ARNALDO DE CARVALHO JUNIOR

**Identificação e Controle de Sistemas Dinâmicos com Rede Neural
Paraconsistente**

São Paulo
2021

ARNALDO DE CARVALHO JUNIOR

**Identificação e Controle de Sistemas Dinâmicos com Rede Neural
Paraconsistente**

Versão Corrigida

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do
Título de Doutor em Ciências.

São Paulo
2021

ARNALDO DE CARVALHO JUNIOR

**Identificação e Controle de Sistemas Dinâmicos com Rede Neural
Paraconsistente**

Versão Corrigida

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do
Título de Doutor em Ciências.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. João Francisco Justo Filho

São Paulo
2021

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 27 de agosto de 2021.

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

DE CARVALHO JUNIOR, ARNALDO

Identificação e Controle de Sistemas Dinâmicos com Rede Neural Paraconsistente / A. DE CARVALHO JUNIOR – versão corr. – São Paulo, 2021.
196 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo.
Departamento de Engenharia de Sistemas Eletrônicos.

1.Lógica Paraconsistente 2.Redes Neurais Artificiais Paraconsistentes 3.Função de Ativação LPA2v 4.Aprendizado Supervisionado 5.Identificação e Controle I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

Dedico este trabalho primeiramente aos meus pais, Arnaldo (in memoriam) e Maria Helena (in memoriam), pelo amor, exemplos de vida e por me incentivarem a aprender, sempre. A minha esposa, Silvana, por estar ao meu lado nesta jornada. Aos meus filhos, Augusto e Alvaro, por tornarem mais plena a minha vida. A todos vocês, a minha eterna gratidão!

AGRADECIMENTOS

À Deus, em primeiro lugar, por iluminar o meu caminho.

Ao Professor Dr. João Francisco Justo Filho, por sua sapiência, compreensão e orientação para a realização deste trabalho.

Ao Professor Dr. Bruno Augusto Angélico, pelo suporte laboratorial, ensinamentos em controle de sistemas e coorientação para a realização deste trabalho.

Ao Professor Dr. João Inácio da Silva Filho, minha eterna gratidão por suas sábias orientações em Lógica Paraconsistente.

Ao Professor Dr. Alexandre Maniçoba de Oliveira, antes meu aluno e agora colega de trabalho e mentor, pelo incentivo para meu ingresso no doutoramento.

Ao Professor Dr. Charles Artur Santos de Oliveira, por suas palavras de motivação e exemplo acadêmico profissional.

Aos professores da banca de avaliação por suas valorosas contribuições.

Agradeço aos meus familiares e amigos que sempre me apoiaram para a conclusão deste trabalho.

Aos professores do programa de pós-graduação em Engenharia Elétrica da Escola Politécnica (Universidade de São Paulo) pelos valiosos ensinamentos.

Aos meus colegas do Laboratório de Controle Aplicado pelo suporte durante as simulações e ensaios.

Aos servidores da secretaria do departamento de Pós-Graduação em Engenharia Elétrica, pela paciência e atendimento prestado.

Aos colegas do IFSP campus Cubatão pelo apoio e flexibilização de horários de trabalho, que possibilitaram o desenvolvimento deste trabalho.

RESUMO

DE CARVALHO JUNIOR, A. **Identificação e Controle de Sistemas Dinâmicos com Rede Neural Paraconsistente**. 2021. 196 p. Tese (Doutorado) – Programa de Engenharia Elétrica, Escola Politécnica, Universidade de São Paulo, São Paulo, 2021.

Redes neurais artificiais (RNAs) têm sido amplamente empregadas nas últimas décadas em diversas áreas do conhecimento humano para classificação, previsão e reconhecimento de padrões. As RNAs são investigadas principalmente na aproximação de funções em modelos numéricos, devido às suas propriedades de autoaprendizado, adaptabilidade, tolerância a falhas, não linearidade e antecipação do mapeamento entrada-saída. O estado da arte na identificação e controle de sistemas dinâmicos emprega métodos de computação suave, decisões inteligentes baseadas em ambientes de incerteza, inteligência artificial e cognitiva, e técnicas de aprendizagem com redes neurais. Entretanto, problemas envolvendo situações contraditórias podem ser um desafio para as RNAs. A lógica paraconsistente (LP) é uma ferramenta poderosa para lidar com incertezas e informações contraditórias, representando uma alternativa potencialmente robusta à RNA clássica. A lógica paraconsistente anotada evidencial $\varepsilon\tau$, ou de lógica paraconsistente anotada com anotação de dois valores (LPA2v), uma variante da LP, tem sido empregada em sistemas especialistas que consideram informações incertas na tomada de decisão. Redes neurais artificiais paraconsistentes (RNAP), baseadas em LPA2v, são aplicadas principalmente na classificação de padrões, porém utilizando limiares e fatores de tolerância ao invés de pesos, sendo ainda um tema não explorado na identificação e controle de sistemas dinâmicos. O objetivo desta tese é desenvolver uma nova abordagem de rede neural paraconsistente (RNP), com função de ativação LPA2v treinável, como contribuição original ao estado da arte na identificação e controle de sistemas dinâmicos. Um pêndulo invertido rotativo (PIR) é utilizado para a validação dos resultados. Esta tese apresenta ainda o arcabouço matemático para análise e projeto de redes neurais paraconsistentes de aprendizado supervisionado.

Palavras-chave: Lógica Paraconsistente, Rede Neural Artificial Paraconsistente, Função de Ativação LPA2v, Aprendizado Supervisionado, Identificação e Controle.

ABSTRACT

DE CARVALHO JUNIOR, A. **Identification and Control of Dynamic Systems with Paraconsistent Neural Network**. 2021. 196 p. Tese (Doutorado) – Programa de Engenharia Elétrica, Escola Politécnica, Universidade de São Paulo, São Paulo, 2021.

Artificial neural networks (ANNs) have been widely used in the last decades in several areas of human knowledge for classification, prediction, and pattern recognition. ANNs are investigated mainly in the approximation of functions in numerical models, due to their properties of self-learning, adaptability, fault tolerance, non-linearity, and anticipation of input-output mapping. The state of the art in the identification and control of dynamic systems employs soft computing methods, intelligent decisions based on uncertainty environments, artificial and cognitive intelligence, and neural network learning techniques. However, problems involving contradictory situations can be a challenge for ANNs. Paraconsistent logic (PL) is a powerful tool for dealing with uncertainties and contradictory information, representing a potentially robust alternative to classical ANN. The paraconsistent annotated evidential Logic $\varepsilon\tau$, or paraconsistent annotated logic with annotation of two values (PAL2v), a variant of PL, has been used in expert systems that consider uncertain information in decision making. Paraconsistent artificial neural networks (PANN), based on PAL2v, are mainly applied in pattern classification, but using thresholds and tolerance factors instead of weights, being still an unexplored topic in the identification and control of dynamical systems. The objective of this thesis is to propose a novel paraconsistent neural network (PNN) approach, with a trainable non-linear PAL2v activation function, as an original contribution to the state of the art in the identification and control of dynamical systems. A rotary inverted pendulum (RIP) system is used for validation of results. This thesis also describes the mathematical framework for the analysis and design of paraconsistent neural networks of supervised learning.

Keywords: Paraconsistent Logic, Paraconsistent Artificial Neural Network, PAL2v Activation Function, Supervised Learning, Identification and Control.

LISTA DE ILUSTRAÇÕES

Figura 1 – Sistema dinâmico com variável manipulada de entrada (u), saída (y), perturbação medida (w) e não medida (v).	41
Figura 2 – Evolução dos métodos de engenharia de controle.....	43
Figura 3 – Controle LQG para um sistema dinâmico não linear.....	48
Figura 4 – Controle SMC para o RIP.....	48
Figura 5 – Controle IMC para um sistema dinâmico.....	50
Figura 6 – ANN utilizada na aproximação de função.	51
Figura 7 – Representação de um neurônio biológico.	51
Figura 8 – Representação de neurônio artificial de múltiplas entradas.....	52
Figura 9 – RNA de 3 camadas.	55
Figura 10 – Aprendizado supervisionado do controle (a). RNA em controle adaptativo (b).	57
Figura 11 – Identificação do modelo invertido do sistema pela RNA (a). Controle invertido direto por RNA (b).	57
Figura 12 – Identificação do modelo do pêndulo invertido pela RNA.	58
Figura 13 – RNA como controle IMC.....	59
Figura 14 – Estrutura MRC.	60
Figura 15 – Lattice de 4 vértices da LPA(a); Lattice finito de Hasse com anotações LPA2v (b).	66
Figura 16 – Transformação QUPC (a). Lattice da LPA2v com valores (b).	68
Figura 17 – Limites de D	69
Figura 18 – Símbolo do NAP.	71
Figura 19 – Símbolos da CNAPp.	72
Figura 20 – CNAPap.	75
Figura 21 – Saída μ_E para diferentes valores de F_L	76
Figura 22 – CNAPap para diferentes valores de μ e F_L , equação (34) (a). CNAPap para diferentes valores de μ e F_L , equação (35) (b).	77
Figura 23 – Saída μ_E da CNAPap _{CTX} para diferentes valores de μ e F_L	78
Figura 24 – Saída μ_{ECT} da CNAPap _{CTX} para diferentes valores de μ e F_L	78
Figura 25 – Neurônio Paraconsistente (Paraperceptron) de 2 entradas.....	80
Figura 26 – Neurônio Paraconsistente (Paraperceptron) de 8 entradas.....	81
Figura 27 – Exemplo de RNAP.	82

Figura 28 – Saídas μ_E e μ_{ER} para μ variável e λ com 3 valores fixos (a). Saídas μ_E e μ_{ER} para λ variável e μ com 3 valores fixos (b).	86
Figura 29 – Saídas μ_E e μ_{ER} normalizadas entre [0,1], para μ variável e λ fixo.	86
Figura 30 – Neurônio LPA2v proposto de 2 entradas.	87
Figura 31 – Neurônio padrão (a) e Neurônio LPA2v com 4 entradas (b).	88
Figura 32 – Comportamento de $\hat{\mu}_{ER}/\hat{\mu}$ pela média das equações (49) e (51) ou nulo na condição de $D \geq 1$.	93
Figura 33 – Método <i>backpropagation</i> para o neurônio paraconsistente.	95
Figura 34 – Tabela Verdade XOR (a), RNA (b) e RNP (c).	102
Figura 35 – Base de dados simplefit do Matlab.	103
Figura 36 – Um Neurônio LPA2v de 4 entradas (a). RNP com 2 neurônios de 4 entradas na camada oculta (b). RNA gerada pelo Matlab com n neurônios na camada oculta (c).	105
Figura 37 – Protótipo RIP.	106
Figura 38 – Diagrama de Blocos sistema RIP-Controlador.	109
Figura 39 – RNP para identificação do sistema dinâmico RIP.	110
Figura 40 – Estimador Paraconsistente Estático do RIP em Simulink.	111
Figura 41 – Estimador paraconsistente dinâmico.	112
Figura 42 – Aprendizado RNP e RNA. Entradas [0,0] e Saída [0] (a) e MSE(b).	116
Figura 43 – Aprendizado RNP e RNA. Entradas [1,0] e Saída [1] (a) e MSE(b).	117
Figura 44 – Aprendizado RNP e RNA. Entradas [1,1] e Saída [0] (a) e MSE(b).	118
Figura 45 – Aprendizado RNP e RNA. Entradas [0.92;0.14] e Saída [0.27] (a) e MSE(b).	119
Figura 46 – Função Simplefit: entrada (verde), saída alvo (preto), resultado de 1 neurônio paraconsistente (vermelho) e de 1 RNP com 2 neurônios na camada oculta (azul) (a) e resultado nftool para RNA de 4 neurônios na camada oculta (b).	121
Figura 47 – Diagrama da planta, controle e entrada-saída das redes neurais.	123
Figura 48 – Controlador ajustado. Velocidade do Braço e Pêndulo (a), Identificação do Modelo do Braço pela RNP1 (b), Identificação do Modelo do Pêndulo pela RNP2 (c) e identificação do modelo invertido (controlador) pela RNP3 (d).	126
Figura 49 – Controlador com desempenho inferior e modelos levantados por RNPs com controlador de melhor desempenho. Sinais da velocidade do braço e pêndulo (a),	

resposta do modelo do braço pela RNP1 (b), resposta do modelo do pêndulo pela RNP2 (c) e resposta do modelo invertido (controlador) pela RNP3 (d).....	128
Figura 50 – Modelo do RIP, Controle LQR com filtro LPF e derivador.....	129
Figura 51 – Modelo do RIP, Controle LQR e estimador paraconsistente dinâmico.	130
Figura 52 – Simulação com baixa variância de ruído. Referência (r) (preto), ângulo do braço por LPF (azul) e por estimador LPA2v (vermelho) (a). Ângulo do pêndulo por LPF (azul) e por estimador LPA2v (vermelho) (b). Velocidade do braço derivado e filtrado (azul) e pelo estimador LPA2v (vermelho) (c). Velocidade do pêndulo derivado e filtrado (azul) e pelo estimador LPA2v (vermelho) (d). FFT da saída do controlador com sinais do RIP derivados e filtrados por LPF e com estimador LPA2v (e).	132
Figura 53 – Simulação com elevada variância de ruído. Ângulo do braço filtrado (azul) e pelo estimador paraconsistente (vermelho) (a). Ângulo do pêndulo filtrado (azul) e pelo estimador paraconsistente (vermelho) (b). Velocidade do braço derivado e filtrado (azul) e pelo estimador paraconsistente (vermelho) (c). Velocidade do pêndulo derivado e filtrado (azul) e pelo estimador paraconsistente (vermelho) (d). Gráfico da FFT da saída do controlador com sinais de saída do RIP derivados e filtrados e com estimador paraconsistente.	135
Figura 54 – Controle do RIP por Rede Neural Paraconsistente Recorrente.	137
Figura 55 – Rede Neural Paraconsistente Recorrente (RNPR).	139
Figura 56 – Sinal pseudoaleatório de referência (r) para a identificação do braço e pêndulo do RIP.	140
Figura 57 – Ângulo do braço (θ_0). Ângulo do pêndulo (θ_1). Velocidade do braço ($\dot{\theta}_0$). Velocidade do pêndulo ($\dot{\theta}_1$). Saída do controlador PP (u).	142
Figura 58 – Evolução do MSE de treinamento para RNR com funções de ativação tanh e LReLU e RNPR1 com função de ativação LPA2v.	145
Figura 59 – Sinais adicionais para treinamento da RNPR3, e_B (a) e e_P (b).....	147
Figura 60 – Aprendizado RNPR3. Curva de MSE (a) e comparativo de amostras de validação entre o sinal u e a saída da RNPR3.	148
Figura 61 – Comparação controle PP e CRNPR. Ângulo do braço do RIP (θ_0). Ângulo do pêndulo do RIP (θ_1). Saída do controlador (u).....	150
Figura 62 –Gráfico da FFT da saída do controlador PP e CRNPR com LPA2v.	151
Figura 63 – Perturbações aplicada ao pêndulo do RIP com controlador PP. Ângulo do braço (a), ângulo do pêndulo (b) e sinal de controle (c).....	153

Figura 64 – Perturbações aplicada ao pêndulo do RIP com controlador CRNPR. Ângulo do braço (a), ângulo do pêndulo (b) e sinal de controle (c).	154
Figura 65 – CNAPap _{CTX} (PANCL _{CTX}) em Simulink (a) e estrutura interna (b).....	177
Figura 66 – CNAPp (PANCL), saídas lineares em Simulink (a) e estrutura interna simplificada (b).	177
Figura 67 – Estrutura interna completa da CNAPp, todas as saídas em Simulink..	178

LISTA DE TABELAS

Tabela 1 – Atraso ou iterações da CNAPap _{CTX} em função de F_L	96
Tabela 2 – Parâmetros do RIP.	107
Tabela 3 – Pesos e MSE para aproximação por 1 neurônio LPA2v e RNP com Dois neurônios.	120
Tabela 4 – Comparação entre RNP e diferentes RNA para aproximação função simplefit.	122
Tabela 5 – Comparação entre RNP e diferentes RNA para identificação modelo sistema RIP-controle ótimo.	124
Tabela 6 – Pesos aprendidos pelas RNP1, RNP2 e RNP3.....	124
Tabela 7 – MSE para número variado de neurônios na camada oculta, na RNPR1 e RNPR2..	143
Tabela 8 – Comparação entre RNPR1 e RNR com funções tanh e LReLU.....	144
Tabela 9 – Pesos aprendidos pela RNPR1.....	145
Tabela 10 – Pesos aprendidos pela RNPR2.	145
Tabela 11 – MSE para número variado de neurônios na camada oculta na RNPR3..	148
Tabela 12 – Pesos aprendidos pela RNPR3.	149

LISTA DE ABREVIATURAS E SIGLAS EM PORTUGUÊS

AR	Autorregressivo
CMR	Controle por Modelo de Referencia
CNAP	Célula Neural Artificial Paraconsistente
CNAPp	Célula Neural Artificial Paraconsistente padrão
CNAPap	Célula Neural Artificial Paraconsistente de Aprendizagem
CNAPap _{CTX}	Célula Neural Artificial Paraconsistente de Aprendizagem por extração da contradição
CRNPR	Controle por Rede Neural Paraconsistente Recorrente
PI	Proporcional-Integral
PID	Proporcional-Integral-Derivativo
FPB	Filtro Passa Baixas Frequências
LP	Lógica Paraconsistente
LPA	Lógica Paraconsistente Anotada
LPAE	Lógica Paraconsistente Anotada Evidencial
LPA2v	Lógica Paraconsistente Anotada com Anotação de 2 valores
MCP	Modelo de Controle Preditivo
MIC	Modelo Interno de Controle
NAP	Nó de Análise Paraconsistente
PID	Proporcional-Integral-Derivativo
PIR	Pêndulo invertido rotativo
QUPC	Quadrado Unitário no Plano Cartesiano
RAP	Rede de Análise Paraconsistente
RNA	Rede Neural Artificial
RNAP	Rede Neural Artificial Paraconsistente
RNP	Rede Neural Paraconsistente
RNR	Rede Neural Recorrente
RNPR	Rede Neural Paraconsistente Recorrente
SNAP	Sistema Neural Artificial Paraconsistente
TRF	Transformada Rápida de Fourier

LIST OF ABBREVIATIONS AND ACRONYMS IN ENGLISH

ANN	Artificial Neural Network
AR	Autoregressive
DC	Direct Current
DoF	Degrees of Freedom
FFT	Fast Fourier Transform
FSF	Full State Feedback
GA	Genetic Algorithm
I4C	Identification for control
IMC	Internal Model Control
LPF	Low Pass Filter
LQR	Linear Quadratic Regulator
LQE	Linear Quadratic Estimator
LQG	Linear Quadratic Gaussian
LTR	Linear Transfer Recovery
MISO	Multiple-input single output
MIMO	Multiple-input multiple-output
ML	Machine Learning
MLP	Multilayer Perceptron
MPC	Model Predictive Control
MRC	Model Reference Control
MRAC	Model Reference Adaptive Control
MSE	Mean Square Error
NARX	Nonlinear Autoregressive Exogenous Model
NFC	Neuro-Fuzzy Controller
NFN	Neuro-Fuzzy Network
PAL	Paraconsistent Annotated Logic
PAL2v	Paraconsistent Annotated Logic with annotation of two values
PAN	Paraconsistent Artificial Node
PANet	Paraconsistent Analysis Network
PANC	Paraconsistent Artificial Neural Cell
PANCS	Paraconsistent Artificial Neural Cell Standard

PANCL	Paraconsistent Artificial Neural Cell of Learning
PANCL _{CTX}	Paraconsistent Artificial Neural Cell of Learning by Contradiction Extraction
PANet	Paraconsistent Analysis Network
PANN	Paraconsistent Artificial Neural Network
PANS	Paraconsistent Artificial Neural System
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PL	Paraconsistent Logic
PNN	Paraconsistent Neural Network
PP	Pole Placement
PWM	Pulse Width Modulation
RIP	Rotary Inverted Pendulum
RMS	Root Mean Square Error
RNN	Recurrent Neural Network
RPNN	Recurrent Paraconsistent Neural Network
RPNNC	Recurrent Paraconsistent Neural Network Control
SCM	Soft-computing Methods
SDoF	Single-degree-of-freedom
SIMO	Single-Input Multiple-Output
SISO	Single-Input Single-Output
SMC	Sliding Mode Control
TLU	Threshold Logic Unit
ZOH	Zero-order Hold

LISTA DE SÍMBOLOS DO SISTEMA E CONTROLE

A	Matriz do sistema
B	Matriz de entrada
C	Matriz de saída
D	Matriz de avanço (<i>feed forward</i>)
d	Distância entre o eixo da base para o centro de massa do braço
$e, e(t)$	Sinal de Erro
e_B	Sinal de erro entre ângulo do braço (θ_0) do RIP e ângulo do braço estimado ($\hat{\theta}_0$)
e_P	Sinal de erro entre ângulo do pêndulo (θ_1) do RIP e ângulo do pêndulo estimado ($\hat{\theta}_1$)
g	Aceleração da gravidade
F	Frequência em Hertz (Hz)
I_0	Inércia do braço
I_1	Inércia do pêndulo
J	Função quadrática de custo
K_T	Constante de torque do motor, incluindo a redução
K_E	Constante da equação do motor, incluindo a redução
L	Equação Lagrangiana
L_0	Comprimento da metade do braço
L_1	Comprimento da metade do pêndulo
m_0	Massa do braço
m_1	Massa do pêndulo
p_{CL}	Polos de malha fechada (<i>closed loop poles</i>)
r	Distância do eixo base para o pêndulo
$ref, r(t)$	Sinal de referência ou trajetória
R_M	Resistência da armadura do motor DC
t	Tempo
T	Energia Cinética
T_S	Tempo de Amostragem
θ_0	Ângulo do braço (positivo para sentido anti-horário)
θ_1	Ângulo do pêndulo (positivo para sentido horário)

$\hat{\theta}_0$	Ângulo do braço estimado (positivo para sentido anti-horário)
$\hat{\theta}_1$	Ângulo do pêndulo estimado (positivo para sentido horário)
$\dot{\theta}_0$	Velocidade do braço
$\dot{\theta}_1$	Velocidade do pêndulo
$u, u(t)$	Vetor de saída do controlador
$\hat{u}, \hat{u}(t)$	Vetor de saída estimada do controlador
V	Energia Potencial
$x, x(t)$	Vetor de estado
$y, y(t)$	Vetor de saída da planta

LISTA DE SÍMBOLOS LPA2v

D_C	Certainty degree (Grau de Certeza - G_C)
D_{CT}	Contradiction degree (Grau de Contradição - G_{CT})
D_{CR}	Real certainty degree (Grau de Certeza Real - G_{CR})
F	Falso
Ft_C	Certainty tolerance factor (fator de tolerância à certeza)
Ft_{CT}	Contradiction tolerance factor (fator de tolerância à contradição)
Ft_D	Decision tolerance factor (fator de tolerância à decisão)
F_L	Learning factor (fator de aprendizagem)
$\sim$	Operador de negação
μ	Favorable evidence degree (grau de evidência favorável)
μ_{ECT}	Resulting contradiction degree (grau de contradição resultante)
μ_E	Resulting evidence degree (grau de evidência resultante)
μ_{ER}	Resulting real evidence degree (grau de evidência resultante real)
λ	Unfavorable evidence degree (grau de evidência desfavorável)
$\perp$	Paracompleto ou Indeterminado
P	Proposição Inicial
$P_{\top}$	Conotação de inconsistente à proposição P
$P_{\vee}$	Conotação de verdade à proposição P
$P_{\neg}$	Conotação de falsidade à proposição P
$P_{\perp}$	Conotação de indeterminação à proposição P
P_{μ}	Sentença proposicional
φ	Intervalo de certeza
$\varphi(\pm)$	Intervalo de Certeza sinalizado
φ_E	Intervalo De Evidência Resultante
$\varphi_E(\pm)$	Intervalo De Evidência Resultante Sinalizado
$\mathbb{R}$	Conjunto dos números reais
S_1, S_2, S_3	Saídas da PANC
τ	Reticulado finito
T	Inconsistente
V	Verdadeiro
Vl_V	Valor limite verdade

Vl_F	Valor limite falsidade
Vl_{ICC}	Valor inferior controle certeza
Vl_{SCC}	Valor superior controle certeza
Vl_{ICCT}	Valor inferior controle contradição
Vl_{SCCT}	Valor superior controle contradição

SUMÁRIO

1. INTRODUÇÃO	33
1.1 OBJETIVOS	36
1.2 JUSTIFICATIVA	37
1.3 CONTRIBUIÇÕES DO TRABALHO PROPOSTO	38
1.4 ORGANIZAÇÃO DA TESE	39
2. ESTRATÉGIAS DE IDENTIFICAÇÃO E CONTROLE DE SISTEMAS DINÂMICOS	41
2.1 IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS	41
2.2 ESTRATÉGIAS DE CONTROLE DE SISTEMAS DINÂMICOS	43
2.2.1 Controlador Linear por Alocação de Polos	44
2.2.2 Controlador Linear Ótimo	45
2.2.3 Controlador Linear e Estimador Ótimo	46
2.2.4 Controlador Robusto Não Linear	48
2.2.5 Controle de Modelo Interno	49
2.3 REDES NEURAIS APLICADAS À IDENTIFICAÇÃO E CONTROLE	50
2.3.1 Identificação e Controle de Sistemas com Redes Neurais	56
2.4 REVISÃO DA LITERATURA SOBRE APLICAÇÕES DA LPA2v EM IDENTIFICAÇÃO E CONTROLE	61
2.5 SUMÁRIO DO CAPÍTULO	62
3. LÓGICA PARACONSISTENTE APLICADA À REDE NEURAL	65
3.1 EVOLUÇÃO DA LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL $\varepsilon\tau$	65
3.2 DESCRIÇÃO DA LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL $\varepsilon\tau$	66
3.3 CÉLULAS E REDES NEURAIS ARTIFICIAIS PARACONSISTENTES	70
3.4 Neurônio e Rede Neural Paraconsistente	79
3.5 SUMÁRIO DO CAPÍTULO	83
4. SISTEMA DE ESTUDO	85

4.1. NEURÔNIO PARACONSISTENTE PROPOSTO.....	85
4.2. CÁLCULO DE DERIVADAS PARA O NEURÔNIO LPA2v PROPOSTO	90
4.3 APRENDIZAGEM DO NEURÔNIO LPA2v PROPOSTO	94
4.4 REDE NEURAL PARACONSISTENTE COM CNAPap _{CTX}	95
4.5 CNAPap _{CTX} COMO INTEGRADOR, DERIVADOR E FILTRO.....	96
4.6 SUMÁRIO DO CAPÍTULO	100
5. METODOLOGIA.....	101
5.1. RNP PARA APROXIMAÇÃO DE VALOR – ANÁLISE DO PROBLEMA EX-OR	101
5.2. RNP PARA APROXIMAÇÃO DE FUNÇÃO ENTRADA-SAÍDA	103
5.3. IDENTIFICAÇÃO DE SISTEMA DINÂMICO COM RIP	106
5.4 ESTIMADOR PARACONSISTENTE	110
5.4.1 Estimador Paraconsistente Estático.....	111
5.4.2 Estimador Paraconsistente Dinâmico	112
5.5 SUMÁRIO DO CAPÍTULO	113
6. RESULTADOS PRELIMINARES	115
6.1 RESULTADOS DA ANÁLISE DO PROBLEMA EX-OR	115
6.2. COMPARAÇÃO ENTRE RNP E RNA NA APROXIMAÇÃO DE FUNÇÃO ENTRADA-SAÍDA.....	120
6.3. RESULTADOS DA IDENTIFICAÇÃO DE SISTEMA DINÂMICO COM RIP.....	122
6.4. RESULTADOS DO ESTIMADOR PARACONSISTENTE	129
6.5. SUMÁRIO DO CAPÍTULO	136
7. CONTROLE DE RIP POR RNPR: RESULTADOS E DISCUSSÃO.....	137
7.1. ARQUITETURA MRC COM RNPR	137
7.2. IDENTIFICAÇÃO DOS MODELOS DOS ÂNGULOS DO BRAÇO E PÊNDULO DO RIP.....	139
7.3. IDENTIFICAÇÃO E MODELAGEM DO CONTROLADOR POR REDE NEURAL PARACONSISTENTE RECORRENTE.....	146

7.4. ANÁLISE DA PERFORMANCE DO CONTROLE CRNPR.....	151
7.5. SUMÁRIO DO CAPÍTULO	155
8. CONCLUSÃO	157
8.1. TRABALHOS FUTUROS	158
REFERÊNCIAS.....	159
APÊNDICE A – TRABALHOS PUBLICADOS	167
APÊNDICE B – CÓDIGO DA RNP E RNA PARA PROBLEMA EX-OR	169
APÊNDICE C – CÉLULAS NEURAIS ARTIFICIAIS PARA CONSISTENTES EM SIMULINK.....	177
APÊNDICE D – ALGORITMO DE APRENDIZADO DO CONTROLADOR POR REDE NEURAL PARA CONSISTENTE RECORRENTE (RNPR3)	179
APÊNDICE E – CRNPR-MRC NO CONTROLE DO PROTÓTIPO RIP.....	183
ANEXO A – DESCRIÇÃO DO MODELAMENTO DO RIP.....	191
ANEXO B – CÓDIGO MODELO RIP EM MATLAB DESENVOLVIDO PELO LAC – POLI/USP	195

1. INTRODUÇÃO

A modelagem matemática é a área do conhecimento que procura descrever matematicamente um fenômeno de modo a poder simular sistemas reais e assim prever o seu comportamento, sendo muito empregada em diversas áreas da engenharia, como automação e robótica.

Um sistema é dito dinâmico se a sua saída, em qualquer instante de tempo, depende de seu histórico e não apenas da entrada atual. Exemplos de sistemas dinâmicos incluem o pêndulo de um relógio, a população de uma determinada espécie (em relação a variação da quantidade ao longo do tempo), o preço de um produto, entre outros. Modelos de sistemas dinâmicos podem ser construídos a partir de dados observados (KEESMAN, 2011). Duas abordagens podem ser utilizadas para descrever um sistema. A primeira consiste na descrição matemática dos fenômenos envolvidos, chamada de *modelagem fenomenológica* ou *caixa branca*. Outra abordagem é a *identificação de sistemas*, ou modelagem empírica, onde ferramentas matemáticas são utilizadas para construir modelos dinâmicos, apenas utilizando os dados medidos de entrada e saída. Como este método não dispõe de quase ou nenhuma informação sobre o sistema, ele também é conhecido como *identificação caixa preta* (GARG; TAI; PANDA, 2017).

Para que um sistema dinâmico apresente em sua saída um determinado comportamento é necessário o desenvolvimento de um controlador. A função deste controlador é manipular as entradas do sistema de modo a obter a saída desejada. Para a aplicação da moderna teoria de controle, um modelo do sistema dinâmico é necessário, seja por sua descrição matemática ou identificação (KOZÁK, 2014).

Várias estratégias de controle têm sido estudadas com sistemas dinâmicos (AGUILAR-AVELAR; MORENO-VALENZUELA, 2016; HAMZA *et al.*, 2019; PARK; KIM; LEE, 2011). O método de controle por alocação de polos (*pole placement – PP*), também chamado de retroalimentação de estado completo (*full state feedback – FSF*), é muito utilizado pois permite uma relação clara entre os parâmetros ajustados e o comportamento do controlador resultante (HALICIOĞLU; ÖKSÜZ, 2018; SARKAR; DEWAN, 2017a; SHEHU *et al.*, 2016). Outra técnica poderosa é chamada de regulador quadrático linear (*linear quadratic regulator – LQR*), que busca valores ótimos de controlador para uma função de custo mínimo (KUMAR; JEROME, 2013) e

que tem sido aplicado com sucesso ao pêndulo invertido (SUBRAMANIAN; ELUMALAI, 2016; HAMZA *et al.*, 2019).

Na moderna teoria de controle, o controle robusto é uma abordagem de projeto do controlador que trabalha explicitamente com a incerteza, funcionando satisfatoriamente, mesmo se o sistema dinâmico apresentasse diferenças significativas em relação ao modelo matemático de referência (AGUIRRE *et al.*, 2007). Quando o ruído gaussiano está presente ou algumas variáveis de estado não são bem conhecidas, a combinação de um estimador como o filtro Kalman e o LQR podem ser aplicados em conjunto, sendo denominado controlador gaussiano quadrático linear (*linear quadratic gaussian – LQG*) (ZHENG; ZHONG; YUE, 2016). O projeto LQG pode ser difícil para sistemas com grandes dimensões (WILLIGENBURG; DE KONING, 2000) e sem robustez garantida (ANGÉLICO *et al.*, 2017; SHALABY; EL-HOSSAINY; ABO-ZALAM, 2019; GREEN; LIMBEER, 2012).

O estado da arte em engenharia de controle trabalha com métodos de controle por *computação suave* (*soft computing methods – SCM*) (KOZÁK, 2014). As SCM são técnicas baseadas em inteligência artificial e seleção natural, fornecendo uma solução rápida e econômica para problemas complexos cuja formulação analítica é muito difícil de ser obtida ou inexistente. As técnicas SCM incluem aprendizado de máquina (*machine learning*), raciocínio probabilístico (*probabilistic reasoning*), redes neurais artificiais – RNA (*artificial neural network – ANN*), lógica difusa (*Fuzzy logic*) e algoritmo genético (*genetic algorithm - GA*). As vantagens associadas às SCM incluem a não exigência de uma formulação matemática abrangente relativa ao problema analisado, são capazes de lidar com múltiplas variáveis simultaneamente, além de serem métodos naturalmente adaptativos, escaláveis e robustos. Por essas características, as SCMs têm sido reconhecidas como alternativa atraente ao paradigma bem estabelecido de computação rígida (MALHOTRA; SINGH; SINGH, 2011).

Uma das linhas de pesquisa atual em controle moderno emprega RNAs (PRETORIUS; DU PLESSIS; GONSALVES, 2017; YANG *et al.*, 2014). Pesquisas com RNAs em controle de equipamentos robóticos, capazes de lidar com as incertezas e perturbações do sistema, tem sido investigadas (HE *et al.*, 2018; JAFARI; HAGAN, 2018; JIANG *et al.*, 2017; MORENO-VALENZUELA *et al.*, 2016). Além disso, as RNAs tem demonstrado serem mais eficientes na aproximação, identificação e tratamento

de sistemas não lineares (CHANDRAN *et al.*, 2015; RAJA GENAL, 2019; TUTUNJI, 2016; ZHANG *et al.*, 2017).

Estratégias de identificação e controle de sistemas dinâmicos tem sido implementados com lógica difusa (EINI; ABDELWAHED, 2018; PERI; SIMON, 2005; ROOSE; YAHYA; AL-RIZZO, 2017), baseados na teoria dos conjuntos difusos proposto por Lotfi A. Zadeh, permitindo que estados indeterminados possam ser tratados pelo dispositivo controlador (MOHAMMADIAN, 2017). Vários estudos implementam sistemas de controle combinando redes neurais com lógica difusa (*neuro-fuzzy networks – NFN, Neuro-Fuzzy Controller - NFC*) (LIN; LIAN, 2009; MEENAKSHI; MANIMOZHI, 2016; SHIHABUDHEEN; PILLAI, 2018). Estratégias de controle com lógica difusa são baseadas em regras advindas da experiência de um especialista, apresentam robustez em sistemas não lineares e não requerem o uso de modelos matemáticos complexos. Todavia, a utilização da lógica difusa em sistemas de controle pode requerer uma grande quantidade de parâmetros, regras, número de funções de pertinência e inferências, baseadas na experiência de um especialista (LILLY, 2010).

A lógica paraconsistente – LP (*paraconsistent logic – PL*) é uma alternativa à lógica difusa. A LP permite a análise de uma proposição com informações inconsistentes de modo controlado e discriminador (GUIDO *et al.*, 2013; PRIEST; KOJI; WEBER, 2018). A principal característica da LP é admitir a análise de informações contraditórias em sua estrutura teórica (DA SILVA FILHO, 2012; DE CARVALHO; ABE, 2018). A abordagem paraconsistente tem sido aplicada principalmente a problemas de engenharia envolvendo processamento de sinais e suporte à decisão em sistemas especialistas (DA FILHO FILHO, 2012; GARCIA *et al.*, 2019; ZAMANSKY, 2019). A lógica paraconsistente anotada evidencial – *LPAE $\varepsilon\tau$* , também chamada de lógica paraconsistente anotada com anotação de 2 valores – *LPA2v (paraconsistent annotated logic with 2 value annotation – PAL2v)* é uma extensão da LP onde a anotação de valores de duas variáveis e suas proposições são representadas em um reticulado finito. Algoritmos baseados na LPA2v oferecem maior velocidade de processamento, podendo ser implementados em sistemas de tempo real (DA SILVA FILHO *et al.*, 2016). O nó de análise paraconsistente - NAP (*paraconsistent analysis node – PAN*) é um bloco de programação baseado nos algoritmos e definições da LPA2v que pode ser aplicado em controle de sistemas (COELHO *et al.*, 2019). Por apresentar características similares ao neurônio humano,

o NAP é chamado de célula neural artificial paraconsistente - CNAP (*paraconsistent artificial neural cell – PANC*) (CARVALHO JUNIOR *et al.*, 2018; DA SILVA FILHO, 2012).

A LP permite que inconsistências sejam tratadas sem o perigo da trivialização da lógica clássica (DA SILVA FILHO; ABE; TORRES, 2008). A lógica clássica não separa os conceitos de inconsistência e trivialidade, pois se for aceita uma contradição, então qualquer conclusão se torna possível. Por ser indesejável, a lógica clássica não admite a contradição como elemento válido sem torna-la trivial. (DA SILVA FILHO; ABE; TORRES, 2008). Ao revogar o princípio da não contradição, a LP e suas extensões, como LPA2v, podem ser usadas para projetar sistemas capazes de lidar com problemas da vida real de forma mais eficiente que a lógica ordinária (FONSECA *et al.*, 2020; GUIDO *et al.*, 2013). A LPA2v permite manipular sinais contraditórios e assim apresentar uma ação mais rápida e confiável para a decisão, onde as informações para modelagem e controle podem ser vagas, ambíguas ou inconsistentes (DA SILVA FILHO; ROCCO, 2008). Redes de análise paraconsistentes, fundamentados na LPA2v, permitem que conceitos mais próximos da realidade possam ser considerados para a construção de modelos e ferramentas, capazes de manipular contradições e ambiguidades, abrindo espaço para o desenvolvimento de novas tecnologias, como aplicadas em automação e controle (DA SILVA FILHO; ABE; TORRES, 2008).

A principal contribuição desta investigação é demonstrar que a LPA2v pode ser empregada como função de ativação de neurônios treináveis para a construção de redes neurais paraconsistentes (RNP) de aprendizado supervisionado.

