UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DANIEL VARGAS ZANETTE

ESTABILIDADE DE SISTEMAS DINÂMICOS EM TEMPO DISCRETO BASEADOS EM REDES NEURAIS RELU

Porto Alegre
2026

DANIEL VARGAS ZANETTE

ESTABILIDADE DE SISTEMAS DINÂMICOS EM TEMPO DISCRETO BASEADOS EM REDES NEURAIS RELU

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Automação e Controle

ORIENTADOR: Prof. Dr. João Manoel Gomes da Silva Jr.

Porto Alegre
2026

DANIEL VARGAS ZANETTE

ESTABILIDADE DE SISTEMAS DINÂMICOS EM TEMPO DISCRETO BASEADOS EM REDES NEURAIS RELU

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. João Manoel Gomes da Silva Jr., UFRGS
Doutor pela Université Paul Sabatier – Toulouse, França

Banca Examinadora:

Prof. Dr. Valter Júnior de Souza Leite, CEFET/MG
Doutor pela Universidade Estadual de Campinas - Campinas, Brazil

Prof. Dr. Marcelo Menezes Morato, UFSC
Doutor pela Universidade Federal de Santa Catarina Florianópolis, Brasil

Prof. Dr. Alexandre Sanfelici Bazanella, UFRGS
Doutor pela Universidade Federal de Santa Catarina - Florianópolis, Brasil

Coordenador do PPGEE: _____
Prof. Dr.-Ing. Luís Alberto Pereira

Porto Alegre, abril de 2026.

RESUMO

Este trabalho estuda o problema da análise de estabilidade exponencial da origem de sistemas dinâmicos em tempo discreto cujo campo vetorial é dado por uma Rede Neural (RN) de função de ativação ReLU (do inglês *Rectified Linear activation Unit*). Primeiramente, é apresentado como descrever RNs em uma representação implícita baseada em funções rampa (isto é, ReLU). Em seguida, explorando propriedades verificadas pela função ReLU escalar e usando funções de Lyapunov quadráticas por partes, são derivadas condições no formato de desigualdades matriciais para a certificação da estabilidade da origem do sistema em contexto global e local. A abordagem permite representar RNs de múltiplas camadas e diversas arquiteturas, como RNs Recorrentes e RNs Convolucionais. Além disso, a capacidade de representar conexões em série, paralelo e realimentação nesta representação possibilita a modelagem de RNs heterogêneas com camadas de diferentes arquiteturas e a análise de sistemas dados pela composição de sistemas lineares ou afins por partes com RNs. A aplicação dos resultados propostos é ilustrada em diversos exemplos numéricos.

Palavras-chave: Redes Neurais ReLU, Análise de Estabilidade, Métodos de Lyapunov, Sistemas Dinâmicos.

ABSTRACT

This work studies the problem of exponential stability analysis of the origin for discrete-time dynamical systems whose vector field is given by a Neural Network (NN) with ReLU (Rectified Linear activation Unit) activation function. First, it is shown how to describe NNs in an implicit representation based on ramp functions (i.e., ReLU). Next, by exploiting properties satisfied by the scalar ReLU function and using piecewise quadratic Lyapunov functions, matrix inequality conditions are derived to certify the stability of the origin of the system in both global and local contexts. The approach makes it possible to represent multi-layer NNs and various architectures, such as Recurrent NNs and Convolutional NNs. Moreover, the ability to represent series, parallel, and feedback connections in this framework enables the modeling of heterogeneous NNs with layers of different architectures, as well as the analysis of systems given by the composition of linear systems or piecewise affine systems with NNs. The application of the proposed results is illustrated through several numerical examples.

Keywords: ReLU Neural Network, Stability Analisys, Lyapunov Methods, Dinamic Systems.

LISTA DE ILUSTRAÇÕES

Figura 1 - Exemplo de neurônio artificial. 17
Figura 2 - Exemplo de uma RN com 3 entradas, 2 saídas e 3 camadas ocultas (contendo 4, 5 e 3 neurônios, respectivamente). 17
Figura 3 - Gráfico de funções de ativação típicas de RN. 18
Figura 4 - RN *feedforward* com 3 entradas (z_1, z_2, z_3), 2 saídas (v_1, v_2) e 1 camada oculta com 4 neurônios. 20
Figura 5 - RN residual com 2 entradas, 2 saídas e 1 camada oculta com 3 neurônios. Conexões residuais de z_1 para v_1 e z_2 para v_2. 22
Figura 6 - Conexão de realimentação. 26
Figura 7 - Gráfico da função de saturação. 27
Figura 8 - Saturação como uma RN ReLU para $n_z = 1$. 29
Figura 9 - Gráfico da função de zona morta. 29
Figura 10 - Zona morta como uma RN ReLU para $n_z = 1$. 30
Figura 11 - Evolução dos estados do sistema do exemplo 1 em MF para uma condição inicial de $x_1 = x_2 = 20$. A resposta do sistema com controlador MPC é representado pelos círculos vermelhos, enquanto o sistema com controlador de RN é representado pelas cruzes azuis. . . 47
Figura 12 - Sinal de controle ao longo da trajetória do sistema do exemplo 1 em MF para uma condição inicial de $x_1 = x_2 = 20$. A ação de controle do MPC e da RN são representados, respectivamente, por círculos vermelhos e por cruzes azuis. 47
Figura 13 - RAO estimada (linha vermelha tracejada), $\mathcal{D}$ (linha azul traço-ponto), restrições de estado (linha preta sólida), partições PWA do sistema (linha verde sólida), trajetórias convergentes (linhas pretas pontilhadas), trajetórias divergentes (linhas laranjas pontilhadas) para o controlador de RN. 57
Figura 14 - RAO estimado (linha vermelha tracejada), $\mathcal{D}$ (linha azul traço-ponto), partições PWA do sistema (linha verde contínua) e trajetórias (linhas pretas pontilhadas) para o MPC. 58
Figura 15 - RAO estimado pelo controlador RN (linha azul tracejada) vs RAO estimado pelo MPC (linha vermelha contínua). 59
Figura 16 - Comparação da evolução dos estados do sistema e do sinal de controle para o sistema com o controlador de RN (cruzes azuis) e para o sistema com o controlador MPC (círculos vermelhos). 60
Figura 17 - Exemplo de neurônio recorrente. 62
Figura 18 - RN Recorrente com 3 entradas, 2 saídas e uma camada oculta com 4 neurônios com recorrência independente. 64

Figura 19 - Pontos de equilíbrio e regiões de atração calculadas pelo Teorema 2: x_{e_1} (elipse roxa), x_{e_2} (elipse vermelha) e x_{e_3} (elipse amarela) e regiões de atração calculadas por (ZHANG *et al.*, 2009): x_{e_1} (retângulo roxo), x_{e_2} (retângulo azul claro) e x_{e_3} (retângulo verde). Ponto de vista 1. 71
Figura 20 - Pontos de equilíbrio e regiões de atração calculadas pelo Teorema 2: x_{e_1} (elipse roxa), x_{e_2} (elipse vermelha) e x_{e_3} (elipse amarela) e regiões de atração calculadas por (ZHANG *et al.*, 2009): x_{e_1} (retângulo roxo), x_{e_2} (retângulo azul claro) e x_{e_3} (retângulo verde). Ponto de vista 2. 72
Figura 21 - Exemplo de uma camada de convolução. 75
Figura 22 - Exemplo de uma camada de max pooling. 80
Figura 23 - Exemplo de uma camada de ativação ReLU. 87
Figura 24 - Exemplo de uma operação de preenchimento. 87
Figura 25 - Definição dos parâmetros de preenchimento ρ. 88
Figura 26 - Gráfico da função de *leaky* ReLU. 92
Figura 27 - Gráfico da função de hard Tanh. 94

LISTA DE ABREVIATURAS

BMI	*Bilenear Matrix Inequality* (Desigualdade Matricial Bilinear)
DEQ	*Deep Equilibrium Model* (Modelo de Equilíbrio Profundo)
GRU	*Gated Recurrent Unit* (Unidade Recorrente Controlada)
LCP	*Linear Complementarity Problem* (Problema de Complementaridade Linear)
LMI	*Linear Matrix Inequality* (Desigualdade Matricial Linear)
LPV	*Linear Parameter-Varying* (Linear com Parâmetros Variantes)
LSTM	*Long Short-Term Memory* (Memória de Longo e Curto Prazo)
MF	Malha Fechada
MPC	*Model Predictive Control* (Controle Preditivo Baseado em Modelo)
NARX	*Nonlinear Autoregressive with Exogenous inputs* (Autorregressivo Não Linear com Entradas Exógenas)
PWA	*Piecewise Affine* (Afim por Partes)
PWQ	*Piecewise Quadratic* (Quadrática por Partes)
QP	*Quadratic Programming* (Programação Quadrática)
RAO	Região de Atração da Origem
ReLU	*Rectified Linear activation Unit* (Unidade Linear Retificada)
RN	Rede Neural
RNC	Rede Neural Convolutiva
RNR	Rede Neural Recorrente
SDP	*Semidefined Programming* (Programação semidefinida)
Tanh	Tangente Hiperbólica

LISTA DE SÍMBOLOS

$\sum$	Somatório
$\in$	Pertence
$\forall$	Para todo
$:=$	Definição
$v_{(i)}$	Indica o ***i***-ésimo elemento do vetor v
$v_{(i:j)}$	Indica um vetor composto do elemento i até o elemento j do vetor v
$M_{(i,j)}$	Indica o elemento (i, j) da matriz M
$M_{(i:j,k:l)}$	Indica uma matriz composta pelos elementos contidos nas linhas de i até j e nas colunas de k até l da matriz M
I_n	Matriz identidade de dimensão $n \times n$
$\mathbf{1}_n$	Matriz de uns de dimensão $n \times n$
J_n	Matriz anti-identidade de dimensão $n \times n$
0_n	Matriz nula de dimensão $n \times n$
$0_{n,m}$	Matriz nula de dimensão $n \times m$
$\mathbf{1}_{n,m}$	Matriz de uns de dimensão $n \times m$
$v^\top$	Vetor transposto de v
$M^\top$	Matriz transposta de M
$\mathbb{R}^n$	Espaço euclidiano de dimensão n
$\mathbb{R}^{n\times m}$	Conjunto das matrizes reais de dimensão $n \times m$
$\mathbb{S}^n$	Conjunto das matrizes simétricas de dimensão $n \times n$
$\mathbb{D}^n$	Conjunto das matrizes diagonais de dimensão $n \times n$
$\mathbb{P}^n$	Conjunto das matrizes positivas e simétricas de dimensão $n \times n$
$\text{He}(M)$	Soma com a matriz transposta, $\text{He}(M) := M + M^\top$
$\text{tr}(M)$	Traço da matriz M, considerando $M \in \mathbb{R}^{n\times n}$, $\text{tr}(M) := \sum_{i=1}^{n} M_{(i,i)}$
$\text{diag}(v)$	Matriz diagonal do vetor v
$\preccurlyeq$	Maior ou igual definido elemento a elemento

$\|\|x\|\|$	Norma euclidiana de x
$\ln(x)$	Logaritmo natural de x
$a \bmod b$	Resto da divisão (módulo) de a por b
$\lfloor \frac{a}{b} \rfloor$	Função piso da divisão (divisão inteira) de a por b
$\otimes$	Produto de Kronecker
x^+	Sinal x no próximo instante de tempo, $x(k+1)$
x^-	Sinal x no instante anterior de tempo, $x(k-1)$

SUMÁRIO

1 INTRODUÇÃO 12

2 REDES NEURAIS 16
2.1 Visão Geral 16
2.2 RN Feedforward 19
2.3 RN Residual 22
2.4 RN Implícita 23
2.5 Interconexão de RNs ReLU 25
2.5.1 Conexão Paralela 25
2.5.2 Conexão em Série 25
2.5.3 Conexão de Realimentação 26
2.6 Funções que Podem Ser Descritas como RN ReLU 27
2.6.1 Saturação como RN ReLU 27
2.6.2 Zona Morta como RN ReLU 29
2.6.3 MPC como RN ReLU 30
2.7 Conclusões 33

3 FORMULAÇÃO DO PROBLEMA E PRELIMINARES 34
3.1 Formulação do Problema 34
3.2 Propriedades da Função ReLU 35
3.3 Função de Lyapunov Candidata 37
3.4 Positividade de Funções PWQ 38
3.5 Conclusões 40

4 ESTABILIDADE GLOBAL 42
4.1 Condições de Estabilidade 42
4.2 Exemplos Numéricos 44
4.2.1 Exemplo 1 45
4.2.2 Exemplo 2 48
4.3 Conclusões 49

5 ESTABILIDADE LOCAL 50
5.1 Condições de Estabilidade 50
5.2 Restrições de Estados 53
5.3 Determinação de Estimativas da RAO 54
5.4 Exemplos Numéricos 55
5.4.1 Exemplo 1 56
5.5 Conclusões 61

6 EXTENSÕES 62
6.1 RN Recorrente 62
6.1.1 RNR Completamente Conectada 66
6.1.2 Atraso de Ordem Superior 67
6.1.3 Estados Generalizados 67
6.1.4 Estabilidade 68
6.1.5 Exemplos Numéricos 69
6.2 RN Convolucional 73
6.2.1 Camada de Vetorização e Camada Densa 74
6.2.2 Camada de Convolução 75
6.2.3 Agrupamento 79
6.2.4 Ativação 86
6.2.5 Preenchimento 87
6.3 Outras Funções de Ativação 89
6.3.1 Representação Implícita ReLU de RN com Função de Ativação Genérica . 90
6.3.2 *Leaky* ReLU e *Parametric* ReLU 92
6.3.3 Tangente Hiperbólica 93
6.4 Conclusões 95

7 CONCLUSÕES 96
7.1 Publicações Associadas à Dissertação 97

REFERÊNCIAS 99

1 INTRODUÇÃO

Redes Neurais (RN) e aplicações de inteligência computacional têm uma base teórica e conceitos estabelecidos desde 1960 (ROSENBLATT, 1958). Contudo, foi apenas com os avanços recentes nas tecnologias da informação, especialmente o aumento do poder computacional e a otimização dos processos para a geração, análise e armazenamento de dados, que o treinamento das RNs e, consequentemente, a sua aplicação em problemas reais tornaram-se viáveis (AGGARWAL, 2019). Como resultado, o uso de RNs para o controle de sistemas dinâmicos tem sido amplamente investigado na academia, particularmente para o controle de sistemas não lineares ou sistemas lineares sujeitos a não linearidades (WANG; HUANG, 2005; LIU *et al.*, 2023; YANG *et al.*, 2020), uma vez que as RNs possuem alta capacidade de modelar funções não lineares.

Além de serem usadas diretamente como controladores, as RNs também podem ser utilizadas para emular o comportamento de controladores mais complexos ou que envolvam problemas de otimização, como por exemplo o MPC (do inglês *Model Predictive Control*, ou Controle Preditivo Baseado em Modelo). Frequentemente, tais controladores enfrentam limitações em sistemas com taxas de amostragem elevadas, nos quais pode não ser possível realizar o cálculo da ação de controle dentro do período de amostragem devido à complexidade do problema de otimização a ser resolvido (QIN; BADGWELL, 2003; CAMACHO; BORDONS, 2007). Nesse contexto, RNs podem ser treinadas para emular a lei de controle do MPC, reduzindo drasticamente o esforço computacional em tempo real. Entretanto, não há garantia de que a RN terá o comportamento desejado em todas as condições de operação e que o sistema em Malha Fechada (MF) será estável.

Certificar a estabilidade nessas aplicações de controle é fundamental para garantir que o sistema responda de forma segura e previsível. Assim, garante-se que até mesmo pequenas perturbações ou variações na entrada ou nas condições iniciais não resultem em uma saída imprevisível ou ilimitada, o que pode resultar em danos materiais, falhas catastróficas ou riscos à vida, dependendo do sistema.

Extrapolando o campo do controle, a análise da estabilidade de RNs tem utilidade em sistemas de memória associativa. Nesses sistemas, os pontos de equilíbrio representam padrões armazenados, e suas respectivas regiões de atração definem a capacidade da rede

de recuperá-los com confiabilidade. Nesse contexto, a análise de estabilidade assegura que cada ponto de equilíbrio (padrão armazenado) seja robusto a perturbações dentro de seu conjunto de atração, o que possibilita a recuperação precisa de dados mesmo sob a influência de ruídos ou entradas parciais (LIU; WANG; ZENG, 2023; DAMAK; HAMMAMI, 2026).

Contudo, sistemas baseados em RNs ainda enfrentam barreiras quanto à certificação de estabilidade e segurança, dada a complexidade estrutural desses modelos. Uma possível abordagem é utilizar conjuntos positivamente invariantes para estimar fronteiras de atração e garantir atratores exponencialmente estáveis, como feito em (ZHANG *et al.*, 2009) e (SHEN; PETERSEN, 2016).

Dado que as funções de ativação são, em geral, funções delimitadas por setores, RNs podem ser modeladas como sistemas de Lurie generalizados. Sob essa perspectiva, trabalhos como (YIN; SEILER; ARCAK, 2022) e (WU; CHEN; CHEN, 2022) utilizam restrições quadráticas integrais para formular condições de estabilidade. Também utilizando condições de setor, (RICHARDSON; TURNER; GUNN, 2023) apresenta restrições quadráticas que aprimoram critérios clássicos, como os critérios do círculo e de Popov, para um sistema de Lurie especializado. A caracterização de funções de ativação por meio de seus limites setoriais e de inclinação leva, entretanto, a condições tipicamente conservadoras, uma vez que tais limites são válidos não apenas para a função de ativação específica em análise, mas também para um amplo conjunto de não linearidades que compartilham as mesmas restrições. Além disso, a modelagem da função ReLU como uma não linearidade limitada em setor impede o estreitamento setorial para a derivação de condições locais, como feito em funções como tangente hiperbólica (Tanh), sigmoide ou de saturação (YIN; SEILER; ARCAK, 2022).

Para mitigar esse conservadorismo, esforços recentes focam em propriedades específicas, como a não negatividade e a complementaridade da função ReLU (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2019, 2023). Em (EBIHARA *et al.*, 2021a), os autores exploram a positividade da ReLU para computar o ganho induzido ℓ_2 positivo para RNs Recorrentes (RNR) ReLU de camada única e, posteriormente, em (EBIHARA *et al.*, 2021b) consideram multiplicadores co-positivos em um arcabouço de restrições quadráticas integrais. Restrições quadráticas também são utilizadas para obter condições suficientes para a estabilidade e o desempenho em norma ℓ_2 de RNRs ReLU multicamadas em (NOORI *et al.*, 2024, 2025).

Considerando apenas RNs ReLU com uma única camada, (SAMANIPOUR; POONAWALA, 2023) propõe uma representação exata de RN por meio de uma representação Afim por Partes (PWA, do inglês *Piecewise Affine*) explícita (JOHANSSON, 2003). O estudo busca uma função de Lyapunov linear por partes, formulando um problema de programação linear para os parâmetros da função de Lyapunov. No entanto, estender esse método para redes com mais camadas pode ser desafiador devido às exigências da

representação explícita e enumeração de regiões.

Em conclusão, a literatura apresenta limitações importantes: a maioria das abordagens ignora termos de *bias*, restringe-se a redes de camada única ou possui apenas métodos para a avaliação da estabilidade global para RNs ReLU. A análise da estabilidade local é fundamental, visto que a garantia de estabilidade global nem sempre é possível devido a restrições intrínsecas do sistema ou a existência de múltiplos pontos de equilíbrio. Nesses casos, a verificação da estabilidade local e a estimativa da região de atração definem um domínio operacional onde o desempenho do sistema é garantido.

Diante disso, este trabalho propõe uma metodologia para analisar a estabilidade exponencial da origem de sistemas dinâmicos em tempo discreto, cujo campo vetorial é dado por uma RN ReLU. Utilizando o fato de que a função ReLU é uma função PWA (JOHANSSON, 2003), é utilizada a representação implícita baseada em funções rampa vetoriais (ou seja, funções ReLU vetoriais) para sistemas PWA, a qual foi introduzida em (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2019). Com base nessa representação e nas técnicas expostas em (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2023) e (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025), é utilizada uma função de Lyapunov Quadrática por Partes (PWQ, do inglês *Piecewise Quadratic*) para formular condições de estabilidade baseadas em Programação Semidefinida (SDP, do inglês *Semidefined Programming*).

Diferentemente das abordagens mencionadas anteriormente, o proposto neste trabalho permite tanto a certificação da estabilidade global quanto da estabilidade local do ponto de equilíbrio do sistema. Na estabilidade local, as condições permitem calcular uma estimativa da região de atração do ponto de equilíbrio, dada por um conjunto delimitado pela superfície de nível da função de Lyapunov PWQ.

A análise de estabilidade proposta neste trabalho destaca-se por utilizar uma representação exata da função ReLU, de forma a obter resultados menos conservadores em comparação com abordagens que utilizam condições de setor e limitações de inclinação (*slope boundedness*) para considerar o comportamento não linear induzido pelas funções ReLU, como demonstrado em (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2022). Além disso, a utilização dessa representação possibilita a aplicação do método para RNs de múltiplas camadas, com termos de *bias* e diferentes arquiteturas, como RNs Recorrentes e RNs Convolucionais. O método oferece ainda a vantagem de modularidade, sendo capaz de representar e acoplar camadas de RN com arquiteturas variadas. Essa característica permite traduzir, de forma direta, a estrutura multicamadas e os diversos paradigmas de conectividade das RNs, facilitando a modelagem de sistemas heterogêneos.

Com este escopo, a organização deste trabalho se dá da seguinte forma: no Capítulo 2, é introduzido o conceito de RNs e apresentada a representação implícita baseada em funções ReLU a ser utilizada. Nele, também é apresentada a formulação para a interconexão de RNs na representação implícita proposta e como escrever algumas funções comuns em

aplicações de controle com RNs ReLU. No Capítulo 3, é apresentado o problema a ser tratado, junto com alguns desenvolvimentos preliminares utilizados para abordá-lo. Na sequência, nos capítulos 4 e 5, são apresentadas, respectivamente, as condições para a estabilidade global e local do sistema dinâmico em tempo discreto cujo campo vetorial é dado por uma RN ReLU, seguidos de alguns exemplos numéricos nos quais essas condições são utilizadas. Já no Capítulo 6, são apresentadas extensões dos métodos propostos, de forma a abordar RNRs, RNCs e RNs com funções de ativação diferentes da ReLU. Por fim, no Capítulo 7, são apresentadas conclusões sobre os resultados e possíveis trabalhos futuros.

2 REDES NEURAIS

Neste capítulo, são estabelecidos os fundamentos teóricos sobre Redes Neurais (RNs) e a representação implícita baseada em funções ReLU a ser utilizada. Inicialmente, na Seção 2.1, são introduzidos os conceitos fundamentais de RNs e delimita-se o escopo deste trabalho às arquiteturas com função de ativação ReLU. Na Seção 2.2, apresenta-se a arquitetura *feedforward* para RNs e como se pode escrevê-la na representação implícita proposta. Nas seções 2.3 e 2.4, são apresentadas, respectivamente, as arquiteturas residual e implícita para RNs, assim como suas representações na formulação implícita utilizada. Na Seção 2.5, mostra-se como representar diferentes conexões entre RNs na representação implícita, incluindo associações em série, paralelo e realimentação. Por fim, na Seção 2.6, mostra-se como representar algumas funções comumente utilizadas em aplicações de controle, como saturação, zona morta e controladores MPC, por meio de RNs ReLU.

2.1 Visão Geral

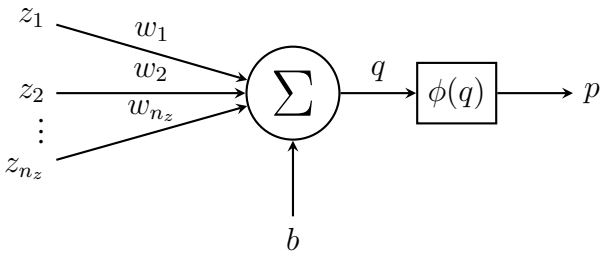
As RNs artificiais (neste trabalho chamadas apenas de RNs por simplicidade) são modelos inspirados no comportamento das RNs biológicas do cérebro humano e buscam modelar as capacidades de processamento de informações dos sistemas nervosos (ROJAS, 1996). Segundo (HAYKIN, 2001), uma RN é um processador distribuído paralelamente constituído de unidades de processamento simples, com propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Uma RN é composta de diversos neurônios baseados no modelo *Perceptron* de um neurônio artificial (ROSENBLATT, 1958). Esse neurônio, apresentado na Figura 1, é composto por um conjunto n_z de entradas $z_i, i = 1, \ldots, n_z$, seus respectivos pesos $w_i, i = 1, \ldots, n_z$, um valor de *bias* b e uma função de ativação ϕ.

A saída desse neurônio, p, é o resultado da aplicação da função de ativação a uma variável q, correspondente ao somatório das n_z entradas multiplicadas pelos seus pesos e somadas ao *bias*, ou seja,

$$p = \phi\left(\sum_{i=1}^{n_z} w_i z_i + b\right). \tag{1}$$

Em uma RN, esses neurônios são conectados de forma que as entradas de um neurônio

Figura 1 – Exemplo de neurônio artificial.

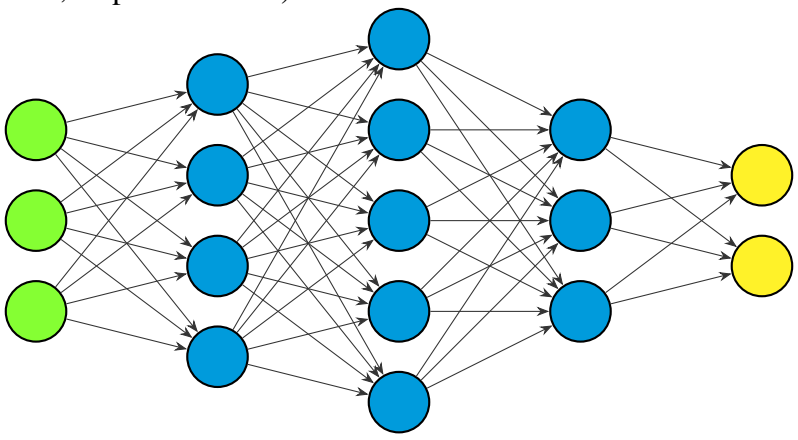


Fonte: Autor

são compostas pelas saídas de outros. A alta complexidade desses sistemas, gerada pela interconexão dos diferentes neurônios, torna as RNs uma ótima ferramenta para aproximação e/ou controle de sistemas dinâmicos não lineares, como visto em (LEE *et al.*, 2020; ZHOU *et al.*, 2019; BOSE, 2007; LI *et al.*, 2015).

Em uma RN, o processamento das informações ocorre tipicamente por meio de camadas organizadas em sequência. A camada de entrada é responsável por receber os dados brutos, como características de imagens ou valores numéricos, e repassá-los para as próximas etapas. As camadas ocultas ficam entre a camada de entrada e a camada de saída; nelas ocorre o principal processamento, onde os neurônios combinam e transformam os sinais recebidos, extraindo padrões e relações complexas. Por fim, a camada de saída produz o resultado final da rede, como uma classificação, uma previsão numérica ou uma decisão, com base nas transformações realizadas pelas camadas anteriores (JAIN; MAO; MOHIUDDIN, 1996). Na Figura 2 é apresentado um exemplo de uma RN dividida em camadas, em que os neurônios são representados pelos círculos coloridos e as conexões entre eles, por setas. A cor verde representa a camada de entrada, a azul as camadas ocultas e a amarela a camada de saída.

Figura 2 – Exemplo de uma RN com 3 entradas, 2 saídas e 3 camadas ocultas (contendo 4, 5 e 3 neurônios, respectivamente).



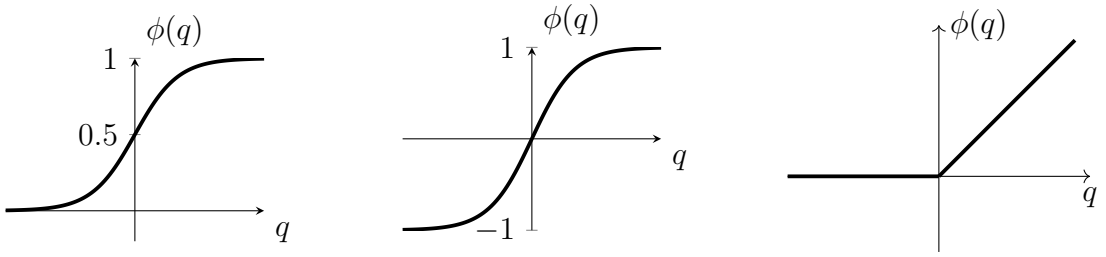
Fonte: Autor

Para que uma RN apresente um desempenho satisfatório, seja na aproximação de

funções e sistemas ou no processamento de dados para predições e controle, é necessário calibrar os valores de seus pesos e *biases*. O processo de otimização desses parâmetros internos, visando o desempenho desejado, é chamado de treinamento. Para fins de concisão, o treinamento de RNs não constitui o escopo principal deste trabalho, detalhes técnicos sobre algoritmos de otimização e metodologias de aprendizado podem ser consultados em (GOODFELLOW; BENGIO; COURVILLE, 2016) e (LECUN *et al.*, 1998).

Uma RN pode empregar diferentes funções de ativação em suas camadas ou neurônios. A definição de quais funções serão utilizadas é condicionada à arquitetura da rede e à natureza da tarefa proposta. Entre as funções de ativação mais comuns para as camadas ocultas, pode-se citar a sigmoide, a Tanh e a ReLU. O objetivo dessas funções é introduzir não-linearidades no modelo, permitindo que a rede aprenda padrões complexos. Historicamente, a sigmoide e a Tanh foram as escolhas predominantes devido à sua forma suave e limitada. Contudo, em arquiteturas modernas, a função ReLU tornou-se o padrão por ser matematicamente mais simples e por acelerar significativamente o processo de treinamento da rede (GOODFELLOW; BENGIO; COURVILLE, 2016). Na Figura 3 são apresentados os gráficos de algumas dessas funções de ativação.

Figura 3 – Gráfico de funções de ativação típicas de RN.



(a) Sigmoide (b) Tanh (c) ReLU

Fonte: Autor

Já para a função de ativação da camada de saída, em problemas de classificação, costuma-se utilizar a função *softmax*,

$$\phi_{(j)}(q) = \frac{\exp(q_{(j)})}{\sum_{i=1}^{n_q} \exp(q_{(i)})}, \;\; j = 1, \ldots, n_v, \tag{2}$$

onde $n_v = n_q$, de forma que cada uma das saídas da camada final da RN apresentará um resultado entre 0 e 1 que indicará a probabilidade dos dados de entrada serem de uma determinada classe e então a classe com maior probabilidade é escolhida como resultado da classificação. Já, para problemas de regressão/emulação, normalmente são utilizadas funções de ativação lineares, ou seja

$$\phi(q) = q, \tag{3}$$

pois deseja-se que a saída da RN não seja restrita (como é o caso quando são utilizadas

as funções de ativação como sigmoide e *softmax*) e que possa assumir qualquer valor (BISHOP, 2006).

Um ponto importante a ser ressaltado sobre as funções de ativação das RN é que algumas delas são não linearidades de setor, como é o caso da função ReLU e da função Tanh. Portanto, podem ser modeladas como tal e utilizadas técnicas desenvolvidas para essa classe de não linearidades para derivar condições de estabilidade para sistemas que contenham tais RN (YIN; SEILER; ARCAK, 2022; FAZLYAB; MORARI; PAPPAS, 2022). Porém, esta abordagem envolve certo conservadorismo na análise, pois acaba certificando a estabilidade para todas as possíveis não linearidades no setor.

Neste trabalho, serão considerados dois tipos distintos de funções de ativação: uma função de ativação associada às camadas ocultas da rede, denotada por ϕ, e uma função de ativação associada à camada de saída, denotada por ψ. A partir deste ponto, a não ser que dito o contrário, as RNs representadas neste trabalho tem em sua camada de saída uma função de ativação linear, dada por

$$\psi(q) = q, \tag{4}$$

enquanto as camadas ocultas tem uma função de ativação ReLU. Assim, ϕ é uma função ReLU vetorial, definida elemento a elemento a partir da função rampa escalar $r : \mathbb{R} \to \mathbb{R}$, de modo que

$$\phi_{(j)}(q) = r(q_{(j)}) := \begin{cases} 0 & \text{se } q_{(j)} < 0, \\ q_{(j)} & \text{se } q_{(j)} \geq 0, \quad j = 1, \ldots, n_q. \end{cases} \tag{5}$$

Por fim, é importante destacar que existem vários atributos que influenciam o treinamento e o desempenho da RN. Todos esses atributos selecionados antes do treinamento da RN são considerados hiperparâmetros, os quais consistem no número de neurônios e camadas ocultas da RN, na função de ativação de cada camada, no algoritmo de treinamento escolhido, nos parâmetros do algoritmo de treinamento, nos critérios de parada para o treinamento da RN, entre outros. A escolha dos hiperparâmetros utilizados para a RN é uma etapa muito importante para o desenvolvimento da RN e tem grande influência no desempenho da mesma, existindo vários estudos e métodos de otimização para estes, como busca por grade, busca aleatória, algoritmo de enxame e algoritmo genético (MAN; TANG; KWONG, 1996). Porém, cabe ressaltar que o foco deste trabalho é na obtenção de critérios para a análise da estabilidade de sistemas controlados por RN e não em obter a melhor RN possível para um certo problema. Assim, será assumido que a RN é dada e foi convenientemente treinada.

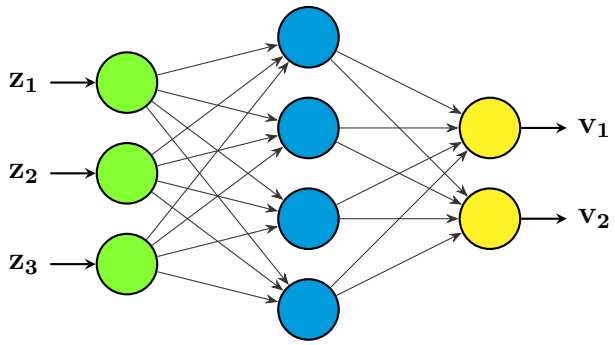
2.2 RN Feedforward

Existem diversas formas de realizar a conexão dos neurônios em uma RN. A forma mais simples e também a mais utilizada é a RN *feedforward*, caracterizada pelo fluxo

de informações em uma única direção: da camada de entrada, passando pelas camadas ocultas até a camada de saída. Essa arquitetura têm seus neurônios agrupados em camadas e a conexão entre eles se dá de forma que as entradas dos neurônios de uma camada i são as saídas dos neurônios da camada anterior $i-1$.

Na prática, as RNs *feedforward* são empregadas em uma ampla variedade de domínios, abrangendo desde tarefas de classificação de dados financeiros até o reconhecimento de padrões em sinais biológicos. Um aspecto importante dessas redes é o fato de constituírem unidades estruturais elementares para arquiteturas mais complexas. Em RNs Convolucionais, por exemplo, as camadas finais costumam adotar uma topologia *feedforward*, sendo responsáveis pela etapa de classificação após a extração de características (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Em modelos de linguagem avançados, uma vez analisado o contexto semântico e sintático das palavras, é tipicamente uma camada *feedforward* que processa essa informação para estimar a próxima palavra da sequência (VASWANI *et al.*, 2017). Dessa forma, mesmo em sistemas de inteligência artificial de ponta, a arquitetura *feedforward* permanece essencial para a transformação e integração final dos dados (GOODFELLOW; BENGIO; COURVILLE, 2016). Um exemplo deste tipo de RN pode ser visto na Figura 4.

Figura 4 – RN *feedforward* com 3 entradas (z_1, z_2, z_3), 2 saídas (v_1, v_2) e 1 camada oculta com 4 neurônios.



