ANDRÉ LUCAS DE OLIVEIRA DUARTE

Aplicações de *reservoir computing* em processamento de sinais

São Paulo
2023

ANDRÉ LUCAS DE OLIVEIRA DUARTE

Aplicações de *reservoir computing* em processamento de sinais

Versão Corrigida

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

São Paulo
2023

ANDRÉ LUCAS DE OLIVEIRA DUARTE

Aplicações de *reservoir computing* em processamento de sinais

Versão Corrigida

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

Área de Concentração:

Sistemas Eletrônicos

Orientador:

Marcio Eisencraft

São Paulo
2023

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

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Duarte, André Lucas
Aplicações de reservoir computing em processamento de sinais / A. L. Duarte -- versão corr. -- São Paulo, 2023.
146 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.REDES NEURAIS 2.CAOS (SISTEMAS DINÂMICOS) 3.PROCESSAMENTO DIGITAL DE SINAIS I I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

Agradecimentos

Ao meu orientador, Prof. Dr. Marcio Eisencraft, pelas inúmeras reuniões, ensinamentos, presteza e conselhos valiosos durante a orientação deste mestrado.

Aos professores Dr. Magno T. M. Silva (USP) e Dr. Murilo B. Loiola (UFABC), pela participação na banca examinadora de qualificação e pelas observações que foram essenciais para o aprimoramento deste trabalho.

À toda a minha família, por todo o carinho e consideração ao longo dos anos. Em especial, aos meus queridos pais e avós, cujo valor e importância para mim não se traduzem em palavras.

À minha namorada, pelas incontáveis conversas, convivência e apoio, me fazendo refletir e dando forças durante a produção deste e dos demais trabalhos relacionados.

À Escola Politécnica da Universidade de São Paulo, por ter possibilitado a realização deste mestrado, assim como à Universidade Federal do ABC, por ter me proporcionado a formação como engenheiro.

Aos meus colegas que, direta ou indiretamente, tiveram uma parcela de colaboração neste trabalho.

"Não há estradas reais para a geometria."

Euclides (c. 325 a.C. - c. 265 a.C.)

Resumo

Neste trabalho, investiga-se o desempenho do *reservoir computing*, por meio das *echo state networks* (ESNs), em problemas de processamento de sinais. Após uma breve revisão da literatura, exploram-se três aplicações. Inicia-se com o uso de uma ESN para mitigação de ruído branco sobre um sinal caótico. Apesar de simples, o problema de *denoising* é de interesse fundamental e surge nas mais diversas áreas de processamento de sinais. Em seguida, integra-se uma ESN como o receptor de um sistema de comunicação baseado em caos sujeito a condições não ideais. Estudam-se isoladamente os efeitos introduzidos por ruído branco e filtragem do canal antes de investigar ambos simultaneamente. Por fim, submete-se uma ESN ao problema de predição de sinais musicais. Isto é, dado uma nota, estimar qual seria a seguinte numa peça musical. Dessa forma, tangencia-se a questão de se uma rede neural é capaz de emular a criatividade humana.

Nos dois primeiros estudos, emprega-se o mapa tenda inclinada para geração dos sinais caóticos por este possuir um único parâmetro que influencia diretamente em suas características espectrais de forma bem conhecida. O método de filtragem linear de Wiener é usado para comparação com os desempenhos obtidos por meio do uso de ESNs.

No último problema abordado, empregam-se sinais contendo informações musicais oriundas de obras clássicas, salvas em arquivos no formato *musical instrument digital interface* (MIDI).

Os resultados das simulações indicam que a escolha do mapa e de seus parâmetros, do tipo de codificação no caso da segunda aplicação, bem como dos parâmetros da ESN influenciam diretamente os desempenhos obtidos. Sobretudo, constata-se que o uso de ESN nas duas primeiras aplicações estudadas fornece desempenhos superiores àqueles obtidos usando-se o filtro de Wiener, o método de filtragem linear ótimo.

Palavras-chave – caos, *echo state networks*, *machine learning*, música, processamento de sinais, redes neurais recorrentes, sistemas de comunicação baseados em caos, sistemas dinâmicos.

Abstract

This thesis investigates the performance of reservoir computing, through the use of echo state networks (ESNs) in signal processing tasks. After a brief literature review, three different ESN applications are studied. In the first one, an ESN is used to attenuate additive white gaussian noise over a chaotic signal. Despite its simplicity, the so-called denoising task is very much relevant as it is found in a variety of different areas within signal processing. An ESN is then integrated as the receiver into a chaos-based communication system under non-ideal conditions. The effects of noise and channel filtering are studied separately before considering both simultaneously. Ultimately, an ESN is used to predict musical sequences. This means that, when a note from a piece of music is fed into the network, it must estimate which would be the following note. This way, the question of whether a neural network can emulate human musical creativity is slightly explored.

In the first two applications the skew tent map is used to obtain the chaotic signals, since its spectral features are known and determined by its unique parameter. The Wiener linear filtering method is used as benchmark for evaluating the performance of the designed ESNs.

In the last problem, classical musical pieces files available in the musical instrument digital interface format are utilized to generate the musical sequences used.

The results achieved indicate that the choice of the map and its parameters, as well as the network parameters have a direct impact on the performance of the ESNs. They have outperformed the Wiener filter, the optimal linear filtering method, in both first and second tasks.

Keywords – chaos, echo state networks, machine learning, music, signal processing, recurrent neural networks, chaos-based communication systems, dynamical systems.

Lista de figuras

2.1 Características do mapa tenda inclinada $f_I(\cdot)$ (2.5): a) $f_I(x)$ em função de x; b) expoente de Lyapunov (2.6); c) número de Lyapunov. 10

2.2 Exemplos de trajetórias de $x(n)$ para $\alpha = 0.5$ e condições iniciais ligeiramente distintas, ilustrando a DSCI: a) $x_0 = 0.9 + 10^{-3}$; b) $x_0 = 0.9 + 10^{-7}$; c) $x_0 = 0.9 + 10^{-11}$. As curvas na cor azul representam a trajetória para $x_0 = 0.9$. 11

2.3 SAC (2.7) do mapa $f_I(\cdot)$ (2.5) para $|\alpha| = 0.0,\ 0.2, \ldots,\ 0.8,\ 0.9$. 12

2.4 DEP (2.8) do mapa $f_I(\cdot)$ (2.5) para $\alpha = -0.8,\ -0.6,\ \ldots,\ 0.8$. 13

2.5 SCBC de tempo discreto estudado. 13

2.6 Captura de tela de um arquivo MIDI aberto no software MidiEditor. . . . 16

2.7 Ilustração da relação entre a notação das notas da Tabela 2.1 e as teclas de um piano. 16

2.8 Diferenças entre processamentos dos arquivos MIDI: a) notas agrupadas por *track*; b) notas na sequência de execução. 20

2.9 Arquitetura de uma RNN, em que todas as conexões (entrada-RNN, internas e RNN-saída) são treinadas. 22

2.10 Arquitetura de uma ESN, indicando os sinais $\boldsymbol{u}(n)$, $\boldsymbol{y}(n)$, $\boldsymbol{d}(n)$ e seu estado $\boldsymbol{r}(n)$. 23

2.11 Nós de entrada e nós do *reservoir* transmitindo seus estados, no instante n, ao k-ésimo nó do *reservoir*. 27

2.12 Nós do *reservoir* transmitindo seus estados, no instante n, ao k-ésimo nó da camada de saída. 29

2.13 Estados dos nós da rede no instante $n = 6$. 35

3.1 Diagrama de blocos do problema de redução de ruído de sinais caóticos: a) período de treinamento da ESN; b) período de teste da ESN. 42

3.2 Exemplo de sinal caótico corrompido por ruído: a) sinal caótico livre de ruído; b) função-amostra de sinal AWGN para $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB; c) sinal de entrada da ESN (linha azul) e sinal caótico livre de ruído (linha preta). . . 43

