



UNIVERSIDADE FEDERAL DA BAHIA
ESCOLA POLITÉCNICA

JESSICA SOUZA DA SILVA

**FERRAMENTA INTERATIVA PARA ANÁLISE DE SISTEMAS NÃO
LINEARES**

Salvador

2019

JESSICA SOUZA DA SILVA

**FERRAMENTA INTERATIVA PARA ANÁLISE DE SISTEMAS NÃO
LINEARES**

Trabalho de conclusão de curso apresentado ao colegiado do curso de Engenharia de Controle e Automação de Processos da Universidade Federal da Bahia como requisito para obtenção do título de Bacharela em Engenharia de Controle e Automação de Processos.

Orientador: Prof^o. Me. Daniel Diniz Santana

Salvador

2019

JESSICA SOUZA DA SILVA

**FERRAMENTA INTERATIVA PARA ANÁLISE DE SISTEMAS
NÃO LINEARES**


Trabalho de conclusão de curso apresentado ao Colegiado do Curso de Graduação em Engenharia de Controle e Automação de Processos da Universidade Federal da Bahia como requisito para obtenção do título de Bacharela em Engenharia de Controle e Automação de Processos.

29 de novembro de 2019.

Comissão examinadora:



Profº Me. Daniel Diniz Santana
Mestre em Engenharia Industrial
Universidade Federal da Bahia



Profº. Dr. Marcus Vinícius Americano da Costa Filho
Doutor em Engenharia de Automação e Sistemas
Universidade Federal da Bahia



Profº Me. Raony Maia Fontes
Mestre em Engenharia Industrial
Universidade Federal da Bahia

Salvador
2019

RESUMO

Este Trabalho tem por objetivo desenvolver a ferramenta computacional PEBICLI (abreviação de Ponto de Equilíbrio, Bifurcação e Ciclo Limite) para análise de sistemas autônomo de equações diferenciais ordinárias não lineares de segunda ordem. A ferramenta será capaz de caracterizar pontos de equilíbrio, verificar ocorrências de ciclos limites e de bifurcações. O objetivo é proporcionar auxílio didático e computacional na disciplina ENGF97 – Controle e Sistemas Não lineares do curso de graduação de Engenharia de Controle e Automação de Processo da UFBA. Serão apresentados guia de uso da ferramenta e o detalhamento da solução proposta.

Palavras-chave: ponto de equilíbrio, ciclo limite, bifurcação, MATLAB.

ABSTRACT

This paper aims to develop the computational tool PEBICLI (abbreviation of equilibrium point, bifurcation and limit cycle) for autonomous system analysis of second order nonlinear ordinary differential equations. The tool will be able to characterize equilibrium points, check for occurrences of limit cycles and bifurcations. The aim is to provide didactic and computational assistance for the discipline ENGF97 - Control and Nonlinear Systems of the Control and Automation Processes Engineering undergraduate course at UFBa. It will be presented user's tool guide and the detailing of the proposed solution.

Keywords: equilibrium point, limit cycle, bifurcation, MATLAB.

SUMÁRIO

| | | |
|----------|---|-----------|
| 1 | INTRODUÇÃO | 7 |
| 1.1 | SOLUÇÃO PROPOSTA | 8 |
| 1.2 | OBJETIVO GERAL | 9 |
| 1.3 | OBJETIVOS ESPECÍFICOS | 9 |
| 1.4 | ESTRUTURA DO TRABALHO | 10 |
| 2 | CONCEITOS TEÓRICOS..... | 11 |
| 2.1 | SISTEMAS DINÂMICOS NÃO LINEARES | 11 |
| 2.2 | PONTOS DE EQUILÍBRIO | 11 |
| 2.3 | LINEARIZAÇÃO | 12 |
| 2.4 | CICLO LIMITE..... | 14 |
| 2.4.1 | EXISTÊNCIA DE CICLOS LIMITES | 14 |
| 2.4.2 | NÃO EXISTÊNCIA DE CICLOS LIMITES..... | 16 |
| 2.5 | BIFURCAÇÃO | 16 |
| 3 | APRESENTAÇÃO DA FERRAMENTA | 22 |
| 4 | PROJETO DA FERRAMENTA | 25 |
| 4.1 | LINGUAGEM..... | 26 |
| 4.2 | IMPLEMENTAÇÃO..... | 27 |
| 4.3 | LIMITAÇÕES COMPUTACIONAIS..... | 43 |
| 5 | GUIA DE USO | 45 |
| 6 | CONCLUSÃO | 56 |
| 6.1 | TRABALHOS FUTUROS | 56 |
| | REFERÊNCIAS..... | 58 |

1 INTRODUÇÃO

Os métodos utilizados para análise de sistema dinâmico linear implicam em algumas considerações que são válidas apenas para uma faixa restrita. Todavia, como os processos reais são praticamente todos não lineares e, de acordo com Slotine (1991), faz-se necessário em muitos casos o estudo do sistema em uma escala maior no qual a faixa de operação é considerada grande. Ainda segundo Slotine (1991), os sistemas físicos podem ser descritos por equações diferenciais não-lineares e isso explica o grande interesse existente na área de pesquisa que aborda a análise destes sistemas.

O estudo voltado à Sistemas Não Lineares tem como requisito um aprofundado direcionamento para as análises qualitativa e quantitativa destes sistemas, onde se faz necessário o conhecimento consolidado de conceitos importantes. Características inerentes ao sistema, como por exemplo, análises dos pontos de equilíbrio, interpretação dos retratos de fase, verificação da existência de ciclo limite e condições de bifurcação são conhecidas através de aplicações de ferramentas, critérios e teoremas fundamentais para uma conclusão coesa sobre o sistema estudado. Monteiro (2006) justifica a análise de sistemas dinâmicos necessária para conhecer e compreender o comportamento futuro (ou explicar o passado) de modo científico. Segundo Franklin et al (2013), as abordagens de análise de sistemas não lineares são utilizadas no projeto de controle não linear.

Para Belhot et al (2001), durante a consolidação dos conteúdos através do método tradicional de aprendizagem, o aluno assimila informações expostas na sala de aula como um passivo receptor de conhecimentos. Em algumas áreas da ciência, depara-se com dificuldade de aprendizagem ao lidar com conceitos abstratos, conforme Fiolhais e Trindade (2003) relatam no processo de ensino-aprendizagem de fenômenos físicos, onde os conceitos são complexos e difíceis de visualizar e deveriam ser instruídos através de técnicas atraentes. No estudo de sistemas não lineares, alguns fenômenos são melhor compreendidos através da visualização gráfica computacional, a exemplo da bifurcação.

De acordo com Coelho Neto e Imamura (2005), a utilização dos recursos computacionais tornou-se uma das melhores ferramentas utilizadas no apoio aos estudos, tanto para docentes quanto para estudantes e pesquisadores na área de educação, devido aos resultados pedagógicos benéficos que são proporcionados. A interatividade com aparatos computacionais desenvolve uma aprendizagem diferenciada e, conforme Neto e Schuvartz (2007), consegue promover uma boa sinergia entre a teoria e sua aplicação. Segundo Martins

e Maschio (2014), o uso de softwares educativos permite desenvolver novos formatos de atividades interativas.

Para Silva et al. (2017), a absorção de conteúdos matemáticos que possuem um grau de dificuldade em suas resoluções analíticas é melhor compreendida e verificada a partir da tecnologia como facilitadora, porém nem sempre existem ferramentas computacionais disponíveis para auxiliar o processo de aprendizagem. Para análises de sistemas não lineares, existe uma carência quanto às aplicações computacionais direcionadas ao suporte no aprendizado dos conteúdos. Ferramentas como PPLANE e AURALIUS são alguns dos poucos recursos disponíveis. De acordo com Polking e Arnold (1999), a ferramenta PLANE permite ao usuário plotar curvas de solução no plano de fase, além de encontrar pontos de equilíbrio, exibir linearizações e plotar separatrizes. Inspirado no PPLANE, Manurung (2017) relata que AURALIUS gera o retrato de fase de sistemas de terceira ordem.

Diante da carência de ferramentas computacionais disponíveis para as análises das características de sistemas não lineares e devido as existentes abrangerem apenas retrato de fase, surgem oportunidades de enriquecer esta área ampliando recursos computacionais para auxílio de outras análises, de forma a colocar o estudante frente à problemática de análise de sistemas dinâmicos considerando as não linearidades presentes nas aplicações práticas.

1.1 SOLUÇÃO PROPOSTA

Para estimular o aprendizado através de ferramentas computacionais e devido o cenário destas estarem em quantidades deficitárias, tem-se como contribuição a criação de uma ferramenta computacional interativa para análises de sistemas não lineares de segunda ordem, utilizando a aplicação Guide (do inglês *Graphical User Interface Development Environment*), do software MATLAB. Batizada de PEBICLI (abreviação de Ponto de Equilíbrio, Bifurcação e Ciclo Limite), esta ferramenta interativa viabiliza um apoio ao aprendizado da obtenção e análises das características de sistemas não lineares por meio de interfaces gráficas que serão apresentadas ao usuário através de objetos gráficos e virtuais, permitindo que o usuário: verifique os pontos de equilíbrio do sistema com suas respectivas classificações; verifique se o sistema possui ciclo limite através da aplicação dos Teoremas de Poincaré Bendixson, de Bendixson e de Poincaré, proporcionando também a validação da região de ocorrência do ciclo limite; visualize graficamente o fenômeno de bifurcação através de alterações dos

parâmetros do sistema. Visando aperfeiçoar a aprendizagem, a ferramenta pode ser utilizada como apoio didático para diminuir as dificuldades dos alunos na compreensão do conteúdo, possibilitando torná-lo mais interessante e atraente do ponto de vista pedagógico e incentivando-os também a explorar o conteúdo de maneira mais integrada ao ciclo do curso.

Na disciplina Controle e Sistemas Não Lineares, o aluno põe em prática os conhecimentos adquiridos resolvendo manualmente exercícios teóricos, porém nem sempre dispõe de solucionários de livros da área de sistemas não lineares ou de exercícios elaborados pelos docentes. Com a ferramenta será possível fazer estas análises de forma rápida e automática, permitindo realizar mudança nos valores de alguns parâmetros do sistema estudado, o que pode mudar completamente a análise e requerendo novos cálculos. Uma interface gráfica informará ao usuário os resultados dos testes e análises requeridas. Dessa forma, é possível tanto agilizar o desenvolvimento de exercícios quanto verificar se o resultado obtido de forma automática coincide com o obtido de forma manual.

1.2 OBJETIVO GERAL

Criação da ferramenta computacional interativa para análises de sistemas não lineares, utilizando os recursos do Guide, no software MATLAB, através de uma interface gráfica amigável, possibilitando apoio e consulta rápida, otimizando assim o processo de aprendizado e o tornando mais atraente do ponto de vista pedagógico.

1.3 OBJETIVOS ESPECÍFICOS

De acordo o objetivo geral, destacam-se as finalidades do trabalho:

- a) Analisar comportamentos de sistemas autônomos de equações diferenciais ordinárias não lineares de segunda ordem de maneira interativa;
- b) Servir de complemento ao programa PPLANE, o qual gera retratos de fases de sistema não lineares;
- c) Validar os resultados de exercícios resolvidos manualmente;
- d) É possível que existam programas computacionais de variados autores para auxiliar, isoladamente, em uma das análises de sistemas não lineares (retrato de fases, ponto de equilíbrio, bifurcação ou ciclo limite). A ferramenta permitirá concentrar os três

blocos de conteúdo (ponto de equilíbrio, bifurcação e ciclo limite) em um único programa, permitindo não somente verificação matemática como também gráfica.

1.4 ESTRUTURA DO TRABALHO

A disposição deste trabalho é a seguinte: no segundo capítulo serão discutidos brevemente os conceitos teóricos de classificações de ponto de equilíbrio, ciclo limite e bifurcação. No terceiro capítulo será apresentada a ferramenta. O quarto capítulo mostra os principais recursos utilizados na elaboração da mesma bem como alguns tópicos da implementação e limitações. Um guia de uso está contido no capítulo 5, com alguns exemplos para os quais o software é testado.

2 CONCEITOS TEÓRICOS

Nesta seção serão introduzidos os conceitos associados aos métodos de análise de sistemas não lineares de segunda ordem, objetos de estudo deste trabalho. Serão abordados alguns fenômenos de sistemas dinâmicos, incluindo linearização de sistemas, classificação de pontos fixos, ciclo limite e bifurcação.

2.1 SISTEMAS DINÂMICOS NÃO LINEARES

De acordo com Ogata (1998), um sistema é dito não linear quando não obedece ao princípio da superposição. Desta forma, a resposta do sistema a diversas entradas simultâneas é diferente da soma das respostas às entradas individuais, levando o sistema para a instabilidade ou mesmo não sendo observando nenhuma variação nas variáveis de estado.

Conforme Silva (2006), a dinâmica de um sistema não linear é extremamente rica, em comparação àquela de um sistema linear, devido a fenômenos significativos que não podem ser descritos em aproximações lineares. São exemplos de fenômenos essencialmente não lineares, segundo Von Zuben (2003): tempo de escape finito, múltiplos pontos de equilíbrio, dependência da amplitude de excitação, não unicidade da solução, dependência crítica dos parâmetros, bifurcações, ciclos limites e dependência crítica das condições iniciais.

2.2 PONTOS DE EQUILÍBRIO

Seja o sistema não linear de segunda ordem descrito por:

$$\begin{cases} \dot{x}_1(t) = f_1[x_1(t), x_2(t), \mu] \\ \dot{x}_2(t) = f_2[x_1(t), x_2(t), \mu] \end{cases} \quad (1)$$

em que $t \in \mathbb{R}^+$ e f_1 e f_2 funções suaves de x_1 e de x_2 , μ é o parâmetro do sistema ($\mu \in \mathbb{R}$).

De acordo com Slotine (1991), um ponto de equilíbrio é um ponto em que os estados do sistema ficam estacionários. Iniciado o sistema nesse ponto, o sistema permanece indefinidamente neles, mantidas as entradas constantes.

A determinação dos pontos de equilíbrio de um sistema consiste em igualar às derivadas dos estados a zero, encontrando-se os respectivos valores de estados estacionários, conforme:

$$\dot{x}_i = 0 \quad (i=1,2,\dots,n) \quad (2)$$

em que n é o número de estados.

2.3 LINEARIZAÇÃO

Sistemas linearizados de segunda ordem possuem duas componentes da variável de estado designadas por x_1 e x_2 , sendo representados pela seguinte forma:

$$\begin{cases} \dot{x}_1(t) = a_1x_1(t) + b_1x_2(t) \\ \dot{x}_2(t) = a_2x_1(t) + b_2x_2(t) \end{cases} \quad (3)$$

em que $t \in \mathbb{R}^+$ e $a_i, b_j, x_1, x_2 \in \mathbb{R}$, sendo:

$$\begin{aligned} a_1 &= \left. \frac{\partial f_1(x_1, x_2)}{\partial x_1} \right|_{PE}, & b_1 &= \left. \frac{\partial f_1(x_1, x_2)}{\partial x_2} \right|_{PE} \\ a_2 &= \left. \frac{\partial f_2(x_1, x_2)}{\partial x_1} \right|_{PE}, & b_2 &= \left. \frac{\partial f_2(x_1, x_2)}{\partial x_2} \right|_{PE} \end{aligned} \quad (4)$$

A técnica da linearização de sistemas não lineares consiste em encontrar a Equação (3) a partir de (1), em torno do ponto de equilíbrio (PE).