Fonte: Autor

Observação 1. *Um fato importante sobre as RNs feedforward é que elas são aproximadores universais, podendo aproximar, com precisão arbitrária, qualquer função contínua em subconjuntos compactos, desde que possuam um número suficientemente grande de neurônios e/ou camadas (HORNIK; STINCHCOMBE; WHITE, 1989; LESHNO et al., 1993). A aplicação prática dessa propriedade, no entanto, depende da viabilidade do tamanho da rede e da capacidade dos algoritmos de treinamento em encontrar tais soluções.*

Uma RN *feedforward* de ℓ camadas ocultas ($\ell \geq 1 \in \mathbb{N}$) pode ser representada mate-

maticamente pelas seguintes equações (YIN; SEILER; ARCAK, 2022)

$$\begin{aligned} p_0 &= z, \\ p_i &= \phi(W_i p_{i-1} + b_i), \quad i = 1, \ldots, \ell, \\ v &= \psi(W_{\ell+1} p_\ell + b_{\ell+1}), \end{aligned} \tag{6}$$

onde $z \in \mathbb{R}^{n_z}$ é o vetor de entradas da RN, $v \in \mathbb{R}^{n_v}$ é o vetor de saída da RN, $p_i \in \mathbb{R}^{n_{p_i}}$ é o vetor de saída da i-ésima camada (ou seja, a saída dos n_{p_i} neurônios da i-ésima camada), $b_i \in \mathbb{R}^{n_{p_i}}$ é o vetor dos *bias* e $W_i \in \mathbb{R}^{n_{p_i} \times n_{p_{i-1}}}$ é a matriz dos pesos da i-ésima camada.

Além disso, estas RNs com função de ativação ReLU, são genericamente denotadas por um mapeamento entrada-saída $f_{\text{ReLU}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_v}$, ou seja $v = f_{\text{ReLU}}(z)$. Neste caso, f_{ReLU} é uma função Afim por Partes (PWA) (PRINCE, 2023), que pode ser reescrita utilizando a representação implícita baseada em rampas introduzida em (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2019). Para isso, define-se $q_i := W_i p_{i-1} + b_i$ como a entrada da função de ativação vetorial da i-ésima camada e os vetores $q \in \mathbb{R}^{n_q}$ e $p \in \mathbb{R}^{n_p}$ como a concatenação das entradas e saídas da função de ativação de cada camada, respectivamente, isto é

$$q := \begin{bmatrix} q_1 \\ \vdots \\ q_\ell \end{bmatrix} \quad \text{e} \quad p := \begin{bmatrix} p_1 \\ \vdots \\ p_\ell \end{bmatrix}, \tag{7}$$

onde $n_q = n_p = n_{p_1} + \cdots + n_{p_\ell}$. Assim, segue que $p = \phi(q)$ e

$$\begin{bmatrix} q \\ v \end{bmatrix} = N \begin{bmatrix} z \\ p \end{bmatrix} + e, \tag{8}$$

com

$$\begin{aligned} N &:= \left[\begin{array}{c|cccc} W_1 & 0_{n_{p_1}, n_{p_1}} & \cdots & 0_{n_{p_1}, n_{p_{\ell-1}}} & 0_{n_{p_1}, n_{p_\ell}} \\ 0_{n_{p_2}, n_z} & W_2 & \cdots & 0_{n_{p_2}, n_{p_{\ell-1}}} & 0_{n_{p_2}, n_{p_\ell}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{n_{p_\ell}, n_z} & 0_{n_{p_\ell}, n_{p_1}} & \cdots & W_\ell & 0_{n_{p_\ell}, n_{p_\ell}} \\ \hline 0_{n_v, n_z} & 0_{n_v, n_{p_1}} & \cdots & 0_{n_v, n_{p_{\ell-1}}} & W_{\ell+1} \end{array}\right] = \begin{bmatrix} N_3 & N_4 \\ N_1 & N_2 \end{bmatrix} \text{ e} \\ e &:= \left[\begin{array}{cccc|c} b_0^\top & b_1^\top & \ldots & b_{\ell-1}^\top & b_\ell^\top \end{array}\right]^\top = \begin{bmatrix} e_1^\top & e_0^\top \end{bmatrix}^\top. \end{aligned} \tag{9}$$

Então, a representação implícita baseada em rampas da função PWA f_{ReLU} é dada por

$$f_{\text{ReLU}}(z) = N_1 z + N_2 \phi(q) + e_0 \tag{10a}$$

$$q = N_3 z + N_4 \phi(q) + e_1, \tag{10b}$$

com $N_1 \in \mathbb{R}^{n_v \times n_z}$, $N_2 \in \mathbb{R}^{n_v \times n_q}$, $N_3 \in \mathbb{R}^{n_q \times n_z}$, $N_4 \in \mathbb{R}^{n_q \times n_q}$, $e_0 \in \mathbb{R}^{n_v}$, e $e_1 \in \mathbb{R}^{n_q}$.

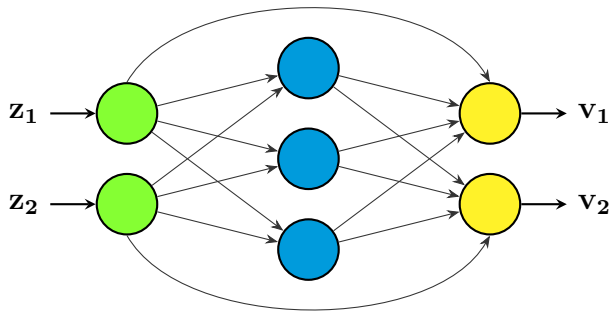
Observação 2. *Para redes feedforward, observa-se que* $N_1 = 0_{n_v,n_z}$. *Entretanto, esse termo não é omitido de* (10)*, pois tal forma implícita será empregada posteriormente para representar outras arquiteturas de RNs, nas quais o termo* N_1 *pode ser não nulo.*

Observação 3. *Note que* (10b) *é, em geral, uma equação implícita, uma vez que* q *depende de* $\phi(q)$. *No entanto, graças ao fato de* N_4 *ser uma matriz estritamente triangular inferior, consequência da estrutura da RN feedforward,* (10b) *admite uma solução única para todo* $z \in \mathbb{R}^{n_z}$. *Esta estrutura da matriz* N_4 *é mantida na maioria das arquiteturas de RNs.*

2.3 RN Residual

As RN residuais são arquiteturas de aprendizagem profunda que introduzem o conceito de "atalhos" para permitir que o sinal "pule" uma ou mais camadas. Essa técnica surgiu para resolver problemas de degradação em redes muito profundas, onde um maior número de camadas costumava resultar em erros maiores no treinamento devido ao desaparecimento do gradiente [1] Seus principais usos são em aplicações de visão computacional, servindo como base para tarefas de classificação de imagens, detecção de objetos e segmentação semântica (HE *et al.*, 2016). Um exemplo de RN residual pode ser visto na Figura 5.

Figura 5 – RN residual com 2 entradas, 2 saídas e 1 camada oculta com 3 neurônios. Conexões residuais de z_1 para v_1 e z_2 para v_2.



Fonte: Autor

[1] Fenômeno em que as informações de ajuste dos pesos diminuem drasticamente ao atravessar muitas camadas, dificultando a passagem dessa informação para a correção de camadas iniciais em redes profundas.

Pode-se utilizar a seguinte representação matemática para uma RN residual

$$\begin{aligned} p_0 &= z, \\ p_i &= \phi\left(\sum_{j=0}^{i-1} W_{ji} p_j + b_i\right), \quad i = 1, \ldots, \ell, \\ v &= \psi\left(\sum_{j=0}^{\ell} W_{j(\ell+1)} p_j + b_{\ell+1}\right), \end{aligned} \tag{11}$$

onde $z \in \mathbb{R}^{n_z}$ é o vetor de entradas da RN, $v \in \mathbb{R}^{n_v}$ é o vetor de saída da RN, e para a i-ésima camada $p_i \in \mathbb{R}^{n_{p_i}}$ é o vetor de saída (ou seja, a saída dos n_{p_i} neurônios da *i*-ésima camada), $b_i \in \mathbb{R}^{n_{p_i}}$ é o vetor dos *bias* e $W_{ji} \in \mathbb{R}^{n_{p_i} \times n_{p_j}}$ é a matriz dos pesos relativa à influência da camada j na camada i, considerando $i > j$. Desta forma, semelhante ao realizado para RNs *feedforward*, a RN pode ser expressa por

$$\begin{bmatrix} q \\ v \end{bmatrix} = N \begin{bmatrix} z \\ p \end{bmatrix} + e, \tag{12}$$

onde

$$N := \left[\begin{array}{c|ccc|c} W_{01} & 0_{n_{p_1}} & \ldots & 0_{n_{p_1}, n_{p_{\ell-1}}} & 0_{n_{p_1}, n_{p_\ell}} \\ W_{02} & W_{12} & \ldots & 0_{n_{p_2}, n_{p_{\ell-1}}} & 0_{n_{p_2}, n_{p_\ell}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ W_{0\ell} & W_{1\ell} & \ldots & W_\ell & 0_{n_{p_\ell}} \\ \hline W_{0(\ell+1)} & W_{1(\ell+1)} & \ldots & W_{(\ell-1)(\ell+1)} & W_{\ell(\ell+1)} \end{array}\right] = \begin{bmatrix} N_3 & N_4 \\ N_1 & N_2 \end{bmatrix} \text{ e}$$
$$e := \left[\begin{array}{ccc|c} b_0^\top & b_1^\top & \ldots & b_\ell^\top \end{array}\right]^\top = \begin{bmatrix} e_1^\top & e_0^\top \end{bmatrix}^\top. \tag{13}$$

A partir destas novas matrizes N_1, N_2, N_3, N_4 e vetores e_0 e e_1, pode-se escrever a RN residual na forma implícita baseada na função ReLU utilizando (10).

2.4 RN Implícita

Diferentemente das arquiteturas tradicionais de RNs que são baseadas no empilhamento de camadas explícitas, as RNs implícitas (também conhecidas como *Deep Equilibrium Models*, DEQs) introduzem o conceito de camada implícita. Nestas camadas, ao invés de processar a entrada por meio de uma sequência fixa de transformações, a saída é definida como a solução de uma equação não linear implícita, estabelecendo uma relação de dependência mútua entre seus neurônios. A principal vantagem dessa arquitetura reside na sua capacidade de modelar dinâmicas complexas cuja representação exigiria o empilhamento de múltiplas camadas em redes *feedforward* convencionais (BAI; KOLTER; KOLTUN, 2019). Contudo, sua implementação impõe desafios significativos, como a complexidade dos algoritmos para a resolução da equação implícita, resultando em um elevado custo computacional por iteração.

Matematicamente, uma DEQ com uma camada implícita pode ser representada da seguinte maneira

$$\begin{aligned} &\text{Encontrar } p = p^* \text{ tal que: } p = \phi(W_{pp}p + W_{pz}z + b_p), \\ &v = \psi(W_{vp}p^* + W_{vz}z + b_v), \end{aligned} \tag{14}$$

onde $z \in \mathbb{R}^{n_z}$ é o vetor de entradas da RN, $v \in \mathbb{R}^{n_v}$ é o vetor de saída da RN, e $p \in \mathbb{R}^{n_p}$ é o vetor de saída dos neurônios da camada implícita, $p^* \in \mathbb{R}^{n_p}$ é a solução da equação implícita $p = \phi(W_{pp}p + W_{pz}z + b_p)$, $W_{pp} \in \mathbb{R}^{n_p \times n_p}$ é a matriz dos pesos em relação aos neurônios da camada implícita, $W_{pz} \in \mathbb{R}^{n_p \times n_z}$ é a matriz dos pesos em relação a entrada da camada implícita, $W_{vp} \in \mathbb{R}^{n_v \times n_p}$ é a matriz dos pesos da influência da camada implícita na saída da RN, $W_{vz} \in \mathbb{R}^{n_v \times n_z}$ é a matriz dos pesos da influência da entrada na saída da RN , $b_p \in \mathbb{R}^{n_p}$ é o vetor dos *bias* da camada implícita e $b_v \in \mathbb{R}^{n_v}$ é o vetor dos *bias* da camada de saída.

Desta forma, semelhante ao realizado para RNs *feedforward*, define-se $q := W_{pp}p + W_{pz}z + b_p$ como a entrada da função de ativação vetorial da camada implícita. Assim, segue que $p = \phi(q)$ e a RN implícita pode ser expressa por

$$\begin{bmatrix} q \\ v \end{bmatrix} = N \begin{bmatrix} z \\ p \end{bmatrix} + e, \tag{15}$$

onde

$$N := \left[\begin{array}{c|c} W_{pz} & W_{pp} \\ \hline W_{vz} & W_{vp} \end{array}\right] = \begin{bmatrix} N_3 & N_4 \\ N_1 & N_2 \end{bmatrix} \text{ e } e := \left[\begin{array}{c} b_p \\ \hline b_v \end{array}\right] = \begin{bmatrix} e_1 \\ e_0 \end{bmatrix}. \tag{16}$$

A partir destas novas matrizes N_1, N_2, N_3, N_4 e vetores e_0 e e_1, pode-se escrever a DEQ implícita na forma implícita baseada na função ReLU utilizando (10).

Observação 4. *Ao contrário do que ocorre em outras arquiteturas, nas RNs implícitas o fato de* (10b) *ser uma equação implícita significa que* (10b) *deve ser bem posta (ter solução e a solução ser única) apenas para um conjunto específico de entradas $z \in \mathcal{W} \subset \mathbb{R}^{n_z}$. Isto ocorre porque nas DEQs a matriz N_4 não tem a estrutura estritamente triangular inferior, o que impede uma resolução sequencial direta.*

Existem diversos métodos para resolver a equação implícita (14). Isso pode ser feito por meio de métodos iterativos simples, como a iteração de ponto fixo (BURDEN; FAIRES, 2010), ou algoritmos de busca de raiz mais robustos, como, por exemplo, o método de Newton (KELLEY, 2003) e a aceleração de Anderson (ANDERSON, 1965). Quando a função de ativação da camada implícita é uma ReLU, esse problema pode ser modelado como um Problema de Complementaridade Linear (LCP, do inglês *Linear Complementarity Problem*) (COTTLE; PANG; STONE, 2009). Essa equivalência ocorre porque o comportamento da função ReLU impõe restrições de desigualdade e complementaridade idênticas às de um LCP (HEEMELS; SCHUMACHER; WEILAND, 2000). Dessa forma,

é possível empregar algoritmos especializados que garantem a identificação de uma solução condizente com as descontinuidades da ativação, conferindo maior rigor matemático à convergência da camada implícita.

2.5 Interconexão de RNs ReLU

Nesta seção, é mostrado como representar diferentes composições de funções f_{ReLU} na forma (10). Tal procedimento é relevante, pois possibilita a formulação sistemática das conexões entre RNs, bem como entre RNs e outras classes de funções e sistemas dinâmicos.

Para isso, consideramos duas RNs ReLU gerais, $f^a_{\text{ReLU}} : \mathbb{R}^{n_{z_a}} \to \mathbb{R}^{n_{v_a}}$ e $f^b_{\text{ReLU}} : \mathbb{R}^{n_{z_b}} \to \mathbb{R}^{n_{v_b}}$ descritas genericamente como

$$\begin{aligned} f^a_{\text{ReLU}}(z_a) &= N_{1a} z_a + N_{2a}\phi(q_a) + e_{0a} \\ q_a &= N_{3a} z_a + N_{4a}\phi(q_a) + e_{1a}, \end{aligned} \tag{17}$$

e

$$\begin{aligned} f^b_{\text{ReLU}}(z_b) &= N_{1b} z_b + N_{2b}\phi(q_b) + e_{0b} \\ q_b &= N_{3b} z_b + N_{4b}\phi(q_b) + e_{1b}. \end{aligned} \tag{18}$$

2.5.1 Conexão Paralela

Assumindo que $n_{z_a} = n_{z_b}$ e $n_{v_a} = n_{v_b}$. A conexão em paralelo de f^a_{ReLU} e f^b_{ReLU} é dada por

$$f_{\text{ReLU}}(z) = f^a_{\text{ReLU}}(z) + f^b_{\text{ReLU}}(z).$$

Segue-se que $f_{\text{ReLU}}(z)$ pode ser representada como em (10) com

$$\begin{aligned} &N_1 = N_{1a} + N_{1b},\ N_2 = \begin{bmatrix} N_{2a} & N_{2b} \end{bmatrix},\ e_0 = e_{0a} + e_{0b}, \\ &q = \begin{bmatrix} q_a \\ q_b \end{bmatrix},\ N_3 = \begin{bmatrix} N_{3a} \\ N_{3b} \end{bmatrix},\ N_4 = \text{diag}(N_{4a}, N_{4b}),\ \text{e}\ e_1 = \begin{bmatrix} e_{1a} \\ e_{1b} \end{bmatrix}. \end{aligned} \tag{19}$$

2.5.2 Conexão em Série

Assumindo que $n_{z_b} = n_{v_a}$, a ligação em série entre f^a_{ReLU} e f^b_{ReLU} é dada pela função composta

$$f_{\text{ReLU}}(z) = f^b_{\text{ReLU}}(f^a_{\text{ReLU}}(z)),$$

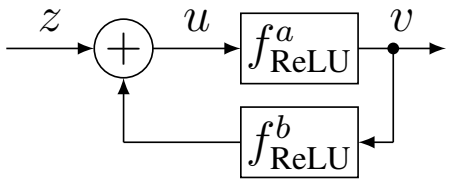
a qual pode ser descrita como em (10) com

$$\begin{aligned} &N_1 = N_{1b}N_{1a},\ N_2 = \begin{bmatrix} N_{1b}N_{2a} & N_{2b} \end{bmatrix},\ e_0 = N_{1b}e_{0a} + e_{0b}, \\ &q = \begin{bmatrix} q_a \\ q_b \end{bmatrix},\ N_3 = \begin{bmatrix} N_{3a} \\ N_{3b}N_{1a} \end{bmatrix},\ N_4 = \begin{bmatrix} N_{4a} & 0_{n_{qa},n_{qb}} \\ N_{3b}N_{2a} & N_{4b} \end{bmatrix},\ e_1 = \begin{bmatrix} e_{1a} \\ N_{3b}e_{0a} + e_{1b} \end{bmatrix}. \end{aligned} \tag{20}$$

2.5.3 Conexão de Realimentação

Supondo que $n_{v_a} = n_{z_b}$ e $n_{z_a} = n_{v_b}$, a conexão de realimentação entre f^a_{ReLU} e f^b_{ReLU} com $z \in \mathbb{R}^{n_{z_a}}$ e $v \in \mathbb{R}^{n_{v_a}}$, é ilustrada na Figura 6.

Figura 6 – Conexão de realimentação.



Fonte: Autor

Para obter a função composta resultante $f_{\text{ReLU}}(z)$, primeiro considera-se a função expandida

$$v = f^a_{\text{ReLU}}(u) = N_{1a}u + N_{2a}\phi(q_a) + e_{0a}, \tag{21a}$$
$$q_a = N_{3a}u + N_{4a}\phi(q_a) + e_{1a}, \tag{21b}$$
$$u = z + f^b_{\text{ReLU}}(v), \tag{21c}$$
$$f^b_{\text{ReLU}}(v) = N_{1b}v + N_{2b}\phi(q_b) + e_{0b}, \tag{21d}$$
$$q_b = N_{3b}v + N_{4b}\phi(q_b) + e_{1b}. \tag{21e}$$

Para obter a equação de v, substitui-se (21c) e (21d) em (21a), resultando em:

$$v = N_{1a}(z + N_{1b}v + N_{2b}\phi(q_b) + e_{0b}) + N_{2a}\phi(q_a) + e_{0a}. \tag{22}$$

Ao agrupar os termos de v no lado esquerdo da igualdade, tem-se:

$$(I - N_{1a}N_{1b})v = N_{1a}z + N_{2a}\phi(q_a) + N_{1a}N_{2b}\phi(q_b) + (e_{0a} + N_{1a}e_{0b}). \tag{23}$$

Então, definindo

$$N_v = (I - N_{1a}N_{1b})^{-1}, \tag{24}$$

e pré-multiplicando ambos os lados da equação por N_v, a variável v pode ser explicitada da seguinte maneira:

$$v = N_v N_{1a} z + N_v \begin{bmatrix} N_{2a} & N_{1a}N_{2b} \end{bmatrix} \begin{bmatrix} \phi(q_a) \\ \phi(q_b) \end{bmatrix} + N_v(e_{0a} + N_{1a}e_{0b}). \tag{25}$$

Substituindo (21c) e (21d) em (21b), obtém-se:

$$q_a = N_{3a}(z + N_{1b}v + N_{2b}\phi(q_b) + e_{0b}) + N_{4a}\phi(q_a) + e_{1a}. \tag{26}$$

E então, ao incorporar a expressão de v (25) e agrupar os termos de z, $\phi(q_a)$ e $\phi(q_b)$, tem-se:

$$\begin{aligned} q_a = {} & N_{3a}(I + N_{1b}N_vN_{1a})z + (N_{4a} + N_{3a}N_{1b}N_vN_{2a})\phi(q_a) \\ & + N_{3a}(N_{2b} + N_{1b}N_vN_{1a}N_{2b})\phi(q_b) + e_{1a} + N_{3a}(e_{0b} + N_{1b}N_v(e_{0a} + N_{1a}e_{ob})). \end{aligned} \tag{27}$$

Por fim, substituindo (25) em (21e), chega-se a:

$$\begin{aligned} q_b = {} & N_{3b}N_vN_{1a}z + N_{3b}N_vN_{2a}\phi(q_a) + (N_{4b} + N_{3b}N_vN_{1a}N_{2b})\phi(q_b) \\ & + e_{1b} + N_{3b}N_v(e_{0a} + N_{1a}e_{ob}). \end{aligned} \tag{28}$$

Assim, assumindo que a matriz de acoplamento $(I - N_{1a}N_{1b})$ é não-singular, a conexão de realimentação mostrada na Figura 6 pode ser escrita como em (10) com

$$\begin{aligned} & N_1 = N_vN_{1a},\ N_2 = N_v\begin{bmatrix} N_{2a} & N_{1a}N_{2b} \end{bmatrix},\ e_0 = N_v(e_{0a} + N_{1a}e_{0b}), \\ & N_3 = \begin{bmatrix} N_{3a}(I + N_{1b}N_vN_{1a}) \\ N_{3b}N_vN_{1a} \end{bmatrix},\ e_1 = \begin{bmatrix} e_{1a} + N_{3a}(e_{0b} + N_{1b}N_v(e_{0a} + N_{1a}e_{ob})) \\ e_{1b} + N_{3b}N_v(e_{0a} + N_{1a}e_{ob}) \end{bmatrix}, \\ & N_4 = \begin{bmatrix} N_{4a} + N_{3a}N_{1b}N_vN_{2a} & N_{3a}(N_{2b} + N_{1b}N_vN_{1a}N_{2b}) \\ N_{3b}N_vN_{2a} & N_{4b} + N_{3b}N_vN_{1a}N_{2b} \end{bmatrix},\ q = \begin{bmatrix} q_a \\ q_b \end{bmatrix}. \end{aligned} \tag{29}$$

Observação 5. *Diferentemente das conexões em série e em paralelo, a conexão de realimentação pode não ser bem-posta mesmo quando tanto f^a_{ReLU} quanto f^b_{ReLU} são individualmente bem-postas, devido à introdução de um laço algébrico.*

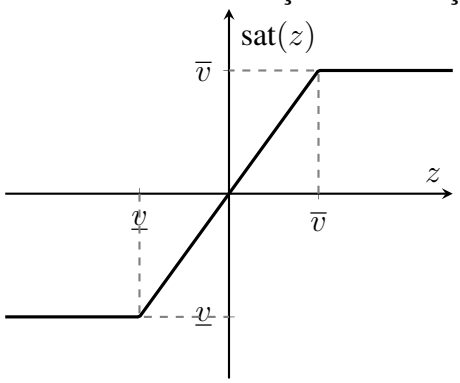
2.6 Funções que Podem Ser Descritas como RN ReLU

Nesta seção, é exposto como representar algumas funções comumente utilizadas em aplicações de controle como uma função f_{ReLU} no formato (10). Desta forma, utilizando as relações elencadas na Seção 2.5, pode-se facilmente integrar essas funções nas análises de estabilidade que serão propostas nos próximos capítulos.

2.6.1 Saturação como RN ReLU

Considerando sat : $\mathbb{R} \to \mathbb{R}$ uma função de saturação escalar, apresentada na Figura 7.

Figura 7 – Gráfico da função de saturação.



Fonte: Autor

O sinal limitado pela saturação $v \in \mathbb{R}^{n_z}$ é dado por

$$v_{(j)} = \text{sat}(z_{(j)}) := \begin{cases} \underline{v}_{(j)} & \text{se } z_{(j)} < \underline{v}_{(j)}, \\ z_{(j)} & \text{se } \underline{v}_{(j)} \leq z_{(j)} \leq \overline{v}_{(j)}, \\ \overline{v}_{(j)} & \text{se } z_{(j)} > \overline{v}_{(j)}, \end{cases} \tag{30}$$

para $j = 1, \ldots, n_z$, em que $\underline{v} \preccurlyeq \overline{v} \in \mathbb{R}^{n_z}$ são, respectivamente, os vetores que concatenam os limites inferiores e superiores de saturação de cada uma das n_z entradas (GROFF, 2020; CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2022).

Para escrever (30), observa-se que a função de saturação escalar pode ser escrita como uma soma de funções ReLU

$$\text{sat}(z_{(j)}) = r(z_{(j)} - \underline{v}_{(j)}) - r(z_{(j)} - \overline{v}_{(j)}) + \underline{v}_{(j)}. \tag{31}$$

Essa equivalência pode ser verificada considerando os três casos possíveis:

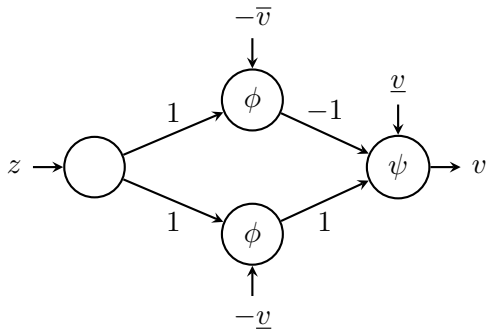
1. Saturação no limite inferior, $z_{(j)} < \underline{v}_{(j)}$:
   - $z_{(j)} - \overline{v}_{(j)} < 0 \Rightarrow r(z_{(j)} - \overline{v}_{(j)}) = 0$;
   - $z_{(j)} - \underline{v}_{(j)} < 0 \Rightarrow r(z_{(j)} - \underline{v}_{(j)}) = 0$;
   - Consequentemente $v_{(j)} = \underline{v}_{(j)}$.
2. Saturação na região linear, $\underline{v}_{(j)} \leq z_{(j)} \leq \overline{v}_{(j)}$:
   - $z_{(j)} - \overline{v}_{(j)} < 0 \Rightarrow r(z_{(j)} - \overline{v}_{(j)}) = 0$;
   - $z_{(j)} - \underline{v}_{(j)} > 0 \Rightarrow r(z_{(j)} - \underline{v}_{(j)}) = z_{(j)} - \underline{v}_{(j)}$;
   - Consequentemente $v_{(j)} = z_{(j)} - \underline{v}_{(j)} + \underline{v}_{(j)} = z_{(j)}$.
3. Saturação no limite superior, $z_{(j)} > \overline{v}_{(j)}$:
   - $z_{(j)} - \overline{v}_{(j)} > 0 \Rightarrow r(z_{(j)} - \overline{v}_{(j)}) = z_{(j)} - \overline{v}_{(j)}$;
   - $z_{(j)} - \underline{v}_{(j)} > 0 \Rightarrow r(z_{(j)} - \underline{v}_{(j)}) = z_{(j)} - \underline{v}_{(j)}$;
   - Consequentemente $v_{(j)} = -(z_{(j)} - \overline{v}_{(j)}) + z_{(j)} - \underline{v}_{(j)} + \underline{v}_{(j)} = \overline{v}_{(j)}$.

Desta forma, a função de saturação aplicada a um vetor $\text{sat}(z)$ pode ser escrita na forma (10) com

$$\begin{aligned} &N_1 = 0_{n_z},\ N_2 = \begin{bmatrix} -I_{n_z} & I_{n_z} \end{bmatrix},\ e_0 = \underline{v}, \\ &N_3 = \begin{bmatrix} I_{n_z} \\ I_{n_z} \end{bmatrix},\ N_4 = 0_{n_z},\ e_1 = \begin{bmatrix} -\overline{v} \\ -\underline{v} \end{bmatrix}. \end{aligned} \tag{32}$$

A Figura 8 apresenta uma implementação da função de saturação para $n_z = 1$, utilizando uma RN ReLU. Caso $n_z > 1$, basta replicar esta estrutura de forma paralela.

Figura 8 – Saturação como uma RN ReLU para $n_z = 1$.

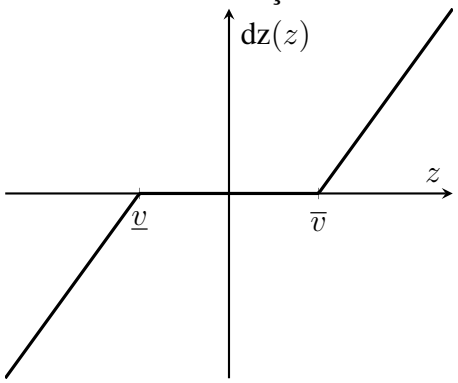


Fonte: Autor

2.6.2 Zona Morta como RN ReLU

Considerando dz : $\mathbb{R} \to \mathbb{R}$ uma função escalar de zona morta, apresentada na Figura 9.

Figura 9 – Gráfico da função de zona morta.



Fonte: Autor

O sinal limitado $v \in \mathbb{R}^{n_z}$ pela zona morta é dada por

$$v_{(j)} = \text{dz}(z_{(j)}) := \begin{cases} z_{(j)} - \underline{v}_{(j)} & \text{se } z_{(j)} < \underline{v}_{(j)}, \\ 0 & \text{se } \underline{v}_{(j)} \leq z_{(j)} \leq \overline{v}_{(j)}, \\ z_{(j)} - \overline{v}_{(j)} & \text{se } z_{(j)} > \overline{v}_{(j)}, \end{cases} \tag{33}$$

para $j = 1, \ldots, n_z$, em que $\underline{v} \preccurlyeq \overline{v} \in \mathbb{R}^{n_z}$ são, respectivamente, os vetores que concatenam os limites inferior e superior da zona morta de cada uma das n_z entradas.

Para escrever (33), observa-se que a função de zona morta escalar pode ser escrita como uma soma de funções ReLU

$$\text{dz}(z_{(j)}) = r(z_{(j)} - \overline{v}_{(j)}) - r(\underline{v}_{(j)} - z_{(j)}) \tag{34}$$

Essa equivalência pode ser verificada considerando os três casos possíveis:

1. Zona morta no limite inferior, $z_{(j)} < \underline{v}_{(j)}$:

- $z_{(j)} - \overline{v}_{(j)} < 0 \Rightarrow r(z_{(j)} - \overline{v}_{(j)}) = 0$;
- $\underline{v}_{(j)} - z_{(j)} > 0 \Rightarrow r(\underline{v}_{(j)} - z_{(j)}) = \underline{v}_{(j)} - z_{(j)}$;
- Consequentemente $v_{(j)} = -(\underline{v}_{(j)} - z_{(j)}) = z_{(j)} - \underline{v}_{(j)}$.

2. Zona morta na região morta, $\underline{v}_{(j)} \leq z_{(j)} \leq \overline{v}_{(j)}$:

- $z_{(j)} - \overline{v}_{(j)} < 0 \Rightarrow r(z_{(j)} - \overline{v}_{(j)}) = 0$;
- $\underline{v}_{(j)} - z_{(j)} < 0 \Rightarrow r(\underline{v}_{(j)} - z_{(j)}) = 0$;
- Consequentemente $v_{(j)} = 0$.

3. Zona morta no limite superior, $z_{(j)} > \overline{v}_{(j)}$:

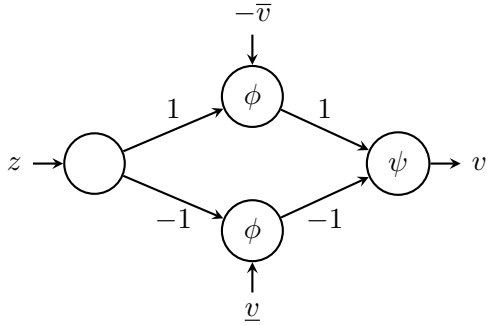
- $z_{(j)} - \overline{v}_{(j)} > 0 \Rightarrow r(z_{(j)} - \overline{v}_{(j)}) = z_{(j)} - \overline{v}_{(j)}$;
- $\underline{v}_{(j)} - z_{(j)} < 0 \Rightarrow r(\underline{v}_{(j)} - z_{(j)}) = 0$;
- Consequentemente $v_{(j)} = z_{(j)} - \overline{v}_{(j)}$.

Desta forma, a função de zona morta aplicada a um vetor dz(z) pode ser escrita na forma (10) com

$$\begin{aligned} &N_1 = 0_{n_z},\ N_2 = \begin{bmatrix} I_{n_z} & -I_{n_z} \end{bmatrix},\ e_0 = 0_{n_z,1}, \\ &N_3 = \begin{bmatrix} I_{n_z} \\ -I_{n_z} \end{bmatrix},\ N_4 = 0_{n_z},\ e_1 = \begin{bmatrix} -\overline{v} \\ \underline{v} \end{bmatrix}. \end{aligned} \tag{35}$$

A Figura 10 apresenta uma implementação da função de zona morta para $n_z = 1$, utilizando uma RN ReLU. Caso $n_z > 1$, basta replicar esta estrutura de forma paralela.

Figura 10 – Zona morta como uma RN ReLU para $n_z = 1$.



Fonte: Autor

2.6.3 MPC como RN ReLU

Técnicas MPC (do inglês *Model Predictive Control*) consistem em utilizar o modelo dinâmico do processo para controlá-lo satisfazendo restrições de estado e/ou de entrada de controle. Dado um sistema linear a tempo discreto representado por

$$x(k+1) = Ax(k) + Bu(k), \tag{36}$$

onde $x(k)$ e $x(k+1) \in \mathbb{R}^{n_x}$ são, respectivamente, o estado atual e o sucessor em uma dada amostra $k \geq 0 \in \mathbb{N}$, $u(k) \in \mathbb{R}^{n_u}$ é a entrada de controle nesta mesma amostra k e as matrizes $A \in \mathbb{R}^{n_x \times n_x}$ e $B \in \mathbb{R}^{n_x \times n_u}$ definem o sistema. O MPC, neste caso, pode ser definido a partir de um problema de Programação Quadrática (QP, do inglês *Quadratic Programming*), como, por exemplo (HERCEG *et al.*, 2013):

$$\begin{aligned} &\min_{U} \sum_{i=0}^{N_h-1} x(k+i|k)^\top Q_x x(k+i|k) + u(k+i|k)^\top Q_u u(k+i|k), \\ &\text{sujeito a (36), } \; GU \preccurlyeq R + Ex(k|k), \end{aligned} \tag{37}$$

onde $x(k+i|k)$ representa a predição do estado para o instante $k+i$ baseada na informação disponível no instante k, dada por $x(k|k) = x(k)$, e $u(k+i|k)$ representa a sequência de ações de controle futuras a ser otimizada. As matrizes $0 \leq Q_x \in \mathbb{S}^{n_x}$ e $0 < Q_u \in \mathbb{S}^{n_u}$ são os pesos de ponderação da função custo e N_h é o tamanho do horizonte de predição. O vetor $U^\top = \begin{bmatrix} u(k|k)^\top & \dots & u(k+N_h-1|k)^\top \end{bmatrix} \in \mathbb{R}^{N_h n_u}$ representa a sequência de entradas de controle e as matrizes $G \in \mathbb{R}^{n_r \times N_h n_u}$, $E \in \mathbb{R}^{n_r \times n_x}$ e $R \in \mathbb{R}^{n_r}$ representam as n_r restrições de (37), que podem representar limites nas entradas e/ou estados. Uma vez que a sequência de controle ótima U^* é computada, seu primeiro elemento, ou seja, $u(k|k)^*$, é aplicado no processo e uma política de horizonte deslizante é empregada (CAMACHO; BORDONS, 2007; MACIEJOWSKI, 2000).