3.3 Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral de $\boldsymbol{W}$; c) e d) parâmetros das distribuições uniformes usadas para obter $\boldsymbol{W}^{\mathrm{in}}$; e) ganho de processamento. 44

3.4 Mitigação de ruído para $\alpha = 0.9$ e $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB: a) sinal de entrada $u(n)$; b) sinal estimado pelo FW $\widehat{d}_W(n)$; c) sinal estimado pela ESN $\widehat{d}(n)$. As curvas na cor preta representam o sinal desejado $d(n)$. 45

3.5 Mitigação de ruído para $\alpha = 0.1$ e $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB: a) sinal de entrada $u(n)$; b) sinal estimado pelo FW $\widehat{d}_W(n)$; c) sinal estimado pela ESN $\widehat{d}(n)$. As curvas na cor preta representam o sinal desejado $d(n)$. 46

3.6 Ganho de processamento em dB em função do parâmetro do mapa tenda inclinada α usando uma ESN e um FW para $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB. 47

4.1 Diagrama de blocos do SCBC sujeito à AWGN (Cenário I): a) período de treinamento da ESN; b) período de teste da ESN. 51

4.2 Exemplos de sinais do SCBC da Fig. 4.1: a) mensagem binária; b) sinal transmitido; c) sinal transmitido (linha azul) e sinal recebido (linha vermelha). 52

4.3 Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral de $\boldsymbol{W}$; c) e d) parâmetros das distribuições uniformes usadas para obter $\boldsymbol{W}^{\mathrm{in}}$; e) taxa de erro de bit. 53

4.4 Exemplos de sinais do SCBC da Fig. 4.1: a) mensagem binária; b) sinal transmitido (linha azul) e sinal recebido (linha vermelha); c) decisão da ESN. Pontos na cor vermelha indicam decisões errôneas. 54

4.5 Curvas de BER em função da SNR no Cenário I. 55

4.6 Diagrama de blocos do SCBC somente com canal (Cenário II): a) período de treinamento da ESN; b) período de teste ESN. 56

4.7 Exemplos de sinais do SCBC da Fig. 4.6: a) mensagem binária; b) sinal transmitido; c) sinal transmitido (linha azul) e sinal recebido (linha vermelha). 57

4.8 Resposta em frequência normalizada do filtro $H(z) = h_0 + z^{-1} + h_0 z^{-2}$: a) curvas para $h_0 = 0.2, 0.4, \ldots, 1.0$; b) mapa de cores. 58

4.9 Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral; c) e d) parâmetros da distribuições uniformes usadas para obter $\boldsymbol{W}^{\text{in}}$; e) taxa de erro de bit. 59

4.10 Exemplos de sinais do SCBC da Fig. 4.6: a) mensagem binária; b) sinal transmitido (linha azul) e sinal recebido (linha vermelha); c) decisão da ESN. Pontos na cor vermelha indicam decisões errôneas. 60

4.11 Curvas de BER em função de h_0 no Cenário II. 61

4.12 Diagrama de blocos do SCBC com canal e ruído (Cenário III): a) período de treinamento da ESN; b) período de teste da ESN. 62

4.13 Exemplos de sinais do SCBC da Fig. 4.12: a) mensagem binária; b) sinal transmitido; c) sinal transmitido (linha azul) e sinal recebido (linha vermelha). 63

4.14 Evolução dos parâmetros durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral; c) e d) parâmetros das distribuições uniformes usadas para gerar $\boldsymbol{W}^{\text{in}}$; e) taxa de erro de bit. 64

4.15 Exemplos de sinais do SCBC da Fig. 4.12: a) mensagem binária; b) sinal transmitido (linha azul) e sinal recebido (linha vermelha); c) decisão da ESN. Pontos na cor vermelha indicam decisões errôneas. 65

4.16 Mapas de BER × SNR × h_0 do Cenário III: a) ESN (2 amostras); b) filtro de Wiener seguido de sincronismo caótico; c) somente sincronismo caótico. 66

4.17 Curvas de BER do Cenário III: a) SNR fixa em 20.0 dB; b) h_0 fixo em 0.1. 67

5.1 Diagrama de blocos do problema de predição de sinais musicais: a) período de treinamento da ESN. b) período de teste da ESN. 69

5.2 Sinal musical disponível para treinamento e teste da ESN. 70

5.3 Histograma de amostras de altura $s_1(n)$. 71

5.4 Sinal musical normalizado para treinamento e teste da ESN. 72

5.5 Primeiras amostras usadas para treinamento da ESN: a) altura; b) instante de início; c) instante de término. 73

5.6 Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral; c) e d) parâmetros das distribuições uniformes usadas para gerar $\boldsymbol{W}^{\text{in}}$; e) relação sinal-ruído. 74

5.7 Curvas de erro de predição médio: a) altura; b) instante de início; c) instante de término. 75

5.8 Sinais estimados durante o período de teste: a) altura; b) instante de início; c) instante de término. Marcadores dourados com contorno preto indicam uma estimativa bem-sucedida. 76

5.9 Curvas de erro de predição médio obtidas reordenando-se $\boldsymbol{s}(n)$: a) altura; b) instante de início; c) instante de término. 78

D.1 Processamento de um dos *tracks* do compasso 13 da obra Opus 24, No. 18: a) extrato da partitura musical correspondente; b) trecho do arquivo MIDI correspondente; c) à esquerda: altura das notas. À direita: instantes de início (verde) e de término (vermelho) em *ticks*. 102

Lista de tabelas

2.1 Mapeamento das notas musicais em números inteiros nos arquivos MIDI. . 17

2.2 Parâmetros utilizados no exemplo. 32

3.1 Parâmetros utilizados no problema de redução de ruído. 45

4.1 Parâmetros utilizados no problema do SCBC considerando apenas ruído. . 53

4.2 Parâmetros utilizados no problema do SCBC considerando apenas canal. . 57

4.3 Parâmetros utilizados no problema do SCBC considerando ruído e canal. . 64

5.1 Médias e desvios padrão de $\boldsymbol{s}_k$. 72

5.2 Parâmetros utilizados no problema de predição de sinais musicais. 74

5.3 Valores de predição e desejados obtidos, referentes ao primeiro compasso apresentado na Fig. 5.8. 77

5.4 Parâmetros para comparação entre os desempenhos da ESN em ambos os casos. 79

6.1 Principais contribuições da dissertação. 83

C.1 Lista das peças de Frédéric Chopin utilizadas para o estudo do Caso 1, realizado no Capítulo 5, e os instantes correspondentes. 98

Lista de abreviaturas

ANN	Rede neural artificial (do inglês *artificial neural network*)
AWGN	Ruído gaussiano branco e aditivo (do inglês *additive white gaussian noise*)
BER	Taxa de erro de bit (do inglês *bit error rate*)
bit	Digito binário (do inglês *binary digit*)
bpm	Batidas por minuto
CNMAC	Congresso nacional de matemática aplicada e computacional
DEP	Densidade espectral de potência
DSCI	Dependência sensível às condições iniciais
ESN	Rede neural com estado de eco (do inglês *echo state network*)
FW	Filtro de Wiener
IoT	Internet das coisas (do inglês *internet of things*)
MIDI	Interface digital de instrumento musical (do inglês *musical instrument digital interface*)
RC	Computação de reservatório (do inglês *reservoir computing*)
RMSE	Raíz do erro quadrático médio (do inglês *root mean square error*)
RNN	Rede neural recorrente (do inglês *recurrent neural network*)
SAC	Sequência de autocorrelação
SBrT	Simpósio brasileiro de telecomunicações e processamento de sinais
SC	Sincronismo caótico
SCBC	Sistema de comunicação baseado em caos
SNR	Relação sinal-ruído (do inglês *signal-to-noise ratio*)

Lista de algoritmos

1 Geração e treinamento de uma ESN. 31
2 Obtenção dos coeficientes de um filtro de Wiener. 39

Lista de símbolos

Neste texto, escalares são representados por letras maiúsculas ou minúsculas em itálico, como k, M e $s(n)$. Vetores-coluna são denotados por letras minúsculas em negrito, por exemplo $\boldsymbol{b}$ e $\boldsymbol{x}(n)$. Matrizes são escritas usando-se letras maiúsculas em negrito, como $\boldsymbol{W}$ e $\boldsymbol{A}$.