Reescrevendo o sistema (3) na forma matricial, tem-se:

$$\dot{x} = Ax \quad (5)$$

em que,

$$A = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \text{ e } X = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \quad (6)$$

Um autovalor da matriz A (conhecida por matriz Jacobiana) é designado por λ caso exista um vetor não nulo v tal que:

$$Av = \lambda v \quad (7)$$

De acordo com Jordan (2007), o vetor v é chamado de autovetor. A Equação (7) pode ser rearranjada da seguinte forma:

$$(A - \lambda \cdot I)v = 0 \quad (8)$$

em que I é a matriz identidade.

Ainda segundo Jordan (2007), sabe-se pela teoria algébrica que a Equação (8) possui soluções diferentes de zero para v somente se o determinante de $(A - \lambda \cdot I)$ for igual a zero. Portanto,

$$\det(A - \lambda \cdot I) = 0 \quad (9)$$

Calculando o referido determinante, encontra-se Equação (10), a qual é chamada de equação característica:

$$\lambda^2 - (a_1 + b_2)\lambda + (a_1b_2 - b_1a_2) = 0 \quad (10)$$

Agrupando as constantes e fazendo $m = (a_1 + b_2)$ e $n = (a_1b_2 - b_1a_2)$, obtém-se:

$$\lambda^2 - m\lambda + n = 0 \quad (11)$$

Denota-se Δ como sendo o discriminante de uma equação do segundo grau:

$$\Delta = m^2 - 4n \quad (12)$$

As soluções da Equação (11) correspondem a:

$$\lambda_1 = \frac{-m + \sqrt{\Delta}}{2} \quad \lambda_2 = \frac{-m - \sqrt{\Delta}}{2} \quad (13)$$

De acordo com Monteiro (2006), um ponto de equilíbrio é classificado como hiperbólico se todos os autovalores tiverem parte real não nula. Caso contrário, tem-se um ponto de equilíbrio não hiperbólico.

Consoante os valores de λ_1 e λ_2 , observam-se as classificações dos pontos de equilíbrio hiperbólicos no Quadro 1:

Quadro 1 - Classificação dos pontos de equilíbrio

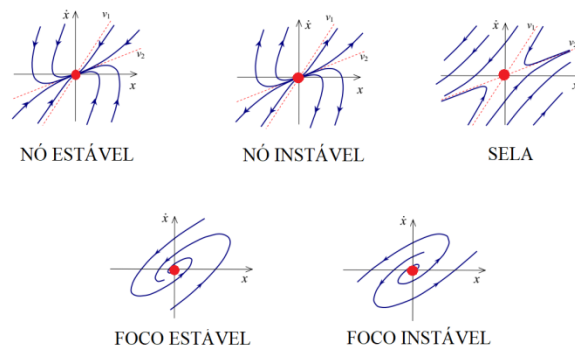
| Valores de λ_1 e λ_2 | Classificação |
|--------------------------------------|---------------|
| Reais e Negativos | Nó estável |
| Reais e Positivos | Nó instável |
| Complexos com parte real Negativa | Foco estável |
| Complexos com parte real Positiva | Foco instável |
| Reais e sinais contrários | Sela |
| Complexos com parte real Nula | Centro |

Fonte: Elaborado pela autora.

Quando ao menos um dos autovalores for nulo, não há classificação para o ponto de equilíbrio.

A Figura 1 mostra o esboço de cada tipo de ponto de equilíbrio.

Figura 1 - Classificação dos pontos de equilíbrio.



Fonte: Adaptado de Silva (2006).

2.4 CICLO LIMITE

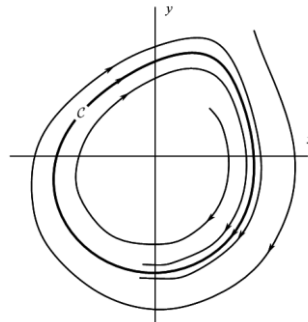
Para Oliveira e Rosolen (2011), os ciclos limite desempenham um fenômeno importante na teoria qualitativa dos sistemas dinâmicos e, de acordo com Strogatz (1994), podem ocorrer apenas em sistemas não lineares. Para determinar o comportamento qualitativo de um sistema, faz-se necessário saber quando existem órbitas periódicas nas proximidades do ponto de equilíbrio. Segundo Jordan (2007), um ciclo limite é uma solução periódica isolada de um sistema autônomo representado no plano de fase por um caminho fechado isolado.

O ciclo limite é uma trajetória periódica, pois, após um certo período T , os valores das variáveis retornam aos seus valores iniciais, ou seja:

$$(x_1(t + T), x_2(t + T)) = (x_1(t), x_2(t)) \quad (14)$$

Se o caminho fechado existir, representado na Figura 2 por C , as trajetórias nas suas proximidades devem se comportar de maneira parecida com C .

Figura 2 - Ciclo Limite.



Fonte: JORDAN 2007.

Segundo Silva (2006), um ciclo limite é estável quando qualquer trajetória que se inicie na sua vizinhança se aproxima dele, quaisquer que sejam as condições iniciais. Caso contrário, é classificado como instável. Além disso, pode ser semi-estável se as trajetórias se aproximam por um lado, mas se afastam pelo outro.

2.4.1 EXISTÊNCIA DE CICLOS LIMITES

O teorema de Poincaré-Bendixson fornece uma condição para a existência de órbitas periódicas em sistemas não lineares de segunda ordem. O teorema não será demonstrado neste trabalho, porém, conforme Khalil (2002), poderá ser observado um corolário do teorema que

resume como o teorema é aplicado. Refere-se a esse corolário como critério de Poincaré-Bendixson.

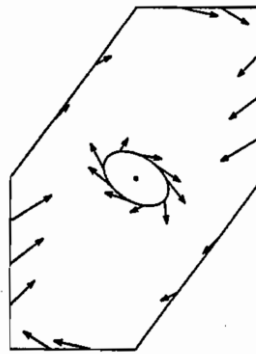
Critério de Poincaré Bendixson: Considere um sistema autônomo de segunda ordem e M um subconjunto fechado de plano de estado tal que:

- i) M não contém nenhum ponto de equilíbrio ou contém um ponto de equilíbrio com parte real positiva (nó ou foco instável);
- ii) Toda trajetória partindo de M permanece em M , $\forall t$.

Logo, M contém uma órbita periódica.

De acordo com Khalil (2002), ao redefinir o conjunto M , representado na Figura 3, para excluir a região delimitada por uma simples curva fechada continuamente diferenciável, tem-se um conjunto livre de pontos de equilíbrio e todas as trajetórias ficam presas nele.

Figura 3 - Redefinição do conjunto M para excluir a vizinhança de um foco ou nó instável.



Fonte: KHALIL, 2002.

Para encontrar uma região delimitada, especificamente de forma circular, Perko (1991) converte o sistema descrito pela Equação 1 em coordenadas polares, expressando-a em termos de raio r e de ângulo θ , de forma que:

$$x_1 = r \cdot \cos\theta, x_2 = r \cdot \sin\theta \quad (15)$$

Considerando-se que $r^2 = x_1^2 + x_2^2$, $\theta = \tan^{-1}\left(\frac{x_2}{x_1}\right)$ e derivando-se implicitamente, tem-se:

$$r \cdot \dot{r} = x_1 \cdot \dot{x}_1 + x_2 \cdot \dot{x}_2 \quad (16)$$

e

$$r^2 \cdot \dot{\theta} = x_1 \cdot \dot{x}_2 - x_2 \cdot \dot{x}_1 \quad (17)$$

Segundo Enns e McGuire (1997), a partir das Equações 16 e 17 é possível encontrar uma região delimitada, sendo r_{min} o limite interno e r_{max} o limite externo, de forma que:

$$\begin{cases} \dot{r} > 0, & \text{para } r_{max} \\ \dot{r} < 0, & \text{para } r_{min} \end{cases} \quad (18)$$

e, portanto, pelo Teorema de Poincaré Bendixson isso implica em uma órbita periódica circular.

2.4.2 NÃO EXISTÊNCIA DE CICLOS LIMITES

De acordo com Silva (2006) e Monteiro (2006), os teoremas a seguir fornecem condições suficientes para a inexistência de ciclos limites.

Teorema de Poincaré: Existirá ciclo limite no sistema autônomo de segunda ordem sistema se:

$$N = S + 1 \quad (19)$$

em que N número de nós e focos e S o número de pontos de sela.

Se a Equação (19) não for válida, não existe ciclo limite.

A demonstração deste teorema não será apresentada neste trabalho. “Como o corolário deste teorema tem-se se $S = 0$, $N = 1$, ou seja, um ciclo limite envolve necessariamente pelo menos um ponto de equilíbrio” (SILVA, 2006, p. 50).

Teorema de Bendixson: Conforme Savi (2006), se uma região Ω simplesmente conexa do plano de fases, a expressão $\frac{\partial f_1}{\partial x_1} + \frac{\partial f_2}{\partial x_2}$ não é identicamente igual a zero ou não mudar de sinal, então o sistema não possui ciclo limite.

2.5 BIFURCAÇÃO

“O termo bifurcação, introduzido por Poincaré em 1885, refere-se a mudança qualitativa no retrato de fases de um sistema dinâmico, conforme algum parâmetro do sistema passa por um valor crítico” (MONTEIRO, 2006, p.313). Essas mudanças do comportamento dinâmico podem ser do tipo: estável para instável, simétrico para assimétrico, estacionário para movimento regular, ordem para caos. Segundo Prado et al.(1994), as matrizes jacobianas, autovalores e autovetores dependerão de μ na Equação 1. Ao variar-se o parâmetro de

controle, podem mudar tanto a posição como as características qualitativas dos pontos de equilíbrio. Quando μ atinge um valor crítico, diz-se que é o ponto de bifurcação.

“O diagrama de bifurcação esboça uma medida da amplitude (ou norma) dos pontos de equilíbrio versus o parâmetro de bifurcação” (KHALIL, 2002, p.70). Linhas tracejadas representam pontos instáveis ou selas, enquanto linhas cheias representam pontos estáveis. As setas de direção indicam o sentido das trajetórias do sistema na vizinhança da região representada.

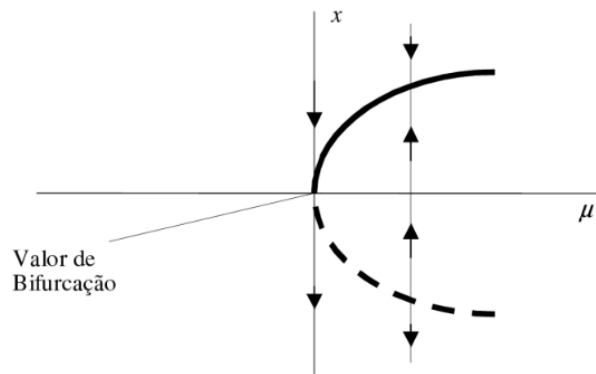
De acordo com Savi (2006), a teoria da bifurcação é normalmente desenvolvida em duas formas:

- i) Bifurcações locais: consiste em estudar a natureza das soluções com a variação dos parâmetros nas redondezas de um ponto de equilíbrio. Pode-se analisar a bifurcação através dos autovalores obtidos via linearização. São exemplos desse tipo a Bifurcação Sela-Nó, Transcrítica e Tridente e de Hopf.
- ii) Bifurcações globais: representam mudanças qualitativas nos aspectos globais do fluxo de um sistema dinâmico. Envolve mudanças nas bacias de atração e considerações sobre as órbitas do sistema e mudanças de estrutura. Não podem ser inferidas a partir de análises locais. São exemplos desse tipo as Bifurcações Homoclínicas e Heteroclínicas.

A seguir serão descritas características de alguns tipos de bifurcações.

Bifurcação Sela-Nó: é o caso em que um par de pontos fixos aparece em uma região onde não havia, com a variação do parâmetro. Segundo Alligood et al. (2006), na bifurcação Sela-Nó, dois pontos fixos do sistema, um estável e um instável, nascem à medida que o parâmetro é aumentado a partir do ponto de bifurcação, conforme Figura 4. Pode-se dizer também que dois pontos fixos coalescem no ponto de bifurcação.

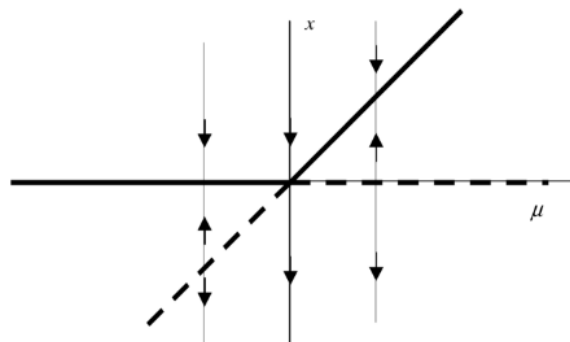
Figura 4 - Bifurcação Sela-Nó.



Fonte: Livro Dinâmica Não-linear e Caos (SAVI, 2006).

Bifurcação Transcrítica: “está relacionada a situações físicas onde um ponto de equilíbrio não pode ser destruído, mas altera as suas características associadas à estabilidade com a variação de um determinado parâmetro” (SAVI, 2006, p.148). Também conhecida por Bifurcação Induzida por limite, pode ser verificada na Figura 5

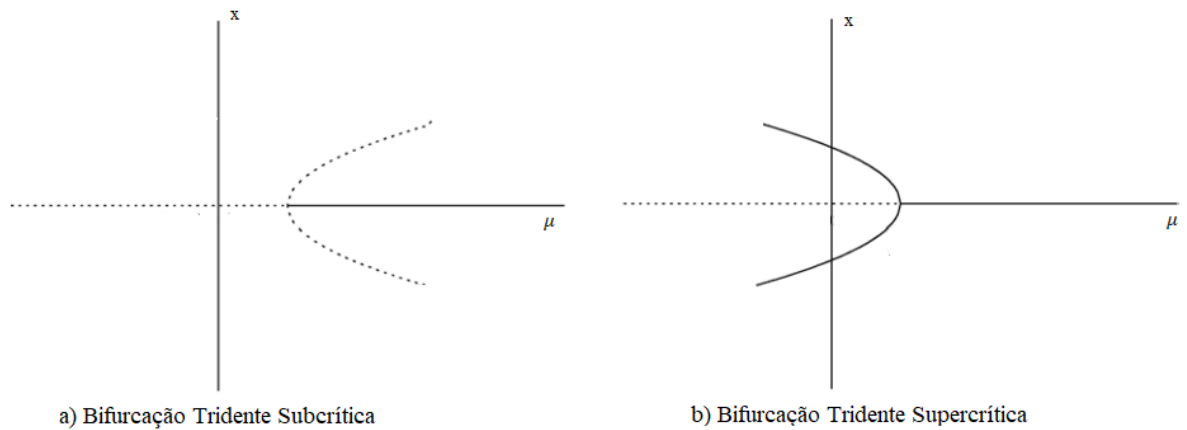
Figura 5 - Bifurcação Transcrítica.



Fonte: Livro Dinâmica Não-linear e Caos (SAVI, 2006).

Bifurcação Tridente: o ponto de bifurcação separa dois ramos de pontos de equilíbrio, conforme Figura 6. Na Bifurcação Tridente Supercrítica se tem, de um lado, um ramo de pontos de equilíbrio estáveis e, do outro lado, dois ramos de pontos de equilíbrio estáveis mais um ramo de pontos instáveis. No caso da Bifurcação Tridente Subcrítica, tem-se de um lado dois ramos de pontos de equilíbrio instáveis mais um ramo de pontos de equilíbrio estáveis e, do outro lado, um ramo de pontos instáveis. Também conhecida por Bifurcação Forquilha ou Por Quebra de Simetria.