O problema QP em (37) pode ser escrito de forma compacta como

$$\begin{aligned} &\min_{U} \left\{ \frac{1}{2} U^\top H_m U + x(k|k)^\top F_m^\top U \right\} + x(k|k)^\top Y_m x(k|k), \\ &\text{sujeito a: } \; GU \preccurlyeq R + Ex(k|k). \end{aligned} \tag{38}$$

As matrizes $H_m \in \mathbb{S}^{N_h n_u}$, $F_m \in \mathbb{R}^{N_h n_u \times n_x}$ e $Y_m \in \mathbb{S}^{n_x}$ são calculadas a partir de Q_x, Q_u e da dinâmica do sistema, conforme a formulação clássica para problemas de controle preditivo com restrições (CAMACHO; BORDONS, 2007; MACIEJOWSKI, 2000):

$$\begin{aligned} H_m &= 2(\tilde{B}^\top (I_{N_h} \otimes Q_x) \tilde{B} + (I_{N_h} \otimes Q_u)), \\ F_m^\top &= 2\tilde{A}^\top (I_{N_h} \otimes Q_x) \tilde{B}, \\ Y_m &= \tilde{A}^\top (I_{N_h} \otimes Q_x) \tilde{A}, \end{aligned}$$

onde

$$\tilde{A} := \begin{bmatrix} I_{n_x} \\ A \\ A^2 \\ \vdots \\ A^{N_h-2} \\ A^{N_h-1} \end{bmatrix} \quad \text{e} \quad \tilde{B} := \begin{bmatrix} 0_{n_x,n_u} & 0_{n_x,n_u} & 0_{n_x,n_u} & \dots & 0_{n_x,n_u} & 0_{n_x,n_u} \\ B & 0_{n_x,n_u} & 0_{n_x,n_u} & \dots & 0_{n_x,n_u} & 0_{n_x,n_u} \\ AB & B & 0_{n_x,n_u} & \dots & 0_{n_x,n_u} & 0_{n_x,n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{N_h-3}B & A^{N_h-4}B & A^{N_h-5}B & \dots & 0_{n_x,n_u} & 0_{n_x,n_u} \\ A^{N_h-2}B & A^{N_h-3}B & A^{N_h-4}B & \dots & B & 0_{n_x,n_u} \end{bmatrix}.$$

O tamanho do problema de otimização depende do horizonte de predição N_h, da dimensão do sistema e das restrições. Dependendo do tamanho do problema de otimização, pode ser difícil resolvê-lo no tempo entre amostragens (VALMORBIDA; HOVD, 2023). Uma alternativa à solução online de problemas de otimização quadráticos no MPC é o chamado MPC explícito (BEMPORAD *et al.*, 2002). Neste caso, é sabido que a lei de controle resultante do MPC oriundo de (38) corresponde a uma função PWA do estado.

Na formulação de (BEMPORAD *et al.*, 2002) é utilizada uma representação explícita da função PWA correspondente ao MPC. Tal representação mapeia o espaço de estados em poliedros disjuntos, onde para cada região a saída é definida por uma expressão afim distinta (JOHANSSON, 2003) . Assim a lei de controle ótima pode ser calculada offline como uma função PWA. Desta forma, durante a operação, o controlador localiza a região correspondente ao estado atual e aplica a lei de controle afim associada (HERCEG *et al.*, 2013).

Uma alternativa à implementação do MPC explícito usando a representação explícita de funções PWA é o uso de RNs para calcular a lei de controle do mesmo. Como a lei de controle MPC é uma função PWA do estado no início do horizonte de otimização, a solução do MPC pode ser representada exatamente por uma RN ReLU implícita. Assim o MPC pode ser representado por (10) com as seguintes matrizes (VALMORBIDA; HOVD, 2023):

$$\begin{aligned} &N_1 = -K_m H_m^{-1} F_m,\ N_2 = -K_m H_m^{-1} G^\top,\ e_0 = 0_{n_u,1},\ K_m = \begin{bmatrix} I_{n_u} & 0_{n_u,n_u(N_h-1)} \end{bmatrix}, \\ &N_3 = -(E + G H_m^{-1} F_m),\ N_4 = I_{n_r} - G H_m^{-1} G^\top, \text{ e } e_1 = -R. \end{aligned} \tag{39}$$

Observação 6. *Como o problema de otimização* (37) *pode ter restrições de estado, uma solução de* (10) *com matrizes como em* (39) *pode não existir para todo $x \in \mathbb{R}^{n_x}$. Nesse caso, o conjunto de bem-postura (well-posedness) do MPC não é, em geral, todo o espaço de estados ($\mathcal{W} \neq \mathbb{R}^{n_x}$).*

Entretanto, uma RN implícita também apresenta desafios de implementação, pois tipicamente requer a solução de um problema de ponto fixo ou de otimização a cada passo (BAI; KOLTER; KOLTUN, 2019).

Uma opção alternativa a RN implícita, de forma a evitar a necessidade de resolver um problema de otimização a cada instante de amostragem, é aproximar a lei de controle MPC ótima através de uma RN *feedforward* com possivelmente menos neurônios. No entanto, embora seja possível garantir um limite superior para o erro entre a lei de controle do MPC e a aproximação por RN (ROSS *et al.*, 2022), isso não garante a estabilidade da origem do sistema em MF. Outro problema que surge ao emular o comportamento do MPC com uma RN reduzida é que as restrições de entrada e de estado impostas pelo MPC não podem ser garantidas *a priori*.

Para resolver esses problemas provenientes da aproximação do MPC por uma RN, este trabalho propõe

- Utilizar os teoremas que serão apresentados nos capítulos 4 e 5 para verificar, respectivamente, a estabilidade global e a estabilidade local da origem do sistema;
- Utilizar uma função de saturação aplicada a saída da RN para garantir as restrições de entrada impostas pelo MPC. Esta função pode ser facilmente expressa com uma RN, conforme exposto na Subseção 2.6.1, e então integrada a formulação da RN através da conexão em série exposta na Seção 2.5;
- Utilizar restrições adicionais na verificação da estabilidade local do sistema para garantir as restrições de estado impostas pelo MPC. Estas restrições serão expostas na Seção 5.2.

2.7 Conclusões

Neste capítulo, apresentou-se a fundamentação teórica sobre RNs e estabeleceu-se o foco do trabalho em RNs que utilizam a função de ativação ReLU. A escolha dessa classe de redes dá-se pelo fato de ser uma das funções de ativação mais utilizadas atualmente e permitir sua representação como uma função PWA, o que possibilita uma análise matemática baseada nos desenvolvimentos de (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2023) e (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025).

O ponto central deste capítulo foi apresentar a representação implícita baseada em funções ReLU e como diferentes arquiteturas de RN, como as redes *feedforward*, residuais e implícitas, podem ser representadas neste formato. Esta representação é o foco principal deste trabalho e, a partir dela, nos capítulos seguintes, serão derivadas condições de estabilidade global e local para sistemas dinâmicos em tempo discreto cujo campo vetorial pode ser descrito por RNs ReLU.

Além disso, neste capítulo, foi mostrado como representar diferentes composições de funções na representação implícita baseada em funções ReLU. Tal procedimento possibilita a formulação sistemática das conexões entre RNs, bem como entre RNs e outras classes de funções e sistemas dinâmicos.

Também foi apresentado como descrever outras funções como RNs ReLU, tais como as funções de saturação e zona morta. Em especial, ao discutir como representar o MPC como RN ReLU, destacou-se a possibilidade de representa-lo de forma exata utilizando uma RN implícita ou de aproximar seu comportamento por uma RN *feedforward*, visando a redução do esforço computacional. O capítulo encerra elencando problemas decorrentes dessa emulação e introduzindo como este trabalho propõe resolvê-los.

3 FORMULAÇÃO DO PROBLEMA E PRELIMINARES

Neste capítulo, são formulados os problemas de estabilidade global e local a serem investigados e, em seguida, são apresentados os desenvolvimentos preliminares necessários para suas respectivas soluções. Na Seção 3.1, são formalmente estabelecidos os problemas a serem tratados. Na Seção 3.2, são apresentadas as propriedades da função ReLU, as quais servirão de base para a formulação das condições de estabilidade. Na Seção 3.3, é introduzida a função de Lyapunov candidata a ser utilizada, uma função Quadrática por Partes (PWQ). Devido a essa característica, na Seção 3.4, são elencadas proposições para verificar a positividade, local e global, deste tipo de função.

3.1 Formulação do Problema

Considere um sistema linear em tempo discreto

$$x^+ = Ax + Bu, \tag{40}$$

onde x e $x^+ \in \mathbb{R}^{n_x}$ são, respectivamente, o estado atual ($x(k)$) e sucessor ($x(k+1)$) do sistema, $u \in \mathbb{R}^{n_u}$ é a entrada de controle e as matrizes $A \in \mathbb{R}^{n_x \times n_x}$ e $B \in \mathbb{R}^{n_x \times n_u}$ definem o sistema em malha aberta (planta). Este sistema é controlado por uma RN ReLU f^a_{ReLU}, de forma que

$$u = f^a_{\text{ReLU}}(x). \tag{41}$$

Assim, considerando a descrição da RN como em (10), o sistema em MF é representado pelo seguinte conjunto de equações:

$$x^+ = N_1 x + N_2 \phi(q) + e_0 \tag{42a}$$

$$q = N_3 x + N_4 \phi(q) + e_1, \tag{42b}$$

com

$$\begin{aligned} &N_1 = (A + BN_{1a}),\ N_2 = BN_{2a},\ e_0 = Be_{0a}, \\ &N_3 = N_{3a},\ N_4 = N_{4a},\ e_1 = e_{1a},\ q = q_a. \end{aligned} \tag{43}$$

Apesar de inicialmente ser motivado como o sistema resultante do controle de um sistema linear controlado por uma RN ReLU, o sistema (42) pode representar qualquer

sistema dinâmico em tempo discreto cujo campo vetorial é dado por uma RN ReLU, como discutido posteriormente no Capítulo 6.

Como (42) é não linear, esse sistema pode conter múltiplos pontos de equilíbrio, os quais são definidos como os valores de x_e que satisfazem

$$x_e = N_1 x_e + N_2 \phi(q_e) + e_0 \tag{44a}$$

$$q_e = N_3 x_e + N_4 \phi(q_e) + e_1. \tag{44b}$$

Observação 7. *Devido à presença da função ReLU, a determinação dos pontos de equilíbrio x_e é não trivial. Ela pode ser realizada levando-se em conta as 2^{n_q} possibilidades da função ϕ, isto é, o fato de uma determinada componente de $\phi(q)$ ser dada por $\phi_i(q) = \alpha_i q_i$, com $\alpha_i \in \{0, 1\}$. Neste caso, uma dada combinação de α_i está associada a uma partição do sistema PWA (CABRAL, 2024). Para cada partição, a equação* (44) *é resolvida e o x_e obtido é um ponto de equilíbrio se realmente pertencer a essa partição.*

Considerando agora que x_e é um ponto de equilíbrio e definindo uma variável auxiliar $\bar{x} := x - x_e$, segue-se que

$$\begin{aligned} \bar{x}^+ &= N_1 \bar{x} + N_2 \phi(q) + (N_1 - I_{n_x}) x_e + e_0 = N_1 \bar{x} + N_2 \phi(q) + \bar{e}_0 \\ q &= N_3 \bar{x} + N_4 \phi(q) + N_3 x_e + e_1 = N_3 \bar{x} + N_4 \phi(q) + \bar{e}_1, \end{aligned} \tag{45}$$

o que significa que o sistema com estado transladado $\bar{x}$ (45) tem a mesma estrutura que o sistema original dado por (42). Além disso, se x_e é um ponto de equilíbrio de (42), então segue que a origem é um ponto de equilíbrio de (45). Assim, pode-se assumir que a origem é o ponto de equilíbrio de interesse do sistema (42) sem perda de generalidade.

Desta forma, considera-se a seguinte suposição:

Hipótese 1. *A origem do sistema* (42) *é um ponto de equilíbrio.*

Levando em conta a Hipótese 1, os problemas a serem tratados são enunciados formalmente como:

Problema 1. *Certificar a estabilidade exponencial global da origem do sistema* (42)*.*

Problema 2. *Certificar a estabilidade exponencial local da origem do sistema* (42)*, juntamente com o cálculo de uma estimativa positivamente invariante da Região de Atração da Origem (RAO).*

3.2 Propriedades da Função ReLU

A fim de deduzir condições para resolver os problemas 1 e 2, é importante explorar as propriedades da função ReLU (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2023; CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025).

Primeiramente, note que as seguintes desigualdades e a identidade descrevem precisamente a função rampa (ReLU) escalar $r : \mathbb{R} \to \mathbb{R}$:

$$r(q) \geq 0, \tag{46a}$$

$$r(-q) \geq 0, \tag{46b}$$

$$r(q)(r(q) - q) = 0, \tag{46c}$$

para qualquer escalar $q \in \mathbb{R}$. Note que (46a) e (46b) descrevem a não negatividade da função rampa escalar. Além disso, observando que $r(q) - q = r(-q)$, (46c) é uma restrição de complementaridade, afirmando que não existe q tal que $r(-q) > 0$ e $r(q) > 0$.

Note que, diferentemente de restrições de setor que são válidas para uma variedade de funções diferentes, o conjunto das restrições (46) é válido apenas para a função ReLU.

Os seguintes lemas exploram (46) por meio de expressões quadráticas generalizadas, considerando a função rampa (ReLU) vetorial $\phi : \mathbb{R}^{n_q} \to \mathbb{R}^{n_q}$. Esses lemas serão utilizados para levar em conta a presença de funções ReLU na dinâmica do sistema e na função de Lyapunov, permitindo converter informações do campo vetorial em condições de estabilidade.

Lema 1. *Para qualquer matriz $T \in \mathbb{D}^{n_q}$, a seguinte expressão é verificada*

$$s_1(T, q) := \begin{bmatrix} \phi(q) \\ \phi(q) - q \end{bmatrix}^\top \begin{bmatrix} 0_{n_q} & T \\ T & 0_{n_q} \end{bmatrix} \begin{bmatrix} \phi(q) \\ \phi(q) - q \end{bmatrix} = 0 \quad \forall q \in \mathbb{R}^{n_q}. \tag{47}$$

Prova. Usando (46c), se obtém

$$s_1(T, q) = 2 \sum_{i=1}^{n_q} T_{(i,i)} r(q_{(i)})(r(q_{(i)}) - q) = 0 \quad \forall q \in \mathbb{R}^{n_q}.$$

□

Lema 2. *Para qualquer matriz $M \in \mathbb{P}^{1+2n_q}$, a seguinte expressão é verificada*

$$s_2(M, q) := \begin{bmatrix} 1 \\ \phi(q) \\ \phi(q) - q \end{bmatrix}^\top M \begin{bmatrix} 1 \\ \phi(q) \\ \phi(q) - q \end{bmatrix} \geq 0 \quad \forall q \in \mathbb{R}^{n_q}. \tag{48}$$

Prova. Usando (46a) e (46b), observa-se que tanto $\phi(q)$ quanto $\phi(q) - q$ são não negativos elemento a elemento para qualquer $q \in \mathbb{R}^{n_q}$. Como M possui apenas elementos não negativos, então $s_2(M, q) \geq 0$ para todo $q \in \mathbb{R}^{n_q}$. □

Observação 8. *Em (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025), o Lema 2 é enunciado considerando a matriz M como copositiva, ou seja $x^\top M x \geq 0$ para todo vetor x que tenha apenas elementos não-negativos. Neste trabalho, por simplicidade, consideramos M elemento a elemento não negativa, o que constitui uma condição particular e apenas suficiente para a copositividade de uma matriz.*

É importante enfatizar que as propriedades enunciadas nos lemas acima provém diretamente da definição da função ReLU vetorial e aplicam-se única e exclusivamente a este tipo de função.

3.3 Função de Lyapunov Candidata

Para certificar a estabilidade do sistema (42), será utilizada a teoria de Lyapunov, em particular, será usada uma função de Lyapunov Quadrática por Partes, ou do inglês *piecewise quadratic* (PWQ), como função candidata. Este tipo de função pode ser definida de forma compacta a partir da função ReLU vetorial como $V : \mathbb{R}^{n_x} \to \mathbb{R}$ dada por

$$V(x) := \begin{bmatrix} x \\ \phi(q) \end{bmatrix}^\top P \begin{bmatrix} x \\ \phi(q) \end{bmatrix}. \tag{49}$$

com $P \in \mathbb{S}^{n_x+n_q}$ e q dado por (42b), $q = N_3 x + N_4 \phi(q) + e_1$. A função de Lyapunov $V(x)$ é considerada PWQ pois, devido à natureza afim da função ϕ, esta função tem comportamentos distintos para cada partição do espaço de estados.

A partir de uma função (49) e fazendo uso das propriedades da função ReLU vetorial descritas em 3.2, serão derivadas condições de estabilidade em termos de Desigualdades Matriciais Lineares (LMIs, do inglês *Linear Matrix Inequalities*). Com esse objetivo, é conveniente expressar $V(x)$ em função do vetor

$$\chi^\top(x) := \begin{bmatrix} 1 & x^\top & \phi(q)^\top & (\phi(q) - q)^\top \end{bmatrix}.$$

Para isto, pode-se re-escrever $V(x)$ como

$$V(x) = \chi^\top(x) \Pi^\top P \Pi \chi(x) \tag{50}$$

com

$$\Pi := \begin{bmatrix} 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} & 0_{n_q} \end{bmatrix}. \tag{51}$$

Para derivar as condições de estabilidade, também é preciso escrever $\Delta V(x) = V(x^+) - V(x)$ como uma função PWQ na forma compacta. Para isso, note que a partir de (49), $V(x^+)$, isto é, o valor da função de Lyapunov na amostra seguinte, é dado por

$$V(x^+) := \begin{bmatrix} x^+ \\ \phi(q^+) \end{bmatrix}^\top P \begin{bmatrix} x^+ \\ \phi(q^+) \end{bmatrix}, \tag{52}$$

onde x^+ e q^+ são, respectivamente, o estado sucessor da amostra atual e a entrada da função ReLU para este estado sucessor. A partir de (42) tem-se que

$$\begin{aligned} q^+ &= q(x^+) = q(N_1 x^+ + N_2 \phi(q) + e_0) \\ &= N_3 N_1 x + N_3 N_2 \phi(q) + N_4 \phi(q^+) + N_3 e_0 + e_1. \end{aligned}$$

Então, definindo o vetor estendido

$$\tilde{q} := \begin{bmatrix} q^\top & q^{+\top} \end{bmatrix}^\top = \tilde{N}_3 x + \tilde{N}_4 \tilde{q} + \tilde{e}_1, \tag{53}$$

com

$$\tilde{N}_3 := \begin{bmatrix} N_3 \\ N_3 N_1 \end{bmatrix}, \; \tilde{N}_4 := \begin{bmatrix} N_4 & 0_{n_q} \\ N_3 N_2 & N_4 \end{bmatrix}, \; \tilde{e}_1 := \begin{bmatrix} e_1 \\ N_3 e_0 + e_1 \end{bmatrix}. \tag{54}$$

É conveniente escrever $\Delta V(x)$ em função do vetor

$$\tilde{\chi}^\top(x) := \begin{bmatrix} 1 & x^\top & \phi(\tilde{q})^\top & (\phi(\tilde{q}) - \tilde{q})^\top \end{bmatrix}.$$

Para tanto, reescrevem-se $V(x)$ e $V(x^+)$ como

$$V(x) = \tilde{\chi}^\top(x) \tilde{\Pi}^\top P \tilde{\Pi} \tilde{\chi}(x) \tag{55a}$$

$$V(x^+) = \tilde{\chi}^\top(x) \tilde{\Pi}_+^\top P \tilde{\Pi}_+ \tilde{\chi}(x) \tag{55b}$$

com

$$\tilde{\Pi} := \left[\begin{array}{c|c|cc|cc} 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} & 0_{n_q} & 0_{n_q} & 0_{n_q} \end{array}\right], \tag{56}$$

$$\tilde{\Pi}_+ := \left[\begin{array}{c|c|cc|cc} e_0 & N_1 & N_2 & 0_{n_x,n_q} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & 0_{n_q} & I_{n_q} & 0_{n_q} & 0_{n_q} \end{array}\right]. \tag{57}$$

Desta forma, os problemas 1 e 2 se resumem a provar que

- $V(x)$ é definida positiva:

$$V(x) > 0, \; \forall x \neq 0 \tag{58}$$

- $\Delta V(x)$ é definida negativa:

$$\Delta V(x) < 0 \tag{59}$$

  - Para todo $x \in \mathbb{R}^{n_x}$ no caso do Problema 1 (estabilidade global);
  - Para todo $x \in \mathcal{D} \subset \mathbb{R}^{n_x}$ no caso do Problema 2 (estabilidade local).

3.4 Positividade de Funções PWQ

Tendo em vista a função de Lyapunov PWQ apresentada na seção anterior, nesta seção são enunciadas proposições para certificar a positividade global e local de funções PWQ genéricas a partir das propriedades da função ReLU discutidas na Seção 3.2. Essas proposições serão utilizadas nos próximos capítulos com o objetivo de verificar as condições (58) e (59).

Primeiramente, é apresentada a proposição referente à positividade global de uma função PWQ (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2023).

Proposição 1. *Dada uma função PWQ* $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$

$$h(x) = \chi(x)^\top H \chi(x)$$

com

$$\chi^\top(x) := \begin{bmatrix} 1 & x^\top & \phi(q)^\top & (\phi(q) - q)^\top \end{bmatrix},$$

se existirem matrizes $T \in \mathbb{D}^{n_q}$ *e* $M \in \mathbb{P}^{1+2n_q}$ *tais que a seguinte desigualdade seja satisfeita*

$$h(x) + s_1(T, q) - s_2(M, q) \geq 0, \tag{60}$$

onde $s_1(T, q)$ *e* $s_2(M, q)$ *são definidos, respectivamente, em* (47) *e* (48)*, então* $h(x) \geq 0 \quad \forall x \in \mathbb{R}^{n_x}$.

Prova. Utilizando os lemas 1 e 2, temos que $s_1(T, q) = 0$ e $s_2(M, q) \geq 0$ para todo $q \in \mathbb{R}^{n_q}$, com isso, pode-se reescrever (60) como

$$h(x) \geq s_2(M, q) \geq 0 \quad \forall x \in \mathbb{R}^{n_x}.$$

□

Antes de enunciar a proposição referente à positividade local de funções PWQs, é preciso apresentar o seguinte lema proposto em (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025). Este lema é uma versão local do Lema 2, de forma a garantir a não negatividade da expressão $s_2(M, q)$ não para todo x, mas sim localmente em uma região $\mathcal{D}$. Isso é realizado por meio de uma parametrização da matriz M em função de x. A partir deste lema, será formulada uma proposição para certificar a positividade local de funções PWQ genéricas.

Lema 3. *Seja um conjunto* $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ *e considere uma função matricial* $M : \mathbb{R}^{n_x} \to \mathbb{S}^{1+2n_q}$ *tal que* $M(x)$ *é não negativa para todo* $x \in \mathcal{D}$*; então a seguinte expressão é verdadeira*

$$s_2(M(x), q) := \begin{bmatrix} 1 \\ \phi(q) \\ \phi(q) - q \end{bmatrix}^\top M(x) \begin{bmatrix} 1 \\ \phi(q) \\ \phi(q) - q \end{bmatrix} \geq 0 \quad \forall x \in \mathcal{D}. \tag{61}$$

Neste trabalho, é considerada a mesma parametrização da função matricial M proposta em (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025), de modo a satisfazer a não negatividade local de M, isto é, $M(x) \in \mathbb{P}^{1+2n_q}, \forall x \in \mathcal{D}$, como segue

$$M(x) := \begin{bmatrix} f_a(x) & f_b^\top(x) \\ f_b(x) & S \end{bmatrix}, \tag{62a}$$

com $S \in \mathbb{P}^{2n_q}$ e as funções $f_a : \mathbb{R}^{n_x} \to \mathbb{R}$ e $f_b : \mathbb{R}^{n_x} \to \mathbb{R}^{2n_q}$ dadas por

$$f_a(x) := \chi^\top(x) F_a \chi(x), \tag{62b}$$

$$f_b(x) := F_b \chi(x), \tag{62c}$$

com $F_a \in \mathbb{S}^{n_\chi}$, $F_b \in \mathbb{R}^{2n_q \times n_\chi}$ e $n_\chi = 1 + n_x + 2n_q$.

Dessa forma, desde que $f_a(x) \geq 0$ e $f_b(x) \geq 0$ para todo $x \in \mathcal{D}$, então $M(x) \in \mathbb{P}^{1+2n_q}, \forall x \in \mathcal{D}$. Portanto, no Lema 3, como a função matricial M é afim em S, F_a e F_b, $s_2(M(x), y)$ pode ser dada pela seguinte expressão PWQ

$$\chi^\top \Upsilon M(x) \Upsilon^\top \chi = \chi^\top \bar{M}(S, F_a, F_b)\chi \tag{63}$$

com

$$\Upsilon^T := \begin{bmatrix} 1 & 0_{1,n_x} & 0_{1,2n_q} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & I_{2n_q} \end{bmatrix} \text{ e}$$

$$\bar{M}(S, F_a, F_b) := F_a + \text{He}\left(\begin{bmatrix} 0_{1+n_x,n_\chi} \\ F_b \end{bmatrix}\right) + \begin{bmatrix} 0_{1+n_x} & 0_{1+n_x,2n_q} \\ 0_{2n_q,1+n_x} & S \end{bmatrix}.$$

Observe que tanto f_a quanto cada elemento de f_b, isto é,

$$f_{b(i)}(x) = \frac{1}{2}\chi^\top(x)\left(\text{He}\left(\begin{bmatrix} F_{b(i,:)} \\ 0_{(n_x+2n_q),n_\chi} \end{bmatrix}\right)\right)\chi(x), \tag{64}$$

para $i = 1, \ldots, 2n_q$, também são funções PWQ em x.

Por fim, utilizando o Lema 3, é enunciada a seguinte proposição para certificar a positividade local de uma função PWQ.

Proposição 2. *Dada uma função PWQ* $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$

$$h(x) = \chi(x)^\top H \chi(x)$$

se existir a matriz $T \in \mathbb{D}^{n_q}$ *e a função matricial* $M(x)$*, tal que* $M(x) \in \mathbb{P}^{1+2n_q}, \forall x \in \mathcal{D}$ *e a seguinte desigualdade seja satisfeita*

$$h(x) + s_1(T, q) - s_2(M(x), q) \geq 0, \forall x \in \mathcal{D} \tag{65}$$

onde $s_1(T, q)$ *e* $s_2(M(x), q)$ *são definidos, respectivamente, em* (47) *e* (61)*, então* $h(x) \geq 0 \quad \forall x \in \mathcal{D}$.

3.5 Conclusões

Este capítulo formaliza o problema central abordado neste trabalho. Motivada pela análise de estabilidade, tanto global quanto local, da origem em sistemas lineares controlados por RNs ReLU, a abordagem proposta estende-se para qualquer sistema dinâmico em tempo discreto cujo campo vetorial é descrito por RNs ReLU. Para a análise local, além da verificação da estabilidade, definiu-se como objetivo a estimativa da RAO.

Para viabilizar tal análise, revisaram-se propriedades fundamentais da função de ativação ReLU, as quais servirão como base analítica para a subsequente derivação das condições de estabilidade.

Adicionalmente, as seções finais estabeleceram resultados preliminares necessários para o desenvolvimento dos métodos propostos. Foi introduzida a função de Lyapunov candidata a ser utilizada, uma função de Lyapunov PWQ, e, a partir das propriedades da função ReLU apresentadas, foram estabelecidas proposições para certificar a positividade (global e local) de funções PWQ.

Nos próximos capítulos, essas proposições serão aplicadas para verificar que a função de Lyapunov PWQ é definida positiva e sua derivada é definida negativa. Esse arcabouço permitirá a derivação das condições de estabilidade e a determinação da estimativa da RAO.

4 ESTABILIDADE GLOBAL

Neste capítulo, são apresentadas condições para garantir a estabilidade global da origem de sistemas cujo campo vetorial é dado por uma RN ReLU. Utilizando o arcabouço matemático estabelecido na seção anterior, propõem-se condições de estabilidade no formato de LMIs, as quais são apresentadas na Seção 4.1. Para ilustrar o método proposto, na Seção 4.2, são apresentados exemplos numéricos nos quais as condições são aplicadas.

4.1 Condições de Estabilidade

Nesta seção, com o propósito de apresentar uma solução para o Problema 1, são estabelecidas condições na forma de LMIs que certificam a estabilidade exponencial da origem do sistema

$$x^+ = N_1 x + N_2 \phi(q) + e_0 \tag{66a}$$

$$q = N_3 x + N_4 \phi(q) + e_1. \tag{66b}$$

Para derivar tais condições de estabilidade, empregam-se os lemas e proposições elencados nas seções 3.2 e 3.4.

Observação 9. *Caso a matriz N_4 não tenha estrutura estritamente triangular inferior e consequentemente* (66b) *seja mal posta para algum $x \in \mathbb{R}^{n_x}$, a estabilidade global do sistema* (66) *não poderá ser garantida. Nesse cenário, deve-se verificar a estabilidade local dentro do conjunto $\mathcal{W} \subset \mathbb{R}^{n_x}$ onde* (66b) *é bem posta.*

A ideia central é explorar as propriedades da função ReLU vetorial de forma a garantir que uma função candidata de Lyapunov PWQ $V(x)$, apresentada na Seção 3.3, seja definida positiva e tenha um decaimento exponencial ao longo das trajetórias do sistema. Dessa forma, utilizando a Proposição 1, estabelecem-se condições de estabilidade na forma de LMIs, conforme enunciado no Teorema 1.

Teorema 1. *Considerando um sistema dinâmico representado por* (66)*, se existirem matrizes $P \in \mathbb{S}^{n_x+n_q}$, $T_1 \in \mathbb{D}^{n_q}$, $T_2 \in \mathbb{D}^{n_q}$, $\tilde{T} \in \mathbb{D}^{2n_q}$, $M_1 \in \mathbb{P}^{1+2n_q}$, $M_2 \in \mathbb{P}^{1+2n_q}$,*

$\tilde{M} \in \mathbb{P}^{1+4n_q}$, e escalares $\epsilon_1 > 0$, $\epsilon_2 > 0$ e $\eta < 1$, de forma que as seguintes desigualdades matriciais sejam satisfeitas

$$\Pi_\chi^\top (H(P,\epsilon_1) + \Psi(T_1) - \Upsilon M_1 \Upsilon^\top)\Pi_\chi > 0, \tag{67a}$$
$$\Pi_\chi^\top (-H(P,\epsilon_2) + \Psi(T_2) - \Upsilon M_2 \Upsilon^\top)\Pi_\chi > 0, \tag{67b}$$
$$\tilde{\Pi}_\chi^\top (\tilde{H}(P,\eta) + \tilde{\Psi}(\tilde{T}) - \tilde{\Upsilon} \tilde{M} \tilde{\Upsilon}^\top)\tilde{\Pi}_\chi > 0, \tag{67c}$$

sendo

$$H(P,\epsilon) := \Pi^\top \left(P - \epsilon \begin{bmatrix} I_{n_x} & 0_{n_x,n_q} \\ 0_{n_q,n_x} & 0_{n_q} \end{bmatrix} \right) \Pi, \; \tilde{H}(P,\eta) := (1-\eta)\tilde{\Pi}^\top P \tilde{\Pi} - \tilde{\Pi}_+^\top P \tilde{\Pi}_+,$$

$$\Psi(T) := \begin{bmatrix} 0 & 0_{1,n_x} & 0_{1,n_q} & 0_{1,n_q} \\ 0_{n_x,1} & 0_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & 0_{n_q} & T \\ 0_{n_q,1} & 0_{n_q,n_x} & T & 0_{n_q} \end{bmatrix}, \; \tilde{\Psi}(\tilde{T}) := \begin{bmatrix} 0 & 0_{1,n_x} & 0_{1,2n_q} & 0_{1,2n_q} \\ 0_{n_x,1} & 0_n & 0_{n_x,2n_q} & 0_{n_x,2n_q} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & 0_{2n_q} & \tilde{T} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & \tilde{T} & 0_{2n_q} \end{bmatrix},$$

$$\Upsilon := \begin{bmatrix} 1 & 0_{1,2n_q} \\ 0_{n_x,1} & 0_{n_x,2n_q} \\ 0_{2n_q,1} & I_{2n_q} \end{bmatrix}, \; \tilde{\Upsilon} := \begin{bmatrix} 1 & 0_{1,4n_q} \\ 0_{n_x,1} & 0_{n_x,4n_q} \\ 0_{4n_q,1} & I_{4n_q} \end{bmatrix}, \; \Pi_\chi := \begin{bmatrix} 1 & 0_{1,n_x} & 0_{1,n_q} \\ 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} \\ -e_1 & -N_3 & I_{n_q} - N_4 \end{bmatrix},$$

$$\tilde{\Pi}_\chi := \begin{bmatrix} 1 & 0_{1,n_x} & 0_{1,2n_q} \\ 0_{n_x,1} & I_{n_x} & 0_{n_x,2n_q} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & I_{2n_q} \\ -\tilde{e}_1 & -\tilde{N}_3 & I_{2n_q} - \tilde{N}_4 \end{bmatrix}, \; \tilde{\Pi} := \begin{bmatrix} 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} & 0_{n_x,2n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} & 0_{n_q} & 0_{2n_q} \end{bmatrix},$$

$$\tilde{\Pi}_+ := \begin{bmatrix} e_0 & N_1 & N_2 & 0_{n_x,n_q} & 0_{n_x,2n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & 0_{n_q} & I_{n_q} & 0_{2n_q} \end{bmatrix}, \; \Pi := \begin{bmatrix} 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} & 0_{n_q} \end{bmatrix}.$$

então a origem do sistema (66) *é globalmente exponencialmente estável.*

Prova. Considere uma função de Lyapunov PWQ $V(x)$ definida como (49) e seu valor no próximo instante de tempo $V(x^+)$ definido em (52).

Agora, note que $\chi(x) = \Pi_\chi \begin{bmatrix} 1 & x^\top & \phi^\top(q) \end{bmatrix}^\top$ devido a (66b) e suponha que as desigualdades (67a) e (67b) sejam satisfeitas. Então, ao multiplicar à direita e à esquerda o lado esquerdo de (67a) e (67b) pelos vetores $[1 \quad x^\top \quad \phi^\top(q)]^\top$ e sua transposta, respectivamente, obtém-se

$$V(x) - \epsilon_1 \|x\|_2^2 + s_1(T_1, q) - s_2(M_1, q) \geq 0, \tag{68a}$$
$$-V(x) + \epsilon_2 \|x\|_2^2 + s_1(T_2, q) - s_2(M_2, q) \geq 0. \tag{68b}$$

Assim, usando a Proposição 1 considerando respectivamente $h(x) = V(x) - \epsilon_1 x^\top x$ e $h(x) = -V(x) + \epsilon_2 x^\top x$, pode-se concluir que

$$\epsilon_1 \|x\|_2^2 \leq V(x) \leq \epsilon_2 \|x\|_2^2, \tag{69}$$

o que implica que $V(x)$ é definida positiva e radialmente ilimitada.

De forma semelhante, a partir de (53), tem-se que $\tilde{\chi}(x) = \tilde{\Pi}_{\chi}[1 \quad x^{\top} \quad \phi^{\top}(\tilde{q})]^{\top}$ com $\tilde{q} = \begin{bmatrix} q^{\top} & (q^{+})^{\top} \end{bmatrix}^{\top}$. Assumindo que (67c) é satisfeita, ao multiplicar à direita e à esquerda o lado esquerdo de (67c) pelo vetor $[1 \quad x^{\top} \quad \phi^{\top}(\tilde{q})]^{\top}$ e sua transposta, respectivamente, tem-se que

$$\eta V(x) - V(x^{+}) + s_1(\tilde{T}, \tilde{q}) - s_2(\tilde{M}, \tilde{q}) \geq 0.$$

Assim, utilizando novamente a Proposição 1, agora com $h(x) = \eta V(x) - V(x^{+})$, a expressão acima implica

$$V(x^{+}) \leq (1 - \eta)V(x), \tag{70}$$

de forma que

$$\Delta V(x) = V(x^{+}) - V(x) \leq -\eta V(x) < 0. \tag{71}$$

Assim, (69) e (70) nos permitem concluir que $\|x(k)\| \leq \kappa e^{\delta k}\|x(0)\|$ com $\kappa = (\frac{\epsilon_2}{\epsilon_1})^{\frac{1}{2}}$ e $\delta = \ln(\sqrt{1-\eta})$, para toda condição inicial $x(0) \in \mathbb{R}^{n_x}$, o que implica a estabilidade exponencial global da origem (KHALIL, 2002). □

Note que a desigualdade (67c) é uma BMI (do inglês *Bilenear Matrix Inequality*,ou Desigualdade Matricial Bilinear), pois η multiplica P. Para evitar tomar simultaneamente η e P como variáveis, fixamos um pequeno valor para η. Como o parâmetro η estabelece a relação entre $V(x)$ e $V(x^{+})$, conforme indicado em (70), ele define a convergência exponencial do sistema. Especificamente, quanto menor o valor de η, com $\eta < 1$, mais rápida é a convergência da função de Lyapunov. Entretanto, se o η escolhido for muito pequeno, pode não ser possível obter uma solução para as LMIs.

