui algumas funções relativas a geração do sinal numérico representativo da força $\boldsymbol{u}$ e do processamento do sistema não linear.

```
1
2  #tirar os zeros do inicio e refazer o proporcional r
3  function PulseGen(n)
4      r = round(Int,n/4);
5      Z = zeros(r);
6      O = ones(r);
7      S = [O; -O; Z;Z; 0];
8      return S'
9  end
10
11 #gera o dotE de acordo com a formula  o detalhada da IC
12 function dotEgen(X,u)
13
14   x = X[1];
15      = X[2];
16   v = X[3];
17      = X[4];
18
19   m2glsin = m2*l*sin(  );
20   m2glcos = m2*l*cos(  );
21
22   M = [
23     m1+m2 -m2glcos;
24     -m2glcos J0+m2*l^2 ;
25   ];
26
27   invM = inv(M);
28
29   F = [
30     c1*v+m2glsin*  ^2 + k*x-u;
31     beta*   + m2glsin
32   ];
33
34
35   return [
36     v;
37       ;
38     -invM*F
39     ];
40
41 end
42
43
44 #runge kutta 4
45 function RK4(X0,U,dt,maxT,n)
46     T = 0:dt:maxT;
47     X = zeros(size(X0)[1],n);
48     X[:,1] = X0;
```

```
49      h = dt;
50      for i=1:n-1
51        u = U[i];
52        k1 = dotEgen(X[:,i],u);
53        k2 = dotEgen(X[:,i]+0.5*k1*h,u);
54        k3 = dotEgen(X[:,i]+0.5*k2*h,u);
55        k4 = dotEgen(X[:,i]+0.5*k3*h,u);
56        hPhi = h*(k1+2*k2+2*k3+k4)/6;
57        X[:,i+1] = X[:,i] + hPhi;
58      end
59
60      C = [1 0 0 0; 0 1 0 0];
61
62      Y = C*X
63
64      return Y,X,T
65  end
66
67  function metrica(Y1,Y2)
68    return norm(Y2-Y1)/norm(Y1)
69  end
```

### sysid.jl

Todas as funções de identificação de sistemas implementadas

```
1
2   using Flux
3   using CUDA
4
5
6   #School of Mechanical Engineering - State University of Campinas
7   #Paulo Yoshio Kuga
8
9   #This package is beign written for "An lise de Robustez em M todos de ...
        Identifica  o de Sistemas", a undergraduate ressearch for PIBIC.
10
11  #Main content: sysid algorithms and it's support functions
12
13  #----------------------------------------GENERAL ...
        FUNCTIONS-------------------------------------------------
14
15  function CheckData(Y,ny,ns)
16      if ns < ny
17          Y = Y';
18          ny,ns = size(Y);
19      end
20      return Y, ny, ns
21  end
22
23  #----------------------------------------ERA RELATED ...
        FUNCTIONS-------------------------------------------------
24
25  function Hankel(h,p,ny)
```

```
26      h = h[:,2:p+1];
27      ch = round(Int,p/2); #colunas da matriz de hankel
28      rh = ch; #linhas da matriz de hankel
29      H = zeros(rh*ny,ch+1); #estabelecer matriz de hankel baseada no ...
            n m e r o  de linhas e entradas desejadas
30      for i in 1:rh #preenche a matriz de hankel (dificil explicar a ...
            logica, mas em suma tenta tomar os e s p a o s  j   ...