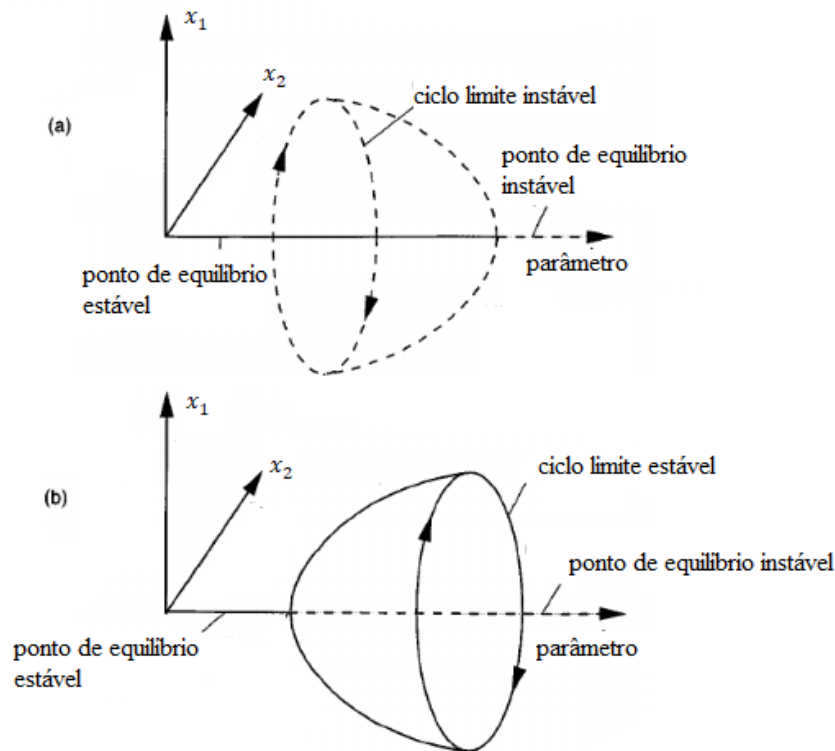
Figura 6 - Bifurcação Tridente.



Fonte: Elaborado pela autora.

Bifurcação Hopf: É caracterizada pela existência de um par de autovalores imaginários puros relacionados ao ponto de bifurcação. Segundo Chen et al. (2003) tem-se uma Bifurcação Hopf Subcrítica quando o ponto de equilíbrio estável e um ciclo limite instável coalescem e desaparecem e emerge um ponto de equilíbrio instável. Já na Bifurcação Hopf Supercrítica, o ponto de equilíbrio estável se bifurca em um ponto de equilíbrio instável, cercado por um ciclo limite estável. Os dois tipos de bifurcação Hopf pode ser verificada através da Figura 7.

Figura 7 - Bifurcação Hopf: Subcritica (a) e Supercritica (b).



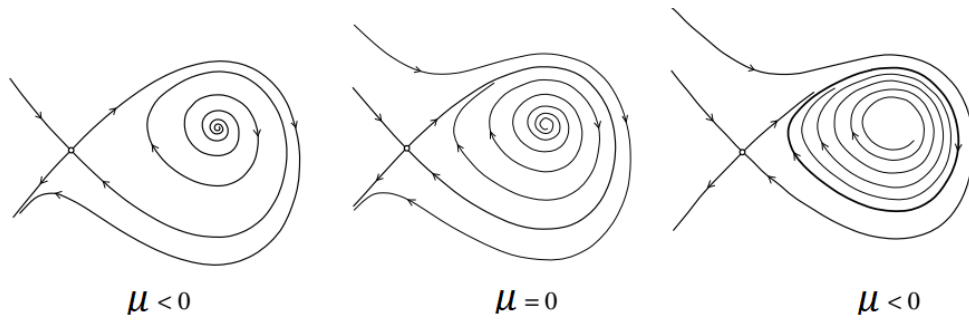
Fonte: Adaptado de Thompson e Stewart (1986).

Antes de conceituar as Bifurcações Homoclínica e Heteroclínica é necessário introduzir o significado de variedades estáveis e instáveis. Um ponto de sela é considerado instável, o que implica que valores iniciais próximos a ele tendem a se afastar. Segundo Alligood (2006), há um conjunto de valores iniciais que convergem para o ponto de sela quando $t \rightarrow +\infty$. A este conjunto dar-se o nome de variedade estável (caso $t \rightarrow -\infty$, denomina-se variedade instável).

Bifurcação Homoclínica: Um ponto é chamado de homoclínico se ele está em uma trajetória que é, ao mesmo tempo, variedade estável e variedade instável de um ponto de sela. “A trajetória percorrida por um ponto homoclínico é chamada de trajetória homoclínica” (MONTEIRO, 2006, p.349).

Uma bifurcação homoclínica, como pode ser vista através da Figura 8, leva a criação ou destruição de um ou mais ciclos limites variando parâmetro de bifurcação para valores pequenos. Ocorre quando parte de um ciclo limite se aproxima de um ponto de sela. Nesta bifurcação, o ciclo toca o ponto de sela e se torna uma órbita homoclínica.

Figura 8 - Evolução do Retrato de Fase com mudança do parâmetro.



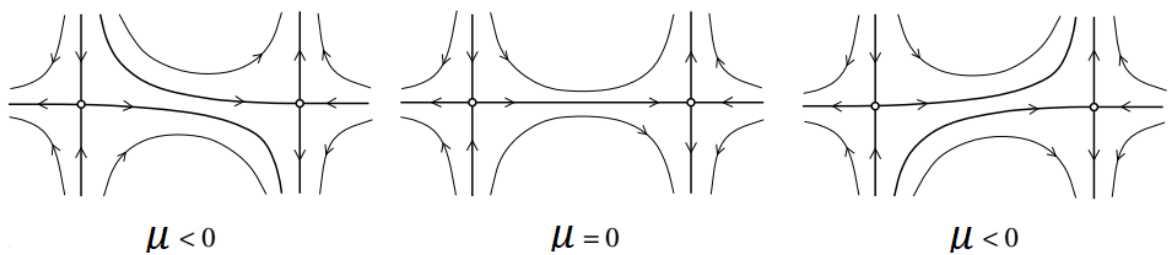
Fonte: Adaptado de Kuznetsov (1998).

Bifurcação Heteroclínica: Um ponto é chamado de heteroclínico se ele está na variedade instável de um ponto de sela e essa variedade é tangente à variedade estável de outro ponto de sela. A trajetória percorrida pelo ponto heteroclínico conecta dois pontos de sela e é chamada de trajetória heteroclínica.

Segundo Kuznetsov (1998), tem-se uma bifurcação heteroclínica quando a variação do parâmetro leva ao desaparecimento da trajetória heteroclínica.

A Figura 9 representa a bifurcação heteroclínica através da evolução do parâmetro de bifurcação no retrato de fase.

Figura 9 - Evolução do Retrato de Fase com mudança do parâmetro.



Fonte: Adaptado de Kuznetsov (1998).

3 APRESENTAÇÃO DA FERRAMENTA

A Figura 10 apresenta a tela principal da ferramenta desenvolvida no Guide/Matlab.

Figura 10 - Menu Principal da PEBICLI.

PEBiCLI FERRAMENTA

FERRAMENTA PEBICLI

EDO 2ª Ordem

=

=

Parâmetros

= = =

= = =

Módulos

Pontos de Equilíbrio **Bifurcação** **Ciclo Limite**

Universidade Federal da Bahia - Escola Politécnica, Dezembro de 2019

FECHAR

Fonte: Ferramenta PEBICLI - Software MATLAB.

O leiaute do menu é simples e dividido em três partes para auxiliar o usuário na sua utilização. Na parte superior está a criação do sistema de segunda ordem não linear, ou seja, a parte inicial da utilização da ferramenta. Logo abaixo está a localização dos campos para a entrada de dados dos parâmetros. Por último, no lado inferior da tela há três botões com os módulos da ferramenta.

Na Figura 11, tem-se o módulo Ponto de Equilíbrio. O sistema informado pelo usuário será carregado neste módulo. Consta de um botão de verificação e de uma listagem no qual serão exibidos todos os pontos de equilíbrios do sistema e sua respectiva classificação.

Figura 11 - Módulo Ponto de Equilíbrio.

Fonte: Ferramenta PEBICLI - Software MATLAB.

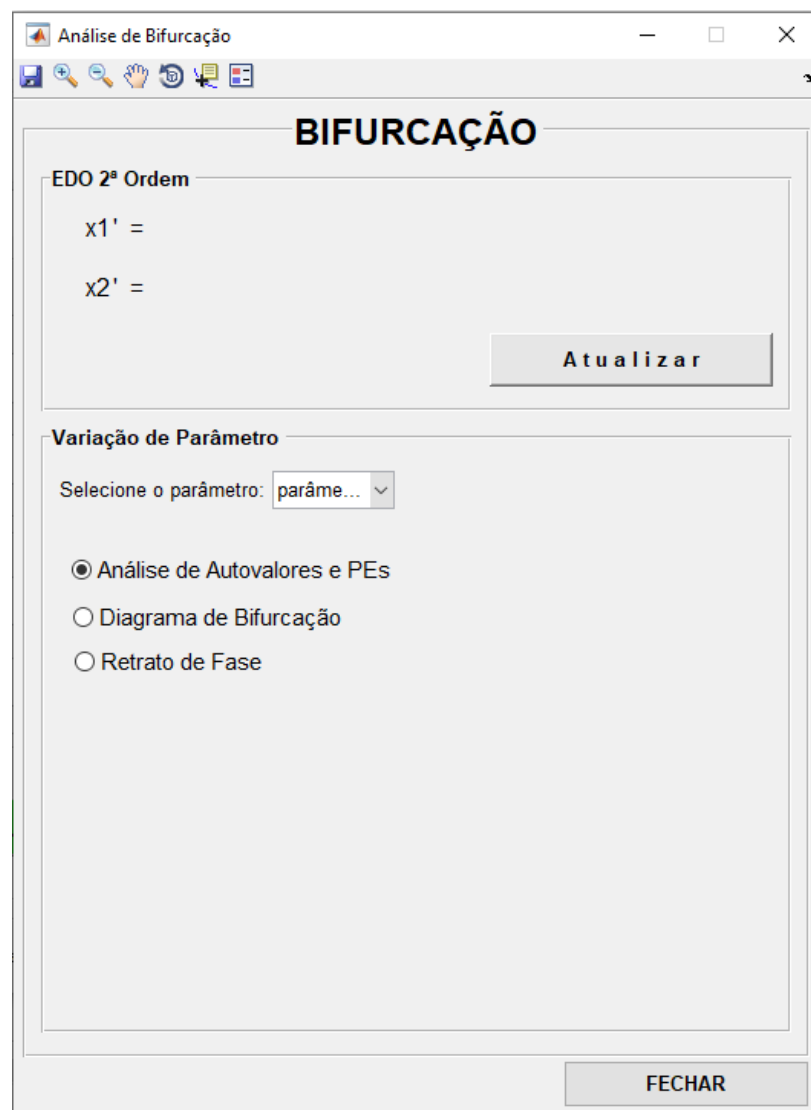
O módulo ciclo limite é composto por 3 partes: Poincaré, Bendixson e Poincaré Bendixson, conforme Figura 12. Na primeira e na segunda parte tem-se a validação da não existência ou da possibilidade de existência do ciclo limite através de um quadrante que exibirá na tela a informação. A terceira parte tem o objetivo de exibir na tela uma figura com a região, restrita ao formato de circunferência, de existência do ciclo limite pela resolução do Teorema de Poincaré Bendixson, sendo necessário que o usuário informe os limites dos eixos da figura a ser plotada.

Figura 12 - Módulo Ciclo Limite.

Fonte: Ferramenta PEBICLI - Software MATLAB.

Por fim, a ferramenta é completada com o módulo da Bifurcação, conforme Figura 13. Consta de Menu Pop-up para seleção do parâmetro de bifurcação. Três figuras podem ser geradas: análise dos pontos de equilíbrio e autovalores, diagrama de bifurcação e retrato de fase. Seu objetivo é mostrar as mudanças ocorridas no sistema com a variação do parâmetro através da análise visual dessas três figuras. Alguns tipos de bifurcações não serão exibidos através do Diagrama de Bifurcação, como a Heteroclínica e Homoclínica, sendo possível visualizar o fenômeno da bifurcação através do Retrato de Fase.

Figura 13 - Módulo da Bifurcação.

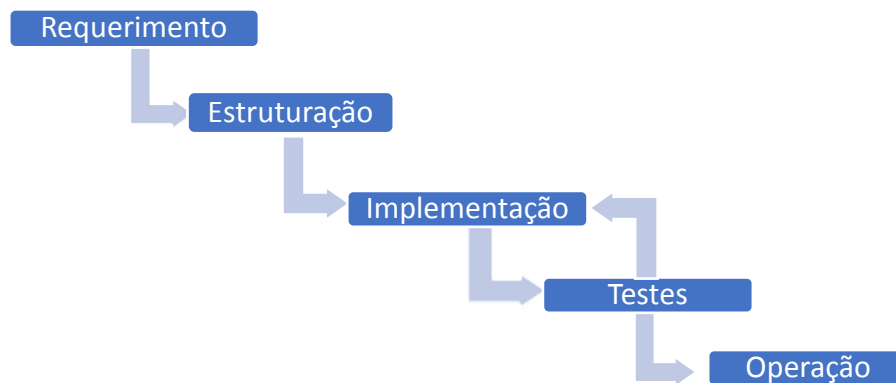


Fonte: Ferramenta PEBICLI - Software MATLAB.

4 PROJETO DA FERRAMENTA

O fluxograma a seguir ilustra as etapas para desenvolvimento da ferramenta, conforme Figura 14.

Figura 14 - Fluxograma das etapas do desenvolvimento do PEBICLI.



Fonte: Elaborado pela autora.

➤ Requerimento

Esta etapa inicial compreende a especificação dos recursos para elaboração da ferramenta, como a escolha da linguagem de programação. O ambiente de programação escolhido, o MATLAB, foi devido principalmente:

- Facilidade de compreensão, devido ao uso de funções mais elementares do MATLAB, incentivando aos usuários a aprender a trabalhar com essa ferramenta;
- Interface gráfica com o usuário, através do GUIDE, possibilitando interação com botões, campos de texto e menus;
- A ampla utilização, sendo utilizado como ambiente de outros programas na área de estudo de análise de sistemas não lineares, como o PPLANE.

➤ Estruturação

Cumprindo-se de um dos seus principais objetivos, o programa reuniu diversas funções para a análise de sistemas não lineares. Para tanto, adotou-se uma interface gráfica dividida em módulos, os quais foram definidos de acordo com assuntos abordados no Capítulo 2: Ponto de equilíbrio, Bifurcação e Ciclo limite.

Cada módulo foi estruturado em um diferente arquivo com extensão *.m. Essa estruturação permite melhor organização do código. As telas gráficas do Guide possuem o mesmo nome de arquivo de sua respectiva função, com extensão .fig.

- pebicly.m, pebicly.fig : Menu principal onde consta os três módulos, o sistema e seus parâmetros;
- pontosEquilibrio.m , pontosEquilibrio.fig: Exibe na tela as classificações dos pontos de equilíbrio;
- bifurcação.m , bifurcação.fig: Avalia se há bifurcação por parâmetro e plota o diagrama a bifurcação.
- cicloLimite.m e cicloLimite.fig: Avalia se há ciclo limite com base nos teoremas expostos no Capítulo 2.

➤ Implementação e Testes

Compreende a criação dos códigos e desenvolvimento da interface. Após estruturação da ferramenta e criação das telas gráficas, elaboraram-se as principais rotinas computacionais para exibir os resultados das análises dos sistemas. As funções foram testadas à medida que foram finalizadas e, caso algum erro acontecesse, revisava-se para correção o erro. Em seguida as algumas outras funções foram criadas a partir da necessidade de organização da ferramenta. Na implementação da ferramenta, pode-se verificar algumas limitações computacionais do código. O resultado desta etapa inclui a indicação das restrições e limitações da ferramenta.