O foco deste trabalho é o desenvolvimento de redes neurais paraconsistentes, fundamentada nos algoritmos da LPA2v, que, de forma inédita, possam ser aplicadas na identificação e controle de sistemas ou processos dinâmicos.

1.1 OBJETIVOS

O objetivo principal desta investigação é propor uma metodologia de descrição e elaboração de arquiteturas de redes neurais construídas com neurônios fundamentados em LPA2v para serem aplicadas na aproximação de funções, identificação e desenvolvimento de modelos para controle de sistemas dinâmicos.

Para tanto, tem-se como objetivos específicos:

- a) A descrição matemática da aprendizagem do neurônio LPA2v;
- b) Elaboração do algoritmo da rede neural paraconsistente proposta;
- c) Identificação para controle (*identification for control – I4C*) em malha fechada de sistema dinâmico não linear, utilizando o protótipo de um pêndulo invertido rotativo - PIR (*rotary inverted pendulum - RIP*) como referência;
- d) Desenvolvimento de controle de sistema dinâmico não linear com RNP, utilizando o RIP como referência, sendo chamado de Controle por Rede Neural Paraconsistente Recorrente (CRNPR).

1.2 JUSTIFICATIVA

Diversas abordagens têm sido utilizadas no controle de sistemas dinâmicos não lineares. Muitos pesquisadores continuam implementando controlador linear LQR em sistemas não lineares devido à garantia de uma lei de controle ótima (HAMZA *et al.*, 2019). O aumento da capacidade computacional dos microprocessadores permitiu que o algoritmo de Kalman fosse largamente utilizado em conjunto com o LQR, formando o controlador LQG (VALADE *et al.*, 2017). Entretanto, não há garantias sobre as margens de estabilidade de um controlador LQG (DOYLE, 1978), cuja robustez pode ser recuperada se técnicas mais elaboradas como de recuperação de transferência de laço (*loop transfer recovery – LTR*) forem implementadas (DA CRUZ, 1996). Algoritmos de modos adaptativo e deslizante (*sliding mode control – SMC*) possuem o atrativo de serem robustos, apesar da maior complexidade e o potencial de aparecimento do fenômeno de trepidação (*chattering phenomena*) capaz de excitar a dinâmica do sistema e prejudicá-lo (KRAFES; CHALH; SAKA, 2016; NATH; DEWAN, 2019).

O projeto de controle baseado em redes neurais requer um conjunto grande de dados coletados por experimentos para o treinamento e teste da rede. O controle por lógica nebulosa requer enquadramento de regras, resultando em dezenas ou até centenas de linhas de código de decisão. Técnicas computacionais evolutivas, como algoritmo genético (GA) são lentas na computação, podendo ser limitantes em sistemas embarcados, com reduzida capacidade de processamento (PRASAD; TYAGI; GUPTA, 2014). A LPA2v pode ser uma alternativa de técnica de computação suave para identificação e controle, por apresentar uma fundamentação matemática simples e robusta, empregar poucas regras e considerações, resposta rápida e

confiável mesmo com informações incertas e permitir a construção de redes neurais paraconsistentes. Pesquisas com a LPA2v tem recebido interesse em diversas áreas do conhecimento humano nas últimas décadas (ABE; NAKAMATSU; DA SILVA FILHO, 2019). Rede de análise paraconsistente, baseada em LPA2v, foi recentemente implementada com sucesso na identificação de padrões acústicos (PAULO, 2016). Coelho *et al.* (2019) desenvolveu um controle híbrido PI-LPA2v para controle de processo de planta industrial com resultados promissores. A utilização de algoritmos fundamentados na LPA2v, com base matemática de baixa complexidade, reversibilidade e que consideram imprecisões de modelos e incertezas, deve ser investigada como parte da estratégia para identificação e controle de sistemas dinâmicos.

1.3 CONTRIBUIÇÕES DO TRABALHO PROPOSTO

O presente trabalho tem como principal contribuição o estudo, desenvolvimento e implementação de uma rede neural artificial, com função de ativação treinável fundamenta em LPA2v para a aproximação de funções, identificação, modelagem e controle de sistemas dinâmicos não lineares. O trabalho também apresenta o arcabouço matemático e propõe um algoritmo para o cálculo dos elementos que compõem a rede neural paraconsistente (RNP) implementada.

O emprego de LP em controle é relativamente recente, quando se trata de aplicações práticas, ainda assim participando em soluções híbridas no tratamento dos sinais ou como ator periférico da ação de controle, com emprego de nós de análise paraconsistente (NAP) pontuais, sem o contexto de aprendizagem caraterístico de redes neurais (ABE; NAKAMATSU; DA SILVA FILHO, 2019; COELHO *et al.*, 2019; ZAMANSKY, 2019).

É possível mencionar que a grande maioria dos trabalhos que utilizam lógica paraconsistente não tratam as evidências como sinais discretos e, portanto, não apresentam o desenvolvimento matemático da LPA2v e as CNAPs derivadas delas com a abordagem apresentada neste trabalho.

Este trabalho faz uma distinção entre as redes neurais artificiais paraconsistentes (RNAP) presentes na literatura, da RNP proposta. As RNAPs fazem uso de diferentes configurações de CNAPs, operando sobre limiares e são utilizadas principalmente na classificação de padrões, processamento de imagens e suporte a

decisão (ZAMANSKY, 2019). A RNP proposta, por outro lado, é constituída de neurônios com função de ativação LPA2v treinável por pesos e *bias*.

Finalmente, este trabalho propõe contribuir para o desenvolvimento de novas pesquisas e tecnologias voltadas para sistemas de controle aplicados em automação e robótica.

1.4 ORGANIZAÇÃO DA TESE

O capítulo 1 apresenta uma descrição do problema e os objetivos da tese. O capítulo 2 apresenta uma revisão da literatura e breve descrição sobre identificação de sistemas, estado da arte em controle, redes neurais e suas aplicações na identificação e controle de sistemas dinâmicos. Além disso, uma descrição da lógica paraconsistente anotada com anotação de 2 valores é também apresentada neste capítulo. O capítulo 3 apresenta o conjunto de equações e interpretações do reticulado associado à LPA2v, assim como as células neurais e rede neural artificial paraconsistente, disponível na literatura. O neurônio paraconsistente e o modelamento da rede neural paraconsistente, proposta neste trabalho para ser utilizada na aproximação de funções, identificação e controle, são apresentados no capítulo 4. O capítulo 5 apresenta a metodologia empregada e as ferramentas computacionais utilizadas para a análise e avaliação dos resultados. O capítulo 6 apresenta os resultados preliminares da utilização da RNP em aproximação de funções, identificação, modelagem e controle de sistemas dinâmicos. O capítulo 7 descreve as etapas realizadas a construção de RNPR, topologia de controlador, identificação e modelagem do sistema não linear utilizado e resultados obtidos dos ensaios realizados com o CRNPR. Finalmente, o capítulo 8 apresenta as considerações finais esperadas sobre o projeto de pesquisa deste trabalho e sugestões de continuidade.

2. ESTRATÉGIAS DE IDENTIFICAÇÃO E CONTROLE DE SISTEMAS DINÂMICOS

Este capítulo apresenta os conceitos fundamentais sobre identificação, teoria de controle e redes neurais artificiais. O capítulo apresenta ainda um levantamento de algumas das principais técnicas em identificação e controle de sistemas dinâmicos.

2.1 IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS

A aplicação de estratégias de controle avançado sobre um sistema ou processo dinâmico, requer o uso de um modelo matemático capaz de descrever a dinâmica deste sistema. A identificação de sistemas reúne ferramentas matemáticas e algoritmos que permitem a construção de modelos matemáticos de sistemas dinâmicos fazendo uso somente dos respectivos dados de entrada e saída observados (LJUNG, 2010).

Segundo Ljung (LJUNG, 1999, 2010), a metodologia de identificação de sistemas é quase uma arte, que consiste de um processo iterativo, envolvendo basicamente três ingredientes:

- a) Dados de entrada-saída;
- b) Conjunto de modelos candidatos (estrutura do modelo);
- c) Critério de seleção de um modelo dentre os disponíveis no conjunto, baseado na informação proveniente dos dados (método de identificação).

A figura 1 apresenta um sistema dinâmico e os sinais de interesse para a identificação do modelo.

Figura 1 – Sistema dinâmico com variável manipulada de entrada (u), saída (y), perturbação medida (w) e não medida (v).

Fonte: Ljung (1999).

A variável manipulada (u), ou sinal de controle, é aplicada na entrada do sistema. A saída do sistema (y), influenciada por (u), pode então ser medida. Perturbações

podem estar presentes no sistema que também afetem a saída (y). Elas podem ser perturbações medidas, mas não controladas (w) e perturbações que não podem ser nem medidas, nem controladas (v).

Os modelos para sistemas dinâmicos podem ser descritos de forma gráfica (modelos gráficos), através de modelos analíticos (matemáticos) ou computacionais, para sistemas complexos (LJUNG, 1999).

De acordo com o tipo de equação utilizado em sua formulação, eles podem ser classificados em estático (ou estacionário) ou dinâmico; linear ou não linear; única entrada – única saída (single input single output – SISO), múltiplas entradas – única saída (*multiple input single output* – MISO) ou múltiplas entradas – múltiplas saídas (*multiple input multiple output* – MIMO); paramétricos ou não- paramétricos; variáveis no tempo ou invariáveis no tempo; no domínio do tempo ou no domínio da frequência; tempo contínuo ou tempo discreto; amplitude contínua ou amplitude discreta; parâmetros concentrados ou distribuídos e determinísticos ou estocásticos.

A identificação de sistemas é uma aproximação do sistema. Procura-se encontrar um modelo com precisão aceitável a partir dos dados coletados, sendo que o modelo resultante é, necessariamente, uma aproximação da descrição verdadeira (LJUNG, 2010). Os modelos obtidos por metodologia de identificação de sistemas possuem uma faixa de validade limitada, por exemplo para um certo ponto de operação ou para um certo tipo de entrada. Quando a identificação do modelo é utilizada para fins de controle, o importante não é obter o que melhor se ajusta aos dados, como na identificação de sistemas clássica, mas sim um modelo que atenda bem aos critérios de desempenho do sistema controlado em malha fechada (*closed loop*). Essa abordagem de identificação orientada ao controle recebe o nome de identificação para controle (I4C) (FORMENTIN; CHIUSO, 2019; GEVERS, 2005).

Redes neurais artificiais tem recebido bastante atenção na identificação de sistemas, principalmente por serem eficazes para resolver um grande número de problemas de estimativa não linear (LJUNG, 2010). As RNAs são eficazes na modelagem de sistemas, como na implementação da identificação do sistema e mapeamentos complexos (ABIODUN *et al.*, 2018). Gautam (GAUTAM, 2016) demonstrou o emprego de RNAs na identificação de um sistema simulado de pêndulo invertido sobre carro. Tutunji (TUTUNJI, 2016) demonstrou que uma correlação matemática entre os pesos aprendidos por uma RNA e os parâmetros da função de transferência de um modelo de sistema não linear podem ser obtidos. Uma rede neural

artificial convolucional profunda (*deep convolutional neural network* – CNN) pode ser utilizada para estimar a resposta dinâmica de sistemas lineares e não lineares com grau único de liberdade (*single-degree-of-freedom* – SDOF) (WU; JAHANSHAH, 2019). Foi demonstrado que uma RNA multicamadas exibiu uma boa aproximação de modelo para um manipulador robótico flexível (JAMALI *et al.*, 2018).

2.2 ESTRATÉGIAS DE CONTROLE DE SISTEMAS DINÂMICOS

A pesquisa, desenvolvimento e implementação de novos princípios de controle é muito dinâmica no campo da automação (KOZÁK, 2014). A figura 2 apresenta a evolução e tendências no desenvolvimento de métodos de engenharia de controle.

Figura 2 – Evolução dos métodos de engenharia de controle.

Fonte: Adaptado de Kozák (2014, p.2).

Várias estratégias de controle são pesquisadas para sistemas dinâmicos, podendo ser classificadas em lineares, não lineares e controladores inteligentes (KRAFES; CHALH; SAKA, 2016). Controladores lineares como PP (ou FSF), LQR e LQG, utilizam técnicas de linearização para sintetizar uma lei de controle capaz de

estabilizar o sistema. Controladores não lineares apresentam boa robustez e tem sido desenvolvidos para o problema da estabilização do sistema (KRAFES; CHALH; SAKA, 2016). O estado da arte em teoria de controle inclui técnicas SCM, como *soft-sensors*, algoritmos de predição e otimização, sistemas especialistas, aprendizado, inteligência artificial e cognitiva, lógica difusa ou RNA (KOZÁK, 2014).

2.2.1 Controlador Linear por Alocação de Polos

A resposta transitória aprimorada ou a estabilidade de um sistema naturalmente instável, pode ser alcançada pelo posicionamento de polos de malha fechada (p_{CL}) na metade esquerda do plano do complexo s (SARKAR; DEWAN, 2017). Esse tipo de abordagem é chamado de realimentação de estado completo (*full state feedback* - FSF) ou controlador de posicionamento de polos (*pole placement* - PP) (SHEHU *et al.*, 2016). A representação em espaço de estados de um sistema é dada pela equação (1). Pode-se reescrever a (1) na forma discreta, sendo Φ e Γ os equivalentes discretos das matrizes A e B do sistema no tempo de amostragem utilizado (T_s) (2). Para um sistema controlável, pode-se determinar uma matriz de ganho K de realimentação de estados do controlador FSF, tal que a lei de controle $u = -Kx$ garanta a alocação arbitrária de polos do sistema em malha fechada, onde os p_{CL} são os autovalores da matriz $[\Phi - \Gamma K]$, conforme equação (3).

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{1}$$

$$\begin{aligned}x_{[n+1]} &= \Phi x_{[n]} + \Gamma u_{[n]} \\ y_{[n]} &= Cx_{[n]}\end{aligned}\tag{2}$$

$$x_{[n+1]} = [\Phi - \Gamma K]x_{[n]}\tag{3}$$

Se o sistema for totalmente controlável, existe uma grande liberdade de escolha para polos de malha fechada, tomando-se o cuidado de que isso não resulte em altos valores de K (SARKAR; DEWAN, 2017). Outra restrição do controlador FSF é que se o sistema a ser controlado for MIMO a matriz K não será única e, assim, a escolha dos valores de K não será trivial (FRANKLIN; POWELL; EMANI-NAEINI, 2018).

2.2.2 Controlador Linear Ótimo

A teoria do controle ótimo visa encontrar uma lei de controle para um determinado sistema, de forma que o melhor comportamento possível seja atendido para um critério específico de otimização (PRASAD; TYAGI; GUPTA, 2014). O regulador quadrático linear (LQR), introduzido por Kalman em 1964, é um controle quadrático ótimo aplicado a um problema de tempo infinito, celebrado por sua inerente robustez (NATH; DEWAN, 2019). Com base no modelo de espaço de estados (Φ, Γ) do RIP e, ao resolver a equação algébrica de Riccati, o controlador LQR visa ajustar a entrada ótima do sinal de controle (SUBRAMANIAN; ELUMALAI, 2016; KUMAR; JEROME, 2013). Dada a lei de controle da equação (4), deve-se encontrar os valores de K que minimizem o seguinte índice de desempenho (5) (FRANKLIN; POWELL; EMAMI-NAEINI, 2014).

$$u_{[n]} = -K_{[n]}x_{[n]} \quad (4)$$

$$J(u) = \frac{1}{2} \sum_{n=0}^{\infty} (x_{[n]}^T Q x_{[n]} + u_{[n]}^T R u_{[n]} + 2x_{[n]}^T N u_{[n]}) \quad (5)$$

Por definição $N = 0$ e pode ser omitido. A matriz Q ($k \times k$) é simétrica definida semi positiva, enquanto que a matriz R ($r \times r$) é simétrica definida positiva. Pode-se utilizar a regra de Bryson, como ponto de partida para escolha das matrizes Q e R , diagonais, conforme as equações (6) e (7), para se obter por tentativa e erro o comportamento desejado do sistema quando em malha fechada (FRANKLIN; POWELL; EMAMI-NAEINI, 2014). Os valores x_{imax}^2 e u_{imax}^2 são os valores máximos aceitáveis para x e u respectivamente.

$$Q_{ii} = \frac{1}{x_{imax}^2}, i = 1, 2, \dots, k \quad (6)$$

$$R_{jj} = \frac{1}{u_{jmax}^2}, j = 1, 2, \dots, r \quad (7)$$

Além de atender a uma lei de controle ótima, características apontadas para o controlador LQR com sistema não linear do tipo pêndulo invertido, em comparação

com outras estratégias de controle, incluem simplicidade, rejeição a distúrbios, resposta e trajetória média, precisão e trepidação baixas e adaptabilidade alta (KRAFES; CHALH; SAKA, 2016). Apesar disso, resultados de comparações de performance entre o LQR e outros controladores lineares não podem ser generalizadas em razão das especificidades do modelo do sistema e ajuste fino do controlador correspondente (HAMZA *et al.*, 2019).

2.2.3 Controlador Linear e Estimador Ótimo

Para o projeto de controladores FSF e LQR, todos os estados devem ser observáveis, ou seja, mensuráveis. Quando alguns estados não estão disponíveis, eles devem ser estimados. O filtro de Kalman (e suas variantes) é um dos algoritmos mais celebrados em processamento de sinais, sendo geralmente derivado através de álgebra vetorial, como mínimo estimador médio quadrático (FARAGHER, 2012). O filtro de Kalman é composto por um conjunto de equações recursivas capaz de estimar os estados de um sistema definido pelas equações (8) e (9), onde os valores de $w_{[n]}$ e $v_{[n]}$ representam o ruído do processo e ruído de medição, ambos com características de ruído branco gaussiano. Assim sendo, o filtro Kalman é considerado um estimador ótimo, também chamado de estimador quadrático linear (*linear quadratic estimator* – LQE).

$$x_{[n+1]} = \Phi x_{[n]} + \Gamma u_{[n]} + \Psi w_{[n]} \quad (8)$$

$$y_{[n]} = C x_{[n]} + v_{[n]} \quad (9)$$

O filtro Kalman propaga a média e covariância da função distribuição da probabilidade do modelo de estados em uma condição ótima com minimização do erro médio quadrático. A saída do filtro Kalman é resultado de ciclos consecutivos de predição e filtragem (GYÖRGY, 2019).

Considerando-se a atualização da medida na equação (10) e atualização temporal na equação (11):

$$\hat{x}_{[n]} = \bar{x}_{[n]} + L_{C[n]}(y_{[n]} - C\bar{x}_{[n]}) \quad (10)$$

$$\bar{x}_{[n]} = \Phi \hat{x}_{[n-1]} + \Gamma u_{[n-1]} \quad (11)$$

Definem-se como erros de estimação dados nas equações (12) e (13):

$$\tilde{x}_{[n]} = \hat{x}_{[n]} - x_{[n]} \quad (12)$$

$$\tilde{\bar{x}}_{[n]} = \bar{x}_{[n]} - x_{[n]} \quad (13)$$

As covariâncias dos erros de estimação da atualização da medida e atualização temporal são dadas respectivamente pelas equações (14) e (15):

$$P_{[n]} = \mathbb{E}\{\tilde{x}_{[n]}\tilde{x}_{[n]}^T\} \quad (14)$$

$$M_{[n]} = \mathbb{E}\{\tilde{\bar{x}}_{[n]}\tilde{\bar{x}}_{[n]}^T\} \quad (15)$$

O algoritmo para o cálculo do filtro de Kalman é apresentado a seguir, onde n é o número da amostra, R_W e R_V são as matrizes de covariância invariantes no tempo do ruído do processo e do ruído da medição e L_C o ganho do filtro (FARAGHER, 2012; GYÖRGY, 2019).

1. Definir:

$$R_V, R_W, \bar{x}_{[0]}, M_{[0]}$$

2. Fazer: $n = 0$

3. Calcular:

$$P_{[n]} = M_{[n]} - M_{[n]}C^T(CM_{[n]}C^T + R_V)^{-1}CM_{[n]}$$

$$L_{C[n]} = P_{[n]}C^TR_V^{-1}$$

$$\hat{x}_{[n]} = \bar{x}_{[n]} + L_{C[n]}(y_{[n]} - C\bar{x}_{[n]})$$

4. Atualizar:

$$\bar{x}_{[n+1]} = \Phi\hat{x}_{[n]} + \Gamma u_{[n]}$$

$$M_{[n+1]} = \Phi P_{[n]} \Phi^T + \Psi R_W \Psi^T$$

5. Avançar: $n = n + 1$

6. Retornar ao passo 3.

A combinação dos algoritmos do estimador Kalman e controlador LQR resultam no controlador linear quadrático gaussiano (*linear quadratic gaussian – LQG*). O LQG pode ser projetado calculando-se independentemente o estimador Kalman e o controlador LQR, conforme apresentado na figura 3 (ZHENG; ZHONG; YUE, 2016).

Figura 3 – Controle LQG para um sistema dinâmico não linear.

Fonte: Adaptado de Zheng, Zhong e Yue (2016).

Boas propriedades de robustez são garantidas no LQR quando a matriz R é diagonal. Entretanto, não há garantias sobre margens de estabilidade para o controlador (DOYLE, 1978). Pode-se recuperar a robustez do LQG com o procedimento denominado recuperação de transferência do laço (*loop transfer recovery* – LTR) (DA CRUZ, 1996).

2.2.4 Controlador Robusto Não Linear

O controle de modo deslizante (*sliding mode control* – SMC) pertence à família de controle não linear, baseada nas considerações de estabilidade do sistema de Lyapunov e que apresenta alta robustez e precisão, sendo por essas características muito estudado em problemas de pêndulo invertido (KRAFES; CHALH; SAKA, 2016). O SMC é um dos sistemas de controle robustos mais conhecidos e que utiliza uma superfície deslizante para formular a lei de controle (NATH; DEWAN, 2019). A figura 4 apresenta o diagrama de blocos do controle SMC, onde r é o sinal de referência e o erro, e .

Figura 4 – Controle SMC para o RIP.

Fonte: Adaptado de Krafes, Chalh e Saka (2016).

O projeto do controlador SMC ocorre em duas etapas. A etapa inicial que consiste em escolher a superfície deslizante na qual o sistema apresente a dinâmica desejada. A etapa seguinte consiste em garantir o alcance e modo deslizante pela escolha da lei de controle aproximada (NATH; DEWAN, 2019). Entretanto, esse controle leva ao aparecimento do fenômeno da trepidação (*chattering*), que pode excitar a dinâmica do sistema, podendo danificar ou até destruir o sistema estudado (KRAFES; CHALH; SAKA, 2016).

O projeto do SMC para o RIP é descrito por Nath e Dewan (2019). A superfície de comutação (*switching surface*) é dada pela equação (16), sendo que SB deve ser uma hipótese não singular. A equação de controle pode ser entendida como a soma de uma parcela de controle linear com outra de controle não linear, conforme (17).

$$s(t) = Sx(t) \quad (16)$$

$$u(t) = u_l(t) + u_n(t) \quad (17)$$

Nos resultados simulados apresentados por Nath e Dewan (2019), apesar de o controlador SMC apresentar maior robustez que o LQR para o RIP, nota-se que devido ao efeito de *chattering*, o controle SMC trepida muito mais do que o LQR, além de uma variação de transitório duas ordens de grandeza maior, quando na presença de ruído (NATH; DEWAN, 2019).

2.2.5 Controle de Modelo Interno

O controle de modelo interno (*internal model control* – IMC) está fundamentado na teoria do modelo interno, que afirma que o controle somente pode ser alcançado se o sistema de controle incorporar, de modo implícito ou explícito, alguma representação do processo a ser controlado. Caso o controle seja desenvolvido com base em um modelo exato do processo, o controle perfeito será teoricamente possível se as seguintes considerações forem atendidas (OTHMAN; CHEIKH; BOUZAOUACHE, 2017):

- a) A função do controlador é igual ao inverso do modelo da planta;
- b) O estado inicial do controlador é igual ao estado inicial do modelo da planta;
- c) O sistema IMC é estável em malha fechada.

Para que o IMC possa ser implementado, o modelo invertido da planta deve ser possível. O principal problema dessa estrutura é como inverter o modelo da planta a fim de se obter um controlador coerente que garanta a estabilidade e rejeição aos distúrbios (HANNACHI; SOUDANI, 2016).

A figura 5 apresenta um diagrama do IMC combinando os modelos direto e inverso da planta. O controlador representa o inverso do modelo da planta ($C(z)$). O modelo direto da planta é dado por ($M(z)$). Distúrbios, ruídos e incertezas são adicionados por (p) na saída da planta. Um filtro de primeira ordem pode ser adicionado na entrada do controlador para compensar eventuais descasamentos entre o RIP e o seu modelo, aumentando a robustez do sistema. Os blocos de retenção de ordem zero (*zero order hold – ZOH*) foram omitidos por simplificação.

Figura 5 – Controle IMC para um sistema dinâmico.

Fonte: Adaptado de Othman, Cheikh e Bouzaouache (2017).

Se os modelos forem capazes de representar as não linearidades da planta, o IMC será um controle não linear. Além disso, ele apresenta características interessantes, como robustez, design simples e forte anti-interferência (ZHANG; TAN; MA, 2014). Assumindo que os modelos podem ser adquiridos e modelos existentes possam ser atualizados, a estrutura do IMC pode servir de base para o desenvolvimento de controles adaptativos e com RNAs (JIANG *et al.*, 2017).

2.3 REDES NEURAIS APLICADAS À IDENTIFICAÇÃO E CONTROLE

Redes neurais artificiais são algoritmos matemáticos capazes de “aprender” uma determinada função ou reconhecimento de padrões, despertando interesse em

identificação e controle de sistemas (ABIODUN *et al.*, 2018). A figura 6 apresenta o diagrama de blocos conceitual de uma RNA para aproximar uma função.

Figura 6 – ANN utilizada na aproximação de função.

Fonte: Adaptado de Hagan, Denuth e De Jesús (2002, p.11).

As redes neurais são constituídas de neurônios artificiais cujas funções matemáticas são inspiradas em neurônios biológicos, constituindo a base de redes neurais artificiais. Apesar de não ter a mesma complexidade de um cérebro, as RNAs apresentam duas similaridades básicas com as redes neurais biológicas: a possibilidade de descrição de seus blocos de construção por dispositivos computacionais simples e as conexões entre os neurônios determinam a função da rede (HAGAN *et al.*, 2014). A representação de um neurônio biológico é apresentada na figura 7.

Figura 7 – Representação de um neurônio biológico.

Fonte: Adaptado de Jiang, *et al.* (2017, p.2).

Os dendritos são filamentos responsáveis por receber os sinais de informações para as células. O corpo celular, ou soma, reúne as informações recebidas pelos dendritos. Já o axônio transporta os impulsos elétricos que partem do corpo celular até as diversas ramificações com dilatações bulbosas conhecidas como terminais axônicos (ou terminais nervosos), que estabelecem as conexões sinápticas com outras células.

Um dos primeiros neurônios artificiais foi introduzido em 1943 por W. McCulloch e W. Pitts (HAGAN *et al.*, 2014). A figura 8 apresenta uma representação de neurônio artificial de múltiplas entradas.

Figura 8 – Representação de neurônio artificial de múltiplas entradas.

Fonte: Adaptado de Hagan, *et al.* (2014, p. 2-7).

Os pesos (w) representam a força das sinapses e são multiplicados aos valores das respectivas entradas e somados, juntamente com um valor de ajuste ou *bias* (b). O resultado desta soma (n) é então aplicado a uma função de ativação (f) e apresentado na saída (a) do neurônio artificial. O neurônio artificial de McCulloch e Pitts, é também chamado de unidade lógica de limiar (*threshold logic unit – TLU*) pois utilizava uma função de ativação de limite binário (*hard limit transfer function*). A primeira aplicação prática de RNA surgiu no final da década de 1950, com a invenção da rede neural perceptron, por Frank Rosenblatt (HAGAN *et al.*, 2014), mediante a interligação de vários neurônios do tipo TLU.

O neurônio artificial pode ser descrito pelas equações (18) e (19), sendo que esta última equação está na forma de matriz.

$$n = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,3}p_3 + \dots + w_{1,R}p_R \quad (18)$$

$$a = f(Wp + b) \quad (19)$$

Onde W é a matriz formada pelos pesos $[w_{1,1}; w_{1,2}; \dots w_{1,R}]$ e b é uma correção de *bias*.

Várias funções f podem ser utilizadas, de acordo com a finalidade, tais como: limite binário (*hard limit*), linear, log-sigmoide (σ), sigmoide tangente hiperbólica ($\tanh$), unidade linear retificada (*rectified linear unit* – ReLU) e suas variações como Leaky-ReLU (LReLU), entre outras (APICELLA *et al.*, 2021; HAGAN *et al.*, 2014). A combinação de pesos, *bias* e função permite que o neurônio aproxime qualquer função. É importante que a função f adotada seja derivável, de modo a permitir a elaboração de algoritmos de regressão para a calibração (“aprendizado”) dos pesos e *bias* do neurônio (HAGAN *et al.*, 2014).

Pode ser demonstrado que se a função de ativação for não linear, uma RNA de duas camadas pode ser um aproximador universal de função (SONODA; MURATA, 2017). Por exemplo, em controle de sistemas, o objetivo é obter uma função de realimentação (*feedback*) apropriada que mapeie de saídas medidas para entradas de controle (HAGAN; DEMUTH; DE JESÚS, 2002). Uma das funções de ativação mais utilizadas nas RNAs é a log-sigmoide, por ser não linear e pela facilidade de implementação de sua derivada no processo ajuste dos pesos, conforme as equações (20) e (21).

$$a = \sigma(n) = \frac{1}{1 + e^{-n}} \quad (20)$$

$$\sigma'(n) = \sigma(n)(1 - \sigma(n)) \quad (21)$$

A saída varia de 0 (deativação) a 1 (ativação). Sua derivada tende a 0 para valores de entrada maiores que +5 e abaixo de -5.

A $\tanh$ é outra função do tipo sigmoide, variando entre -1 e +1, cuja expressão e derivada são dadas pelas equações (22) e (23), respectivamente.

$$a = \tanh(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}} \quad (22)$$

$$\tanh'(n) = 1 - \tanh^2(n) \quad (23)$$

A função ReLU é muito parecida com a função identidade, fazendo com o processo de aprendizagem da RNA, baseado nessa função de ativação, seja mais rápido que por sigmoide (LIU *et al.*, 2019). A função ReLU e sua derivada são expressas conforme equações (24) e (25).

$$a = \text{LReLU}(n) = \begin{cases} 0 & n \leq 0 \\ n & n > 0 \end{cases} \quad (24)$$

$$\text{ReLU}'(n) = \begin{cases} 0 & n \leq 0 \\ 1 & n > 0 \end{cases} \quad (25)$$

Uma desvantagem da ReLU é que os neurônios tendem a “morrer” durante o treinamento, causando a saída do neurônio iniciar a produzir apenas zeros. Uma variação da ReLU, chamada Leaky-ReLU (LReLU) evita isso (LIU *et al.*, 2019), cuja função e sua derivada são apresentadas pelas equações (26) e (27).

$$a = \text{ReLU}(n) = \begin{cases} \alpha n & n \leq 0 \\ n & n > 0 \end{cases} \quad (26)$$

$$\text{ReLU}'(n) = \begin{cases} \alpha & n \leq 0 \\ 1 & n > 0 \end{cases} \quad (27)$$

Onde α é um parâmetro introduzido na LReLU, com valores propostos entre 0.01 e 0.2 (LIU *et al.*, 2019). A LReLU permite o neurônio ter um pequeno gradiente quando ele não está ativo ($n < 0$), reduzindo o problema potencial mencionado sobre ReLU (APICELLA *et al.*, 2021). Uma análise da robustez de redes neurais em sistemas de controle usando funções não lineares tanh e ReLU pode ser vista em (EVERETT; HABIBI; HOW, 2020).

A interligação de vários neurônios em camadas formam as redes neurais artificiais (JAFARI; HAGAN, 2018). Vários são os arranjos na interconexão dos neurônios, de modo a compor uma RNA. A figura 9 apresenta um exemplo de RNA baseado no modelo perceptron, com três camadas de neurônios, R entradas e S neurônios por camada e S saídas.

Figura 9 – RNA de 3 camadas.

Fonte: Adaptado de Hagan, *et al.* (2014, p. 2-11).

O aprendizado de uma RNA consiste na modificação dos pesos das conexões entre os neurônios de forma iterativa, por um algoritmo que segue um dos seguintes processos de aprendizado (HAGAN *et al.*, 2014):

- Aprendizado Supervisionado: consiste de um conjunto de treino, com entradas correspondentes e saídas desejadas. A regra de aprendizado é utilizada para ajustar pesos e *bias* da rede neural de modo que as saídas da RNA se aproximem das saídas desejadas;
- Aprendizado por Reforço: Similar ao aprendizado supervisionado, exceto que, em vez de receber a saída correta para cada entrada da rede, o algoritmo recebe apenas uma pontuação (*grade* ou *score*), como uma medida do desempenho da rede para uma determinada sequência de entradas;
- Aprendizado Não-supervisionado: A rede atualiza seus pesos e *bias* apenas em resposta às entradas da rede. Não há saída alvo disponível. Em geral, algoritmos de aprendizado não-supervisionado executam algum tipo de operação de agrupamento (*clustering*). Eles aprendem a categorizar os padrões de entrada em um número finito de classes.

O algoritmo de retropropagação (*backpropagation*) é amplamente utilizado no aprendizado de RNA, especialmente em redes neurais do tipo *feedforward* para aprendizado supervisionado. O algoritmo de *backpropagation* funciona calculando-se

o gradiente da função de perda em relação a cada peso pela regra da cadeia (*chain rule*), ou seja, a contribuição de cada peso aplicado na entrada correspondente para a função perda de uma determinada saída em relação à referência. O gradiente descendente (*gradiente descent*) é calculado uma camada por vez, repetindo a última camada para evitar cálculos redundantes de termos intermediários na regra da cadeia.

Quando a RNA utiliza um efeito memória, isto é, quando a saída atual de um neurônio depende dos valores aplicados em tempos anteriores em suas entradas, ela é chamada de rede neural artificial recorrente - RNR (*recurrent neural network* – RNN). As RNRs apresentam poderosas habilidades computacionais, porém são mais difíceis de serem treinadas de forma consistente (PHAN; HAGAN, 2013).

2.3.1 Identificação e Controle de Sistemas com Redes Neurais

As RNAs podem ser utilizadas na aproximação, ou aprendizado, de funções, sendo boas candidatas para representar sistemas dinâmicos não lineares, tanto em identificação como controle de sistemas (JAFARI; HAGAN, 2018). As razões pelas quais as RNAs despertam interesse em controle são a capacidade de aprendizagem, auto-organização, a tolerância a falhas, a flexibilidade e a possibilidade de ser utilizada em tempo real (HAGAN *et al.*, 2014).

Pode-se utilizar a RNA para aprender, de forma supervisionada, a se comportar como um controle existente, conforme a figura 10a. Quando o aprendizado da RNA está concluído, o controlador de referência pode ser retirado do circuito e a RNA deve ser capaz de controlar o sistema com desempenho semelhante. Aparentemente, não há vantagem em utilizar uma RNA para “copiar” um controle que já está implementado. Uma vantagem de utilização da RNA é que ela seja capaz de se adaptar às incertezas e variações da planta, *online*, podendo exercer a função de controle em situações que o controlador clássico não conseguiria. Na figura 10b, um sinal de erro é utilizado para calibrar *online* os pesos da RNA, baseado na diferença entre a saída da planta e a saída desejada (y_d), formando assim um controle adaptativo. A RNA pode ser utilizada para implementar o controle invertido direto, conforme a figura 11.

(a)

(b)

Figura 10 – Aprendizado supervisionado do controle (a). RNA em controle adaptativo (b).

(a)

(b)

Figura 11 – Identificação do modelo invertido do sistema pela RNA (a). Controle invertido direto por RNA (b).

Fonte: Adaptado de Lewis e Ge (2015, p. 4)

Através de uma base de dados previamente coletada, pode-se utilizar uma RNA para aprender o modelo invertido da planta em modo *offline*, conforme a figura 11a. Os pesos aprendidos são então utilizados para programar a RNA para atuar como controle invertido da planta (LEWIS; GE, 2015). Como o controlador invertido está em

série com o sistema, conforme a figura 11b, a saída combinada do sistema terá o mesmo valor da referência (r). O modelo invertido por RNA deve ser bem preciso, de modo que as não linearidades do controlador invertido anulem as não linearidades do sistema.

Por operar com várias camadas e vários neurônios em paralelo, a RNA pode ser utilizada para modelar um sistema dinâmico MIMO (CHANDRAN *et al.*, 2015). Gautam (GAUTAM, 2016) demonstrou que uma RNA pode ser utilizada para modelar um pêndulo invertido, como o pêndulo invertido rotativo (*rotary inverted pendulum* – RIP). O pêndulo invertido é um sistema instável em laço aberto. Para a RNA aprender e ser utilizada com o pêndulo, o mesmo deve ser estabilizado de alguma maneira, por exemplo utilizando-se algum método de controle clássico realimentado (malha fechada). Para uma melhor modelagem das não linearidades e dinâmicas do pêndulo, pode-se utilizar um controlador descalibrado, no limiar da instabilidade do sistema (GAUTAM, 2016). Utilizando-se uma RNA com um número adequado de neurônios, camadas e saídas, sendo posicionada em paralelo com a planta, pode-se identificar o modelo de um sistema dinâmico não linear, como o pêndulo invertido, conforme a figura 12 (RAJA GENAL, 2019).

Figura 12 – Identificação do modelo do pêndulo invertido pela RNA.

Fonte: Adaptado de Raja Genal (2019, p. 1834).

Diversas topologias de controle de sistemas com realimentação (*feedback control systems*) utilizando RNA têm sido propostas como uma evolução elegante das topologias padrão em controle adaptativo (LEWIS; GE, 2015). Por exemplo, derivações de controle IMC com RNA, como a incorporação de modelo de referência na arquitetura (*model reference control* – MRC) (JAFARI; HAGAN, 2018) e uso de

controle baseado em previsão de modelo (*model predictive control – MPC*) (EINI; ABDELWAHED, 2018).

A figura 13 apresenta uma topologia de IMC robusto adaptativo utilizando RNA. O modelo neural inverso do controlador pode ser obtido conforme a figura 11a a partir de um controle de referência existente. Primeiro treina-se a RNA do modelo direto da planta. O próximo passo é treinar o modelo inverso do controlador. Aplicações de IMC com RNA no controle de sistemas de pêndulo invertido tem sido propostas (SHI *et al.*, 2019; SHOJAEI *et al.*, 2012).

Figura 13 – RNA como controle IMC.

Fonte: Adaptado de Mares, Dolezel e Hrnčirik (2011, p. 362).