Símbolos gerais

$f(\cdot)$	função com domínio e contradomínio V
V	subconjunto de $\mathbb{R}^M$
$\subseteq$	é subconjunto de
$\mathbb{R}^M$	conjunto dos vetores de ordem M com elementos reais
$\mathbb{N}$	conjunto dos números naturais
$\mathbb{N}^*$	conjunto dos números naturais não nulos
$f^n(\cdot)$	n-ésima iterada de $f(\cdot)$
$\boldsymbol{x}(\cdot)$	elemento de V
$\|\boldsymbol{x}\|$	norma euclidiana do vetor $\boldsymbol{x}$
$x_1, x_2, \ldots, x_M$	elementos do vetor $\boldsymbol{x}$
$\{a_{ij}\}_{M\times N}$	matriz $M \times N$ cujo elemento da i-ésima linha e j-ésima coluna é a_{ij}
$f^{'}(\cdot)$	derivada da função $f(\cdot)$ com relação ao seu argumento
$E[\cdot]$	operador esperança matemática
$\mathbf{0}$	matriz nula de dimensões adequadas ao contexto
$\mathbf{1}$	matriz de entradas unitárias e dimensões adequadas ao contexto
$\sim$	distribuído de acordo com
$U[a, b]$	distribuição uniforme contínua no intervalo $[a, b]$
$*$	soma de convolução
$\odot$	produto de Hadamard
$\delta(\cdot)$	função impulso unitário de tempo discreto
$\mathrm{sgn}(\cdot)$	função sinal (do inglês *signal*)
$\mathrm{nint}(\cdot)$	função inteiro mais próximo (do inglês *nearest integer*)

Sinais caóticos

n	variável independente discreta (tempo), $n \in \mathbb{N}$
$\boldsymbol{x}_0$	condição inicial
$\boldsymbol{x}(n;\boldsymbol{x}_0)$	n-ésima amostra do sinal $\boldsymbol{x}(n)$, com condição inicial $\boldsymbol{x}_0$
$X_{\boldsymbol{x}_0}$	trajetória de $\boldsymbol{x}(n;\boldsymbol{x}_0)$
$\boldsymbol{q}$	ponto eventualmente periódico de $\boldsymbol{x}(n;\boldsymbol{x}_0)$
τ	período de $\boldsymbol{x}(n;\boldsymbol{x}_0)$
$L(\boldsymbol{x}_0)$	número de Lyapunov de $\boldsymbol{x}(n;\boldsymbol{x}_0)$
$h(\boldsymbol{x}_0)$	expoente de Lyapunov de $\boldsymbol{x}(n;\boldsymbol{x}_0)$
$f_I(\cdot)$	mapa tenda inclinada
α	parâmetro do mapa tenda inclinada
k	passo da autocorrelação
$R(k)$	sequência de autocorrelação
ω	frequência discreta em radianos/amostra
$P(\omega)$	densidade espectral de potência
h_I	expoente de Lyapunov do mapa tenda inclinada
L_I	número de Lyapunov do mapa tenda inclinada
$\boldsymbol{A}$	matriz de sincronismo
$\boldsymbol{e}(n)$	erro de sincronização
$\lambda_1, \lambda_2, \ldots, \lambda_M$	autovalores da matriz $\boldsymbol{A}$
γ	parâmetro da codificação soma
$w(n)$	sinal AWGN
$m(n)$	mensagem a ser transmitida
$s(n)$	sinal transmitido
$r(n)$	sinal recebido
$h_c(n)$	resposta ao impulso unitário do canal
$c(\cdot)$	função de codificação caótica
$\widehat{m}(n)$	estimativa da mensagem no receptor

Reservoir computing

$\boldsymbol{u}(n)$	sinal de entrada
N_u	dimensão de $\boldsymbol{u}(n)$
$\boldsymbol{d}(n)$	sinal desejado
N_d	dimensão de $\boldsymbol{d}(n)$
L	comprimento de treinamento

$E(\boldsymbol{y}, \boldsymbol{d})$	medida de erro entre $\boldsymbol{y}$ e $\boldsymbol{d}$
$\boldsymbol{W}^{\text{in}}$	matriz de entrada
$\boldsymbol{W}$	matriz de pesos internos
$\boldsymbol{W}^{\text{aux}}$	matriz auxiliar
N	número de nós
$\boldsymbol{r}(n)$	vetor de estados
$\boldsymbol{y}(n)$	sinal de saída
a	parâmetro de *leakage*
ℓ	comprimento de transiente
λ	raio espectral de $\boldsymbol{W}$
λ_{aux}	raio espectral de $\boldsymbol{W}^{\text{aux}}$
p, q	parâmetros das distribuições uniformes de $\boldsymbol{W}^{\text{in}}$
$\boldsymbol{T}$	matriz de trajetória
$\boldsymbol{D}$	matriz de sinal desejado
$\boldsymbol{T}^{+}$	matriz pseudo-inversa de Moore-Penrose de $\boldsymbol{T}$

Filtro de Wiener

M	número de coeficientes do filtro
$w_0, w_1, \ldots, w_{M-1}$	coeficientes do filtro
$u(n)$	sinal de entrada do filtro
$y(n)$	sinal saída do filtro
$e(n)$	sinal de erro do filtro
$d(n)$	sinal desejado
J	função custo
$r_{uu}(n)$	função de autocorrelação de $u(n)$
$r_{ud}(n)$	função de correlação cruzada de $u(n)$ e $d(n)$
$\boldsymbol{R}$	matriz de autocorrelação
$\boldsymbol{p}$	vetor de correlação cruzada
$\boldsymbol{w}_{\text{o}}$	vetor de coeficientes ótimos

Sumário

Lista de figuras **iii**

Lista de tabelas **vii**

Lista de abreviaturas **viii**

Lista de algoritmos **ix**

Lista de símbolos **x**

1 Introdução **1**

1.1 Objetivos 3

1.2 Metodologia 4

1.3 Contribuições originais e artigos 4

1.4 Estrutura da dissertação 5

2 Revisão bibliográfica **7**

2.1 Sinais de interesse 7

2.1.1 Sinais caóticos 7

2.1.1.1 Definições 8

2.1.1.2 Mapa tenda inclinada 9

2.1.1.3 Aplicação em comunicações 12

2.1.1.4 Codificação e sincronismo caótico 14

2.1.2 Sinais musicais 15

2.1.2.1 Processamento dos arquivos MIDI 19

2.2 A técnica de *reservoir computing* empregada: *echo state networks* 21

2.2.1 Camada de entrada 23
2.2.2 *Reservoir* 24
2.2.3 Camada de saída 28
2.2.4 Treinamento 29
2.2.5 Exemplo: predição de um sinal periódico 32
2.2.6 Outros comentários 35
2.3 Filtro de Wiener 37
2.4 Resumo do capítulo 38

3 Redução de ruído em sinais caóticos 41
3.1 O problema de redução de ruído 42
3.2 Resultados de simulações 43
3.3 Conclusões 47

4 Sistema de comunicação baseado em caos utilizando ESN 49
4.1 Cenário I: somente ruído 50
4.1.1 Resultados de simulações 51
4.2 Cenário II: somente canal 54
4.2.1 Resultados de simulações 56
4.3 Cenário III: canal e ruído 61
4.3.1 Resultados de simulações 62
4.4 Conclusões 66

5 Predição de sinais musicais 68
5.1 O problema de predição musical 69
5.2 Resultados de simulações 73
5.3 Conclusões 79

6 Conclusões e trabalhos futuros 81

6.1 Contribuições 83
6.2 Trabalhos futuros 85

Anexo A – Código em Python para processamento de arquivos MIDI 93

Anexo B – Código em MATLAB® para solução do exemplo 96

Anexo C – Obras musicais utilizadas 98

Anexo D – Exemplo de processamento MIDI 101

Anexo E – Artigo submetido *Signal Processing* 2023 103

Anexo F – Artigo CNMAC 2021 117

Anexo G – Artigo SBrT 2021 120

Capítulo 1

Introdução

O estudo de redes neurais abriu caminho para que o desenvolvimento de máquinas inteligentes, empregando redes neurais artificiais (ANNs - *artificial neural networks*) baseadas no modelo biológico de neurônios e suas conexões, se tornasse possível [1].