➤ Operação

Essa etapa ocorre após conclusão da ferramenta, sendo formada pela instalação e operação da ferramenta. O usuário poderá encontrar erros que não foram previamente detectados ou identificar alguma necessidade durante a utilização. Sendo assim, essa etapa consiste em alterações e melhorias, podendo ser considerada etapa inicial de outros projetos.

4.1 LINGUAGEM

Para o desenvolvimento do produto, foi utilizado o software de programação e simulação MATLAB (abreviatura de Matrix Laboratory). Trata-se de um software poderoso e multifacetado de computação para cálculo numérico que auxilia estudantes e profissionais de ciências aplicadas e engenharia na resolução de modelos matemáticos, simulações, projetos de algoritmos, dentre outros.

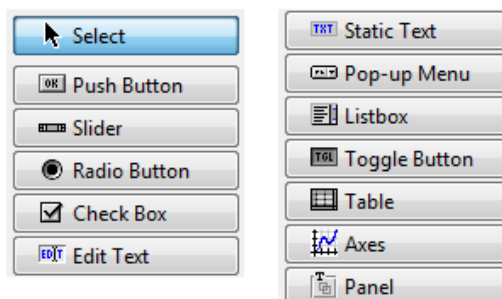
“O MATLAB apresenta uma série de funções matemáticas já implementadas que podem ser utilizadas em uma rotina construída pelo usuário. Estas funções são agrupadas de

acordo com a área de interesse em *toolboxes* e armazenadas em diretórios específicos” (FARINA; POSSER, 1999, p.3).

Segundo Chapman (2003), uma GUI (do inglês *Graphical User Interface*) proporciona ao usuário um ambiente familiar para trabalhar e contém componentes como botões, chaves, listas, menus, caixas de texto, etc. As entradas são denominadas eventos e o código executado em resposta a determinado evento é conhecido como chamada de retorno (conhecidas também por *callback*). Todos os componentes da interface são estruturas do tipo *handles*. Cada função de chamada de retorno é responsável por um único componente da GUI.

A Figura 15 mostra os principais componentes gráficos do Guide.

Figura 15 - Principais componentes do GUIDE.



Fonte: Mathworks¹.

4.2 IMPLEMENTAÇÃO

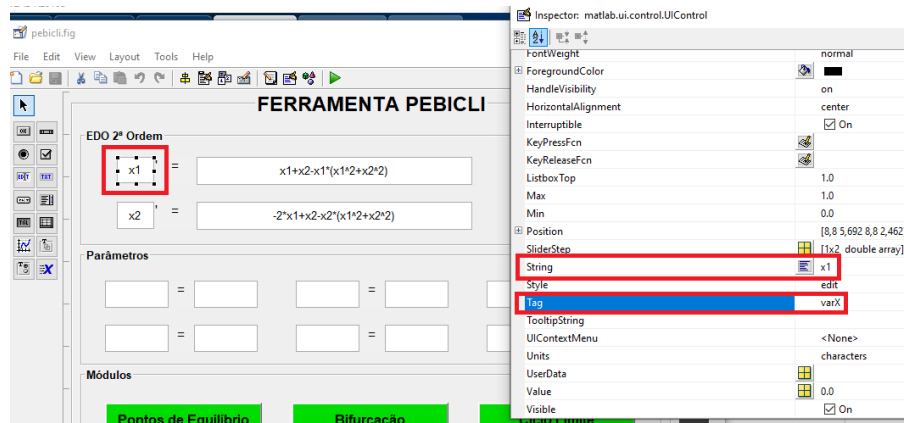
A seguir serão apresentadas as principais rotinas computacionais desenvolvidas para cada módulo.

i. Menu Principal

Uma tela principal foi pensada para centralizar as informações e o usuário informar à aplicação do sistema que pretende analisar. Logo, ela dispõe de caixas de texto para inserir e editar uma string referente ao sistema e seus parâmetros. Trata-se de componentes denominadas “Edit Text”. Os valores fornecidos pelo usuário podem ser obtidos acessando a propriedade “String” do componente. Como pode ser visto na Figura 16, cada componente da interface gráfica possui um Tag associado, o qual define o nome pelo qual estes são identificados no código.

¹ Disponível em: < https://www.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html#f3-998645 > Acesso em set. 2019.

Figura 16 - Componente Edit Text

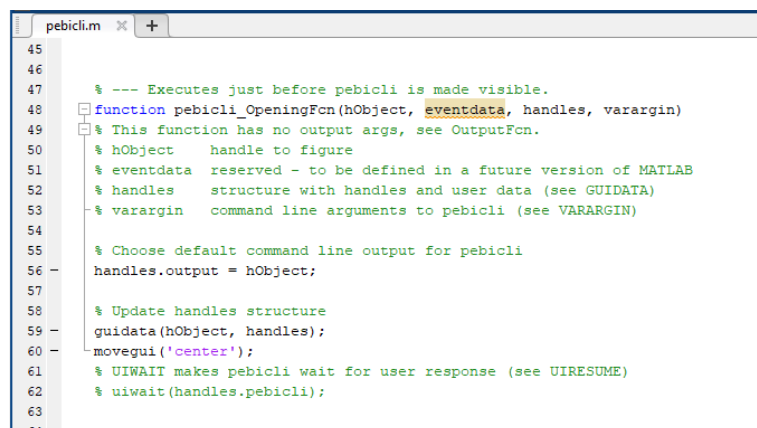


Fonte: Ferramenta PEBICLI - Software MATLAB.

Os botões de cada módulo são componentes do tipo Push Button e realizam a ação quando ativado por um clique do botão esquerdo do mouse. Esta ação de resposta do programa ao evento de clique do mouse sobre o botão foi programada em uma função *callback* que é disparada no instante em que se libera o clique no interior do botão. O nome da função de chamada de retorno será o valor da propriedade Tag do componente da GUI seguido dos caracteres “_Callback”.

Após selecionado o diretório e salvo a GUI, um arquivo .m foi gerado com as funções *callback* programadas. Neste arquivo .m, as funções *callback* relacionadas aos principais eventos são automaticamente geradas. Essas funções possuem comentário que indicam sua utilização, conforme exemplo da Figura 17.

Figura 17 - Comentários das Funções do Guide.



Fonte: Software Matlab.

`pebicly_OpeningFcn`: Função de abertura da ferramenta, que contém as propriedades iniciais dos componentes e tudo que deve ocorrer no momento em que o Menu Principal é aberto.

`pebicly_OutputFcn`: Função cujo retorno é exibido na janela de comando.

Cada um dos seis parâmetros possui as seguintes funções, onde () representa a numeração do parâmetro:

`param()_Callback`: Função que programa a resposta da ferramenta ao editar o texto da caixa de texto com Tag “`param()`”

`param()_CreateFcn`: Função executada no momento em que o componente “`param()`” é criado.

As mesmas funções são geradas para cada caixa de texto onde o usuário vai preencher com o valor do parâmetro, cujo tag é `valorparam()`.

As funções digitadas possuem tag `funcaoX` e `funcaoY` e seus valores `varX` e `varY`, respectivamente.

Os botões são funções as `PBpontosEquilibrio_Callback`, `PBbifurcacao_Callback` e `PBcicloLimite_Callback` e dentro de cada uma há a chamada do código que irá abrir os módulos.

A função do botão `FECHAR`, `PBfechar1_Callback`, chama a função `CloseRequestFcn` a qual exclui a figura usando o comando `delete`.

ii. Ponto de Equilíbrio

A função de abertura do módulo é a `pontosEquilibrio_OpeningFcn`, a qual contém a função `PBatualizarPE_Callback` cujo objetivo principal é recuperar o sistema e seus parâmetros informados no Menu Inicial.

A rotina computacional está contida na Listagem 1. O comando `findobj` localiza todos os objetos da GUI para os quais se atribuiu um valor à propriedade Tag no código `pebicly.m` e o comando `guidata` retorna dados armazenados anteriormente salvando na variável `pebiclydata`. Variáveis são declaradas como globais com o intuito de poder ser compartilhadas entre diversas funções. As informações preenchidas nos campos do Menu Principal são acessadas através do comando `get`. O comando `set` é utilizado para alterar uma determinada propriedade de um componente. Sendo assim, os textos estáticos do sistema e suas variáveis são exibidos na tela Ponto de Equilíbrio devido a este comando, que atribui à propriedade da `handle` os dados coletados pelo comando `get` ou dados declarados na função.

Listagem 1 – Obtenção dos dados do Menu Principal

```
h = findobj('Tag','pebicly');
% if exists (not empty)
global flag;
flag = 0; %False
if ~isempty(h)
    global pebiclyData;
    flag = 1; %True
    pebiclyData = guidata(h);

    global xvar yvar vars fg;
    xvar = get(pebiclyData.varX, 'string');
    yvar = get(pebiclyData.varY, 'string');

    set(handles.strVarXPE, 'string', xvar);
    set(handles.strVarYPE, 'string', yvar);

    strParOrdenado = strcat('(' , xvar, ', ', yvar, ')');

    set(handles.tabelaPontosClassificacao, 'columnname',...
        {strParOrdenado, 'Classificação (autovalores)'});

    xder = get(pebiclyData.funcaoX, 'string');
    yder = get(pebiclyData.funcaoY, 'string');

    param1 = get(pebiclyData.param1, 'string');
    val1 = get(pebiclyData.valorparam1, 'string');
    param2 = get(pebiclyData.param2, 'string');
    val2 = get(pebiclyData.valorparam2, 'string');
    param3 = get(pebiclyData.param3, 'string');
    val3 = get(pebiclyData.valorparam3, 'string');
    param4 = get(pebiclyData.param4, 'string');
    val4 = get(pebiclyData.valorparam4, 'string');
    param5 = get(pebiclyData.param5, 'string');
    val5 = get(pebiclyData.valorparam5, 'string');
    param6 = get(pebiclyData.param6, 'string');
    val6 = get(pebiclyData.valorparam6, 'string');
```

Fonte: Elaborado pela autora.

Para associar o parâmetro (tag pname) ao valor digitado pelo usuário (tag pval), tem-se a função assignParametros, a qual utiliza o comando assignin para atribuir cada valor de pval à uma nova variável indicada dentro de pname, conforme Listagem 2.

Listagem 2 – Associação do parâmetro a seu valor

```
function assignParametros (pname, pval)
    for i = 1:length(pname)
        assignin('caller',pname{i},str2double(pval{i}));
    end
```

Fonte: Elaborado pela autora.

A Listagem 3 verifica se há campos vazios de parâmetros (vetor pname) e de seus respectivos valores (vetor pval). Caso o usuário não tenha entrado com dados nos campos dos parâmetros, mas tenha digitado um valor associado ao parâmetro, o código elimina esses

valores. Se o usuário definiu o parâmetro, mas não atribui valor, o código efetua essa análise e atribui valor zero.

Listagem 3 – Verificação de campos vazios dos parâmetros

```
pname = {param1, param2, param3, param4, param5, param6};
pval = {val1, val2, val3, val4, val5, val6};

pval = pval(~cellfun('isempty',pname)); %elimina os valores cujas
variaveis não foram definidas

pos = cellfun('isempty',pval); % verifica se uma variavel foi
definida porém o valor não foi definido
if any(pos(:))
    pval{pos} = '0'; %Se uma variável foi definida e o valor não
foi definido, atribui-se valor zero para a variável
end

pname = pname(~cellfun('isempty',pname)); %mantém as variáveis que
foram definidas
assignParametros(pname, pval);
```

Fonte: Elaborado pela autora.

Da mesma forma que o programa verifica se o parâmetro e os valores estão vazios, verifica se as funções estão vazias e atribui valor zero, conforme Listagem 4.

Listagem 4 – Verificação de campos vazios das funções

```
vars = [sym(xvar), sym(yvar)]; %Cria as variáveis a partir dos
strings de entrada:
syms(xvar); syms(yvar);

if isempty(xder)
    xder=sym();
end
xder = sym(eval(xder)); %se entrada de função vazia, assume f(xvar,
yvar) = 0
if isempty(yder)
    yder=sym();
end
yder = sym(eval(yder));
```

Fonte: Elaborado pela autora.

Na Listagem 5 é possível verificar como foi desenvolvido o conteúdo principal do módulo Ponto de Equilíbrio através da função `PBverificar_Callback`, associada ao botão Verificar. A função `PBatualizarPE_Callback`, executada ao abrir o módulo, carrega todos os dados necessários para calcular a principal rotina deste módulo, a qual gera a classificação dos pontos.

O vetor chamado de `vars` contém as variáveis do sistema, em formato simbólico. Já o vetor chamado por `fg` contém as funções. A matriz jacobiana é calculada através do comando `jacobian` e os pontos de equilíbrios são encontrados através do `solve`. Este último comando encontra as soluções reais do sistema (`fg`) em função das variáveis (`vars`), obtendo solução

explícita para essas equações chamando o argumento 'MaxDegree'. A opção especifica o grau máximo de polinômios para os quais o argumento retorna soluções explícitas, encontradas na função MaxDegree, a qual verifica o grau através da quantidade de coeficientes do sistema.

Os autovalores da matriz jacobiana são encontrados através do comando eig. A execução da string é feita pelo comando eval. Conforme detalhado no Quadro 1 do referencial teórico, a rotina computacional armazena cada ponto de equilíbrio e sua classificação em uma célula, para ser exibido na tabela da Gui.

Listagem 5 – Classificação dos pontos de equilíbrio

```
J = jacobian(fg, vars);
S = solve(fg, vars, 'MaxDegree', MaxDegree, 'Real', true);
pEq = [S.(xvar), S.(yvar)]; %Pontos de Equilibrio

resultado = {};
if size(pEq,1) > 0
    for i=1:size(pEq,1)%avalia o número de soluções(linhas da
matriz)
        z = pEq(i,:);
        assignPontoEquilibrio(xvar, yvar, z);
        A = subs(J);
        lambda = eval(eig(A));
        lambda1 = lambda(1);
        lambda2 = lambda(2);
        if real(lambda1)<0 && real(lambda2)<0 &&...
            imag(lambda1)==0 && imag(lambda2)==0
            string1 = 'Há um Nó Estável.';
            string2 = '';
        elseif (real(lambda1)==0 && real(lambda2)==0)...
            && imag(lambda1)~=0 && imag(lambda2)~=0
            string1 = 'Há um Centro.';
            string2 = 'Nada se pode afirmar.';
        elseif real(lambda1)<0 && real(lambda2)<0 && ...
            imag(lambda1)~=0 && imag(lambda2)~=0
            string1 = 'Há um Foco Estável.';
            string2 = '';

        elseif real(lambda1)>0 && real(lambda2)>0 && ...
            imag(lambda1)~=0 && imag(lambda2)~=0
            string1 = 'Há um Foco Instável.';
            string2 = '';
        elseif real(lambda1)>0 && real(lambda2)>0 && ...
            imag(lambda1)==0 && imag(lambda2)==0
            string1 = 'Há um Nó Instável.';
            string2 = '';
        elseif (real(lambda1)<0 && real(lambda2)>0)...
            || (real(lambda1)>0 && real(lambda2)<0)
            string1 = 'Há um ponto de Sela.';
            string2 = '';
        else
            string1 = 'Há um ponto de equilíbrio.';
            string2 = 'Seu tipo específico não foi determinado.';
        end
    end
end
```

Fonte: Elaborado pela autora.

iii. Ciclo Limite

O módulo Ciclo Limite recupera os dados do Menu Principal da mesma forma que o módulo Pontos de Equilíbrio. A função de abertura do módulo, a `cicloLimite_OpeningFcn`, contém a função `PBatualizarCL_Callback`.

Na Listagem 6 contém a função `PBpoincare_Callback`, a qual verifica a quantidade de nós, focos e selas do sistemas e armazena nas variáveis `N`, `F` e `S`, respectivamente, e cria outra variável denominada `poincare = N+F-S`.