Outra abordagem utilizando modelos da planta e do controlador é o MRC ou MRC adaptativo (*model reference adaptive control - MRAC*), cuja arquitetura pode ser observada na figura 14 (JAFARI; HAGAN, 2018). O MRC com RNA foi proposto inicialmente por Narendra e Parthasarathy (1990) em seu artigo sobre identificação e controle de sistemas dinâmicos utilizando redes neurais.

Uma consideração implícita na formulação do problema MRAC é o que projetista esteja suficientemente familiarizado com a planta sob consideração de modo que ele possa especificar o comportamento desejado da planta em termos da saída de um modelo de referência (NARENDRA; PARTHASARATHY, 1990).

Javari e Hagan (2018) propuseram o MRC para o controle de uma planta dinâmica não linear de levitador magnético (JAFARI; HAGAN, 2018). Nesse controle o MRC faz uso de duas redes neurais recorrentes – RNR (*recurrent neural network – RNN*), que são uma classe de redes neurais que incluem conexões ponderadas dentro de uma camada, na forma de loops, podendo armazenar informações ao processar novas entradas. A primeira RNA é treinada para modelar a planta, identificada

utilizando um modelo não linear autorregressivo exógeno (*nonlinear autoregressive exogenous model* – NARX). A segunda RNA é o controlador, que é treinada de modo que a saída da planta siga a saída do modelo de referência.

Figura 14 – Estrutura MRC.

Fonte: Adaptado de Jafari e Hagan (2018, p. 318).

RNRs são boas candidatas para identificação e controle de sistemas dinâmicos não lineares, apesar da dificuldade em serem treinadas devido a problemas de desaparecimento (*vanishing*) e explosão de gradiente. Esse problema é mais relevante em aplicações de RNA processamento de linguagem natural e pouco significativo para identificação de sistemas não lineares (HAGAN *et al.*, 2014; JAFARI; HAGAN, 2018).

O NARX é um modelo autorregressivo (AR) não linear com entradas exógenas, utilizado na modelagem de séries temporais (AGUIRRE, 2020). O modelo relaciona o valor atual de uma série temporal a valores anteriores da mesma série e a valores atuais e passados da série motriz (exógena), ou seja, da série determinada externamente que influencia a série de interesse. O modelo contém ainda um termo de erro sendo que o conhecimento dos outros termos não permite que o valor atual da série temporal seja previsto com exatidão. Algebricamente o modelo NARX pode ser visto conforme equação (28).

$$y(k) = F[y(k-1), \dots, y(k-n_y), u(k-d), \dots, u(k-d-n_u+1)] + e(k) \quad (28)$$

Onde n_y e n_u são máximos atrasos nos sinais de saída e entrada, e o termo de erro (algumas vezes chamado de ruído) e d é o atraso puro de tempo, medido em períodos de amostragem T_s . A função $F[\cdot]$ é não linear, podendo ser representada por diversas representações matemáticas, como por redes neurais multicamadas (*multilayer perceptron* – MLP) (AGUIRRE, 2020).

São propostas na literatura diversas abordagens de controle de sistema de pêndulo invertido com MRC (MENDEZ-FLORES *et al.*, 2019; PAWAR; PARVAT, 2015; SARKAR; DEWAN, 2017).

Apesar das vantagens apresentadas pelas RNAs em identificação de sistemas e controle adaptativo, os desafios continuam sendo a escolha da função de ativação, definição da topologia, quantidade de neurônios e camadas da rede, quantidade de amostras e ciclos ou épocas necessários para aproximação da função (*epoch*), tempo para convergência e aprendizado da rede neural (JIANG *et al.*, 2017).

2.4 REVISÃO DA LITERATURA SOBRE APLICAÇÕES DA LPA2v EM IDENTIFICAÇÃO E CONTROLE

Incertezas na modelagem e descrição de um sistema a ser controlado, ruídos e perturbações são aspectos inerentes à engenharia de controle. A lógica paraconsistente permite a construção de redes neurais e, por tratar contradições, tem sido utilizada em sistemas especialistas que consideram informações incertas na tomada de decisão (DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010).

Uma das primeiras aplicações práticas de lógica paraconsistente consistiu da divisão do diagrama Lattice da LPA2v em 12 regiões, de modo a classificar os sinais recebidos por dois sensores de ultrassom, permitindo a tomada de decisão no controle de direção e desvio de obstáculos pelo robô Emmy. Esse controle foi embarcado em um microcontrolador e recebeu o nome de Paracontrol (ABE; NAKAMATSU; DA SILVA FILHO, 2019). Nakamatsu, K. (1999) propôs o programa de lógica anotada de vetor estendido (EVALP) sem forte negação para um sistema automatizado de verificação de segurança de controle de tráfego aéreo (ABE; NAKAMATSU; SILVA FILHO, 2019). Mario (MARIO, 2006) utilizou uma rede neural artificial paraconsistente para desenvolver um modelo de análise de variáveis craniométricas. A identificação de padrões de sinais acústicos através de uma classificação empregando LPA2v foi utilizada em (PAULO, 2016). Torres (2010) propôs a utilização de RNAP para a

escolha da rota com menor distância, para navegação de robôs móvel autônomo em ambientes não estruturados (ABE; NAKAMATSU; DA SILVA FILHO, 2019). A utilização de nós de análise paraconsistente no tratamento de variáveis para o controle proporcional-integral (PI) de uma planta industrial foi proposto por (COELHO *et al.*, 2019), recebendo o nome de *Hybrid PI Controller*.

A divisão do reticulado da LPA2v tem sido utilizada para a tomada de decisão em diferentes aplicações. Aplicações com redes ou sistemas de análise paraconsistente, empregando blocos de código da LPA2v também são encontradas na literatura (GARCIA *et al.*, 2019). A rede neural artificial paraconsistente utilizada em diferentes aplicações, não “aprende” ou aproxima funções mediante a obtenção dos valores de pesos corretos, mas toma decisões dentro da rede através do ajuste de limiares de transição, ou fatores, utilizados em diferentes células neurais paraconsistentes. Por esta razão, há muitos exemplos de aplicações na classificação de padrões (DA SILVA FILHO; ABE; TORRES, 2008; ABE; LOPES; RENATO, 2015).

A proposta deste trabalho é a utilização das equações e interpretações da LPA2v, de forma inovadora como função de ativação dos neurônios, para compor redes neurais paraconsistentes (RNP) capazes de aprender através do ajuste de pesos. Assim, será demonstrado neste trabalho que uma RNP possa ser utilizada na aproximação de funções, com resultados melhores que outras funções de ativação (tipos sigmóides e ReLU), tomando como referência a identificação e controle de sistemas dinâmicos, ampliando o potencial de uso da LPA2v.

O próximo capítulo apresenta o arcabouço matemático da LPA2v, base para o desenvolvimento deste trabalho.

2.5 SUMÁRIO DO CAPÍTULO

Este capítulo mostrou que, com recursos e ferramentas matemáticas utilizados sobre dados de entrada e saída, é possível realizar o modelamento ou identificação de um sistema.

Diversas abordagens utilizadas pela teoria de controle moderno para sistemas dinâmicos, fazem uso de modelos matemáticos, podendo ser divididos em lineares, não lineares e controladores inteligentes.

Uma das linhas de pesquisa atual na modelagem ou identificação e controle de sistemas, trabalham com redes neurais artificiais – RNAs.

As redes neurais são circuitos de neurônios artificiais utilizando uma determinada função de ativação. Atuando-se sobre os pesos entre as conexões, as RNAs podem aprender a aproximar funções. Além disso, a característica de auto-organização, a tolerância a falhas, a flexibilidade e a possibilidade de ser utilizada em tempo real tem despertado aplicações no estado da arte com redes neurais, nas áreas de identificação e controle de sistemas dinâmicos.

Estratégias adaptativas com redes neurais, como MRC, podem ser utilizadas na identificação e controle de sistemas dinâmicos não lineares, como o pêndulo invertido.

Pesquisas e aplicações utilizando LPA2v no reconhecimento de padrões e controle foram apresentadas. Nessas pesquisas, a LPA2v foi utilizada de forma tradicional (reticulado), através redes de análise paraconsistente (NAP) e redes neurais artificiais paraconsistentes para a tomada de decisões e classificação de padrões, mediante ajuste de limiares de transição.

3. LÓGICA PARACONSISTENTE APLICADA À REDE NEURAL

Este capítulo apresenta um breve histórico da evolução da Lógica Paraconsistente Evidencial, também chamada de Lógica Paraconsistente Anotada, com anotação de 2 valores, bem como o arcabouço matemático que fundamenta esta pesquisa.

3.1 EVOLUÇÃO DA LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL $\varepsilon\tau$

Vários pesquisadores têm contribuído ao longo dos séculos no estudo da contradição. O próprio Aristóteles (384-322 a.C.), pai da lógica clássica, menciona Heráclito como um dos que negavam o princípio da não contradição. Este princípio da lógica clássica afirma que entre duas proposições contraditórias, uma delas é falsa. Entre os precursores da lógica paraconsistente - LP (*paraconsistent logic* - PL) destacam-se os poloneses Jean Lukasiewicz (1876-1956) e seu discípulo Stanislaw Jaśkowski (1906 – 1965), além do russo Nicolai Vasiliev (1880 – 1940) que de forma independente, no início do século XX, constataram que alguns princípios da lógica clássica admitiam revisão, inclusive o princípio da contradição (CARNIELLI; CONIGLIO, 2016). Em 1948, Jaśkowski apresentou uma lógica para sistemas envolvendo contradições, chamada de lógica discutível (*discussive logic*), porém sem ser trivial (CARNIELLI; CONIGLIO, 2016). Os trabalhos do matemático brasileiro Newton C. A. da Costa (1929 -), a partir da década de 1950, deram um tratamento tanto filosófico quanto matemático, desenvolvendo sistemas formais envolvendo contradições e lógica paraconsistente (ABE; NAKAMATSU; SILVA FILHO, 2019; CARNIELLI; CONIGLIO, 2016), sendo considerado um dos criadores da LP. Desde então um grupo ativo de lógicos tem pesquisado a LP continuamente, especialmente em São Paulo, Brasil, com foco em lógicas de inconsistência formal (PRIEST; KOJI; WEBER, 2018).

O filósofo peruano Francisco Miró Quesada (1918-2019) criou o termo “lógica paraconsistente”, em substituição a denominação até então usada de “Teoria dos Sistemas Formais Inconsistentes” (BORGES DE LIMA *et al.*, 2018).

Por revogar o princípio da contradição, a LP trabalha com situações onde ambos uma sentença e sua negação podem ser verdadeiras (DA COSTA; KRAUSE; BUENO, 2007). Em adição aos estados lógicos verdadeiro e falso da lógica clássica, uma

proposição pode ainda ter outros estados, como inconsistente (T) e indeterminado ($\perp$) (DA SILVA FILHO, 2012).

A LP é classificada como lógica paraconsistente anotada - LPA (*paraconsistent annotated logic – PAL*) quando uma proposição P é acompanhada com valores correspondentes à fórmula proposicional, composta de graus de “crença” ou evidência (μ). O termo “anotado” foi inicialmente proposto por Subrahmanian (ABE; NAKAMATSU; SILVA FILHO, 2019). Os graus de evidência são representados em um diagrama de Hasse (τ) composto por um Lattice de quatro vértices, também chamado de *reticulado de valores-verdade*. O grau de evidência é uma constante anotacional do reticulado. Assim sendo, cada grau de evidência atribuído à proposição está contido no conjunto de valores composto pelas constantes anotacionais do reticulado $\{T, V, F, \perp\}$ (DA SILVA FILHO, 2012).

3.2 DESCRIÇÃO DA LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL $\varepsilon\tau$

Na LP, uma proposição P pode ser suportada por 1, 2 ou mais graus de evidência. Quando um par de valores simbólicos em um reticulado no plano real é utilizado para expressar o conhecimento sobre a proposição P , na forma de fórmula atômica evidencial $P(\mu, \lambda)$, a lógica é chamada de lógica paraconsistente anotada evidencial (LPAE) (ABE; NAKAMATSU, 2013) ou LPA com anotação de 2 valores – LPA2v (*paraconsistent annotated logic with 2 value annotation - PAL2v*), onde $\tau = \{(\mu, \lambda) | \mu, \lambda \in [0, 1] \subset \mathbb{R}\}$ (COELHO *et al.*, 2019; DA SILVA FILHO, 2012). A figura 15 apresenta o diagrama de Lattice para LPA e LPA2v.

Figura 15 – Lattice de 4 vértices da LPA(a); Lattice finito de Hasse com anotações LPA2v (b).

Fonte: Adaptado de Coelho, et al. (2019, p. 114).

Os valores do grau de evidência favorável (μ) e grau de evidência desfavorável (λ) são completamente independentes. Nesse caso, um operador de negação $\sim$ é introduzido: $|\tau| \rightarrow |\tau|$. Assim sendo:

- $\sim T = T$ a negação de uma proposição “inconsistente” é “inconsistente”;
- $\sim V = F$: a negação de uma proposição “verdadeira” é “falsa”;
- $\sim F = V$: a negação de uma proposição “falsa” é “verdadeira”;
- $\sim \perp = \perp$: a negação de uma proposição “paracompleta” é “paracompleta”.
- $\sim(\mu, \lambda) = (\lambda, \mu)$

Da figura 15b, pode-se deduzir que:

- a) $P_V = P_{(1,0)} \rightarrow$ indica existência de evidência favorável total e evidência desfavorável nula, apontando uma conotação lógica de *Verdade* à proposição P .
- b) $P_F = P_{(0,1)} \rightarrow$ indica existência de evidência favorável nula e evidência desfavorável total, apontando uma conotação lógica de *Falsidade* à proposição P .
- c) $P_T = P_{(1,1)} \rightarrow$ indica a existência de ambos, evidência favorável e desfavorável totais, apontando uma conotação lógica de *Inconsistência* à proposição P .
- d) $P_\perp = P_{(0,0)} \rightarrow$ indica a existência de ambos, evidência favorável e desfavorável nulas, apontando uma conotação lógica de *Indeterminação* à proposição P .

A coleta de valores para μ e λ objetivam solucionar o desafio de sinais contraditórios, em que, através de análises sucessivas, é possível modificar o comportamento de um sistema de modo a diminuir a “intensidade” das contradições (DA SILVA FILHO, 2012).

Através de transformações lineares em um quadrado unitário no plano cartesiano (QUPC) (DA SILVA FILHO, 2012), com sua rotação em 45° no sentido anti-horário para o reticulado da LPA, é possível chegar à transformação apresentada na equação (29), conforme a figura 16.

$$T(X, Y) = (x - y, x + y - 1) \quad (29)$$

aplicados no eixo vertical, chamado de “Eixo dos Graus de Contradição”. O valor máximo de +1 significa que o estado lógico final da análise paraconsistente é inconsistente (T), ao passo que D_{CT} igual ao valor mínimo -1 significa que o estado lógico final da análise é indeterminado ou paracompleto ($\perp$). O ponto $\varepsilon\tau$, formado pela coordenada (D_C, D_{CT}) é o estado lógico paraconsistente (COELHO *et al.*, 2019).

Conforme Da Silva Filho (2012), um sistema decisório capaz de analisar dados, de conhecimento impreciso, pode ser mais robusto se apresentar um resultado final representando o valor da certeza puro, cujos efeitos das incertezas foram filtrados. Assim, o grau de certeza real (*real certainty degree* - D_{CR}) representa o grau de certeza livre dos efeitos da contradição e inconsistências das informações, podendo ser calculado a partir da Figura 17, como uma projeção do segmento de reta D , como a distância do ponto $\varepsilon\tau$ até o vértice V (ou F se D_C for menor que zero), sobre o eixo dos graus de certeza, a partir do próprio vértice V (ou F), conforme as equações (33) a (35) (DA SILVA FILHO, 2012).

Figura 17 – Limites de D .

Fonte: Adaptado de Da Silva Filho, Abe, Torres (2008, p. 59).

$$D = \sqrt{(1 - |D_C|)^2 + D_{CT}^2} \quad (33)$$

$$D = \begin{cases} 1 & D \geq 1 \\ D & 0 \leq D < 1 \end{cases} \quad (34)$$

$$D_{CR} = \begin{cases} 1 - D & D_C > 0 \\ D - 1 & D_C < 0 \\ 0 & D_C = 0 \end{cases} \quad (35)$$

Nota-se que o valor de D em (33) está sujeita à (34), conforme (34). Ou seja, há uma saturação do valor de D em 1. Isso significa que independentemente dos valores de $[\mu, \lambda]$ o valor máximo de D é [1], conforme indicado pelas linhas tracejadas de limites de D na figura 17. Logo, D_{CR} sempre será nulo para D maior ou igual a 1. Observa-se ainda que, segundo as equações (30) e (32), as saídas D_C e D_{CT} são lineares. Já a saída D_{CR} , que extrai os efeitos da contradição, é não linear, dada pela função de radiciação da equação (33), pela saturação apontada na equação (34) e pela inversão de sinal da equação (35).

O intervalo de certeza (φ) é definido como a faixa de valores de certeza em que o D_C pode variar sem ser limitado por D_{CT} e pode ser calculado conforme a equação (36), a partir do grau de contradição. Se o D_{CT} tende a inconsistente, φ é marcado como “+”; já se D_{CT} tende a indeterminado, φ é marcado como “-” (DA SILVA FILHO, 2012).

$$\varphi = (1 - |D_{CT}|) \quad (36)$$

Segundo Da Silva Filho (2012), o sinal de saída de um sistema paraconsistente de tratamento de incertezas fornecerá o resultado de grau de certeza resultante real (*resulting real certainty degree* - D_{Crr}) representado pelos valores (D_{CR} , $\varphi (\pm)$).

3.3 CÉLULAS E REDES NEURAIS ARTIFICIAIS PARACONSISTENTES

Define-se como nó de análise paraconsistente - NAP (*paraconsistent analysis node* - PAN) o algoritmo que contemple o conjunto de equações matemáticas da LPA2v e suas interpretações na forma de um bloco de programação (DA SILVA FILHO, 2012), cujo símbolo é apresentado na figura 18 (COELHO *et al.*, 2019).

A análise paraconsistente gera valores de D_C e D_{CT} entre [-1,1]. Para ser consistente com as entradas (μ, λ) e permitir a combinação de NAPs em aplicações práticas, as saídas D_C , D_{CT} e D_{CR} devem ser normalizadas e seus valores limitados a [0,1]. Assim, o grau de evidência resultante (*resulting evidence degree* - μ_E) é a saída normalizada de D_C (37), o grau de contradição resultante, (*resulting contradiction*

degree - μ_{ECT}) é a saída normalizada de D_{CT} (38) e o grau de evidência resultante real (*resulting real evidence degree* - μ_{ER}) é a saída normalizada de D_{CR} (39) (COELHO *et al.*, 2019; DA SILVA FILHO; ABE; TORRES, 2008).

Figura 18 – Símbolo do NAP.

Fonte: Adaptado de Coelho, *et al.* (2019, p. 115).

Já o intervalo de evidência resultante (φ_E) é definido como o valor normalizado do intervalo de certeza, dado pela equação (40) (DA SILVA FILHO; ABE; TORRES, 2008).

$$\mu_E = \frac{D_C + 1}{2} \quad (37)$$

$$\mu_{ECT} = \frac{D_{CT} + 1}{2} \quad (38)$$

$$\mu_{ER} = \frac{D_{CR} + 1}{2} \quad (39)$$

$$\varphi_E = (1 - |2\mu_{ECT} - 1|) \quad (40)$$

Redes de processamento de informação podem ser constituídas a partir da interligação de NAPs, recebendo o nome de rede de análise paraconsistente - RAP (*paraconsistent analysis network – PANet*) (COELHO *et al.*, 2019).

O NAP é o bloco básico para compor algoritmos mais complexos de LPA2v de modo a apresentar características similares a um neurônio. Esses algoritmos recebem o nome de células neurais artificiais paraconsistentes - CNAP (*paraconsistent artificial neural cell – PANC*) (ABE, 2004; DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010). A literatura apresenta uma série de CNAPs baseadas em aplicações da LPA2v e com

diferentes finalidades (ABE, 2004; DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010). A célula neural artificial paraconsistente padrão - CNAPp (*standard paraconsistent artificial neural cell – PANCS*) possui praticamente as mesmas fórmulas de um NAP, com a diferença de que podem fazer uso de algumas entradas adicionais com a capacidade de influenciar no resultado apresentado em sua saída. Na CNAPp, as entradas fator de tolerância à contradição (*contradiction tolerance factor - F_{tCT}*), fator de tolerância à certeza (*certainty tolerance factor - F_{tC}*), fator de tolerância à decisão (*decision tolerance factor - F_{tD}*) e fator de aprendizagem (*learning factor - F_L*) são valores externos de ajuste entre [0,1] (DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010). A figura 19 apresenta o símbolo da CNAP (CARVALHO JUNIOR *et al.*, 2018).

Figura 19 – Símbolos da CNAPp.

Fonte: Carvalho Junior, *et al.* (2018, p. 198).

O fator de ajuste F_{tC} altera os limites mínimo e máximo de μ_{ER} . O fator de ajuste F_{tCT} faz com que a saída μ_{ER} seja levada à indefinição (0.5), caso o valor calculado de μ_{ECT} seja maior que o valor de F_{tCT} . Já o fator F_{tD} é utilizado para se calcular os valores limites de verdade (Vl_V) e falsidade (Vl_F). Se a saída μ_{ER} for maior que Vl_V ela é levada ao estado verdadeiro (1), ao passo que se ela for menor que Vl_F ela é levada ao estado falso (0). Um exemplo é apresentado a seguir:

a) Algoritmo F_{tC}

$$V_{ICC} = \frac{1 - F_{tC}}{2}$$

$$V_{SCC} = \frac{1 + F_{tC}}{2}$$

Onde V_{ICC} e V_{SCC} são respectivamente os valores inferior e superior do controle da certeza.

Se:

$$V_{ICC} < \mu_E < V_{SCC}$$

Calcula-se μ_{ER}

Do contrário, faz-se μ_{ER} com saída indefinida: $\mu_{ER} = 0.5$.

Dados $\mu = 0.9$ e $\lambda = 0.28$.

Sem F_{tC} . Aplicando-se as equações (30) a (39), resultam em:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81; D_{CR} = 0.58 \text{ e } \mu_{ER} = 0.79.$$

Com $F_{tC} = 0.6$.

$$V_{ICC} = 0.2; V_{SCC} = 0.8.$$

Recalculando, tem-se que:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81 > V_{SCC}, \text{ logo: } \mu_{ER} = 0.5.$$

b) Algoritmo F_{tCT}

$$V_{ICCT} = \frac{1 - F_{tCT}}{2}$$

$$V_{SCCT} = \frac{1 + F_{tCT}}{2}$$

Onde V_{ICCT} e V_{SCCT} são respectivamente os valores inferior e superior do controle da contradição.

Se:

$$V_{SCCT} > \mu_{ECT} > V_{ICCT}$$

Calcula-se μ_{ER}

Do contrário, faz-se μ_{ER} com saída indefinida: $\mu_{ER} = 0.5$.

Dados $\mu = 0.9$ e $\lambda = 0.28$.

Sem F_{tCT} . Aplicando-se as equações (30) a (39), resultam em:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81; \mu_{ECT} = 0.59; D_{CR} = 0.58 \text{ e } \mu_{ER} = 0.79.$$

Com $F_{tCT} = 0.15$.

$$V_{ICCT} = 0.425; V_{ICCT} = 0.575.$$

Recalculando, tem-se que:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81; \mu_{ECT} = 0.59 > V_{SCCT}, \text{ logo: } \mu_{ER} = 0.5.$$

c) Algoritmo F_{tD}

$$Vl_F = \frac{1 - F_{tD}}{2}$$

$$Vl_V = \frac{1 + F_{tD}}{2}$$

Onde Vl_F e Vl_V são respectivamente os valores limite da falsidade e verdade.

Se:

$$\mu_E \geq Vl_V \rightarrow \mu_{ER} = 1$$

$$\mu_E \leq Vl_F \rightarrow \mu_{ER} = 0$$

Do contrário, faz-se μ_{ER} com saída indefinida: $\mu_{ER} = 0.5$.

Dados $\mu = 0.9$ e $\lambda = 0.28$.

Sem F_{tD} . Aplicando-se as equações (30) a (39), resultam em:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81; \mu_{ECT} = 0.59; D_{CR} = 0.58 \text{ e } \mu_{ER} = 0.79.$$

Com $F_{tD} = 0.7$.

$$Vl_F = 0.85; Vl_V = 0.15.$$

Aplicando-se as equações (30) a (39), resultam em:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81; Vl_F \leq \mu_E \leq Vl_V, \text{ logo: } \mu_{ER} = 0.5.$$

Com $F_{tD} = 0.5$.

$$Vl_F = 0.75; Vl_V = 0.25.$$

Recalculando, tem-se que:

$$D_C = 0.62; D_{CT} = 0.18; \mu_E = 0.81; \mu_E \geq Vl_V, \text{ logo: } \mu_{ER} = 1.$$

Dados $\mu = 0.7$ e $\lambda = 0.9$.

Sem F_{tD} . Aplicando-se as equações (30) a (39), resultam em:

$$D_C = -0.2; D_{CT} = 0.6; \mu_E = 0.4; \mu_{ECT} = 0.8; D_{CR} = 0.0 \text{ e } \mu_{ER} = 0.5.$$

Com $F_{tD} = 0.1$.

$$Vl_F = 0.55; Vl_V = 0.45.$$

Aplicando-se as equações (30) a (39), resultam em:

$$D_C = -0.2; D_{CT} = 0.6; \mu_E = 0.4; \mu_E < V_L, \text{ logo: } \mu_{ER} = 0.$$

Da Silva Filho; Abe; Torres (2008) apresenta a descrição detalhada dos algoritmos de fatores de ajuste (F_{tCT}, F_{tC}, F_{tD}). Observa-se, no entanto, que a CNAP não aprende como uma rede neural clássica, mas apenas configuram-se limiares de transição dos resultados de saída. Além disso, a forma como se encontram os valores ideais de (F_{tCT}, F_{tC}, F_{tD}) não é apresentada na literatura, o que resulta em um exaustivo esforço de tentativa e erro para treinar a CNAPp.

Na literatura, são propostas diversas células neurais paraconsistentes (DA SILVA FILHO; ABE; TORRES, 2008; DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010). Uma CNAP muito utilizada é a célula neural artificial paraconsistente de aprendizagem - CNAPap (*paraconsistent artificial neural cell of learning* - PANCL), derivada da CNAPp. Na CNAPap a saída atual $\mu_{E(k)}$ é conectada com a entrada $\mu_{2(k)}$, conforme apresentado na Figura 20, que será complementada para se obter o $\lambda_{(k)}$, conforme a equação (31), para calcular o próximo valor de $\mu_{E(k+1)}$ (DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010). O valor futuro de μ_E pode ser obtido conforme a equação (41), onde k é o número da amostra atual presente nas entradas da célula e o fator F_L é aplicado sobre D_C (DE CARVALHO JUNIOR *et al.*, 2018).

Figura 20 – CNAPap.

Fonte: Carvalho Junior, *et al.* (2018, p. 198).

$$\mu_{E(k+1)} = \frac{\{(D_{C(k)})F_L + 1\}}{2} = \frac{\{(\mu_{(k)} - \lambda_{(k)})F_L + 1\}}{2}$$

$$\mu_{E(k+1)} = \frac{\{(\mu_{(k)} - (1 - \mu_{E(k)}))F_L + 1\}}{2} \quad (41)$$

Na literatura (DA SILVA FILHO; ABE; TORRES, 2008), também é possível encontrar a equação (42) para representar a saída μ_E da CNAPap. Nesse caso o fator F_L é aplicado apenas sobre a entrada λ .

$$\mu_{E(k+1)} = \frac{\{\mu_{(k)} - \lambda_{(k)}F_L + 1\}}{2} = \frac{\{\mu_{(k)} - (1 - \mu_{E(k)})F_L + 1\}}{2} \quad (42)$$

A CNAPap recebe esse nome pois a saída μ_E tende a seguir, ou “aprender”, o valor aplicado na entrada μ após um certo número de iterações (DA SILVA FILHO; ABE; TORRES, 2008).

Conforme apresentado na figura 21, o valor verdade 1 (um) apresentado na entrada μ é apresentado na saída μ_E após um certo número de iterações, que depende do fator de aprendizagem (F_L) aplicado, onde n é o número da amostra e T_s o tempo de amostragem. Quando F_L é máximo (um), a CNAPap necessita de 10 iterações para refletir a entrada μ na saída μ_E . Ainda conforme apresentado na figura 21, a CNAPap apresenta o comportamento de um integrador (DE CARVALHO JUNIOR *et al.*, 2018; DA SILVA FILHO; ABE; TORRES, 2008).

Figura 21 – Saída μ_E para diferentes valores de F_L .

Segundo Da Silva Filho *et al.* (2010) a CNAPap pode se comportar como uma memória em redes de células paraconsistentes e é muito importante e útil para análise e processamento de sinais. Além disso, Da Silva Filho *et al.* (2016) demonstraram que uma rede de CNAPap em cascata pode ser utilizada como extrator de média móvel. Carvalho Junior *et al.* (2018) apresentou estudos que mostram que a CNAPap pode ser utilizada como filtro passa baixas.

Figura 22 – CNAPap para diferentes valores de μ e F_L , equação (34) (a). CNAPap para diferentes valores de μ e F_L , equação (35) (b).

A CNAPap apresenta uma limitação, conforme sugerido por (MINICZ; *et al.*, 2014) e demonstrado em (DE CARVALHO JUNIOR; *et al.*, 2020) que ocorre quando valores diferentes de 1 (um) são apresentados na entrada μ , a convergência na saída somente ocorre se o F_L for de valor unitário. A não convergência ocorre seja utilizando

a equação (41) ou (42), conforme as simulações apresentadas nas figuras 22a e 22b, sendo esta condição utilizada geralmente nas aplicações envolvendo CNAPap (DA SILVA FILHO; ABE; TORRES, 2008).

De modo a solucionar o problema de convergência da CNAPap para diferentes valores de μ e F_L , Minicz, *et al.* (2014) propôs uma nova célula de aprendizagem, chamada célula neural artificial paraconsistente de aprendizagem por extração da contradição - CNAPap_{CTX} (*paraconsistent artificial neural cell of learning by contradiction extraction* – PANCL_{CTX}), cuja equação é apresentada na equação (43). A figura 23 apresenta os resultados simulados para a saída μ_E , seguindo a equação (43) e a figura 24 para a saída μ_{ECT} , de acordo com a equação (38).

$$\mu_{E(k+1)} = \mu_{E(k)} + D_{CT} F_L \quad (43)$$

Figura 23 – Saída μ_E da CNAPap_{CTX} para diferentes valores de μ e F_L .

Figura 24 – Saída μ_{ECT} da CNAPap_{CTX} para diferentes valores de μ e F_L .

O símbolo da CNAPap_{CTX} é o mesmo que o da CNAPap. De acordo com a figura 23, para a saída μ_E , e figura 24 para a saída μ_{ECT} , percebe-se que a CNAPap_{CTX} converge de forma correta para qualquer valor de μ e F_L , além de funcionar melhor como integrador e derivador. De Carvalho Jr (DE CARVALHO JUNIOR; *et al*, 2020) apresenta estudos CNAPap_{CTX} como filtro e redução ativa de ruído, dentro de determinadas condições.

A equação (43) pode ser reescrita, substituindo-se D_{CT} pela equação (32) apresentada anteriormente (DE CARVALHO JUNIOR; *et al*, 2020). Com isso, tem-se:

$$\begin{aligned}
 \mu_{E(k+1)} &= \mu_{E(k)} + (\mu_{(k)} + \lambda_{(k)} - 1)F_L \\
 \mu_{E(k+1)} &= \mu_{E(k)} + (\mu_{(k)} + \lambda_{(k)} - 1)F_L \\
 \mu_{E(k+1)} &= \mu_{E(k)} + (\mu_{(k)} + (1 - \mu_{E(k)}) - 1)F_L \\
 \mu_{E(k+1)} &= \mu_{(k)}F_L + \mu_{E(k)}(1 - F_L)
 \end{aligned} \tag{43}$$

Comparando-se a equação (43) reorganizada com a equação (41), nota-se que na primeira a saída atrasada μ_E é multiplicada pelo complemento de F_L ($1 - F_L$), enquanto que na segunda ela é multiplicada pelo próprio valor de F_L , além de aparecer o termo $(1 - F_L)$ isolado.

Já comparando-se a equação (43) reorganizada com a equação (42), nota-se que na primeira a entrada μ é multiplicada por F_L , enquanto que na segunda ela não é multiplicada por nenhum fator. Além disso, na primeira a saída atrasada μ_E é multiplicada pelo complemento de F_L ($1 - F_L$), enquanto que na segunda ela é multiplicada pelo próprio valor de F_L , além de também aparecer o termo $(1 - F_L)$ isolado.

3.4 Neurônio e Rede Neural Paraconsistente

Da Silva Filho *et al.* (DA SILVA FILHO; ABE; TORRES, 2008) apresentam o neurônio paraconsistente ou paraperceptron de 2 entradas, constituído pela interligação de uma CNAPp com uma CNAPap, conforme apresentado na figura 25.

Figura 25 – Neurônio Paraconsistente (Paraperceptron) de 2 entradas.

Fonte: Adaptado de Da Silva Filho, Abe, Torres (2008, p. 253).

Como já apresentado anteriormente, os fatores Ft_C e Ft_{CT} alteram limiares de transição da saída da primeira célula, enquanto que o fator F_L introduz um atraso na saída da segunda célula.

Para um número maior de entradas, é sugerida a composição de células conforme a figura 26. A interligação de células neurais paraconsistentes formam as redes neurais artificiais paraconsistentes - RNAP (*paraconsistente artificial neural network – PANN*) (DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010).

Figura 26 – Neurônio Paraconsistente (Paraperceptron) de 8 entradas.

Fonte: Adaptado de Da Silva Filho, Abe, Torres (2008, p. 255).

Dos Santos; *et al.* (2016) apresenta um exemplo de RNAP utilizada no estudo de padrões mutacionais do subtipo F das cepas virais do HIV-1 na terapia antirretroviral, reproduzida na figura 27.

Figura 27 – Exemplo de RNAP.

Fonte: Dos Santos, *et al.* (2016, p. 7).

A RNAP consiste essencialmente de uma rede de CNAPp interligadas, com 3 entradas e 2 saídas e consiste em um sistema neural paraconsistente de tratamento de contradições. Nota-se que não há atribuição de pesos nas entradas e entre as células, mas sim limiares de operação, ou seleção, da saída em função de F_{IC} e F_{ICT} . Esse tipo de RNAP funciona como classificador de padrões, sendo que o sinal apresentado na entrada percorrerá determinados caminhos dentro da rede, em razão de seleção de máximos e mínimos realizados por células específicas.

3.5 SUMÁRIO DO CAPÍTULO

A lógica paraconsistente anotada evidencial, também conhecida como LPA2v, por revogar o princípio da contradição, permite o tratamento de situações onde a informação pode ser incerta, ou contaminada por ruídos e imprecisões. O uso de 2 valores de evidências permite um maior grau de representatividade do estado lógico paraconsistente. Devido a suas características, a LPA2v tem sido empregada por sistemas especialistas na tomada de decisão.

A utilização do grau de certeza real (D_{CR}) ao invés do grau de certeza (D_C), na análise da LPA2v, permite a obtenção de resultados livre dos efeitos da contradição e inconsistências das informações.

Redes neurais artificiais paraconsistentes são propostas na literatura, sendo bastante utilizados o NAP, a CNAPp e a CNAPap. Entretanto a RNAP não “aprende”, no conceito clássico de redes neurais com aplicação de pesos, mas utiliza em sua construção, fatores de ajuste, ou limiares de transição (F_{tCT} , F_{tC} , F_{tD}). Além disso, as CNAPap ou CNAPap_{CTX}, apesar de serem chamadas de aprendizagem, elas na verdade apresentam efeito acumulador, ou “memória”, ajustável pelo (F_L). Assim sendo, as RNAP sugeridas na literatura são utilizadas na classificação de sinais. No próximo capítulo é apresentada, de forma inovadora, uma proposição de rede neural que faz uso da LPA2v como função de ativação de neurônios treináveis, para o aprendizado e aproximação de funções. De forma a distinguir das RNAPs presentes na literatura, as redes neurais propostas nesta pesquisa serão denominadas de rede neural paraconsistente (RNP) e rede neural paraconsistente recorrente (RNPR). Com isso, a RNP proposta amplia o potencial de aplicações da LPA2v, podendo ser empregada na modelagem, identificação e controle de sistemas dinâmicos.

4. SISTEMA DE ESTUDO

Este capítulo apresenta a proposta desta tese para a construção de um neurônio LPA2v e da rede neural paraconsistente para aproximação de funções.

4.1. NEURÔNIO PARACONSISTENTE PROPOSTO

Conforme discutido na seção 3.4, a literatura apresenta o neurônio paraconsistente, chamado de paraperceptron, como uma composição de células padrão (CNAPp) e de aprendizagem (CNAPap). Além disso, o paraperceptron apresentado na literatura via de regra faz uso da saída de grau de evidência resultante (μ_E) (DA SILVA FILHO; ABE; TORRES, 2008). Neste trabalho, propõe-se utilizar como função de ativação do neurônio paraconsistente, ou neurônio LPA2v, a saída do grau de evidência resultante real (μ_{ER}), por ser não linear (função de D_{CR} e D), derivável, saturável entre $[0,1]$ e que apresenta uma curvatura sigmoide, conforme apresentado na figura 28. Uma simulação considerando-se a entrada μ entre $[0,1]$, e diferentes valores fixos aplicados na entrada λ , operando como um “bias” ou “offset” é apresentada na figura 28a. Já na figura 28b, a entrada μ tem valores fixos enquanto que a entrada λ , é variável entre $[0,1]$.

Figura 28 – Saídas μ_E e μ_{ER} para μ variável e λ com 3 valores fixos (a). Saídas μ_E e μ_{ER} para λ variável e μ com 3 valores fixos (b).

A figura 29 repete o exemplo da figura 28a, porém com a aplicação de uma normalização que expande os valores de μ_E e μ_{ER} para a faixa entre $[0,1]$. A equação (44) apresenta o processo de normalização aplicado.