Desta forma, se o objetivo é avaliar a estabilidade exponencial da origem sem se preocupar com a taxa de decaimento, pode-se fixar η suficientemente próximo de 1 (satisfazendo $\eta < 1$). Por outro lado, caso haja interesse em estimar a taxa de convergência das trajetórias do sistema à origem, pode-se usar um método de bisseção (resolvendo as LMIs a cada iteração) para calcular o valor mínimo de η para o qual as condições em (67) são factíveis.

4.2 Exemplos Numéricos

Nesta seção, são apresentados alguns exemplos numéricos[1] para ilustrar a aplicação das condições de estabilidade global propostas.

[1]Os exemplos numéricos foram implementados em linguagem Python, utilizando as bibliotecas NumPy (HARRIS *et al.*, 2020), CvXpy (DIAMOND; BOYD, 2016) e TensorFlow (ABADI *et al.*, 2015), com o auxílio do solver MOSEK (MOSEK ApS, 2024).

4.2.1 Exemplo 1

Seja um sistema linear a tempo discreto dado por (40) com

$$A = \begin{bmatrix} 0.7326 & -0.0861 \\ 0.1722 & 0.9909 \end{bmatrix} \text{ e } B = \begin{bmatrix} 0.6090 \\ 0.0640 \end{bmatrix}. \tag{72}$$

Para controlar esse sistema, deseja-se implementar um MPC levando em conta as seguintes restrições de amplitude sobre a entrada de controle $\underline{u} \leq u \leq \overline{u}$ com $\overline{u} = 1$ e $\underline{u} = -2$.

Para evitar a solução de um problema de otimização a cada instante de amostragem, o que pode ser proibitivo em certas aplicações, será treinada uma RN reduzida para aproximar o comportamento do MPC. Para garantir as restrições de entrada desejadas, será adicionada uma função de saturação na saída da RN treinada. Para garantir que a origem do sistema em MF seja um ponto de equilíbrio, a RN treinada terá o valor do *bias* da sua última camada ajustado de forma que a saída da RN seja zero para uma entrada nula. Por fim, para certificar a estabilidade do sistema em MF controlado pela RN reduzida, as condições do Teorema 1 serão empregadas.

4.2.1.1 Projeto do MPC

O MPC projetado para o sistema é dado pelo QP (37) e inclui as restrições no sinal de controle $\overline{u} = 1$ e $\underline{u} = -2$. Desta forma, as matrizes do problema QP utilizadas para calcular o valor da entrada de controle u, são $Q_x = I_2$, $Q_u = 0.5$, $N_h = 5$, $E = 0_{10,2}$, $R^\top = \begin{bmatrix} \mathbf{1}_{1,5} & 2\mathbf{1}_{1,5} \end{bmatrix}$ e $G^\top = \begin{bmatrix} I_5 & -I_5 \end{bmatrix}$.

4.2.1.2 RN Reduzida

Foi treinada uma RN *feedforward* a partir do MPC com objetivo de aproximar o comportamento do mesmo. Escolheu-se como estrutura uma RN ReLU com 2 entradas, 1 saída e 2 camadas com 2 neurônios em cada. Para treinar essa rede, foram calculadas as saídas do MPC para diferentes estados do sistema. Foi feita uma varredura no espaço de estados do sistema usando um quadrado onde os valores de cada um dos estados varia de -5 até 5 com espaço de 0.1 entre eles. Esses valores foram então utilizados para o treinamento da RN com o uso do algoritmo *Levenberg-Marquardt backpropagation* (MARQUARDT, 1963).

Após o treinamento da RN, o valor do *bias* da camada de saída da mesma ($b_{\ell+1}$) foi ajustado para a garantir que saída da RN tenha valor zero para uma entrada nula, assim garantindo que a origem do sistema em MF é um ponto de equilíbrio. Para isso o valor de $b_{\ell+1}$ foi atualizado da seguinte forma:

$$b_{\ell+1} = \bar{b}_{\ell+1} - v_0, \tag{73}$$

onde $\bar{b}_{\ell+1}$ é o valor do *bias* da última camada obtido a partir do processo de treinamento e v_0 é o valor residual da saída da RN treinada para uma entrada nula.

A RN obtida é descrita por (6) com as matrizes

$$W_1 = \begin{bmatrix} 1.0445 & 0.5765 \\ -0.3736 & 0.1259 \end{bmatrix},\ b_1 = \begin{bmatrix} 2.1176 \\ -2.7835 \end{bmatrix},$$
$$W_2 = \begin{bmatrix} 0.3493 & 0.0131 \\ 1.4562 & -0.5387 \end{bmatrix},\ b_2 = \begin{bmatrix} -1.7966 \\ -0.8850 \end{bmatrix},$$
$$W_3 = \begin{bmatrix} 1.8936 & -0.4543 \end{bmatrix},\ \text{e}\ b_3 = 0.9987.$$

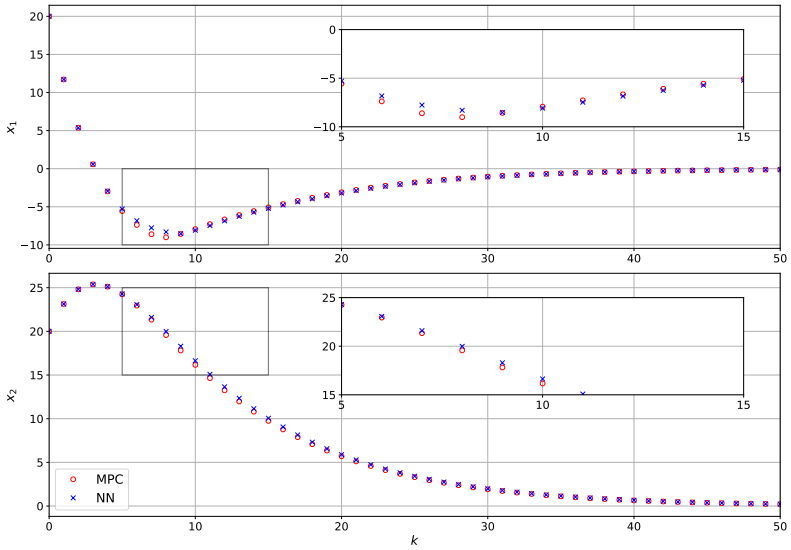
Além disso, foi adicionada uma saturação à saída desta RN (30) com $\overline{v} = 1$ e $\underline{v} = -2$. Resultando em um sistema em MF como em (66) com as seguintes matrizes

$$N_1 = \begin{bmatrix} 0.7326 & -0.0861 \\ 0.1722 & 0.9909 \end{bmatrix},\ N_2^\top = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1.1532 & 0.1212 \\ -0.2766 & -0.0291 \\ -0.6090 & -0.0640 \\ 0.6090 & 0.0640 \end{bmatrix},$$
$$N_3 = \begin{bmatrix} 1.0445 & 0.5765 \\ -0.3736 & 0.1259 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},\ e_0 = \begin{bmatrix} 0.6082 \\ 0.0639 \end{bmatrix}, e_1 = \begin{bmatrix} 2.1176 \\ -2.7835 \\ -1.7966 \\ -0.8850 \\ -0.0013 \\ -2.9987 \end{bmatrix},$$
$$N_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3493 & 0.0131 & 0 & 0 & 0 & 0 \\ 1.4562 & -0.5387 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.8936 & -0.4543 & 0 & 0 \\ 0 & 0 & -1.8936 & 0.4543 & 0 & 0 \end{bmatrix}.$$

Para chegar a esta representação para o sistema, primeiro foram obtidas as representações implícitas baseadas em funções ReLU da RN *feedforward* e da função de saturação, utilizando, respectivamente, (9) e (32). Depois, foi obtida a representação da RN saturada utilizando a interconexão em série de RNs (20), e, por fim, foi utilizado (43) para obter a representação do sistema em MF.

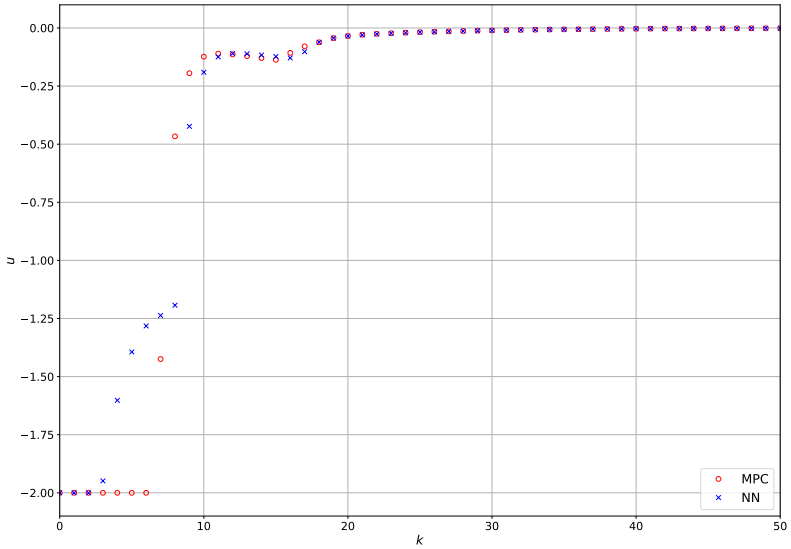
A Figura 11 apresenta a comparação de uma trajetória do sistema controlado pelo MPC e pela RN saturada que emula seu comportamento. Na Figura 12 pode-se observar a comparação do sinal de controle gerado pelo MPC com o sinal de controle da RN saturada.

Figura 11 – Evolução dos estados do sistema do exemplo 1 em MF para uma condição inicial de $x_1 = x_2 = 20$. A resposta do sistema com controlador MPC é representado pelos círculos vermelhos, enquanto o sistema com controlador de RN é representado pelas cruzes azuis.



Fonte: Autor

Figura 12 – Sinal de controle ao longo da trajetória do sistema do exemplo 1 em MF para uma condição inicial de $x_1 = x_2 = 20$. A ação de controle do MPC e da RN são representados, respectivamente, por círculos vermelhos e por cruzes azuis.



Fonte: Autor

Nas duas figuras, pode-se observar que a RN treinada tem o desempenho desejado, apresentando um comportamento próximo do MPC projetado, mesmo para condições iniciais fora do quadrado utilizado para o treinamento da mesma. Nos pontos das amostras de 5 a 10, onde ocorre a saturação do sinal de controle em -2, ocorreu a maior diferença entre o sinal de controle da RN e o sinal de controle do MPC, indicando que a RN treinada não conseguiu emular tão bem o comportamento do MPC para esses valores, o que ocorre devido ao baixo número de neurônios da RN.

Destaca-se que o tempo necessário para calcular a ação de controle do MPC, calculada usando o pacote MPT3 (HERCEG *et al.*, 2013), foi da ordem de grandeza de milissegundos, já o tempo necessário para calcular a ação de controle da RN, calculada através de (6) e (30), foi da ordem de grandeza de microssegundos, de forma a cumprir o objetivo de reduzir o custo computacional para o cálculo da ação de controle.

4.2.1.3 Estabilidade Global da Origem

Aplicando o Teorema 1, certificou-se a estabilidade exponencial global da origem do sistema em MF ao calcular uma solução para as LMIs (67), resultando em $\epsilon_1 = 0.8182$, $\epsilon_2 = 11.6107$ e

$$P = \begin{bmatrix} 6.3785 & 2.3828 & 0.4363 & -0.0373 & 2.3178 & -0.6793 & -1.1741 & 1.0914 \\ 2.3828 & -3.2789 & 0.1005 & 0.0390 & 1.1411 & -0.2398 & -0.5684 & 0.5035 \\ 0.4363 & 0.1005 & -0.6675 & 0.3705 & -0.0539 & -0.6675 & 0.3705 & -0.0539 \\ -0.0373 & 0.0390 & 0.2143 & 0.0097 & -0.2400 & 0.2143 & 0.0097 & -0.2400 \\ 2.3178 & 1.1411 & -0.7492 & -2476.1 & 1.5638 & -0.7492 & -2476.1 & 1.5638 \\ -0.6793 & -0.2398 & -0.6675 & 0.2143 & -0.7492 & 1.2364 & 593.81 & -0.4971 \\ -1.1741 & -0.5684 & 0.3705 & 0.0097 & -2476.1 & 593.81 & 2614.0 & -1.1605 \\ 1.0914 & 0.5035 & -0.0539 & -0.2400 & 1.5638 & -0.4971 & -1.1605 & 0.8044 \end{bmatrix}.$$

Neste exemplo, pode-se observar que as condições de estabilidade propostas foram capazes de verificar a estabilidade global de uma RN *feedforward* ReLU de múltiplas camadas e com bias, dois pontos que são limitações em métodos encontrados na literatura.

4.2.2 Exemplo 2

Considere um sistema discreto no tempo, linear e de segunda ordem

$$x^+ = \begin{bmatrix} 0.7326 & -0.0861 \\ \sigma + 0.1722 & 0.9909 \end{bmatrix} x + \begin{bmatrix} 0.6090 \\ 0.0640 \end{bmatrix} u,$$

que é controlado por uma RN ReLU (isto é, $u = f_{\text{ReLU}}(x)$), esta RN foi treinada para emular o comportamento do mesmo MPC do exemplo anterior e tem o formato de (6) com as seguintes matrizes

$$W_1 = \begin{bmatrix} 0.0194 & -0.2412 \\ -1.0523 & -0.6516 \\ 1.0600 & 0.6563 \end{bmatrix}, b_1 = 0_{3,1},$$

$$W_2 = \begin{bmatrix} -0.0002 & -0.2924 & 0.5923 \\ 0.6852 & -0.0741 & -0.5251 \\ -0.0007 & 0.8464 & 0.1932 \end{bmatrix}, b_2 = 0_{3,1,}$$

$$W_3 = \begin{bmatrix} -0.2503 & 0.0002 & 0.1434 \end{bmatrix} \text{ e } b_3 = 0.$$

Esta RN foi treinada com um *bias* igual a zero para que seja possível aplicar as condições propostas em (YIN; SEILER; ARCAK, 2022), as quais se baseiam na utilização de condições de setor e de limitação da derivada (*slope-boundedness*), possibilitando, assim, a comparação dos resultados obtidos com aqueles decorrentes do Teorema 1.

Desta forma, procuramos verificar a estabilidade global do sistema em MF, utilizando ambas as condições para diferentes valores de $\sigma \geq 0$. Para o método proposto (Teorema 1), a estabilidade exponencial global da origem é certificada para $0 \leq \sigma \leq 2.47$. Por outro lado, a abordagem proposta em (YIN; SEILER; ARCAK, 2022) produz resultados mais conservadores, certificando a estabilidade exponencial da origem apenas para $0 \leq \sigma \leq 0.73$.

Por meio de simulações numéricas, verificamos que, para $\sigma = 2.48$, o sistema em MF não é exponencialmente estável, apresentando trajetórias divergentes, o que mostra que as condições propostas, embora apenas suficientes, são pouco conservadoras neste caso.

4.3 Conclusões

Neste capítulo, foi abordado o problema da certificação de estabilidade global de um sistema não linear representado por uma RN ReLU. Para isso, foi utilizada a representação implícita baseada em funções ReLU, a qual foi apresentada no Capítulo 2.

As principais vantagens do método proposto são:

- Diferentemente de (SAMANIPOUR; POONAWALA, 2023), em que apenas RNs de camada única podem ser tratadas, o método proposto pode ser aplicado tanto a redes de uma única camada quanto a redes multicamadas.
- Pode levar em conta RNs com termos de *bias*, o que não é possível em (YIN; SEILER; ARCAK, 2022) e (NOORI *et al.*, 2024).
- Devido a representação exata da função ReLU, a análise de estabilidade proposta se mostrou menos conservadora do que abordagens que utilizam condições de setor.

Outro ponto abordado neste capítulo foi a utilização de uma RN *feedforward* para aproximar o comportamento de um MPC. Sobre esse ponto, as principais vantagens dessa abordagem foram a redução do custo computacional para o cálculo da ação de controle, comparado a métodos como o MPC ou RN implícitas, a certificação da estabilidade do sistema em MF e a consideração explícita dos limites da ação de controle como uma saturação na saída da RN.

5 ESTABILIDADE LOCAL

Neste capítulo, são apresentadas condições para garantir a estabilidade local da origem de sistemas representados por RNs ReLU e estimar a RAO destes. Com base no arcabouço matemático fundamentado anteriormente, na Seção 5.1 são propostas condições de estabilidade no formato de LMIs. Na Seção 5.2, são apresentados procedimentos para incorporar a presença de restrições de estado às condições propostas, enquanto na Seção 5.3, são abordadas estratégias para determinar estimativas da RAO. Por fim, na Seção 5.4, são apresentados exemplos numéricos como forma de ilustrar a metodologia proposta.

5.1 Condições de Estabilidade

Nesta seção, com o propósito de apresentar uma solução para o Problema 2, são estabelecidas condições para verificar a estabilidade exponencial local da origem de sistemas não lineares na forma de

$$x^+ = N_1 x + N_2 \phi(q) + e_0 \tag{74a}$$

$$q = N_3 x + N_4 \phi(q) + e_1. \tag{74b}$$

A ideia central consiste em verificar

$$\epsilon_1 \|x\|_2^2 \leq V(x) \leq \epsilon_2 \|x\|_2^2,$$

e

$$V(x^+) \leq (1-\eta) V(x),$$

apenas para um subconjunto $\mathcal{D} \subset \mathbb{R}^{n_x}$ do espaço de estados, considerando uma função de Lyapunov PWQ candidata $V(x)$ conforme em (49). Em particular, é considerada uma região $\mathcal{D}$ descrita como o conjunto de nível de uma função PWQ,

$$\mathcal{D} := \left\{ x \in \mathbb{R}^{n_x} | \begin{bmatrix} x \\ \phi(q) \end{bmatrix}^\top D \begin{bmatrix} x \\ \phi(q) \end{bmatrix} \leq 1 \right\}, \tag{75}$$

com $D \in \mathbb{S}^{n_x + n_q}$.

Observação 10. *Caso a matriz N_4 não tenha estrutura estritamente triangular inferior e consequentemente* (74b) *seja mal posta para algum $x \in \mathbb{R}^{n_x}$, deve-se utilizar um conjunto $\mathcal{D}$ dentro do conjunto $\mathcal{W} \subset \mathbb{R}^{n_x}$ onde* (74b) *é bem posta, ou seja, $\mathcal{D} \subset \mathcal{W}$.*

Com base nos lemas e proposições elencados nas seções 3.2 e 3.4, o teorema a seguir fornece condições no formato de LMIs para garantir que uma função PWQ $V(x)$ seja uma função de Lyapunov no domínio $\mathcal{D}$, a partir da aplicação das proposições 1 e 2.

Teorema 2. *Considerando um sistema dado por* (74) *e uma matriz D que define uma região $\mathcal{D}$ com em* (75)*, se existirem matrizes $P \in \mathbb{S}^{n_x+n_q}$, $T_1 \in \mathbb{D}^{n_q}$, $M_1 \in \mathbb{P}^{1+2n_q}$, $T_2 \in \mathbb{D}^{n_q}$, $M_2 \in \mathbb{P}^{1+2n_q}$, $T_3 \in \mathbb{D}^{n_q}$, $M_3 \in \mathbb{P}^{1+2n_q}$, $\tilde{T} \in \mathbb{D}^{2n_q}$, $F_a \in \mathbb{S}^{n_{\tilde{\chi}}}$, $F_b \in \mathbb{R}^{4n_q \times n_{\tilde{\chi}}}$ ($n_{\tilde{\chi}} := 1 + n_x + 4n_q$), $S \in \mathbb{P}^{4n_q}$, $\tilde{T}_0$, $\tilde{T}_i \in \mathbb{D}^{2n_q}$, Ξ_0, $\Xi_i \in \mathbb{P}^{1+4n_q}$, escalares positivos ϵ_1, ϵ_2, α, γ, $\eta < 1$ e escalares não negativos β_0, β_i, para $i = 1, \ldots, 4n_q$ de modo que as seguintes desigualdades matriciais sejam satisfeitas*

$$\Pi_\chi^\top \{H(P, \epsilon_1) + \Psi(T_1) - \Upsilon M_1 \Upsilon^\top\} \Pi_\chi \geq 0, \tag{76a}$$

$$\Pi_\chi^\top \{-H(P, \epsilon_2) + \Psi(T_2) - \Upsilon M_2 \Upsilon^\top\} \Pi_\chi \geq 0, \tag{76b}$$

$$\tilde{\Pi}_\chi^\top \{\tilde{H}(P, \eta) + \tilde{\Psi}(\tilde{T}) - \tilde{M}(F_a, F_b, S)\} \tilde{\Pi}_\chi \geq 0, \tag{76c}$$

$$\tilde{\Pi}_\chi^\top \{F_a + \tilde{\Psi}(\tilde{T}_0) - \tilde{\Upsilon} \Xi_0 \tilde{\Upsilon}^\top\} \tilde{\Pi}_\chi - \beta_0 \bar{D} \geq 0, \tag{76d}$$

$$\tilde{\Pi}_\chi^\top \{\bar{F}_{bi} + \tilde{\Psi}(\tilde{T}_i) - \tilde{\Upsilon} \Xi_i \tilde{\Upsilon}^\top\} \tilde{\Pi}_\chi - \beta_i \bar{D} \geq 0, \tag{76e}$$

$$\alpha \bar{D} - \bar{P} + \Pi_\chi^\top \{\Psi(T_3) - \Upsilon M_3 \Upsilon^\top\} \Pi_\chi \geq 0, \tag{76f}$$

onde

$$H(P, \epsilon) := \Pi^\top \left(P - \epsilon \begin{bmatrix} I_{n_x} & 0_{n_x, n_q} \\ 0_{n_q, n_x} & 0_{n_q} \end{bmatrix} \right) \Pi, \; \tilde{H}(P, \eta) := (1 - \eta) \tilde{\Pi}^\top P \tilde{\Pi} - \tilde{\Pi}_+^\top P \tilde{\Pi}_+,$$

$$\tilde{M}(S, F_a, F_b) := F_a + \mathbf{He} \left(\begin{bmatrix} 0_{1+n_x, n_{\tilde{\chi}}} \\ F_b \end{bmatrix} \right) + \begin{bmatrix} 0_{1+n_x} & 0_{1+n_x, 4n_q} \\ 0_{4n_q, 1+n_x} & S \end{bmatrix},$$

$$\Psi(T) := \begin{bmatrix} 0 & 0_{1,n_x} & 0_{1,n_q} & 0_{1,n_q} \\ 0_{n_x,1} & 0_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & 0_{n_q} & T \\ 0_{n_q,1} & 0_{n_q,n_x} & T & 0_{n_q} \end{bmatrix}, \; \tilde{\Psi}(\tilde{T}) := \begin{bmatrix} 0 & 0_{1,n_x} & 0_{1,2n_q} & 0_{1,2n_q} \\ 0_{n_x,1} & 0_n & 0_{n_x,2n_q} & 0_{n_x,2n_q} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & 0_{2n_q} & \tilde{T} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & \tilde{T} & 0_{2n_q} \end{bmatrix},$$

$$\Upsilon := \begin{bmatrix} 1 & 0_{1,2n_q} \\ 0_{n_x,1} & 0_{n_x,2n_q} \\ 0_{2n_q,1} & I_{2n_q} \end{bmatrix}, \; \tilde{\Upsilon} := \begin{bmatrix} 1 & 0_{1,4n_q} \\ 0_{n_x,1} & 0_{n_x,4n_q} \\ 0_{4n_q,1} & I_{4n_q} \end{bmatrix}, \; \Pi_\chi := \begin{bmatrix} 1 & 0_{1,n_x} & 0_{1,n_q} \\ 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} \\ -e_1 & -N_3 & I_{n_q} - N_4 \end{bmatrix},$$

$$\tilde{\Pi}_\chi := \begin{bmatrix} 1 & 0_{1,n_x} & 0_{1,2n_q} \\ 0_{n_x,1} & I_{n_x} & 0_{n_x,2n_q} \\ 0_{2n_q,1} & 0_{2n_q,n_x} & I_{2n_q} \\ -\tilde{e}_1 & -\tilde{N}_3 & I_{2n_q} - \tilde{N}_4 \end{bmatrix}, \; \tilde{\Pi} := \begin{bmatrix} 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} & 0_{n_x,2n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} & 0_{n_q} & 0_{2n_q} \end{bmatrix},$$

$$\tilde{\Pi}_+ := \begin{bmatrix} e_0 & N_1 & N_2 & 0_{n_x,n_q} & 0_{n_x,2n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & 0_{n_q} & I_{n_q} & 0_{2n_q} \end{bmatrix},\ \Pi := \begin{bmatrix} 0_{n_x,1} & I_{n_x} & 0_{n_x,n_q} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & I_{n_q} & 0_{n_q} \end{bmatrix},$$
$$\bar{P} := \begin{bmatrix} \gamma & 0_{1,n_x+n_q} \\ 0_{n_x+n_q,1} & -P \end{bmatrix},\ \bar{D} := \begin{bmatrix} 1 & 0_{1,n_x+n_q} \\ 0_{n_x+n_q,1} & -D \end{bmatrix}, \bar{F}_{bi} := \mathit{He}\left(\begin{bmatrix} F_{b(i,:)} \\ 0_{n_x+4n_q,n_{\tilde{\chi}}} \end{bmatrix} \right),$$

então:

a) a origem do sistema (74) *é localmente exponencialmente estável;*

b) para todas as condições iniciais $x(0) \in \mathcal{E}(P,\gamma) \subset \mathcal{D}$*, com*

$$\mathcal{E}(P,\gamma) := \left\{ x \in \mathbb{R}^{n_x} \mid \begin{bmatrix} x \\ \phi(q) \end{bmatrix}^\top P \begin{bmatrix} x \\ \phi(q) \end{bmatrix} \leq \gamma \right\} \tag{77}$$

segue que $\lim_{k\to\infty} x(k) \to 0\ \forall x(0) \in \mathcal{E}(P,\gamma)$.

Prova. Da mesma forma que no Teorema 1, as inequações (76a) e (76b) garantem que $V(x)$ é definido positivo.

De forma semelhante ao realizado no Teorema 1, ao multiplicar à direita e à esquerda o lado esquerdo de (76c) pelo vetor $\tilde{\xi} := \begin{bmatrix} 1 & x^\top & \phi^\top(\tilde{q}) \end{bmatrix}^\top$ e pela sua transposta, respectivamente, considerando $\tilde{q} := \begin{bmatrix} q^\top & (q^+)^\top \end{bmatrix}^\top$, $\tilde{M}(x)$ com a estrutura dada em (62) e usando (63), obtém-se

$$(1-\eta)V(x) - V(x^+) + s_1(\tilde{T},\tilde{q}) - s_2(\tilde{M}(x),\tilde{q}) \geq 0. \tag{78}$$

Assim, a partir da Proposição 2 considerando $h(x) = (1-\eta)V(x) - V(x^+)$, segue que, se $\tilde{M}(x)$ é não negativa $\forall x \in \mathcal{D}$, então

$$V(x^+) \leq (1-\eta)V(x), \forall x \in \mathcal{D},$$

o que implica que

$$\Delta V(x) = V(x^+) - V(x) \leq -\eta V(x) < 0 \quad \forall x \in \mathcal{D}. \tag{79}$$

Agora, considerando $\xi := \begin{bmatrix} 1 & x^\top & \phi^\top(q) \end{bmatrix}^\top$ e

$$d(x) = \xi^\top \bar{D} \xi, \tag{80}$$

o conjunto $\mathcal{D}$ em (75) pode ser descrito alternativamente da seguinte forma

$$\mathcal{D} = \{x \in \mathbb{R}^{n_x} \mid d(x) \geq 0\}. \tag{81}$$

Então, ao multiplicar à direita e à esquerda os lados esquerdos de (76d) e (76e) pelo vetor $\tilde{\xi}$ e sua transposta e utilizando (62b) e (64), obtém-se, respectivamente, que

$$f_a(x) - \beta_0 d(x) + s_1(\tilde{T}_0,\tilde{q}) - s_2(\Xi_0,\tilde{q}) \geq 0, \tag{82a}$$

$$f_{b(i)}(x) - \beta_i d(x) + s_1(\tilde{T}_1, \tilde{q}) - s_2(\Xi_i, \tilde{q}) \geq 0, \tag{82b}$$

Usando a Proposição 1 com $h(x) = f_a(x) - \beta_0 d(x)$ e $h(x) = f_{b(i)}(x) - \beta_i d(x)$, respectivamente para (82a) e (82b) tem-se que

$$f_a(x) \geq \beta_0 d(x) \text{ e } f_{b(i)}(x) \geq \beta_i d(x).$$

Assim, se $x \in \mathcal{D}$ conforme definido em (75), segue que $f_a(x) \geq 0$ e $f_b(x) \geq 0 \; \forall x \in \mathcal{D}$, e pode-se concluir que $\tilde{M}(x)$, com a estrutura dada em (62), é não negativa para todo $x \in \mathcal{D}$.

Finalmente, ao multiplicar à direita e à esquerda o lado esquerdo de (76f) pelo vetor ξ e pela sua transposta, respectivamente, e utilizando novamente a Proposição 1, agora com $h(x) = \alpha \xi^\top \bar{D} \xi - \xi^\top \bar{P} \xi$, obtém-se que

$$\alpha d(x) \geq \xi^\top \bar{P} \xi \tag{83}$$

Levando em conta que $\xi^\top \bar{P} \xi = \gamma - V(x)$, pode-se concluir que

$$\forall x \in \mathcal{E}(P, \gamma) \; \Rightarrow \; \alpha d(x) \geq 0,$$

e, como α é positivo, conclui-se que

$$\mathcal{E}(P, \gamma) \subset \mathcal{D}.$$

A partir de (79) conclui-se que é $\mathcal{E}(P, \gamma)$ é um conjunto positivamente invariante e contrativo com relação às trajetórias do sistema (74). Assim, obtém-se que $\lim_{k \to \infty} x(k) \to 0$, $\forall x(0) \in \mathcal{E}(P, \gamma)$, e a partir de (69) obtém-se que

$$||x(k)|| \leq \left(\frac{\epsilon_2}{\epsilon_1} \right)^{\frac{1}{2}} e^{\ln(\sqrt{1-\eta})k} ||x(0)|| \quad \forall x(0) \in \mathcal{E}(P, \gamma) \subset \mathcal{D}. \tag{84}$$

Portanto, concluímos a estabilidade exponencial da origem e que qualquer superfície de nível de V contido em $\mathcal{D}$ constitui uma estimativa da RAO. $\square$

5.2 Restrições de Estados

Nesta seção, serão apresentadas condições suplementares que podem ser incorporadas a LMIs (76) de forma a garantir que as trajetórias do sistema dadas por (74) respeitem as restrições de estados impostas, dado que $x(0) \in \mathcal{E}(P, \gamma)$.

Para considerar as restrições de estado impostas ao sistema, considera-se que elas são descritas genericamente por um politopo no espaço de estados

$$\mathcal{R}_H := \{x \in \mathbb{R}^n \mid |h_r x| \leq c_r, r = 1, \ldots, n_r\}, \tag{85}$$

onde $c_r > 0$ e $h_r \in \mathbb{R}^{n_x}$.

A primeira opção para garantir tais restrições de estados é restringir a região $\mathcal{D}$ para que esta esteja dentro deste poliedro, isto é, $\mathcal{D} \subset \mathcal{R}_H$. Assim, se $\mathcal{E}(P,\gamma) \subset \mathcal{D}$, segue-se que, para todo $x(0) \in \mathcal{E}(P,\gamma)$, as trajetórias correspondentes irão satisfazer as restrições de estado.

Outra alternativa seria restringir $\mathcal{E}(P,\gamma)$ para que seja incluído diretamente em $\mathcal{R}_H$. Para isso, é necessário adicionar as seguintes LMIs a (76), considerando $r = 1, \ldots, n_r$

$$\alpha_r R_r - \bar{P} + \Pi^\top \{\Psi(\hat{T}_r) - \Upsilon \hat{M}_r \Upsilon^\top\} \Pi \geq 0, \tag{86}$$

com a matriz auxiliar

$$R_r := \begin{bmatrix} 2c_r & -h_r & 0_{1,n_q} \\ -h_r^\top & 0_{n_x} & 0_{n_x,n_q} \\ 0_{n_q,1} & 0_{n_q,n_x} & 0_{n_x,n_q} \end{bmatrix} \tag{87}$$

e as seguintes variáveis de decisão: $\hat{T}_r \in \mathbb{D}^{n_q}$, $\hat{M}_r \in \mathbb{P}^{1+2n_q}$, e o escalar positivo α_r.

De forma semelhante a (76f), (86) faz uso da Proposição 1 para garantir que $\mathcal{E}(P,\gamma)$ esteja inclusa em cada uma das restrições do politopo $\mathcal{R}_H$. Note que, ao multiplicar o lado esquerdo de (86) à direita e à esquerda pelo vetor $\xi = \begin{bmatrix} 1 & x^\top & \phi(q)^\top \end{bmatrix}^\top$ e pela sua transposta, respectivamente, obtêm-se

$$\alpha_r \xi^\top R_r \xi - \xi^\top \bar{P} \xi + s_1(\hat{T}_r, q) - s_2(\hat{M}_r, q) \geq 0.$$

Então, utilizando a Proposição 1 com $h(x) = \alpha_r \xi^\top R_r \xi - \xi^\top \bar{P} \xi$, tem-se que

$$\alpha_r \xi^\top R_r \xi \geq \xi^\top \bar{P} \xi.$$

Por fim, levando em conta que $\xi^\top R_r \xi = 2(c_r - h_r x)$, $\xi^\top \bar{P} \xi = \gamma - V(x)$ e $\alpha_r > 0$, conclui-se que

$$\mathcal{E}(P,\gamma) \subset \mathcal{R}_H.$$

Isso pode relaxar as restrições de forma a possibilitar uma estimativa maior da RAO, porém resulta em um maior número de variáveis de decisão e, consequentemente, um maior esforço computacional para calcular a estimativa.

5.3 Determinação de Estimativas da RAO

Esta seção apresenta problemas de otimização voltados à maximização da estimativa da RAO, correspondente ao tamanho de $\mathcal{E}(P,\gamma)$ obtido no Teorema 2.

Diferentemente de regiões definidas por funções quadráticas, cujo tamanho pode ser maximizado por meio da minimização do traço da matriz definidora (entre outros critérios (TARBOURIECH *et al.*, 2011)), não existe uma relação direta para regiões PWQ, como é o caso de $\mathcal{E}(P,\gamma)$. Isto ocorre porque, em regiões PWQ, o espaço de estados está particionado em diferentes domínios, de forma que, em cada uma dessas partições, a

função de Lyapunov assume uma geometria distinta. O resultado é um tamanho total que consiste na composição de diferentes segmentos elipsoidais, o que impede uma relação direta e simplificada com o traço da matriz. Diante disso, são elencadas estratégias para buscar a maximização de $\mathcal{E}(P, \gamma)$ nesse contexto.

Para calcular uma estimativa maximizada do RAO, propõe-se o seguinte problema de otimização:

$$\begin{aligned} \text{minimizar} : \ & \text{tr}(P_1) - \gamma \\ \text{sujeito a} : \ & (76). \end{aligned} \tag{88}$$

com $P_1 := P_{(1:n_x, 1:n_x)}$ sendo a porção da matriz P referente aos estados do sistema. Aqui novamente a desigualdade (76c) é uma BMI em η e P. Devido a isso, é fixado um η pequeno, já que se deseja maximizar a estimativa da RAO.