Listagem 6 – Poincaré

```
J = jacobian(fg, vars);
S = solve(fg, vars, 'MaxDegree', MaxDegree, 'Real', true);
pEq = [S.(xvar), S.(yvar)]; %Pontos de Equilibrio
%%%% VARIÁVEIS PARA CRITÉRIO POINCARÉ %%%%
N = 0; F = 0; S = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if size(pEq,1) > 0
    for i=1:size(pEq,1) %avalia o número de soluções (linhas da
        matriz)

            z = pEq(i,:);
            assignValorEquilibrio(xvar, yvar, z);
            A = subs(J);
            %Classificação:
            lambda = eval(eig(A)); %lambda: autovalores de A
            lambda1 = lambda(1);
            lambda2 = lambda(2);
            if real(lambda1)<0 && real(lambda2)<0 &&...
                imag(lambda1)==0 && imag(lambda2)==0
                N = N + 1;
            elseif (real(lambda1)==0 && real(lambda2)==0)...
                && imag(lambda1)~=0 && imag(lambda2)~=0
            elseif real(lambda1)<0 && real(lambda2)<0 && ...
                imag(lambda1)~=0 && imag(lambda2)~=0
                F = F + 1;
            elseif real(lambda1)>0 && real(lambda2)>0 && ...
                imag(lambda1)~=0 && imag(lambda2)~=0
                F = F + 1;
            elseif real(lambda1)>0 && real(lambda2)>0 && ...
                imag(lambda1)==0 && imag(lambda2)==0
                N = N + 1;
            elseif (real(lambda1)<0 && real(lambda2)>0)...
                || (real(lambda1)>0 && real(lambda2)<0)
                S = S + 1;
            else
            end
        end
    end
end
```

Fonte: Elaborado pela autora.

Se `poincare` for igual a 1, será exibido na tela que Pode haver ciclo limite. Caso contrário, será exibido na tela a string “Não há Ciclo Limite!”, conforme indicado na Listagem 7. A variável `poincareOk` funciona como um verificador da função `PBpoincare_Callback`, através do comando `if`. Sendo assim, se a função `PBpoincare_Callback` der que pode haver ciclo limite, a variável `poincareOk` assume valor 1 e assim é possível executar as instruções.

Listagem 7 – Avaliação Poincaré

```

##### ANALISE CRITERIO POINCARÉ #####
set(handles.poincareN, 'String', num2str(N+F));
set(handles.poincareS, 'String', num2str(S));
poincare = N+F-S;
if poincare == 1
    poincareOk = 1;
    set(handles.strExistenciaCicloLimite, 'string',...
        'Pode haver Ciclo Limite!');
    set(handles.poincarePanel, 'backgroundColor', [0.7,1,0.7]);
    set(handles.text18, 'backgroundColor', [0.7,1,0.7]);
    set(handles.text19, 'backgroundColor', [0.7,1,0.7]);
    set(handles.poincareN, 'backgroundColor', [0.7,1,0.7]);
    set(handles.poincareS, 'backgroundColor', [0.7,1,0.7]);

    set(handles.PBbendixson, 'Enable', 'On');
else
    set(handles.strExistenciaCicloLimite, 'string',...
        'Não há Ciclo Limite!');
    set(handles.poincarePanel, 'backgroundColor', [1,0.7,0.7]);
    set(handles.text18, 'backgroundColor', [1,0.7,0.7]);
    set(handles.text19, 'backgroundColor', [1,0.7,0.7]);
    set(handles.poincareN, 'backgroundColor', [1,0.7,0.7]);
    set(handles.poincareS, 'backgroundColor', [1,0.7,0.7]);
end
% FIM COROLÁRIO POINCARÉ

```

Fonte: Elaborado pela autora.

O teorema de Bendixson utiliza como principal recurso o comando `divergence`, o qual retorna a expressão da divergência do campo vetorial de f_g em relação as variáveis, declarada como `div`. Caso o divergente resulta em 0, o programa retorna na tela que “Pode haver Ciclo Limite”. Caso contrário, o programa analisa se houve mudança de sinal, conforme pode ser visto na Listagem 8.

Listagem 8 – Teorema de Bendixon avaliado pela mudança de sinal

```
s = solve(div, vars, 'MaxDegree', MaxDegree, 'Real', true);
assignin('base',xvar,...
[vpa(s.(xvar))-eps(1), vpa(s.(xvar))+eps(1)]);
assignin('base',yvar,...
[vpa(s.(yvar))-eps(1), vpa(s.(yvar))+eps(1)]);

if any(prod(sign(eval(subs(div))),2)<0)
    %se Verdade: houve mudança de sinal
    %SE MUDA DE SINAL = PODE TER CICLO LIMITE
    bendixsonOk = 1;
    set(handles.strMudancaSinal, 'string',...
        'Há mudança de sinal!');
    set(handles.strExistenciaCicloLimite, 'string',...
        'Pode haver Ciclo Limite!');
    set(handles.strMudancaSinal,...
        'backgroundColor', [0.7,1,0.7]);
    set(handles.bendixsonPanel,...
        'backgroundColor', [0.7,1,0.7]);
    set(handles.text15,...
        'backgroundColor', [0.7,1,0.7]);
else
    bendixsonOk = 0;
    % senão: Função não muda de sinal
    set(handles.strMudancaSinal, 'string',...
        'Não há mudança de sinal!');
    set(handles.strExistenciaCicloLimite, 'string',...
        'Não há Ciclo Limite!');
    set(handles.strMudancaSinal,...
        'backgroundColor', [1,0.7,0.7]);
    set(handles.bendixsonPanel,...
        'backgroundColor', [1,0.7,0.7]);
    set(handles.text15,...
        'backgroundColor', [1,0.7,0.7]);
end

end
```

Fonte: Elaborado pela autora.

O `solve` calcula o valor da função em um ponto, somado a um erro, e outro ponto, subtraído deste mesmo erro, o qual foi obtido através do comando `eps()`. O comando `assignin` é responsável por atribuir os valores a dois pontos (`xvar`, `yvar`). Se o produto dos sinais em cada ponto resultar em valor negativo (menor que zero), significa que houve mudança de sinal:

$++ = +$ ou $-- = +$; >0 : não há mudança de sinal

$+* = -$ ou $-* = -$; <0 : há mudança de sinal

Este código avalia a mudança de sinal referente às variáveis do sistema. Os parâmetros são substituídos numericamente, assumindo o valor inserido pelo usuário no Menu Principal. Sendo assim, não entram nessa análise de mudança de sinal, pois estão sendo considerados constantes invariáveis.

Da mesma forma que o Poincaré foi validado, atribui-se 1 a variável `bendixsonOk` quando o há mudança de sinal.

A terceira parte deste módulo é referente ao Teorema de Poincaré Bendixson. O botão “Verificar” exibirá na tela a figuraCicloLimite, região da ocorrência do ciclo limite.

O programa obtém os valores de entrada do usuário para plotar a figura. O comando `meshgrid` cria a malha onde a figura será plotada, retornando coordenadas da grade 2D. O comando `streamslice` chama a linhas de corrente bem espaçadas (com setas de direção) a partir de dados de volume vetor `fg`.

O objetivo da função `PBpoincareBendixson_Callback`, contida na Listagem 9, é encontrar a solução periódica do sistema. O método utilizado no programa consiste em construir uma região circular de captura, isto é, um conjunto conectado fechado de modo que o campo vetorial aponte "para dentro" em todos os lugares nos limites da região. Esta análise é efetuada pela transformação da equação em coordenadas polares, usando a derivação implícita da curva circular, $r\dot{r} = x\dot{x} + y\dot{y}$, e fazendo $\text{valuex1} = r \cos \theta$ e $\text{valuex2} = r \sin \theta$. O programa resolve sistema transformado ao longo do plano de coordenadas polares. Caso haja indeterminação em algum ponto, o programa atribui `NaN` no lugar da indeterminação para que o código seja executado sem interrupção.

Listagem 9 – Transformação em coordenadas polares

```
syms (xvar, yvar);
% Coordenadas polares:
syms r a
dx = eval(fg(1)); dy = eval(fg(2));
dr = (sym(xvar)*dx+sym(yvar)*dy)/r;
valuex1 = r*cos(a); valuex2 = r*sin(a);
assignValor(xvar, valuex1)
assignValor(yvar, valuex2)
drpol = eval(dr); drpol = simplify(drpol);

sr = solve(drpol, r, 'Real', true);

theta = linspace(0, 2*pi, 1000);

rho = NaN.*ones(length(sr),length(theta));
for i=1:length(theta)
    a = theta(i);
    try
        rho(:,i) = eval(subs(sr));
    catch
    end
end
```

Fonte: Elaborado pela autora.

Conforme Listagem 10, o programa plota os raios de modo que todas as trajetórias tenham um componente radial. Esta tarefa é equivalente a encontrar o raio para o qual $\dot{r} = 0$ para todos θ . A região é fechada e delimitada. Portanto, existe uma solução periódica.

Listagem 10 – Figura da solução periódica

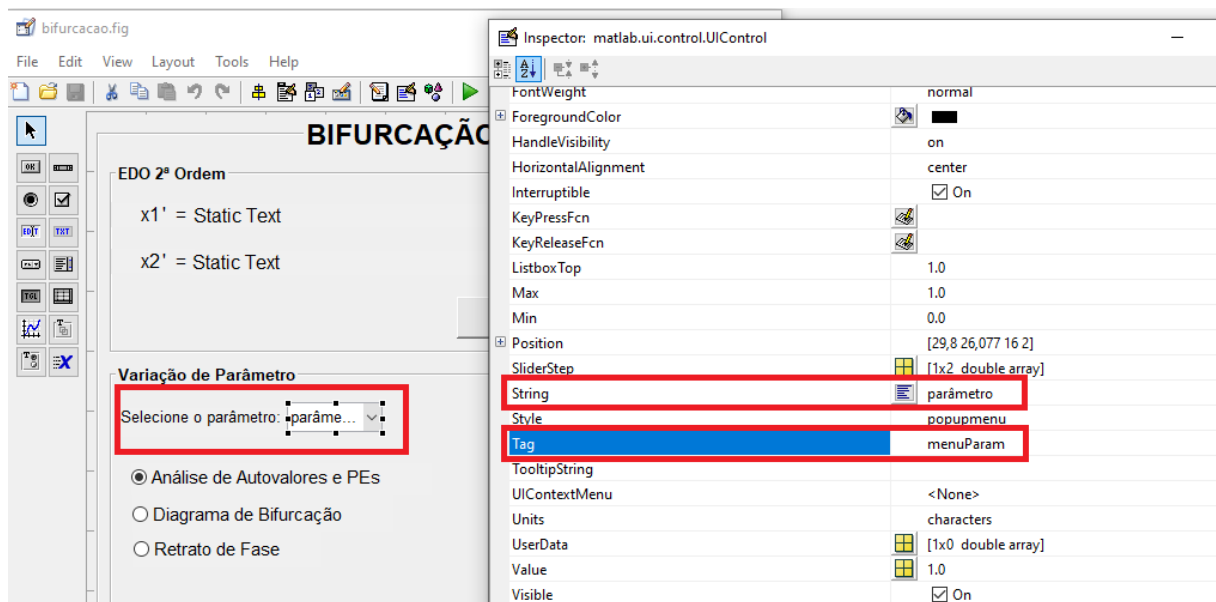
```
raio = [];  
for i=1:size(rho,1)  
    raio = [raio; real(max(rho(i,:))); real(min(rho(i,:)))];  
end  
raio = raio(raio>0);  
  
x1 = raio*cos(theta); x2 = raio*sin(theta);  
  
figure(figuraCicloLimite); hold on;  
plot(x1,x2,'b.', 'MarkerSize', 4);  
drawnow();
```

Fonte: Elaborado pela autora.

iv. Bifurcação

A função `bifurcacao_OpeningFcn` chama outras duas funções: `PBatualizarBI_Callback` e `sliderParamContinuousValueCallback`. A primeira recupera dados de entrada do Menu Inicial. Os parâmetros do Menu Principal são resgatados através do comando `set(handles.menuParam,'String',{'parâmetro', pname})`, o qual carrega no Menu Pop-up (Figura 18), as opções de parâmetros do sistema.

Figura 18 - Menu Pop-up



Fonte: Ferramenta PEBICLI - Software MATLAB.

A função `menuParam_Callback` refere-se ao Menu Pop-up e retorna na tela as opções de parâmetros do Menu Principal, permitindo ao usuário escolher um dos parâmetros do sistema para análise da bifurcação por parâmetro. Para se obter conteúdo do menu, utiliza-se o comando `cellstr`, que retorna o conteúdo do Menu Pop-up como matriz de células. O comando `contents` retorna o item selecionado do Menu, conforme Listagem 11.

Listagem 11 – Menu Pop-up

```
global xvar yvar vars muvar xder yder fg MaxDegree pname pval;
global figuraBifurcacao;
global flag flag2;

contents = cellstr(get(hObject, 'String'));
muvar = contents{get(hObject, 'Value')};
```

Fonte: Elaborado pela autora.

O comando `strcmp`, contido na Listagem 12, compara strings. Se a seleção for diferente da string 'parâmetro', o programa cria as expressões `xder` e `yder` a partir dos strings de entrada, mantendo o parâmetro escolhido como simbólico, e exibe na tela a expressão do sistema.

Listagem 12 – Expressão do sistema em função do parâmetro de bifurcação

```
flag2 = false;
if flag && ~strcmp(muvar, 'parâmetro')
    flag2 = true;
    assignParameters(pname, pval);

    vars = [sym(xvar), sym(yvar)]; %syms(vars);
    syms(xvar); syms(yvar);
    syms(muvar);

    %se entrada de função vazia, assume f(xvar, yvar) = 0
    if isempty(xder)
        xder=sym();
    end
    if isempty(yder)
        yder=sym();
    end

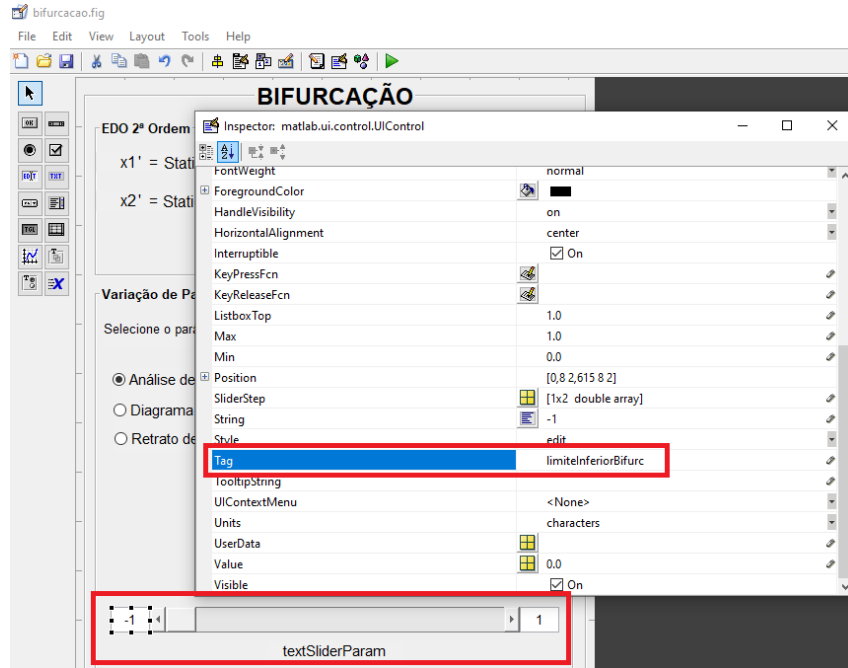
    fg = sym([eval(xder), eval(yder)]);
    set(handles.strFuncaoXBI, 'String', char(fg(1)));
    set(handles.strFuncaoYBI, 'String', char(fg(2)));
```

Fonte: Elaborado pela autora.