Figura 29 – Saídas μ_E e μ_{ER} normalizadas entre $[0,1]$, para μ variável e λ fixo.

$$\mu_{ER} = \frac{\mu_{ER} - \min(\mu_{ER})}{\max(\mu_{ER}) - \min(\mu_{ER})} \quad (44)$$

Se uma não linearidade mais acentuada é exigida, pode-se propor outras equações dentro dos limites do Lattice LPA2v, como por exemplo μ_{ER}^n , sendo n número inteiro (1, 2, 3, ...) e corrigir a amplitude pico a pico via normalizador na saída. Outra

opção é adicionalmente aplicar uma função não linear aos sinais de entrada antes de atravessarem o neurônio paraconsistente, conforme a equação (45), onde x é o valor do sinal aplicado à entrada μ , da CNAPp. A não linearidade adicional aplicada às entradas μ e λ antes da análise LPA2v foi proposto em (DA SILVA FILHO; ABE; TORRES, 2008).

$$\mu = \begin{cases} 0 & \text{se } x < x_{min} \\ 2 \left(\frac{x - x_{min}}{x_{max} - x_{min}} \right)^2 & \text{se } x_{min} \leq x \leq \left(\frac{x_{max} + x_{min}}{2} \right) \\ 1 - 2 \left(\frac{x - x_{max}}{x_{max} - x_{min}} \right)^2 & \text{se } \left(\frac{x_{max} + x_{min}}{2} \right) \leq x \leq x_{max} \\ 1 & \text{se } x > x_{max} \end{cases} \quad (45)$$

Neste trabalho, propõe-se que simplesmente um NAP ou CNAPp seja suficiente para a construção do neurônio paraconsistente, neurônio LPA2v ou função de ativação LPA2v, com 2 entradas treináveis mediante pesos, conforme figura 30.

Figura 30 – Neurônio LPA2v proposto de 2 entradas.

Observa-se que, diferentemente do registrado na literatura para RNAP, o neurônio LPA2v utiliza a saída μ_{ER} , não linear e que extrai os efeitos da contradição, ao invés da saída linear μ_E . Além disso, pesos e *bias* são aplicados aos sinais antes de serem apresentados às entradas μ e λ , limitados a faixa entre [0,1].

De modo a atender as restrições da lógica, blocos limitadores entre [0,1] são aplicados nas entradas μ e λ . Os valores w_1 e w_2 são os pesos aplicados na entrada do neurônio paraconsistente.

Há uma similaridade entre o neurônio LPA2v da figura 30 com o perceptron apresentado na figura 8. A função de ativação do neurônio LPA2v é a sigmoide de saída do grau de evidência resultante real (μ_{ER}), recebendo o nome de função

Sigmoide_{LPA2v}. Já a saída do grau de evidência resultante (μ_E), de uma CNAPp simples ou um “meio” neurônio LPA2v, funciona como uma função de ativação linear, chamada de Linear_{LPA2v}.

Para a concepção de um neurônio paraconsistente com número de entradas maior do que 2, uma combinação de CNAPp conectadas com saída linear μ_E pode ser empregada. A figura 31 apresenta um exemplo de neurônio paraconsistente de 4 entradas, comparado ao neurônio padrão. De modo a melhorar a visualização do conceito empregado, uma simbologia simplificada de célula é adotada. Na figura 30b, o bloco somador de entrada pode ser simplificado em uma equação de soma simples, conforme apresentado na sequência da equação (46), resultando em uma equação equivalente do somador de entrada do neurônio paraconsistente. A saída do somador (μ_{E3}) é aplicada à entrada (μ_4) da célula responsável pelo sigmoide e a saída μ_{ER4} é calculada conforme as equações (30 a (35) e (39) apresentadas no capítulo 3.

Figura 31 – Neurônio padrão (a) e Neurônio LPA2v com 4 entradas (b).

$$\begin{aligned}
\mu_{E1} &= \frac{\mu_1 - \lambda_1 + 1}{2} \\
\mu_{E2} &= \frac{\mu_2 - \lambda_2 + 1}{2} \\
\mu_{E3} &= \frac{\mu_3 - \lambda_3}{2} = \left(\frac{\frac{\mu_1 - \lambda_1 + 1}{2} - \left(\frac{\mu_2 - \lambda_2 + 1}{2} \right) + 1}{2} \right) \\
\mu_{E3} &= \left(\frac{0.5\mu_1 - 0.5\lambda_1 + 0.5 - 0.5\mu_2 + 0.5\lambda_2 - 0.5 + 1}{2} \right) \\
\mu_{E3} &= \left(\frac{0.5\mu_1 - 0.5\lambda_1 - 0.5\mu_2 + 0.5\lambda_2 + 1}{2} \right) \\
\mu_{E3} &= \frac{1}{4}(\mu_1 - \lambda_1 - \mu_2 + \lambda_2) + \frac{1}{2}
\end{aligned} \tag{46}$$

A equação (46) mostra um fator multiplicador de 1/4 é aplicado a todas as entradas $(\mu_1, \lambda_1, \mu_2, \lambda_2)$, mais um valor médio de 1/2. Para que um único fator multiplicador seja aplicado a todas as entradas igualmente, sugere-se trabalhar com progressão geométrica de 2 (2, 4, 8, 16, ...). Caso o número de entradas seja ímpar, pode-se adotar o próximo valor da progressão e considerar as entradas excedentes com valor zero. Exemplo: Para 7 entradas, adota-se uma rede de CNAPp de 3 camadas, 7 células, resultando na equação (47), com 8 entradas e fazendo-se uma delas igual a zero.

$$\begin{aligned}
\mu_{E7} &= \left[\frac{\left(\frac{1}{4}(\mu_1 - \lambda_1 - \mu_2 + \lambda_2) + \frac{1}{2} \right) - \left(\frac{1}{4}(\mu_3 - \lambda_3 - \mu_4 + \lambda_4) + \frac{1}{2} \right) + 1}{2} \right] \\
\mu_{E7} &= \left[\frac{\left(\frac{1}{4}(\mu_1 - \lambda_1 - \mu_2 + \lambda_2 - \mu_3 + \lambda_3 + \mu_4 - \lambda_4) \right) + 1}{2} \right] \\
\mu_{E7} &= \left(\frac{1}{8}(\mu_1 - \lambda_1 - \mu_2 + \lambda_2 - \mu_3 + \lambda_3 + \mu_4 - \lambda_4) \right) + \frac{1}{2}
\end{aligned} \tag{47}$$

Na figura 31b há 2 camadas de células no somador, sendo que a soma das entradas $(\mu_1, \lambda_1, \mu_2, \lambda_2)$, resulta em um fator multiplicador de 1/4. Na equação (47), para

até 8 entradas, utilizam-se 3 camadas de CNAPp, resultando em um fator multiplicador de $1/8$. Assim, pode-se considerar no cálculo do somador este fator de multiplicador e aplica-lo juntamente com o valor do peso a ser utilizado na entrada. O fator multiplicador depende do número de entradas e, conseqüentemente, camadas do bloco somador do neurônio LPA2v, conforme a equação (48), onde F é o fator multiplicador e N o número de camadas de células do bloco soma_{LPA2v}.

$$F = 1/2^N \quad (48)$$

Nota-se, que as equações (46) e (47), resultam em uma soma, ainda que algumas das entradas estejam com sinal negativo. Isso pode ser tratado no bloco normalizador ou através dos pesos aprendidos e aplicados às entradas do neurônio, após a fase de treinamento.

Com a topologia proposta neste trabalho, o neurônio paraconsistente é constituído a partir de um bloco somador (Soma_{LPA2v}), formado por uma rede de células CNAPp de saída linear (μ_E) e uma CNAPp com saída (μ_{ER}), como bloco de função de ativação sigmoide utilizada (Sigmoide_{LPA2v}), apresentando uma estrutura similar ao neurônio perceptron apresentado na figura 7 da seção 2.3, porém atendendo aos fundamentos da LPA2v. Pesos são aplicados às entradas, de modo que o neurônio paraconsistente possa ser utilizado na aproximação de funções.

O uso combinado de vários neurônios LPA2v, conforme proposto neste trabalho, forma uma rede neural paraconsistente – RNP (*paraconsistent neural network* – PNN), que aprende através de valores de pesos e *bias* aplicados nas entradas μ e λ dos neurônios, em contraste à RNAP proposta na literatura que apresenta saída baseada em transições de limiares de ajuste nas células (F_{tCT} , F_{tC} , F_{tD}) (DA SILVA FILHO; ABE; TORRES, 2008; DA SILVA FILHO; LAMBERT-TORRES; ABE, 2010).

4.2. CÁLCULO DE DERIVADAS PARA O NEURÔNIO LPA2v PROPOSTO

Existem vários métodos de aprendizado do neurônio artificial. Para fins deste trabalho, o processo de aprendizagem do neurônio proposto é mesmo de *backpropagation* das redes neurais clássicas.

Para o cálculo do impacto ou influência que cada entrada e seu peso representam no erro ou diferença entre a saída do neurônio paraconsistente e a saída

desejada, é necessário a derivada da função de saída em função da entrada utilizada na última CNAPp. A última célula do neurônio paraconsistente, no caso da figura 28a a CNAPp4, possui 2 entradas e uma saída. Assim, pode-se calcular as derivadas $(\partial\mu_{ER}/\partial\mu)$ e $(\partial\mu_{ER}/\partial\lambda)$, conforme as equações (49) e (50) para D_{CR} maior que zero, equações (51) e (52) para D_{CR} menor que zero e equações (53) e (54) para D_{CR} igual a zero. A equação (55) apresenta a derivada para a saída linear (μ_E), para cada CNAPp utilizada na somaLPA2v.

$$\mu_{ER} = \frac{D_{CR} + 1}{2}$$

a) Para $D_C > 0; \mu > \lambda$:

$$\mu_{ER} = \frac{\left(1 - \sqrt{(1 - |D_C|)^2 + D_{CT}^2}\right) + 1}{2}$$

$$\mu_{ER} = \frac{(2 - \sqrt{(1 - |\mu - \lambda|)^2 + (\mu + \lambda - 1)^2})}{2}$$

$$\mu_{ER} = \frac{(2 - \sqrt{(1 - 2\mu + 2\lambda - 2\mu\lambda + \mu^2 + \lambda^2) + (\mu^2 + \lambda^2 + 2\mu\lambda - 2\mu - 2\lambda + 1)})}{2}$$

$$\mu_{ER} = \frac{(2 - \sqrt{2\mu^2 + 2\lambda^2 - 4\mu + 2})}{2}$$

$$\frac{\partial\mu_{ER}}{\partial\mu} = -\frac{1}{2} \frac{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{-0.5}(4\mu - 4)}{2} = \frac{-(4\mu - 4)}{4(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}}$$

$$\frac{\partial\mu_{ER}}{\partial\mu} = \frac{(1 - \mu)}{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}} \quad (49)$$

$$\frac{\partial\mu_{ER}}{\partial\lambda} = -\frac{1}{2} \frac{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{-0.5}(4\lambda)}{2}$$

$$\frac{\partial\mu_{ER}}{\partial\lambda} = -\frac{\lambda}{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}} \quad (50)$$

b) Para $D_C < 0; \mu < \lambda$:

$$\mu_{ER} = \frac{\left(\sqrt{(1 - |D_C|)^2 + D_{CT}^2} - 1\right) + 1}{2}$$

$$\mu_{ER} = \frac{\sqrt{(1 - |\mu - \lambda|)^2 + (\mu + \lambda - 1)^2}}{2}$$

$$\mu_{ER} = \frac{\left(\sqrt{(1 + \mu - \lambda)^2 + (\mu + \lambda - 1)^2}\right)}{2}$$

$$\mu_{ER} = \frac{(\sqrt{(1 + 2\mu - 2\lambda - 2\mu\lambda + \mu^2 + \lambda^2)} + (\mu^2 + \lambda^2 + 2\mu\lambda - 2\mu - 2\lambda + 1))}{2}$$

$$\mu_{ER} = \frac{(\sqrt{2\mu^2 + 2\lambda^2 - 4\lambda + 2})}{2}$$

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{1}{2} \frac{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{-0.5}(4\mu)}{2} = \frac{4\mu}{4(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}}$$

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{\mu}{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}} \quad (51)$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = \frac{1}{2} \frac{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{-0.5}(4\lambda - 4)}{2} = \frac{4(\lambda - 1)}{4(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}}$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = \frac{\lambda - 1}{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}} \quad (52)$$

c) Para $D_C = 0 \Rightarrow \mu = \lambda$:

$\mu_{ER} = 0.5$ (indefinição).

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{\left(\frac{(1 - \mu)}{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}}\right) + \left(\frac{\mu}{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}}\right)}{2}$$

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{1}{2(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}} \quad (53)$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = \frac{\left(-\frac{\lambda}{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}} \right) + \left(\frac{\lambda - 1}{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}} \right)}{2}$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = -\frac{1}{2(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}} \quad (54)$$

Nota: $D \geq 1$, implica em $\mu_{ER} = 0.5$, constante, resultando em uma derivada nula. Para o treinamento do neurônio, propõe-se igualar as derivadas para $D_C > 0$ e $D_C < 0$, conforme as equações (54) e (55). A figura 32 apresenta uma simulação desta situação. Na figura, μ varia na faixa $[0,1]$ e λ está fixo em 0.5. Apenas a derivada $\partial \mu_{ER} / \partial \mu$ é calculada. Na condição de $\mu_{ER} = 0$, a derivada $\partial \mu_{ER} / \partial \mu$ (a) é calculada conforme a equação (53), ao passo que $\partial \mu_{ER} / \partial \mu$ (b) é nula. Observa-se que a equação (53) permite uma transição mais coerente no cálculo da derivada do que o vale causado pela derivada nula. A mesma análise pode ser observada para outros valores de λ e também para a derivada $\partial \mu_{ER} / \partial \lambda$, pela equação (54).

Figura 32 – Comportamento de $\partial \mu_{ER} / \partial \mu$ pela média das equações (49) e (51) ou nulo na condição de $D \geq 1$.

d) Derivadas para saída CNApp com saída linear (μ_E):

$$\frac{\partial \mu_E}{\partial \mu} = 0.5; \quad \frac{\partial \mu_E}{\partial \lambda} = -0.5 \quad (55)$$

Propõe-se neste trabalho que o neurônio paraconsistente possa ser construído apenas fazendo uso de CNAPp. Pesos e *bias* devem ser aplicados com RNP de maneira semelhante à RNA padrão.

4.3 APRENDIZAGEM DO NEURÔNIO LPA2v PROPOSTO

O aprendizado do neurônio paraconsistente proposto neste trabalho segue o método de *backpropagation*, adotado para redes neurais:

- a) Seleciona-se um conjunto de amostras dos sinais de entrada e saída desejada;
- b) Identifica-se os valores máximo e mínimo dos sinais de entrada (pico a pico) para o cálculo do ganho do normalizador de entrada, conforme equação (56);

$$G_I = \frac{1}{round(\max(\max(Entradas) - \min(Entradas)) * 2)} \quad (56)$$

- c) Faz-se o normalizador da saída com ganho inverso ao adotado para as entradas, considerando-se o número (N) de camadas de células do SomaLPA2v, conforme a equação (57);

$$G_O = \frac{1}{G_I} 2^N \quad (57)$$

- d) Escolhe-se aleatoriamente valores do conjunto de amostras, representando uma porcentagem do total do conjunto;
- e) Atribui-se pesos aleatórios iniciais às entradas do neurônio;
- f) Em um loop (época), para o conjunto de amostras aleatórias, executa-se a equação do neurônio na direção *forward*;
- g) Compara-se o erro da saída normalizada do neurônio com a saída desejada;
- h) Ajustam-se os pesos para cada set de entradas/saída de modo a diminuir o erro, segundo um fator de aprendizagem adotado para o neurônio;
- i) Após todas as amostras aleatórias do conjunto terem passado pelo neurônio e com os últimos pesos obtidos, inicia-se um novo loop;
- j) Repete-se o processo, a partir de (f) até que o erro médio quadrático (*mean square error – MSE*) entre a saída do neurônio e a saída desejada seja mínimo.

Para se obter o grau de influência e correção de um peso sobre o erro total, usa-se a regra da cadeia para derivadas, conforme a equação (58), onde $\partial a / \partial n$ é obtido através das equações (49)-(55). O $Erro_{total}$ é calculado de acordo com a equação de erro médio quadrático (MSE) dada pela equação (59), onde S_{Target} é a saída desejada e S_{PAL} a saída do neurônio. A mesma metodologia é adotada para uma RNP, observando-se a arquitetura da rede, a quantidade de neurônios e camadas.

$$\frac{\partial Erro_{total}}{\partial w_1} = \frac{\partial Erro_{total}}{\partial a} * \frac{\partial a}{\partial n} * \frac{\partial n}{\partial w_1} \quad (58)$$

$$Erro_{total} = \frac{1}{2} (S_{Target} - S_{PAL})^2 \quad (59)$$

A representação das sequências para a frente (*forward*) e para trás (*backward*) do neurônio paraconsistente é apresentada na figura 33.

Figura 33 – Método *backpropagation* para o neurônio paraconsistente.

4.4 REDE NEURAL PARACONSISTENTE COM CNAPap_{CTX}

Conforme apresentado na seção 3.3, a CNAPap e a CNAPap_{CTX} introduzem um atraso na saída em relação ao sinal aplicado à entrada. Foi observado que se utilizado F_L diferente de (1), a saída da CNAPap não converge com o valor de entrada. Se for

utilizado F_L fixo em (1), a saída da CNAPap precisará de 10 iterações (ou amostragens) para refletir, ou “aprender”, o valor da entrada (vide figura 21). A CNAPap_{CTX} pode substituir a CNAPap com vantagens pois permite que o atraso seja controlado por F_L , independentemente do valor presente na entrada do grau de evidência favorável (μ). Além disso, se a informação aplicada na entrada for mantida durante o tempo de atraso, a saída converge para este valor. A Tabela 1 apresenta o atraso a CNAPap_{CTX} para diferentes valores de F_L , a partir da equação (43) apresentada no capítulo anterior.

O uso combinado de CNAPp e CNAPap_{CTX}, devido ao atraso, ou efeito memória do segundo, permite o *design* de redes neurais paraconsistentes recorrentes - RNPR (*recurrent paraconsistent neural network* - *RPNN*). A CNAPap_{CTX}, pode ser empregada para gerar um atraso nas entradas da RNP, na saída, na interligação entre camadas ou até mesmo no caminho de realimentação da saída quando aplicada de volta à entrada.

Tabela 1 – Atraso ou iterações da CNAPap_{CTX} em função de F_L .

F_L	Iterações
1.00	1
0.90	4
0.75	6
0.50	11
0.25	26
0.10	68
0.00	Indefinido (saída congelada)

4.5 CNAPap_{CTX} COMO INTEGRADOR, DERIVADOR E FILTRO

A equação (60) apresenta o desdobramento da saída (μ_E) da CNAPap_{CTX}, na forma de equação das diferenças. Nota-se que o termo ($\mu[n] - \mu_E[n-1]$), correspondendo a entrada atual menos saída anterior, multiplicado por F_L denota o efeito integrador da saída μ_E atual.

$$D_{CT} = \mu + \lambda - 1$$

$$\begin{aligned}
\mu_{Ek} &= \mu_{E(k-1)} + D_{CTk} * F_L \\
\mu_E[n] &= \mu_E[n-1] + (\mu[n] + \lambda[n] - 1)F_L \\
\mu_E[n] &= \mu_E[n-1] + (\mu[n] + (1 - \mu_E[n-1] - 1)F_L \\
\mu_E[n] &= \mu_E[n-1] + (\mu[n] - \mu_E[n-1])F_L
\end{aligned} \tag{60}$$

A equação (61) apresenta a função de transferência no plano Z da CNAPap_{CTX}, quando a saída (μ_E) é utilizada.

$$\begin{aligned}
\mu_E(z) &= z^{-1}\mu_E(z) + F_L\mu(z) - F_Lz^{-1}\mu_E(z) \\
\mu_E(z)(1 - (F_L - 1)z^{-1}) &= F_L\mu(z) \\
\frac{\mu_E(z)}{\mu(z)} &= \frac{F_L}{1 - (1 - F_L)z^{-1}}
\end{aligned} \tag{61}$$

Aplicando-se o mesmo conceito, pode-se obter a equação das diferenças e a função de transferência no plano Z para a saída (μ_{ECT}) da CNAPap_{CTX}.

$$\begin{aligned}
\mu_{ECT} &= \frac{D_{CT} + 1}{2} = \frac{(\mu + \lambda - 1) + 1}{2} = \frac{\mu + \lambda}{2} \\
\mu_{ECT}[n] &= \frac{\mu[n] + \lambda[n]}{2} \\
\mu_{ECT}[n] &= \frac{\mu[n] + (1 - \mu_E[n-1])}{2} = \frac{1}{2}(\mu[n] - \mu_E[n-1]) + \frac{1}{2}
\end{aligned}$$

Utilizando-se um termo auxiliar para simplificação, de μ'_{ECT} , e aplicando-se a transformada Z, tem-se a equação (62) como função de transferência, quando a saída μ_{ECT} é utilizada:

$$\begin{aligned}
\mu'_{ECT}[n] &= \frac{1}{2}\mu[n] - \frac{1}{2}\mu_E[n-1] \rightarrow \mu'_{ECT}[n] = \mu_{ECT}[n] - \frac{1}{2} \\
\mu'_{ECT}[n] &= \frac{1}{2}(\mu[n] - \mu_E[n-1]) \\
\mu'_{ECT}(z) &= \frac{1}{2}(\mu(z) - z^{-1}\mu_E(z)) = \frac{1}{2}\left(\mu(z) - z^{-1}\left(\frac{F_L}{1 - (1 - F_L)z^{-1}}\right)\mu(z)\right)
\end{aligned}$$

$$\mu'_{ECT}(z) = \frac{1}{2}\mu(z) \left(1 - z^{-1} \left(\frac{F_L}{1 - (1 - F_L)z^{-1}} \right) \right)$$

$$\frac{\mu'_{ECT}(z)}{\mu(z)} = \frac{1}{2} \left(1 - \left(\frac{F_L z^{-1}}{1 - (1 - F_L)z^{-1}} \right) \right)$$

$$\frac{\mu'_{ECT}(z)}{\mu(z)} = \frac{1}{2} \left(\frac{1 - (1 - F_L)z^{-1} - F_L z^{-1}}{1 - (1 - F_L)z^{-1}} \right) = \frac{1}{2} \left(\frac{1 + (F_L - 1 - F_L)z^{-1}}{1 - (1 - F_L)z^{-1}} \right)$$

$$\frac{\mu'_{ECT}(z)}{\mu(z)} = \frac{1}{2} \left(\frac{1 - z^{-1}}{1 - (1 - F_L)z^{-1}} \right) = \frac{1}{2} \left(\frac{z - 1}{z - (1 - F_L)} \right) \quad (62)$$

$$\frac{\mu_{ECT}(z)}{\mu(z)} = \frac{1}{2} \left(\frac{z - 1}{z - (1 - F_L)} \right) + \frac{1}{2} \quad (63)$$

Partindo-se da equação (62) e convertendo novamente para equação das diferenças, tem-se que:

$$\frac{\mu'_{ECT}(z)}{\mu(z)} = \frac{1}{2} \left(\frac{1 - z^{-1}}{1 - (1 - F_L)z^{-1}} \right)$$

$$\mu'_{ECT}(z) = \frac{1}{2} \left(\frac{1 - z^{-1}}{1 - (1 - F_L)z^{-1}} \right) \mu(z)$$

$$\mu'_{ECT}(z)(1 - (1 - F_L)z^{-1}) = \frac{1}{2}(1 - z^{-1})\mu(z)$$

$$\mu'_{ECT}(z) - (1 - F_L)z^{-1}\mu'_{ECT}(z) = \frac{1}{2}(1 - z^{-1})\mu(z)$$

$$\mu'_{ECT}(z) = \frac{1}{2}(1 - z^{-1})\mu(z) + (1 - F_L)z^{-1}\mu'_{ECT}(z)$$

$$\mu'_{ECT}[n] = \frac{1}{2}(\mu[n] - \mu[n - 1]) + (1 - F_L)\mu'_{ECT}[n - 1]$$

$$\mu_{ECT}[n] = \frac{1}{2}(\mu[n] - \mu[n-1]) + (1 - F_L) \left(\mu_{ECT}[n-1] - \frac{1}{2} \right) + \frac{1}{2} \quad (64)$$

Nota-se que o termo $(\mu[n] - \mu[n-1])$ na equação (64), correspondendo a entrada atual menos a entrada anterior, denota o efeito derivador da saída μ_{ECT} atual.

A CNAPap pode ser utilizada como extrator de média, estimador ou filtro passa baixas (CARVALHO JUNIOR *et al.*, 2018; DA CRUZ *et al.*, 2015). Da mesma forma pode-se utilizar a CNAPap_{CTX}, podendo a atuação do extrator de média, estimador ou filtro ser variável, em função de F_L .

Partindo-se da equação da CNAPap_{CTX} pode-se chegar a estrutura de um filtro digital de 1º ordem (LATHI; GREEN, 2017), onde F_L pode ser obtido através de uma relação com o tempo de amostragem (T_S), conforme a equação (65).

$$\begin{aligned} \mu_{E(k+1)} &= \mu_{E(k)} + D_{CT} F_L \\ \mu_{E(k+1)} &= \mu_{E(k)} + (\mu_{(k+1)} + \lambda_{(k+1)} - 1) * F_L \\ \mu_{E(k+1)} &= \mu_{E(k)} + \mu_{(k+1)} * F_L + \lambda_{(k+1)} * F_L - F_L \\ \mu_{E(k+1)} &= \mu_{E(k)} + \mu_{(k+1)} * F_L + (1 - \mu_{E(k)}) * F_L - F_L \\ \mu_{E(k+1)} &= \mu_{E(k)} + \mu_{(k+1)} * F_L + F_L - \mu_{E(k)} * F_L - F_L \\ \mu_{E(k+1)} &= \mu_{E(k)} + \mu_{(k+1)} * F_L - \mu_{E(k)} * F_L \\ \mu_{E(k+1)} &= \mu_{(k+1)} * F_L + (1 - F_L) * \mu_{E(k)} \\ F_L &= \frac{T_S}{T_S + \tau} ; (1 - F_L) = \frac{\tau}{T_S + \tau} \end{aligned} \quad (65)$$

Filtros de ordem superior podem ser obtidos através da ligação de CNAPap_{CTX} em cascata. Por exemplo, adotando-se F_L de 0.25, utilizando-se 2 CNAPap_{CTX} em cascata, tem-se na equação (66):

$$\frac{\mu_E(z)}{\mu(z)} = \frac{0.0625}{1 - 1.5z^{-1} + 0.5625z^{-2}} \quad (66)$$

4.6 SUMÁRIO DO CAPÍTULO

Este capítulo apresentou o neurônio paraconsistente (função de ativação LPA2v) proposto neste trabalho, para a composição da rede neural paraconsistente (RNP) a ser projetada para atividades de aproximação de funções, identificação e controle de sistemas dinâmicos.

A descrição matemática, cálculo das derivadas utilizadas no método de *backpropagation* para ajuste dos pesos também é descrito neste capítulo.

Pelo fato de a CNAPap_{CTX} refletir na saída um valor igual ao da entrada, após um certo número de iterações, para diferentes valores de F_L utilizados, diferentemente da CNAPap, sugere-se que o emprego destas células com a RNP permite a construção de rede neural paraconsistente recorrente (RNPR).

Também, neste capítulo foi demonstrado que a CNAPap_{CTX} pode ser utilizada como integrador, diferenciador ou filtro, dependendo da saída utilizada, μ_E ou μ_{ECT} .

O próximo capítulo apresenta a proposição de identificação de sistemas com o uso de rede neural paraconsistente, bem como a metodologia utilizada no projeto e aprendizado da rede.

5. METODOLOGIA

A metodologia adotada para o desenvolvimento deste trabalho envolve o uso intensivo de ferramenta computacional Matlab® e Simulink, bem como experimentações práticas em um protótipo de RIP, disponível no LAC da Escola Politécnica da USP, como planta dinâmica não linear.

Para a verificação da capacidade de aproximação de funções e utilização na identificação e controle de sistemas dinâmicos, foram estabelecidos quatro experimentos diferentes, a saber.

- a) O primeiro experimento consiste na verificação da capacidade de aproximação da RNP, através da análise comparativa, via Matlab®, do problema clássico de rede neural com porta OU exclusivo (EX-OR).
- b) O segundo experimento consiste na verificação da capacidade de aproximação da RNP para uma função de entrada-saída, base de dados simplefit, disponível como exemplo da ferramenta de rede neural (*neural fitting tool- nftool*) do Matlab®.
- c) O terceiro experimento consiste na identificação para controle de modelo do protótipo do RIP operando em malha fechada.
- d) O quarto experimento consiste no desenvolvimento e implantação de um controle tipo MRC com RNPR para o protótipo RIP a partir dos resultados e das considerações do terceiro experimento.

5.1. RNP PARA APROXIMAÇÃO DE VALOR – ANÁLISE DO PROBLEMA EX-OR

Treinar uma rede neural para reproduzir uma porta ou exclusivo (exclusive or – EX-OR) é um problema clássico de análise de redes neurais (ZHAO; DENG; WANG, 2002). Neste trabalho, cinco redes neurais foram projetadas, quatro RNAs usando a função log-sigmoide (σ), tanh, ReLU, LReLU e uma RNP correspondente com neurônio LPA2v. As redes foram projetadas consistem em duas entradas, três neurônios ocultos e um neurônio de saída.

A figura 34a apresenta a tabela verdade da porta ou exclusivo, enquanto que a figura 34b apresenta a RNA padrão, no caso com função log-sigmoide, e a figura 34c a RNP equivalente.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

(a)

(b)

(c)

Figura 34 – Tabela Verdade XOR (a), RNA (b) e RNP (c).

Nas duas redes, quatro neurônios com função de ativação não linear foram usados, sendo 3 na camada oculta e um na camada de saída. Pesos são aplicados do neurônio oculto ao neurônio de saída em todas as redes, ou seja, todos os neurônios são “treináveis”.

Para o RNP, é necessário um CNAPp (linear) adicional para combinar dois neurônios (CNAPp1 e CNAPp2) em uma saída (CNAPp4), conforme figura 34c. Para o cálculo da descida do gradiente, a derivada $\text{Linear}_{\text{LPA2v}}$ para o CNAPp4 é apenas uma constante, conforme a Equação (55).

Conforme as figuras 34b e 34c, as redes RNA e RNP utilizam nove pesos entre os neurônios. Os mesmos pesos iniciais aleatoriamente definidos são aplicados para todas as redes. As RNAs e a RNP foram criadas com um *script* Matlab®, apresentado no apêndice A.

A regra da cadeia é usada como parte do método de retropropagação (*backpropagation*) para calcular os pesos. A convergência para o alvo de saída e o erro quadrado médio (MSE) são usados para comparação. Os resultados obtidos são apresentados no capítulo 6, seção 6.1.

5.2. RNP PARA APROXIMAÇÃO DE FUNÇÃO ENTRADA-SAÍDA

O conjunto de dados “*simplefit*”, disponível no aplicativo nftool do Matlab®, foi usado, conforme apresentado na figura 35.

Figura 35 – Base de dados simplefit do Matlab

A RNP é comparada com a RNA disponível na ferramenta nftool do MATLAB®, utilizando função de ativação tanh na camada oculta e purelin (linear) na camada de saída. Ambas as redes utilizam múltiplas entradas. No caso da RNP, o neurônio paraconsistente de múltiplas entradas é conforme mostrado na figura 36a. Com isso uma RNP com apenas dois neurônios paraconsistentes é projetada, como mostrado na figura 36b é utilizado na comparação com a RNA disponível na ferramenta nftool (figura 36c).

Figura 36 – Um Neurônio LPA2v de 4 entradas (a). RNP com 2 neurônios de 4 entradas na camada oculta (b). RNA gerada pelo Matlab com n neurônios na camada oculta (c).

Valores aleatórios são aplicados aos pesos (w) e *offset* (b) no início do processo de aprendizado. O treinamento em lote com 90% de pontos para treinamento é selecionado para o processo de aprendizado.

No caso da RNP, os pesos estão sendo aplicados apenas na entrada da rede. Não há pesos entre a camada oculta e a CNAPp de saída. Em outras palavras, os únicos neurônios treináveis são os da camada oculta.

Se um peso for levado a um valor negativo durante o processo de aprendizado do neurônio paraconsistente e RNP, na próxima época o lote aplicado à entrada relativa a este peso é complementada como apresentado na equação (67). Isso permite que a RNP trabalhe com funções inversas, da mesma forma que a RNA utilizada na ferramenta do Matlab®, que utiliza função de ativação tanh.

Independentemente do valor do peso, a entrada da CNAP deve ser limitada a valores entre [0,1], de modo a atender a LPA2v. Assim sendo, o algoritmo para o neurônio paraconsistente e também para a RNP deve garantir que o bloco de saturação seja aplicado antes das entradas μ , conforme apresentado anteriormente na Figura 30. Além disso, os valores de *bias* (b) também são limitados entre [0,1].

$$\begin{aligned}
 w_n < 0 &\rightarrow entrada = abs(w_n) * (1 - entrada) \\
 w_n > 0 &\rightarrow entrada = w_n * entrada
 \end{aligned}
 \tag{67}$$

A ferramenta nftool do Matlab® cria redes neurais com um número de neurônios selecionável na camada oculta, com função de ativação tanh e um neurônio na camada de saída com função purelin (linear). Assim sendo, a RNP apresentada na figura 35b corresponderia a uma RNA com 2 neurônios na ferramenta nftool do Matlab®. Os resultados obtidos são apresentados no capítulo 6, na seção 6.2.

É importante observar que há uma normalização da saída da última CNAPp, tanto do neurônio paraconsistente quanto da RNP conforme a equação (68). Isso garante que a saída da última CNAPp apresente um valor pico a pico entre 0 e 1. A ferramenta nftool também utiliza um bloco normalizador após a saída do último neurônio.

$$\mu_{E(saida)} = \frac{\mu_E - \min(\mu_E)}{\max(\mu_E) - \min(\mu_E)} \quad (68)$$

5.3. IDENTIFICAÇÃO DE SISTEMA DINÂMICO COM RIP

Para a verificação da capacidade de aproximação de função e identificação de um sistema dinâmico real, validando a RNP para problemas práticos, foi utilizado o protótipo de um pêndulo invertido rotativo (RIP), apresentado na Figura 37.

Figura 37 – Protótipo RIP.

Fonte: Adaptado de <http://www.lac.usp.br/pendulorotacional.html>, acessado em 02/03/2020.

O RIP consiste em um braço acionado que gira no plano horizontal com um pêndulo conectado, livre para girar no plano vertical. O RIP possui características

importantes, como não linearidade, dinâmica rápida, múltiplas variáveis e movimento espacial multidimensional, permitindo avaliar vários elementos da moderna teoria de controle (AGUILAR-AVELAR; MORENO-VALENZUELA, 2015; SEMAN *et al.*, 2013; SINGH; YADAV, 2012).

O protótipo do RIP, apresentado na Figura 37, possui uma entrada para o sinal de controle (u) do motor fixado no braço e duas saídas, fornecidas pelos *encoders*, como sensores do ângulo horizontal do braço (θ_0), ângulo vertical do pêndulo (θ_1). O sistema é controlado por um microcontrolador Teensy 3.2 com processador Cortex-M4 de 32 bits, relógio de 72 MHz, memória *flash* de 256 kBytes e RAM de 64 kBytes. Duas outras saídas, velocidade do braço $[\dot{\theta}_0]$ e velocidade do pêndulo $[\dot{\theta}_1]$ são calculadas pelo método da derivada da variação do ângulo do braço e do pêndulo pelo próprio *software* do microcontrolador que controla o RIP. O tempo de amostragem adotado foi de 10 mseg e nenhuma referência é aplicada. As especificações do RIP são apresentadas na Tabela 2.

Tabela 2 – Parâmetros do RIP.

PARAMETRO	VALOR	DESCRIÇÃO
d	0.137 (m)	distância do eixo base ao centro de massa
m_0	0.393 (kg)	massa do braço
m_1	0.068 (kg)	massa do pêndulo
L_0	0.1825 (m)	comprimento da metade do braço
L_1	0.1035 (m)	comprimento da metade do pêndulo
r	0.23 (m)	distância do eixo base ao pêndulo
R_m	2.4 (Ohms)	resistência da armadura do motor
K_t	0.08 (Nm/A)	constante de torque do motor, incluindo redução
K_e	0.2 (V/(rad/s))	constante da equação do motor, incluindo redução
g	9.81 (m/s ²)	aceleração da gravidade
q_0	rad	ângulo do braço (positivo para sentido anti-horário)
q_1	rad	ângulo do pêndulo (positivo para sentido horário)
T_s	0.01 s	tempo de amostragem

A descrição matemática do RIP e os métodos de projeto para controlá-lo estão fora do escopo deste trabalho, mas estão disponíveis na literatura (AGUILAR-AVELAR; MORENO-VALENZUELA, 2015; SEMAN *et al.*, 2013; SINGH; YADAV, 2012). No Anexo A é apresentada uma breve descrição da modelagem do RIP. Com base nos valores da Tabela 2, a função Lagrange, descartando o atrito e aplicando

outras simplificações (MATHEW; RAO; SIVAKUMARAN, 2013), as matrizes A e B são calculadas utilizando-se o *script* em Matlab apresentado no Anexo B, resultando em (69), a ser aplicado na representação do espaço de estados.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -9.1047 & -0.6066 & 0.0132 \\ 0 & 86.2615 & 1.0110 & -0.1249 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 31.6490 \\ -52.7483 \end{bmatrix} \quad (69)$$

O modelo linearizado do RIP possui 4 variáveis $[\theta_0; \theta_1; \dot{\theta}_0; \dot{\theta}_1]$, no entanto, o protótipo apenas possui medições de $[\theta_0; \theta_1]$. Assim, as variáveis de velocidade do braço e pêndulo $[\dot{\theta}_0; \dot{\theta}_1]$ devem ser estimadas. Elas podem ser calculadas por derivada *backward*, conforme a equação (70).

$$\dot{\theta}_0 = \frac{(\theta_0[n] - \theta_0[n-1])}{T_s} \text{ e } \dot{\theta}_1 = \frac{(\theta_1[n] - \theta_1[n-1])}{T_s} \quad (70)$$

De modo a reduzir o ruído das medições e amostragem dos *encoders* de ângulo do braço e pêndulo, bem como de suas derivadas, um filtro-passa-baixas (*low-pass filter* – LPF) de 1ª ordem pode ser implementado, (LATHI; GREEN, 2017), conforme a equação (71) para o ângulo do braço $[\theta_0]$ e n é o número de uma determinada amostra e τ a constante de tempo do filtro. A figura 38 apresenta um diagrama de blocos do sistema RIP-Controle.

$$\theta_0[n]_{filt} = \left(\frac{\tau}{\tau + T_s} \right) \theta_0[n-1]_{filt} + \left(\frac{T_s}{\tau + T_s} \right) \theta_0[n] \quad (71)$$

Após a conversão das matrizes A e B para sistema discreto, o ganho do controlador (K) pode ser calculado pelo método de posicionamento dos polos, disponível em Matlab®, para os polos $[-2 \ -2,5 \ -11 \ -10]$ adotados. Assim, a equação (72) apresenta a lei de controle resultante.

$$u[k] = -K[k].x[k]$$

$$K = \begin{bmatrix} -0.2160 & -5.3459 & -0.2563 & -0.5934 \end{bmatrix}$$

(72)

Figura 38 – Diagrama de Blocos sistema RIP-Controlador.