Dentre os primeiros trabalhos da área, destacam-se [2–4]. Em [2], considera-se que é proposta a arquitetura das primeiras ANNs, por meio da combinação de vários elementos simples, conhecidos hoje por neurônios de McCulloch-Pitts. Em [3], encontra-se a primeira lei de aprendizagem de ANNs, que foi aprimorada em [4] para permitir simulações computacionais [1].

A ideia de futuras máquinas inteligentes já existia mesmo antes do surgimento dos primeiros computadores [5]. Hoje, tais máquinas dotadas de inteligência artificial permeiam cada vez mais a vida em sociedade, porque possuem, sobretudo, grande capacidade de adaptação e são aplicáveis numa ampla gama de problemas [5].

Por exemplo, pesquisas recentes vêm investigando a integração de ANNs em sistemas de internet das coisas (IoT - *internet of things*), para uma adaptação em tempo real do sistema às necessidades dos usuários e ao ambiente físico da IoT, otimizando os recursos disponíveis [6]. Outras aplicações em que ANNs estão sendo empregadas são a detecção de objetos [7], a predição das condições de trânsito [8] e até a detecção de COVID-19 em indivíduos [9].

Dado este cenário atraente e rico em oportunidades para pesquisas, decidiu-se estudar as ANNs neste trabalho de mestrado. Em particular, é investigado o comportamento das *echo state networks* (ESNs), um dos alicerces do que se tornou conhecido por *reservoir computing* (RC) [10–13]. De forma simplificada, pode-se dizer que RC refere-se a técnicas de treinamento de redes neurais recorrentes (RNNs - *recurrent neural networks*) e que uma ESN é, a grosso modo, uma RNN treinada por tais técnicas [10, 13].

O grande diferencial de uma ESN para uma RNN tradicional encontra-se no algoritmo de treinamento. Enquanto nesta é adaptado um grande número de parâmetros, naquela

somente um grupo mais restrito de parâmetros é otimizado pelo algoritmo de treinamento [13]. As ESNs mostraram-se capazes de resolver diversos problemas importantes, como o diagnóstico de falhas [14], a predição de consumo de energia nos já citados sistemas IoT [15], o rastreamento de *pitch* [16] e a detecção do início de uma nova nota [17] em música, a equalização de canais de comunicação [18] e a separação de sinais caóticos [19]. Essas três últimas áreas - música, sistemas de comunicação e sinais caóticos -, são temas de pesquisa deste mestrado.

Sinais caóticos são limitados em amplitude, aperiódicos e apresentam dependência sensível em relação às condições iniciais [20]. Isto é, pequenas alterações nas condições iniciais podem levar a comportamentos completamente distintos.

Seu estudo é relevante devido ao grande número de processos naturais que podem apresentar comportamento caótico [21], desde a evolução do universo [22] até funções cerebrais [23]. Além disso, pesquisas recentes vêm estudando diferentes alternativas de emprego de sinais caóticos em sistemas de comunicação [24], o que levanta interesse na busca por métodos para melhorar o desempenho de tais sistemas.

Sendo assim, dentre as três aplicações de RC estudadas neste trabalho, uma lida com a redução de ruído sobre um sinal caótico por meio do adequado treinamento e uso de uma ESN. A inspiração para este trabalho vem principalmente de [19], em que se utiliza uma ESN para separar uma mistura de dois sinais caóticos de tempo contínuo, gerados a partir do sistema de Lorenz [21].

Aqui, em vez de dois sinais caóticos de tempo contínuo, considera-se uma mistura de um sinal de ruído branco gaussiano e aditivo (AWGN - *additive white gaussian noise*) [25] com um sinal caótico de tempo discreto, obtido por meio do mapa tenda inclinada [20]. Para comparação, o filtro de Wiener (FW) [26] é empregado como *benchmark* para resolver o mesmo problema, por se tratar do método de filtragem linear ótimo.

Na segunda aplicação, estuda-se a equalização de canal do sistema de comunicação baseado em caos (SCBC) proposto em [27], inspirado no clássico trabalho de Wu e Chua [28]. Nesse SCBC, a mensagem a ser transmitida é binária e misturada a uma variável de estado, vinda de um mapa [20] do sistema transmissor por meio de uma função de codificação caótica. Aqui, emprega-se o mapa tenda inclinada [29] e a função de codificação soma [30]. O resultado dessa mistura é o sinal transmitido, que passa a ser realimentado no transmissor [31]. No receptor, utiliza-se o mesmo mapa e a função de codificação caótica inversa para realizar o sincronismo caótico e recuperar a mensagem transmitida.

São examinadas três situações distintas: na primeira, o canal adiciona apenas AWGN

ao sinal transmitido; na segunda, não há ruído e o canal equivale a um filtro com resposta ao impulso finita; e na terceira é levado em consideração ambos ruído e canal anteriores. Em todos os casos, empregam-se ESNs a fim de substituir todo o receptor do SCBC descrito em [27]. Para comparação, o uso do filtro de Wiener [26] para equalizar o canal, seguido da técnica de sincronismo caótico para recuperar a mensagem transmitida, é considerado como *benchmark*.

Por fim, visando mostrar a versatilidade do RC, a última aplicação lida com um cenário completamente distinto dos anteriores. Inspirado por [32], investiga-se a possibilidade de uma ESN emular a criatividade musical humana. Dada uma sequência de notas vindas de diversos trabalhos do famoso compositor e pianista polonês, radicado na França, Frédéric Chopin (⋆1810 - †1849), treina-se uma ESN a fim de, dada uma nota, predizer qual seria a próxima nota na sequência.

A escolha de obras musicais feitas para piano possibilitou a investigação de dois casos particulares. Num deles considera-se, para cada obra, todas as notas tocadas por uma das mãos do pianista em sequência, para só então considerar as notas tocadas em sequência pela outra mão. No outro considera-se, para cada obra, as notas na sequência em que são tocadas, sem o agrupamento por mão.

Apesar de haver um número considerável de trabalhos publicados em congressos e periódicos sobre o uso de ANNs para predição em música, como [33–35], não há muitos estudos sobre o uso específico das ESNs para tal fim, fator este que motivou ainda mais o estudo, brevemente explanado no parágrafo anterior, neste trabalho.

1.1 Objetivos

Os objetivos da dissertação são:

1. Determinar o desempenho de uma ESN para a mitigação de AWGN sobre um sinal caótico gerado pelo mapa tenda inclinada. A medida escolhida para quantificar este desempenho é o ganho de processamento;

2. Observar o quanto o desempenho anterior é influenciado pelo parâmetro do mapa tenda inclinada usado para gerar o sinal caótico;

3. Avaliar o desempenho do SCBC binário proposto em [27], fazendo uso da função de codificação soma e do mapa tenda inclinada, ao se empregar uma ESN para

estimar a mensagem transmitida dado o sinal recebido. Neste caso, o desempenho é quantificado por meio da taxa de erro de bit (BER - *bit error rate*);

4. Comparar os desempenhos obtidos nos itens anteriores com aqueles obtidos empregando-se um filtro de Wiener para a realização da mesma tarefa; e

5. Verificar, por meio do erro médio, da relação sinal-ruído (SNR - *signal-to-noise ratio*) e outros parâmetros pertinentes, o desempenho de uma ESN para predição de um sinal musical. Além da nota em si, isso significa também predizer quando cada nota é tocada e sua respectiva duração.

Este trabalho envolve conhecimentos sobre quatro áreas diferentes: redes neurais, sistemas dinâmicos, processamento digital de sinais e música. São incluídos os conceitos e resultados necessários para que a compreensão deste texto não seja prejudicada.