As funções `limiteInferiorBifurc_Callback` e `limiteSuperiorBifurc_Callback` obtêm os dados informados pelo usuário referente a faixa de valores do Slider (a barra deslizante que se move para a esquerda e direita, conforme Figura 19), ou seja, a faixa de valores do parâmetro de bifurcação. Se o valor máximo digitado pelo usuário for menor ou igual a valor mínimo, o programa soma o valor máximo a um erro para que seja possível

trabalha em uma faixa de valores. Caso o valor mínimo seja maior que o valor máximo, subtrai um erro.

Figura 19 - Limite inferior do Slider



Fonte: Ferramenta PEBICLI - Software MATLAB.

A função `sliderParamContinuousValueCallback` trata do resultado ao se movimentar o Slider, representado a faixa de valores do parâmetro de bifurcação, ou seja, é o retorno de chamada em qualquer alteração em seu valor. A propriedade `Value` informa a posição atual no Slider até que o botão esquerdo do mouse seja liberado. Na Listagem 13, o programa substitui o valor do parâmetro obtido na propriedade `Value` e encontra as soluções do sistema.

Listagem 13 – Valor da Barra de rolagem

```
handles = guidata(hFigure);
sliderValue = get(handles.sliderParam, 'Value');
set(handles.textSliderParam, ...
    'String', num2str(sliderValue));

global xvar yvar vars muvar fg MaxDegree;
global figuraBifurcacao;

global flag2;
if flag2
    assignValor(muvar, sliderValue)
    syms(xvar); syms(yvar);
    fgParam = sym(eval(fg));
    S = solve(fgParam, vars, 'MaxDegree', MaxDegree, 'Real', true);
    pEq = [S.(xvar), S.(yvar)]; %Pontos de Equilibrio
    pEq = vpa(pEq);
```

Fonte: Elaborado pela autora.

A ativação de cada um dos três Radios Buttons (Análise de Autovalores e PEs – Radio Button de tag `rb1`, Diagrama de Bifurcação – Radio Button de tag `rb2`, Retrato de Fase – Radio Button de tag `rb3`) exibe na tela os componentes intrínsecos para execução do código, como exemplo o componente do tipo Edit Text para inserção dos limites do Retrato de Fases. Isto é feito através da função `rbGroup_SelectionChangedFcn`.

A ativação do retrato de fase através do botão de opção (radiobutton de tag `rb3`) gera o retrato de fase sistemas através do comando `streamslice`, com setas de direções, conforme Listagem 14. Os pontos de equilíbrio são plotados da tela com o comando `plot`.

Listagem 14 – Retrato de Fase

```
xldot = eval(fgParam(1)); x2dot = eval(fgParam(2));
try
    figure(figuraBifurcacao); clf;
    figuraBifurcacao.Name = 'Retrato de Fase';
catch
    figuraBifurcacao = figure('Name','Retrato de Fase',...
        'NumberTitle','off', 'MenuBar', 'none',...
        'ToolBar', 'figure', 'Pointer', 'crosshair');
    movegui('west');
end
densidade = 1;
streamslice(valuex1, valuex2, xldot, x2dot, densidade);
hold on;
plot(pEq(:,1), pEq(:,2), 'ro');
title(['',...
    sprintf('%s = ', muvar), num2str/sliderValue]);
xlabel(xvar); ylabel(yvar);
try axis(limites); catch, end;
axis tight;
axis manual;
box on;
drawnow();
```

Fonte: Elaborado pela autora.

A ativação do botão de opção da Análise dos Autovalores e Pontos de Equilíbrio (Radio Button de tag `rb1`) exibe na tela os pontos de equilíbrios do sistema, sendo no máximo 5 pontos.

A função de chamado do radiobutton de tag `rb2`, denominada `PBGerarDiagrama_Callback`, foi elaborada separadamente pois não depende da variação da barra deslizante, mas sim dos limites que o usuário definir para então plotar o diagrama de bifurcação. Inicialmente, o programa soluciona o sistema na forma cartesiana para encontrar os pontos de equilíbrios. Em seguida coloca-se o sistema na forma polar para encontrar os pontos de equilíbrios que dão origem aos ciclos limites, similar a implementação do Poincaré Bendixson, sendo que agora o programa encontra não somente o raio, mas também o ângulo, conforme Listagem 15.

Listagem 15 – Solução Cartesiana e Polar

```
syms(xvar); syms(yvar);
syms(muvar);
Scart = solve(fg, vars,...
    'MaxDegree', max(MaxDegree, 1),...
    'Real', true);

%Coordenadas Polares
syms r_ a_ % a: alfa (ou theta); r: rho (ou r);
dx = eval(fg(1)); dy = eval(fg(2));
dr = (sym(xvar)*dx+sym(yvar)*dy)/r_; %dr = (x1*dx+x2*dy)/r;
da = (dy*sym(xvar)-dx*sym(yvar))/r_^2; %da = (dy*x1-dx*x2)/r^2;
assignValor(xvar, r_*cos(a_));
assignValor(yvar, r_*sin(a_));
drpol = simplify(eval(dr)); dapol = simplify(eval(da));
odespol = [drpol; dapol];
```

Fonte: Elaborado pela autora.

Na Listagem 16, cria-se um vetor denominado “mu” com os limites definidos pelo usuário na tela Bifurcação. Os pontos de equilíbrios (variáveis x_{-} e y_{-}) são calculados na forma numérica, substituindo o valor do parâmetro. Valores complexos são eliminados. Para cada parâmetro, representado por 'mu', o programa resolve o ponto de equilíbrio na forma cartesiana e plota uma linha a cada dois valores de 'mu' ('mu' anterior, 'mu' atual). Faz-se u_{-} ter o mesmo tamanho de x_{-} . Um gráfico 3D é gerado através do comando plot3.

Listagem 16 – Diagrama de Bifurcação

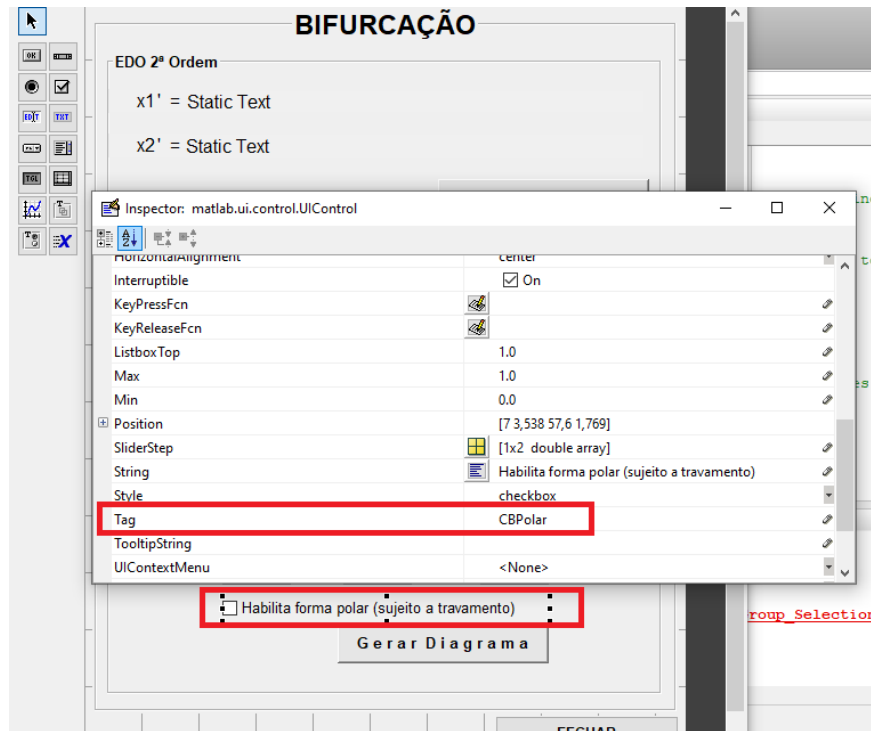
```
view(45,45);
minMu = str2double(get(handles.paramMinBIV, 'String'));
maxMu = str2double(get(handles.paramMaxBIV, 'String'));
npMu = str2double(get(handles.paramNPBIV, 'String'));
mu_ = linspace(minMu,maxMu, npMu);
for i_ = 1:length(mu_)
    syms a_
    u_ = mu_(i_);
    assignValor(muvar, u_);
    x_ = eval(Scart.(xvar)); x_(imag(x_)==0) = NaN;
    y_ = eval(Scart.(yvar)); y_(imag(y_)==0) = NaN;
    u_ = u_ * ones(size(x_));
    if i_ > 1
        for k_ = 1:size(x_,1)
            plot3([u_1(k,:),u_(k,:)], ...
                [x_1(k,:),x_(k,:)], ...
                [y_1(k,:),y_(k,:)], 'b-');
        end
        xlim([min(mu_) mu_(i_)])
        drawnow();
    end
    x_1 = x_; y_1 = y_; u_1 = u_; %Guarda os valores anteriores
```

Fonte: Elaborado pela autora.

Caso o botão de seleção da forma polar (cujo tag é CBPolar, conforme Figura 20) seja ativado pelo usuário, o programa resolve o sistema na forma polar para que sejam plotados os

círculos que representam ciclos limites, conforme Listagem 17. Essa opção de seleção tem o objetivo de evitar que alguns sistemas travem a execução do programa caso sempre esteja habilitada a forma polar.

Figura 20 - Habilitação da forma polar



Fonte: Elaborado pela autora.

Listagem 17 – Resolução do sistema na forma polar

```

if handles.CBPolar.Value
    S = solve(odespol, [r_a_], 'MaxDegree', max(MaxDegree, 1));
    if isempty(S.r_) && isempty(S.a_)
        %Resolver individualmente
        Sr = solve(drp_ol, r_, 'MaxDegree', max(MaxDegree, 1));
        Sr = simplify(Sr);
        Sa = solve(dapol, a_, 'MaxDegree', max(MaxDegree, 1));
        if isempty(Sa)
            %Sa não tem solução. Possivelmente dapol é uma
            constante. Integrar:
            Sa = int(dapol, a_);
            Sa = Sa.*ones(length(Sr),1);
            Sa = simplify(Sa);
        else
            Sa = simplify(Sa);
        end
    else
        Sr = simplify(S.r_);
        Sa = simplify(S.a_);
    end
end
end

```

Fonte: Elaborado pela autora.

A Listagem 18 mostra como é gerado o Diagrama de Bifurcação quando há ciclos limites. Atribui-se a variável `rho` à solução do sistema em função do raio e a variável `theta` à solução em função do ângulo. Substitui-se o valor do ângulo (arbitrariamente 50 pontos do plano) nas equações que representam o sistema na forma polar. Os pontos de equilíbrios são convertidos para forma cartesiana e o programa plota o gráfico 3D.

Listagem 18 – Diagrama de Bifurcação

```

try
    rho = eval(Sr);
    theta = eval(Sa);
    a_ = linspace(0, 2*pi, 50); %50 pontos para gerar uma curva
    rho = subs(rho);
    theta = subs(theta);
    theta = eval(theta);
    rho = eval(rho);

    if length(theta) > length(rho)
        rho = rho*ones(1, length(theta));
    elseif length(theta) < length(rho)
        theta = theta*ones(1, length(rho));
    end
    [x_,y_] = pol2cart(theta, rho);
    u_ = u_*ones(size(x_));
    x_(imag(x_)==0) = NaN;
    y_(imag(y_)==0) = NaN;

    for j_ = 1:size(x_,1)
        plot3(u_(j_,:), x_(j_,:), y_(j_,:), 'b-');
        drawnow();
    end
catch
end
end

```

Fonte: Elaborado pela autora.

4.3 LIMITAÇÕES COMPUTACIONAIS

- Gráficos podem demorar de carregar devido a processamento do código e/ou desempenho da máquina;
- Alguns erros minoritários foram previstos e tratados, porém outros erros não previstos na implementação podem ocorrer;
- Algumas funções podem não funcionar, dependendo da versão do MATLAB. No guia de uso do próximo capítulo informa os requisitos da ferramenta para seu correto funcionamento;
- O diagrama de bifurcação não exibe linhas pontilhadas que representam pontos instáveis, apenas linhas contínuas. Seria possível personalizar com cores ou

formatos diferentes, porém para que isso aconteça é necessário que o programa seja capaz de verificar se aquele ponto de equilíbrio é estável ou instável, verificando a característica de cada ponto de equilíbrio para cada valor do parâmetro e separando os pontos instáveis dos estáveis. Somente após isso geraria os gráficos, tornando a execução do código pesada;

- Para alguns sistemas o código pode demorar em encontrar a solução. Sendo assim, foi inserido um aviso na figura do Diagrama de Bifurcação com uma forma de interromper a execução do código (Ctrl+C);
- O programa não é capaz de gerar todos os tipos de bifurcações. Devido a complexidade de alguns sistemas a resolução computacional não é efetuada.

5 GUIA DE USO

➤ REQUISITOS

- A ferramenta PEBICLI requer que o usuário tenha o software Matlab® instalado no seu computador.

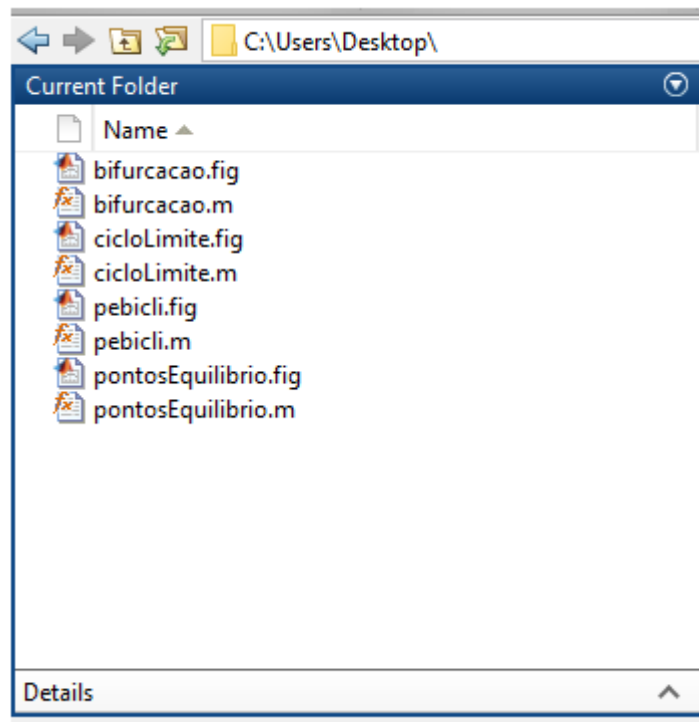
➤ ACESSO A FERRAMENTA

- <https://github.com/jessicasouzads/PEBICLI.git>

➤ INICIALIZAÇÃO DO PEBICLI

- Extraia a pasta “PEBICLI” em algum local de fácil acesso.
- Execute o Matlab® e navegue até a pasta “PEBICLI” utilizando o navegador de arquivos.