Como o RIP é um sistema altamente instável, a identificação foi realizada com o controle de malha fechada. Foram utilizadas 3 RNPs com quatro entradas e uma saída cada, conforme apresentado na Figura 39.

Uma RNP é utilizada para a identificação do modelo do ângulo do braço, uma para o ângulo do pêndulo e uma para o inverso do sistema, ou seja, o modelo do sinal de controle. Esse tipo de modelo inverso é utilizado por algumas técnicas de controle, como o IMC, conforme apresentado na seção 2.2.5. O CNAPap_{CTX} também ajuda a filtrar os transientes e ruído dos sinais.

Cada RNP consiste em dois neurônios paraconsistentes de quatro entradas na camada oculta, conforme apresentado anteriormente na Figura 36a, combinados por uma célula CNAPp linear, fornecendo uma saída. Os dados das entradas e saídas do RIP são exportados em arquivos formato txt, para serem utilizados no treinamento *offline* da RNP. São aplicados pesos apenas nas entradas da rede, mas não entre a camada oculta e a CNAPp de saída.

Figura 39 – RNP para identificação do sistema dinâmico RIP.

Em um segundo experimento, os modelos obtidos pelas RNPs foram comparados com medições utilizando um controlador com desempenho inferior. Este controlador causa uma maior variação do pêndulo em relação ao ponto de equilíbrio, mas ainda é capaz de manter a estabilidade. O vetor (K) deste controlador é apresentado em (73), alterando-se ligeiramente os valores apresentados em (72). Os resultados são apresentados e analisados na seção 6.3.

$$K = [-0.2059 \quad -4.5056 \quad -0.1864 \quad -0.4968] \quad (73)$$

5.4 ESTIMADOR PARACONSISTENTE

Os ensaios com o protótipo RIP utiliza derivadores para a obtenção das velocidades do braço e pêndulo e também filtros de 1º ordem para as 4 variáveis de entrada antes de serem aplicadas ao controlador. Um resultado semelhante pode ser obtido utilizando-se circuitos de CNAPap_{CTX}, permitindo assim uma solução totalmente baseada na LPA2v.

O estimador paraconsistente utiliza um conjunto de células do tipo CNAPap_{CTX}, como integrador e derivador, tendo como entrada os sinais $[\theta_0; \theta_1]$ fornecidos pelo RIP, entregando na saída os sinais $[\theta_{0filt}; \theta_{1filt}; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$, filtrados. O estimador

pode ser estático ou dinâmico. As células CNAP e CNAPap_{CTX} utilizadas nas simulações deste projeto são apresentadas no Apêndice C.

5.4.1 Estimador Paraconsistente Estático

No estimador paraconsistente estático, a filtragem dos sinais é obtida pela saída μ_E das CNAPap_{CTX}, como um filtro de 1ª ordem discreto. As saídas μ_{ECT} da primeira camada de células [#1,#2] fornece a derivada dos ângulos para posterior filtragem. Para isso, o FL delas tem valor unitário. A figura 40 apresenta a topologia do estimador paraconsistente estático para o RIP.

Figura 40 – Estimador Paraconsistente Estático do RIP em Simulink.

Sem referência, os sinais de saída do RIP apresentam baixa amplitude. Entretanto, para o caso de o braço do pêndulo seguir uma referência de onda retangular com amplitude, ou deslocamento, de até $\pm \pi/3$ rad, adota-se um ganho de entrada (G_E) de $1/4$ e ganho de saída (G_{S1}) de $1/G_E$. Para uma amplitude adequada dos sinais de derivadas $[\dot{\theta}_0; \dot{\theta}_1]$, o ganho de saída (G_{S2}) deve ser igual a $2G_{S1}/T_S$.

As entradas $[X0, X1]$ correspondem aos ângulos do braço e pêndulo $[\theta_0; \theta_1]$, enquanto que as saídas $[X0est, dX0est, X1est, dX1est]$ correspondem às saídas filtradas $[\theta_{0filt}; \theta_{1filt}; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$. Para os valores adotados, a resposta da rede de células da

figura 40 é matematicamente idêntica ao obtido através das equações (70) de derivadas dos ângulos e (71) para os filtros. Duas equações de derivada e quatro equações de filtragem, exatamente o mesmo número de células no estimador paraconsistente estático. Em uma composição de estimador mais controlador, o ganho de normalização da saída pode ser incorporado ao ganho do controlador, para fins de simplificação.

5.4.2 Estimador Paraconsistente Dinâmico

Com uma pequena variação na configuração da rede de células e utilizando a saída do intervalo de certeza (φ_E), conforme apresentado no capítulo 3 pela equação (40), pode-se construir um estimador paraconsistente dinâmico. A figura 41 apresenta a topologia deste estimador para o RIP. Para maior eficácia do estimador como filtro, com menor atraso na ausência de ruído, foram utilizadas 2 células em cascata, formando um filtro passa baixas discreto de 2ª ordem.

Figura 41 – Estimador paraconsistente dinâmico.

Para um sinal sem ruído, não há contradição (D_{CT}) e, portanto, o intervalo de certeza é unitário. Quanto maior a amplitude do ruído, mais próximo de zero é o valor de φ_E . Nota-se pela figura 41, que a saída φ_E (PHI) das células #1 e #6 se conectam às entradas F_L dos filtros de 2º ordem, após uma correção de *offset*. De acordo com a equação (40) de φ_E quanto maiores os picos instantâneos de ruído, menor o valor

instantâneo do intervalo de certeza e mais baixo o valor de F_L , causando uma maior atuação das células como filtro. Um ajuste fino de *offset* coloca o estimador no ponto ideal. Por simulações verificou-se que para variâncias de ruído abaixo de 0.01, o ajuste do *offset* deve estar no intervalo de [0.45,0.65].

5.5 SUMÁRIO DO CAPÍTULO

Este capítulo apresentou as etapas de teste e validação de uma rede neural com aprendizado supervisionado, baseado em lógica LPA2v, chamada de RNP. Na primeira etapa de testes, uma RNP é comparada com uma RNA, com mesmas características e valores de partida, para o problema do Ex-OR. Na segunda etapa, uma função disponível no Matlab® para teste de redes neurais (simplefit) é utilizada para comparação de resultados de MSE entre uma RNP com 2 neurônios na camada oculta e RNA, construída pela ferramenta nftool do Matlab®, com diferentes quantidades de neurônios na camada oculta. O terceiro teste e validação é proposto utilizando-se um sistema dinâmico, não linear, de múltiplas variáveis, como o RIP, para identificação de modelos. Um estimador paraconsistente, utilizando células CNAPp e CNAPap_{CTX}, é apresentado e pode ser implementado em substituição aos derivadores e filtros para o tratamento das variáveis antes de serem aplicadas ao controle. Isso permite a construção de todo um sistema de estimação, filtragem e controle paraconsistente. O próximo capítulo apresenta os resultados obtidos nos ensaios.

6. RESULTADOS PRELIMINARES

Este capítulo apresenta os resultados preliminares do uso das RNPs.

6.1 RESULTADOS DA ANÁLISE DO PROBLEMA EX-OR

O *script* em Matlab®, apresentado no Apêndice B, pede que sejam introduzidos os valores das entradas e saída esperado. O *script* permite outras combinações de portas, não somente da porta ou exclusivo, como, por exemplo, entradas [0,0] e saída [1]. Ao ser executado, ele gera os gráficos das saídas das RNAs com diferentes funções de ativação (sigmoide (σ), tanh, ReLU, LReLU) e RNP e também o MSE de cada rede. Recapitulando o que foi apresentado na seção 5.1, a RNP e as RNAs possuem o mesmo número de neurônios. Os mesmos valores de pesos iniciais, randomicamente definidos, são aplicados a todas as redes neurais e mesma taxa de aprendizagem (*learning rate*) de valor unitário.

A figura 42 apresenta o aprendizado de entradas em [0,0] e saída também [0]. As RNAs ReLU e LReLU por serem funções similares à função identidade e o valor de saída é o mesmo do apresentado nas entradas, não requerem aprendizado para esta situação.

Figura 42 – Aprendizado RNP e RNA. Entradas [0,0] e Saída [0] (a) e MSE(b).

Nota-se pela figura 42a que a RNA log-sigmoide (linha magenta) atinge o valor de 0.1 após 100 épocas, enquanto que o RNP (linha azul) consegue chegar a um valor final mais próximo ao da saída desejada que o da RNA tanh, ainda que até a época 80 estava com valor maior que o desta RNA. A figura 42b apresenta o gráfico de MSE. Após 100 épocas, a saída da RNP está em 0.0131, a saída da RNA com tanh está em 0.0207 e a saída da RNA log-sigmoide está apenas em 0.1003.

A figura 43 apresenta o aprendizado de entradas em [1,0] e saída também [1]. Nota-se pela figura 43a que o RNP consegue chegar ao resultado (0.9933) com aproximadamente 30 épocas, enquanto que com 100 épocas a RNA log-sigmoide ainda está em 0.9323 e a RNA com tanh está em 0.9247. Nota-se também pela figura que as RNAs com funções de ativação ReLU e LReLU ficam oscilando para mais e para menos em torno do valor alvo de saída. Analisando-se a figura 43b percebe-se que apesar de partir de um MSE inicial mais alto, a RNP consegue ultrapassar a aproximação de todas as RNAs, a partir da época 12. A figura 43b mostra ainda que a RNP conseguiu reduzir o MSE para próximo de zero com menos de 25 épocas, muito mais rapidamente que todas as RNAs. Ao final de 100 épocas a RNP atingiu um MSE de 1,0094e-06, a RNA log-sigmoide obteve um MSE de 0.0028, o da RNA tanh foi de 0.006, das RNAs ReLU e LReLU foi de 0.0022.

Figura 43 – Aprendizagem RNP e RNA. Entradas [1,0] e Saída [1] (a) e MSE(b).

A aplicação de uma taxa de aprendizagem menor que um faz com que a pequena transição observada para a RNP em torno de 4 épocas e o comportamento divergente observado para as redes ReLU e LReLU ao longo de todo o gráfico 43c desapareçam, permitindo que os gráficos de MSE passem a apresentar curvas suaves. Entretanto, um compromisso deve ser atingido, pois taxa de aprendizagem menor implica em maior número de épocas para o aprendizado da rede.

A figura 44 apresenta o aprendizado de entradas em [1,1] e saída também [0]. Nota-se pela figura 44a que a RNA ReLU aprende muito rapidamente o resultado desejado. O aprendizado da RNP acompanha o da LReLU a partir 25 épocas, assim

como o da RNA tanh. Novamente, a RNA log-sigmoide é a que necessita de mais épocas para aprender o valor de saída. Ao final de 100 épocas a RNP atingiu o valor de 0.0127, a RNA ReLU alcançou o valor 0 na segunda época, a LReLU alcançou 0.0017, a tanh chegou a 0.0095 e a log-sigmoide apenas 0.0710.

Figura 44 – Aprendizado RNP e RNA. Entradas [1,1] e Saída [0] (a) e MSE(b).

Em termos de MSE, a RNP ficou próxima da ReLU e da LReLU. Após 100 épocas a RNP alcançou um MSE de $8,0402e-05$, a RNA LReLU obteve um MSE de $1,4194e-06$, a tanh de $4,5145e-05$ e a log-sigmoide de apenas 0,0025.

Partindo-se de dois valores aleatórios de entrada [0.92;0.14] e um de saída [0.27], com taxa de aprendizagem de 0.5 para todas as redes, tem-se os resultados apresentados na figura 45.

(a)

(b)

Figura 45 – Aprendizado RNP e RNA. Entradas [0.92;0.14] e Saída [0.27] (a) e MSE(b).

Observa-se na figura 45a que as redes ReLU e LReLU tem um aprendizado muito rápido, com cerca de 10 épocas nesta condição, enquanto que a RNP precisou de 90 épocas para se aproximar do resultado das redes ReLU e LReLU. Já as redes log-sigmoide e tanh mesmo com 100 épocas ficaram longe do valor alvo. A figura 45b

confirma que em termos de MSE a RNP foi a que mais se aproximou dos resultados das redes ReLU e LReLU.

Os resultados apresentados mostram que a RNP consegue uma maior aproximação da saída com um menor número de épocas do que uma RNA sigmoide clássica (log-sigmoide ou tanh) e resultados próximos de redes do tipo ReLU, todas com o mesmo número de neurônios, quantidade e mesmos valores iniciais para os pesos utilizados.

6.2. COMPARAÇÃO ENTRE RNP E RNA NA APROXIMAÇÃO DE FUNÇÃO ENTRADA-SAÍDA

Os pesos obtidos na melhor simulação de aproximação de função entrada-saída disponível na base de dados da ferramenta nftool do Matlab®, para a RNP são apresentados na Tabela 3.

Tabela 3 – Pesos e MSE da aproximação por um neurônio LPA2v e RNP com dois neurônios.

Pesos	1 neurônio LPA2v	RNP	
		Neuronio 1	Neuronio 2
w_1	-3,080	1,690	1,010
w_2	2,730	1,070	1,010
w_3	-3,720	2,590	-3,130
w_4	4,160	10,350	-5,720
b_1	0,000	0,920	0,790
b_2	0,570	0,670	0,010
b_3	0,880	0,530	1,000
b_4	0,610	0,320	0,800
MSE	0,253	0,058	

A figura 46a apresenta a entrada (verde), saída de destino (preto), os melhores resultados de saída para um neurônio paraconsistente (vermelho) de 4 entradas e uma RNP com 2 neurônios de 4 entradas na camada oculta (azul). A figura 46b apresenta o resultado da ferramenta nftool, para uma RNA com 4 neurônios na camada oculta. Nota-se que com 4 neurônios a rede não foi capaz de aproximar os picos e vales nas amostras 15 e 40, mas fez uma média entre eles.

(a)

(b)

Figura 46 – Função Simplefit: entrada (verde), saída alvo (preto), resultado de 1 neurônio paraconsistente (vermelho) e de 1 RNP com 2 neurônios na camada oculta (azul) (a) e resultado nftool para RNA de 4 neurônios na camada oculta (b).

A Tabela 4 apresenta os resultados comparativos entre RNP e RNA pela nftool. São apresentados os resultados de apenas um neurônio LPA2v, da RNP com 2 neurônios de 4 entradas na camada oculta, contra diferentes configurações de RNAs da ferramenta nftool do Matlab® e método de treinamento *Scaled Conjugate Gradient*.

Tabela 4 – Comparação entre RNP e diferentes RNA para aproximação função simplefit.

Rede Neural	MSE
1 neurônio PAL2v com 4 entradas	0,277
RNP com 2 neurônios camada oculta e 4 entradas	0,074
RNA com 1 neurônio camada oculta	1,319
RNA com 2 neurônios camada oculta	1,308
RNA com 3 neurônios camada oculta	0,564
RNA com 4 neurônios camada oculta	0,062
RNA com 5 neurônios camada oculta	0,01

Os resultados de MSE demonstram que um neurônio LPA2v com quatro entradas apresentou um resultado melhor que uma RNA com três neurônios ocultos. A RNP com apenas dois neurônios ocultos de quatro entradas possui uma boa representação do alvo, como a linha azul da Figura 46a, e apresentou um resultado MSE melhor do que uma RNA clássica com 4 neurônios ocultos.

6.3. RESULTADOS DA IDENTIFICAÇÃO DE SISTEMA DINÂMICO COM RIP

Para o controle perfeitamente ajustado, em malha fechada, não foi necessário a inclusão das CNAPap_{CTX} na entrada e saída da rede neural paraconsistente. Apenas 2 CNAPap_{CTX} foram aplicadas na entrada, para tratar os sinais $[\dot{\theta}_0; \dot{\theta}_1]$, como filtro de 1º ordem, com F_L de 0.1667, equivalente à $\tau = 5T_s$. Assim, a rede neural paraconsistente utilizada foi uma RNP, não recorrente. A figura 47 apresenta o diagrama da planta, controle e das redes neurais, considerando que as RNPs aprenderam em modo *offline*.

Os sinais de controle, ângulo do pêndulo, velocidade do braço e velocidade do pêndulo filtrados $[u; \theta_{1filt}; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$ são aplicados às entradas do RNP1 e a saída é comparada ao ângulo do braço $[\theta_0]$. Os sinais de ângulo do braço, controle, velocidade do braço e do pêndulo $[\theta_{0filt}; u; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$ são aplicados às entradas do RNP2 e a saída é comparada ao ângulo do pêndulo $[\theta_1]$. Finalmente, os ângulos do braço e do pêndulo e as velocidades do braço e do pêndulo $[\theta_{0filt}; \theta_{1filt}; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$ são aplicados às entradas do RNP3 e a saída é comparada ao sinal de controle $[u]$. A

RNP1 e a RNP2 são usadas para modelar a planta RIP, também chamada de modelo direto, já a RNP3 resulta no modelo do controlador.

Figura 47 – Diagrama da planta, controle e entrada-saída das redes neurais.

Vinte mil amostras do sinal de controle $[u]$ aplicado ao RIP e sua saída $[\theta_{0filt}; \theta_{1filt}; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$ foram coletadas. O processo de aprendizado utilizou 2000 valores aleatórios das primeiras 8000 amostras para treinar as RNPs. Como normalização do sinal de entrada e saída, considerando-se o maior valor pico a pico entre os sinais de controle e das saídas do RIP, foi adotado um ganho G_E igual para todas as entradas de todas as três RNPs, no valor de 0.125 e um ganho de saída da RNP, G_S , conforme a equação (74). A Tabela 5 apresenta os resultados do processo de aprendizagem das 3 RNPs, comparado com RNAs equivalentes via nftool do Matlab e método de treinamento *Scaled Conjugate Gradient*.

$$G_S = \frac{1}{G_E} * 2^N \quad (74)$$

Tabela 5 – Comparação entre RNP e diferentes RNA para identificação modelo sistema RIP- controle ótimo.

Rede Neural (camada oculta)	MSE Treinamento	MSE Validação	Epochs
RNA 2 neurônios	3,99E-05	4,68E-05	105
RNA 4 neurônios	8,45E-05	6,96E-05	119
RNP1	8,97E-06	7,18E-05	60
RNA 2 neurônios	2,13E-07	3,29E-07	155
RNA 4 neurônios	3,40E-07	5,14E-07	172
RNP2	2,07E-07	7,10E-08	10
RNA 2 neurônios	3,69E-06	3,80E-06	253
RNA 4 neurônios	2,82E-06	2,53E-06	277
RNP3	9,66E-07	4,33E-06	30

Os resultados indicam que o RNP com apenas 2 neurônios na camada oculta pôde aprender muito bem as saídas de ângulo do braço, ângulo do pêndulo e controlador nesta condição. Nota-se que nas RNPs utilizadas, diferentemente das RNAs do Matlab®, não houve a necessidade de aplicação e treinamento de pesos entre a camada oculta e a de saída da rede neural. Além disso, a Tabela 5 indica que a RNP apresenta resultados MSE mais baixos e requer menos épocas do que a RNA padrão criada por nftool de Matlab®, mesmo comparando a RNA com 4 neurônios na camada oculta contra RNP com apenas 2 neurônios. A RNP1 e a RNP3 obtiveram melhores resultados de treinamento e valores muito próximos de validação aos das RNAs com 2 e 4 neurônios na camada oculta. A RNP2 obteve melhores valores de treinamento e validação que as RNAs com 2 e 4 neurônios na camada oculta. Em todos os casos as RNPs apresentaram convergência mais rápida que as RNAs.

A Tabela 6 apresenta os pesos encontrados para RNP1, RNP2 e RNP3. É possível notar que os pesos para a RNP3 equivaleriam aos ganhos do controlador utilizado, cuja matriz K é repetida a seguir. Como as equações da LPA2v são bem definidas, com os pesos estabelecidos é possível elaborar equações que descrevam o comportamento das RNPs pós aprendizado.

$$K = [-0.2160 \ -5.3459 \ -0.2563 \ -0.5934]$$

A figura 48 apresenta os resultados da identificação, considerando amostras de 13000 a 14000. Como o RIP é naturalmente instável, é muito difícil realizar sua identificação.

Tabela 6 – Pesos aprendidos pelas RNP1, RNP2 e RNP3.

RNP		Pesos Aprendidos			
		w_1	w_2	w_3	w_4
RNP1	Entrada Neurônio 1	4,0008	19,5192	0,81293	-1,0993
	Entrada Neurônio 2	-3,9164	-19,233	-0,8423	1,1936
RNP2	Entrada Neurônio 1	0,0546	-0,2096	0,16307	0,0759
	Entrada Neurônio 2	0,2082	0,28338	0,05992	0,2172
RNP3	Entrada Neurônio 1	0,44709	-5,79829	-0,40066	0,71895
	Entrada Neurônio 2	-0,3023	5,88384	0,14811	-0,5792

(a)

(b)

(c)

(d)

Figura 48 – Controlador ajustado. Velocidade do Braço e Pêndulo (a), Identificação do Modelo do Braço pela RNP1 (b), Identificação do Modelo do Pêndulo pela RNP2 (c) e identificação do modelo invertido (controlador) pela RNP3 (d).

O controlador mantém o RIP em equilíbrio, com o pêndulo para cima, conforme a variação de pequena amplitude mostrada na Figura 48c. Apesar de uma pequena diferença nos picos positivos da figura 48b para o ângulo do braço, os resultados indicaram que as RNPs, com apenas 2 neurônios na camada oculta, foram capazes de aprender muito bem as saídas de braço, pêndulo e controlador.

No segundo experimento, foram coletadas medidas com um controlador com desempenho inferior, cujo vetor de ganho K apresentado em (73) é repetido a seguir, de modo a causar uma maior variação do pêndulo em relação ao ponto de equilíbrio, mas ainda capaz de manter a estabilidade, conforme figura 49.

$$K = \begin{bmatrix} -0.2059 & -4.5056 & -0.1864 & -0.4968 \end{bmatrix}$$

(a)

(b)

(c)

(d)

Figura 49 – Controlador com desempenho inferior e modelos levantados por RNPs com controlador de melhor desempenho. Sinais da velocidade do braço e pêndulo (a), resposta do modelo do braço pela RNP1 (b), resposta do modelo do pêndulo pela RNP2 (c) e resposta do modelo invertido (controlador) pela RNP3 (d).

Os sinais de entrada e saída do RIP foram aplicados aos modelos identificados com o treinamento da RNP utilizando o controlador melhor ajustado, ou seja, as RNPs não foram treinadas para a nova condição do controlador.

Comparando as Figuras 48a e 49a, é possível observar que a segunda é menos regular e com maior amplitude de velocidade, principalmente para o braço (linha preta). Considerando que os modelos RNP1, RNP2 e RNP3 foram treinados com um controlador bem ajustado, a Fig. 49b mostra que RNP1 apresenta um comportamento (azul) próximo ao braço do pêndulo (vermelho), apesar da diferença de amplitude, principalmente na metade negativa do sinal, quando o controlador de desempenho inferior é aplicado ao RIP. De acordo com a Fig. 49c, a resposta de saída do RNP2 (azul) é muito próxima da saída do ângulo do pêndulo RIP, com valores máximos e mínimos de ângulo semelhantes, quando o controle de performance inferior é aplicado. O mesmo é observado na Fig. 49d, onde a resposta de saída do modelo invertido (RNP 3) (azul) segue muito próxima da saída do controlador de performance inferior (vermelho). Nesta situação, o MSE para RNP1 é 0,0056, o MSE para RNP2 é 0,00006 e o MSE para RNP3 é 0,0012.

6.4. RESULTADOS DO ESTIMADOR PARACONSISTENTE

De modo a verificar o uso da LPA2v para estimar e filtrar variáveis não medidas por sensores, foram realizadas simulações utilizando o modelo do RIP, conforme matrizes A e B apresentadas em (69), controlador do tipo LQR com ganho conforme dado a seguir (75) e adicionado ruído do processo com variância de 1×10^{-5} e ruído de medida com variância de 1×10^{-7} .

$$K = [0.3490 \quad -4.8884 \quad -0.2400 \quad -0.5420 \quad 0.0018] \quad (75)$$

A Figura 50 apresenta o modelo simulado em Matlab - Simulink do RIP (Plant). Derivadores, Filtros de 1ª ordem e controlador estão todos no bloco “Controller LQR com int.”, conforme indicado na Figura 38, da seção anterior.

Figura 50 – Modelo do RIP, Controle LQR com filtro LPF e derivador.

A Figura 51 apresenta o modelo simulado do RIP (Plant), controle LQR e bloco “Estimador Paraconsistente Dinâmico”, conforme indicado na Figura 41 da seção anterior.

Figura 51 – Modelo do RIP, Controle LQR e estimador paraconsistente dinâmico.

A figura 52 apresenta o resultado comparativo, conforme a seguir.

- Sinal de referência (r), do ângulo do braço (θ_0) do modelo do RIP, que segue a referência (linha preta), filtrado por LPF (azul) e pelo estimador paraconsistente (vermelho).
- Sinal do ângulo do pêndulo (θ_1) do modelo do RIP filtrado (azul) e pelo estimador paraconsistente (vermelho).
- Sinal da velocidade do braço ($\dot{\theta}_0$) do modelo do RIP derivado e filtrado (azul) e pelo estimador paraconsistente (vermelho).
- Sinal da velocidade do pêndulo ($\dot{\theta}_1$) do modelo do RIP derivado e filtrado (azul) e pelo estimador paraconsistente (vermelho).
- Gráfico da transformada rápida de Fourier (FFT) da saída do controlador (u), quando utilizando os sinais de saída do RIP derivados e filtrados por LPD e com estimador paraconsistente.

(d)

(e)

Figura 52 – Simulação com baixa variância de ruído. Referência (r) (preto), ângulo do braço por LPF (azul) e por estimador LPA2v (vermelho) (a). Ângulo do pêndulo por LPF (azul) e por estimador LPA2v (vermelho) (b). Velocidade do braço derivado e filtrado (azul) e pelo estimador LPA2v (vermelho) (c). Velocidade do pêndulo derivado e filtrado (azul) e pelo estimador LPA2v (vermelho) (d). FFT da saída do controlador com sinais do RIP derivados e filtrados por LPF e com estimador LPA2v (e).

Na figura 52a tem as saídas comparadas do ângulo do braço pelo filtro LPF (linha azul - $\theta_{0(filt)}$) e estimador LPA2v (linha vermelha - $\theta_{0(LPA2vest)}$), onde não há uma diferença perceptível. A figura 52b tem as saídas comparadas do ângulo do pêndulo pelo filtro LPF (linha azul - $\theta_{1(filt)}$) e estimador LPA2v (linha vermelha - $\theta_{1(LPA2vest)}$). Nota-se uma pequena diferença nos valores de pico a menor para o estimador LPA2v. A velocidade (derivada do ângulo) do pêndulo é apresentada na figura 52c, onde

praticamente os sinais pelo filtro LPF (linha azul - $\dot{\theta}_{1(filt)}$) e estimador LPA2v (linha vermelha - $\dot{\theta}_{0(LPA2vest)}$) são semelhantes. Novamente na figura 52d os valores de pico da velocidade do braço são menores para a saída do estimador LPA2v (linha vermelha - $\dot{\theta}_{1(LPA2vest)}$) em comparação ao filtro LPF mais derivador (linha azul - $\dot{\theta}_{1(filt)}$) tradicional. A transformada rápida de Fourier foi aplicada aos sinais do controlador nas duas simulações e apresentado na figura 52e. O espectro de potência é praticamente igual para as duas simulações, com uma pequena componente maior em torno da frequência de 2.5 Hz, para o conjunto filtro LPF e derivador (linha azul - $u_{(derivadas + LPF)}$) do que para o estimador LPA2v (linha vermelha - $u_{(LPA2vest)}$). Calculando-se a potência total, como soma de todas as componentes do FFT, tem-se que a Potência do sinal do controlador com filtros e derivadores resulta em 1.2163, ao passo que do controlador com estimador paraconsistente resulta em 1.1605, ou uma redução de 4.59%.

A simulação é novamente realizada, com o ângulo do braço (θ_0) seguindo uma referência (r) de onda quadrada (linha preta), agora com ruído do processo com variância de 1×10^{-2} e ruído de medida com variância de 1×10^{-4} . A figura 53 apresenta o resultado comparativo.

(a)

(b)

(c)

(d)

Figura 53 – Simulação com elevada variância de ruído. Ângulo do braço filtrado (azul) e pelo estimador paraconsistente (vermelho) (a). Ângulo do pêndulo filtrado (azul) e pelo estimador paraconsistente (vermelho) (b). Velocidade do braço derivado e filtrado (azul) e pelo estimador paraconsistente (vermelho) (c). Velocidade do pêndulo derivado e filtrado (azul) e pelo estimador paraconsistente (vermelho) (d). Gráfico da FFT da saída do controlador com sinais de saída do RIP derivados e filtrados e com estimador paraconsistente.

Na figura 53a tem-se as saídas comparadas do ângulo do braço pelo filtro LPF (linha azul - $\theta_{0(filt)}$) e estimador LPA2v (linha vermelha - $\theta_{0(LPA2vest)}$), ainda sem uma diferença perceptível. A figura 53b tem-se as saídas comparadas do ângulo do pêndulo pelo filtro LPF (linha azul - $\theta_{1(filt)}$) e estimador LPA2v (linha vermelha - $\theta_{1(LPA2vest)}$). Nota-se que os valores de pico do sinal filtrado (azul) é maior do que o proveniente do estimador LPA2v. O mesmo ocorre para as velocidades (derivadas) do braço, na figura 53c, e pêndulo, na figura 53d. Nota-se, pela figura 53e, que a distribuição espectral de potência do sinal do controlador utilizando as saídas do estimador LPA2v está mais concentrada e com um pico ligeiramente maior em torno da frequência 2.5 Hz. Já o controlador com derivadas e LPF apresenta várias componentes espectrais altas até 50 Hz. Calculando-se a potência total, como soma de todas as componentes do FFT, tem-se que a Potência do sinal do controlador com filtros e derivadores resulta em 146.012, ao passo que do controlador com estimador paraconsistente resulta em 129.945, ou uma redução de 11.00%.

Comparando-se os resultados apresentados na figura 52 e da figura 53, com ruído da ordem de 1000 vezes maior, percebe-se que o estimador paraconsistente atuou de forma dinâmica, resultando em uma potência total do sinal do controlador, 4.59% no primeiro cenário e 11.00% no segundo cenário, sem alterar as

características do sistema, já que não há atraso do sinal do ângulo do braço, nem do ângulo do pêndulo, conforme as figuras 53a e 53b.

6.5. SUMÁRIO DO CAPÍTULO

Este capítulo apresentou os resultados das etapas de teste e validação de uma rede neural com aprendizado supervisionado, baseado em lógica LPA2v, chamada de rede neural paraconsistente (RNP). Na comparação de uma RNP e uma RNA clássica, com mesmas características e valores de partida, para o problema do Ex-OR, a RNP precisou de praticamente 50% a menos de épocas do que as RNAs baseadas em funções sigmóides e apresentou resultados competitivos com RNAs baseadas em funções tipo ReLU.

No segundo experimento, uma RNP com 2 neurônios e 4 entradas na camada oculta apresentou resultados similares a RNA, construída pela ferramenta nftool do Matlab®, com 4 neurônios na camada oculta.

A RNP se mostrou viável para identificação de um sistema dinâmico, como o RIP, em controle de malha fechada, sendo capaz de identificar tanto no sentido direto as variáveis de saída do pêndulo, bem como no sentido invertido, a variável de controle.

Finalmente, utilizando-se as características de integrador (pela saída μ_E) e derivador (pela saída μ_{ECT}), conforme explicado na seção 3.3, foram realizadas simulações com o estimador paraconsistente dinâmico proposto em 5.4.2.

Os resultados obtidos se mostraram promissores e sugerem que uma rede neural paraconsistente possa ser empregada na aproximação de funções, podendo ser utilizada na identificação de sistemas para controle (I4C), quando modelos podem ser levantados com foco nas condições de trabalho, controle de malha fechada, e não genéricos para qualquer situação. Com isso a RNP, baseada na LPA2v pode ser considerada como parte da estratégia de controle de um sistema dinâmico. Além disso, redes de células CNAPP e CNAPap_{CTX} podem compor um estimador paraconsistente, para o tratamento e estimação de variáveis utilizadas no controle.

7. CONTROLE DE RIP POR RNPR: RESULTADOS E DISCUSSÃO

Este Capítulo descreve o controle do RIP através de rede neural paraconsistente recorrente (RNPR), fundamentada em LPA2v, utilizando como referência a técnica MRC descrita na seção 2.3.1 (figura 14), a identificação da planta e do controlador conforme apresentado na seção 6.3, com a diferença de as RNPs serem do tipo recorrente, com séries temporais de sinais e mais neurônios na camada oculta.

7.1. ARQUITETURA MRC COM RNPR

A figura 54 apresenta a arquitetura da estratégia de controle MRC com RNPR. De modo que apenas o controle seja efetivamente analisado, não foi implementado o estimador LPA2v, sendo que as mesmas variáveis estimadas pelo método da derivada e filtros LPF $[\theta_{0filt}; \theta_{1filt}; \dot{\theta}_{0filt}; \dot{\theta}_{1filt}]$ foram utilizadas para o aprendizado das RNPRs. Conforme observado na figura, o sistema é composto por 3 redes neurais paraconsistentes.

Figura 54 – Controle do RIP por Rede Neural Paraconsistente Recorrente.

Os sinais $[\theta_{0filt[k-1]}; \theta_{1filt[k-1]}; \dot{\theta}_{0filt[k-1]}; \dot{\theta}_{1filt[k-1]}; u_{[k-1]}]$, atrasados em uma amostra são ponderados (pesos), somados e o resultado é aplicado na entrada do grau de evidência favorável (μ) dos neurônios LPA2v da camada oculta da RNPR1. O

signal presente na saída da RNPR1 é atrasado em uma amostra e aplicado com peso de forma recorrente na entrada do grau de evidência desfavorável (λ) dos neurônios LPA2v da camada oculta. Com isso a RNPR1 deve estimar a próxima saída do modelo do braço do RIP ($\hat{\theta}_{0filt[k]}$). O sinal (e_B) consiste na diferença entre a saída real filtrada do braço do RIP (θ_{0filt}) e a saída estimada pela RNPR1 ($\hat{\theta}_{0filt}$). Ele irá somar com o sinal de referência (r) e será aplicado na entrada do controlador RNPR3.

A RNPR2 é semelhante à RNPR1, exceto pelo fato de que ela deve estimar a próxima saída do modelo do pêndulo do RIP ($\hat{\theta}_{1filt[k]}$). O sinal (e_P) consiste na diferença entre a saída real filtrada do braço do RIP (θ_{1filt}) e a saída estimada pela RNPR1 ($\hat{\theta}_{1filt}$), sendo aplicado à entrada do controlador RNPR3.

Os sinais $[\theta_{0filt[k-1]}; \theta_{1filt[k-1]}; \dot{\theta}_{0filt[k-1]}; \dot{\theta}_{1filt[k-1]}; (r[k-1] + e_{B[k-1]}); e_{P[k-1]}]$, atrasados em uma amostra, são ponderados (pesos), somados e o resultado é aplicado na entrada do grau de evidência favorável (μ) dos neurônios LPA2v da camada oculta da RNPR3. O sinal presente na saída da RNPR3 é atrasado em uma amostra e ponderado com peso e aplicado de forma recorrente na entrada do grau de evidência desfavorável (λ) dos neurônios LPA2v da camada oculta. Com isso a RNPR3 deve estimar a próxima saída ($\hat{u}_{[k]}$) do controlador.

A figura 55 apresenta a topologia das RNPRs utilizadas no controle do RIP, utilizando o conceito de redes NARX apresentado na seção 2.3.1, com sinais atrasados em uma amostra. As entradas $[p_1; p_2; \dots; p_m]$ correspondem aos sinais que serão aplicados pesos e somados. O resultado é combinado com um valor de *bias* $[b_{\mu,1}; b_{\mu,2}; \dots; b_{\mu,n}]$ e aplicado na porta μ de cada neurônio LPA2v, onde m é o número de entradas e n o número de neurônios na camada oculta. A entrada $[p_{FB}]$, proveniente da realimentação (*feedback*) da saída da rede neural paraconsistente, corresponde ao sinal de entrada que será aplicado um peso resultante do aprendizado da rede neural, somado a um *bias* $[b_{\lambda,1}; b_{\lambda,2}; \dots; b_{\lambda,n}]$ correspondente e aplicado na porta λ dos neurônios LPA2v na camada oculta.

Apesar de não ser mandatário na RNPR, neste estudo são treináveis apenas os neurônios na camada oculta, ou seja, pesos são aplicados entre a entrada e a camada oculta da rede. Não há pesos aplicados entre a camada oculta e a camada de saída. O neurônio de saída utiliza função linear simples, ou seja, é uma soma das saídas dos neurônios da camada oculta $[\mu_{ER1}; \mu_{ER2}; \dots; \mu_{ERn}]$ mais um *bias* $[b_2]$. As 3 redes

RNPR1, RNPR2 e RNPR3 possuem a mesma arquitetura. Apenas variam o número de entradas e quantidade de neurônios na camada oculta.

Figura 55 – Rede Neural Paraconsistente Recorrente (RNPR).

7.2. IDENTIFICAÇÃO DOS MODELOS DOS ÂNGULOS DO BRAÇO E PÊNDULO DO RIP

O primeiro passo da concepção do controle MRC é a identificação dos modelos dos ângulos braço e pêndulo. Pelo fato de o RIP ser altamente instável, assim como descrito na seção 5.3 e apresentado na seção 6.3, novamente o processo foi realizado em malha fechada utilizando como referência o controlador por alocação de polos.

Entretanto, diferentemente do apresentado anteriormente, aqui é utilizado um sinal pseudoaleatório de referência, conforme apresentado na figura 56.

Figura 56 – Sinal pseudoaleatório de referência (r) para a identificação do braço e pêndulo do RIP.

Importante destacar que o sinal de referência tem que ser programado no código do microcontrolador que exerce o controle do protótipo do RIP. A seguir é apresentado o código em linguagem C para a geração do sinal de referência.

```
// Referência para NARX
if (t < 5)
    r = 0;
else {
    if (t_2 > 5) {
        rnd=rand()% 2000;
        r=rnd/1000-1;
        t_2 = 0;
    }
    //r = r_aux;
    //r=0;
}
```

Assim, diferentemente do estudo apresentado na seção 6.3, a lei de controle utilizada com o sinal de referência conforme a equação 76.

$$K = [-0.2160 \quad -5.3459 \quad -0.2563 \quad -0.5934]$$

$$u_{[k]} = K[1,1]r_{[k]} - K_{[k]}x_{[k]} \quad (76)$$

Onde $K[1,1] = -0.2160$ e $x_{[k]} = [\theta_{0filt[k]}; \theta_{1filt[k]}; \dot{\theta}_{0filt[k]}; \dot{\theta}_{1filt[k]}]$.