            p r  -determinados para preencher a matriz)
31          H[(1:ny).+((i-1)*ny),:] = h[:,(1:ch+1).+(i-1)]
32      end
33      H1 = H[:,1:ch];
34      H2 = H[:,(1:ch).+1];
35      return H1, H2
36  end
37
38  function ERA_K(Y,nx,nu,dt,p,ny)
39      H1,H2 = Hankel(Y,p,ny);
40      U, S, V = svd(H1); #faz a SVD
41      S = diagm(S[1:nx]); #transforma os autovalores do svd em uma matriz ...
            diagonal
42      U = U[:,1:nx]; #toma as colunas relativas a ordem do modelo
43      V = V[:,1:nx]; #confirmar com o prof a hip t e s e do V ser tomado ...
            como coluna devido a t r a n s p o s i  o e da d e f i n i  o do SVD em si
44      G = S.^0.5; #tira a raiz quadrada dos autovalores
45      Q = inv(G); #inverte-os
46      Or = U*G; #observabilidade do modelo
47      Cr = G*V'; #controlabilidade do modelo
48      A = Q*U'*H2*V*Q; #gera a matriz A
49      C = Or[1:ny,:]; #com base no n m e r o de entradas do modelo, gera o
50      B = Cr[:,1:nu]; #numero de entradas    um dado do problema
51      D = Y[:,1];
52      return ss(A,B,C,D,dt)
53  end
54
55  #Y: the system impulse response
56  #nx: model order
57  #nu: number of inputs
58  #dt: Y sample period
59  function ImpulseERA(Y,nx,nu,dt,p)
60      ny,ns = size(Y);
61      Y,ny,ns = CheckData(Y,ny,ns);
62      return ERA_K(Y,nx,nu,dt,p,ny)
63  end
64
65  #bulid the U matrix presented in Dirceu's Thesis.
66  function BulidU(U,p,nu,ns)
67      U = U[:,1:ns-1];
68      cut = ns-p  #cut parameter for lines
69      Hu = zeros(nu*cut,ns-1);
70      for i=1:cut
71          Hu[(1:nu).+((i-1)*nu),i:ns-1] = U[:,1:ns-1-(i-1)];
72      end
73      return Hu
74  end
75
76  function GRA(Y,U,nx,dt,p)
```

```
77      ny,ns = size(Y);
78      nu, = size(U); #yes, for now, i'm putting some faith in user.
79      Y,ny,ns = CheckData(Y,ny,ns);
80      U,nu,ns = CheckData(U,nu,ns);
81
82      Uh = BulidU(U,p,nu,ns);
83      Yh = Y[:,1:ns-1];
84
85      h = Yh*Uh'*inv(Uh*Uh');
86      return ERA_K(h,nx,nu,dt,p,ny)
87
88
89  end
90
91
92  #----------------------------------------ARX RELATED ...
        FUNCTIONS-------------------------------------------------
93
94  #Hankel pr -determinado
95  #np ordem do polinimio
96  #ns numero de amostras
97  function Hankel_PD(Y,np,linhas)
98      H = zeros(linhas,np); #estabelecer matriz de hankel baseada no ...
            n m e r o de linhas e entradas desejadas
99      ny,ns = size(Y);
100     Yf = reshape(Y,(ny*ns,1));
101     for i in 1:np
102         H[:,i] = ...
                Yf[(1:linhas).+(i-1)*ny]#reshape(Y[:,(1:(ns-np)).+(i-1)],linhas,1);
103     end
104     return H
105 end
106
107 function Hankel_PD2(U,np,lines)
108     nu,ns = size(U);
109     Ut = reshape(U,(1,ns*nu));
110     F = zeros(lines,np*nu); #estabelecer matriz de hankel baseada no ...
            n m e r o de linhas e entradas desejadas
111     for i in 1:lines
112         F[i,:] = Ut[(1:np*nu).+(i-1)*nu];
113     end
114     return F
115 end
116
117 #Y: multiple output signal
118 #U: multiple input signal
119 #na: denominator polynomial order na≥nb
120 #nb: numerator polynomial order nb≥1
121 function ARX_K(Y,U, na, nb,dt)
122     ny,ns = size(Y); #determines the number of outputs and the number of ...
            samples
123     nu,_ = size(U); #determines the number of inputs
124     #if the signal or the input come as rows being the samples, it ...
            changes the order for columns being samples
125     Y,ny,ns = CheckData(Y,ny,ns);
126     U,nu,ns = CheckData(U,nu,ns);
```

```
127
128     linhas = (ns-na)*ny; #since we want na coefficients to our ...
            denominator, we need to organize the samples.