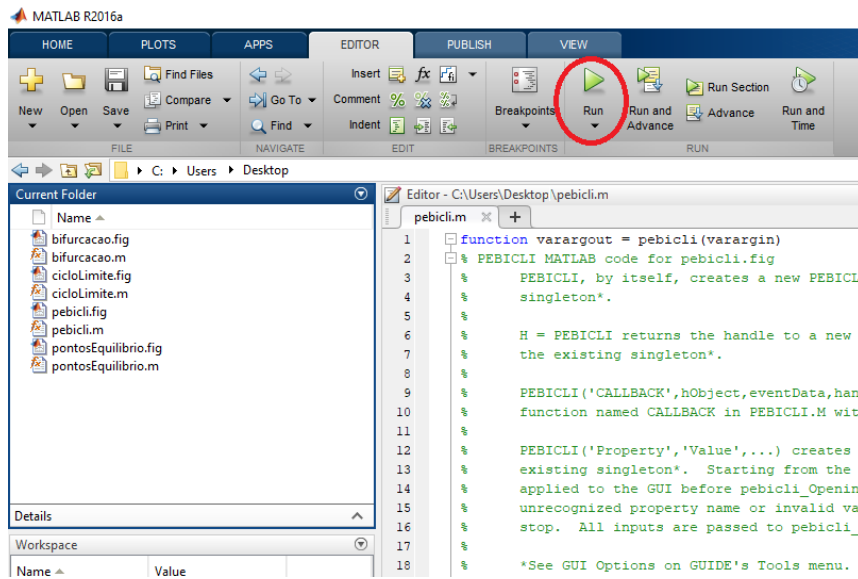
Figura 21 - Explorador de Arquivos.



Fonte: Software MATLAB.

- Abra o arquivo “pebicli.m” que se encontra na pasta.
- Execute o código através do botão Run, conforme indicado na Figura 22.

Figura 22 - Botão para executar o código no Matlab.



Fonte: Software MATLAB.

➤ MENU PRINCIPAL

- Será exibida a tela inicial que contém a ferramentas desenvolvida.
- Insira as equações que representam o sistema de segunda ordem. A ferramenta aceita quaisquer entradas representando as variáveis do sistema. Por convenção, o sistema de exemplo da Figura 23 possui x_1 e x_2 como variáveis. Caso exista, os parâmetros devem ser informados nos campos localizados na região central do Menu Principal, com seus respectivos valores.
- O sistema deve ser inserido da forma padrão que o Matlab consegue rodar. Por exemplo, função seno deve ser digitada como $\sin(\text{variável})$. Além disso, recomenda-se trabalhar com parênteses para fatorar ou organizar a equação.
- Se as letras que representam os parâmetros do sistema forem iguais as variáveis, o código pode acabar gerando erros, logo é aconselhável informar dados diferentes.

Figura 23 - Menu Principal da PEBICLI.

FERRAMENTA PEBICLI

EDO 2ª Ordem

$x_1' = x_1(u - x_1^2 - x_2^2) - x_2$

$x_2' = x_2(u - x_1^2 - x_2^2) + x_1$

Parâmetros

$u = 2$

Módulos

Pontos de Equilibrio **Bifurcação** **Ciclo Limite**

Universidade Federal da Bahia - Escola Politécnica, Dezembro de 2019

FECHAR

Fonte: Ferramenta PEBICLI - Software MATLAB.

- Clique sobre o botão da ferramenta de análise que deseja utilizar.

➤ PONTOS DE EQUILÍBRIO

- Ao se apertar o botão Ponto de equilíbrio, a tela representada na Figura 24 é exibida na tela.

Figura 24 - Módulo Pontos de Equilíbrio.

Análise de Pontos de Equilíbrio

FERRAMENTA DE PONTOS DE EQUILÍBRIO

EDO 2ª Ordem

$$x_1' = -x_2 - x_1(x_1^2 + x_2^2 - 2)$$
$$x_2' = x_1 - x_2(x_1^2 + x_2^2 - 2)$$

Atualizar

Verificar

RESULTADO: (?) Classificação por linearização nos pontos ☐ Forma decimal

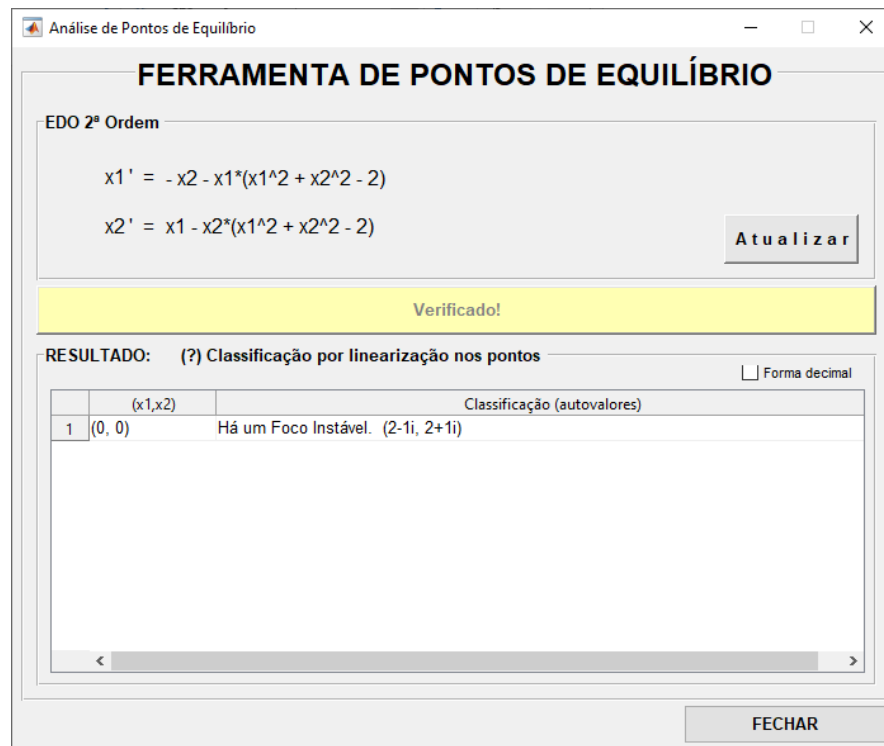
| (x1,x2) | Classificação (autovalores) |
|---------|-----------------------------|
|---------|-----------------------------|

FECHAR

Fonte: Ferramenta PEBICLI - Software MATLAB.

- Clique no botão Verificar para que os pontos de equilíbrios sejam exibidos com sua respectiva classificação.
- Segundo Khalil (2002), o sistema exemplo da Figura 25 possui o ponto de equilíbrio (0,0) classificado por Foco Instável.

Figura 25 - Classificação dos Pontos de Equilíbrio.



Fonte: Ferramenta PEBICLI - Software MATLAB.

➤ CICLO LIMITE

- Ao abrir o módulo, comece a análise pelo Corolário. A Figura 26 ilustra um exemplo no qual há apenas um ponto de equilíbrio classificado por Foco Instável. Logo, ao verificar, o programa retorna a mensagem “Pode haver Ciclo Limite!”.

Figura 26 - Módulo Ciclo Limite.

Fonte: Ferramenta PEBICLI - Software MATLAB..

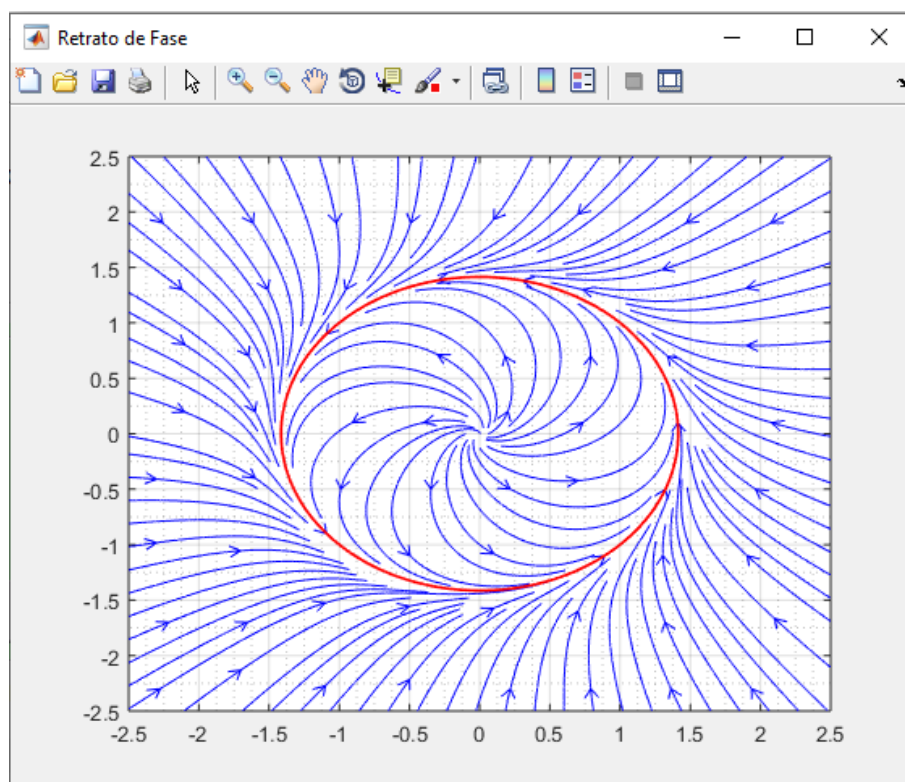
- Avance na análise verificando se o Critério de Bendixson permite a existência do Ciclo Limite no sistema.
- O sistema exemplo da Figura 27 retorna que pode haver, pois dependendo do valor que x_1 assuma, o resultado da expressão $\frac{\partial f_1}{\partial x_1} + \frac{\partial f_2}{\partial x_2}$ pode assumir valores positivos ou negativos.

Figura 27 - Verificação do Critério de Bendixson.

Fonte: Ferramenta PEBICLI - Software MATLAB..

- Antes de verificar a região de existência do ciclo através do Teorema de Poincaré Bendixson, defina os limites do gráfico. Caso a região seja do ciclo seja maior que os limites, redefina-os.
- A grade é referente a quantidade de pontos plotados dentro do limite definido.
- Para exibir as setas do campo de direção, clique na caixa Check Box.
- Clique no botão Verificar. A região de ocorrência do Ciclo Limite será exibida, como exemplo da Figura 28.

Figura 28 - Região de Ocorrência do Ciclo Limite.

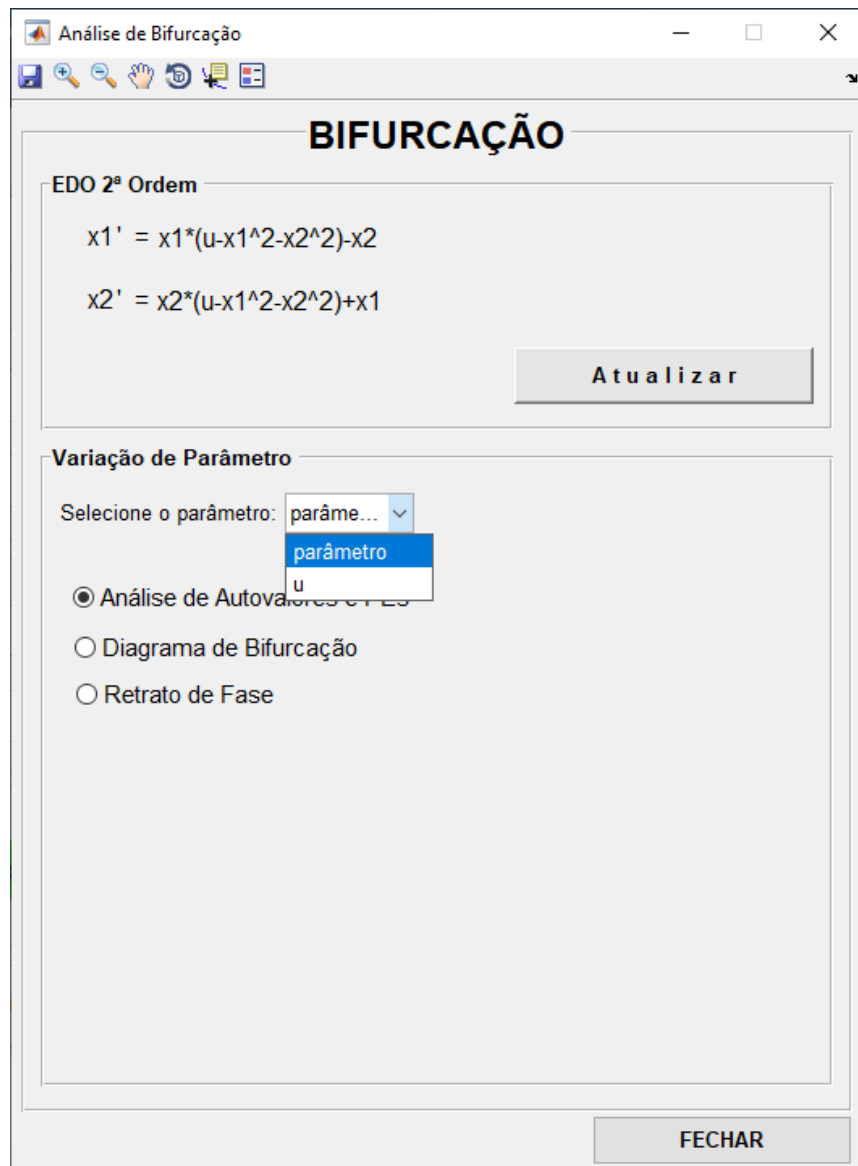


Fonte: Ferramenta PEBICLI - Software MATLAB.

➤ BIFURCAÇÃO

- Escolha o parâmetro para análise da Bifurcação, conforme exemplo da Figura 29.

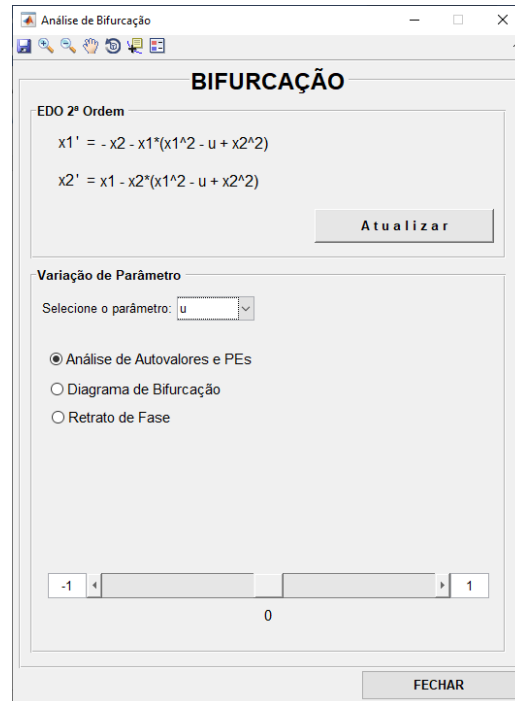
Figura 29 - Seleção do parâmetro.



Fonte: Ferramenta PEBICLI - Software MATLAB.

- Selecione o meio para verificar a ocorrência da Bifurcação: Análise de Autovalores e PEs, Diagrama de Bifurcação ou Retrato de Fases.
- Caso opte verificar o fenômeno da bifurcação através da Análise de Autovalores e Pontos de Equilíbrios, ao selecionar o parâmetro, uma barra deslizante aparecerá na tela. Defina a faixa de valores, conforme exemplo da Figura 30.

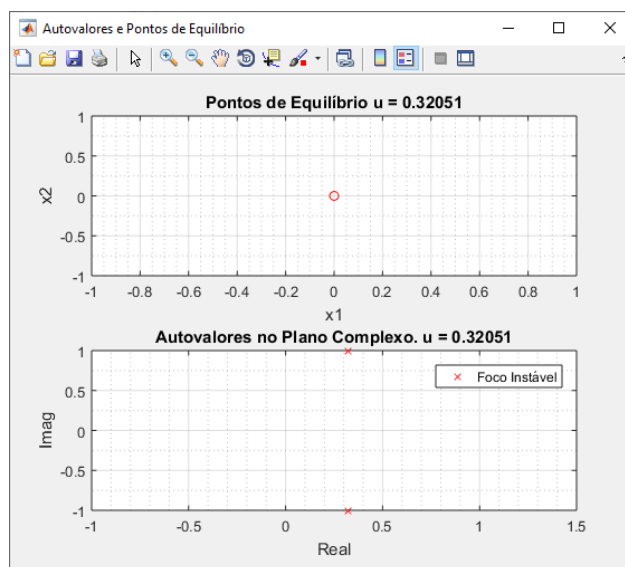
Figura 30 - Barra deslizante.