A figura 57 apresenta os sinais resultantes da saída do RIP ($x_{[k]}$) e ($u_{[k]}$).

(a)

(b)

Figura 57 – Ângulo do braço (θ_0). Ângulo do pêndulo (θ_1). Velocidade do braço ($\dot{\theta}_0$). Velocidade do pêndulo ($\dot{\theta}_1$). Saída do controlador PP (u).

Conforme observado pela figura 57a, o braço do RIP tenta acompanhar o sinal de referência, apesar do atraso da resposta e da amplitude pico a pico ultrapassar o valor de $[-1;+1]$ devido a características mecânicas do sistema, folga (*backlash*), atraso de resposta, etc. Já na figura 57b é possível verificar que apesar de picos durante as transições do sinal de referência, o ângulo do pêndulo é mantido em equilíbrio, com pequena variação de ângulo em torno de 0 rad. A velocidade do braço e do pêndulo, calculadas através de derivada e filtro LPF também são apresentadas nas figuras 57c e 57d. Finalmente, a figura 57e apresenta o sinal de controle. Observa-se picos de $[-0.5;+0.5]$ nas transições do sinal de referência.

O aprendizado da RNPR1 e RNPR2 é realizado *offline*. Para isso os sinais são exportados em arquivo log. Como o tempo de amostragem $[T_s = 0.01s]$ utilizado pelo RIP é muito baixo, de modo a reduzir o tamanho do arquivo log, o programa no microcontrolador do RIP foi alterado para gerar um log a cada 10 amostras. Com isso, um arquivo de 100 segundos, possui apenas 1000 amostras ao invés de 10.000. A razão é diminuir o tempo necessário para a execução das épocas das redes neurais. Além disso, por serem redes recorrentes, a informação temporal é relevante e as amostras não são embaralhadas antes de serem utilizadas no treinamento das RNPs. Assim sendo, 500 amostras foram utilizadas para treinamento em 1000 épocas com taxa de aprendizagem de 0.5 e outras 500 para validação. A Tabela 7 apresenta os resultados de MSE para diferentes quantidades de neurônios para RNPR1 e RNPR2.

Tabela 7 – MSE para número variado de neurônios na camada oculta, na RNPR1 e RNPR2.

Rede Neural Paraconsistente Recorrente	Número de Neurônios na Camada Oculta	MSE Treinamento	MSE Validação
RNPR1	2	1.20×10^{-5}	6.01×10^{-4}
	4	1.31×10^{-6}	2.20×10^{-5}
	6	3.29×10^{-6}	3.40×10^{-5}
	8	9.76×10^{-7}	$1,77 \times 10^{-5}$
RNPR2	2	3.64×10^{-7}	3.54×10^{-7}
	4	4.30×10^{-7}	$5,85 \times 10^{-7}$
	6	1.02×10^{-6}	1.55×10^{-6}
	8	1.54×10^{-6}	6.56×10^{-6}

Como o algoritmo das redes neurais terá que ser criado como parte da programação do microcontrolador embarcado no sistema do RIP, a complexidade e

consequentemente o número de neurônios é uma preocupação. Observando-se a Tabela 7, para RNPR1, quatro neurônios ou mais resultam em MSE de validação menor do que com dois neurônios. Para a RNPR2, dois ou quatro neurônios resultaram em MSE de validação melhor do que com maior número de neurônios. Assim sendo, foram adotadas RNPR1 e RNPR2 com 4 neurônios na camada oculta.

Uma comparação entre RNPR1, estimador do braço do RIP, com quatro neurônios foi comparada com redes neurais equivalentes construídas com função de ativação tanh e LReLU com parâmetro alfa de 0.2. Uma comparação direta não é muito fácil, haja visto que cada função de ativação tem suas particularidades, como a questão de normalização do sinal de entrada, já que tanh e LReLU aceitam sinais positivos e negativos. Além disso, a função LReLU necessita de taxa de aprendizagem menor que a requerida pelas funções de ativação tanh e LPA2v, conforme resultados apresentados na seção 6.1. A Tabela 8 apresenta os resultados comparativos de MSE, considerando-se 500 épocas, 500 amostras e taxa de aprendizagem mais baixa, de 0.01 segundos.

Tabela 8 – Comparação entre RNPR1 e RNR com funções tanh e LReLU.

Rede Neural	Número de Neurônios na Camada Oculta	MSE Treinamento	MSE Validação
RNR tanh	4	1.17×10^{-4}	1.90×10^{-3}
RNR LReLU	4	1.62×10^{-5}	5.04×10^{-4}
RNRP1	4	1.63×10^{-4}	3.89×10^{-4}

Observa-se pela tabela que a despeito dos resultados obtidos de MSE durante o treinamento, a RNPR1 e RNR com LReLU ficaram muito próximas no MSE de validação e da ordem de 10 vezes menores que o da RNR utilizando função de ativação tanh.

A figura 58 apresenta a evolução do MSE de treinamento para a RNR utilizando função tanh (verde), LReLU (vermelho) e RNPR com função de ativação LPA2v (azul). Conforme esperado, a LReLU apresenta uma convergência de MSE mais rápida que a tanh nas primeiras 10 épocas. A RNPR apresenta uma curva de evolução do MSE muito próxima da RNR com LReLU. Importante lembrar que a taxa de aprendizagem foi reduzida de 0.5 para 0.01 para fins de comparação com a RNR com função de

ativação LReLU, sem o efeito de comportamento divergente por ultrapassar o ponto ideal (*overshoot*) durante o processo de aprendizado por gradiente descendente.

Figura 58 – Evolução do MSE de treinamento para RNPR com funções de ativação tanh e LReLU e RNPR1 com função de ativação LPA2v.

Os pesos aprendidos para a RNPR1 são apresentados na Tabela 9 enquanto que a Tabela 10 apresenta os pesos da RNPR2.

Tabela 9 – Pesos aprendidos pela RNPR1.

Neurônio	Entrada (μ)					Entrada (λ)
	p_1	p_2	p_3	p_4	p_5	p_{FB}
1	0.2698	0.1289	-0.0485	-0.0488	0.0063	0.3419
2	0.1973	-0.1716	0.0182	-0.0084	0.0447	0.3885
3	0.9220	0.3216	-0.0619	0.0136	0.0116	-0.2482
4	0.5885	-0.3051	0.1131	0.1131	-0.0552	-0.4971

Tabela 10 – Pesos aprendidos pela RNPR2.

Neurônio	Entrada (μ)					Entrada (λ)
	p_1	p_2	p_3	p_4	p_5	p_{FB}
1	-0.0907	0.7128	-0.2844	-0.2677	0.0375	-0.3226
2	0.1141	0.6093	0.2888	0.1719	0.1072	0.4281
3	0.2668	0.2033	-0.1315	-0.1345	0.0577	-0.4056
4	-0.2918	0.5321	0.1282	0.2537	-0.2059	0.3140

A linha 3 da Tabela 9, referente a um neurônio, pode ser interpretada da seguinte forma: ela contém a matriz de pesos $W_1 = [W_{\mu,1} \ W_{\lambda,1}] = [w_{1,1} \ w_{1,2} \ w_{1,3} \ w_{1,4} \ w_{1,5} \ w_{1,6}]$,

onde as 5 primeiras colunas correspondem aos pesos aplicados aos sinais presentes nas entradas p_1 a p_5 , ou $W_{\mu,1}$, que são somados e o resultado entregue à entrada μ , além do peso $w_{1,6}$, correspondente a $W_{\lambda,1}$, aplicado ao sinal de saída realimentado, presente na entrada p_{FB} , antes de ser entregue à entrada λ . As demais linhas referem-se aos pesos dos demais neurônios. A equação 77 refere-se ao cálculo das entradas do primeiro neurônio LPA2v das redes RNPR1. A mesma análise pode ser feita para a Tabela 10, referente à RNPR2, já que ambas as redes apresentam o mesmo número de entradas e neurônios na camada oculta. O cálculo da saída das redes RNPR1 e RNPR2 pode ser verificado na equação 78.

$$\begin{aligned}\mu_1 &= G_E W_{\mu,1} [p_1 \ p_2 \ p_3 \ p_4 \ p_5]^T + b_{\mu,1} \\ \mu_1 &= G_E (w_{1,1} \theta_{0filt[k-1]} + w_{1,2} \theta_{1filt[k-1]} + w_{1,3} \dot{\theta}_{0filt[k-1]} + w_{1,4} \dot{\theta}_{1filt[k-1]} + \dots \\ &\dots + w_{1,5} u_{[k-1]}) + b_{\mu,1}; \\ \lambda_1 &= G_E W_{\lambda,1} p_{FB} + b_{\lambda,1}; \\ \lambda_1 &= G_E w_{1,6} \mu_{ER1[k-1]} + b_{\lambda,1}\end{aligned}\tag{77}$$

$$\begin{aligned}\text{RNPR1} &= \hat{\theta}_{0filt[k]} = G_S (\mu_{ER1} - 0.5 + \mu_{ER2} - 0.5 + \mu_{ER3} - 0.5 + \mu_{ER4} - 0.5) + b_2 \\ \text{RNPR2} &= \hat{\theta}_{1filt[k]} = G_S (\mu_{ER1} - 0.5 + \mu_{ER2} - 0.5 + \mu_{ER3} - 0.5 + \mu_{ER4} - 0.5) + b_2\end{aligned}\tag{78}$$

Onde G_E é o ganho de normalização de entrada, no valor de 0.125 para todos os sinais de entrada da rede. Este valor foi definido pois observando-se a figura 57, vários sinais de entrada ultrapassam a amplitude pico a pico de 2, sendo que a velocidade pico a pico do braço é maior que 4 rad/seg. Como os sinais são simétricos em zero, pode-se adotar os valores de *bias* $b_{\mu,1}$ e $b_{\lambda,1}$ fixos em 0.5. O ganho de saída é G_S , no valor adotado de 4 e o *bias* de saída, b_2 , no valor de zero.

7.3. IDENTIFICAÇÃO E MODELAGEM DO CONTROLADOR POR REDE NEURAL PARA CONSISTENTE RECORRENTE

Com as redes RNPR1 e RNPR2 aprendidas, o próximo passo é implementá-las na programação do microcontrolador do RIP, executar novamente o programa com o sinal de referência pseudoaleatório, porém agora gerando também os sinais de erro, $e_B = \theta_{0filt} - \hat{\theta}_{0filt}$ e $e_P = \theta_{1filt} - \hat{\theta}_{1filt}$. A figura 59 apresenta os sinais e_B e e_P obtidos.

Apesar de não ter sido utilizado neste estudo, opcionalmente pode-se incluir um filtro LPF para os sinais e_B e principalmente e_P antes de serem aplicados à entrada da RNPR3. Como o pêndulo está em equilíbrio, o sinal e_P apresenta uma amplitude da ordem de 10 vezes menor que o sinal e_B .

Figura 59 – Sinais adicionais para treinamento da RNPR3, e_B (a) e e_P (b).

Novamente 500 amostras em 50 segundos, foram utilizadas para o treinamento da RNPR3 utilizada no controlador MRC, com taxa de aprendizagem de 0.5. Da mesma forma que para RNPR1 e RNPR2, também foram feitos testes com redes de 2 a 10 neurônios na camada oculta, sendo a configuração definida com 6 neurônios LPA2v. A Tabela 11 apresenta os resultados de MSE de treinamento e validação para as diferentes quantidades de neurônios LP2v na camada oculta. A figura 60 apresenta

o gráfico de evolução do MSE de treinamento e o resultado comparativo entre o sinal u medido e modelado pela RNPR3, utilizando as amostras de validação.

Tabela 11 – MSE para número variado de neurônios na camada oculta na RNPR3.

Rede Neural Paraconsistente Recorrente	Número de Neurônios na Camada Oculta	MSE Treinamento	MSE Validação
RNPR3	2	3.56×10^{-4}	3.90×10^{-4}
	4	3.51×10^{-4}	4.10×10^{-4}
	6	3.60×10^{-4}	4.61×10^{-4}
	8	3.84×10^{-4}	5.53×10^{-4}
	10	3.87×10^{-4}	6.60×10^{-4}

Figura 60 – Aprendizado RNPR3. Curva de MSE (a) e comparativo de amostras de validação entre o sinal u e a saída da RNPR3.

A Tabela 12 apresenta os pesos finais aprendidos pela RNPR3 na configuração adotada. O cálculo das entradas do primeiro neurônio LPA2v das redes RNPR3 são apresentados na equação 79. O cálculo da saída da rede RNPR3 pode ser verificado na equação 80.

Tabela 12 – Pesos aprendidos pela RNPR3.

Neurônio	Entrada (μ)						Entrada (λ)
	p_1	p_2	p_3	p_4	p_5	p_6	p_{FB}
1	0.4368	3.4771	1.2851	0.4147	-0.7061	0.6398	0.2175
2	0.4512	3.8034	-0.0598	1.0520	0.0685	-0.0421	-0.4752
3	-0.3405	4.3227	-0.1484	0.6035	0.3641	0.2019	0.7546
4	0.3516	4.1752	-0.3026	-0.0286	-0.3855	-0.026	-0.7899
5	0.0368	3.5782	-0.0430	0.7043	0.0923	0.0160	0.7939
6	-0.0053	4.2051	0.3989	0.9000	-0.3409	0.7512	-0.1762

$$\mu_1 = G_E W_{\mu,1} [p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ p_6]^T + b_{\mu,1}$$

$$\mu_1 = G_E (w_{1,1} \theta_{0filt[k-1]} + w_{1,2} \theta_{1filt[k-1]} + w_{1,3} \dot{\theta}_{0filt[k-1]} + w_{1,4} \dot{\theta}_{1filt[k-1]} + \dots \\ \dots + w_{1,5} (r_{B[k-1]} + e_{B[k-1]})) + b_{\mu,1};$$

$$\lambda_1 = G_E W_{\lambda,1} p_{FB} + b_{\lambda,1};$$

$$\lambda_1 = G_E w_{1,7} \mu_{ER1[k-1]} + b_{\lambda,1} \quad (79)$$

$$RNPR3 = \hat{\theta}_{1filt[k]} = G_S ((\mu_{ER1} - 0.5) + (\mu_{ER2} - 0.5) + (\mu_{ER3} - 0.5) + (\mu_{ER4} - 0.5) + \dots \\ \dots + (\mu_{ER5} - 0.5) + (\mu_{ER6} - 0.5)) + b_2 \quad (80)$$

Onde $G_E = 0.125$ para todos os sinais de entrada da rede. Como os sinais são simétricos em zero, pode-se adotar os valores de *bias* $b_{\mu,1}$ e $b_{\lambda,1}$ fixos em 0.5. O ganho de saída é G_S , no valor adotado de 4 e o *bias* de saída, b_2 , no valor de zero.

Após o treinamento da RNPR3 o controle por rede neural paraconsistente recorrente (CRNPR), baseado no MRC foi implementado no microcontrolador do sistema RIP, conforme a figura 54 apresentada na seção 7.1.

A figura 61 apresenta os resultados comparados do sistema RIP para uma referência onda retangular, com o controle PP e o novo controle CRNPR.

Figura 61 – Comparação controle PP e CRNPR. Ângulo do braço do RIP (θ_0). Ângulo do pêndulo do RIP (θ_1). Saída do controlador (u).

Os resultados foram gerados considerando-se todas as amostras, ou seja, uma amostra a cada T_s . Observa-se na figura 61a que o movimento do braço é mais uniforme quando o CRNPR (linha azul) está controlando o RIP, comparado quando o PP (linha vermelha) assume o controle. Apesar de ambos os controladores manterem o pêndulo em equilíbrio, a variação pico a pico do ângulo do pêndulo é maior quando controlado pelo PP (linha vermelha) do que pelo CRNPR (linha azul), conforme mostrado na figura 61b. De acordo com a figura 61c, o esforço do controlador também é menor com o CRNPR (linha azul), cujos valores pico a pico durante as transições da referência, é menor que com o controlador PP.

7.4. ANÁLISE DA PERFORMANCE DO CONTROLE CRNPR.

A figura 62 apresenta a comparação da densidade espectral de potência, gerada a partir da FTT do sinal de saída (u) dos controladores PP e CRNPR, com 2048 componentes elevadas ao quadrado.

Figura 62 –Gráfico da FFT da saída do controlador PP e CRNPR com LPA2v.

Observa-se que no caso do controlador CRNPR a potência está mais concentrada em torno de 0.45 Hz, ao passo que para o controlador PP ela está mais espalhada entre 0.8 e 1.5 Hz. Apesar da figura, o cálculo da integral das curvas da figura 62, como a soma de todas as componentes espectrais de potência, resultou em um valor de potência total 24.08 para o sinal de controle CRNPR contra 31.77 do controle PP, ou 24.21% menor.

A análise de robustez foi realizada através da aplicação de perturbações diretamente sobre o pêndulo durante o funcionamento. Importante destacar que o algoritmo do microcontrolador introduz uma restrição ao controlador, limitando a amplitude máxima do sinal u dentro do intervalo $[-1,+1]$ apenas. A figura 63 apresenta as saídas do controlador PP e dos ângulos do braço e pêndulo do RIP durante a aplicação de perturbações. O mesmo pode ser observado na figura 64 quando o controlador CRNPR é utilizado.

As perturbações no pêndulo com controle PP são aplicadas a partir da amostra 39500. A figura 63a mostra os picos de desvio do ângulo do braço do RIP, em um esforço de manter o equilíbrio do pêndulo, observado pelos picos do sinal apresentado na figura entre as amostras 39500 e 41500. Entretanto, uma perturbação ligeiramente mais forte faz com que o controlador não consiga recuperar o controle, perdendo totalmente o equilíbrio, conforme apresentado nas últimas amostras da figura 63b. A figura 63c apresenta o sinal de controle. As linhas tracejadas denotam o limite de controle estabelecido no algoritmo do microcontrolador do RIP. Observa-se entre as amostras 39500 e 41500 que para pequenas perturbações no pêndulo, ainda que o sinal de controle atinge o valor máximo, o controlador é capaz de recuperar o controle do pêndulo. Observa-se nas últimas amostras, próximo a 42000, que o sinal de controle atinge o valor máximo e permanece nesse platô, quando o sistema perde o controle do RIP, antes da intervenção do operador para desligar o sistema.

Conforme apresentado na figura 61 da seção 7.3, o RIP apresentou menores variações pico a pico do ângulo do pêndulo quando controlado pelo CRNPR do que pelo controlador PP. Isso posto, perturbações insistentes foram aplicadas ao sistema, com esforço do controlador em atuar no ângulo do braço de até 1.5 rad acima do sinal de referência em curso sem que o pêndulo perdesse o equilíbrio. Isso pode ser observado na figura 64a, onde as perturbações começam na amostra 55000 e atingem valores pico de +2.5 e -2.5 rad entre as amostras 66000 e 72000. Ainda assim, observa-se na figura 64b que o ângulo do pêndulo se desloca entre +0.15 e -0.25 rad. A figura 64c mostra que durante as perturbações o sinal de controle atinge o limite máximo $[-1,+1]$ e ainda assim consegue recuperar o controle do RIP.

Figura 63 – Perturbações aplicada ao pêndulo do RIP com controlador PP. Ângulo do braço (a), ângulo do pêndulo (b) e sinal de controle (c).

Figura 64 – Perturbações aplicada ao pêndulo do RIP com controlador CRNPR. Ângulo do braço (a), ângulo do pêndulo (b) e sinal de controle (c).

Os resultados apresentados indicam que, sob as mesmas condições, o pêndulo controlado pelo CRNPR apresentou menor esforço de controle e maior robustez às perturbações aplicadas diretamente sobre o pêndulo do que quando controlado pelo controlador PP.

7.5. SUMÁRIO DO CAPÍTULO

Este capítulo descreveu as etapas do projeto, implementação e verificação de um controle por rede neural paraconsistente recorrente, baseado na arquitetura MRC, para o controle do pêndulo invertido.

Resultados competitivos de aprendizado das redes neurais RNPR com função de ativação LPA2v contra RNAs equivalentes com funções $\tanh$ e LReLU foram apresentados.

O processo de identificação se deu pelo método NARX, com sinais atrasados em uma amostra. O controle é desenvolvido em duas etapas. A primeira etapa modela-se o sistema, no caso o RIP. Para a identificação dos modelos foi utilizado um sinal pseudoaleatório. A segunda etapa modela-se o controle. O CRNPR proposto para o RIP faz uso de 3 redes neurais paraconsistentes recorrentes. Uma para modelar o ângulo do braço do sistema e outra para o ângulo do pêndulo. A terceira aprende o sinal de controle.

Com as 3 redes treinadas, o controle é CRNPR é introduzido no código do microcontrolador do RIP. Resultados de robustez e menor esforço de controle foram analisados ainda neste capítulo.

O Apêndice C apresenta o algoritmo de aprendizado da RNPR desenvolvido para modelar o controlador. O Apêndice D apresenta o código introduzido para o controle CRNPR, baseado na arquitetura MRC, no algoritmo do microcontrolador do RIP.

8. CONCLUSÃO

A literatura registra a criação de redes neurais artificiais paraconsistentes, construídas com células neurais baseadas em LP. Entretanto, conforme apresentado neste documento, elas não aprendem no conceito de pesos entre as sinapses, mas utilizam limiares de transição e decisão, sendo utilizadas em diversas aplicações de classificação de sinais.

Esta investigação apresentou uma metodologia inédita para construir RNA utilizando as equações e regras de LPA2v, ou LPA evidencial, aqui denominada RNP. Diferentemente do registrado na literatura, os neurônios LPA2v descritos neste documento são treináveis através de pesos e *bias* aplicados aos sinais em suas entradas de grau de evidência favorável e desfavorável. A função de ativação LPA2v com duas entradas ortogonais permite uma maior flexibilidade e liberdade para o projetista distribuir os sinais entre as entradas e combinar os neurônios da rede, como no exemplo prático da RNPR.

Os resultados desta investigação demonstraram que uma rede neural paraconsistente, utilizando funções de ativação LPA2v nos neurônios treináveis, apresentou resultados competitivos quando contrastada com redes utilizando funções do tipo sigmoide e ReLU, nas mesmas condições e parâmetros.

As redes RNP utilizadas nesta investigação foram treinadas por *backpropagation* com gradiente descendente. Embora se possam investigar outras estratégias de aprendizagem, esse método provou ser bom o suficiente.

Os resultados obtidos aqui para o RIP sugerem que a RNP é uma alternativa para identificação e controle adaptativo de sistemas dinâmicos não lineares. Um controle por rede neural paraconsistente recorrente (CRNPR) foi desenvolvido e implementado, fundamentado em uma arquitetura MRC, como estado da arte em controle por redes neurais artificiais. O CRNPR foi capaz de assumir o controle de um pêndulo invertido rotativo, como sistema não linear, apresentando interessantes características de robustez e baixo esforço do controlador.

O algoritmo de neurônio paraconsistente requer algumas questões de decisão para calcular o ponto de ruptura de D e μ_{ER} , assim como as funções de ativação baseadas em ReLU. Embora uma avaliação do tempo de processamento não tenha sido realizada, o microcontrolador embarcado no sistema RIP executou três RNPR, sendo duas redes com quatro e uma com seis neurônios na camada oculta, durante

o intervalo de amostragem de 0.01 segundos, sem prejuízo para o controle do pêndulo, mesmo com todas as declarações *if-then* exigidas por cada neurônio LPA2v.

Com o arcabouço matemático apresentado neste estudo, os projetistas podem propor RNPs mais complexas, seja aumentando o número de células e camadas paraconsistentes, configurando uma rede neural paraconsistente recorrente (RNPR), combinando RNP com CNAPap e CNAPap_{CTX} para compor estimadores, integradores e derivadores ou construindo redes híbridas RNA+RNP de aprendizado profundo, permitindo trabalhar em aplicações de engenharia ainda mais desafiadoras.

8.1. TRABALHOS FUTUROS

A seguir são apresentados alguns pontos a serem estudados em trabalhos futuros:

- a) Novas técnicas para os estimadores de variáveis como a velocidade do braço e pêndulo com LPA2v, utilizando células PANCL_{CTX} e treinamento de RNPs;
- b) Outros métodos de aprendizagem para a RNP/RNPR;
- c) Avaliar a complexidade computacional, o impacto do atraso de processamento da RNP e comparação com outros métodos e funções de ativação de redes neurais;
- d) RNPR em outras aplicações e estratégias de controle, tais como IMC, MPC, etc.;
- e) RNPR com controlador adaptativo *on line* e aprendizado não supervisionado;
- f) Novas configurações de RNP, como rede neural profunda, combinada com lógica difusa, comparação com função de ativação radial, etc.;
- g) Novas aplicações para RNP e RNPR.

REFERÊNCIAS

- ABE, J. M. **Paraconsistent artificial neural networks: An introduction**. Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, KES 2004, 2004, p 20-25.
- ABE, J. M.; DA SILVA LOPES, H. F.; RENATO, A. **Paraconsistent Neurocomputing and Biological Signals Analysis**, Paraconsistent Intelligent-Based Systems, Intelligent Systems Reference Library, 2015., p 273-306.
- ABE, J. M.; NAKAMATSU, K. **Paraconsistent annotated evidential logic E_T and applications in automation and robotics**. In: The Handbook on Reasoning-Based Intelligent Systems, 2012, p 331-352.
- ABE, J. M.; NAKAMATSU, K.; SILVA FILHO, J. I. **Three decades of paraconsistent annotated logics: A review paper on some applications**. Procedia Computer Science. Anais, Procedia Computer Science, Vol. 159, 2019, p 1175-1181.
- ABIODUN, O. I.; *et al.* **State-of-the-art in artificial neural network applications: A survey**, Heliyon, Volume 4, Issue 11, 2018, 41 p.
- AGUILAR-AVELAR, C.; MORENO-VALENZUELA, J. **A composite controller for trajectory tracking applied to the Furuta pendulum**, ISA Transactions, Volume 57, 2015, p. 286-294.
- AGUILAR-AVELAR, C.; MORENO-VALENZUELA, J. **New feedback linearization-based control for arm trajectory tracking of the furuta pendulum**. IEEE/ASME Transactions on Mechatronics, 2016, p. 638-648.
- AGUIRRE, L. A. **Sistemas Dinâmicos Não Lineares: Conceitos e Análise de Dados**. eBook Amazon, 2020, 790 p.
- AGUIRRE, L. A.; *et al.* **Enciclopédia de automática: controle & automação**. [s.l.] Blucher, 2007.
- APICELLA, A.; DONNARUMMA, F.; ISGRÒ, F.; Prevete, R. **A survey on modern trainable activation functions**, Neural Networks, Volume 138, p. 14-32, 2021.
- BORGES DE LIMA, A. W.; *et al.* **Paraconsistent annotated logic artificial intelligence study in support of manager decision-making**. ACM International Conference Proceeding Series. Anais...2018, p. 154–157.
- CARNIELLI, W.; CONIGLIO, M. E. **Contradiction and (in)Consistency**. In: Paraconsistent Logic: Consistency, Contradiction and Negation. Logic, Epistemology, and the Unity of Science, vol 40, 2016, p. 1-28.
- CARNIELLI, W.; CONIGLIO, M. E. **LFIs Based on Other Logics**. In: Paraconsistent Logic: Consistency, Contradiction and Negation. Logic, Epistemology, and the Unity of Science, vol 40, 2016, p. 171-236.

CHANDRAN, D.; *et al.* **Model identification of rotary inverted pendulum using artificial neural networks**. 2015 Int. Conference on Recent Developments in Control, Automation and Power Engineering, RDCAPE 2015. Anais...2015, p. 146-150.

COELHO, M. S.; *et al.* **Hybrid PI controller constructed with paraconsistent annotated logic**. Control Engineering Practice, Volume 84, 2019, p. 112-124.

CRISTINA SILVA PAULO, K. **Identificação de padrões de sinais acústicos com base em classificação paraconsistente**. 2016, 78 p. Tese de Doutorado, Universidade de São Paulo, São Paulo, Brasil, 2016.

DA COSTA, N. C. A.; KRAUSE, D. A.; BUENO, O. A. **Paraconsistent Logics and Paraconsistency**. Handbook of the Philosophy of Science, 2007, p. 791-911.

DA CRUZ, J. M. **Controle Robusto Multivariável: O Método LQG/LTR**. São Paulo: EDUSP, 168 p., 1996.

DA SILVA FILHO, J. I.; ABE, J. M.; TORRES, G. L. **Inteligência Artificial com Redes de Análises Paraconsistentes**. 1ª edição, LTC, 314 p., 2008.

DA SILVA FILHO, J. I. **Treatment of Uncertainties with Algorithms of the Paraconsistent Annotated Logic**. Journal of Intelligent Learning Systems and Applications, vol. 4, nº 2, 2012, p. 144-153.

DA SILVA FILHO, J. I.; *et al.* **Support at Decision in Electrical Systems of subtransmission through selection of Topologies by a Paraconsistent Simulator**. IEEE Latin America Transactions, vol. 14, no. 4, 2016, p. 1993-1999.

DA SILVA FILHO, J. I.; LAMBERT-TORRES, G.; ABE, J. M. **Uncertainty treatment using paraconsistent logic: Introducing paraconsistent artificial neural networks**. Frontiers in Artificial Intelligence and Applications, 2010, 311 p.

DA SILVA FILHO, J. I. **Algorithms based on paraconsistent annotated logic for applications in expert systems**. Chapter 1, Expert System Software: Engineering, Advantages and Applications, vol. 211, 2012, 40 p.

DA SILVA FILHO, J. I.; ROCCO, A. **Power systems outage possibilities analysis by Paraconsistent logic**. 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, Pittsburgh, PA, 2008, p. 1-6.

DE CARVALHO, F. R.; ABE, J. M. **A paraconsistent decision-making method**. Springer, 2018, 225 p.

DE CARVALHO JR, A.; *et al.* **A Study of Paraconsistent Artificial Neural Cell of Learning Applied as PAL2v Filter**. IEEE Latin America Transactions, vol. 16, no. 1, 2018, p. 202-209.

DE CARVALHO JR, A.; *et al.* **Paraconsistent Logic Approach for Active Noise Reduction**. Journal of Mechatronics Engineering, v. 3, n. 1, 2020, p. 2–8.

DOS SANTOS, P. C. C. *et al.* **Paraconsistentes artificial neural networks applied to the study of mutational patterns of the F subtype of the viral strains of HIV-1 to antiretroviral therapy.** Anais da Academia Brasileira de Ciencias, vol 88, 2016, p.323-334.

DOYLE, J. C. **Guaranteed Margins for LQG Regulators.** IEEE Transactions on Automatic Control, Vol. 23 , Issue 4, p. 756-757, 1978.

EINI, R.; ABDELWAHED, S. **Indirect Adaptive fuzzy Controller Design for a Rotational Inverted Pendulum.** Proceedings of the American Control Conference. Anais...2018, p. 1677-1682.

EVERETT, M.; HABIBI. G.; and HOW, J. P. **Robustness Analysis of Neural Networks via Efficient Partitioning With Applications in Control Systems,** in IEEE Control Systems Letters, vol. 5, no. 6, pp. 2114-2119, 2021, doi: 10.1109/LCSYS.2020.3045323.

FARAGHER, R. **Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes],** IEEE Signal Processing Magazine, vol. 29, no. 5, 2012, p. 128-132.

FONSECA, E. S.; *et al.* **Acoustic investigation of speech pathologies based on the discriminative paraconsistent machine (DPM).** Biomedical Signal Processing and Control, Vol. 55, 2020.

FORMENTIN, S.; CHIUSO, A. **CoRe: Control-oriented Regularization for System Identification.** Proceedings of the IEEE Conference on Decision and Control. Anais...2019, p. 2253-2258.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Feedback Control of Dynamic Systems,** Pearson, 8^o edição, 2018, 918 p.

GANAPATHY SUBRAMANIAN, R.; ELUMALAI, V. K. **Robust MRAC augmented baseline LQR for tracking control of 2 DoF helicopter.** Robotics and Autonomous Systems, Vol. 86, 2016, p. 70-77.

GARCIA, D. V.; *et al.* **Analysis of Raman spectroscopy data with algorithms based on paraconsistent logic for characterization of skin cancer lesions.** Vibrational Spectroscopy, v. 103, 2019, 10 p.

GARG, A.; TAI, K.; PANDA, B. N. **System identification: Survey on modeling methods and models.** Advances in Intelligent Systems and Computing. In book: Artificial Intelligence and Evolutionary Computations in Engineering Systems, p.607-615, 2017.

GAUTAM, P. **System identification of nonlinear Inverted Pendulum using artificial neural network.** 2016 International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2016. Anais...2016, p. 1-5.

GEVERS, M. **Identification for control: From the early achievements to the revival**

of experiment design. European Journal of Control, Volume 11, Issues 4–5, 2005, p. 335-352.

GREEN, M.; LIMBEER, D. **Linear Robust Control.** Reprint of ed. [s.l.] Dover Publications, 2012, 538 p.

GUIDO, R. C.; *et al.* **Introducing the discriminative paraconsistent machine (DPM),** Information Sciences, Vol. 221, p. 389-402, 2013.

GYÖRGY, K. **The LQG Control Algorithms for Nonlinear Dynamic Systems.** Procedia Manufacturing, Procedia Manufacturing, Vol. 32, 2019, p. 553-563.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H.. **Neural Network Design,** Martin Hagan; 2^o edition, 2014, 802 p.

HAGAN, M. T.; DEMUTH, H. B.; DE JESÚS, O. **An introduction to the use of neural networks in control systems.** International Journal of Robust and Nonlinear Control, Vol.12, Issue11, 2002, p. 959-985.

HALICIOĞLU, R.; ÖKSÜZ, M. **Alternative Controller Design for Rotary Inverted Pendulum.** Tehnički glasnik, vol. 12, n^o 3, 2018, p. 139-145.

HAMZA, M. F.; *et al.* **Current development on using Rotary Inverted Pendulum as a benchmark for testing linear and nonlinear control algorithms,** Mechanical Systems and Signal Processing, Vol. 116, 2019, p. 347-369.

HANNACHI, M.; SOUDANI, D. **Internal model control of multivariable discrete-time systems.** Proceedings of 2015 7th International Conference on Modelling, Identification and Control, ICMIC 2015. Anais...2016, p. 1-6.

HE, W.; *et al.* **Neural-Learning-Based Control for a Constrained Robotic Manipulator With Flexible Joints.** IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 12, 2018, p. 5993-6003.

JAFARI, A. H.; HAGAN, M. T. **Application of new training methods for neural model reference control.** Engineering Applications of Artificial Intelligence, Vol. 74, 2018, p. 312-321.

JAMALI, A.; *et al.* **Intelligent modeling of double link flexible robotic manipulator using artificial neural network.** Journal of Vibroengineering, Vol. 20, Issue 2, 2018, p. 1021-1034.

JIANG, Y.; *et al.* **A brief review of neural networks based learning and control and their applications for robots,** Complexity, vol. 2017, 2017, 14 p.

KEESMAN, K. J. **System identification: An introduction.** In: Advanced Textbooks in Control and Signal Processing, 2011, 247 p.

KOZÁK, Š. **State-of-the-art in control engineering.** Journal of Electrical Systems and Information Technology, Vol 1, Issue 1, 2014, p. 1-9.

KRAFES, S.; CHALH, Z.; SAKA, A. **Review: Linear, nonlinear and intelligent controllers for the inverted pendulum problem**. Proceedings of 2016 International Conference on Electrical and Information Technologies, ICEIT 2016. Anais...2016, p. 136-141.

LATHI, B. P.; GREEN, R. **Linear Systems and Signals**. Oxford University Press, third edition, 2017, 1008 p.

LEWIS, F. L.; GE, S. S. **Neural Networks in Feedback Control Systems**. In: Mechanical Engineers' Handbook, vol. 2, third edition, p. 791-825.

LILLY, J. H. **Fuzzy Control and Identification**, 1st edition, Wiley, 2011, 250 p.

LIN, J.; LIAN, R. J. **Hybrid fuzzy-logic and neural-network controller for MIMO systems**. Mechatronics, Vol. 19, Issue 6, 2009, p. 972-986.

LIU, X.; JIA, R.; LIU, Q.; ZHAO, C. AND SUN, H. **Coastline Extraction Method Based on Convolutional Neural Networks—A Case Study of Jiaozhou Bay in Qingdao, China**, in IEEE Access, vol. 7, p. 180281-180291, 2019.

LJUNG, L. **System Identification: Theory for the User**. Prentice Hall, second edition, 1999, 610 p.

LJUNG, L. System identification. In: **The Control Systems Handbook: Control System Advanced Methods**, CRC Press, Second Edition, 198 p.

LJUNG, L. **Perspectives on system identification**. Annual Reviews in Control, Vol. 34, Issue 1, 2010, p 1-12.

MALHOTRA, R.; NARINDER SINGH; YADUVIR SINGH. **Soft Computing Techniques for Process Control Applications**. International Journal on Soft Computing, Vol.2, No.3, 2011, p. 32-44.

MARES, J.; DOLEZEL, P.; HRNCIRIK, P. **Artificial Neural Network – Possible Approach to Nonlinear System Control**. In: Artificial Neural Networks - Industrial and Control Engineering Applications, chapter 17, 2011, p. 359-376.

MARIO, M. C. **Modelo de análise de variáveis craniométricas através das redes neurais artificiais paraconsistentes**. Tese de Doutorado. Universidade de São Paulo, São Paulo, Brasil, 2006, 176 p.

MATHEW, N. J.; RAO, K. K.; SIVAKUMARAN, N. **Swing up and stabilization control of a rotary inverted pendulum**. IFAC Proceedings Volumes, Vol. 46, Issue 32, 2013, p. 654-659.

MEENAKSHI, R.; MANIMOZHI, M. **Adaptive Neuro fuzzy inference system controller design for single stage Inverted Pendulum**. 2016 International Conference on Computation of Power, Energy, Information and Communication, ICCPEIC 2016. Anais...2016, p. 472-476.

MENDEZ-FLORES, E.; *et al.* **Design, Implementation and Nonlinear Control Analysis of a Furuta Pendulum System**, 2019 4th International Conference on Control and Robotics Engineering (ICCRE), 2019, p. 65-69.

MINICZ, M. F.; MATUCK, G. R.; TASINAFFO, P. M.; DA SILVA FILHO, J. I. **Célula Neural Artificial Paraconsistente de Aprendizagem por Extração do Efeito da Contradição**. Revista Seleção Documental, p. 3–9, 2014 (portuguese). Available at: https://www.researchgate.net/publication/294259986_Celula_Neural_Artificial_Paraconsistente_de_Aprendizagem_por_Extracao_do_Efeito_da_Contradicao.

MISSENO DA CRUZ, C. *et al.* **Application of Paraconsistent Artificial Neural Network in Statistical Process Control acting on voltage level monitoring in Electrical Power Systems**, 18th International Conference on Intelligent System Application to Power Systems, ISAP 2015. Anais...2015, p. 1-6.