129     colunas = na+nb*ny*nu; #it's related to the number of coefficients.
130     H = zeros(linhas,colunas); #hankel matrix is reserved in memory
131
132     #for building our StateSpace system, we storage a matrix with the TF ...
            discrete type, to populate the array
133     G = Matrix{TransferFunction{Discrete{Float64}, ...
            ControlSystemsBase.SisoRational{Float64}}}(undef,ny,nu);
134
135
136     #professor kurka formulation for MIMO-ARX with least square
137     Id = Matrix{Float64}(I,ny,ny);
138     H[:,1:nb*ny*nu] = kron(Hankel_PD2(U,nb,ns-na),Id);
139     H[:,(1:na).+nb*ny*nu] = -Hankel_PD(Y,na,linhas);
140     b = reshape(Y[:,na+1:ns],linhas,1);
141
142     Coef = H'*H\(H'*b);
143     A = reverse(push!(Coef[(1:na).+nb*ny],1));
144     tfA = tf(1,A,dt);
145
146
147     B = Coef[1:(nb*ny)];
148
149
150     #TF functions assembly and asssignment to the G matrix (the TF ...
            function one)
151     Bc = zeros(ny,nb+1); #a new matrix is called
152     Bc[:,1:nb] = reverse(reshape(B,(ny*nu,nb)),dims=2); #it gets the ...
            coefficients and separe them into a nb row matrix, nu*ny ...
            columns. then the matrix is transposed, and reversed
153     for i=1:ny
154         for j = 1:nu
155             G[i,j] = 1/tf(1,Bc[i+j-1,:],dt)*tfA;
156         end
157     end
158
159
160
161     return array2mimo(G); #conversion to a mimo system
162     #returns the space state form of the system, not the TF.
163 end
164
165
166 #----------------------------------------NARX RELATED ...
        FUNCTIONS-------------------------------------------------
167
168 function ChainParameter(na,nb,p)
169     f1 = Dense(na+nb,p,  );
170     f2 = Dense(p,1,relu);
171     return Chain(f1,f2);
172 end
173
174 #does a ARX line conversion
175 function DataLoader(Y,U,na,nb,ny,nu,ns)
```

```
176     H = zeros(na*ny+nu*nb,ns-na); #this is the matrix who allow signal ...
            data to be inputed at NN
177
178     for i in 1:na
179         H[(1:ny).+(i-1)*ny,:] = Y[:,(1:ns-na).+(i-1)];
180     end
181
182
183     for i in 1:nb
184         H[(1:nu).+(na*ny+i-1),:] = U[:,(1:ns-na).+(i-1)];
185     end
186
187     return H
188 end
189
190 function NARX(Y,U, na, nb, p , dt)
191     U = [9 10 11 12]
192     Y = [1 2 3 4; 5 6 7 8]
193     na = 2
194     nb = 1
195     ny,ns = size(Y); #determines the number of outputs and the number of ...
            samples
196     nu,_ = size(U); #determines the number of inputs
197
198     #if the signal or the input come as rows being the samples, it ...
            changes the order for columns being samples
199     Y,ny,ns = CheckData(Y,ny,ns);
200     U,nu,ns = CheckData(U,nu,ns);
201
202     X = Float32.(DataLoader(Y,U,na,nb,ny,nu,ns));
203     Yr = Float32.(Y[:,na+1:ns]);
204
205
206
207     p =na*ny+nb*nu-1
208
209
210
211     f1 = Dense(na*ny+nb*nu,p,  );
212     f2 = Dense(p,ny,relu);
213     model = Chain(f1,f2);
214
215     function L(x,y)
216         return Flux.mse(x,y);
217     end
218
219     Flux.train!(L,model,zip(X,Yr),Descent);
220
221 end
```

### blockpack.jl

Simula o sistema Massa Mola Amortecedor para um conjunto de parâmetros.