1.2 Metodologia

O trabalho começou com um estudo da bibliografia relacionada, principalmente [12,19, 20,32,36]. Então, foi feito um desenvolvimento teórico dos conceitos e principais resultados necessários para realização da etapa final que consistiu em simulações computacionais por meio de códigos desenvolvidos em MATLAB®[1] [37] e Python [38]. Vale ressaltar que foi utilizado a biblioteca mido [39] do Python para processamento de arquivos MIDI (*musical instrument digital interface*) [40]. Para as simulações em MATLAB®, não foi utilizada nenhuma biblioteca mais específica, mas sim apenas funções básicas dessa linguagem.

1.3 Contribuições originais e artigos

Considera-se que as contribuições originais desse trabalho de mestrado são:

A. Determinação da influência do parâmetro do mapa tenda inclinada, quando da mitigação de AWGN sobre um sinal caótico gerado pelo mesmo mapa, por meio do emprego de uma ESN;

B. Integração de uma ESN para equalização de canal do SCBC proposto em [27] em diferentes cenários; e

[1] MATLAB® é uma marca registrada de The MathWorks, Inc. (www.mathworks.com).

C. Verificação dos efeitos do agrupamento de notas pela mão correspondente que as toca no piano, quando do uso de uma ESN para predição de sinais musicais.

Com os resultados obtidos, submeteu-se um artigo [41] para o periódico internacional *Signal Processing*[2], que possui fator de impacto 4.729. A versão completa do artigo submetido, que encontra-se em revisão, está disponível no Anexo E.

Outros dois artigos foram apresentados e publicados nos anais de dois congressos nacionais:

- **A.L. Duarte**, M. Eisencraft *"Aplicação de Reservoir Computing para Filtragem de Sinais Caóticos Imersos em Ruído Branco Gaussiano,"* XL Congresso nacional de matemática aplicada e computacional - CNMAC 2021, São Carlos - SP, Brasil, Set. 2021. Disponível em[3].
- **A.L. Duarte**, M. Eisencraft, *"Aplicação de Reservoir Computing para Filtragem de Sinais Caóticos Imersos em Ruído,"* XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT'21, Fortaleza - CE, Brasil, Set. 2021. doi: 10.14209/sbrt.2021.1570724246

Esses trabalhos estão anexados, respectivamente, nos Anexos F e G deste relatório.

Um quarto artigo, com o título "Equalização de um sistema de comunicação baseado em caos usando *echo state networks*", foi submetido ao *XLII Congresso Nacional de Matemática Aplicada e Computacional*[4], a ser realizado de 18 a 22 de setembro de 2023 em Bonito - MS.

1.4 Estrutura da dissertação

A dissertação é composta por 6 capítulos, sendo um deles esta introdução.

No Capítulo 2, explicam-se os conceitos fundamentais sobre os sinais aqui utilizados e desenvolve-se a teoria de interesse sobre RC, voltada para ESNs.

No Capítulo 3, formula-se matematicamente o problema de redução de ruído em sinais caóticos usando-se ESN, discutem-se os resultados obtidos mediante simulações computacionais e apresentam-se as conclusões obtidas.

[2]https://www.sciencedirect.com/journal/signal-processing. Acesso em: 4 de maio de 2023.
[3]https://proceedings.sbmac.org.br/sbmac/article/view/134324. Acesso em: 4 de maio de 2023.
[4]http://www.cnmac.org.br. Acesso em: 4 de maio de 2023.

No Capítulo 4, detalha-se como é feita a integração da ESN ao SCBC estudado em três diferentes cenários, fazem-se discussões sobre os resultados obtidos por meio de simulações computacionais e apresentam-se as conclusões obtidas.

No Capítulo 5, apresenta-se formalmente o problema de predição de sinais musicais utilizando-se ESN, detalham-se os resultados obtidos e apresentam-se as conclusões alcançadas.

Por fim, no Capítulo 6, encontram-se as principais conclusões inferidas mediante o estudo teórico e prático conduzido para elaboração deste trabalho, além de sugestões de trabalhos futuros.

Capítulo 2

Revisão bibliográfica

Neste capítulo, revisitam-se alguns dos principais conceitos teóricos por trás das aplicações investigadas no restante do trabalho.

Inicia-se na Seção 2.1, em que são descritos os sinais e sistemas utilizados neste trabalho. Na Subseção 2.1.1, revisam-se as principais características de sinais caóticos e são apresentados o mapa tenda inclinada, largamente utilizado nos Capítulos 3 e 4, e o SCBC estudado no Capítulo 4. Na Subseção 2.1.2, encontram-se as definições e especificidades de sinais musicais, empregados no Capítulo 5.

Na Seção 2.2, é detalhado o funcionamento de uma ESN, parte do que é hoje conhecido por *reservoir computing* e principal tema abordado neste trabalho.

Na Seção 2.3, deduz-se a expressão para determinação dos coeficientes de um filtro de Wiener, utilizado como *benchmark* por se tratar do método ótimo de filtragem linear.

Finalmente, encerra-se o capítulo na Seção 2.4 com um breve resumo do que foi apresentado.

2.1 Sinais de interesse

Nesta seção, são introduzidos os sinais utilizados ao longo deste trabalho, a saber sinais caóticos de tempo discreto e sinais musicais.

2.1.1 Sinais caóticos

Sinais caóticos são limitados em amplitude, aperiódicos e apresentam dependência sensível em relação às condições iniciais (DSCI) [20], isto é, mudanças infinitesimais nas condições iniciais geram divergências exponenciais entre os sinais gerados.

Seu estudo é relevante devido ao grande número de processos naturais que apresentam

comportamento caótico [21], desde a evolução do universo [22] até funções cerebrais [23]. Dentre as aplicações que fazem uso de sinais caóticos, encontram-se a criptografia de imagens [42] e técnicas de modulação para comunicação [43].

2.1.1.1 Definições

No que concerne sinais caóticos neste trabalho, as seguintes definições são importantes [20]:

Definição 2.1. *Seja* $f(\cdot) : V \to V$*, com* $V \subseteq \mathbb{R}^M$ *e* $M \in \mathbb{N}^*$*. A equação*

$$\boldsymbol{x}(n+1) = f(\boldsymbol{x}(n)), \tag{2.1}$$

definida para $n \in \mathbb{N}$ *e* $\boldsymbol{x}(0) = \boldsymbol{x}_0$*, representa um sistema dinâmico de tempo discreto ou mapa.*

Definição 2.2. *O sinal gerado por* $f(\cdot)$*, também chamado de órbita, é dado por* $\boldsymbol{x}(n) = f^n(\boldsymbol{x}_0)$*, em que* $f^n(\cdot)$ *é a n-ésima aplicação sucessiva de* $f(\cdot)$*. O ponto* $\boldsymbol{x}_0$ *é chamado de condição inicial. Usa-se a notação* $\boldsymbol{x}(n; \boldsymbol{x}_0)$ *quando se quer destacar a condição inicial.*

Definição 2.3. *A trajetória do sinal* $\boldsymbol{x}(n; \boldsymbol{x}_0)$ *é o conjunto*

$$X_{\boldsymbol{x}_0} = \{\boldsymbol{x}(n) \mid n \in \mathbb{N}\}. \tag{2.2}$$

Definição 2.4. *O sinal* $\boldsymbol{x}(n)$ *é eventualmente periódico de período* τ*, se houver* $\boldsymbol{q} \in V$ *tal que* $f^{n+\tau}(\boldsymbol{q}) = f^n(\boldsymbol{q})$ *para todo* $n \geq n_*$*. No caso em que* $n_* = 0$*,* $\boldsymbol{x}(n)$ *é dito periódico de período* τ*.*

Definição 2.5. *O sinal* $\boldsymbol{x}(n)$ *é assintoticamente periódico se existir um sinal periódico* $\boldsymbol{y}(n)$ *tal que* $\lim_{n\to\infty} \|\boldsymbol{x}(n) - \boldsymbol{y}(n)\| = 0$*. Caso contrário, o sinal* $\boldsymbol{x}(n)$ *é aperiódico.*