MOHAMMADIAN, M. **Modelling, control and prediction using hierarchical fuzzy logic systems: Design and development**. International Journal of Fuzzy System Applications, 2017, p. 105-123.

MORENO-VALENZUELA, J.; *et al.* **Adaptive Neural Network Control for the Trajectory Tracking of the Furuta Pendulum**. IEEE Transactions on Cybernetics, vol. 46, no. 12, 2016, p. 3439-3452.

NATH, K.; DEWAN, L. **A comparative analysis of linear quadratic regulator and sliding mode control for a rotary inverted pendulum**. 2018 International Conference on Recent Trends in Electrical, Control and Communication (RTECC), Malaysia, 2018, p. 302-307.

OTHMAN, C.; CHEIKH, I. B.; BOUZAOUACHE, H. **A nonlinear internal model controller for affine nonlinear discrete-Time systems**. 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, STA 2016 - Proceedings. Anais...2017, p. 48-53.

PARK, M.; KIM, Y. J.; LEE, J. J. **Swing-up and LQR stabilization of a rotary inverted pendulum**. Artificial Life and Robotics, vol 16, 2011, p. 94–97.

PAULO, K. C. S. **Identificação de Sinais Acústicos com Base em Classificação Paraconsistente**. Tese de Doutorado. Universidade de São Paulo, São Paulo, Brasil, 2016, 78 p.

PAWAR, R. J.; PARVAT, B. J. **Design and implementation of MRAC and modified MRAC technique for inverted pendulum**, 2015 International Conference on Pervasive Computing (ICPC), 2015, p. 1-6.

PERI, V. M.; SIMON, D. **Fuzzy logic control for an autonomous robot**. Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS. Anais...2005, p. 337-342.

PHAN, M. C.; HAGAN, M. T. **Error surface of recurrent neural networks**. IEEE Transactions on Neural Networks and Learning Systems, vol 24, issue 11, 2013, p.

1709-1721.

PRASAD, L. B.; TYAGI, B.; GUPTA, H. O. **Optimal control of nonlinear inverted pendulum system using PID controller and LQR: Performance analysis without and with disturbance input.** International Journal of Automation and Computing, vol 11, 2014, p. 661–670.

PRETORIUS, C. J.; DU PLESSIS, M. C.; GONSALVES, J. W. **Neuroevolution of Inverted Pendulum Control: A Comparative Study of Simulation Techniques.** Journal of Intelligent and Robotic Systems: Theory and Applications, vol. 86, 2017, p. 419–445.

PRIEST, G.; KOJI, T.; WEBER, Z. **Paraconsistent Logic**, The Stanford Encyclopedia of Philosophy (Summer 2018 Edition), Edward N. Zalta (ed.), URL = <<https://plato.stanford.edu/archives/sum2018/entries/logic-paraconsistent/>>.

RAJA GENAL, R. B. **Nonlinear System Identification and Control: Rop Scheme and Neural Networks.** Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018. Anais...2019, p. 1831-1836.

ROOSE, A. I.; YAHYA, S.; AL-RIZZO, H. **Fuzzy-logic control of an inverted pendulum on a cart.** Computers and Electrical Engineering, vol. 61, 2017, p. 31-47.

SARKAR, T. T.; DEWAN, L. **Pole-placement, PID and genetic algorithm based stabilization of inverted pendulum.** 8th Int. Conference on Computing, Communications and Networking Technologies, ICCCNT 2017. Anais...2017, p. 1-6.

SEMAN, P.; *et al.* **Swinging up the furuta pendulum and its stabilization via model predictive control.** Journal of Electrical Engineering, vol. 64, nº 3, 2013, p. 152-158.

SHALABY, R.; EL-HOSSAINY, M.; ABO-ZALAM, B. **Nonlinear SPKF-Based Time-Varying LQG for Inverted Pendulum System.** Arabian Journal for Science and Engineering, vol. 44, 2019, p. 6783–6793.

SHEHU, M.; *et al.* **LQR, double-PID and pole placement stabilization and tracking control of single link inverted pendulum.** Proceedings - 5th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2015. Anais...2016, p. 218-223.

SHI, H.; XU, Z.; SUN, T. and WU, C. **Double Inverted Pendulum System Control Based on Internal Model Principle**, 2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), Jinzhou, China, 2019, pp. 666-670.

SHIHABUDHEEN, K. V.; PILLAI, G. N. **Recent advances in neuro-fuzzy system: A survey.** Knowledge-Based Systems, Vol. 152, 2018, p. 136-162.

SHOJAEI, A. A.; OTHMAN, Mohd. F.; RAHMANI, R. and RANI, M. R. **Implementation of recurrent neural network to control rotational inverted pendulum using IMC scheme.** Australian Journal of Basic and Applied Sciences, 6 (7). p. 299-306, 2012.

SINGH, N.; YADAV, S. **Comparison of LQR and PD controller for stabilizing Double Inverted Pendulum System**. International Journal of Engineering, Vol. 1, Issue 12, 2012, p. 69-74.

SONODA, S.; MURATA, N. **Neural network with unbounded activation functions is universal approximator**. Applied and Computational Harmonic Analysis, Vol. 43, Issue 2, 2017, p. 233-268.

TUTUNJI, T. A. **Parametric system identification using neural networks**. Applied Soft Computing, Vol. 47, 2016, p. 251-261.

VALADE, A. *et al.* **A study about kalman filters applied to embedded sensors**. Sensors (Switzerland), 2017, 18 p.

VAN WILLIGENBURG, L. G.; DE KONING, W. L. **Numerical algorithms and issues concerning the discrete-time optimal projection equations**. European Journal of Control, Vol. 6, Issue 1, 2000, p. 93-110.

VINODH KUMAR, E.; JEROME, J. **Robust LQR controller design for stabilizing and trajectory tracking of inverted pendulum**. Procedia Engineering, Vol. 64, 2013, p. 169-178.

WU, R. T.; JAHANSHAHI, M. R. **Deep convolutional neural network for structural dynamic response estimation and system identification**. Journal of Engineering Mechanics, vol. 145, issue 1, 2019, 25 p.

YANG, C.; *et al.* **Neural network-based motion control of an underactuated wheeled inverted pendulum model**. IEEE Transactions on Neural Networks and Learning Systems, 2014, p. 2004-2016.

ZAMANSKY, A. **On recent applications of paraconsistent logic: an exploratory literature review**, Journal of Applied Non-Classical Logics, Vol 29, issue 4, 2019, p. 382-391.

ZHANG, T.; TAN, Y.; MA, F. **Nonlinear internal model control based on fuzzy rough granular Neural Networks**. 26th Chinese Control and Decision Conference, CCDC 2014. Anais...2014, p. 757-761.

ZHANG, X.; *et al.* **Neural network controller design for uncertain nonlinear systems based on backstepping control algorithm**. Proceedings of the 29th Chinese Control and Decision Conference, CCDC 2017. Anais...2017, p. 1623-1627.

ZHAO, Y.; DENG, B.; WANG, Z. **Analysis and study of perceptron to solve XOR problem**. Proceedings - 2nd International Workshop on Autonomous Decentralized System, IWADS 2002. Anais...2002, p. 168-173.

ZHENG, Y.; ZHONG, P.; YUE, Q. **Double Inverted Pendulum Based on LQG Optimal Control**. Conference: 2016 International Conference on Automatic Control and Information Engineering, Anais...2016, p. 83-87.

APÊNDICE A – TRABALHOS PUBLICADOS

7. DE CARVALHO JR, A. JUSTO, J. F.; ANGELICO, B. A.; DA SILVA FILHO, J. I. **Rotary Inverted Pendulum Identification for Control by Paraconsistent Neural Network**, IEEE Access, vol. 9, pp. 74155-74167, 2021. DOI: 10.1109/ACCESS.2021.3080176.
6. DE OLIVEIRA, A. M.; DE OLIVEIRA, A. M.; PEROTONI, M. B.; NURHAYATI, N.; BAUDRAND, H.; DE CARVALHO JR, A.; JUSTO, J. F. **A Fern Antipodal Vivaldi Antenna for Near-Field Microwave Imaging Medical Applications**, in IEEE Transactions on Antennas and Propagation, 13 p, 2021, DOI: 10.1109/TAP.2021.3096942.
5. DE CARVALHO JR, A.; JUSTO, J. F.; DE OLIVEIRA, A. M.; DE OLIVEIRA, C. A. S.; DA SILVA FILHO, J. I. **Paraconsistent Logic Approach for Active Noise Reduction**, Journal of Mechatronics Engineering, v. 3, n. 1, 2020, p. 2–8.
4. DE CARVALHO JUNIOR, A. **Scilab - Uma abordagem prática aplicada a problemas reais da engenharia: Programação no Scilab** – Sistemas Especialistas Paraconsistentes com Scilab. In: De Oliveira, A. M. (Org), 2ª Edição, São Paulo, Laboratório Maxwell, Artesam Editora, 2020, p. 161-209.
3. GARCIA, D. V.; DA SILVA FILHO, J. I.; SILVEIRA JUNIOR, L.; PECHECO, M. T. T.; ABE, J. M.; DE CARVALHO JUNIOR, A.; BLOS, M. F.; PASQUALUCCI, C. A. G.; MARIO, M. C. **Analysis of Raman spectroscopy data with algorithms based on paraconsistent logic for characterization of skin cancer lesions**, Vibrational Spectroscopy, v. 103, 2019, 10 p.
2. COELHO, M. S.; DA SILVA FILHO, J. I.; CÔRTEZ, H. M.; DE CARVALHO JUNIOR, A.; BLOS, M. F.; MARIO, M. C.; ROCCO, A. **Hybrid PI controller constructed with paraconsistent annotated logic**, Control Engineering Practice, Volume 84, 2019, p. 112-124.
1. DE CARVALHO JR, A.; DA SILVA FILHO, J. I.; MARIO, M. C.; BLOS, M. F.; CRUZ, C. M. **A Study of Paraconsistent Artificial Neural Cell of Learning Applied as PAL2v Filter**, IEEE Latin America Transactions, vol. 16, no. 1, 2018, p. 202-209.

APÊNDICE B – CÓDIGO DA RNP E RNA PARA PROBLEMA EX-OR

```
% Comparação entre NNPAL x NN (Log-sig, tanh, ReLU e LReLU) para XOR Problem
% Exemplo Porta Ou Exclusivo - 4 neurônios não lineares
% sendo 3 neurônios camada oculta e um na camada de saída
% Redes com pouca ou nenhuma simplificação (matrizes) para avaliação da complexidade de script
% Referência: https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

close all; clear all; clc;

set(0,'defaultfigurecolor',[1 1 1]);

a1=1; % Entrada 1 OU-EX
a2=0; % Entrada 2 OU-EX
Yd=1; % Entrada 3 OU-EX

R=1000;
lr=0.01; % mesma taxa de aprendizagem para todas as redes neurais

% Pesos Aleatórios da Rede Neural
w1=rand; w2=rand; w3=rand; w4=rand; w5=rand;
w6=rand; w7=rand; w8=rand; w9=rand;

% *****
% Rede Neural Perceptron Log-sigmoid com 3 Neuronios Ocultos
w11(1)=w1; w12(1)=w2; w13(1)=w3;
w21(1)=w4; w22(1)=w5; w23(1)=w6;
w31(1)=w7; w32(1)=w8; w33(1)=w9;

for N=1:R;
% Cálculo da Soma da Camada Escondida
H11(N)=w11(N)*a1+w21(N)*a2;
H12(N)=w12(N)*a1+w22(N)*a2;
H13(N)=w13(N)*a1+w23(N)*a2;

% Cálculo da Camada Escondida
Hy1(N)=sigmf(H11(1),[1,0]);
Hy2(N)=sigmf(H12(1),[1,0]);
Hy3(N)=sigmf(H13(1),[1,0]);

% Cálculo da Camada de Saída
H21(N)=w31(N)*Hy1(N);
H22(N)=w32(N)*Hy2(N);
H23(N)=w33(N)*Hy3(N);
Y(N)=sigmf((H21(N)+H22(N)+H23(N)),[1,0]);

% Cálculo do Erro
MSE(N)=0.5*(Yd-Y(N))^2;
dMSE(N)=(Y(N)-Yd);

% Ajusta Pesos
w31(N+1)=w31(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*Hy1(N);
w32(N+1)=w32(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*Hy2(N);
w33(N+1)=w33(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*Hy3(N);
w11(N+1)=w11(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*w31(N)*Hy1(N)*(1-Hy1(N))*a1;
w21(N+1)=w21(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*w31(N)*Hy1(N)*(1-Hy1(N))*a2;
w12(N+1)=w12(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*w32(N)*Hy2(N)*(1-Hy2(N))*a1;
w22(N+1)=w22(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*w32(N)*Hy2(N)*(1-Hy2(N))*a2;
w13(N+1)=w13(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*w33(N)*Hy3(N)*(1-Hy3(N))*a1;
```

```
w23(N+1)=w23(N)-lr*dMSE(N)*(Y(N)*(1-Y(N)))*w33(N)*Hy3(N)*(1-Hy3(N))*a2;
end
```

```
% *****
```

```
% Rede Neural Perceptron tanh com 3 Neuronios Ocultos
```

```
w11t(1)=w1; w12t(1)=w2; w13t(1)=w3;
```

```
w21t(1)=w4; w22t(1)=w5; w23t(1)=w6;
```

```
w31t(1)=w7; w32t(1)=w8; w33t(1)=w9;
```

```
aa1=2*a1-1; aa2=2*a2-1; % Normalização entradas
```

```
for N=1:R;
```

```
% Cálculo da Soma da Camada Escondida
```

```
H11t(N)=w11t(N)*a1+w21t(N)*aa2;
```

```
H12t(N)=w12t(N)*a1+w22t(N)*aa2;
```

```
H13t(N)=w13t(N)*a1+w23t(N)*aa2;
```

```
% Cálculo da Camada Escondida
```

```
Hy1t(N)=tanh(H11t(1));dHy1t(N)=1-Hy1t(N)^2;
```

```
Hy2t(N)=tanh(H12t(1));dHy2t(N)=1-Hy2t(N)^2;
```

```
Hy3t(N)=tanh(H13t(1));dHy3t(N)=1-Hy3t(N)^2;
```

```
% Cálculo da Camada de Saída
```

```
H21t(N)=w31t(N)*Hy1t(N);
```

```
H22t(N)=w32t(N)*Hy2t(N);
```

```
H23t(N)=w33t(N)*Hy3t(N);
```

```
Yt(N)=(tanh(H21t(N)+H22t(N)+H23t(N))+1)/2;
```

```
dYt(N)=0.5*(1-Yt(N)^2);
```

```
% Cálculo do Erro
```

```
MSEt(N)=0.5*(Yd-Yt(N))^2;
```

```
dMSEt(N)=(Yt(N)-Yd);
```

```
% Ajusta Pesos
```

```
w31t(N+1)=w31t(N)-lr*dMSEt(N)*dYt(N)*Hy1t(N);
```

```
w32t(N+1)=w32t(N)-lr*dMSEt(N)*dYt(N)*Hy2t(N);
```

```
w33t(N+1)=w33t(N)-lr*dMSEt(N)*dYt(N)*Hy3t(N);
```

```
w11t(N+1)=w11t(N)-lr*dMSEt(N)*dYt(N)*w31t(N)*dHy1t(N)*aa1;
```

```
w21t(N+1)=w21t(N)-lr*dMSEt(N)*dYt(N)*w31t(N)*dHy1t(N)*aa2;
```

```
w12t(N+1)=w12t(N)-lr*dMSEt(N)*dYt(N)*w32t(N)*dHy2t(N)*aa1;
```

```
w22t(N+1)=w22t(N)-lr*dMSEt(N)*dYt(N)*w32t(N)*dHy2t(N)*aa2;
```

```
w13t(N+1)=w13t(N)-lr*dMSEt(N)*dYt(N)*w33t(N)*dHy3t(N)*aa1;
```

```
w23t(N+1)=w23t(N)-lr*dMSEt(N)*dYt(N)*w33t(N)*dHy3t(N)*aa2;
```

```
end
```

```
% *****
```

```
% Rede Neural Perceptron RELU com 3 Neuronios Ocultos
```

```
w11r(1)=w1; w12r(1)=w2; w13r(1)=w3;
```

```
w21r(1)=w4; w22r(1)=w5; w23r(1)=w6;
```

```
w31r(1)=w7; w32r(1)=w8; w33r(1)=w9;
```

```
for N=1:R;
```

```
% Cálculo da Soma da Camada Escondida
```

```
x11(N)=w11r(N)*a1+w21r(N)*a2;
```

```
x12(N)=w12r(N)*a1+w22r(N)*a2;
```

```
x13(N)=w13r(N)*a1+w23r(N)*a2;
```

```
% Cálculo da Camada Escondida
```

```
if x11>0;
```

```
Hy1(N)=x11(N);
```

```

    dHyr1(N)=1;
else
    Hyr1(N)=0;
    dHyr1(N)=0;
end

if x12>0;
    Hyr2(N)=x12(N);
    dHyr2(N)=1;
else
    Hyr2(N)=0;
    dHyr2(N)=0;
end

if x13>0;
    Hyr3(N)=x13(N);
    dHyr3(N)=1;
else
    Hyr3(N)=0;
    dHyr3(N)=0;
end

% Cálculo da Camada de Saída
Hr21(N)=w31r(N)*Hyr1(N);
Hr22(N)=w32r(N)*Hyr2(N);
Hr23(N)=w33r(N)*Hyr3(N);
Hr(N)=Hr21(N)+Hr22(N)+Hr23(N);
if Hr(N)>0;
    Yr(N)=Hr(N);
    dYr(N)=1;
else
    Yr(N)=0;
    dYr(N)=0;
end

% Cálculo do Erro
MSEr(N)=0.5*(Yd-Yr(N))^2;
dMSEr(N)=(Yr(N)-Yd);

% Ajusta Pesos
w31r(N+1)=w31r(N)-lr*dMSEr(N)*dYr(N)*Hyr1(N);
w32r(N+1)=w32r(N)-lr*dMSEr(N)*dYr(N)*Hyr2(N);
w33r(N+1)=w33r(N)-lr*dMSEr(N)*dYr(N)*Hyr3(N);
w11r(N+1)=w11r(N)-lr*dMSEr(N)*dYr(N)*w31r(N)*dHyr1(N)*a1;
w21r(N+1)=w21r(N)-lr*dMSEr(N)*dYr(N)*w31r(N)*dHyr1(N)*a2;
w12r(N+1)=w12r(N)-lr*dMSEr(N)*dYr(N)*w32r(N)*dHyr2(N)*a1;
w22r(N+1)=w22r(N)-lr*dMSEr(N)*dYr(N)*w32r(N)*dHyr2(N)*a2;
w13r(N+1)=w13r(N)-lr*dMSEr(N)*dYr(N)*w33r(N)*dHyr3(N)*a1;
w23r(N+1)=w23r(N)-lr*dMSEr(N)*dYr(N)*w33r(N)*dHyr3(N)*a2;
end

% *****
% Rede Neural Perceptron LRELU com 3 Neuronios Ocultos
w11l(1)=w1; w12l(1)=w2; w13l(1)=w3;
w21l(1)=w4; w22l(1)=w5; w23l(1)=w6;
w31l(1)=w7; w32l(1)=w8; w33l(1)=w9;
alfa=0.05;

for N=1:R;
% Cálculo da Soma da Camada Escondida

```

```

x11l(N)=w11l(N)*a1+w21l(N)*a2;
x12l(N)=w12l(N)*a1+w22l(N)*a2;
x13l(N)=w13l(N)*a1+w23l(N)*a2;

% Cálculo da Camada Escondida
if x11l>=0;
    Hyl1(N)=x11l(N);
    dHyl1(N)=1;
else
    Hyl1(N)=alfa*x11l(N);
    dHyl1(N)=alfa;
end

if x12l>=0;
    Hyl2(N)=x12l(N);
    dHyl2(N)=1;
else
    Hyl2(N)=alfa*x12l(N);
    dHyl2(N)=alfa;
end

if x13l>=0;
    Hyl3(N)=x13l(N);
    dHyl3(N)=1;
else
    Hyl3(N)=alfa*x13l(N);
    dHyl3(N)=alfa;
end

% Cálculo da Camada de Saída
HI21(N)=w31l(N)*Hyl1(N);
HI22(N)=w32l(N)*Hyl2(N);
HI23(N)=w33l(N)*Hyl3(N);
HI(N)=HI21(N)+HI22(N)+HI23(N);
if HI(N)>=0;
    Yl(N)=HI(N);
    dYl(N)=1;
else
    Yl(N)=alfa*HI(N);
    dYl(N)=alfa;
end

% Cálculo do Erro
MSEl(N)=0.5*(Yd-Yl(N))^2;
dMSEl(N)=(Yl(N)-Yd);

% Ajusta Pesos
w31l(N+1)=w31l(N)-lr*dMSEl(N)*dYl(N)*Hyl1(N);
w32l(N+1)=w32l(N)-lr*dMSEl(N)*dYl(N)*Hyl2(N);
w33l(N+1)=w33l(N)-lr*dMSEl(N)*dYl(N)*Hyl3(N);
w11l(N+1)=w11l(N)-lr*dMSEl(N)*dYl(N)*w31l(N)*dHyl1(N)*a1;
w21l(N+1)=w21l(N)-lr*dMSEl(N)*dYl(N)*w31l(N)*dHyl1(N)*a2;
w12l(N+1)=w12l(N)-lr*dMSEl(N)*dYl(N)*w32l(N)*dHyl2(N)*a1;
w22l(N+1)=w22l(N)-lr*dMSEl(N)*dYl(N)*w32l(N)*dHyl2(N)*a2;
w13l(N+1)=w13l(N)-lr*dMSEl(N)*dYl(N)*w33l(N)*dHyl3(N)*a1;
w23l(N+1)=w23l(N)-lr*dMSEl(N)*dYl(N)*w33l(N)*dHyl3(N)*a2;
end

% *****
%Rede Neural Paraconsistente com 3 Neurônios Ocultos e 1 na saída

```

```

w11p(1)=w1; w12p(1)=w2; w13p(1)=w3;
w21p(1)=w4; w22p(1)=w5; w23p(1)=w6;
w31p(1)=w7; w32p(1)=w8; w33p(1)=w9;

```

```

for N=1:R;
    if w11p(N)*a1<0;
        Mi11=0;
    elseif w11p(N)*a1>1;
        Mi11=1;
    else
        Mi11=w11p(N)*a1;
    end

    if w12p(N)*a1<0;
        Mi12=0;
    elseif w12p(N)*a1>1;
        Mi12=1;
    else
        Mi12=w12p(N)*a1;
    end

    if w13p(N)*a1<0;
        Mi13=0;
    elseif w13p(N)*a1>1;
        Mi13=1;
    else
        Mi13=w13p(N)*a1;
    end

    if w21p(N)*a2<0;
        Lbda11=0;
    elseif w21p(N)*a2>1;
        Lbda11=1;
    else
        Lbda11=w21p(N)*a2;
    end

    if w22p(N)*a2<0;
        Lbda12=0;
    elseif w22p(N)*a2>1;
        Lbda12=1;
    else
        Lbda12=w22p(N)*a2;
    end

    if w23p(N)*a2<0;
        Lbda13=0;
    elseif w23p(N)*a2>1;
        Lbda13=1;
    else
        Lbda13=w23p(N)*a2;
    end

    % Célula Oculta 11
    Gc11=Mi11-Lbda11;
    Gct11=Mi11+Lbda11-1;
    D11=((1-abs(Gc11))^2+Gct11^2)^0.5;
    if Gc11>0.00001;
        Gcr11=1-D11;
        Mier11=(Gcr11+1)/2;
    end
end

```

```

    dMi11=(1-Mi11)/((2*Mi11^2+2*Lbda11^2-4*Mi11+2)^0.5);
    dLbda11=-Lbda11/((2*Mi11^2+2*Lbda11^2-4*Mi11+2)^0.5);
elseif Gc11<-0.00001;
    Gcr11=D11-1;
    Mier11=(Gcr11+1)/2;
    dMi11=Mi11/((2*Mi11^2+2*Lbda11^2-4*Lbda11+2)^0.5);
    dLbda11=(Lbda11-1)/((2*Mi11^2+2*Lbda11^2-4*Lbda11+2)^0.5);
else
    Gcr11=0;
    Mier11=0.5;
    dMi11=1/(2*(2*Mi11^2+2*Lbda11^2-4*Mi11+2)^0.5);
    dLbda11=-1/(2*(2*Mi11^2+2*Lbda11^2-4*Lbda11+2)^0.5);
end
if D11>=1;
    D11=1;
    Gcr11=0;
    Mier11=0.5;
    dMi11=1/(2*(2*Mi11^2+2*Lbda11^2-4*Mi11+2)^0.5);
    dLbda11=-1/(2*(2*Mi11^2+2*Lbda11^2-4*Lbda11+2)^0.5);
else
end
    % Célula Oculta 12
Gc12=Mi12-Lbda12;
Gct12=Mi12+Lbda12-1;
D12=((1-abs(Gc12))^2+Gct12^2)^0.5;
if Gc12>0.00001;
    Gcr12=1-D12;
    Mier12=(Gcr12+1)/2;
    dMi12=(1-Mi12)/((2*Mi12^2+2*Lbda12^2-4*Mi12+2)^0.5);
    dLbda12=-Lbda12/((2*Mi12^2+2*Lbda12^2-4*Mi12+2)^0.5);
elseif Gc12<-0.00001;
    Gcr12=D12-1;
    Mier12=(Gcr12+1)/2;
    dMi12=Mi12/((2*Mi12^2+2*Lbda12^2-4*Lbda12+2)^0.5);
    dLbda12=(Lbda12-1)/((2*Mi12^2+2*Lbda12^2-4*Lbda12+2)^0.5);
else
    Gcr12=0;
    Mier12=0.5;
    dMi12=1/(2*(2*Mi12^2+2*Lbda12^2-4*Mi12+2)^0.5);
    dLbda12=-1/(2*(2*Mi12^2+2*Lbda12^2-4*Lbda12+2)^0.5);
end
if D12>=1;
    D12=1;
    Gcr12=0;
    Mier12=0.5;
    dMi12=1/(2*(2*Mi12^2+2*Lbda12^2-4*Mi12+2)^0.5);
    dLbda12=-1/(2*(2*Mi12^2+2*Lbda12^2-4*Lbda12+2)^0.5);
else
end
    % Célula Oculta 13
Gc13=Mi13-Lbda13;
Gct13=Mi13+Lbda13-1;
D13=((1-abs(Gc13))^2+Gct13^2)^0.5;
if Gc13>0.00001;
    Gcr13=1-D13;
    Mier13=(Gcr13+1)/2;
    dMi13=(1-Mi13)/((2*Mi13^2+2*Lbda13^2-4*Mi13+2)^0.5);
    dLbda13=-Lbda13/((2*Mi13^2+2*Lbda13^2-4*Mi13+2)^0.5);
elseif Gc13<-0.00001;
    Gcr13=D13-1;

```

```

Mier13=(Gcr13+1)/2;
dMi13=Mi13/((2*Mi13^2+2*Lbda13^2-4*Lbda13+2)^0.5);
dLbda13=(Lbda13-1)/((2*Mi13^2+2*Lbda13^2-4*Lbda13+2)^0.5);
else
    Gcr13=0;
    Mier13=0.5;
    dMi13=1/(2*(2*Mi13^2+2*Lbda13^2-4*Mi13+2)^0.5);
    dLbda13=-1/(2*(2*Mi13^2+2*Lbda13^2-4*Lbda13+2)^0.5);
end
if D13>=1;
    D13=1;
    Gcr13=0;
    Mier13=0.5;
    dMi13=1/(2*(2*Mi13^2+2*Lbda13^2-4*Mi13+2)^0.5);
    dLbda13=-1/(2*(2*Mi13^2+2*Lbda13^2-4*Lbda13+2)^0.5);
else
end
% Camada Intermediaria CNAP Linear 21
if w31p(N)*Mier11<0;
    Mi21=0;
elseif w31p(N)*Mier11>1;
    Mi21=1;
else
    Mi21=w31p(N)*Mier11;
end

if w32p(N)*Mier12<0;
    Lbda21=0;
elseif w32p(N)*Mier12>1;
    Lbda21=1;
else
    Lbda21=w32p(N)*Mier12;
end

Mie21=(Mi21-Lbda21+1)/2;
dMi21=0.5;
dLbda21=-0.5;

% Célula Saída Mier31
Mi31=Mie21;

if w33p(N)*Mier13<0;
    Lbda31=0;
elseif w33p(N)*Mier13>1;
    Lbda31=1;
else
    Lbda31=w33p(N)*Mier13;
end

Gc31=Mi31-Lbda31;
Gct31=Mi31+Lbda31-1;
D31=((1-abs(Gc31))^2+Gct31^2)^0.5;
if Gc31>0.00001;
    Gcr31=1-D31;
    Mier31=(Gcr31+1)/2;
    dMi31=(1-Mi31)/((2*Mi31^2+2*Lbda31^2-4*Mi31+2)^0.5);
    dLbda31=-Lbda31/((2*Mi31^2+2*Lbda31^2-4*Mi31+2)^0.5);
elseif Gc31<-0.00001;
    Gcr31=D31-1;
    Mier31=(Gcr31+1)/2;

```

```

dMi31=Mi31/((2*Mi31^2+2*Lbda31^2-4*Lbda31+2)^0.5);
dLbda31=(Lbda31-1)/((2*Mi31^2+2*Lbda31^2-4*Lbda31+2)^0.5);
else
    Gcr31=0;
    Mier31=0.5;
    dMi31=1/(2*(2*Mi31^2+2*Lbda31^2-4*Mi31+2)^0.5);
    dLbda31=-1/(2*(2*Mi31^2+2*Lbda31^2-4*Lbda31+2)^0.5);
end
if D31>=1;
    D31=1;
    Gcr31=0;
    Mier31=0.5;
    dMi31=1/(2*(2*Mi31^2+2*Lbda31^2-4*Mi31+2)^0.5);
    dLbda31=-1/(2*(2*Mi31^2+2*Lbda31^2-4*Lbda31+2)^0.5);
else
    end
Ypal(N)=Mier31;
MSEpal(N)=0.5*(Yd-Ypal(N))^2;
dMSEpal(N)=(Ypal(N)-Yd);

% Atualização Pesos
w31p(N+1)=w31p(N)-lr*dMSEpal(N)*dMi31*dMi21*Mier11;
w32p(N+1)=w32p(N)-lr*dMSEpal(N)*dMi31*dLbda21*Mier12;
w33p(N+1)=w33p(N)-lr*dMSEpal(N)*dLbda31*Mier13;
w11p(N+1)=w11p(N)-lr*dMSEpal(N)*dMi31*dMi21*w31p(N)*dMi11*a1;
w12p(N+1)=w12p(N)-lr*dMSEpal(N)*dMi31*dLbda21*w32p(N)*dMi12*a1;
w13p(N+1)=w13p(N)-lr*dMSEpal(N)*dLbda31*w33p(N)*dMi13*a1;
w21p(N+1)=w21p(N)-lr*dMSEpal(N)*dMi31*dMi21*w31p(N)*dLbda11*a2;
w22p(N+1)=w22p(N)-lr*dMSEpal(N)*dMi31*dLbda21*w32p(N)*dLbda12*a2;
w23p(N+1)=w23p(N)-lr*dMSEpal(N)*dLbda31*w33p(N)*dLbda13*a2;
end

%***** Figuras Comparativas *****
figure(1);
plot(Ypal,'b','LineWidth',2); hold on;
plot(Y,'m','LineWidth',2); hold on;
plot(Yt,'g','LineWidth',2); hold on;
plot(Yr,'r','LineWidth',2); hold on;
plot(Yl,'k','LineWidth',2); grid on;
xlabel('épocas','FontSize', 20);
ylabel('saída','FontSize', 20);
set(gca,'FontSize',20);
leg = legend('RNP','RNA $\sigma$','RNA tanh','RNA ReLU','RNA LReLU');
set(leg,'Interpreter','Latex');
set(leg,'Location','best','Fontsize',20);

figure(2);
plot(MSEpal,'b','LineWidth',2); hold on;
plot(MSE,'m','LineWidth',2); hold on on;
plot(MSEt,'g','LineWidth',2); hold on;
plot(MSEr,'r','LineWidth',2); hold on;
plot(MSEl,'k','LineWidth',2); grid on;
xlabel('épocas','FontSize', 20);
ylabel('MSE','FontSize', 20);
set(gca,'FontSize',20);
leg = legend('RNP','RNA $\sigma$','RNA tanh','RNA ReLU','RNA LReLU');
set(leg,'Interpreter','Latex');
set(leg,'Location','best','Fontsize',20);

```


APÊNDICE C – CÉLULAS NEURAIS ARTIFICIAIS PARACONSISTENTES EM SIMULINK

Figura 65 – CNAPap_{CTX} (PANCL_{CTX}) em Simulink (a) e estrutura interna (b).

Figura 66 – CNAPp (PANCL), saídas lineares em Simulink (a) e estrutura interna simplificada (b).

Figura 67 – Estrutura interna completa da CNAPP, todas as saídas em Simulink.