Fonte: Ferramenta PEBICLI - Software MATLAB.

- Movimento a barra deslizante para que a figura seja gerada.

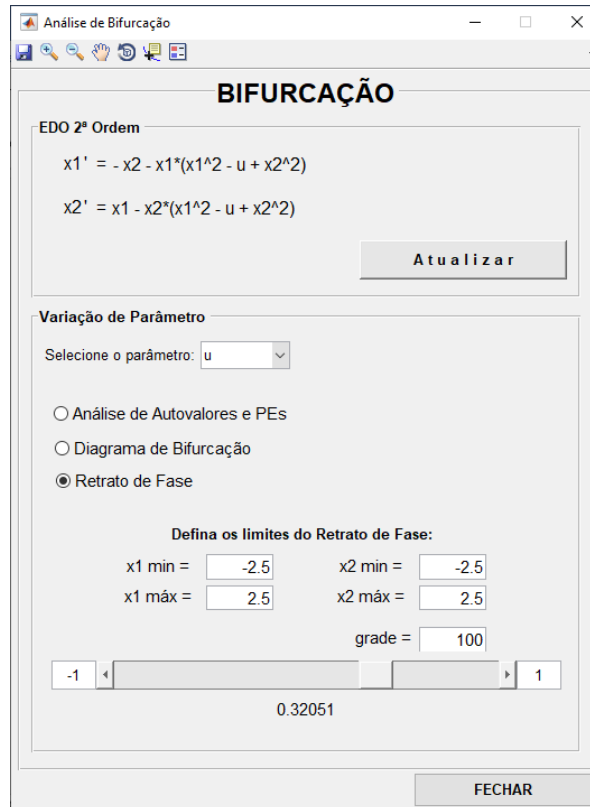
Figura 31 - Análise da Bifurcação através dos Autovalores e Pontos de Equilíbrio.



Fonte: Ferramenta PEBICLI - Software MATLAB.

- Caso deseje verificar ocorrência da Bifurcação através do Retrato de Fases, selecione a opção, defina os limites do gráfico e a quantidade de pontos (Grade).

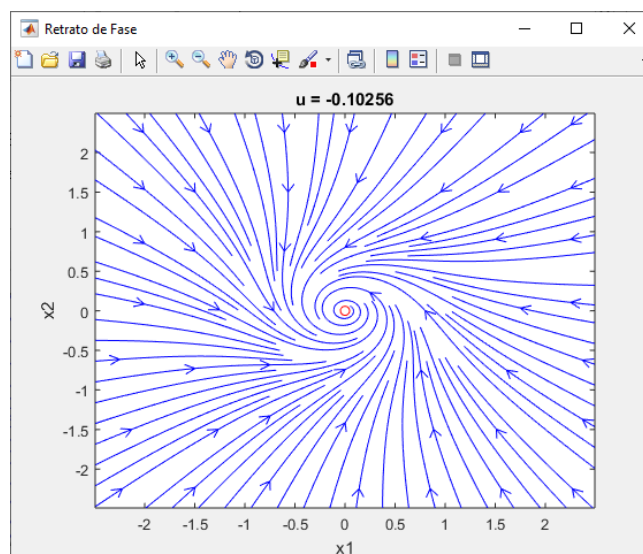
Figura 32 - Opção Retrato de Fases.



Fonte: Ferramenta PEBICLI - Software MATLAB.

- Movimento a barra deslizante.

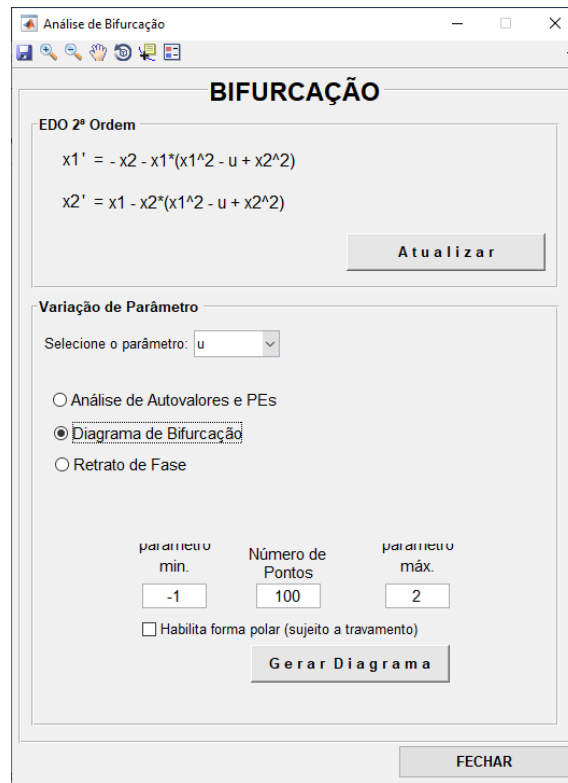
Figura 33 - Retrato de Fases.



Fonte: Ferramenta PEBICLI - Software MATLAB.

- Ao selecionar o Diagrama de Bifurcação, defina a faixa de valores do parâmetro.

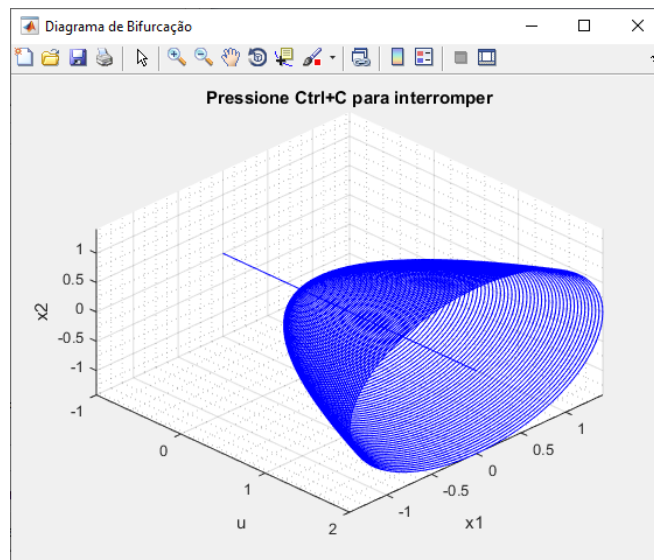
Figura 34 - Faixa de valores do parâmetro.



Fonte: Ferramenta PEBICLI - Software MATLAB.

- Aperte o botão Gerar Diagrama.
- Caso não seja possível verificar ocorrência da Bifurcação, habilite a forma polar.

Figura 35 - Diagrama de Bifurcação.



Fonte: Ferramenta PEBICLI - Software MATLAB.

6 CONCLUSÃO

Neste trabalho foi apresentada uma ferramenta computacional para a análise de sistemas não lineares, capaz de caracterizar pontos de equilíbrio, verificar ocorrências de ciclos limites e de bifurcações. O resultado obtido foi uma ferramenta interativa com usuário, que contém uma interface para a entrada das características do sistema não linear como também para a visualização dos resultados.

Os testes que exemplificaram a utilização da ferramenta, contidos no Guia de Uso, mostraram que o manuseio da PEBICLI é simples. Os resultados dos exemplos condizem com o referencial teórico ao passo que mostraram que o uso da ferramenta pode servir de apoio para a disciplina Controle e Sistemas Não Linear. Dessa forma proporcionará aos alunos uma melhor visualização dos fenômenos abordados em sala de aula, produzindo resultados significativos no processo de aprendizagem, através de elementos que tornam interação dinâmica com o usuário.

A codificação integrada à construção das telas no Guide foi considerada uma etapa desafiadora, pois foi necessário adquirir conhecimento de novas funcionalidades do MATLAB, demandando tempo na implementação da ferramenta, o que culminou uma das dificuldades encontradas neste trabalho. É importante ressaltar a utilidade de ter um programa escrito em código comentado, possibilitando ajustes em versões futuras.

Por fim, considera-se que esse trabalho tenha dado sua parcela de contribuição para o meio acadêmico, devido à carência de ferramentas computacionais no estudo de sistemas não lineares.

6.1 TRABALHOS FUTUROS

Este trabalho limitou-se a analisar sistemas não lineares de segunda ordem para dar suporte didático ao estudo de Sistemas Não Lineares. Uma vez que os objetivos foram atingidos, é recomendado que a ferramenta seja aprimorada em trabalhos futuros, destacando:

- i. Expandir a análise para sistemas de terceira ordem;
- ii. Na parte referente ao Teorema de Poincaré Bendixson, exibir o ciclo, não somente a região, com outras curvas fechadas de formatos mais complexos, como por exemplo forma de retângulo ou losango de vértices arredondados;
- iii. Gerar o Diagrama de Bifurcação para todos os tipos de bifurcação;

- iv. Incluir o módulo de estabilidade de Lyapunov.
- v. Aprimorar o código e substituir funções simbólicas por outras equivalentes que não são, para não gerar problemas de versões do matlab.
- vi. Implementar mensagens de erros no Guide, a fim de alertar ao usuário possíveis inconsistências. Como exemplo, caso o usuário escolha a mesma letra para representar algum parâmetro e utilizar essa mesma letra para um dos estados do sistema, alertar para que seja feita mudança na escolha.

REFERÊNCIAS

ALLIGOOD, Kathleen T; SAUER, Tim; YORKE, James. **Chaos: An Introduction to Dynamical Systems**. 4. ed. Springer Science & Business Media, 2006.

BELHOT, R. V.; MALAVE, C.O.; FIGUEIREDO, R. S. **Uso da Simulação no Ensino de Engenharia In: CONGRESSO BRASILEIRO DE ENSINO DE ENGENHARIA**, Porto Alegre, 2001.

CHAPMAN, S. J. **Programação em MATLAB para engenheiros**. São Paulo: Pioneira Thomson Learning, 2003.

CHEN, Guanrong; HILL, David John; YU, Xinghuo. **Bifurcation Control: Theory and Applications**. Springer Science & Business Media, 2003.

COELHO NETO, J.; IMAMURA, M. M. **Uma Abordagem dos Tipos de Ferramentas Computacionais Utilizados para Auxiliar o Processo Ensino-Aprendizagem da Matemática**. In: II Semana de Computação da UEL - II SECOMP, 2005, Londrina. II Semana de Computação. Londrina: UEL, 2005.

ENNS, Richard H.; MCGUIRE, George C. **Nonlinear Physics with Maple for Scientists and Engineers**. [S. l.]: Birkhäuser Basel, 1997.

FARINA, Luciano André; POSSER, Maurício Simões. **MATLAB Ferramenta matemática para Engenharia**. Universidade Federal do Rio Grande do Sul Escola de Engenharia Departamento de Engenharia Química, 1999. Disponível em: <http://www2.peq.coppe.ufrj.br/Pessoal/Professores/Arge/COQ897/Matlab/>. Acesso em: 14 nov. 2019.

FIOLHAIS, C; TRINDADE, J. **Física no computador: o computador como uma ferramenta no ensino e na aprendizagem das Ciências Física**. Revista Brasileira de Ensino de Física, São Paulo, v. 25, 2003.

FRANKLIN, Gene F.; POWELL, J. David; EMAMI-NAEINI, Abbas. **SISTEMAS DE CONTROLE PARA ENGENHARIA**. Bookman, 2013.

HOW to convert dynamical system to polar coordinates? [closed]. [S. l.], 12 abr. 2016. Disponível em: <https://math.stackexchange.com/questions/1739798/how-to-convert-dynamical-system-to-polar-coordinates>. Acesso em: 1 out. 2019.

JORDAN, D. W.; SMITH, P. **Nonlinear Ordinary Differential Equations**. 4. ed. Keele University: Oxford Univ. Press, 2007.

KHALIL, H. **Nonlinear Systems**. 3. ed. New Jersey: Prentice Hall, 2002.

KUZNETSOV, Y.A. **Elements of Applied Bifurcation Theory**. 2. ed. Springer Science & Business Media, 1998.

MANURUNG, Auralius. **Phase portrait for FIRST, SECOND and THIRD order ODE.**, 5 abr. 2017. Disponível em: https://ww2.mathworks.cn/matlabcentral/fileexchange/62216-phase-plane-with-gui-for-1st-and-2nd-order-ode?s_tid=prof_contriblnk. Acesso em: 12 nov. 2019.

MARTINS, Onilza Borges; MASCHIO, Elaine Cátia Falcade. Revista Electrónica “Actualidades Investigativas en Educación”. **AS TECNOLOGIAS DIGITAIS NA ESCOLA E A FORMAÇÃO DOCENTE: REPRESENTAÇÕES, APROPRIAÇÕES E PRÁTICAS**, 30 set. 2014.

MONTEIRO, L.H.A. **Sistemas dinâmicos**. 2. ed. Editora Livraria da Física, 2006. .

NETO, Wilson Castello Branco; SCHUVARTZ, Aguinaldo Antonio. **Ferramenta Computacional de Apoio ao Processo de Ensino Aprendizagem dos Fundamentos de Programação de Computadores**, Workshop em Informática na Educação (sbie) 2007 XVIII Simpósio Brasileiro de Informática na Educação - SBIE - Mackenzie, 2007.

OGATA, K. **Engenharia de Controle Moderno**. 3. ed. Ed. Minnessota: LTC, 1998.

OLIVEIRA, Profa. Vilma A.; ROSOLEN, José Ricardo. SEL364 - Controle Não Linear Aplicado. **Análise de sistemas não-lineares**, Departamento de Engenharia Elétrica USP - São Carlos, Março 2011. Disponível em: <http://www.sel.eesc.usp.br/lac/disciplinas/sels/arquivos/sel364/private/>. Acesso em: 19 set. 2019.

PERKO, L. **Differential equations and dynamical systems: Texts in Applied Mathematics**, 7. 3. ed. New York: Springer-Verlag, 1991.

POLKING, John C.; ARNOLD , David. **Equações diferenciais ordinárias usando MATLAB por David Arnold e John C. Polking**, 1 ago. 1999. Disponível em: <https://math.rice.edu/~polking/odesoft/>. Acesso em: 12 nov. 2019.

PRADO, C.; FIEDLER-FERRARA, N. **Caos - Uma Introdução**. 1. ed. Editora Blucher, 1994.

SAVI, M.A. **Dinâmica não-linear e caos**. 1. ed. Rio de Janeiro: E-papers, 2006.

SILVA, Fernando Gabriel Souza *et al.* 10 Encontro Internacional De Formação De Professores. **Elaboração De Uma Ferramenta Computacional Para O Ensino De Lógica Matemática E O Seu Impacto No Processo Ensino Aprendizagem**, IFRJ - Campus Nilópolis, 2017.

SILVA, G. V. M. S. **Controle Não Linear**. Lisboa: Escola Superior de Tecnologia Setúbal / IPS, 2006.

SLOTINE, J.-J. E.; WEIPING LI. **Applied Nonlinear Control**. New Jersey: Prentice Hall, 1991.

STROGATZ, Steven. **Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering (Studies in Nonlinearity)**. 1. ed. Canada: CRC Press, 1994.

THOMPSON , M. T.; STEWART, H. B. **Nonlinear dynamics and chaos**. 1. ed. New York: Wiley, 1986.

VON ZUBEN, Fernando José. **Nota de Aula 2 - Teoria de Sistemas Não-Lineares: Limitações básicas da linearização**. DCA/FEEC/Unicamp, 21 out. 2003. Disponível em: <ftp://vm1-dca.fee.unicamp.br/pub/docs/vonzuben/ea932/aula2.pdf>. Acesso em: 12 nov. 2019.