O caso $M = 1$, em que os sinais envolvidos são escalares, é de especial interesse neste trabalho. Neste caso, o sinal gerado $x\,(n; x_0)$ é uma série temporal e tem-se a seguinte definição de expoente de Lyapunov:

Definição 2.6. *Se* $f(\cdot)$ *for unidimensional e diferenciável nos pontos da trajetória de* $x\,(n; x_0)$*, o produtório*

$$L\,(x_0) = \lim_{N\to\infty} \left(\prod_{n=0}^{N-1} |f^{'}\,(x(n))| \right)^{\frac{1}{N}} \tag{2.3}$$

é chamado de número de Lyapunov se o limite existir. Neste caso, o expoente de Lyapunov é calculado como

$$h\,(x_0) = \ln L\,(x_0)\,. \tag{2.4}$$

Esta definição pode ser estendida para o caso M-dimensional utilizando-se o jacobiano do mapa, resultando em M expoentes de Lyapunov associadas a uma órbita, um para cada dimensão. Veja mais detalhes em [20, Cap. 5], por exemplo.

Um resultado importante associado ao expoente de Lyapunov é que uma órbita que apresenta ao menos um de seus expoentes de Lyapunov positivo possui DSCI [20].

Desta forma, tem-se condições de se formalizar a definição de sinal caótico:

Definição 2.7. *Um sinal $\boldsymbol{x}(n;\boldsymbol{x}_0)$ é caótico se é limitado em amplitude, aperiódico e apresenta ao menos um expoente de Lyapunov positivo.*

Cabe destacar que as definições aqui apresentadas não são, de maneira alguma, exaustivas. Definições mais completas e com maior rigor matemático podem ser encontradas, por exemplo, em [20, 21].

Conforme Definições 2.1 e 2.2, sinais caóticos são determinísticos [21], sendo gerados a partir de uma expressão conhecida e característica de um determinado mapa. Na próxima subseção apresenta-se, como exemplo de gerador de sinais caóticos, o mapa tenda inclinada, que é usado para obter os sinais caóticos empregados neste trabalho.

2.1.1.2 Mapa tenda inclinada

O chamado mapa tenda inclinada é um mapa unidimensional ($M = 1$), com $V = (-1, 1)$, definido como

$$x(n+1) = f_I(x(n)) = \begin{cases} \frac{1-\alpha}{1+\alpha} + \frac{2}{1+\alpha}x(n), & -1 < x(n) < \alpha \\ \frac{1+\alpha}{1-\alpha} - \frac{2}{1-\alpha}x(n), & \alpha \leq x(n) < 1 \end{cases}. \tag{2.5}$$

Uma vez que a condição inicial satisfaça $x(0) = x_0 \in (-1, 1)$, $x(n)$ permanecerá limitado ao intervalo $(-1, 1)$ e dependerá apenas de seu valor no instante antecedente e do parâmetro $\alpha \in (-1, 1)$.

O mapa tenda inclinada e outros similares têm sido usados com sucesso em áreas como a criptografia de imagens [44] e a medição de sinais [45]. Na Fig. 2.1 a), encontra-se a curva $f_I(x)$ em função de x. Percebe-se o porquê de $f_I(\cdot)$ receber o nome de mapa tenda inclinada ao observar o formato dessa curva, que sempre muda de inclinação quando $x = \alpha$.

Pode-se demonstrar que o expoente de Lyapunov das órbitas de $f_I(\cdot)$ depende somente

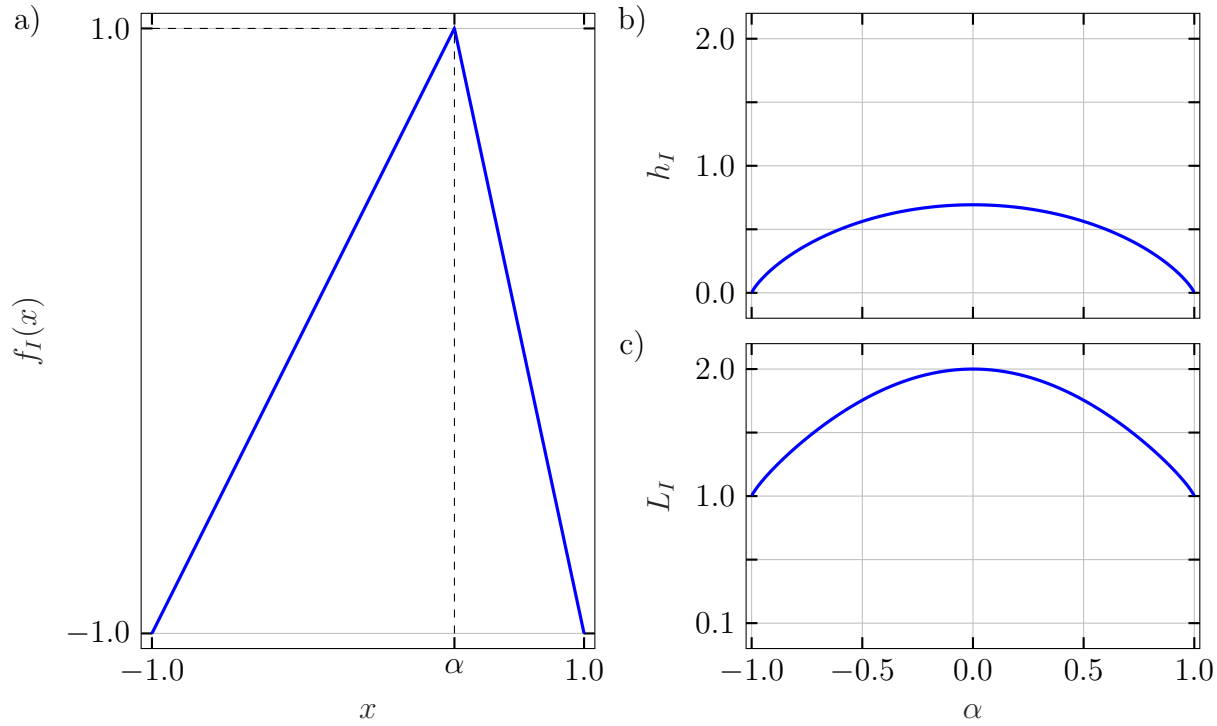

Figura 2.1: Características do mapa tenda inclinada $f_I(\cdot)$ (2.5): a) $f_I(x)$ em função de x; b) expoente de Lyapunov (2.6); c) número de Lyapunov.

do parâmetro α, sendo dado por [46]

$$h_I = \frac{1+\alpha}{2} \ln\left(\frac{2}{1+\alpha}\right) + \frac{1-\alpha}{2} \ln\left(\frac{2}{1-\alpha}\right), \tag{2.6}$$

para $-1 < \alpha < 1$. Na Fig. 2.1 b), apresenta-se h_I em função de α. Nota-se que h_I é sempre positivo e que seu máximo encontra-se em $\alpha = 0$, sendo igual a $\ln 2$.

Relembrando a Definição 2.6, toma-se a liberdade de omitir x_0 da notação pelo fato de, neste caso, o expoente de Lyapunov ser sempre o mesmo, independente da condição inicial. Da mesma definição, segue que o número de Lyapunov L_I das órbitas de $f_I(\cdot)$ pode ser prontamente obtido por meio de $L_I = e^{h_I}$. Na parte c) da Fig. 2.1, encontra-se a curva correspondente obtida.

A fim de ilustrar a propriedade de DSCI, na Fig. 2.2 encontram-se exemplos de trajetórias obtidas para diferentes condições iniciais x_0 e o mesmo parâmetro α. Nota-se que havendo uma diferença, mesmo que pequena, entre as condições iniciais, as trajetórias divergem entre si após um número suficiente de iterações, o que é típico de sinais caóticos. Quanto mais próximas as condições iniciais, maior o número aplicações de $f_I(\cdot)$ necessário para que a trajetória mude consideravelmente.