APÊNDICE D – ALGORITMO DE APRENDIZADO DO CONTROLADOR POR REDE NEURAL PARA CONSISTENTE RECORRENTE (RNPR3)

```
% Rede Neural Paraconsistente Recorrente
% W1 - ganhos camada de entrada; B1 - Bias da camada entrada (opcional)
% W2 - ganhos camada de saída (opcional); B2 - Bias camada de saída (opcional)
% Somador Mi com Saídas do RIP + Ref, Ea e Ep
% Lbda com Saída controlador recorrente
% Planta RIP com referência

clear all; close all; clc;

% Carrega Arquivos Entrada e Saída Planta RIP
Dados=xlsread('medidas.xlsx')/1000;

e=size(Dados);

Entrada(:,1)=Dados(:,6);Entrada(:,2)=Dados(:,7);Entrada(:,3)=Dados(:,8);Entrada(:,4)=Dados(:,9);
Entrada(:,5)=Dados(:,15)+Dados(:,12);Entrada(:,6)=Dados(:,7)-Dados(:,11);
Saída(:,1)=Dados(:,5); % Sinal de Controle Futuro

% Entrada(:,1)=th0_f_a = ângulo braço atrasado;
% Entrada(:,2)=th1_f_a = ângulo pêndulo atrasado;
% Entrada(:,3)=dth0_f_a = velocidade braço atrasado;
% Entrada(:,4)=dth1_f_a = velocidade pêndulo atrasado;
% Entrada(:,5)=r_a+eb = sinal de referência + erro braço (th0_f_a -
% th0_chap_a)
% Entrada(:,6)=ep = erro pêndulo (th1_f_a - th1_chap_a)
% Saída(:,1)=sinal de controle u atual

lr=0.5; % learning rate

%Ge=1/round(max(Me-me)); % Ganho Entrada Inicial
Ge=0.125; % Ganho Inicial de Entrada (Garantir que entradas fiquem entre 0 e 1)
%Gi=2+2^0.5; % Ganho da Camada Intermediária quando Lbda = 0.5 fixo
Gi=1;
%Gs=1/Ge; % Ganho de Referência de Saída quando W2 não é usado.
Gs=4;

GClim=0.0001; % Limiar decisão GC

En=7; % # de entradas
Hn=6; % # neurônios LPA2v camada oculta

% Ajuste Iniciais dos Pesos e Bias
W=rand(Hn,En);W1m=W(:,1:En-1);W1l=W(:,En);
%B1=rand(Hn,1); % Valor Bias Inicial Aleatório
B1=0.5*ones(Hn,1); % Valor de Bias Camada Oculta Otimizado
%W2=rand(1,Hn); % Valor Inicial W2 aleatório
W2=ones(1,Hn); % Valor W2 Otimizado
%B2=rand(1,1); % Valor de Bias Camada 2 Aleatório
B2=0; % Valor Bias Inicial Otimizado

n=500; % Número de epochs

nn=500; % número de pontos de aprendizagem

E=Entrada(1:nn,:); S=Saída(1:nn,1);
% Executa n epochs back propagation batelada
```

```

Spal=0; Spald(1,1)=0; Gfd=1; FL=1;
m=0; e=size(E);

while m<n;

    % Incrementa o contador do loop
    m=m+1;

    % Interage através de todas as amostras
    for i=1:e(1)
        % Input data from current example set

        Ei = [E(i,1),E(i,2),E(i,3),E(i,4),E(i,5),E(i,6)].';
        Di = S(i,:).';

        % Propaga o sinal através da Rede
        for x=1:Hn;
            % Neuronios camada oculta
            Mi(1,x)=Ge*(W1m(x,:)*Ei)+B1(x,1);
            if Mi(1,x)>=1;
                Mi(1,x)=1;
            elseif Mi(1,x)<=0;
                Mi(1,x)=0;
            else
                end
            Lbda(1,x)=Ge*(W1l(x,:)*Spald(i,1))+B1(x,1);
            %Lbda(1,x)=0.5;
            if Lbda(1,x)>=1;
                Lbda(1,x)=1;
            elseif Lbda(1,x)<=0;
                Lbda(1,x)=0;
            else
                end
            Gc(1,x)=Mi(1,x)-Lbda(1,x);
            Gct(1,x)=Mi(1,x)+Lbda(1,x)-1;
            D(1,x)=((1-abs(Gc(1,x)))^2+Gct(1,x)^2)^0.5;
            if Gc(1,x)>GClim;
                GCR(1,x)=1-D(1,x);
                Mier(1,x)=(GCR(1,x)+1)/2;
                dMi(1,x)=(1-Mi(1,x))/((2*Mi(1,x)^2+2*Lbda(1,x)^2-4*Mi(1,x)+2)^0.5);
                dLbda(1,x)=-Lbda(1,x)/((2*Mi(1,x)^2+2*Lbda(1,x)^2-4*Mi(1,x)+2)^0.5);
            elseif Gc(1,x)<-GClim;
                GCR(1,x)=D(1,x)-1;
                Mier(1,x)=(GCR(1,x)+1)/2;
                dMi(1,x)=Mi(1,x)/((2*Mi(1,x)^2+2*Lbda(1,x)^2-4*Lbda(1,x)+2)^0.5);
                dLbda(1,x)=(Lbda(1,x)-1)/((2*Mi(1,x)^2+2*Lbda(1,x)^2-4*Lbda(1,x)+2)^0.5);
            else
                Mier(1,x)=0.5;
                dMi(1,x)=1/(2*(2*Mi(1,x)^2+2*Lbda(1,x)^2-4*Mi(1,x)+2)^0.5);
                dLbda(1,x)=1/(2*(2*Mi(1,x)^2+2*Lbda(1,x)^2-4*Lbda(1,x)+2)^0.5);
            end
            if D(1,x)>=1;
                D(1,x)=1;
                Mier(1,x)=0.5;
                dMi(1,x)=0;
                dLbda(1,x)=0;
            else
                end
            end
        end
        Spal=Gs*Gi*W2*(Mier-0.5)'+B2; % Calcula Saída da Rede Neural Paraconsistente
    end
end

```

```

Spald(i+1,1)=Spal;
Sneuro(i,1)=Spal;

% Calcula a derivada do Erro e ajusta os pesos de Entrada
Erro=Di-Spal;

for x=1:Hn;
    delta1m(1,x)=dMi(1,x)*Gi*Gs*W2(1,x)*Erro;
    delta1l(1,x)=dLbda(1,x)*Gi*Gs*W2(1,x)*Erro;
    W1m(x,:)=W1m(x,:)+lr*delta1m(1,x)*Ge*Ei'; % Ajusta Pesos Camada Oculta
    W1l(x,:)=W1l(x,:)+lr*delta1l(1,x)*Ge*Spald(i,1)'; % Ajusta Pesos Camada Oculta
    %B1(x,1)=B1(x,1)+0.1*lr*delta1m(1,x); % Ajusta Bias Camada Oculta
    %W2(1,x)=W2(1,x)+0.1*lr*Gi*Gs*Erro*Mier(1,x); % Ajusta Pesos Camada Saída
end
W11(i,1:(En-1))=W1m(1,:);W11(i,En)=W1l(1,:);
W12(i,1:(En-1))=W1m(2,:);W12(i,En)=W1l(2,:);
W13(i,1:(En-1))=W1m(3,:);W13(i,En)=W1l(3,:);
W14(i,1:(En-1))=W1m(4,:);W14(i,En)=W1l(4,:);
W15(i,1:(En-1))=W1m(5,:);W15(i,En)=W1l(5,:);
W16(i,1:(En-1))=W1m(6,:);W16(i,En)=W1l(6,:);

%B2=B2+0.1*lr*Erro; % Ajusta Bias Camada Saída
end

W1(:,1:(En-1))=W1m(:,:);W1(:,En)=W1l(:,1);

W1(1,:)=mean(W11);
W1(2,:)=mean(W12);
W1(3,:)=mean(W13);
W1(4,:)=mean(W14);
W1(5,:)=mean(W15);
W1(6,:)=mean(W16);

R(m,1)=rms(S-Sneuro);
MSEt=immse(S,(Sneuro));
MSE(1,m)=MSEt;
end

% Gera Grafico MSE treinamento
% Grafico Erro RMS
figure(1);
plot(MSE);
grid on;
xlabel('Samples','FontSize', 20);
ylabel('MSE','FontSize', 20);

```


APÊNDICE E – CRNPR-MRC NO CONTROLE DO PROTÓTIPO RIP

DECLARAÇÃO DE VARIÁVEIS

//*****

// Parâmetros Modelo Braço (Arm)

```
float Gclim=0.0001;
float GEa=0, Gla=0, GSa=0;
float Lba1=0.5, Lba2=0.5, Lba3=0.5, Lba4=0.5;
double W11a=0, W12a=0, W13a=0, W14a=0, W15a=0, W16a=0;
double W21a=0, W22a=0, W23a=0, W24a=0, W25a=0, W26a=0;
double W31a=0, W32a=0, W33a=0, W34a=0, W35a=0, W36a=0;
double W41a=0, W42a=0, W43a=0, W44a=0, W45a=0, W46a=0;
double Mia1=0.5, Miera1=0.5;
double Mia2=0.5, Miera2=0.5;
double Mia3=0.5, Miera3=0.5;
double Mia4=0.5, Miera4=0.5;
double DDa1=1, DDa2=1, DDa3=1, DDa4=1;
double GCa1=0, GCTa1=0, Pa1=1, Pa2=0;
double GCa2=0, GCTa2=0, Pa3=1, Pa4=0;
double GCa3=0, GCTa3=0, Pa5=1, Pa6=0;
double GCa4=0, GCTa4=0, Pa7=1, Pa8=0;
double th0_chap=0, th0_chap_a=0, dth0_chap=0;
double Ea=0;
```

// Parâmetros Modelo Pen

```
float GEp=0, Glp=0, GSp=0;
float Lbp1=0.5, Lbp2=0.5, Lbp3=0.5, Lbp4=0.5;
double W11p=0, W12p=0, W13p=0, W14p=0, W15p=0, W16p=0;
double W21p=0, W22p=0, W23p=0, W24p=0, W25p=0, W26p=0;
double W31p=0, W32p=0, W33p=0, W34p=0, W35p=0, W36p=0;
double W41p=0, W42p=0, W43p=0, W44p=0, W45p=0, W46p=0;
double Mip1=0.5, Mierp1=0.5;
double Mip2=0.5, Mierp2=0.5;
double Mip3=0.5, Mierp3=0.5;
double Mip4=0.5, Mierp4=0.5;
double DDp1=1, DDp2=1, DDp3=1, DDp4=1;
double GCp1=0, GCTp1=0, Pp1=1, Pp2=0;
double GCp2=0, GCTp2=0, Pp3=1, Pp4=0;
double GCp3=0, GCTp3=0, Pp5=1, Pp6=0;
double GCp4=0, GCTp4=0, Pp7=1, Pp8=0;
double th1_chap=0, th1_chap_a=0, dth1_chap=0;
double Ep=0;
```

// Parâmetros Modelo Controlador

```
float GEc=0, Glc=0, GSc=0, GEcl=1, FL=0.01;
float Lbc1=0.5, Lbc2=0.5, Lbc3=0.5, Lbc4=0.5, Lbc5=0.5, Lbc6=0;
double W11c=0, W12c=0, W13c=0, W14c=0, W15c=0, W16c=0, W17c=0, W18c=0;
double W21c=0, W22c=0, W23c=0, W24c=0, W25c=0, W26c=0, W27c=0, W28c=0;
double W31c=0, W32c=0, W33c=0, W34c=0, W35c=0, W36c=0, W37c=0, W38c=0;
double W41c=0, W42c=0, W43c=0, W44c=0, W45c=0, W46c=0, W47c=0, W48c=0;
double W51c=0, W52c=0, W53c=0, W54c=0, W55c=0, W56c=0, W57c=0, W58c=0;
double W61c=0, W62c=0, W63c=0, W64c=0, W65c=0, W66c=0, W67c=0, W68c=0;
double Mic1=0.5, Mierc1=0.5;
double Mic2=0.5, Mierc2=0.5;
double Mic3=0.5, Mierc3=0.5;
double Mic4=0.5, Mierc4=0.5;
double Mic5=0.5, Mierc5=0.5;
double Mic6=0.5, Mierc6=0.5;
double DDc1=1, DDc2=1, DDc3=1, DDc4=1, DDc5=1, DDc6=1;
```

```
double GCc1=0, GCTc1=0, Pc1=1, Pc2=0;
double GCc2=0, GCTc2=0, Pc3=1, Pc4=0;
double GCc3=0, GCTc3=0, Pc5=1, Pc6=0;
double GCc4=0, GCTc4=0, Pc7=1, Pc8=0;
double GCc5=0, GCTc5=0, Pc9=1, Pc10=0;
double GCc6=0, GCTc6=0, Pc11=1, Pc12=0;
float u_d=0;
float upal=0, Gfb=1;
```

REDES NEURAIS PARA CONSISTENTES RECORRENTES

```
//*****
```

```
// RPNN Braço th0_chap e Ea
```

```
GEa=0.250; Gla=1; GSa=4;
```

```
Lba1=0.5; Lba2=0.5; Lba3=0.5; Lba4=0.5;
```

```
W11a=0.2698; W12a=0.1289; W13a=-0.0485; W14a=-0.0488; W15a=0.0063; W16a=0.3419;
W21a=0.1973; W22a=-0.1716; W23a=0.0182; W24a=-0.0084; W25a=0.0447; W26a=0.3885;
W31a=0.9220; W32a=0.3216; W33a=-0.0619; W34a=0.0136; W35a=0.0116; W36a=-0.2482;
W41a=0.5885; W42a=-0.3051; W43a=0.1131; W44a=0.0385; W45a=-0.0552; W46a=-0.4971;
```

```
// Neurônio 1
```

```
Mia1=GEa*(W11a*th0_f_a+W12a*th1_f_a+W13a*dth0_f_a+W14a*dth1_f_a+W15a*u)+0.5;
```

```
if (Mia1>=1)
```

```
    Mia1=1;
```

```
if (Mia1<=0)
```

```
    Mia1=0;
```

```
Lba1=GEa*(W16a*th0_chap_a)+0.5;
```

```
if (Lba1>=1)
```

```
    Lba1=1;
```

```
if (Lba1<=0)
```

```
    Lba1=0;
```

```
GCa1=Mia1-Lba1;
```

```
GCTa1=Mia1+Lba1-1;
```

```
Pa1=pow(1-abs(GCa1),2);
```

```
Pa2=pow(GCTa1,2);
```

```
DDa1=sqrt(Pa1+Pa2);
```

```
if (DDa1>=1)
```

```
    DDa1=1;
```

```
if (GCa1<-Gclim)
```

```
    Miera1=DDa1*0.5;
```

```
else if (GCa1>Gclim)
```

```
    Miera1=(2-DDa1)*0.5;
```

```
else
```

```
    Miera1=0.5;
```

```
// Neurônio 2
```

```
Mia2=GEa*(W21a*th0_f_a+W22a*th1_f_a+W23a*dth0_f_a+W24a*dth1_f_a+W25a*u)+0.5;
```

```
if (Mia2>=1)
```

```
    Mia2=1;
```

```
if (Mia2<=0)
```

```
    Mia2=0;
```

```
Lba2=GEa*(W26a*th0_chap_a)+0.5;
```

```
if (Lba2>=1)
```

```
    Lba2=1;
```

```
if (Lba2<=0)
```

```
    Lba2=0;
```

```
GCa2=Mia2-Lba2;
```

```
GCTa2=Mia2+Lba2-1;
```

```
Pa3=pow(1-abs(GCa2),2);
```

```
Pa4=pow(GCTa2,2);
```



```

DDa2=sqrt(Pa3+Pa4);
if (DDa2>=1)
    DDa2=1;
if (GCa2<-Gclim)
    Miera2=DDa2*0.5;
else if (GCa2>Gclim)
    Miera2=(2-DDa2)*0.5;
else
    Miera2=0.5;

// Neurônio 3
Mia3=GEa*(W31a*th0_f_a+W32a*th1_f_a+W33a*dth0_f_a+W34a*dth1_f_a+W35a*u)+0.5;
if (Mia3>=1)
    Mia3=1;
if (Mia3<=0)
    Mia3=0;
Lba3=GEa*(W36a*th0_chap_a)+0.5;
if (Lba3>=1)
    Lba3=1;
if (Lba3<=0)
    Lba3=0;
GCa3=Mia3-Lba3;
GCTa3=Mia3+Lba3-1;
Pa5=pow(1-abs(GCa3),2);
Pa6=pow(GCTa3,2);
DDa3=sqrt(Pa5+Pa6);
if (DDa3>=1)
    DDa3=1;
if (GCa3<-Gclim)
    Miera3=DDa3*0.5;
else if (GCa3>Gclim)
    Miera3=(2-DDa3)*0.5;
else
    Miera3=0.5;

// Neurônio 4
Mia4=GEa*(W41a*th0_f_a+W42a*th1_f_a+W43a*dth0_f_a+W44a*dth1_f_a+W45a*u)+0.5;
if (Mia4>=1)
    Mia4=1;
if (Mia4<=0)
    Mia4=0;
Lba4=GEa*(W46a*th0_chap_a)+0.5;
if (Lba4>=1)
    Lba4=1;
if (Lba4<=0)
    Lba4=0;
GCa4=Mia4-Lba4;
GCTa4=Mia4+Lba4-1;
Pa7=pow(1-abs(GCa4),2);
Pa8=pow(GCTa4,2);
DDa4=sqrt(Pa7+Pa8);
if (DDa4>=1)
    DDa4=1;
if (GCa4<-Gclim)
    Miera4=DDa4*0.5;
else if (GCa4>Gclim)
    Miera4=(2-DDa4)*0.5;
else
    Miera4=0.5;

```

```

// Neurônio saída linear
th0_chap=Gla*GSa*((Miera1-0.5)+(Miera2-0.5)+(Miera3-0.5)+(Miera4-0.5)); // Saída PNN ângulo
braço
th0_chap_a=th0_chap;

Ea=th0_f-th0_chap; // erro entre ângulo braço e ângulo braço PNN (observador)

// RPNN Pêndulo th1_chap e Ep
GEp=0.250; Glp=1; GSp=4;
Lbp1=0.5; Lbp2=0.5; Lbp3=0.5; Lbp4=0.5;
W11p=-0.0907; W12p=0.7128; W13p=-0.2844; W14p=-0.2677; W15p=0.0375; W16p=-0.3226;
W21p=0.1141; W22p=0.6093; W23p=0.2888; W24p=0.1719; W25p=0.1072; W26p=0.4281;
W31p=0.2668; W32p=0.2033; W33p=-0.1315; W34p=-0.1345; W35p=0.0577; W36p=-0.4056;
W41p=-0.2918; W42p=0.5321; W43p=0.1282; W44p=0.2537; W45p=-0.2059; W46p=0.3140;

// Neurônio 1
Mip1=GEp*(W11p*th0_f_a+W12p*th1_f_a+W13p*dth0_f_a+W14p*dth1_f_a+W15p*u)+0.5;
if (Mip1>=1)
    Mip1=1;
if (Mip1<=0)
    Mip1=0;
Lbp1=GEp*(W16p*th1_chap_a)+0.5;
if (Lbp1>=1)
    Lbp1=1;
if (Lbp1<=0)
    Lbp1=0;
GCp1=Mip1-Lbp1;
GCTp1=Mip1+Lbp1-1;
Pp1=pow(1-abs(GCp1),2);
Pp2=pow(GCTp1,2);
DDp1=sqrt(Pp1+Pp2);
if (DDp1>=1)
    DDp1=1;
if (GCp1<-Gclim)
    Mierp1=DDp1*0.5;
else if (GCp1>Gclim)
    Mierp1=(2-DDp1)*0.5;
else
    Mierp1=0.5;

// Neurônio 2
Mip2=GEp*(W21p*th0_f_a+W22p*th1_f_a+W23p*dth0_f_a+W24p*dth1_f_a+W25p*u)+0.5;
if (Mip2>=1)
    Mip2=1;
if (Mip2<=0)
    Mip2=0;
Lbp2=GEp*(W26p*th1_chap_a)+0.5;
if (Lbp2>=1)
    Lbp2=1;
if (Lbp2<=0)
    Lbp2=0;
GCp2=Mip2-Lbp2;
GCTp2=Mip2+Lbp2-1;
Pp3=pow(1-abs(GCp2),2);
Pp4=pow(GCTp2,2);
DDp2=sqrt(Pp3+Pp4);
if (DDp2>1)
    DDp2=1;
if (GCp2<-Gclim)
    Mierp2=DDp2*0.5;

```

```

else if (GCp2>Gclim)
    Mierp2=(2-DDp2)*0.5;
else
    Mierp2=0.5;

// Neurônio 3
Mip3=GEp*(W31p*th0_f_a+W32p*th1_f_a+W33p*dth0_f_a+W34p*dth1_f_a+W35p*u)+0.5;
if (Mip3>=1)
    Mip3=1;
if (Mip3<=0)
    Mip3=0;
Lbp3=GEp*(W36p*th1_chap_a)+0.5;
if (Lbp3>=1)
    Lbp3=1;
if (Lbp3<=0)
    Lbp3=0;
GCp3=Mip3-Lbp3;
GCTp3=Mip3+Lbp3-1;
Pp5=pow(1-abs(GCp3),2);
Pp6=pow(GCTp3,2);
DDp3=sqrt(Pp5+Pp6);
if (DDp3>=1)
    DDp3=1;
if (GCp3<-Gclim)
    Mierp3=DDp3*0.5;
else if (GCp3>Gclim)
    Mierp3=(2-DDp3)*0.5;
else
    Mierp3=0.5;

// Neurônio 4
Mip4=GEp*(W41p*th0_f_a+W42p*th1_f_a+W43p*dth0_f_a+W44p*dth1_f_a+W45p*u)+0.5;
if (Mip4>=1)
    Mip4=1;
if (Mip4<=0)
    Mip4=0;
Lbp4=GEp*(W46p*th1_chap_a)+0.5;
if (Lbp4>=1)
    Lbp4=1;
if (Lbp4<=0)
    Lbp4=0;
GCp4=Mip4-Lbp4;
GCTp4=Mip4+Lbp4-1;
Pp7=pow(1-abs(GCp4),2);
Pp8=pow(GCTp4,2);
DDp4=sqrt(Pp7+Pp8);
if (DDp4>=1)
    DDp4=1;
if (GCp4<-Gclim)
    Mierp4=DDp4*0.5;
else if (GCp4>Gclim)
    Mierp4=(2-DDp4)*0.5;
else
    Mierp4=0.5;

// Neurônio saída linear
th1_chap=GIp*GSp*((Mierp1-0.5)+(Mierp2-0.5)+(Mierp3-0.5)+(Mierp4-0.5)); // Saída PNN ângulo
braço
//th1_chap=GIp*GSp*((Mierp1-0.5)+(Mierp2-0.5)+(Mierp3-0.5)+(Mierp4-0.5)); // Saída PNN ângulo
braço

```

```
//dth1_chap=0.5*GEd*GSd*(th1_chap-th1_chap_a)+(1-FLd)*dth1_chap;
th1_chap_a=th1_chap;
```

```
Ep=th1_f-th1_chap; // erro entre ângulo pêndulo e ângulo pêndulo PNN (observador)
```

```
//*****
```

```
// RPNN Controlador (MRC)
```

```
GEc=0.125; GLc=1; GSc=4;
```

```
W11c=0.4368; W12c=3.4771; W13c=1.2851; W14c=0.4147; W15c=-0.7061; W16c=0.6398; W17c=
0.2175;
W21c=0.4512; W22c=3.8034; W23c=-0.0598; W24c=1.052; W25c=0.0685; W26c=-0.0421; W27c= -
0.4752;
W31c=-0.3405; W32c=4.3227; W33c=-0.1484; W34c=0.6035; W35c=0.3641; W36c=0.2019; W37c=
0.7546;
W41c=0.3516; W42c=4.1752; W43c=-0.3026; W44c=-0.0286; W45c=-0.3855; W46c=-0.026; W47c=
-0.7899;
W51c=0.0368; W52c=3.5782; W53c=-0.043; W54c=0.7043; W55c=0.0923; W56c=0.016; W57c=
0.7939;
W61c=-0.0053; W62c=4.2051; W63c=0.3989; W64c=0.9; W65c=-0.3409; W66c=0.7512; W67c= -
0.1762;
```

```
// Neurônio 1
```

```
Mic1=GEc*(W11c*th0_f+W12c*th1_f+W13c*dth0_f+W14c*dth1_f+W15c*(r+Ea)+W16c*Ep)+0.5;
```

```
if (Mic1>=1)
```

```
    Mic1=1;
```

```
if (Mic1<=0)
```

```
    Mic1=0;
```

```
Lbc1=GEc*W17c*upal+0.5;
```

```
if (Lbc1>=1)
```

```
    Lbc1=1;
```

```
if (Lbc1<=0)
```

```
    Lbc1=0;
```

```
GCc1=Mic1-Lbc1;
```

```
GCTc1=Mic1+Lbc1-1;
```

```
Pc1=pow(1-abs(GCc1),2);
```

```
Pc2=pow(GCTc1,2);
```

```
DDc1=sqrt(Pc1+Pc2);
```

```
if (DDc1>=1)
```

```
    DDc1=1;
```

```
if (GCc1<-Gclim)
```

```
    Mierc1=DDc1*0.5;
```

```
else if (GCc1>Gclim)
```

```
    Mierc1=(2-DDc1)*0.5;
```

```
else
```

```
    Mierc1=0.5;
```

```
// Neurônio 2
```

```
Mic2=GEc*(W21c*th0_f+W22c*th1_f+W23c*dth0_f+W24c*dth1_f+W25c*(r+Ea)+W26c*Ep)+0.5;
```

```
if (Mic2>=1)
```

```
    Mic2=1;
```

```
if (Mic2<=0)
```

```
    Mic2=0;
```

```
Lbc2=GEc*W27c*upal+0.5;
```

```
if (Lbc2>=1)
```

```
    Lbc2=1;
```

```
if (Lbc2<=0)
```

```
    Lbc2=0;
```

```
GCc2=Mic2-Lbc2;
```

```
GCTc2=Mic2+Lbc2-1;
```

```

Pc3=pow(1-abs(GCc2),2);
Pc4=pow(GCTc2,2);
DDc2=sqrt(Pc3+Pc4);
if (DDc2>=1)
    DDc2=1;
if (GCc2<-Gclim)
    Mierc2=DDc2*0.5;
else if (GCc2>Gclim)
    Mierc2=(2-DDc2)*0.5;
else
    Mierc2=0.5;

// Neurônio 3
Mic3=GEc*(W31c*th0_f+W32c*th1_f+W33c*dth0_f+W34c*dth1_f+W35c*(r+Ea)+W36c*Ep)+0.5;
if (Mic3>=1)
    Mic3=1;
if (Mic3<=0)
    Mic3=0;
Lbc3=GEcl*W37c*upal+0.5;
if (Lbc3>=1)
    Lbc3=1;
if (Lbc3<=0)
    Lbc3=0;
GCc3=Mic3-Lbc3;
GCTc3=Mic3+Lbc3-1;
Pc5=pow(1-abs(GCc3),2);
Pc6=pow(GCTc3,2);
DDc3=sqrt(Pc5+Pc6);
if (DDc3>=1)
    DDc3=1;
if (GCc3<-Gclim)
    Mierc3=DDc3*0.5;
else if (GCc3>Gclim)
    Mierc3=(2-DDc3)*0.5;
else
    Mierc3=0.5;

// Neurônio 4
Mic4=GEc*(W41c*th0_f+W42c*th1_f+W43c*dth0_f+W44c*dth1_f+W45c*(r+Ea)+W46c*Ep)+0.5;
if (Mic4>=1)
    Mic4=1;
if (Mic4<=0)
    Mic4=0;
Lbc4=GEcl*W47c*upal+0.5;
if (Lbc4>=1)
    Lbc4=1;
if (Lbc4<=0)
    Lbc4=0;
GCc4=Mic4-Lbc4;
GCTc4=Mic4+Lbc4-1;
Pc7=pow(1-abs(GCc4),2);
Pc8=pow(GCTc4,2);
DDc4=sqrt(Pc7+Pc8);
if (DDc4>=1)
    DDc4=1;
if (GCc4<-Gclim)
    Mierc4=DDc4*0.5;
else if (GCc4>Gclim)
    Mierc4=(2-DDc4)*0.5;
else

```

```

Mierc4=0.5;

// Neurônio 5
Mic5=GEc*(W51c*th0_f+W52c*th1_f+W53c*dth0_f+W54c*dth1_f+W55c*(r+Ea)+W56c*Ep)+0.5;
if (Mic5>=1)
    Mic5=1;
if (Mic5<=0)
    Mic5=0;
Lbc5=GEcl*W57c*upal+0.5;
if (Lbc5>=1)
    Lbc5=1;
if (Lbc5<=0)
    Lbc5=0;
GCc5=Mic5-Lbc5;
GCTc5=Mic5+Lbc5-1;
Pc9=pow(1-abs(GCc5),2);
Pc10=pow(GCTc5,2);
DDc5=sqrt(Pc9+Pc10);
if (DDc5>=1)
    DDc5=1;
if (GCc5<-Gclim)
    Mierc5=DDc5*0.5;
else if (GCc5>Gclim)
    Mierc5=(2-DDc5)*0.5;
else
    Mierc5=0.5;

// Neurônio 6
Mic6=GEc*(W61c*th0_f+W62c*th1_f+W63c*dth0_f+W64c*dth1_f+W65c*(r+Ea)+W66c*Ep)+0.5;
if (Mic6>=1)
    Mic6=1;
if (Mic6<=0)
    Mic6=0;
Lbc6=GEcl*W67c*upal+0.5;
if (Lbc6>=1)
    Lbc6=1;
if (Lbc6<=0)
    Lbc6=0;
GCc6=Mic6-Lbc6;
GCTc6=Mic6+Lbc6-1;
Pc11=pow(1-abs(GCc6),2);
Pc12=pow(GCTc6,2);
DDc6=sqrt(Pc11+Pc12);
if (DDc6>=1)
    DDc6=1;
if (GCc6<-Gclim)
    Mierc6=DDc6*0.5;
else if (GCc6>Gclim)
    Mierc6=(2-DDc6)*0.5;
else
    Mierc6=0.5;

```

CONTROLE

```

//*****
upal=Glc*GSc*((Mierc1-0.5)+(Mierc2-0.5)+(Mierc3-0.5)+(Mierc4-0.5)+(Mierc5-0.5)+(Mierc6-0.5));
u=upal;

```

ANEXO A – DESCRIÇÃO DO MODELAMENTO DO RIP

A modelagem de um sistema mecânico como o RIP, descrita em (NATH; DEWAN, 2019), pode ser obtida através das equações de Lagrange envolvendo energia cinética (T) e potencial (V), conforme equação (1), ao invés das equações de Newton que utilizam força e deslocamento.

$$L = T - V \quad (1)$$

Para a modelagem do sistema RIP, as seguintes hipóteses são consideradas:

- a) Condição inicial zero é inferidas quando o sistema RIP inicia a partir do estado de equilíbrio.
- b) A fim de preservar um modelo linear, o pêndulo apresenta um ângulo de desvio mínimo na vertical.
- c) Pequenos distúrbios podem ser acomodados ao linearizar o modelo do RIP.

A energia potencial é dada por (2):

$$V = m_1 g l_1 \cos(\theta_1) \quad (2)$$

O momento de inércia do braço é dado por (3), sendo o primeiro termo da equação derivado do cálculo de barra com comprimento ($2l_1$) com rotação em torno de seu centro de massa e o segundo termo dado pelo teorema dos eixos paralelos, com correção para girar em um ponto com distância d do centro de massa:

$$I_o = \frac{m_0(2L_0)^2}{12} + m_0 d^2 \quad (3)$$

A energia cinética do pêndulo referente à rotação é dada por (4):

$$T_0 = \frac{I_0 \dot{\theta}_0^2}{2} \quad (4)$$

O momento de inércia do pêndulo é dado por (5):

$$I_1 = \frac{m_1(2L_1)^2}{12} \quad (5)$$

A energia cinética do braço é dada por (6):

$$T_{11} = \frac{I_1 \dot{\theta}_1^2}{2} = \frac{m_1 (2L_1)^2 \dot{\theta}_1^2}{12 * 2} = \frac{m_1 L_1^2 \dot{\theta}_1^2}{6} \quad (6)$$

Considera-se também a energia de translação (7):

$$\begin{aligned} c_z &= L_1 \cos(\theta_1) \\ c_x &= r \cos(\theta_0) - L_1 \sin(\theta_1) \sin(\theta_0) \\ c_y &= r \sin(\theta_0) - L_1 \sin(\theta_1) \cos(\theta_0) \\ v_{cz} &= -L_1 \dot{\theta}_1 \sin(\theta_1) \\ v_{cx} &= -r \dot{\theta}_0 \sin(\theta_0) + L_1 \dot{\theta}_1 \cos(\theta_1) \sin(\theta_0) + L_1 \dot{\theta}_0 \sin(\theta_1) \cos(\theta_0) \\ v_{cy} &= -r \dot{\theta}_0 \cos(\theta_0) + L_1 \dot{\theta}_1 \sin(\theta_1) \cos(\theta_0) + L_1 \dot{\theta}_0 \sin(\theta_1) \sin(\theta_0) \\ T_{12} &= \frac{m_1}{2} (v_{cx}^2 + v_{cy}^2 + v_{cz}^2) \\ T_{12} &= \frac{m_1}{2} (L_1^2 \dot{\theta}_1^2 + r^2 \dot{\theta}_0^2 + L_1^2 \dot{\theta}_0^2 \sin^2(\theta_1) + 2rL_1 \dot{\theta}_0 \dot{\theta}_1 \cos(\theta_1)) \\ T_1 &= T_{11} + T_{12} \\ T_1 &= \frac{m_1}{2} \left(\frac{4}{3} L_1^2 \dot{\theta}_1^2 + r^2 \dot{\theta}_0^2 + L_1^2 \dot{\theta}_0^2 \sin^2(\theta_1) + 2rL_1 \dot{\theta}_0 \dot{\theta}_1 \cos(\theta_1) \right) \end{aligned} \quad (7)$$

A energia cinética total é dada por (8):

$$\begin{aligned} T &= T_0 + T_1 \\ T &= \frac{I_0 \dot{\theta}_0^2}{2} + \frac{m_1}{2} \left(\frac{4}{3} L_1^2 \dot{\theta}_1^2 + r^2 \dot{\theta}_0^2 + L_1^2 \dot{\theta}_0^2 \sin^2(\theta_1) + 2rL_1 \dot{\theta}_0 \dot{\theta}_1 \cos(\theta_1) \right) \end{aligned} \quad (8)$$

A equação de Euler-Lagrange de movimento é dada por (NATH; DEWAN, 2019):

$$\frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{\theta}_0} \right) - \frac{\partial L}{\partial \theta_0} = \tau_s - B_{eq} \dot{\theta}_0 \quad (9)$$

$$\frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{\theta}_1} \right) - \frac{\partial L}{\partial \theta_1} = -B_{eq} \dot{\theta}_1 \quad (10)$$

Onde B_{eq} é Amortecimento viscoso equivalente agindo sobre o Eixo do motor DC e τ_s é o torque gerado no eixo do braço, dado por (11):

$$\tau_S = \frac{K_T K_E - K_T K_E \dot{\theta}_0}{R_M} \quad (11)$$

Substituindo-se as equações (9) e (10), aplicando a linearização do modelo ocorre em torno do ponto de equilíbrio, dado em (12) e desprezando termos de ordem superior da série de Taylor, tem-se as equações (13) e (14).

$$\theta_1 = 0; \text{sen}(\theta_1) = 0; \dot{\theta}_1 = 0 \quad (12)$$

$$I_0 \ddot{\theta}_0 + m_1 r^2 \ddot{\theta}_0 - m_1 L_1 \ddot{\theta}_1 = \tau_S - B_{eq} \dot{\theta}_0 \quad (13)$$

$$I_1 \ddot{\theta}_1 + m_1 l_1^2 \ddot{\theta}_1 - m_1 L_1 \ddot{\theta}_0 - m_1 g L_1 \theta_1 = -B_{eq} \dot{\theta}_1 \quad (14)$$

Convertendo para equação de espaço de estado, para um sistema invariante no tempo, conforme equação (15), pode ser expressa como (16):

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (15)$$

$$\begin{bmatrix} \dot{\theta}_0 \\ \dot{\theta}_1 \\ \ddot{\theta}_0 \\ \ddot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{bd}{e} & \frac{-cg}{e} & 0 \\ 0 & \frac{ad}{e} & \frac{-bf}{e} & 0 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dot{\theta}_0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{cK_T}{R_M e} \\ \frac{bK_T}{R_M e} \end{bmatrix} u(t) \quad (16)$$

Onde (17):

$$a = I_0 + m_1 r^2$$

$$b = m_1 L_1 r$$

$$c = I_1 + m_1 r L_1^2$$

$$d = m_1 g L_1$$

$$e = ac - b^2$$

$$f = \frac{K_T K_E}{R_M}$$

$$u(t) = Vm = 12 * PWM \quad (17)$$

Nota: Para o cálculo das matrizes A e B foi utilizado um algoritmo em Matlab®, desenvolvido pelo LAC da POLI/USP, conforme Anexo B, considerando-se o amortecimento viscoso equivalente sobre Eixo do motor DC (B_{eq}) de 0.001 Nms/rad e amortecimento viscoso sobre o pivô do pêndulo (B_p) de 0.0001 Nms/rad. Aplicando-

se os valores da Tabela 2 da seção 4.3, aqui copiada, na equação (16), tem-se o modelo linearizado do RIP (18):

PARAMETRO	VALOR	DESCRIÇÃO
d	0.137 (m)	distância do eixo base ao centro de massa
m_o	0.393 (kg)	massa do braço
m_l	0.068 (kg)	massa do pêndulo
L_o	0.1825 (m)	comprimento da metade do braço
L_l	0.1035 (m)	comprimento da metade do pêndulo
r	0.23 (m)	distância do eixo base ao pêndulo
R_m	2.4 (Ohms)	resistência da armadura do motor
K_t	0.08 (Nm/A)	constante de torque do motor, incluindo redução
K_e	0.2 (V/(rad/s))	constante da equação do motor, incluindo redução
g	9.81 (m/s ²)	aceleração da gravidade
q_o	rad	ângulo do braço (positivo para sentido anti-horário)
q_l	rad	ângulo do pêndulo (positivo para sentido horário)
T_s	0.01 s	tempo de amostragem

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -9.1047 & -0.6066 & 0.0132 \\ 0 & 86.2615 & 1.0110 & -0.1249 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ 31.6490 \\ -52.7483 \end{bmatrix} \quad (18)$$

O modelo linearizado do RIP possui 4 variáveis $[\theta_0; \theta_1; \dot{\theta}_0; \dot{\theta}_1]$, no entanto, o protótipo apenas possui medições de $[\theta_0; \theta_1]$. Assim as variáveis de velocidade do braço e pêndulo $[\dot{\theta}_0; \dot{\theta}_1]$ devem ser estimadas. Elas podem ser calculadas por derivada *backward*, conforme a equação (70) repetida a seguir.

$$\dot{\theta}_0 = \frac{(\theta_0[n] - \theta_0[n-1])}{T_s} \text{ e } \dot{\theta}_1 = \frac{(\theta_1[n] - \theta_1[n-1])}{T_s}$$

De modo a reduzir o ruído das medições e amostragem dos *encoders* de ângulo do braço e pêndulo, e de suas derivadas, um filtro-passa-baixas (*low pass filter* – LPF) de 1ª ordem pode ser implementado (LATHI; GREEN, 2017) conforme equação (71), repetida a seguir para o ângulo do braço $[\theta_0]$ e n é o número de uma determinada amostra e τ a constante de tempo do filtro.

$$\theta_0[n]_{filt} = \left(\frac{\tau}{\tau + T_s} \right) \theta_0[n-1]_{filt} + \left(\frac{T_s}{\tau + T_s} \right) \theta_0[n]$$

ANEXO B – CÓDIGO MODELO RIP EM MATLAB DESENVOLVIDO PELO LAC – POLI/USP

```

clear all; clc; close all;

global Kt Ke Rm r b0 b1 b2 J1 J2 m0 m1 l1 l2 g Ts d l0 dp

syms t th0 th1 J0 J1 m0 m1 l0 l1 g d r dp
th0 = sym('th0(t)');
wth0 = diff(th0,t);
wth0_d = diff(wth0,t);
th1 = sym('th1(t)');
wth1 = diff(th1,t);
wth1_d = diff(wth1,t);
co0 = cos(th0);
se0 = sin(th0);
co1 = cos(th1);
se1 = sin(th1);

% Energia cinetica total
l0 = m0*(2*l0)^2/12+m0*d^2;
K0=l0*wth0^2/2;
K1=(m1/2)*(4/3 *l1^2*wth1^2 + r^2*wth0^2 + l1^2*wth0^2*se1^2 + 2*r*l1*wth0*wth1*co1);
K=simplify(expand(K0+K1),'steps',50);
% Energia potencial total
P = m1*g*l1*co1;

Km = simplify(expand(K - P),'steps',50);

% 1 - com relacao ao th0
% dK\dwth0 -> Derivada parcial em relacao a wth1;
Kd0 = deriv(Km, wth0);
% d(dK\dwth1)\dt -> Derivada no tempo da derivada parcial em relacao a wth1;
Kt0 = diff(Kd0, t);
% dK\dth1 -> Derivada parcial em relacao a th1;
Kq0 = deriv(Km, th0);

L1 = simplify(expand(Kt0 - Kq0));

% 2 - com relacao ao th1
% dK\dwth1 -> Derivada parcial em relacao a wth2;
Kd1 = deriv(Km, wth1);
% d(dK\dwth2)\dt -> Derivada no tempo da derivada parcial em relacao a wth2;
Kt1 = diff(Kd1, t);
% dK\dth2 -> Derivada parcial em relacao a th2;
Kq1 = deriv(Km, th1);

L2 = simplify(expand(Kt1 - Kq1));

syms TH0 TH1 WTH0 WTH1 WTH0_d WTH1_d

L1_ = subs(L1,wth0_d,WTH0_d);
L1_ = subs(L1_,wth1_d,WTH1_d);
L1_ = subs(L1_,wth0,WTH0);
L1_ = subs(L1_,wth1,WTH1);
L1_ = subs(L1_,th0,TH0);
L1_ = subs(L1_,th1,TH1);
L2_ = subs(L2,wth0_d,WTH0_d);

```

```

L2_ = subs(L2_,wth1_d,WTH1_d);
L2_ = subs(L2_,wth0,WTH0);
L2_ = subs(L2_,wth1,WTH1);
L2_ = subs(L2_,th0,TH0);
L2_ = subs(L2_,th1,TH1);

% V = (Im*Rm) + (Ke*w)
% w = x_d / (n*r)
% T = Kt*n*Im
% F = T/r
% PWM = V/12
% Com isso, tem-se:
syms n Kt Ke PWM Rm b0 b1
%F = Kt*n/(Rm*r)*12*PWM - Ke*Kt/(Rm*r^2)*V;
T=Kt*(12*PWM-Ke*WTH0)/(Rm);

S = solve(L1_ == T -b0*WTH0, L2_ == -b1*WTH1, WTH0_d, WTH1_d);

%Derivada do vetor de estados
xp_1=WTH0;
xp_2=WTH1;
xp_3=S.WTH0_d;
xp_4=S.WTH1_d;

%Vetor de estados
x_ = [TH0,TH1,WTH0,WTH1];

%Entrada
u = PWM;

% Linearização
Am=jacobian([xp_1;xp_2;xp_3;xp_4],x_);
Bm=jacobian([xp_1;xp_2;xp_3;xp_4],u);

% Definir valor dos parametros do pendulo
d = 0.137; %deslocamento do CM do braço
dp=0;
m0 = 0.393;
m1 = 0.068;
g = 9.81;
l0 = 0.365/2;
l1 = 0.207/2;
r = 0.23;
% Parametros do motor
n = 18.75;
Rm = 2.4; %12/5 (5 A de corrente stall)
Ke = 0.2; %Velocidade max= 500 rpm
Kt = 0.08;%0.59/(5); % Torque stall=0.59 Nm
b0=0.001; b1=0.0001;

Ts=1/100;

% Ponto de equilibrio
TH0=0; TH1=0; WTH0=0; WTH1=0;
x0 = [0 5*pi/180 0 0];
Trun=60;

A = double(subs(Am));
B = double(subs(Bm));

```