```
1
2  function F2DOF(dt,Param)
3
4      m1 = Param[1];
5      m2 = Param[2];
6
7      c1 = Param[3];
8      c2 = Param[4];
9      c3 = Param[5];
10
11     k1 = Param[6];
12     k2 = Param[7];
13     k3 = Param[8];
14
15     M = [m1 0; 0 m2];
16     C = [c1+c2 -c2; -c2 c2+c3];
17     K = [k1+k2 -k2; -k2 k2+k3];
18
19     Id = [1 0; 0 1];
20     Z = [0 0; 0 0];
21
22     invM = inv(M);
23
24     A = [Z Id; -invM*K -invM*C];
25     B = [Z; invM];
26     B = B[:,1];
27     C = [1 0 0 0; 0 1 0 0];
28     #D = [0 0;0 0];
29
30     sistema = ss(A,B,C,0);
31     sistemaD = c2d(sistema,dt);
32
33     return sistema, sistemaD
34 end
35
36 #gera um sinal normal normatizado manualmente
37 function noise_gen(seed,r,c)
38     merst = MersenneTwister(seed); #mersenne twister generator
39     signal = randn(merst, Float64,(r,c)); #gaussian noise
40     mu = mean(signal,dims=2);
41     signal_mu = signal.-mu;
42     across = (maximum(signal.-mu,dims=2) - minimum(signal.-mu,dims=2))/2;
43     return signal_mu./across
44 end
```

### analysis.jl

A metodologia de comparação do sistemas foi implementada neste código

```
1  #School of Mechanical Engineering - State University of Campinas
2  #Paulo Yoshio Kuga
3
4  #This package is beign written for "An lise de Robustez em M todos de ...
       Identifica o de Sistemas", a undergraduate ressearch for PIBIC.
```

```
5
6  #Main content: automated sys analysis verifications
7
8  #gets the amplitude and the phase angle of a signal shifted fourier ...
       transform
9  function FourierTransformAnalysis(Y)
10     F = fft(Y) |> fftshift;
11     #freqs = fftfreq(ns, fs) |> fftshift;
12     A = abs.(F)
13        = atan.(imag(F),real(F));
14     return A,  #,freqs
15 end
16
17 function FourierTransformAnalysis(Y,ns,fs)
18     F = fft(Y) |> fftshift;
19     freqs = fftfreq(ns, fs) |> fftshift;
20     A = abs.(F)
21        = atan.(imag(F),real(F));
22     return A,  ,freqs
23 end
24
25
26 #this is a non-linear case, where the signal is imputed. when the signal ...
       is non-linear,
27 #it's senseless to apply a bode transform, since we don't know the poles ...
       of the system.