Outra estratégia para aprimorar a estimativa da RAO consiste em adicionar à (88) a seguinte restrição para a inclusão de um ponto específico x_p

$$\begin{bmatrix} x_p \\ q_p(x_p) \end{bmatrix}^\top P \begin{bmatrix} x_p \\ q_p(x_p) \end{bmatrix} - \gamma < 0. \tag{89}$$

Assim, pode-se definir n_d diferentes direções e $x_p = \alpha x_d$, onde x_d é um ponto de norma 1 que define a direção, e α é um fator de escala. A partir disso, é realizada uma busca por bisseção no fator α com o objetivo de maximizar a estimativa da RAO na direção de x_d. Dessa forma, a estimativa final da RAO será o conjunto resultante da união das n_d regiões obtidas (BAZANELLA; Gomes da Silva Junior, 2005).

Observa-se que no Teorema 2 é preciso definir uma região $\mathcal{D}$ na qual $\mathcal{E}(P, \gamma)$ será inclusa. Quanto maior o volume de $\mathcal{D}$, mais $\mathcal{E}(P, \gamma)$ terá espaço para crescer no problema de otimização, porém, caso $\mathcal{D}$ seja muito grande, o problema de otimização pode se tornar infactível. Devido a isso, para se obter a maior região $\mathcal{D}$ onde o problema de otimização é factível, ela é parametrizada como uma bola

$$D = \begin{bmatrix} -dI_{n_x} & 0_{n_x, n_q} \\ 0_{n_q, n_x} & 0_{n_q} \end{bmatrix},$$

e então é realizada uma busca por bisseção em d para encontrar o maior valor possível de d para o qual o problema de otimização é factível.

5.4 Exemplos Numéricos

Nesta seção, são apresentados alguns exemplos numéricos[1] para ilustrar a aplicação das condições de estabilidade local propostas.

[1] Os exemplos numéricos foram implementados em linguagem Python, utilizando as bibliotecas NumPy (HARRIS *et al.*, 2020), CvXpy (DIAMOND; BOYD, 2016) e TensorFlow (ABADI *et al.*, 2015), com o auxílio do solver MOSEK (MOSEK ApS, 2024).

5.4.1 Exemplo 1

Considere o sistema (40), com

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ e } B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

Gostaríamos de controlar este sistema com um MPC que garanta as restrições $|u| < 1$, $|x_{(1)}| < 5$ e $|x_{(2)}| < 5$ (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025). O MPC implementado é dado por (37) com

$$\begin{aligned}
&Q_x = I_2, Q_u = 1, N_h = 2, \\
&E^\top = \begin{bmatrix} 0_{2,4} & -A^\top & -(A^2)^\top & A^\top & (A^2)^\top \end{bmatrix}, \\
&G^\top = \begin{bmatrix} I_2 & -I_2 \begin{bmatrix} B^\top & (AB)\top & -B^\top & -(AB)^\top \\ 0_{1,2} & B^\top & 0_{1,2} & -B^\top \end{bmatrix} \end{bmatrix}, \\
&R^\top = \begin{bmatrix} \mathbf{1}_{1,4} & 5\mathbf{1}_{1,8} \end{bmatrix}.
\end{aligned}$$

Em seguida, uma RN é treinada para aproximar seu comportamento. Neste caso, foi utilizada uma RN com função de ativação ReLU, 2 entradas, 1 saída e 2 camadas ocultas com 3 neurônios cada. Para treinar essa rede, foram calculadas as saídas do MPC para diferentes estados do sistema. Foi realizada uma varredura no espaço de estados do sistema, variando cada um dos estados de -10 a 10 com um espaçamento de 0,05 entre cada ponto. Em seguida, os pontos em que o MPC não era factível foram removidos e os valores restantes foram utilizados para o treinamento da RN, empregando o algoritmo de *backpropagation* de Levenberg-Marquardt (MARQUARDT, 1963).

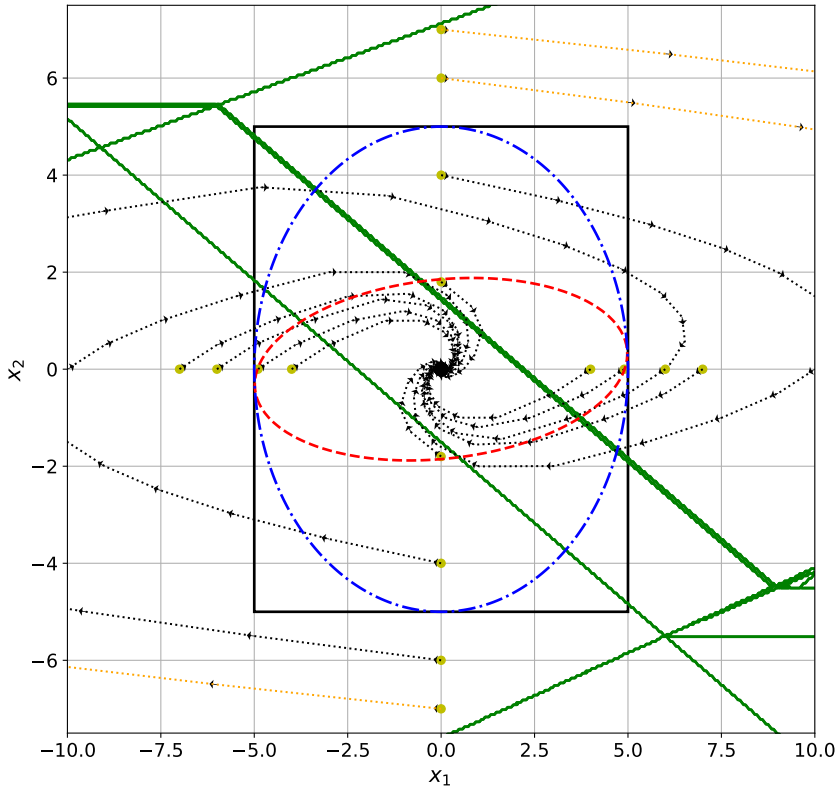
Após o treinamento, o valor de $b_{\ell+1}$ foi ajustado para garantir que a saída da RN seja zero para uma entrada nula, assegurando assim que a origem do sistema em MF seja um ponto de equilíbrio. A RN resultante é dada por

$$\begin{aligned}
W_1 &= \begin{bmatrix} 0.5531 & 0.8301 \\ 0.1448 & -0.4123 \\ -0.1030 & -0.3673 \end{bmatrix}, \; W_3^\top = \begin{bmatrix} 1.6765 \\ 2.0373 \\ -1.3133 \end{bmatrix}, \\
W_2 &= \begin{bmatrix} -0.0188 & 1.7414 & -1.0147 \\ -0.4372 & 1.3219 & -2.2793 \\ -0.0682 & 1.8536 & -0.3862 \end{bmatrix}, \\
b_1 &= \begin{bmatrix} 1.2502 \\ -3.1355 \\ -2.6159 \end{bmatrix}, \; b_2 = \begin{bmatrix} -0.0003 \\ 1.0889 \\ 0.1656 \end{bmatrix} \text{ e } b_3 = -0.9994.
\end{aligned}$$

Então, para garantir as restrições de entrada de controle, foi aplicada uma função saturação (30) com $\overline{v} = -\underline{v} = 1$ na saída da RN.

Por fim, para estimar a RAO do sistema em MF, o problema de otimização (88) foi utilizado com $\eta = 0.0001$ e $d \leq 5^{-2}$ (para manter a região $\mathcal{D}$ dentro da área definida pelas restrições $|x_{(1)}| < 5$ e $|x_{(2)}| < 5$). A Figura 13 mostra a RAO estimada do sistema controlado pela RN. A RAO estimada é representada pela curva vermelha tracejada, enquanto o conjunto $\mathcal{D}$ é representado pela curva azul traço-ponto e as restrições de estado são representadas por uma linha preta sólida. As fronteiras da partição sobre a qual o sistema PWA resultante é definido são indicadas pelas linhas verdes sólidas. Algumas trajetórias do sistema são ilustradas como linhas pontilhadas, com suas condições iniciais marcadas por um círculo amarelo. Linhas pretas pontilhadas indicam trajetórias convergentes, enquanto linhas laranjas pontilhadas indicam trajetórias divergentes.

Figura 13 – RAO estimada (linha vermelha tracejada), $\mathcal{D}$ (linha azul traço-ponto), restrições de estado (linha preta sólida), partições PWA do sistema (linha verde sólida), trajetórias convergentes (linhas pretas pontilhadas), trajetórias divergentes (linhas laranjas pontilhadas) para o controlador de RN.



Fonte: Autor

Enquanto algumas trajetórias que começaram dentro de $\mathcal{D}$ e fora da RAO estimada (i.e. $\mathcal{E}(P, \gamma)$) convergem para a origem, muitas delas não satisfazem as restrições de estado. Por outro lado, todas as trajetórias que começam dentro da RAO estimada convergem para a origem e respeitam as restrições de estado.

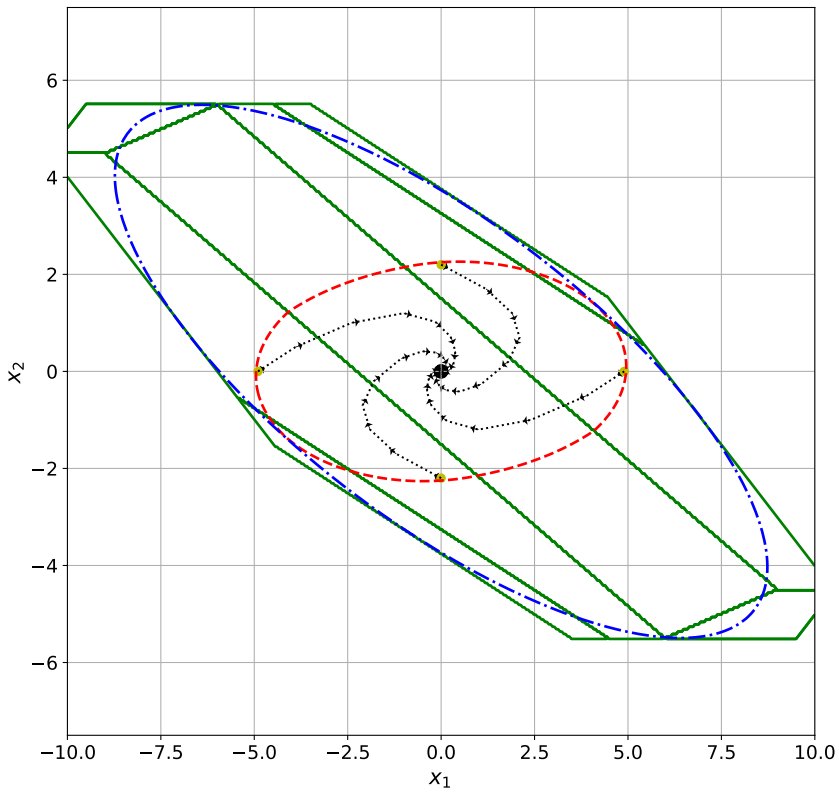
Para fins de comparação, o RAO do sistema controlado pelo MPC foi estimado, representando o MPC como uma RN implícita na forma (10) e utilizando o mesmo método

para obter o sistema em MF e a estimativa da RAO. Neste caso, o Teorema 2 foi utilizado, com o problema de otimização (88), $\eta = 0{,}0001$ e a região $\mathcal{D}$ fixada como um elipsoide na região em que o MPC é factível $\mathcal{D} \subset \mathcal{W}$ (ver (CABRAL; VALMORBIDA; GOMES DA SILVA JR., 2025)). A escolha desta região se da pelo fato de que o MPC, por construção, já garante o cumprimento das restrições de estado e de entrada dentro de sua região de factibilidade, de forma que pode-se utilizar a maior região $\mathcal{D}$ possível. Assim, a matriz que define a região $\mathcal{D}$ é dada por:

$$D = \begin{bmatrix} -0.0284 & -0.0331 & 0_{1,12} \\ -0.0331 & -0.0717 & 0_{1,12} \\ 0_{12,1} & 0_{12,1} & 0_{12,12} \end{bmatrix}.$$

A Figura 14 mostra a RAO estimada do sistema controlado pelo MPC como a curva vermelha tracejada, enquanto o conjunto $\mathcal{D}$ corresponde à curva azul traço-ponto. As partições PWA do sistema resultante em MF são indicadas pelas linhas verdes contínuas. Algumas trajetórias do sistema são representadas por linhas pretas pontilhadas, com suas condições iniciais marcadas por um círculo amarelo.

Figura 14 – RAO estimado (linha vermelha tracejada), $\mathcal{D}$ (linha azul traço-ponto), partições PWA do sistema (linha verde contínua) e trajetórias (linhas pretas pontilhadas) para o MPC.

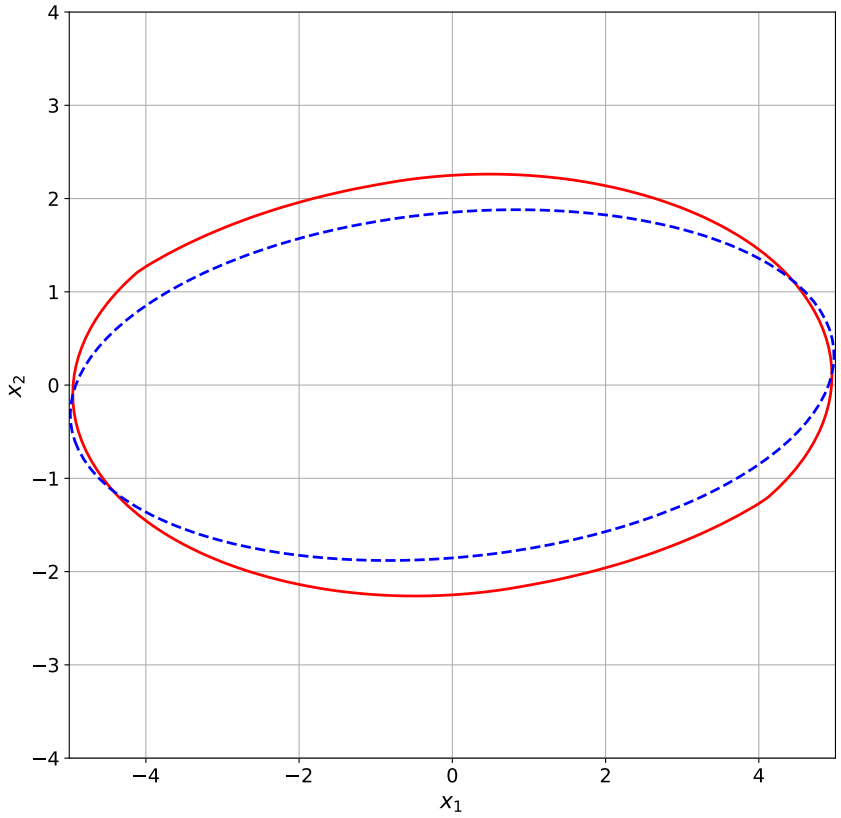


Fonte: Autor

Note que, embora as partições dos sistemas PWA resultantes nas Figuras 13 e 14 não sejam idênticas, nas regiões em que o sistema irá operar, essas partições são, de fato,

semelhantes. Isso mostra que a RN efetivamente aproxima o comportamento do MPC na região de interesse. A Figura 15 apresenta uma comparação entre as duas RAOs estimadas: a do sistema controlado pela RN é representada por uma curva azul tracejada, enquanto a do sistema controlado pelo MPC é representada por uma curva vermelha contínua. Observa-se que a RAO estimada do sistema com o controlador baseado em RN tem uma área aproximadamente do mesmo tamanho que a RAO estimada do sistema com o controlador MPC.

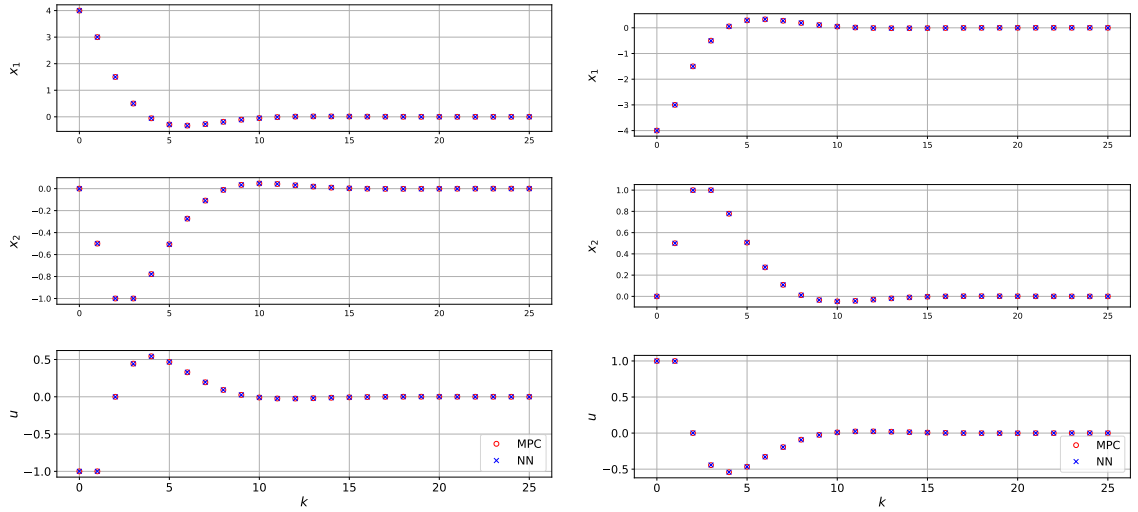
Figura 15 – RAO estimado pelo controlador RN (linha azul tracejada) vs RAO estimado pelo MPC (linha vermelha contínua).



Fonte: Autor

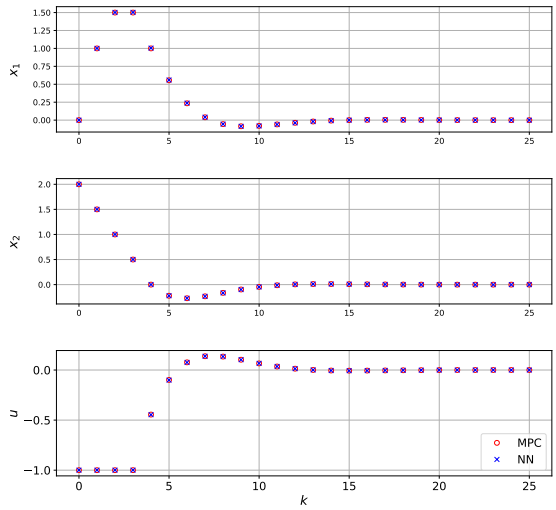
A Figura 16 mostra a evolução dos estados do sistema e dos sinais de controle para os dois controladores diferentes (RN e MPC) com diferentes condições iniciais. Pode-se observar que o comportamento de ambos os sistemas é praticamente idêntico. Por outro lado, o tempo de execução do MPC para cada passo foi, em média, de $2{,}02\ ms$, e o tempo de execução da RN para cada passo foi, em média, de $2{,}77\ \mu s$, resultando em uma redução do tempo de execução de aproximadamente 729 vezes.

Figura 16 – Comparação da evolução dos estados do sistema e do sinal de controle para o sistema com o controlador de RN (cruzes azuis) e para o sistema com o controlador MPC (círculos vermelhos).

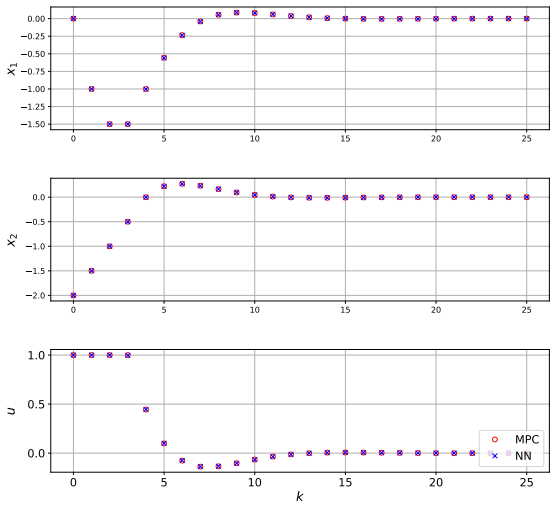


(a) $x(0) = \begin{bmatrix} 4 & 0 \end{bmatrix}$.

(b) $x(0) = \begin{bmatrix} -4 & 0 \end{bmatrix}$.



(c) $x(0) = \begin{bmatrix} 0 & 2 \end{bmatrix}$.



(d) $x(0) = \begin{bmatrix} 0 & -2 \end{bmatrix}$.

Fonte: Autor

É importante notar que a RN proposta, que aproxima a solução de MPC, possui duas camadas com 3 neurônios cada, enquanto a implementação do MPC é equivalente a uma *RN implícita* de 12 neurônios. Além disso, a aplicação em tempo real da RN implícita requer a solução de um problema de otimização a cada instante de tempo para se obter a solução da equação implícita. Por outro lado, graças à estrutura *feedforward* da RN, o valor de $u(x)$ pode ser calculado diretamente, sem qualquer procedimento de otimização.

5.5 Conclusões

Neste capítulo, foi abordado o problema da certificação de estabilidade local e estimativa da RAO de um sistema não linear representado por uma RN ReLU. Para isso, foi utilizada a representação implícita baseada em funções ReLU apresentada no Capítulo 2.

Da mesma forma que as condições de estabilidade global, as principais vantagens do método proposto são:

- Pode ser aplicado tanto a redes de uma única camada quanto a redes multicamadas;
- Pode levar em conta RNs com termos de *bias*;
- Devido a representação exata da função ReLU, a análise de estabilidade proposta se mostrou menos conservadora do que abordagens que utilizam condições de setor.

Destaca-se que a modelagem da função de ativação ReLU como uma não linearidade limitada por setor não permite um estreitamento do setor para derivar condições locais, como é feito em (YIN; SEILER; ARCAK, 2022) para funções como Tanh, sigmoide e saturação.

Outro ponto abordado neste capítulo foi a utilização de uma RN *feedforward* para aproximar o comportamento de um MPC. Aqui, as mesmas vantagens de redução do custo computacional da RN *feedforward*, comparada à implementação do MPC, do MPC explícito ou da RN implícita equivalente ao MPC, são observadas. Além da consideração explícita dos limites da ação de controle como uma saturação na saída da RN, a obediência as restrições de estado foi garantida por meio da inclusão da estimativa da RAO na região definida pelas mesmas.

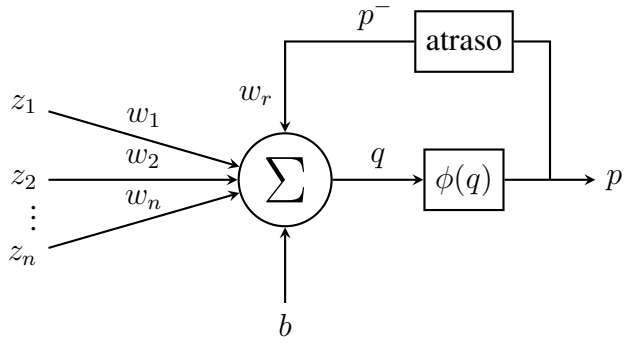
6 EXTENSÕES

Neste capítulo, são apresentadas extensões para o uso da representação implícita baseada em funções ReLU. Nas seções 6.1 e 6.2, é abordada, respectivamente, a modelagem de RNs Recorrentes e RNs Convolucionais na representação implícita utilizada. Já na Seção 6.3, estende-se o uso desta formulação implícita para RNs com funções de ativação não ReLU.

6.1 RN Recorrente

As RNs Recorrentes (RNRs) constituem uma classe de arquiteturas de RNs projetadas para lidar com dados que têm dependência temporal ou sequencial. Diferente dos modelos de *feedforward* convencionais, as RNRs possuem um mecanismo de realimentação o qual permite a persistência de estados internos (ELMAN, 1990). Essa propriedade confere à rede uma "memória" dinâmica, tornando-a capaz de captar a dinâmica de sequências de dados de comprimento variável. A Figura 17 apresenta um exemplo de neurônio de uma RNR.

Figura 17 – Exemplo de neurônio recorrente.



Fonte: Autor

Neste caso, a saída desse neurônio recorrente, p, é o resultado da aplicação da função de ativação a uma variável q. Esta variável corresponde a soma ponderada de três componentes:

- as n_z entradas do neurônio, z_i multiplicadas pelos seus pesos, w_i, $i = 1, 2, \ldots, n_z$;
- a saída do neurônio no instante anterior, p^-, multiplicada pelo seu peso w_r;
- e um valor de *bias* b.

ou seja,

$$p = \phi\left(\sum_{i=1}^{n_z} w_i z_i + w_r p^- + b\right). \tag{90}$$

As RNRs são utilizadas em diversas aplicações onde é necessário levar em conta a dependência temporal de uma sequência. No processamento de linguagem natural, as RNRs são essenciais para interpretar o sentido de uma frase, levando em conta a ordem das palavras e como cada uma delas é influenciada pelo que foi dito anteriormente (SUTSKEVER; VINYALS; LE, 2014). No campo de análise de séries temporais, as RNRs podem ser aplicadas na predição de fenômenos geofísicos, como, por exemplo, níveis de inundação, analisando dados e padrões históricos para mitigar desastres naturais (CHANG *et al.*, 2018). No reconhecimento de fala, a RNR processa uma sequência de ondas sonoras para identificar fonemas e palavras, considerando que o som atual depende do que foi dito imediatamente antes (GRAVES; MOHAMED; HINTON, 2013). Da mesma forma, na análise de movimento, a rede recebe dados de sensores inerciais e correlaciona as acelerações em diferentes instantes para classificar atividades físicas ou identificar padrões de caminhada em pacientes (HAMAD *et al.*, 2016).

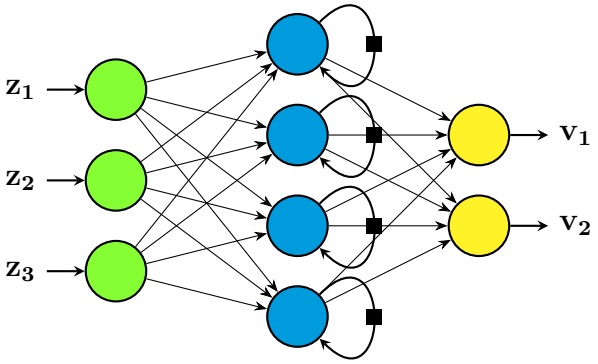
Da mesma forma que as RNs *feedforward* são aproximadores universais de funções contínuas em conjuntos compactos, as RNRs possuem a capacidade de aproximar, com precisão arbitrária, qualquer sistema dinâmico em tempo discreto desde que possuam um número suficientemente grande de neurônios (SIEGELMANN; SONTAG, 1991). Com base nessa capacidade de representação, as RNRs também podem ser empregadas em tarefas de identificação de sistemas, sendo utilizadas para modelar a dinâmica de processos complexos e não lineares a partir de dados experimentais (LJUNG, 1998; PILLONETTO *et al.*, 2023).

Existem diversas arquiteturas de RNRs, com a semelhança entre elas sendo a existência de estados ocultos na sua arquitetura. A Figura 18 apresenta um exemplo de uma RNR.

Primeiramente, são considerados os tipos mais simples de RNRs:

- RNs de recorrência independente: onde a realimentação ocorre apenas no próprio neurônio, como no neurônio da Figura 17 e a RN da Figura 18;
- RNs de recorrência local: onde a saída de um neurônio no instante de tempo anterior influência a saída de todos os neurônios da sua camada.

Figura 18 – RN Recorrente com 3 entradas, 2 saídas e uma camada oculta com 4 neurônios com recorrência independente.



Fonte: Autor

Para obter a representação implícita baseada em funções ReLU dessa arquitetura de RNs, primeiro partimos da representação matemática para uma RNR (WU; CHEN; CHEN, 2022)

$$\begin{aligned} p_0 &= z, \\ p_i &= \phi(W_i p_{i-1} + U_i p_i^- + b_i), \quad i = 1, \ldots, \ell, \\ v &= \psi(W_{(\ell+1)} p_\ell + b_{\ell+1}), \end{aligned} \tag{91}$$

onde $v \in \mathbb{R}^{n_v}$ é o vetor de saída da RN e para a i-ésima camada, $p_i \in \mathbb{R}^{n_{p_i}}$ é o vetor de saída (ou seja, a saída dos n_{p_i} neurônios da *i*-ésima camada), p_i^- é o vetor de saída da camada no instante anterior de tempo, $b_i \in \mathbb{R}^{n_{p_i}}$ é o vetor dos *bias*, $W_i \in \mathbb{R}^{n_{p_i} \times n_{p_{i-1}}}$ é a matriz dos pesos e $U_i \in \mathbb{R}^{n_{p_i} \times n_{p_{i-1}}}$ é a matriz dos pesos recorrentes.

A estrutura de U_i depende da arquitetura da RNR. U_i pode ser diagonal caso se deseje que a memória de cada neurônio só influencie o próprio, como é o caso das RNRs independentes, ou U_i pode ser cheia caso se deseje que a memória de cada neurônio influencie todos os neurônios da camada, como em RNRs locais.

De forma semelhante a RN *feedforward*, define-se $q_i := W_i p_{i-1} + U_i p_i^- + b_i$ como a entrada da função de ativação vetorial da i-ésima camada e os vetores $q \in \mathbb{R}^{n_q}$ e $p \in \mathbb{R}^{n_p}$ como a concatenação das entradas e saídas da função de ativação de cada camada, respectivamente, isto é

$$q := \begin{bmatrix} q_1 \\ \vdots \\ q_\ell \end{bmatrix} \quad \text{e} \quad p := \begin{bmatrix} p_1 \\ \vdots \\ p_\ell \end{bmatrix}, \tag{92}$$

onde $n_q = n_p = n_{p_1} + \cdots + n_{p_\ell}$. Assim, segue que $p = \phi(q)$ e usando $x = p^-$,

$$\begin{bmatrix} q \\ v \\ x^+ \end{bmatrix} = N \begin{bmatrix} z \\ p \\ x \end{bmatrix} + e, \tag{93}$$

com

$$N := \left[\begin{array}{c|ccc|c|ccc} W_1 & 0_{n_{p_1},n_{p_1}} & \cdots & 0_{n_{p_1},n_{p_{\ell-1}}} & 0_{n_{p_1},n_{p_\ell}} & U_1 & 0_{n_{p_1},n_{p_2}} & \cdots & 0_{n_{p_1},n_{p_\ell}} \\ 0_{n_{p_2},n_z} & W_2 & \cdots & 0_{n_{p_2},n_{p_{\ell-1}}} & 0_{n_{p_2},n_{p_\ell}} & 0_{n_{p_2},n_{p_1}} & U_2 & \cdots & 0_{n_{p_2},n_{p_\ell}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{n_{p_\ell},n_z} & 0_{n_{p_\ell},n_{p_1}} & \cdots & W_\ell & 0_{n_{p_\ell},n_{p_\ell}} & 0_{n_{p_\ell},n_{p_1}} & 0_{n_{p_\ell},n_{p_2}} & \cdots & U_\ell \\ \hline 0_{n_v,n_z} & 0_{n_v,n_{p_1}} & \cdots & 0_{n_v,n_{p_{\ell-1}}} & W_{\ell+1} & 0_{n_v,n_{p_1}} & 0_{n_v,n_{p_2}} & \cdots & 0_{n_v,n_{p_\ell}} \\ \hline 0_{n_q,n_z} & \multicolumn{3}{c|}{I_{n_q}} & & \multicolumn{3}{c}{0_{n_q}} \end{array}\right]$$

$$= \begin{bmatrix} N_3 & N_4 & N_{rq} \\ N_1 & N_2 & N_{rv} \\ R_1 & R_2 & R_3 \end{bmatrix} \text{ e } e := \left[\begin{array}{cccc|c|c} b_0^\top & b_1^\top & \cdots & b_{\ell-1}^\top & b_\ell^\top & 0_{1,n_q} \end{array}\right]^\top = \begin{bmatrix} e_1^\top & e_0^\top & e_r^\top \end{bmatrix}^\top. \tag{94}$$

Então, a representação implícita baseada em rampas da função PWA f_{RReLU} é dada por

$$x^+ = R_1 z + R_2 \phi(q) + R_3 x + e_r \tag{95a}$$

$$f_{\text{RReLU}}(z) = N_1 z + N_2 \phi(q) + N_{rv} x + e_0 \tag{95b}$$

$$q = N_3 z + N_4 \phi(q) + N_{rq} x + e_1 \tag{95c}$$

com $N_1 \in \mathbb{R}^{n_v \times n_z}$, $N_2 \in \mathbb{R}^{n_v \times n_q}$, $N_3 \in \mathbb{R}^{n_q \times n_z}$, $N_4 \in \mathbb{R}^{n_q \times n_q}$, $N_{rv} \in \mathbb{R}^{n_v \times n_x}$, $N_{rq} \in \mathbb{R}^{n_q \times n_x}$, $R_1 \in \mathbb{R}^{n_x \times n_z}$, $R_2 \in \mathbb{R}^{n_x \times n_q}$, $R_3 \in \mathbb{R}^{n_x \times n_x}$, $e_0 \in \mathbb{R}^{n_v}$, $e_1 \in \mathbb{R}^{n_q}$, $e_r \in \mathbb{R}^{n_x}$ e $n_x = n_q$.

Observação 11. *Para redes com recorrência simples, o estado oculto x corresponde diretamente à saída dos neurônios da rede no instante anterior. Nessa configuração, a representação implícita simplifica-se ao definir $R_1 = 0_{n_q,n_z}$, $R_2 = I_{n_q}$, $R_3 = 0_{n_q}$ e $e_r = 0_{1,n_q}$ de forma que a equação de estados* (95a) *é dada por*

$$x^+ = I_{n_q} \phi(q). \tag{96}$$

Entretanto, a estrutura genérica apresentada em (95a) *é utilizada para permitir a representação de arquiteturas de recorrência mais complexas, discutidas nas seções seguintes, onde o estado pode ser uma combinação ponderada da entrada, resultados de outras camadas, estados prévios e termos de bias.*

Observação 12. *Uma f_{ReLU} descrita na representação* (10) *pode ser considerada um caso particular da f_{RReLU} na representação* (95) *com $n_x = 0$.*

6.1.1 RNR Completamente Conectada

Também é possível que o estado de um neurônio influencie outras camadas da RN, e não apenas a própria, possibilitando a conexão recorrente entre qualquer neurônio de qualquer camada. Considerando uma RNR completamente conectada, pode-se escrever

$$\begin{aligned} p_0 &= z, \\ p_i &= \phi\left(\sum_{j=0}^{i-1} W_{ji}p_j + \sum_{j=1}^{\ell} U_{ji}p_j^- + b_i\right), \quad i = 1, \ldots, \ell, \\ v &= \psi\left(\sum_{j=0}^{\ell} W_{j(\ell+1)}p_j + \sum_{j=1}^{\ell} U_{j(\ell+1)}p_j^- + b_{\ell+1}\right), \end{aligned} \tag{97}$$

onde $v \in \mathbb{R}^{n_v}$ é o vetor de saída da RN para a i-ésima camada, $p_i \in \mathbb{R}^{n_{p_i}}$ é o vetor de saída (ou seja, a saída dos n_{p_i} neurônios da *i*-ésima camada), p_i^- é o vetor de saída da camada no instante anterior de tempo, $b_i \in \mathbb{R}^{n_{p_i}}$ é o vetor dos *bias*, $W_{ji} \in \mathbb{R}^{n_{p_i} \times n_{p_j}}$ é a matriz dos pesos relativa à influência da camada j na camada i e $U_{ji} \in \mathbb{R}^{n_{p_i} \times n_{p_{i-1}}}$ é a matriz dos pesos recorrentes relativa à influência da camada j no instante anterior na camada i.

$$q := \begin{bmatrix} q_1 \\ \vdots \\ q_\ell \end{bmatrix} \quad \text{e} \quad p := \begin{bmatrix} p_1 \\ \vdots \\ p_\ell \end{bmatrix}, \tag{98}$$

onde $n_q = n_p = n_{p_1} + \cdots + n_{p_\ell}$. Assim, segue que $p = \phi(q)$ e usando $x = p^-$,

$$\begin{bmatrix} q \\ v \\ x^+ \end{bmatrix} = N \begin{bmatrix} z \\ p \\ x \end{bmatrix} + e, \tag{99}$$

com

$$\begin{aligned} N &:= \left[\begin{array}{c|ccccc|cccc} W_{01} & 0_{n_{p_1},n_{p_1}} & \ldots & 0_{n_{p_1},n_{p_{\ell-1}}} & 0_{n_{p_1},n_{p_\ell}} & U_{11} & U_{12} & \ldots & U_{1\ell} \\ W_{02} & W_{12} & \ldots & 0_{n_{p_2},n_{p_{\ell-1}}} & 0_{n_{p_2},n_{p_\ell}} & U_{12} & U_{22} & \ldots & U_{2\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ W_{0\ell} & W_{1\ell} & \ldots & W_\ell & 0_{n_{p_\ell},n_{p_\ell}} & U_{1\ell} & U_{2\ell} & \ldots & U_{\ell\ell} \\ \hline W_{0(\ell+1)} & W_{1(\ell+1)} & \ldots & W_{(\ell-1)(\ell+1)} & W_{\ell(\ell+1)} & U_{1(\ell+1)} & U_{2(\ell+1)} & \ldots & U_{\ell(\ell+1)} \\ \hline 0_{n_q,n_z} & \multicolumn{4}{c|}{I_{n_q}} & \multicolumn{4}{c}{0_{n_q}} \end{array}\right] \\ &= \begin{bmatrix} N_3 & N_4 & N_{rq} \\ N_1 & N_2 & N_{rv} \\ R_1 & R_2 & R_3 \end{bmatrix} \text{ e } e := \left[\begin{array}{cccc|c|c} b_0^\top & b_1^\top & \ldots & b_{\ell-1}^\top & b_\ell^\top & 0_{1,n_q} \end{array}\right]^\top = \begin{bmatrix} e_1^\top & e_0^\top & e_r^\top \end{bmatrix}^\top. \end{aligned} \tag{100}$$

A partir dessas novas matrizes e vetores, pode-se escrever a RNR completamente conectada na forma implícita baseada na função ReLU utilizando (95).