Um aspecto interessante do comportamento de $f_I(\cdot)$ com relação ao seu parâmetro α

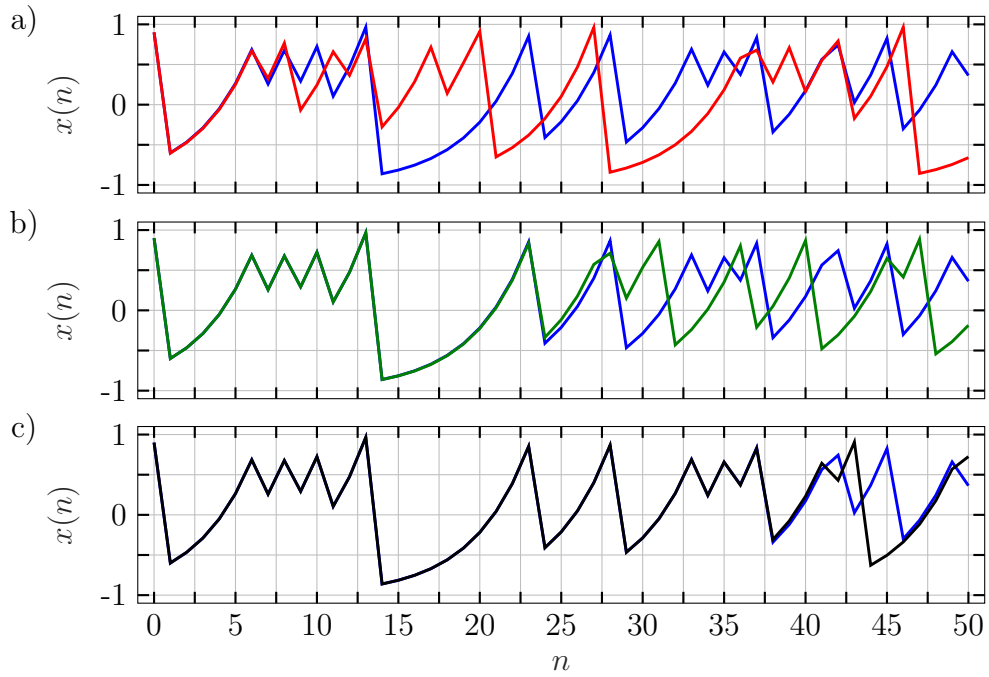

Figura 2.2: Exemplos de trajetórias de $x(n)$ para $\alpha = 0.5$ e condições iniciais ligeiramente distintas, ilustrando a DSCI: a) $x_0 = 0.9 + 10^{-3}$; b) $x_0 = 0.9 + 10^{-7}$; c) $x_0 = 0.9 + 10^{-11}$. As curvas na cor azul representam a trajetória para $x_0 = 0.9$.

é investigado em [46], em que demonstra-se que a sequência de autocorrelação (SAC) das órbitas de $f_I(\cdot)$ é dada por

$$R(k) = \frac{1}{3}\alpha^{|k|}, \tag{2.7}$$

sendo k o passo da correlação.

Na Fig. 2.3, apresentam-se curvas obtidas a partir de (2.7) para diferentes valores de $|\alpha|$. Dela, é possível concluir que quanto menor $|\alpha|$, mais impulsiva é a SAC, assemelhando-se àquela de um ruído branco. Por outro lado, à medida que $|\alpha|$ se aproxima de 1, mais plana torna-se a SAC.

Ainda em [46], é deduzido que a densidade espectral de potência (DEP) do mapa tenda inclinada é dada por

$$P(\omega) = \frac{1 - \alpha^2}{3(1 + \alpha^2 - 2\alpha\cos\omega)}. \tag{2.8}$$

Da análise de sua SAC, é esperado que para $|\alpha|$ próximo de 0, a DEP do mapa tenda inclinada seja quase plana, enquanto que para $|\alpha|$ próximo de 1, a DEP seja mais concentrada. Observando-se a Fig. 2.4, é possível constatar essa observação. Nela são apresentadas curvas obtidas a partir de (2.8) para diferentes valores de $|\alpha|$.

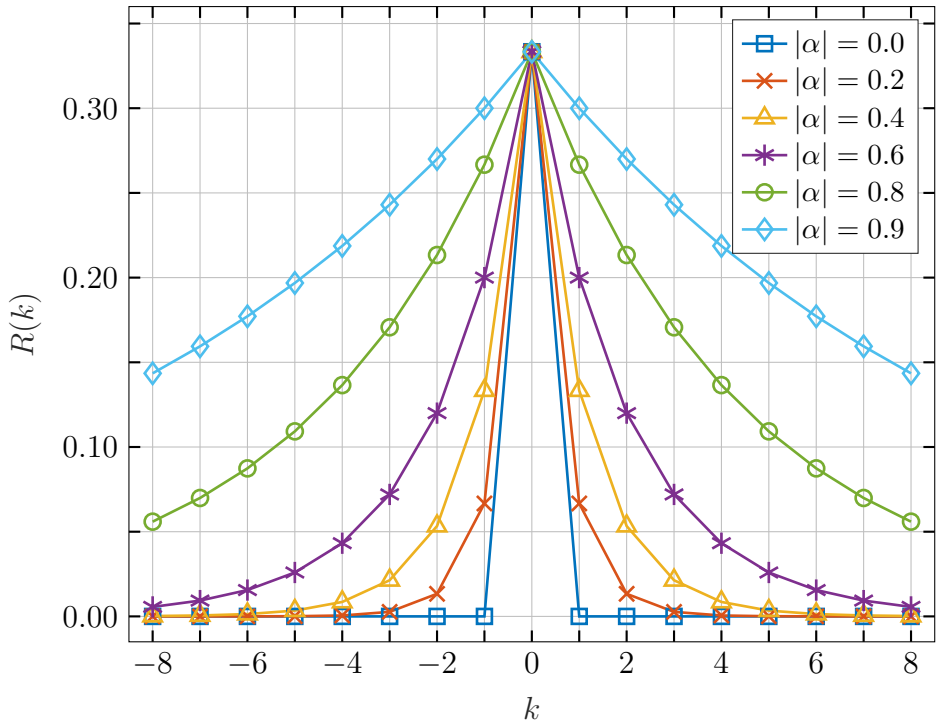

Figura 2.3: SAC (2.7) do mapa $f_I(\cdot)$ (2.5) para $|\alpha| = 0.0,\ 0.2, \ldots,\ 0.8,\ 0.9$.

2.1.1.3 Aplicação em comunicações

Na Fig. 2.5, apresenta-se o diagrama de blocos do SCBC de tempo discreto proposto em [27], adaptado do sistema de tempo contínuo baseado no esquema mestre-escravo proposto por Wu e Chua em seu trabalho divisor de águas [28]. No esquema mestre-escravo de tempo discreto, tem-se dois sistemas dinâmicos que podem ser escritos como

$$\boldsymbol{x}(n+1) = \boldsymbol{A}\boldsymbol{x}(n) + \boldsymbol{b} + f\left(\boldsymbol{x}(n)\right) \text{ e} \tag{2.9}$$

$$\boldsymbol{y}(n+1) = \boldsymbol{A}\boldsymbol{y}(n) + \boldsymbol{b} + f\left(\boldsymbol{x}(n)\right), \tag{2.10}$$

em que $\boldsymbol{x}(n) = \begin{bmatrix} x_1(n) & x_2(n) & \ldots & x_M(n) \end{bmatrix}^T$ e $\boldsymbol{y}(n) = \begin{bmatrix} y_1(n) & y_2(n) & \ldots & y_M(n) \end{bmatrix}^T$. Os parâmetros fixos são a matriz real $\boldsymbol{A}_{M\times M}$ e o vetor $\boldsymbol{b}_{M\times 1}$. A função $f(\cdot)$ é uma aplicação $\mathbb{R}^M \to \mathbb{R}^M$.