28 function runAnalysis(sysT,Yor, U)
29
30     Ysim,_,_= lsim(sysT,U,t);
31
32     mOr, pOr = FourierTransformAnalysis(Yor);
33     msim, psim = FourierTransformAnalysis(Ysim);
34
35
36     timeNorm = norm(Ysim-Yor)/norm(Yor);
37     magNorm = norm(msim-mOr)/norm(mOr);
38     phaseNorm = norm(psim-pOr)/norm(pOr);
39
40     triple = [timeNorm; magNorm; phaseNorm]
41
42     return triple
43
44     return timeNorm
45
46 end
47
48 function runAnalysis(sysT,Yor)
49     Ysim,_,_= impulse(sysT,t);
50     timeNorm = norm(Ysim-Yor)/norm(Yor);
51     return timeNorm
52
53 end
54
55
56 function runAnalysis(sysT,Yor, U,mOr, pOr,t)
57     Ysim,_,_= lsim(sysT,U,t);
```

```
58      msim, psim, = bode(sysT,w);
59
60      timeNorm = norm(Ysim-Yor)/norm(Yor);
61      magNorm = norm(msim-mOr)/norm(mOr);
62      phaseNorm = norm(psim-pOr)/norm(pOr);
63
64      triple = [timeNorm; magNorm; phaseNorm]
65
66      return triple
67
68  end
69
70  function n4sidPatternArg(Yout,U,dt,nx)
71      Data = iddata(Yout,U,dt);
72      sys_n4sid = n4sid(Data, nx,zeroD=true);
73      return sys_n4sid
74  end
75
76
77
78
79
80  # _,_,p = damp(sysD);
81  # _,_,p4 = damp(sys_n4sid);
82  # _,_,pe = damp(sys_era);
83  # _,_,pa = damp(sys_arx);
84
85
86  # v_era = metrica(p,pe)
87  # v_n4sid = metrica(p,p4)
88  # v_arx = metrica(p,pa)
89
90
91
92
93  #sys_arx = arx(Data,2,4,stochastic=false) #arx    MISO
94
95
96
97  #Yi,Ti,Xi = lsim(sys_n4sid,U',t,x0=X0);
98
99  #plot(t,[X' Xi'],layout=4)
100
101 #processar o ru do
102 #avaliar a sensibilidade ao ru do
103
104
105 #FFTf1 = abs.(fft(U));
106 #fw = 500
107 #plot(FFTf1[1:fw])
```

### main.jl

O código principal para a obtenção dos resultados do Massa Mola Amortecedor.

```
#School of Mechanical Engineering - State University of Campinas
#Paulo Yoshio Kuga
#First Release: January 2022

#Present version: 20230715

#This is the main program that runs the analysis to reach out the ...
    ressearch results.

using LinearAlgebra
using ControlSystems
using Plots
using ControlSystemIdentification
using Random
using Statistics
using DelimitedFiles


include("KugaPack.jl")
include("sysid.jl")
include("blockpack.jl")
include("analysis.jl")


#numerical conditions are set up
dt = 1e-3;
maxT = 4;
ns = round(Int,maxT/dt)+1;
t = 0:dt:maxT;

#problem conditions are set up
x0 = [0; 0; 0; 0;];
U = 0.01*PulseGen(ns);

#using the given parameters, the system (discrete space state) is created
Param = [0.77 0.59 2.1 1.2 9 200 200 200];
sysP,sysD = F2DOF(dt,Param);

Y,_,_= lsim(sysD,U,t,x0=x0); #time response
magD,phaseD,w = bode(sysD); #frequency response

#Since F2DOF is a two degree freedom problem, we can state that the ...
    number of outputs is 2
#This implies the noise beign as (2,n)
#In this analysis, we are using MersenneTwister as the random number ...
    generator.

seed = 98832+5556594 #seed for mersenne twister
noise = noise_gen(seed,2,ns);

fineza = 1;

NAmp = [i*1e-2 for i in 0:fineza:100];

#identification parameters
nx = 4;
```

```
54 p = 700; #round(Int,ns/2);
55 na = 4;
56 nb = 1;
57
58
59 nsa, = size(NAmp);
60
61 NORMS = zeros(9,nsa);
62
63 #-----------------------------------------------------------ANALISE ...