6.1.2 Atraso de Ordem Superior

Outra possível extensão das RNR é utilizar unidades de memória com mais do que um atraso no tempo, uma característica central dos modelos NARX (do inglês *Nonlinear Autoregressive with Exogenous inputs*), que permitem capturar dependências temporais complexas em processos dinâmicos (LIN *et al.*, 1996). Utilizando a representação implícita baseada em funções ReLU (95), pode-se levar em conta qualquer quantidade de atrasos. Para isso, considera-se:

$$\begin{aligned} N_{rv} &:= \begin{bmatrix} N_{rv1} & N_{rv1} & \dots & N_{rvn_d} \end{bmatrix} \\ N_{rq} &:= \begin{bmatrix} N_{rq1} & N_{rq2} & \dots & N_{rqn_d} \end{bmatrix} \\ x(k) &:= \begin{bmatrix} x_1(k)^\top & x_2(k)^\top & \dots & x_{n_d}(k)^\top \end{bmatrix}^\top \\ R_1 &:= 0_{n_x,n_z},\ R_2 := \begin{bmatrix} I_{n_q} \\ 0_{n_q(n_d-1),n_q} \end{bmatrix},\ R_3 := \begin{bmatrix} 0_{n_q,n_q(n_d-1)} \\ I_{n_q(n_d-1)} \end{bmatrix},\ e_r := 0_{n_x,1}, \end{aligned} \tag{101}$$

onde $n_d > 0$ é o número de atrasos, $x_i(k) = p(k-i)$, $n_x = n_q n_d$, N_{rqi} é a matriz de pesos de realimentação das camadas ocultas relativa ao atraso i, N_{rvi} é a matriz de pesos de realimentação para a saída relativa ao atraso i.

6.1.3 Estados Generalizados

Por fim, os estados da RNR podem não ser exatamente a saída de cada neurônio, mas sim uma combinação das saídas dos neurônios e dos estados anteriores, podendo inclusive existir um número de estados diferente do número de neurônios recorrentes.

Uma RNR com estados generalizados, de ℓ camadas ocultas e n_x estados, tem a seguinte representação matemática

$$\begin{aligned} p_0 &= z, \\ p_i &= \phi\left(\sum_{j=0}^{i-1} W_{ji} p_j + U_i x + b_i\right), \quad i = 1, \dots, \ell, \\ x^+ &= \sum_{j=0}^{\ell} W_{rj} p_j + U_r x + b_r, \\ v &= \psi\left(\sum_{j=0}^{\ell} W_{j(\ell+1)} p_j + U_{(\ell+1)} x + b_{\ell+1}\right), \end{aligned} \tag{102}$$

onde $v \in \mathbb{R}^{n_v}$ é o vetor de saída da RN para a i-ésima camada, $p_i \in \mathbb{R}^{n_{p_i}}$ é o vetor de saída (ou seja, a saída dos n_{p_i} neurônios da *i*-ésima camada), $b_i \in \mathbb{R}^{n_{p_i}}$ é o vetor dos *bias*, $W_{ji} \in \mathbb{R}^{n_{p_i} \times n_{p_j}}$ é a matriz dos pesos relativa à influência da camada j na camada i, x é o vetor de estados da RNR no instante atual, x^+ é o vetor dos estados da RNR no próximo instante, $U_i \in \mathbb{R}^{n_{p_i} \times n_x}$ é a matriz dos pesos recorrentes relativa à influência do vetor de estados x na camada i, $W_{rj} \in \mathbb{R}^{n_x \times n_{p_i}}$ é a matriz de pesos relativa à influência da saída

da camada j no vetor de estados no próximo instante x^+, $U_r \in \mathbb{R}^{n_x \times n_x}$ é a matriz dos pesos recorrentes relativa à influência do vetor de estados no instante atual x no vetor de estados no próximo instante x^+ e b_r é o vetor de *bias* relativo ao estado x.

$$q := \begin{bmatrix} q_1 \\ \vdots \\ q_\ell \end{bmatrix} \quad \text{e} \quad p := \begin{bmatrix} p_1 \\ \vdots \\ p_\ell \end{bmatrix}, \tag{103}$$

onde $n_q = n_p = n_{p_1} + \cdots + n_{p_\ell}$. Assim, segue que $p = \phi(q)$,

$$\begin{bmatrix} q \\ v \\ x^+ \end{bmatrix} = N \begin{bmatrix} z \\ p \\ x \end{bmatrix} + e, \tag{104}$$

com

$$N := \left[\begin{array}{ccccc|c} W_{01} & 0_{n_{p_1},n_{p_1}} & \dots & 0_{n_{p_1},n_{p_{\ell-1}}} & 0_{n_{p_1},n_{p_\ell}} & U_1 \\ W_{02} & W_{12} & \dots & 0_{n_{p_2},n_{p_{\ell-1}}} & 0_{n_{p_2},n_{p_\ell}} & U_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ W_{0\ell} & W_{1\ell} & \dots & W_\ell & 0_{n_{p_\ell},n_{p_\ell}} & U_\ell \\ \hline W_{0(\ell+1)} & W_{1(\ell+1)} & \dots & W_{(\ell-1)(\ell+1)} & W_{\ell(\ell+1)} & U_{(\ell+1)} \\ \hline W_{r0} & W_{r1} & \dots & W_{r(\ell-1)} & W_{r\ell} & U_r \end{array}\right] = \begin{bmatrix} N_3 & N_4 & N_{rq} \\ N_1 & N_2 & N_{rv} \\ R_1 & R_2 & R_3 \end{bmatrix} \text{ e}$$

$$e := \left[\begin{array}{cccc|c|c} b_0^\top & b_1^\top & \dots & b_{\ell-1}^\top & b_\ell^\top & b_r^\top \end{array}\right]^\top = \begin{bmatrix} e_1^\top & e_0^\top & e_r^\top \end{bmatrix}^\top. \tag{105}$$

A partir dessas novas matrizes e vetores, pode-se escrever a RNR com estados generalizados na forma implícita baseada na função ReLU utilizando (95).

6.1.4 Estabilidade

A presença de laços de realimentação intrínsecos e o consequente comportamento dinâmico das RNRs introduz a necessidade de verificar a estabilidade das mesmas, de forma a evitar comportamentos instáveis, como a divergência dos estados, especialmente quando se utilizam funções de ativação não saturadas, como a ReLU.

Para verificar a estabilidade interna de uma RNR f_{RReLU} descrita como (95), pode-se reescreve-la como um sistema não linear autônomo não forçado (considerando $z = 0$ e focado exclusivamente na dinâmica de estados) no formato (74), com:

$$N_1 = R_3,\ N_2 = R_2,\ e_0 = e_r,\ N_3 = N_{rq},\ N_4 = N_4 \text{ e } e_1 = e_1. \tag{106}$$

Desta forma, é possível utilizar os teoremas 1 e 2 para verificar a estabilidade da RNR.

Esta formulação também é particularmente útil, pois possibilita a análise da estabilidade de distintos sistemas não lineares modelados por meio de RNRs. Além disso, empregando as interconexões descritas na Seção 2.5 (as quais podem facilmente ser adaptadas para a representação (95)), pode-se representar diferentes conexões entre sistemas dinâmicos e funções, sendo estes RNs ou modelados por RNs.

6.1.5 Exemplos Numéricos

Nesta subseção, são apresentados alguns exemplos numéricos[1] para ilustrar a aplicação da forma implícita baseada em funções ReLU para RNRs de forma a permitir a utilização das condições de estabilidade global e local propostas, respectivamente, nos capítulos 4 e 5.

6.1.5.1 Exemplo 1

Considere a RNR de (YI; TAN, 2004), a qual é dada pela seguinte equação:

$$x^+ = W\phi(x) + b \tag{107}$$

com

$$W = \begin{bmatrix} 0 & -3 & 0.5 \\ -7 & 0.5 & -4 \\ 0.5 & -2 & 0 \end{bmatrix} e\ b = \begin{bmatrix} 3 \\ -3 \\ 3 \end{bmatrix}.$$

Esta rede pode ser escrita como uma RNR com estados generalizados (102) como

$$\begin{aligned} p_0 &= z = 0, \\ p_1 &= \phi(x), \\ x^+ &= Wp_1 + b, \\ v &= x. \end{aligned}$$

A partir disso, pode-se representar essa RNR na forma implícita (95), utilizando (105), o que resulta em

$$\begin{aligned} &N_1 = 0_3,\ N_2 = 0_3,\ N_{rq} = I_3,\ e_0 = 0_{3,1}, \\ &N_3 = 0_3,\ N_4 = 0_3,\ N_{rv} = I_3,\ e_1 = 0_{3,1}, \\ &R_1 = 0_3,\ R_2 = W,\ R_3 = 0_3,\ e_r = b. \end{aligned}$$

Para analisar a estabilidade desta RNR pode-se usar (106) para escrevê-la na forma (74) com

$$N_1 = 0_3,\ N_2 = W,\ e_0 = b,\ N_3 = I_3,\ N_4 = 0_3,\ \text{e}\ e_1 = 0_{3,1},$$

o que é equivalente a (107).

Esta RNR possui um ponto de equilíbrio $x_e = [6 \quad -69 \quad 6]^\top$.

Usando (45) para transladar o ponto de equilíbrio para a origem e, em seguida, aplicando o Teorema 1, a estabilidade global do sistema foi certificada com uma função de

[1] Os exemplos numéricos foram implementados em linguagem Python, utilizando as bibliotecas NumPy (HARRIS *et al.*, 2020), CvXpy (DIAMOND; BOYD, 2016) e TensorFlow (ABADI *et al.*, 2015), com o auxílio do solver MOSEK (MOSEK ApS, 2024).

Lyapunov PWQ como em (49), onde

$$P = \begin{bmatrix} 162.095 & -3.513 & -222.026 & 123.715 & 27.819 & -30.585 \\ -3.513 & 1.199 & 8.427 & 3.728 & 25.323 & -3.728 \\ -222.026 & 8.427 & 461.897 & -3.003 & 62.718 & -90.127 \\ 123.715 & 3.728 & -3.003 & -87.849 & -20.374 & -5.281 \\ 27.819 & 25.323 & 62.718 & -20.374 & 810.102 & -32.306 \\ -30.585 & -3.728 & -90.127 & -5.281 & -32.306 & 98.410 \end{bmatrix}.$$

Destaca-se que o método proposto em (YI; TAN, 2004) só é capaz de certificar a estabilidade local do ponto de equilíbrio.

6.1.5.2 Exemplo 2

Considere a RNR de (ZHANG *et al.*, 2009), a qual é dada pela seguinte equação:

$$x^+ = W\phi(x) + b \tag{108}$$

com

$$W = \begin{bmatrix} 0.2 & -3 & -2 \\ -2 & 0.2 & -3 \\ -3 & -4 & 0.2 \end{bmatrix} e\ b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Da mesma forma que no exemplo anterior, a RNR é escrita como uma RNR com estados generalizados na forma implícita (95) e, então, para analisar sua estabilidade, é utilizado (106) para escrevê-la na forma (74), resultando em

$$N_1 = 0_3,\ N_2 = W,\ e_0 = b,\ N_3 = I_3,\ N_4 = 0_3,\ \text{e}\ e_1 = 0_{3,1},$$

o que é equivalente a (108).

Este sistema possui 3 pontos de equilíbrio estáveis:

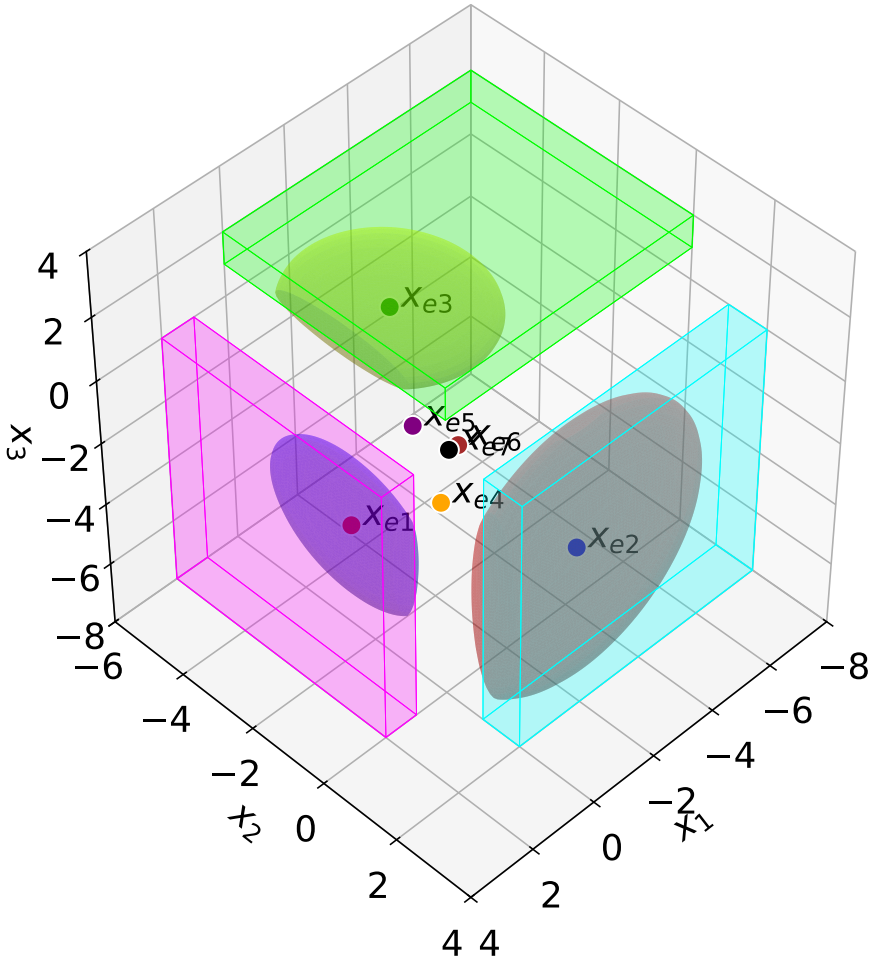
$$x_{e_1} = \begin{bmatrix} 1.25 & -1.5 & -2.75 \end{bmatrix}^\top,\ x_{e_2} = \begin{bmatrix} -2.75 & 1.25 & -4 \end{bmatrix}^\top,\ x_{e_3} = \begin{bmatrix} -1.5 & -2.75 & 1.25 \end{bmatrix}^\top,$$

e 4 pontos de equilíbrio instáveis:

$$x_{e4} = \begin{bmatrix} 0.4104 & 0.2239 & -1.1269 \end{bmatrix}^\top,\ x_{e5} = \begin{bmatrix} 0.2239 & -0.6791 & 0.4104 \end{bmatrix}^\top,$$
$$x_{e6} = \begin{bmatrix} -0.1444 & 0.1937 & 0.2817 \end{bmatrix}^\top,\ x_{e7} = \begin{bmatrix} 0.0675 & 0.1497 & 0.2484 \end{bmatrix}^\top.$$

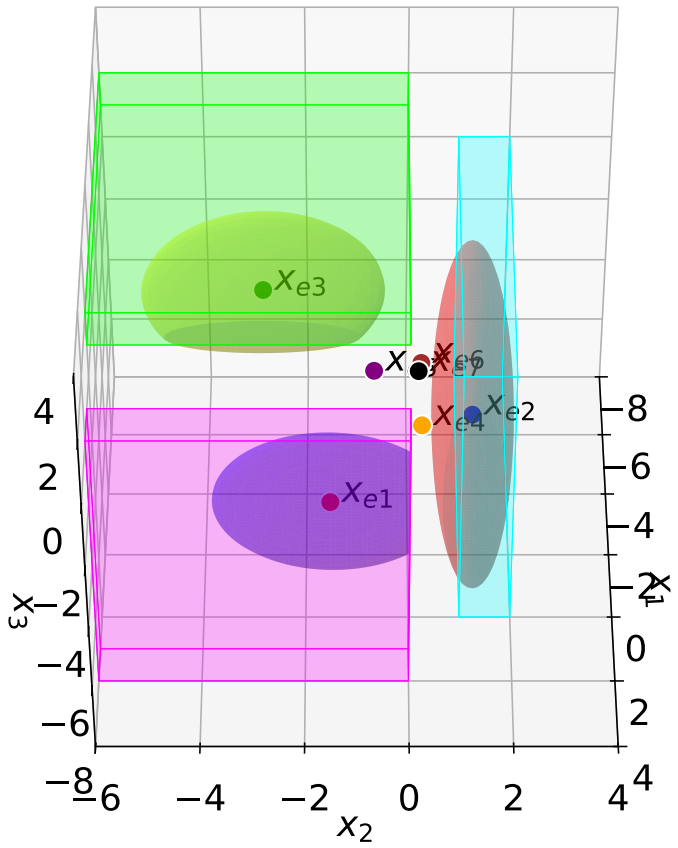
A existência de múltiplos pontos de equilíbrio impede que a convergência individual para esses pontos seja global. Assim, analisamos a estabilidade exponencial local e estimamos a região de atração de cada equilíbrio estável individualmente. Isso pode ser feito transladando o equilíbrio para a origem usando (45) e aplicando o Teorema 2. A partir desse procedimento, estimamos as regiões de atração, as figuras 19 e 20 mostram a comparação destas RAOs com as obtidas por (ZHANG *et al.*, 2009).

Figura 19 – Pontos de equilíbrio e regiões de atração calculadas pelo Teorema 2: x_{e_1} (elipse roxa), x_{e_2} (elipse vermelha) e x_{e_3} (elipse amarela) e regiões de atração calculadas por (ZHANG *et al.*, 2009): x_{e_1} (retângulo roxo), x_{e_2} (retângulo azul claro) e x_{e_3} (retângulo verde). Ponto de vista 1.



Fonte: Autor

Figura 20 – Pontos de equilíbrio e regiões de atração calculadas pelo Teorema 2: x_{e_1} (elipse roxa), x_{e_2} (elipse vermelha) e x_{e_3} (elipse amarela) e regiões de atração calculadas por (ZHANG *et al.*, 2009): x_{e_1} (retângulo roxo), x_{e_2} (retângulo azul claro) e x_{e_3} (retângulo verde). Ponto de vista 2.



Fonte: Autor

Embora as regiões obtidas pelo método proposto sejam menores do que as obtidas por (ZHANG *et al.*, 2009), as RAOs obtidas pelo Teorema 2 têm maior penetração em algumas áreas próximas a outros pontos de equilíbrio. Isso ocorre porque método deste trabalho calcula uma curva de nível de uma função PWA centrada no ponto de equilíbrio e não uma região retangular descentralizada, que é mais adequada para este problema específico.

Um ponto importante a ressaltar é que, utilizando o *framework* proposto neste trabalho, é possível verificar a estabilidade de RNRs multicamadas, diferentemente de (YI; TAN, 2004; ZHANG *et al.*, 2009; EBIHARA *et al.*, 2021b), onde são tratadas apenas RNRs de camada única.

6.2 RN Convolucional

As RNs Convolucionais (RNCs) foram desenvolvidas para processar dados com topologia em grade, sendo projetadas especificamente para emular a percepção visual humana ao extrair padrões espaciais e hierárquicos de forma automatizada. Ao contrário das arquiteturas densas convencionais, as RNCs utilizam operações de convolução para preservar a correlação entre valores de entrada vizinhos, o que as torna uma referência para tarefas de visão computacional (LECUN *et al.*, 1998). Atualmente, suas aplicações abrangem desde o reconhecimento facial (TAIGMAN *et al.*, 2014) e a classificação de imagens médicas (RONNEBERGER; FISCHER; BROX, 2015) até o monitoramento em tempo real em sistemas de direção autônoma (REDMON *et al.*, 2016).

As camadas mais comuns em arquiteturas de RNCs são:

- **C (Convolução):** Responsável pela extração de características espaciais através da aplicação de filtros lineares (*kernels*) que geram mapas de ativação.
- **P (Agrupamento/*Pooling*):** Realiza a subamostragem espacial para reduzir a dimensionalidade e conferir invariância a pequenas translações, utilizando operações de máximo (*max pooling*) ou média (*average pooling*).
- **A (Ativação):** Introduz não-linearidades no sistema (comumente via função ReLU), permitindo a aproximação de funções complexas e a seleção de características relevantes.
- **F (Vetorização/*Flattening*):** Atua como uma interface de transição, transformando os mapas de características multidimensionais em um vetor unidimensional, preparando os dados para as camadas neurais clássicas.
- **D (Densa):** Localizada geralmente ao final da rede, ela utiliza uma estrutura *feed-forward* clássica para mapear as características extraídas pelas camadas anteriores às classes de saída.

Na prática, estas camadas são organizadas em estruturas modulares e repetitivas. Um padrão genérico e amplamente adotado para esta classe de RN define-se pelo empilhamento de blocos de extração e compressão, seguidos pela etapa final de classificação:

$$\underbrace{[(C \to A)^{n_e} \to P]^{m_e}}_{\text{Extração e Compressão}} \longrightarrow \underbrace{F \to D}_{\text{Classificação}}$$

Onde n_e denota o número de etapas de extração (convolução e ativação) dentro de cada bloco, e m_e representa a quantidade de vezes que o bloco completo de extração e compressão é repetido ao longo da arquitetura. Esta configuração modular permite que a rede aprenda representações hierárquicas de complexidade crescente em diferentes escalas espaciais, sendo finalizada pela operação de vetorização (F) que prepara os dados para a tomada de decisão na camada densa (D).

Embora as RNCs sejam menos comuns em aplicações de controle, tornando as condições de estabilidade citadas menos aplicáveis a elas, sua representação implícita baseada em funções ReLU oferece outras utilidades relevantes.

Em (CABRAL, 2024), propõe-se um problema de otimização para estimar a constante de Lipschitz de funções PWA, o que pode ser usado para fornecer uma métrica da sensibilidade da rede a perturbações na entrada (FAZLYAB *et al.*, 2019; HU; SEILER; RAGHUNATHAN, 2022). Em redes classificadoras, essa constante permite quantificar o nível de distorção (como ruídos, borrões ou variações de iluminação) que uma imagem pode sofrer sem que a classificação original seja alterada.

Devido a isto, neste capítulo é mostrado como obter a representação implícita baseada em funções ReLU das diferentes camadas de uma RNC de forma a, utilizando a interconexão em série dessas camadas (20), representar uma RNC completa na forma implícita (10).

Como a representação implícita utilizada neste trabalho supõe vetores para as entradas e saídas, caso a RN considere matrizes como entradas e saídas, como é o caso da maioria das camadas de RNCs, é preciso transformar essas matrizes em vetores. Para isso, é utilizada a seguinte operação

$$\text{vec}(Z) := [Z_{(1,1)}, \ldots, Z_{(1,m)}, Z_{(2,1)}, \ldots, Z_{(2,m)}, \ldots, Z_{(n,1)}, \ldots, Z_{(n,m)}]^\top. \tag{109}$$

Observação 13. *É utilizada a ordenação por linhas para operação de vetorização ao invés da ordenação por coluna, a qual é matematicamente mais comum, para manter o sentido clássico das operações de convolução e pooling (avançando primeiro pelas colunas e depois pelas linhas).*

6.2.1 Camada de Vetorização e Camada Densa

Uma vez que as entradas e saídas das camadas da RNC já se encontram vetorizadas para possibilitar seu tratamento na forma implícita, baseada em funções ReLU, não se faz necessário considerar explicitamente uma camada de vetorização.

Já a Camada Densa pode ser tratada com uma RN *feedforward*, utilizando o exposto na Seção 2.2.

6.2.2 Camada de Convolução

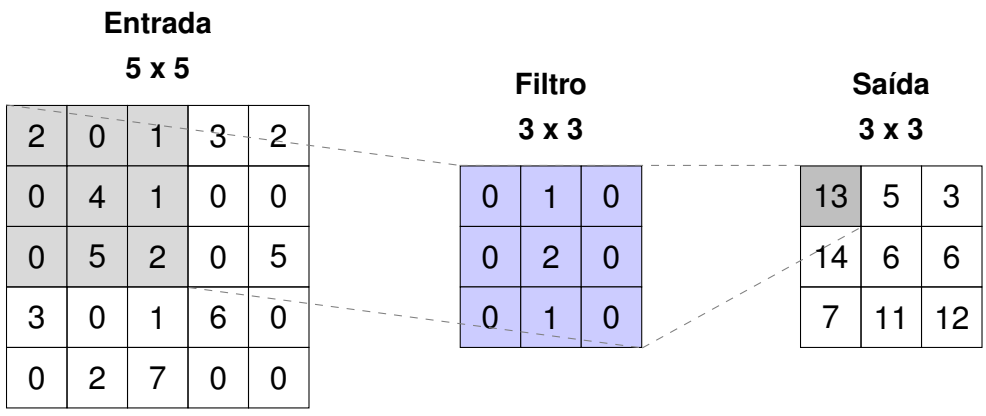
Uma camada de convolução atua como um extrator de características locais. Diferentemente das camadas densas, que processam a entrada inteira de forma simultânea, a convolução opera sobre pequenas sub-regiões do sinal de entrada, permitindo a identificação de padrões específicos, tais como bordas, texturas e formas.

O funcionamento da camada de convolução segue uma lógica de janela deslizante. O processo ocorre na seguinte ordem (GOODFELLOW; BENGIO; COURVILLE, 2016):

- O filtro é posicionado no canto superior esquerdo da entrada.
- Em cada posição, realiza-se a multiplicação elemento a elemento entre os pesos do filtro e os valores da região coberta. Os resultados dessas multiplicações são somados e adicionados a um termo de *bias*, gerando um único valor escalar no mapa de características da saída.
- O filtro desloca-se horizontalmente ao longo das colunas da entrada. Esta varredura continua até que o filtro atinja a borda lateral direita da entrada.
- Após completar a varredura de uma linha, o filtro retorna à extremidade esquerda e desloca-se verticalmente para baixo. O processo de varredura horizontal é então reiniciado para esta nova linha. Esse ciclo se repete até que o canto inferior direito da imagem seja processado.

A Figura 21 apresenta um exemplo de uma camada de convolução sem *bias*.

Figura 21 – Exemplo de uma camada de convolução.



Fonte: Autor

6.2.2.1 Caso 1D

Para facilitar o entendimento, antes de mostrar a representação implícita baseada em funções ReLU de uma camada de convolução típica, em duas dimensões, primeiro é considerado o caso de uma dimensão.

Tendo uma entrada $z \in \mathbb{R}^{n_z}$, um filtro $k \in \mathbb{R}^{n_k}$, $n_k \leq n_z$ e um *bias* $b \in \mathbb{R}$. A saída $v \in \mathbb{R}^{n_v}$, com $n_v = n_z - n_k + 1$, é dada por:

$$v_{(i)} = \sum_{j=1}^{n_k} k_{(j)} z_{(j+i-1)} + b, \text{ para } i = 1, 2, \ldots, n_v. \tag{110}$$

Esta operação pode ser escrita na forma implícita (10) com

$$v = \begin{bmatrix} k_{(1)} & k_{(2)} & \ldots & k_{(n_k)} & 0 & 0 & \ldots & 0 \\ 0 & k_{(1)} & \ldots & k_{(n_k-1)} & k_{(n_k)} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & k_{(1)} & k_{(2)} & \ldots & k_{(n_k)} \end{bmatrix} z + \begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix}. \tag{111}$$

Observa-se que, neste caso, a matriz N_1 é composta por n_v linhas do vetor $k^\top$ deslocado, de modo a posicioná-lo corretamente em relação à entrada z. Com o intuito de simplificar a formulação, define-se uma matriz de seleção S_i, a qual mapeia o filtro k sobre os elementos correspondentes da entrada z para cada componente da saída $v_{(i)}$:

$$S_i := \begin{bmatrix} 0_{n_k,i-1} & I_{n_k} & 0_{n_k,n_z-n_k-(i-1)} \end{bmatrix}, \; i = 1, 2, \ldots, n_v. \tag{112}$$

Assim, a formulação implícita (10) da operação de convolução para o caso 1D é dada por

$$v = \begin{bmatrix} k^\top S_1 \\ k^\top S_2 \\ \vdots \\ k^\top S_{n_v} \end{bmatrix} z + b\mathbf{1}_{n_v,1}. \tag{113}$$

Observação 14. *Nas matrizes de seleção S_i (e em outras definidas em seções posteriores), é possível que um dos sub-blocos internos apresente dimensão nula. Por exemplo, em S_i, a matriz de zeros inicial possui dimensões $n_k, i-1$, de forma que para $i = 1$ a dimensão resultante é $n_k, 0$. Quando isso ocorre, o bloco em questão é considerado uma matriz vazia e deve ser omitido da concatenação.*

6.2.2.2 Caso 2D

Agora, considerando uma entrada $Z \in \mathbb{R}^{n_{z1} \times n_{z2}}$, um filtro $K \in \mathbb{R}^{n_{k1} \times n_{k2}}$, $n_{k1} \leq n_{z1}$, $n_{k2} \leq n_{z2}$, um *bias* $b \in \mathbb{R}$ e uma saída $V \in \mathbb{R}^{n_{v1} \times n_{v2}}$ com $n_{v1} = n_{z1} - n_{k1} + 1$ e $n_{v2} = n_{z2} - n_{k2} + 1$.

Vetorizando as matrizes de entrada, filtro e saída respectivamente, tem-se $z \in \mathbb{R}^{n_z}$, $k \in \mathbb{R}^{n_k}$ e $v \in \mathbb{R}^{n_v}$, com $n_z = n_{z1} n_{z2}$, $n_k = n_{k1} n_{k2}$ e $n_v = n_{v1} n_{v2}$.

Assim, a operação de convolução pode ser expressa matematicamente por

$$v_{(i)} = \sum_{j=1}^{n_{k1}} \sum_{l=1}^{n_{k2}} k_{((j-1)n_{k2}+l)} z_{(n_{ai}+(j-1)n_{z2}+l)} + b \text{ para } i = 1, 2, \ldots, n_v, \tag{114}$$

onde n_{ai} determina a posição inicial do filtro para cada elemento da saída $v_{(i)}$, sendo definido por:

$$n_{ai} := ((i-1) \bmod n_{v2}) + \left\lfloor \frac{i-1}{n_{v2}} \right\rfloor n_{z2}, \tag{115}$$

no qual $\bmod$ denota a operação de resto da divisão e $\lfloor \cdot \rfloor$ a função piso. Especificamente, n_{ai} realiza o mapeamento espacial, traduzindo o índice da saída $v_{(i)}$ no deslocamento necessário para posicionar corretamente o filtro k sobre a entrada vetorizada z.

Desta forma, a operação de convolução para o caso 2D pode ser escrita na forma implícita (10) com

$$N_1 = \begin{bmatrix} k^\top S_1 \\ k^\top S_2 \\ \vdots \\ k^\top S_{n_v} \end{bmatrix}, \; e_0 = b\mathbf{1}_{n_v,1}, \; n_q = 0, \tag{116}$$

onde

$$S_i := \begin{bmatrix} F_{i,1} \\ F_{i,2} \\ \vdots \\ F_{i,n_{k1}} \end{bmatrix}, \; F_{i,j} := \begin{bmatrix} 0_{n_{k2},d_{i,j}} & I_{n_{k2}} & 0_{n_{k2},n_z-n_{k2}-d_{i,j}} \end{bmatrix}, \tag{117}$$

$$d_{i,j} := n_{ai} + (j-1)n_{z2}, \; i = 1, 2, \ldots, n_v \text{ e } j = 1, 2, \ldots, n_{k1}.$$

Nesta estrutura, a matriz de seleção S_i é composta por submatrizes $F_{i,j}$. Para cada elemento da saída $v_{(i)}$, cada matriz $F_{i,j}$ realiza o mapeamento de uma das n_{k1} linhas do filtro K sobre os elementos correspondentes da entrada z.

6.2.2.3 Passo

Em camadas de convolução, o filtro nem sempre se desloca apenas uma unidade por vez. Ele pode avançar mais de uma unidade, tanto na direção horizontal quanto na vertical. A distância que o filtro percorre entre duas posições consecutivas durante a varredura é chamada de passo (*stride*).

Para incorporar esse efeito na modelagem, define-se o vetor de passo $s = \begin{bmatrix} s_1 & s_2 \end{bmatrix}$, onde s_1 representa o passo na direção vertical e s_2 representa o passo na direção horizontal. Com essa definição, a representação implícita (10) de uma camada de convolução mantém a estrutura das matrizes (116) e (117), porém as dimensões da saída são reduzidas pelo passo adotado

$$n_{v1} = \left\lfloor \frac{n_{z1} - n_{k1}}{s_1} \right\rfloor + 1, \; n_{v2} = \left\lfloor \frac{n_{z2} - n_{k2}}{s_2} \right\rfloor + 1. \tag{118}$$

Consequentemente, o deslocamento inicial do filtro para cada elemento da saída, n_{ai}, também é ajustado em função do passo, sendo redefinido como:

$$n_{ai} := ((i-1) \bmod n_{v2})s_2 + \left\lfloor \frac{i-1}{n_{v2}} \right\rfloor n_{z2}s_1, \text{ para } i = 1, 2, \ldots, n_v. \tag{119}$$

Note que s_2 atua no deslocamento entre colunas adjacentes, enquanto o produto $n_{z2}s_1$ define o "salto" correspondente ao avanço de linhas na entrada original vetorizada.