       SOLTA-------------------------------------------------
64
65
66 Yout = Y+NAmp[3]*noise.*maximum(Y,dims=2);
67 sGRA = GRA(Yout,U,nx,dt,p);
68 sN4SID = n4sidPatternArg(Yout,U,dt,nx);
69 sARX = ARX_K(Yout,U,na,nb,dt);
70
71 YERA,_,_= lsim(sGRA,U,t);
72 YN4SID,_,_= lsim(sN4SID,U,t);
73 YARX,_,_= lsim(sARX,U,t);
74
75 plot(t,Yout',
76     label = ["x1 com Ru do" "x2 com Ru do"],
77     xlabel="Tempo (s)",
78     ylabel="Deslocamento (m)",
79     title = "Sa da com Ru do"
80     )
81
82 plot(t,[Y[1,:] ,YERA[1,:], YN4SID[1,:],YARX[1,:]],
83 label = ["Original" "ERA" "N4SID" "ARX"],
84 xlabel="Tempo (s)",
85 ylabel="Deslocamento (m)",
86 title = "Compara  o: Resultado dos Algoritmos"
87 )
88
89 function PlotaGrafico(sysT,legenda,w,channel)
90     msim, psim, = bode(sysT,w);
91     msim = reshape(msim,(2,240));
92     psim = reshape(psim,(2,240));
93     y = msim[channel,:];
94     return plot(w,20*log.(y),xaxis=:log,label=legenda,legend = ...
           :outertopleft)
95 end
96
97 function PlotaGrafico!(sysT,legenda,w,channel)
98     msim, psim, = bode(sysT,w);
99     msim = reshape(msim,(2,240));
100    psim = reshape(psim,(2,240));
101    y = msim[channel,:];
102    return plot!(w,20*log.(y),xaxis=:log,label=legenda,legend = ...
           :outertopleft)
103 end
104
105 PlotaGrafico(sGRA,"ERA",w,1)
106 PlotaGrafico!(sN4SID,"N4SID",w,1)
```

```
PlotaGrafico!(sARX,"ARX",w,1)
#PlotaGrafico!(sysD,"Original",w,1)

magDts = reshape(magD,(2,240));
inp = magDts[1,:]
plot!(w,20*log.(inp),
xaxis=:log,
label="Original",
legend = :outertopleft,
title="Diagrama de Bode - Canal x1",
xlabel="Frequ ncia (rad/s)",
ylabel="Amplitude (dB)")


#gerar a an lise
#-----------------------------------------------------ANALISES ...
    ROBUSTEZ-----------------------------------------------

for i in 1:nsa
    ruido = NAmp[i]*noise.*maximum(Y,dims=2);
    Yout = Y+ruido

    NORMS[1:3,i] = runAnalysis(GRA(Yout,U,nx,dt,p),Y, U,magD, phaseD,t);
    NORMS[4:6,i] = runAnalysis(n4sidPatternArg(Yout,U,dt,nx),Y, U,magD, ...
        phaseD,t);
    NORMS[7:9,i] = runAnalysis(ARX_K(Yout,U,na,nb,dt),Y, U ,magD, phaseD,t);
    print("*")

end



# U = zeros(1,ns);
# U[1] = 1;
# Y,_,_= lsim(sysD,U,t,x0=x0); #time response

# NORMS1 = zeros(9,nsa);

# for i in 2:nsa
#     ruido = NAmp[i]*noise.*maximum(Y,dims=2);
#     Yout = Y+ruido