Observação 15. *Quando as dimensões da entrada não permitem um número inteiro de passos do filtro, o procedimento padrão consiste em realizar a operação apenas nas regiões onde a janela do filtro esteja completamente contida na entrada. Este comportamento é refletido na operação de divisão inteira na definição das dimensões da saída* (118)*. Porém, caso se deseje que o filtro seja aplicado em toda a entrada, pode-se utilizar uma operação de preenchimento de maneira a aumentar a dimensão da entrada, esta operação será vista posteriormente.*

6.2.2.4 Exemplo

Para ilustrar a formulação proposta, considere uma camada de convolução com entrada de dimensões 4×3, vetor de passo $s = \begin{bmatrix} 2 & 1 \end{bmatrix}$, valor de bias b e um filtro

$$K = \begin{bmatrix} k_{(1)} & k_{(2)} \\ k_{(3)} & k_{(4)} \end{bmatrix}.$$

Conforme (118), as dimensões de saída são 2×2. Vetorizando as dimensões de entrada, saída e do filtro, temos $n_z = 12$, $n_v = 4$ e

$$k = \begin{bmatrix} k_{(1)} & k_{(2)} & k_{(3)} & k_{(4)} \end{bmatrix}.$$

Para calcular as matrizes de seleção, primeiro calculam-se os valores para n_{ai}

$$n_{a1} = 0,\ n_{a2} = 1,\ n_{a3} = 6 \text{ e } n_{a4} = 7.$$

Note que $n_{a2} = 1$ reflete o deslocamento horizontal unitário ($s_2 = 1$), enquanto $n_{a3} = 6$ reflete o salto de duas linhas na entrada ($s_1 = 2$ e $n_{z2} = 3$), evidenciando a aplicação do passo vertical. A partir destes valores, são calculados $d_{i,j}$ e $F_{i,j}$ correspondentes

$$\begin{aligned}
&d_{1,1} = 0,\ F_{1,1} = \begin{bmatrix} I_2 & 0_{2,10} \end{bmatrix},\ d_{1,2} = 3,\ F_{1,2} = \begin{bmatrix} 0_{2,3} & I_2 & 0_{2,7} \end{bmatrix}, \\
&d_{2,1} = 1,\ F_{2,1} = \begin{bmatrix} 0_{2,1} & I_2 & 0_{2,9} \end{bmatrix},\ d_{2,2} = 4,\ F_{2,2} = \begin{bmatrix} 0_{2,4} & I_2 & 0_{2,6} \end{bmatrix}, \\
&d_{3,1} = 6,\ F_{3,1} = \begin{bmatrix} 0_{2,6} & I_2 & 0_{2,4} \end{bmatrix},\ d_{3,2} = 9,\ F_{3,2} = \begin{bmatrix} 0_{2,9} & I_2 & 0_{2,1} \end{bmatrix}, \\
&d_{4,1} = 7,\ F_{4,1} = \begin{bmatrix} 0_{2,7} & I_2 & 0_{2,3} \end{bmatrix},\ d_{4,2} = 10,\ F_{4,2} = \begin{bmatrix} 0_{2,10} & I_2 \end{bmatrix},
\end{aligned}$$

e então, as matrizes de seleção

$$S_1 = \begin{bmatrix} F_{1,1} \\ F_{1,2} \end{bmatrix} = \left[\begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$S_2 = \begin{bmatrix} F_{2,1} \\ F_{2,2} \end{bmatrix} = \left[\begin{array}{cccccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$S_3 = \begin{bmatrix} F_{3,1} \\ F_{3,2} \end{bmatrix} = \left[\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right],$$

$$S_4 = \begin{bmatrix} F_{4,1} \\ F_{4,2} \end{bmatrix} = \left[\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Por fim, utilizando (116), obtém-se a representação implícita (10) da operação de convolução deste exemplo

$$N_1 = \left[\begin{array}{ccc:ccc:ccc:ccc} k_{(1)} & k_{(2)} & 0 & k_{(3)} & k_{(4)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{(1)} & k_{(2)} & 0 & k_{(2)} & k_{(4)} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & k_{(1)} & k_{(2)} & 0 & k_{(2)} & k_{(4)} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{(1)} & k_{(2)} & 0 & k_{(2)} & k_{(4)} \end{array}\right], e_0 = \left[\begin{array}{c} b \\ b \\ \hdashline b \\ b \end{array}\right].$$

As linhas tracejadas verticais delimitam as transições de linha na entrada vetorizada, enquanto as horizontais delimitam a mudança de linha na saída vetorizada. Observa-se que o arranjo das componentes do filtro k reflete o mapeamento espacial adequado, com destaque para a mudança de linha da saída, onde o passo vertical $s_1 = 2$ é aplicado.

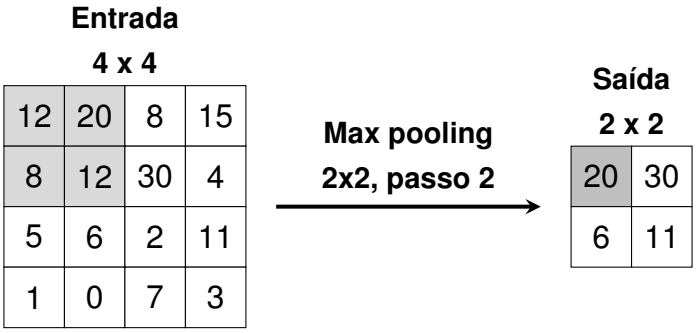
6.2.3 Agrupamento

Uma camada de agrupamento (*pooling*) atua como um mecanismo de subamostragem espacial, ela reduz a resolução dos mapas de características de forma a diminuir o custo computacional e conferir invariância a pequenas distorções. Diferentemente da camada de convolução, que utiliza filtros compostos por pesos para extrair padrões, o agrupamento emprega uma janela de operação fixa, sem parâmetros treináveis, a qual aplica uma regra estatística sobre a região coberta.

De forma análoga à camada de convolução, o agrupamento segue a lógica de janela deslizante. A distinção fundamental reside na substituição da soma ponderada por uma

operação determinística aplicada sobre os elementos da entrada, consolidando a informação local em um valor escalar representativo no mapa de saída (GOODFELLOW; BENGIO; COURVILLE, 2016). A Figura 22 apresenta um exemplo de uma camada de agrupamento na qual se utiliza o valor máximo.

Figura 22 – Exemplo de uma camada de max pooling.



Fonte: Autor

6.2.3.1 Agrupamento por Máximo

Normalmente utilizada para destacar as características mais proeminentes e contrastantes, como bordas e texturas nítidas, a camada de agrupamento por máximo (*max pooling*) atua como um seletor de ativações relevantes, descartando informações de baixa magnitude que podem ser irrelevantes para a identificação de padrões estruturais complexos. Esta camada seleciona o valor de maior intensidade dentro de sua janela deslizante, ou seja, executa uma função de máximo sobre o conjunto de variáveis de entrada contidas na janela.

6.2.3.1.1 Representação Implícita da Função Máximo

Para obter a representação implícita (10) para camadas de agrupamento por máximo, primeiro é preciso obter a representação implícita (10) da função de máximo. Uma vez estabelecida a representação da operação de máximo, esta pode ser estendida para a estrutura da camada de agrupamento por meio das matrizes de seleção.

Para isso, observa-se que uma operação de máximo entre 2 escalares pode ser representada por funções ReLU utilizando:

$$\max(z_1, z_2) = z_1 + r(z_2 - z_1) \tag{120}$$

Essa equivalência pode ser verificada considerando os dois casos possíveis:

1. $z_1 > z_2$:
   - $z_2 - z_1 < 0 \rightarrow r(z_2 - z_1) = 0$
   - Consequentemente: $\max(z_1, z_2) = z_1$

2. $z_1 \leq z_2$:

   - $z_2 - z_1 \geq 0 \rightarrow r(z_2 - z_1) = z_2 - z_1$
   - Consequentemente: $\max(z_1, z_2) = z_1 + z_2 - z_1 = z_2$

Assim, representando a operação de $\max(z_1, z_2)$ na representação implícita (10), tem-se:

$$\begin{aligned} v &= \begin{bmatrix} 1 & 0 \end{bmatrix} z + \phi(q) \\ q &= \begin{bmatrix} -1 & 1 \end{bmatrix} z. \end{aligned} \tag{121}$$

Essa operação pode ser expandida para mais variáveis. Por exemplo, para 3 variáveis tem-se:

$$\max(z_1, z_2, z_3) = z_1 + r(\max(z_2, z_3) - z_1) = z_1 + r(z_2 + r(z_3 - z_2) - z_1), \tag{122}$$

o que resulta na representação implícita

$$\begin{aligned} v &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 & 1 \end{bmatrix} \phi(q) \\ q &= \begin{bmatrix} 0 & -1 & 1 \\ -1 & 1 & 0 \end{bmatrix} z + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \phi(q). \end{aligned} \tag{123}$$

Generalizando, a operação para um vetor de entrada $z \in \mathbb{R}^{n_z}$,

$$v = \max(z), \tag{124}$$

pode ser representada na forma implícita (10), da seguinte forma:

$$\begin{aligned} v &= \begin{bmatrix} 1 & 0_{1,n_z-1} \end{bmatrix} z + \begin{bmatrix} 0_{1,n_z-2} & 1 \end{bmatrix} \phi(q) \\ q &= \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & -1 & 1 \\ 0 & 0 & 0 & \dots & -1 & 1 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & -1 & 1 & \dots & 0 & 0 & 0 \\ -1 & 1 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0_{1,n_z-2} & 0 \\ I_{n_z-2} & 0_{n_z-2,1} \end{bmatrix} \phi(q) \end{aligned} \tag{125}$$

com $v \in \mathbb{R}$ e $q \in \mathbb{R}^{n_z-1}$.

6.2.3.1.2 Caso 1D

Tendo a representação implícita da função de máximo, considera-se primeiro a representação da camada de *max pooling* para uma entrada de uma dimensão, de forma a facilitar o entendimento, assim como foi realizado para a camada de convolução.

Considerando uma entrada $z \in \mathbb{R}^{n_z}$ e uma janela de operação de tamanho n_k, $n_k \leq n_z$, a saída $v \in \mathbb{R}^{n_v}$, $n_v = n_z - n_k + 1$, da camada de *max pooling* pode ser dada por

$$v_{(i)} = \max(S_i z), \quad \text{para } i = 1, 2, \dots, n_v \tag{126}$$

onde S_i é a mesma matriz de seleção utilizada na convolução 1D,

$$S_i := \begin{bmatrix} 0_{n_k,i-1} & I_{n_k} & 0_{n_k,n_z-n_k-(i-1)} \end{bmatrix},\ i = 1, 2, \ldots, n_v. \tag{127}$$

Utilizando a representação implícita da função de máximo (125), a operação de agrupamento por máximo para o caso 1D pode ser escrita na forma implícita (10)

$$\begin{aligned} v &= \begin{bmatrix} S_{a1} \\ S_{a2} \\ \vdots \\ S_{an_v} \end{bmatrix} z + \begin{bmatrix} S_{b1} \\ S_{b2} \\ \vdots \\ S_{bn_v} \end{bmatrix} \phi(q) \\ q &= \begin{bmatrix} N_{3mp}S_1 \\ N_{3mp}S_2 \\ \vdots \\ N_{3mp}S_{n_v} \end{bmatrix} z + \begin{bmatrix} N_{4mp}S_{d1} \\ N_{4mp}S_{d2} \\ \vdots \\ N_{4mp}S_{dn_v} \end{bmatrix} \phi(q) \end{aligned} \tag{128}$$

com $n_q = (n_k - 1)n_v$ e

$$\begin{aligned} N_{3mp} &:= \begin{bmatrix} 0_{n_k-1,1} & J_{n_k-1} \end{bmatrix} - \begin{bmatrix} J_{n_k-1} & 0_{n_k-1,1} \end{bmatrix},\ N_{4mp} := \begin{bmatrix} 0_{1,n_k-2} & 0 \\ I_{n_k-2} & 0_{n_k-2,1} \end{bmatrix}, \\ S_{ai} &:= \begin{bmatrix} 0_{1,i-1} & 1 & 0_{1,n_z-i} \end{bmatrix},\ S_{bi} := \begin{bmatrix} 0_{1,(n_k-1)i-1} & 1 & 0_{1,(n_k-1)(n_v-i)} \end{bmatrix}, \\ S_{di} &:= \begin{bmatrix} 0_{n_k-1,(i-1)(n_k-1)} & I_{n_k-1} & 0_{n_k-1,(n_k-1)(n_v-i)} \end{bmatrix}, \end{aligned} \tag{129}$$

onde J_n representa a matriz anti-identidade de dimensão n, como por exemplo

$$J_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Aqui N_{3mp} e N_{4mp} correspondem às matrizes N_3 e N_4 da representação implícita da função de máximo (125). Já as matrizes S_{ai} e S_{bi} são responsáveis, respectivamente, por selecionar os termos de z e $\phi(q)$ para o cálculo de v, enquanto S_i e S_{di} selecionam os termos de z e $\phi(q)$ para o cálculo de q.

6.2.3.1.3 Caso 2D

Agora, considerando uma entrada $Z \in \mathbb{R}^{n_{z1} \times n_{z2}}$, uma janela de operação de tamanho $n_{k1} \times n_{k2}$, $n_{k1} \leq n_{z1}$, $n_{k2} \leq n_{z2}$, e uma saída $V \in \mathbb{R}^{n_{v1} \times n_{v2}}$ com $n_{v1} = n_{z1} - n_{k1} + 1$ e $n_{v2} = n_{z2} - n_{k2} + 1$.

Vetorizando as matrizes de entrada e saída, tem-se $z \in \mathbb{R}^{n_z}$, $v \in \mathbb{R}^{n_v}$ e uma janela de operação de tamanho n_k com $n_z = n_{z1}n_{z2}$, $n_k = n_{k1}n_{k2}$ e $n_v = n_{v1}n_{v2}$. Desta forma, a operação de agrupamento por máximo pode ser dada por

$$v_{(i)} = \max(S_i z), \tag{130}$$

onde S_i é a mesma matriz de seleção utilizada na convolução 2D

$$S_i := \begin{bmatrix} F_{i,1} \\ F_{i,2} \\ \vdots \\ F_{i,n_{k1}} \end{bmatrix},\ F_{i,j} := \begin{bmatrix} 0_{n_{k2},d_{i,j}} & I_{n_{k2}} & 0_{n_{k2},n_z-n_{k2}-d_{i,j}} \end{bmatrix}, \tag{131}$$
$$d_{i,j} := n_{ai} + (j-1)n_{z2},\ i = 1,2,\ldots,n_v \text{ e } j = 1,2,\ldots,n_{k1},$$

onde

$$n_{ai} := ((i-1) \bmod n_{v2}) + \left\lfloor \frac{i-1}{n_{v2}} \right\rfloor n_{z2},\ \text{para } i = 1,2,\ldots,n_v.$$

Novamente, utilizando a representação implícita da função de máximo (125), a operação de agrupamento por máximo para o caso 2D pode ser escrita na forma implícita (10) com

$$N_1 = \begin{bmatrix} S_{a1} \\ S_{a2} \\ \vdots \\ S_{an_v} \end{bmatrix},\ N_2 = \begin{bmatrix} S_{b1} \\ S_{b2} \\ \vdots \\ S_{bn_v} \end{bmatrix}, e_0 = 0_{n_v,1}$$
$$N_3 = \begin{bmatrix} N_{3mp}S_1 \\ N_{3mp}S_2 \\ \vdots \\ N_{3mp}S_{n_v} \end{bmatrix},\ N_4 = \begin{bmatrix} N_{4mp}S_{d1} \\ N_{4mp}S_{d2} \\ \vdots \\ N_{4mp}S_{dn_v} \end{bmatrix}, e_1 = 0_{n_q,1}, \tag{132}$$

onde $n_q = (n_k-1)n_v$ e

$$N_{3mp} := \begin{bmatrix} 0_{n_k-1,1} & J_{n_k-1} \end{bmatrix} - \begin{bmatrix} J_{n_k-1} & 0_{n_k-1,1} \end{bmatrix},\ N_{4mp} := \begin{bmatrix} 0_{1,n_k-2} & 0 \\ I_{n_k-2} & 0_{n_k-2,1} \end{bmatrix},$$
$$S_{ai} := \begin{bmatrix} 0_{1,n_{ai}} & 1 & 0_{1,n_z-n_{ai}-1} \end{bmatrix},\ S_{bi} := \begin{bmatrix} 0_{1,(n_k-1)i-1} & 1 & 0_{1,(n_k-1)(n_v-i)} \end{bmatrix}, \tag{133}$$
$$S_{di} := \begin{bmatrix} 0_{n_k-1,(i-1)(n_k-1)} & I_{n_k-1} & 0_{n_k-1,(n_k-1)(n_v-i)} \end{bmatrix}.$$

6.2.3.1.4 Passo

Da mesma forma que as camadas de convolução, as camadas de agrupamento também possuem um parâmetro de passo que define a distância que a janela de operação percorre entre duas posições consecutivas durante a varredura da entrada. Usualmente, o passo em uma camada de agrupamento tem o mesmo tamanho que a janela de operação.

Considerando novamente um passo $s = \begin{bmatrix} s_1 & s_2 \end{bmatrix}$, onde s_1 representa o passo na direção vertical e s_2 representa o passo na direção horizontal, a representação implícita (10) de uma camada de agrupamento por máximo mantém a estrutura das matrizes (132) e (133), com os mesmos ajustes realizados em (118) para as dimensões da saída

$$n_{v1} = \left\lfloor \frac{n_{z1}-n_{k1}}{s_1} \right\rfloor + 1,\ n_{v2} = \left\lfloor \frac{n_{z2}-n_{k2}}{s_2} \right\rfloor + 1, \tag{134}$$

e realizados em (119) para o deslocamento inicial do filtro

$$n_{ai} := ((i-1) \bmod n_{v2})s_2 + \left\lfloor \frac{i-1}{n_{v2}} \right\rfloor n_{z2}s_1, \text{ para } i = 1, 2, \ldots, n_v. \tag{135}$$

6.2.3.2 Agrupamento por Média

Normalmente utilizada para capturar informações de preenchimento e de contexto, a camada de agrupamento por média (*average pooling*) atua como um filtro passa-baixa, suavizando o mapa de características e reduzindo o impacto de ruídos ou ativações isoladas. Esta camada calcula a média aritmética de todos os valores dentro de sua janela deslizante.

Considerando uma entrada $Z \in \mathbb{R}^{n_{z1} \times n_{z2}}$, uma janela de operação de tamanho $n_{k1} \times n_{k2}$, $n_{k1} \leq n_{z1}$, $n_{k2} \leq n_{z2}$, um passo $s = \begin{bmatrix} s_1 & s_2 \end{bmatrix}$, e uma saída $V \in \mathbb{R}^{n_{v1} \times n_{v2}}$ com

$$n_{v1} = \left\lfloor \frac{n_{z1} - n_{k1}}{s_1} \right\rfloor + 1, \text{ e } n_{v2} = \left\lfloor \frac{n_{z2} - n_{k2}}{s_2} \right\rfloor + 1.$$

Vetorizando as matrizes de entrada e saída, obtém-se, $z \in \mathbb{R}^{n_z}$, uma janela de operação de tamanho n_k e uma saída $v \in \mathbb{R}^{n_v}$, $n_z = n_{z1}n_{z2}$, $n_k = n_{k1}n_{k2}$, $n_v = n_{v1}n_{v2}$. Desta forma, a operação de *average pooling* pode ser dada por

$$v_{(i)} = \frac{1}{n_k} \sum_{l=1}^{n_{k1}} \sum_{j=1}^{n_{k2}} z_{(n_{ai}+j+(l-1)n_{z2})} \tag{136}$$

com

$$n_{ai} = ((i-1) \bmod n_{v2})s_2 + \left\lfloor \frac{i-1}{n_{v2}} \right\rfloor n_{z2}s_1, \text{ para } i = 1, 2, \ldots, n_v$$

Assim, pode-se observar que a representação implícita baseada em funções ReLU de uma camada de *average pooling* é um caso particular da representação da convolução com $K = \mathbf{1}_{n_{k1},n_{k2}}/n_k$.

6.2.3.3 Exemplo

Para ilustrar a formulação proposta, considere uma camada de *max pooling* com entrada de dimensões 4×4, vetor de passo $s = \begin{bmatrix} 2 & 2 \end{bmatrix}$ e uma janela de operação de dimensões 2×2.

Conforme (134), as dimensões de saída são 2×2. Vetorizando as dimensões de entrada, saída e do filtro, temos $n_z = 16$, $n_v = 4$ e $n_k = 4$.

Primeiramente, obtêm-se as matrizes N_{3mp} e N_{4mp} relativas a operação de máximo para uma entrada de tamanho n_k

$$N_{3mp} = \begin{bmatrix} 0 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix}, \; N_{4mp} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Então calculam-se os valores para n_{ai}

$$n_{a1} = 0,\ n_{a2} = 2,\ n_{a3} = 8 \text{ e } n_{a4} = 10.$$

A partir destes valores, são calculados $d_{i,j}$ e $F_{i,j}$ correspondentes

$$\begin{aligned}
&d_{1,1} = 0,\ F_{1,1} = \begin{bmatrix} I_2 & 0_{2,14} \end{bmatrix},\ d_{1,2} = 4,\ F_{1,2} = \begin{bmatrix} 0_{2,4} & I_2 & 0_{2,10} \end{bmatrix},\\
&d_{2,1} = 2,\ F_{2,1} = \begin{bmatrix} 0_{2,2} & I_2 & 0_{2,12} \end{bmatrix},\ d_{2,2} = 6,\ F_{2,2} = \begin{bmatrix} 0_{2,6} & I_2 & 0_{2,8} \end{bmatrix},\\
&d_{3,1} = 8,\ F_{3,1} = \begin{bmatrix} 0_{2,8} & I_2 & 0_{2,6} \end{bmatrix},\ d_{3,2} = 12,\ F_{3,2} = \begin{bmatrix} 0_{2,12} & I_2 & 0_{2,2} \end{bmatrix},\\
&d_{4,1} = 10,\ F_{4,1} = \begin{bmatrix} 0_{2,10} & I_2 & 0_{2,4} \end{bmatrix},\ d_{4,2} = 14,\ F_{4,2} = \begin{bmatrix} 0_{2,14} & I_2 \end{bmatrix},
\end{aligned}$$

e por fim, as matrizes de seleção:

$$\begin{aligned}
&S_{a1} = \begin{bmatrix} 1 & 0_{1,15} \end{bmatrix},\ S_{b1} = \begin{bmatrix} 0_{1,2} & 1 & 0_{1,9} \end{bmatrix},\ S_{c1} = \begin{bmatrix} F_{1,1} \\ F_{1,2} \end{bmatrix},\ S_{d1} = \begin{bmatrix} I_3 & 0_{3,9} \end{bmatrix},\\
&S_{a2} = \begin{bmatrix} 0_{1,2} & 1 & 0_{1,13} \end{bmatrix},\ S_{b2} = \begin{bmatrix} 0_{1,5} & 1 & 0_{1,6} \end{bmatrix},\ S_{c2} = \begin{bmatrix} F_{2,1} \\ F_{2,2} \end{bmatrix},\ S_{d2} = \begin{bmatrix} 0_{3,3} & I_3 & 0_{3,6} \end{bmatrix},\\
&S_{a3} = \begin{bmatrix} 0_{1,8} & 1 & 0_{1,7} \end{bmatrix},\ S_{b3} = \begin{bmatrix} 0_{1,8} & 1 & 0_{1,3} \end{bmatrix},\ S_{c3} = \begin{bmatrix} F_{3,1} \\ F_{3,2} \end{bmatrix},\ S_{d3} = \begin{bmatrix} 0_{3,6} & I_3 & 0_{3,3} \end{bmatrix},\\
&S_{a4} = \begin{bmatrix} 0_{1,10} & 1 & 0_{1,5} \end{bmatrix},\ S_{b4} = \begin{bmatrix} 0_{1,11} & 1 \end{bmatrix},\ S_{c4} = \begin{bmatrix} F_{4,1} \\ F_{4,2} \end{bmatrix},\ S_{d4} = \begin{bmatrix} 0_{3,9} & I_3 \end{bmatrix},
\end{aligned}$$

Por fim, utilizando (132), obtém-se a representação implícita (10) da operação de *max pooling* deste exemplo

$$N_1 = \left[\begin{array}{cccc:cccc:cccc:cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}\right],$$

$$N_2 = \left[\begin{array}{ccc:ccc:ccc:ccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right],$$

$$N_3 = \left[\begin{array}{cccc:cccc:cccc:cccc}
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0
\end{array}\right],$$

$$N_4 = \left[\begin{array}{ccc:ccc:ccc:ccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}\right], \; e_0 = 0_{4,1}, \; e_1 = 0_{12,1}.$$

Pode-se observar que o mapeamento espacial da janela de operação é corretamente definido, de forma que as matrizes N_1, N_2, N_3 e N_4 selecionam os valores corretos de z e $\phi(q)$ para o cálculo das saídas da operação de *max pooling*.

6.2.4 Ativação

A camada de ativação simplesmente aplica sua respectiva função de ativação, aqui considerada uma função ReLU, em todas as suas entradas. A Figura 23 apresenta um exemplo de uma camada de ativação ReLU.

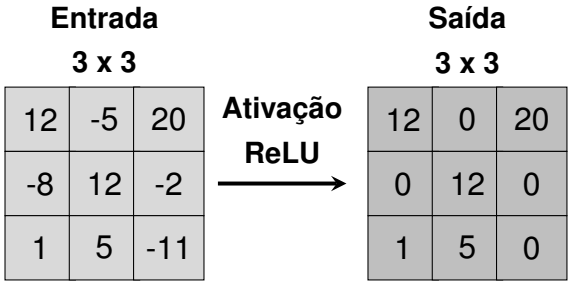
Considerando como entrada e saída as matrizes $Z \in \mathbb{R}^{n_{z1} \times n_{z2}}$ e $V \in \mathbb{R}^{n_{z1} \times n_{z2}}$ respectivamente. Suas versões vetorizadas são a entrada $z \in \mathbb{R}^{n_z}$ e a saída $v \in \mathbb{R}^{n_z}$, com $n_z = n_{z1} n_{z2}$. Assim uma camada de ativação pode ser dada por

$$v = \phi(z), \tag{137}$$

ou na forma implícita (10) com

$$N_1 = 0_{n_z}, \; N_2 = I_{n_z}, \; e_0 = 0_{n_z,1}, \; N_3 = I_{n_z}, \; N_4 = 0_{n_z}, \; e_1 = 0_{n_z,1}. \tag{138}$$

Figura 23 – Exemplo de uma camada de ativação ReLU.



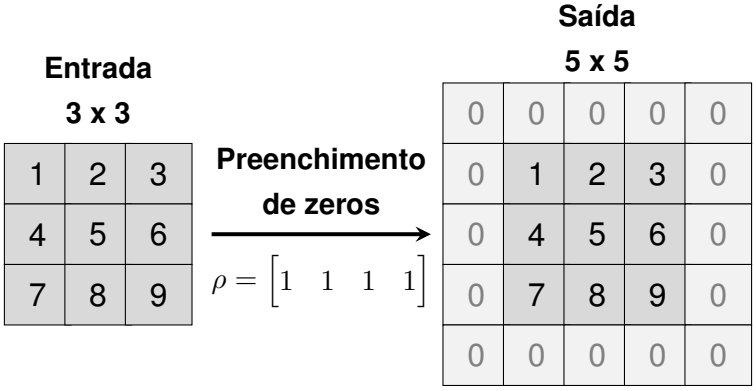
Fonte: Autor

6.2.5 Preenchimento

Normalmente, a operação de preenchimento (*padding*) não é considerada uma camada em si, mas sim uma operação de pré-processamento aplicada à entrada de uma camada. O objetivo principal desta operação é adequar as dimensões dos dados de entrada para aumentar a dimensão da saída de uma operação de convolução ou *pooling*, ou acomodar configurações de filtro e passo cujas dimensões não sejam submúltiplas das dimensões da entrada.

Neste trabalho, escolheu-se tratar a operação de preenchimento como uma camada, a qual pode simplesmente ser adicionada antes da operação desejada utilizando a interconexão em série (20). Comumente, na operação de preenchimento, são adicionados zeros ou uns a matriz de entrada, mas pode-se realizar a operação de preenchimento com qualquer valor. A Figura 24 apresenta um exemplo de uma operação de preenchimento.

Figura 24 – Exemplo de uma operação de preenchimento.



Fonte: Autor

Para uma entrada $Z \in \mathbb{R}^{n_{z1} \times n_{z2}}$, define-se o vetor de preenchimento como:

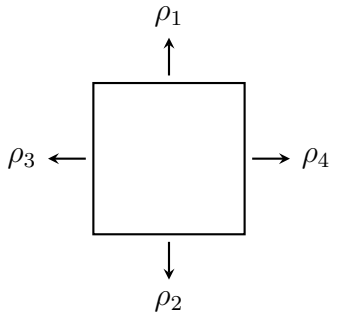
$$\rho = \begin{bmatrix} \rho_1 & \rho_2 & \rho_3 & \rho_4 \end{bmatrix}, \tag{139}$$

onde os índices de ρ representam a quantidade de preenchimento aplicada nas direções superior, inferior, direita e esquerda, respectivamente, conforme ilustrado na Figura 25.

Assim, as dimensões da saída $V \in \mathbb{R}^{n_{v1} \times n_{v2}}$ são dadas por

$$n_{v1} = n_{z1} + \rho_1 + \rho_2 \quad \text{e} \quad n_{v2} = n_{z2} + \rho_3 + \rho_4. \tag{140}$$

Figura 25 – Definição dos parâmetros de preenchimento ρ.



Fonte: Autor

Considerando como entrada a matriz vetorizada $z \in \mathbb{R}^{n_z}$, $n_z = n_{z1}n_{z2}$, um valor de preenchimento c e vetor de preenchimento ρ. A saída vetorizada da operação de preenchimento $v \in \mathbb{R}^{n_v}$, $n_v = n_{v1}n_{v2}$ pode ser dada por

$$v = (P_h \otimes P_w)z + c(Y_o - N_1 Y_i) \tag{141}$$

com

$$P_h := \begin{bmatrix} 0_{n_{z1},\rho_1} \\ I_{n_{z1}} \\ 0_{n_{z1},\rho_2} \end{bmatrix},\ P_w := \begin{bmatrix} 0_{n_{z2},\rho_3} \\ I_{n_{z2}} \\ 0_{n_{z2},\rho_4} \end{bmatrix},\ Y_i := \mathbf{1}_{n_z,1},\ Y_o := \mathbf{1}_{n_v,1}. \tag{142}$$

Ou na forma implícita (10), com

$$N_1 = P_h \otimes P_w,\ e_0 = c(Y_o - N_1 Y_i),\ n_q = 0. \tag{143}$$

6.2.5.1 Exemplo

Para ilustrar a formulação proposta, considere uma entrada de dimensões 3×2, um vetor de preenchimento $\rho = \begin{bmatrix} 1 & 1 & 2 & 0 \end{bmatrix}$ e um valor de preenchimento c, de forma que a saída tem dimensões 5×4.

Primeiramente calcula-se as matrizes P_h e P_w e os vetores Y_i e Y_o,

$$P_h = \begin{bmatrix} 0_{3,1} \\ I_3 \\ 0_{3,1} \end{bmatrix},\ P_w = \begin{bmatrix} 0_{2,2} \\ I_2 \end{bmatrix},\ Y_i = \mathbf{1}_{6,1},\ Y_o = \mathbf{1}_{20,1}.$$

Por fim, representa-se a operação na forma implícita (10) com

$$N_1 = \left[\begin{array}{cc:cc:cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \text{ e } e_0 = \left[\begin{array}{c} c \\ c \\ c \\ c \\ \hdashline c \\ c \\ 0 \\ 0 \\ \hdashline c \\ c \\ 0 \\ 0 \\ \hdashline c \\ c \\ 0 \\ 0 \\ \hdashline c \\ c \\ c \\ c \end{array}\right]$$

As linhas tracejadas verticais delimitam as transições de linha na entrada vetorizada, enquanto as horizontais delimitam a mudança de linha na saída vetorizada. Observe que a primeira e a última linha serão completamente preenchidas com o valor c, respeitando $\rho_1 = \rho_2 = 1$. Adicionalmente, as duas primeiras colunas das linhas 2 até 4 serão preenchidas com c, enquanto as duas últimas serão preenchidas com os valores da entrada, respeitando $\rho_3 = 2$ e $\rho_4 = 0$.

6.3 Outras Funções de Ativação

Uma vez que os resultados apresentados neste trabalho foram desenvolvidos especificamente para funções de ativação do tipo ReLU, obtém-se como principal vantagem a possibilidade de uma representação exata dessa não linearidade, o que garante resultados menos conservadores. Por outro lado, essa especialização acarreta na limitação de que os resultados aqui obtidos não podem ser diretamente aplicados a RNs que empreguem outros tipos de funções de ativação.

Neste capítulo, busca-se contornar essa desvantagem, demonstrando que é possível representar, ou em alguns casos aproximar, outras funções de ativação a partir da soma de funções ReLU, de forma a possibilitar o uso dos resultados deste trabalho.

6.3.1 Representação Implícita ReLU de RN com Função de Ativação Genérica

Para obter a representação implícita (10) de uma RN com função de ativação genérica θ, fundamenta-se na propriedade de que RNs ReLU *feedforward* atuam como aproximadores universais. Inicialmente, define-se a aproximação da função de ativação escalar desejada $\theta_e : \mathbb{R} \to \mathbb{R}$ na forma implícita (10) como:

$$\theta_e(z_{(i)}) = N_{1\theta_e} z_{(i)} + N_{2\theta_e}\phi(q_{\theta_e}) + e_{0\theta_e} \tag{144a}$$

$$q_{\theta_e} = N_{3\theta_e} z_{(i)} + N_{4\theta_e}\phi(q_{\theta_e}) + e_{1\theta_e}. \tag{144b}$$

Também se define a estrutura da RN que utiliza a função de ativação $\theta : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ na forma implícita como $f_\theta : \mathbb{R}^{n_z} \to \mathbb{R}^{n_v}$,

$$f_\theta(z) = N_{1f_\theta} z + N_{2f_\theta}\theta(y) + e_{0f_\theta} \tag{145a}$$

$$y = N_{3f_\theta} z + N_{4f_\theta}\theta(y) + e_{1f_\theta}, \tag{145b}$$

com $N_{1f_\theta} \in \mathbb{R}^{n_v \times n_z}$, $N_{2f_\theta} \in \mathbb{R}^{n_v \times n_y}$, $N_{3f_\theta} \in \mathbb{R}^{n_y \times n_z}$, $N_{4f_\theta} \in \mathbb{R}^{n_y \times n_y}$, $e_{0f_\theta} \in \mathbb{R}^{n_v}$, e $e_{1f_\theta} \in \mathbb{R}^{n_y}$.

Observação 16. *As representações implícitas para as diferentes arquiteturas de RNs utilizando uma função de ativação genérica θ podem ser obtidas utilizando o exposto no capítulo 2 e nas seções 6.1 e 6.2 considerando $\phi(\cdot) = \theta(\cdot)$.*

A partir dessas duas representações, é possível obter a representação implícita baseada em funções ReLU (10) da RN com função de ativação θ em dois passos.