#     NORMS1[1:3,i] = runAnalysis(GRA(Yout,U,nx,dt,p),Y, U,magD, phaseD);
#     NORMS1[4:6,i] = runAnalysis(n4sidPatternArg(Yout,U,dt,nx),Y, ...
    U,magD, phaseD);
#     NORMS1[7:9,i] = runAnalysis(ARX_K(Yout,U,na,nb,dt),Y, U,magD, phaseD);

# end


AVAL = NORMS;
plot(
    NAmp,
    [AVAL[1,:] AVAL[3,:] AVAL[7,:]],
    labels=["GRA (ERA)" "N4SID" "ARX"],
    ylabel = "  ",
```

```
160     xlabel = "Percentual de Amplifica  o do ru do",
161     title  = "   com r e l a   o a Resposta Temporal",
162 )
163
164 amplitude = plot(
165     NAmp,
166     [AVAL[2,:] AVAL[4,:] AVAL[8,:]],
167     labels=["GRA (ERA)" "N4SID" "ARX"],
168     ylabel = "  ",
169     xlabel = "Percentual de Amplifica  o do ru do",
170     title  = "   com r e l a   o a Amplitude (Transformada de Bode)",
171 );
172
173
174
175 fase = plot(
176     NAmp,
177     [AVAL[3,:] AVAL[5,:] AVAL[9,:]],
178     labels=["GRA (ERA)" "N4SID" "ARX"],
179     ylabel = "  ",
180     xlabel = "Percentual de Amplifica  o do ru do",
181     title  = "   com r e l a   o a Fase (Transformada de Bode)",
182 );
183
184
185 png(amplitude,"amplitude_l.png")
186 png(fase,"fase_l.png")
187
188 #plot(amplitude, fase, layout=(2,1))
189
190
191
192 # plot(
193 #     t,
194 #     Y',
195 #     labels=["P o s i   o do Bloco 1" "P o s i   o do Bloco 2"],
196 #     ylabel = "Deslocamento (m)",
197 #     xlabel = "Tempo (s)",
198 #     title  = "Resposta para o sistema linear",
199 # )
```

### main_nl.jl

O código principal para a obtenção dos resultados do Carro-Pêndulo.

```
1
2 using LinearAlgebra
3 using ControlSystems
4 using Plots
5 using ControlSystemIdentification
6 using Random
7 using Statistics
8 using DelimitedFiles
9 using FFTW
```

```
10
11
12 include("KugaPack.jl")
13 include("sysid.jl")
14 include("blockpack.jl")
15 include("analysis.jl")
16
17 #Constantes
18 g = 9.81;#[m/s^2]
19 l = 0.3302;#[m]
20
21 #Inercia
22 m1 = 0.38+0.37;#[kg]
23 m2 = 0.23;#[kg]
24 J0 = 7.88e-3;#[kg*m^3]
25
26 #Amortecimento
27 c1 = 5.4;#[Ns/m^2]
28 beta = 0.0024;#[N*m*s/rad]
29
30 #Rigidez
31 k = 142;#[N/m]
32
33 dt = 5e-2;
34 maxT = 100;
35 ns = round(Int,maxT/dt)+1;
36 t = 0:dt:maxT;
37 fs = 1/dt;
38
39
40 U_or = PulseGen(ns);
41
42
43 fineza = 1;
44 MaxAmp = 100;
45 NAmp = [i for i in 0:fineza:MaxAmp].*1e-2;
46
47
48 nsa, = size(NAmp);
49 X0 = zeros(4,nsa)#1);
50 NORMS = zeros(9,nsa);
51
52 #identification parameters
53 nx = 4
54 p = 300; #round(Int,ns/2);
55 na = 4
56 nb = 1
57
58 Max  = pi/2
59
60
61 X0[2,:] = Max *NAmp';
62
63 #-----------------------------------------------------------ANALISE ...
       SOLTA-----------------------------------------------
64 U = U_or
```

```
65
66 Yout,_,_ = RK4(X0[:,4],0.01*U,dt,maxT,ns);
67
68 plot(t,Yout',
69 label = ["x" "  "],
70 xlabel="Tempo (s)",
71 ylabel="Deslocamento (m e rad)",
72 title = "Sistema para       = 2.7 ")
73
74 sGRA = GRA(Yout,U,nx,dt,p);
75 sN4SID = n4sidPatternArg(Yout,U,dt,nx);
76 sARX = ARX_K(Yout,U,na,nb,dt);
77
78 YERA,_,_= lsim(sGRA,U,t);
79 YN4SID,_,_= lsim(sN4SID,U,t);
80 YARX,_,_= lsim(sARX,U,t);
81
82
83 plot(t,[Yout[1,:] ,YERA[1,:], YN4SID[1,:],YARX[1,:]],
84 label = ["Original" "ERA" "N4SID" "ARX"],
85 xlabel="Tempo (s)",
86 ylabel="Deslocamento (m)",
87 title = "Compara  o: Resultado dos Algoritmos"
88 )
89
90 function PlotaGrafico(Y,legenda,channel,ns,fs)
91     msim, psim, freq = FourierTransformAnalysis(Y,ns,fs);
92     y = 2*msim[channel,1001:2001];
93     w = freq[freq .≥ 0];
94     return plot(w,20*log.(y),label=legenda,legend = :outertopleft)
95 end
96
97 function PlotaGrafico!(Y,legenda,channel,ns,fs)
98     msim, psim, freq = FourierTransformAnalysis(Y,ns,fs);
99     y = 2*msim[channel,1001:2001];
100    w = freq[freq .≥ 0];
101    return plot!(w,20*log.(y),label=legenda,legend = :outertopleft)
102 end
103
104 PlotaGrafico(YERA,"ERA",1,ns,fs)
105 PlotaGrafico!(YN4SID,"N4SID",1,ns,fs)
106 PlotaGrafico!(YARX,"ARX",1,ns,fs)
107 PlotaGrafico!(Yout,"Original",1,ns,fs)
108
109 title!("FFT Amplitude - Canal x")
110 xlabel!("Frequ ncia (rad/s)")
111 ylabel!("Amplitude (dB))")
112
113
114
115 #-----------------------------------------------------ANALISES ...
        ROBUSTEZ-------------------------------------------------
116
117
118 for i in 1:nsa
119     U = U_or#*NAmp[i]
```

```
120     Y,_,_ = RK4(X0[:,i],0.01*U,dt,maxT,ns);
121     NORMS[1:3,i] = runAnalysis(GRA(Y,U,nx,dt,p),Y, U);
122     NORMS[4:6,i] = runAnalysis(n4sidPatternArg(Y,U,dt,nx),Y, U);
123     NORMS[7:9,i] = runAnalysis(ARX_K(Y,U,na,nb,dt),Y, U);
124     print("|")
125 end
126
127    = rad2deg.(X0[2,:])
128
129 AVAL = NORMS;
130 tempo = plot(
131       ,
132     [AVAL[1,:] AVAL[4,:] AVAL[7,:]],
133     labels=["GRA (ERA)" "N4SID" "ARX"],
134     ylabel = "  ",
135     xlabel = "C o n d i   o inicial em Graus (  )",
136     title  = "    com  r e l a   o a Resposta Temporal",
137 );
138
139 amplitude = plot(
140       ,
141     [AVAL[2,:] AVAL[4,:] AVAL[8,:]],
142     labels=["GRA (ERA)" "N4SID" "ARX"],
143     ylabel = "  ",
144     #xlabel = "C o n d i   o inicial em Graus (  )",
145     title  = "    com  r e l a   o a Amplitude (FFT)",
146 );
147
148
149 fase = plot(
150       ,
151     [AVAL[3,:] AVAL[5,:] AVAL[9,:]],
152     labels=["GRA (ERA)" "N4SID" "ARX"],
153     ylabel = "  ",
154     xlabel = "C o n d i   o inicial em Graus (  )",
155     title  = "    com  r e l a   o a Fase (FFT)",
156 );
157 png(amplitude,"amplitude_nl.png")
158 png(fase,"fase_nl.png")
159
160
161 plot(amplitude, fase, layout=(2,1))
162
163
164
165
166 Y,_,_ = RK4([0,0,0,0],U_or,dt,maxT,ns)
167
168 sistema = plot(
169     t,
170     Y',
171     labels=["P o s i   o do Carro" "P o s i   o do P ndulo"],
172     ylabel = "Deslocamento (m) e \n  ngulo  do P ndulo (rad)",
173     xlabel = "Tempo (s)",
174     title  = "Resposta para o sistema  n o -linear",
175 );
```

```
176
177 png(sistema,"pendulo.png")
```