Primeiro, deve-se escalar a representação da função de ativação θ_e para o tamanho da entrada da função de ativação da RN utilizada $y \in \mathbb{R}^{n_y}$, ou seja

$$\begin{aligned}
\theta(y) &= \begin{bmatrix} \theta_e(y_{(1)}) \\ \theta_e(y_{(2)}) \\ \vdots \\ \theta_e(y_{(n_y)}) \end{bmatrix} = \begin{bmatrix} N_{1\theta_e} y_{(1)} + N_{2\theta_e}\phi(q_{\theta_e}(y_{(1)})) + e_{0\theta_e} \\ N_{1\theta_e} y_{(2)} + N_{2\theta_e}\phi(q_{\theta_e}(y_{(2)})) + e_{0\theta_e} \\ \vdots \\ N_{1\theta_e} y_{(n_y)} + N_{2\theta_e}\phi(q_{\theta_e}(y_{(n_y)})) + e_{0\theta_e} \end{bmatrix} \\
q_\theta(y) &= \begin{bmatrix} q_{\theta_e}(y_{(1)}) \\ q_{\theta_e}(y_{(2)}) \\ \vdots \\ q_{\theta_e}(y_{(n_y)}) \end{bmatrix} = \begin{bmatrix} N_{3\theta_e} y_{(1)} + N_{4\theta_e}\phi(q_{\theta_e}(y_{(1)})) + e_{1\theta_e} \\ N_{3\theta_e} y_{(2)} + N_{4\theta_e}\phi(q_{\theta_e}(y_{(2)})) + e_{1\theta_e} \\ \vdots \\ N_{3\theta_e} y_{(n_y)} + N_{4\theta_e}\phi(q_{\theta_e}(y_{(n_y)})) + e_{1\theta_e} \end{bmatrix}.
\end{aligned} \tag{146}$$

Colocando na forma implícita (10),

$$\begin{aligned}
\theta(y) &= \underbrace{\begin{bmatrix} N_{1\theta_e} & 0 & \dots & 0 \\ 0 & N_{1\theta_e} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & N_{1\theta_e} \end{bmatrix}}_{N_{1\theta}} y + \underbrace{\begin{bmatrix} N_{2\theta_e} & 0 & \dots & 0 \\ 0 & N_{2\theta_e} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & N_{2\theta_e} \end{bmatrix}}_{N_{2\theta}} \phi(q_\theta) + \underbrace{\begin{bmatrix} e_{0\theta_e} \\ e_{0\theta_e} \\ \vdots \\ e_{0\theta_e} \end{bmatrix}}_{e_{0\theta}} \\
q_\theta &= \underbrace{\begin{bmatrix} N_{3\theta_e} & 0 & \dots & 0 \\ 0 & N_{3\theta_e} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & N_{3\theta_e} \end{bmatrix}}_{N_{3\theta}} y + \underbrace{\begin{bmatrix} N_{4\theta_e} & 0 & \dots & 0 \\ 0 & N_{4\theta_e} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & N_{4\theta_e} \end{bmatrix}}_{N_{4\theta}} \phi(q_\theta) + \underbrace{\begin{bmatrix} e_{1\theta_e} \\ e_{1\theta_e} \\ \vdots \\ e_{1\theta_e} \end{bmatrix}}_{e_{1\theta}}.
\end{aligned} \tag{147}$$

Assim, para uma RN cuja entrada da função de ativação vetorial é $y \in \mathbb{R}^{n_y}$, a representação implícita (10) de sua função de ativação vetorial θ pode ser dada por:

$$\theta(y) = N_{1\theta} y + N_{2\theta} \phi(q_\theta) + e_{0\theta} \tag{148a}$$

$$q_\theta = N_{3\theta} y + N_{4\theta} \phi(q_\theta) + e_{1\theta}, \tag{148b}$$

com suas matrizes e vetores calculados a partir da representação implícita da função de ativação escalar θ_e (144):

$$\begin{aligned}
&N_{1\theta} = I_{n_y} \otimes N_{1\theta_e},\ N_{2\theta} = I_{n_y} \otimes N_{2\theta_e},\ e_{0\theta} = \mathbf{1}_{n_y,1} \otimes e_{0\theta_e}, \\
&N_{3\theta} = I_{n_y} \otimes N_{3\theta_e},\ N_{4\theta} = I_{n_y} \otimes N_{4\theta_e},\ e_{1\theta} = \mathbf{1}_{n_y,1} \otimes e_{1\theta_e}.
\end{aligned} \tag{149}$$

Depois, deve-se incluir a representação implícita da função de ativação (148) no sistema desejado (145). Para isso, primeiro substitui-se (148a) em (145b), resultando em

$$y = N_{3f_\theta} z + N_{4f_\theta}(N_{1\theta} y + N_{2\theta} \phi(q_\theta) + e_{0\theta}) + e_{1f_\theta}. \tag{150}$$

Após o rearranjo dos termos, a variável y é isolada da seguinte forma:

$$(I_{n_y} - N_{4f_\theta} N_{1\theta}) y = N_{3f_\theta} z + N_{4f_\theta} N_{2\theta} \phi(q_\theta) + N_{4f_\theta} e_{0\theta} + e_{1f_\theta}. \tag{151}$$

Então, definindo

$$N_y = (I_{n_y} - N_{4f_\theta} N_{1\theta})^{-1}, \tag{152}$$

e pré-multiplicando ambos os lados por N_y, chega-se a:

$$y = N_y N_{3f_\theta} z + N_y N_{4f_\theta} N_{2\theta} \phi(q_\theta) + N_y (N_{4f_\theta} e_{0\theta} + e_{1f_\theta}). \tag{153}$$

Subsequentemente, ao inserir (153) em (148b) e agrupar os termos remanescentes, tem-se:

$$q_\theta = N_{3\theta} N_y N_{3f_\theta} z + (N_{4\theta} + N_{3\theta} N_y N_{4f_\theta} N_{2\theta}) \phi(q_\theta) + N_{3\theta} N_y (N_{4f_\theta} e_{0\theta} + e_{1f_\theta}) + e_{1\theta}. \tag{154}$$

Por fim, substituindo (153) e (148a) em (145a) e agrupando os termos se obtêm:

$$\begin{aligned} f_\theta(z) = (N_{1f_\theta} + N_{2f_\theta}N_{1\theta}N_yN_{3f_\theta})z + N_{2f_\theta}(N_{2\theta} + N_{1\theta}N_yN_{4f_\theta}N_{2\theta})\phi(q_\theta) \\ + N_{2f_\theta}(N_{1\theta}N_y(N_{4f_\theta}e_{0\theta} + e_{1f_\theta}) + e_{0\theta}) + e_{0f_\theta} \end{aligned} \tag{155}$$

Então, assumindo que o sistema implícito é bem posto e que a matriz de acoplamento $(I_{n_y} - N_{4f_\theta}N_{1\theta})$ é não-singular, a representação implícita baseada em funções ReLU da RN com função de ativação θ pode ser escrita na forma (10) com:

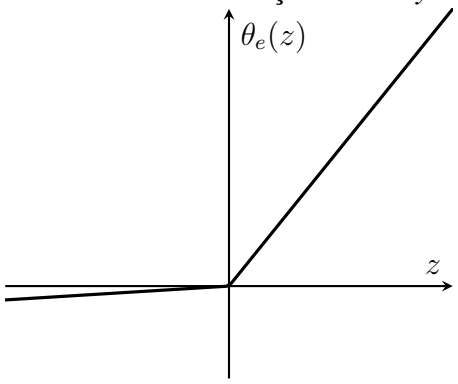
$$\begin{aligned} &N_1 = N_{1f_\theta} + N_{2f_\theta}N_{1\theta}N_yN_{3f_\theta},\ N_2 = N_{2f_\theta}(N_{1\theta}N_yN_{4f_\theta}N_{2\theta} + N_{2\theta}), \\ &e_0 = e_{0f_\theta} + N_{2f_\theta}(N_{1\theta}N_y(N_{4f_\theta}e_{0\theta} + e_{1f_\theta}) + e_{0\theta}),\ q = q_\theta, \\ &N_3 = N_{3\theta}N_yN_{3f_\theta},\ N_4 = N_{4\theta} + N_{3\theta}N_yN_{4f_\theta}N_{2\theta}, \\ &e_1 = e_{1\theta} + N_{3\theta}N_y(N_{4f_\theta}e_{0\theta} + e_{1f_\theta}). \end{aligned} \tag{156}$$

Observação 17. *Este mesmo processo pode ser realizado de forma análoga para RNRs.*

6.3.2 *Leaky* ReLU e *Parametric* ReLU

A *leaky* ReLU é uma variante da função ReLU que surge como uma solução para o problema dos "neurônios mortos" (*dying* ReLU), que ocorre quando uma grande parte da rede para de aprender por estar presa em valores negativos, onde o gradiente é zero. Diferente da ReLU comum, que descarta totalmente as entradas negativas, a *leaky* ReLU permite que uma pequena fração do sinal, controlada pelo parâmetro α "vaze" para o lado negativo. Esse pequeno ajuste garante que os gradientes continuem fluindo durante o treinamento, permitindo que os neurônios se recuperem mesmo após ativações negativas (MAAS; HANNUN; NG, 2013). Na prática, ela é amplamente utilizada em arquiteturas de redes profundas, como em RN Generativas (RADFORD; METZ; CHINTALA, 2016) e modelos de visão computacional (REDMON *et al.*, 2016). Uma ilustração da função *leaky* ReLU, representada por $\theta(z)$ pode ser vista na Figura 26.

Figura 26 – Gráfico da função de *leaky* ReLU.



Fonte: Autor

Matematicamente, a função *leaky* ReLU pode ser escrita como

$$\theta_e(z_{(i)}) = \begin{cases} z_{(i)} & \text{se } z_{(i)} \geq 0 \\ -\alpha z_{(i)} & \text{se } z_{(i)} < 0, \end{cases} \tag{157}$$

onde $z \in \mathbb{R}^{n_{z_\theta}}$ e $\alpha \in \mathbb{R}$. Usualmente $\alpha = 0.1$. A qual pode ser representada a partir da soma de função ReLU,

$$\theta_e(z_{(i)}) = r(z_{(i)}) - \alpha r(-z_{(i)}). \tag{158}$$

Ou então na forma implícita (144) com

$$\begin{aligned} &N_{1\theta_e} = 0,\ N_{2\theta_e} = \begin{bmatrix} 1 & -\alpha \end{bmatrix},\ e_{0\theta_e} = 0 \\ &N_{3\theta_e} = \begin{bmatrix} 1 \\ -1 \end{bmatrix},\ N_{4\theta_e} = 0_2,\ e_{1\theta_e} = 0_{2,1}. \end{aligned} \tag{159}$$

Desta forma, seja uma RN com função de ativação *leaky* ReLU f_{LReLU} dada por

$$f_{\text{LReLU}}(z) = N_{1f_\theta} z + N_{2f_\theta}\theta(y) + e_{0f_\theta} \tag{160a}$$

$$y = N_{3f_\theta} z + N_{4f_\theta}\theta(y) + e_{1\theta}, \tag{160b}$$

com $N_{1f_\theta} \in \mathbb{R}^{n_v \times n_z}$, $N_{2f_\theta} \in \mathbb{R}^{n_v \times n_y}$, $N_{3f_\theta} \in \mathbb{R}^{n_y \times n_z}$, $N_{4f_\theta} \in \mathbb{R}^{n_y \times n_y}$, $e_{0f_\theta} \in \mathbb{R}^{n_v}$, e $e_{1f_\theta} \in \mathbb{R}^{n_y}$. A representação implícita baseada em funções ReLU desta RN pode ser dada por pode ser dada por (10) com

$$\begin{aligned} &N_1 = N_{1f_\theta},\ N_2 = N_{2f_\theta} N_{2\theta},\ e_0 = e_{0f_\theta},\ q = q_\theta, \\ &N_3 = N_{3\theta} N_{3f_\theta},\ N_4 = N_{3\theta} N_{4f_\theta} N_{2\theta},\ e_1 = N_{3\theta} e_{1f_\theta}. \end{aligned} \tag{161}$$

Uma extensão da *leaky* ReLU é a *parametric* ReLU, a qual transforma o coeficiente α em um parâmetro treinável. Em vez de definir um valor fixo e arbitrário para a inclinação, a PReLU permite que o modelo determine, através do treinamento da RN, qual a proporção ideal de sinal que deve "vazar" em cada camada (HE *et al.*, 2015). A função *parametric* ReLU pode ser representada como uma soma de funções ReLU da mesma forma que a *leaky* ReLU tendo em vista que a única diferença entre elas é como o valor do parâmetro α é definido.

6.3.3 Tangente Hiperbólica

A Função tangente hiperbólica (Tanh) é uma evolução da função sigmoide que mapeia os dados para o intervalo entre -1 e 1, sendo definida por

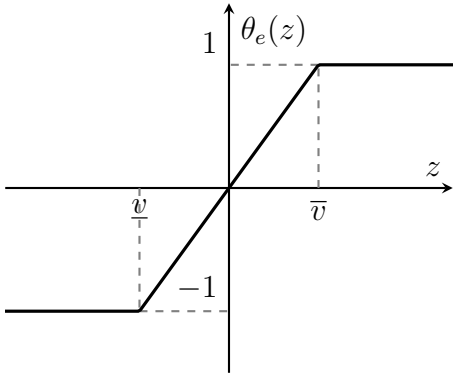
$$f(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}. \tag{162}$$

Sua principal vantagem é ser centrada em zero, o que facilita a convergência durante o treinamento ao manter a média das ativações próxima de zero, evitando deslocamentos

sistemáticos nos pesos (LECUN *et al.*, 1998). Devido a isso, esta função é muito utilizada em RNRs.

Uma aproximação comum para a função Tanh é a *hard* Tanh, uma função de saturação com limites em 1 e -1 que permite o ajuste de um ganho na sua região central. Ela substitui operações exponenciais que podem ser custosas por um simples truncamento de valores (COLLOBERT, 2004). Uma ilustração da função *hard* Tanh pode ser vista na Figura 27.

Figura 27 – Gráfico da função de hard Tanh.



Fonte: Autor

Matematicamente, a função *hard* Tanh pode ser escrita como

$$\theta_e(z_{(i)}) = \begin{cases} -1 & \text{se } z_{(i)} \leq \underline{v} \\ \frac{2(z_{(i)}-\underline{v})}{\overline{v}-\underline{v}} - 1 & \text{se } \underline{v} < z_{(i)} < \overline{v} \\ 1 & \text{se } z_{(i)} \geq \overline{v}, \end{cases} \tag{163}$$

onde $z \in \mathbb{R}^{n_{z_\theta}}$, $\overline{v} \in \mathbb{R}$, $\underline{v} \in \mathbb{R}$ e $\overline{v} > \underline{v}$. Usualmente $\overline{v} = 1$ e $\underline{v} = -1$. A função *hard* Tanh pode ser facilmente representada a partir da soma de função ReLU,

$$\theta(z_{(i)}) = \frac{2r(z_{(i)} - \underline{v}) - 2r(z_{(i)} - \overline{v})}{\overline{v} - \underline{v}} - 1. \tag{164}$$

Ou então na forma implícita (144) com

$$\begin{aligned} &N_{1\theta_e} = 0,\ N_{2\theta_e} = \begin{bmatrix} -\frac{2}{\overline{v}-\underline{v}} & \frac{2}{\overline{v}-\underline{v}} \end{bmatrix},\ e_{0\theta_e} = -1 \\ &N_{3\theta_e} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ N_{4\theta_e} = 0_2,\ e_{1\theta_e} = \begin{bmatrix} -\overline{v} \\ -\underline{v} \end{bmatrix}. \end{aligned} \tag{165}$$

Desta forma, seja uma RN com função de ativação *hard* Tanh f_{HTanh} dada por

$$f_{\text{HTanh}}(z) = N_{1f_\theta} z + N_{2f_\theta}\theta(y) + e_{0f_\theta} \tag{166a}$$

$$y = N_{3f_\theta} z + N_{4f_\theta}\theta(y) + e_{1f_\theta}, \tag{166b}$$

com $N_{1f_\theta} \in \mathbb{R}^{n_v \times n_z}$, $N_{2f_\theta} \in \mathbb{R}^{n_v \times n_y}$, $N_{3f_\theta} \in \mathbb{R}^{n_y \times n_z}$, $N_{4f_\theta} \in \mathbb{R}^{n_y \times n_y}$, $e_{0f_\theta} \in \mathbb{R}^{n_v}$, e $e_{1f_\theta} \in \mathbb{R}^{n_y}$. A representação implícita baseada em funções ReLU desta RN pode ser dada por (10) com

$$\begin{aligned} &N_1 = N_{1f_\theta},\ N_2 = N_{2f_\theta} N_{2\theta},\ e_0 = N_{2f_\theta} e_{0\theta} + e_{0f_\theta},\ q = q_\theta, \\ &N_3 = N_{3\theta} N_{3f_\theta},\ N_4 = N_{3\theta} N_{4f_\theta} N_{2\theta},\ e_1 = N_{3\theta}(N_{4f_\theta} e_{0\theta} + e_{1f_\theta}) + e_{1\theta}. \end{aligned} \tag{167}$$

6.4 Conclusões

Neste capítulo, a representação implícita baseada em funções ReLU foi estendida para arquiteturas mais complexas, como as RNRs e RNCs. Para as RNRs, demonstrou-se que a inclusão de uma equação para os estados ocultos da rede permite incorporar sua dinâmica à formulação. Já para as RNCs, detalhou-se a tradução das operações de convolução, agrupamento e preenchimento por meio do mesmo arcabouço matemático.

Além das arquiteturas, a metodologia foi generalizada para incluir outras funções de ativação, permitindo que redes originalmente não ReLU sejam analisadas sob a mesma ótica. Em suma, as extensões apresentadas consolidam a representação implícita baseada em funções ReLU como uma ferramenta versátil, capaz de descrever RNs de diferentes arquiteturas e funções de ativação. Essa capacidade de descrever RNs de diferentes configurações sob tal formalismo possibilita que as condições de estabilidade e a estimativa da RAO, derivadas nos capítulos anteriores, sejam aplicadas a uma vasta classe de sistemas dinâmicos e RNs.

7 CONCLUSÕES

Neste trabalho, foi abordado o problema da certificação da estabilidade exponencial dos pontos de equilíbrio de sistemas dinâmicos em tempo discreto cujo campo vetorial é dado por RNs ReLU. Para tanto, o sistema dinâmico é descrito na representação implícita baseada em funções ReLU para sistemas afins por partes apresentada em (GROFF; VALMORBIDA; GOMES DA SILVA JR., 2019). Então, fazendo uso de propriedades da função ReLU que a caracterizam de forma exata, é utilizada uma função de Lyapunov PWQ (que também é descrita com base em funções ReLU) para derivar as condições de estabilidade na forma de LMIs.

As principais vantagens da abordagem proposta residem na representação exata da função ReLU, resultando em condições significativamente menos conservadoras quando comparadas às técnicas baseadas em condições de setor. Além disso, a metodologia possibilita a certificação tanto da estabilidade global quanto local, neste último caso permitindo a estimativa da região de atração através da utilização de domínios de Lyapunov.

Outra vantagem reside no fato de que a representação implícita baseada em funções ReLU possibilita a representação de RNs com múltiplas camadas e com termos de *bias*, o que não é possível em diversas abordagens da literatura. Além disso, é possível representar RNs de diversas arquiteturas, desde RN *feedforward* até modelos mais avançados, como RNR, RNC ou DEQ. Destaca-se também a capacidade de representar conexões em série, paralelo e realimentação nesta representação, de forma a possibilitar a modelagem de RNs heterogêneas com camadas de diferentes arquiteturas e a análise de sistemas dados pela composição de sistemas lineares ou afins por partes com RNs. O *framework* apresentado neste trabalho é, portanto, altamente versátil, no sentido de que permite a certificação de diversas topologias e interconexões de sistemas estáticos e dinâmicos representados por RNs com funções de ativação ReLU.

Possíveis trabalhos futuros incluem:

- A generalização do trabalho para a análise de estabilidade de certas classes de sistemas não-lineares controlados por RN ReLU, como LPV (*Linear Parameter-Varying*), racionais e com não linearidades de setor. Nesses casos, modelagens clássicas da literatura podem ser empregadas a fim de obter condições de estabi-

lidade também na forma de LMIs. Embora as RNRs possam aproximar o comportamento desses sistemas dinâmicos, a integração da abordagem proposta com essas modelagens clássicas pode reduzir o esforço computacional para verificar a estabilidade de tais sistemas;

- Integrar as condições de estabilidade diretamente ao processo de treinamento das RNs. Possíveis maneiras seriam a inclusão dos resíduos das LMIs na função de perda utilizada no treinamento ou a aplicação de penalidades na função de perda caso as condições de estabilidade sejam violadas, garantindo, desta forma, que a rede treinada seja estável por construção;

- Investigar formas de representar RNRs com mecanismos de portão (*Gated Neural Networks*) (HOCHREITER; SCHMIDHUBER, 1997) na representação implícita baseada em funções ReLU. Essas arquiteturas, como LSTM (*Long Short-Term Memory*) e GRU (*Gated Recurrent Unit*), utilizam multiplicações elemento a elemento para regular o fluxo de informação relacionado a memória da rede. Atualmente, sua representação no *framework* deste trabalho não é possível, pois tal operação gera termos bilineares que extrapolam a estrutura PWA utilizada;

- Investigação de estratégias de controle amostrado para controladores de RNs em plantas contínuas. A modelagem conjunta da planta contínua e do controlador discreto (via RN) permite analisar o impacto do período de amostragem na estabilidade do sistema em MF, possibilitando a obtenção de limites para o período de amostragem na operação de sistemas físicos;

- Extensão da metodologia para a análise de estabilidade entrada-estado e entrada-saída em RNRs ou sistemas controlados por RNs. Através da aplicação de critérios de dissipatividade, pode-se derivar condições de estabilidade que garantam que os estados/saídas da rede (ou sistema em MF) permaneçam limitados diante de perturbações nas sequências de entrada;

- Aplicação da representação implícita obtida para RNCs no cálculo da estimativa da constante de Lipschitz para RNs de classificação. Essa métrica permite estimar a robustez do modelo em relação a variações nos dados de entrada, como, por exemplo, incertezas ou ruídos.

7.1 Publicações Associadas à Dissertação

Durante o desenvolvimento deste trabalho, os seguintes artigos foram produzidos e publicados (ou submetidos) em eventos e periódicos da área:

- **ZANETTE, D. V.**; CABRAL, L.; GOMES DA SILVA JR, J. M.; VALMORBIDA, G. Análise de estabilidade de sistemas lineares controlados por redes neurais emulando MPC. In: **Anais do XVI Simpósio Brasileiro de Automação Inteligente (SBAI)**, Manaus, 2023.

- **ZANETTE, D. V.**; GOMES DA SILVA JR, J. M.; CABRAL, L. Estabilidade de sistemas descritos por modelo entrada-saída controlados por redes neurais emulando MPC. In: **Anais do XXV Congresso Brasileiro de Automática (CBA)**, Rio de Janeiro, 2024.

- **ZANETTE, D. V.**; CABRAL, L.; GOMES DA SILVA JR, J. M.; VALMORBIDA, G. Regional stability of ReLU NN emulating MPC. Aceito para apresentação no **IFAC World Congress 2026**.

- CABRAL, L.; **ZANETTE, D. V.**; GOMES DA SILVA JR, J. M.; VALMORBIDA, G.; SBARBARO, D. Stability of Dynamical Systems with Vector Fields Given by Multi-Layer ReLU Neural Networks. Submetido à revista **IEEE Transactions on Automatic Control**.

REFERÊNCIAS

ABADI, M. *et al.* **TensorFlow**: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

AGGARWAL, C. C. **Neural Networks and Deep Learning**: a textbook. Berlim: Springer, 2019.

ANDERSON, D. G. Iterative procedures for nonlinear integral equations. **Journal of the ACM (JACM)**, New York, v. 12, n. 4, p. 547–560, 1965.

BAI, S.; KOLTER, J. Z.; KOLTUN, V. Deep Equilibrium Models. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2019, Vancouver. **Proceedings [...]** Curran Associates, 2019. v. 32.

BAZANELLA, A. S.; Gomes da Silva Junior, J. M. **Sistemas de Controle**: princípios e métodos de projeto. Porto Alegre: Editora da UFRGS, 2005.

BEMPORAD, A. *et al.* The explicit linear quadratic regulator for constrained systems. **Automatica**, Amsterdam, v. 38, n. 1, p. 3–20, 2002.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006.

BOSE, B. K. Neural Network Applications in Power Electronics and Motor Drives-An Introduction and Perspective. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 54, n. 1, p. 14–33, 2007.

BURDEN, R. L.; FAIRES, J. D. **Numerical Analysis**. 9th. ed. Boston, MA: Cengage Learning, 2010.

CABRAL, L. **Analysis of piecewise affine systems with a ramp-based implicit representation**. 2024. Tese (Doutorado em engenharia) — Université Paris-Saclay, Gif-sur-Yvette, 2024.

CABRAL, L.; VALMORBIDA, G.; GOMES DA SILVA JR., J. M. Estabilidade Regional de Sistemas Lineares de Tempo Discreto sob Saturação de Controle Assimétrica. *In*: XXIV CONGRESSO BRASILEIRO DE AUTOMáTICA (CBA 2022), 2022, Fortaleza. **Proceedings [...]** SBA, 2022. p. 1–6.

CABRAL, L.; VALMORBIDA, G.; GOMES DA SILVA JR., J. M. Regional Stability Analysis of Discrete-Time Piecewise Affine Systems. **IEEE Transactions on Automatic Control**, Piscataway, v. 70, n. 4, p. 2507–2520, 2025.

CAMACHO, E.; BORDONS, C. **Model Predictive Control**. Berlin: Springer, 2007.

CHANG, L.-C. *et al.* Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control. **Journal of Hydrology**, Amsterdam, v. 563, p. 1054–1064, 2018.

COLLOBERT, R. **Large scale machine learning**. 2004. Tese (Doutorado em engenharia) — Université Paris VI, Paris, 2004.

COTTLE, R. W.; PANG, J.-S.; STONE, R. E. **The Linear Complementarity Problem**. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2009. (Classics in Applied Mathematics). Republicação corrigida da edição de 1992 da Academic Press.

DAMAK, H.; HAMMAMI, M. Generalized practical stability of Hopfield-type neural networks differential equations. **Mathematical Modelling and Analysis**, Vilnius, v. 31, n. 1, p. 149–171, Jan. 2026.

DIAMOND, S.; BOYD, S. CVXPY: a Python-embedded modeling language for convex optimization. **Journal of Machine Learning Research**, Cambridge, v. 17, n. 83, p. 1–5, 2016.

EBIHARA, Y. *et al.* Stability Analysis of Recurrent Neural Networks by IQC with Copositive Multipliers. *In*: IEEE CONFERENCE ON DECISION AND CONTROL (CDC), 60., 2021, Austin. **Proceedings [...]** IEEE, 2021. p. 5098–5103.

EBIHARA, Y. *et al.* l2 induced norm analysis of discrete-time LTI systems for nonnegative input signals and its application to stability analysis of recurrent neural networks. **European Journal of Control**, Amsterdam, v. 62, p. 99–104, 2021.

ELMAN, J. L. Finding Structure in Time. **Cognitive Science**, Hoboken, v. 14, n. 2, p. 179–211, 1990.

FAZLYAB, M.; MORARI, M.; PAPPAS, G. J. Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming. **IEEE Transactions on Automatic Control**, Piscataway, v. 67, n. 1, p. 1–15, 2022.

FAZLYAB, M. *et al.* Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2019. **Proceedings [...]** Curran Associates: Inc., 2019. v. 32.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: MIT Press, 2016. `http://www.deeplearningbook.org`.

GRAVES, A.; MOHAMED, A.-r.; HINTON, G. Speech recognition with deep recurrent neural networks. *In*: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2013, Vancouver. **Proceedings [...]** IEEE, 2013. p. 6645–6649.

GROFF, L. B. **Event-triggered control for piecewise affine discrete-time systems**. 2020. Tese (Doutorado em engenharia) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2020.

GROFF, L. B.; VALMORBIDA, G.; GOMES DA SILVA JR., J. M. Stability analysis of piecewise affine discrete-time systems. *In*: IEEE 58TH CONFERENCE ON DECISION AND CONTROL (CDC), 2019, Nice. **Proceedings [...]** IEEE, 2019. p. 8172–8177.

GROFF, L. B.; VALMORBIDA, G.; GOMES DA SILVA JR., J. M. An implicit representation for the analysis of piecewise affine discrete-time systems. **Automatica**, Amsterdam, v. 147, p. 110730, 2023.

HAMAD, R. A. *et al.* Deep learning for sensor-based activity recognition: a survey. **Pattern Recognition Letters**, Amsterdam, 2016.

HARRIS, C. R. *et al.* Array programming with NumPy. **Nature**, London, v. 585, n. 7825, p. 357–362, 2020.

HAYKIN, S. **Redes Neurais**: princípios e prática. São Paulo: Bookman, 2001. 898 p.

HE, K. *et al.* Delving deep into rectifiers: surpassing human-level performance on imagenet classification. *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2015, Santiago. **Proceedings [...]** IEEE, 2015. p. 1026–1034.

HE, K. *et al.* Deep Residual Learning for Image Recognition. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, Las Vegas. **Proceedings [...]** IEEE, 2016. p. 770–778.

HEEMELS, W.; SCHUMACHER, J. M.; WEILAND, S. Linear complementarity systems. **SIAM Journal on Applied Mathematics**, Philadelphia, v. 60, n. 4, p. 1234–1269, 2000.

HERCEG, M. *et al.* Multi-Parametric Toolbox 3.0. *In*: EUROPEAN CONTROL CONFERENCE (ECC), 2013, Zurich. **Proceedings [...]** IEEE, 2013. p. 502–510.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, Cambridge, v. 9, n. 8, p. 1735–1780, 1997.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, Oxford, v. 2, n. 5, p. 359–366, 1989.

HU, B.; SEILER, P.; RAGHUNATHAN, A. A Complete Set of Quadratic Constraints for Repeated ReLU and ReLU-Like Nonlinearities. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2022, La Jolla. **Proceedings [...]** Curran Associates: Inc., 2022. v. 35, p. 32125–32138.

JAIN, A.; MAO, J.; MOHIUDDIN, K. Artificial neural networks: a tutorial. **Computer**, Los Alamitos, v. 29, n. 3, p. 31–44, 1996.

JOHANSSON, M. **Piecewise Linear Control Systems**: a computational approach. Berlin, Heidelberg: Springer Science & Business Media, 2003. (Lecture Notes in Control and Information Sciences, v. 284).

KELLEY, C. T. **Solving Nonlinear Equations with Newton's Method**. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2003.

KHALIL, H. K. **Nonlinear Systems; 3rd ed.** [S.l.]: Prentice-Hall, 2002.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), 2012, Stateline. **Proceedings [...]** Curran Associates, 2012. v. 25, p. 1097–1105.

LECUN, Y. *et al.* Efficient Backprop. In: **Neural networks**: tricks of the trade. Berlin: Springer, 1998. p. 9–50.

LECUN, Y. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Piscataway, v. 86, n. 11, p. 2278–2324, 1998.

LEE, J. *et al.* Learning quadrupedal locomotion over challenging terrain. **Science Robotics**, Washington, D.C., v. 5, n. 47, 2020.

LESHNO, M. *et al.* Multilayer feedforward networks with a nonpolynomial activation function can approximate any continuous function. **Neural networks**, [S.l.], v. 6, n. 6, p. 861–867, 1993.

LI, Z. *et al.* Missile Guidance Law Based on Robust Model Predictive Control Using Neural-Network Optimization. **IEEE Transactions on Neural Networks and Learning Systems**, Piscataway, v. 26, n. 8, p. 1803–1809, 2015.

LIN, T. *et al.* Learning long-term dependencies in NARX recurrent neural networks. **IEEE Transactions on Neural Networks**, Piscataway, v. 7, n. 6, p. 1329–1338, 1996.

LIU, P.; WANG, J.; ZENG, Z. An Overview of the Stability Analysis of Recurrent Neural Networks With Multiple Equilibria. **IEEE Transactions on Neural Networks and Learning Systems**, Piscataway, v. 34, n. 3, p. 1098–1111, 2023.

LIU, Y.-J. *et al.* Adaptive Neural Network Control for a Class of Nonlinear Systems With Function Constraints on States. **IEEE Transactions on Neural Networks and Learning Systems**, Piscataway, 2023.

LJUNG, L. Neural Networks in System Identification. *In*: SIGNAL ANALYSIS AND PREDICTION, 1998. **Proceedings [...]** Birkhäuser Boston, 1998. p. 163–182.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. *In*: INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), 30., 2013, Atlanta. **Proceedings [...]** PMLR, 2013. v. 30, n. 1, p. 3.

MACIEJOWSKI, J. M. **Predictive Control with Constraints**. Harlow: Pearson, 2000.

MAN, K.; TANG, K.; KWONG, S. Genetic algorithms: concepts and applications [in engineering design]. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 43, n. 5, p. 519–534, 1996.

MARQUARDT, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. **Journal of the Society for Industrial and Applied Mathematics**, Philadelphia, v. 11, n. 2, p. 431–441, 1963.

MOSEK ApS. **The MOSEK Python Optimizer API manual. Version 10.1.** 2024.

NOORI, S. *et al.* Stability and performance analysis of discrete-time ReLU recurrent neural networks. *In*: IEEE CONFERENCE ON DECISION AND CONTROL (CDC), 63., 2024, Milan. **Proceedings [...]** IEEE, 2024. p. 8626–8632.

NOORI, S. V. *et al.* A Complete Set of Quadratic Constraints for Repeated ReLU and Generalizations. **IEEE Transactions on Automatic Control**, Piscataway, p. 1–14, 2025.

PILLONETTO, G. *et al.* Deep networks for system identification: a survey. **Automatica**, Amsterdam, v. 151, p. 110935, 2023.

PRINCE, S. J. **Understanding Deep Learning**. Cambridge: MIT Press, 2023.

QIN, S.; BADGWELL, T. A. A survey of industrial model predictive control technology. **Control Engineering Practice**, Oxford, v. 11, n. 7, p. 733–764, 2003.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *In*: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR), 4., 2016, San Juan. **Proceedings [...]** ICLR, 2016.

REDMON, J. *et al.* You Only Look Once: unified, real-time object detection. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, Las Vegas. **Proceedings [...]** IEEE, 2016. p. 779–788.

RICHARDSON, C. R.; TURNER, M. C.; GUNN, S. R. Strengthened Circle and Popov Criteria for the Stability Analysis of Feedback Systems With ReLU Neural Networks. **IEEE Control Systems Letters**, Piscataway, v. 7, p. 2635–2640, 2023.

ROJAS, R. **Neural Networks - A Systematic Introduction**. Berlin: Springer-Verlag, 1996.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: convolutional networks for biomedical image segmentation. *In*: INTERNATIONAL CONFERENCE ON MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION (MICCAI), 2015. **Proceedings [...]** Springer, 2015. p. 234–241.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, Washington, DC, v. 65, n. 6, p. 386–408, 1958.

ROSS, R. *et al.* Bounding the Difference Between Model Predictive Control and Neural Networks. *In*: THE 4TH ANNUAL LEARNING FOR DYNAMICS AND CONTROL CONFERENCE, 2022, Palo Alto. **Proceedings [...]** PMLR, 2022. p. 817–829. (Proceedings of Machine Learning Research, v. 168).

SAMANIPOUR, P.; POONAWALA, H. A. Stability Analysis and Controller Synthesis using Single-hidden-layer ReLU Neural Networks. **IEEE Transactions on Automatic Control**, Piscataway, p. 1–12, 2023.

SHEN, T.; PETERSEN, I. R. Linear Threshold Discrete-Time Recurrent Neural Networks: stability and globally attractive sets. **IEEE Transactions on Automatic Control**, Piscataway, v. 61, n. 9, p. 2650–2656, 2016.

SIEGELMANN, H. T.; SONTAG, E. D. Recurrent neural networks are universal approximators. **International Journal of Neural Systems**, Singapore, v. 2, n. 03, p. 213–222, 1991.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to Sequence Learning with Neural Networks. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2014, Montreal. **Proceedings [...]** Curran Associates, 2014. v. 27, p. 3104–3112.

TAIGMAN, Y. *et al.* Deepface: closing the gap to human-level performance in face verification. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2014, Columbus. **Proceedings [...]** IEEE, 2014. p. 1701–1708.

TARBOURIECH, S. *et al.* **Stability and Stabilization of Linear Systems with Saturating Actuators**. London: Springer Science & Business Media, 2011.

VALMORBIDA, G.; HOVD, M. Quadratic programming with ramp functions and fast online QP-MPC solutions. **Automatica**, Amsterdam, v. 153, p. 111011, 2023.

VASWANI, A. *et al.* Attention is all you need. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2017, Long Beach. **Proceedings [...]** Curran Associates, 2017. v. 30, p. 5998–6008.

WANG, D.; HUANG, J. Neural network-based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict-feedback form. **IEEE Transactions on Neural Networks**, Piscataway, 2005.

WU, W.; CHEN, J.; CHEN, J. Stability Analysis of Systems with Recurrent Neural Network Controllers. **IFAC-PapersOnLine**, Amsterdam, v. 55, n. 12, p. 170–175, 2022. 14th IFAC Workshop on Adaptive and Learning Control Systems (ALCOS 2022).

YANG, T. *et al.* Neural Network-Based Adaptive Antiswing Control of an Underactuated Ship-Mounted Crane with Roll Motions and Input Dead Zones. **IEEE Transactions on Neural Networks and Learning Systems**, Piscataway, 2020.

YI, Z.; TAN, K. K. Multistability of discrete-time recurrent neural networks with unsaturating piecewise linear activation functions. **IEEE Transactions on Neural Networks**, Piscataway, v. 15, n. 2, p. 329–336, 2004.

YIN, H.; SEILER, P.; ARCAK, M. Stability Analysis Using Quadratic Constraints for Systems With Neural Network Controllers. **IEEE Transactions on Automatic Control**, New York, v. 67, n. 4, p. 1980–1987, 2022.

ZHANG, L. *et al.* Activity invariant sets and exponentially stable attractors of linear threshold discrete-time recurrent neural networks. **IEEE Transactions on Automatic Control**, Piscataway, v. 54, p. 1341–1347, 2009.

ZHOU, Q. *et al.* Adaptive Neural Network Tracking Control for Robotic Manipulators With Dead Zone. **IEEE Transactions on Neural Networks and Learning Systems**, New York, v. 30, n. 12, p. 3611–3620, 2019.