Nesta situação, o sistema dinâmico de tempo discreto descrito por (2.9) é chamado de mestre, enquanto que aquele representado por (2.10) recebe o nome de escravo. Tendo em mente a ideia de recuperar $\boldsymbol{x}(n)$ no escravo de algum modo, é conveniente definir o chamado erro de sincronização $\boldsymbol{e}(n) \triangleq \boldsymbol{y}(n) - \boldsymbol{x}(n)$. Com esta definição, subtraindo-se (2.10) de (2.9) obtém-se

$$\boldsymbol{e}(n+1) = \boldsymbol{A}\boldsymbol{e}(n). \tag{2.11}$$

Diz-se que o sistema escravo está completamente sincronizado com o sistema mestre se

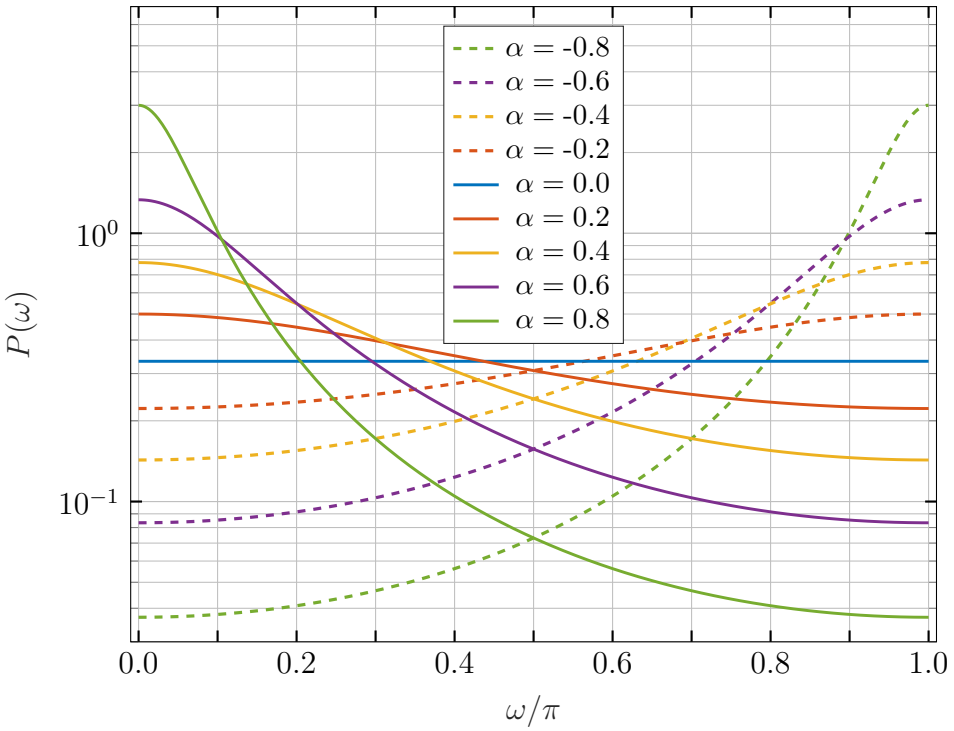

Figura 2.4: DEP (2.8) do mapa $f_I(\cdot)$ (2.5) para $\alpha = -0.8,\ -0.6,\ \ldots,\ 0.8$.

$\boldsymbol{e}(n) \to 0$ à medida que n aumenta. Neste caso, é possível recuperar perfeitamente $\boldsymbol{x}(n)$ no escravo. De (2.11), uma condição necessária e suficiente para que isso ocorra é [47]

$$|\lambda_i| < 1,\ i = 1, 2, \ldots, M, \tag{2.12}$$

em que λ_i é um autovalor da matriz $\boldsymbol{A}$.

Desta maneira, o conjunto mestre-escravo constitui um sistema de comunicação pois, se $\boldsymbol{x}(n)$ contém informação, esta pode ser recuperada no sistema escravo. A ideia-chave para fazer com que informação seja armazenada em $\boldsymbol{x}(n)$, consiste na codificação caótica

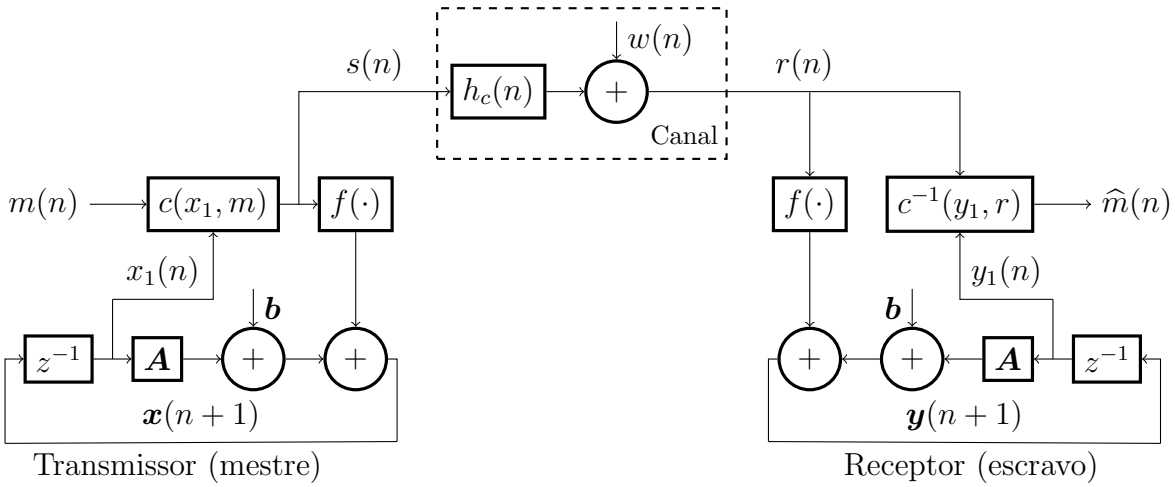

Figura 2.5: SCBC de tempo discreto estudado.

[27], tratada sucintamente a seguir.

2.1.1.4 Codificação e sincronismo caótico

Considere o sistema de comunicação apresentado na Fig. 2.5, em que a mensagem a ser transmitida $m(n) \in \{-1, 1\}$ é codificada pela componente caótica $x_1(n)$ de $\boldsymbol{x}(n)$. Neste trabalho, o sinal $\boldsymbol{x}(n)$ será gerado pelo mapa tenda inclinada $f_I(\cdot)$ (2.5), que é unidimensional ($M = 1$). Assim, $\boldsymbol{x}(n) = x(n) = x_1(n)$, $\boldsymbol{A} = 1$ e $\boldsymbol{b} = 0$.

A função de codificação $c(\cdot)$ pode ser qualquer função inversível. Neste trabalho, adota-se a função de codificação soma [30], em que o sinal transmitido é uma combinação linear da mensagem e da componente caótica, de modo que

$$s(n) = (1 - \gamma)x_1(n) + \gamma m(n), \tag{2.13}$$

em que $\gamma \in (0, 1)$ é um parâmetro que define os pesos relativos da componente caótica $x_1(n)$ e do sinal de informação $m(n)$ no sinal transmitido $s(n)$.

O sinal $r(n)$ que chega ao receptor pode ser escrito como

$$r(n) = s(n) * h_c(n) + w(n), \tag{2.14}$$

em que $h_c(n)$ é a resposta ao impulso do canal e $w(n)$ é AWGN. O receptor é formado pelo sistema escravo (2.10) - com a diferença de que o argumento de $f(\cdot)$ é $r(n)$ ao invés de $\boldsymbol{x}(n)$ - que sincroniza com o sistema mestre do transmissor. Para recuperar a mensagem, utiliza-se a função inversa de (2.13) com $r(n)$ e $y_1(n)$ no lugar de $s(n)$ e $x_1(n)$, respectivamente, o que leva a

$$\widehat{m}(n) = \frac{r(n) - (1 - \gamma)y_1(n)}{\gamma}. \tag{2.15}$$

Em [27], foi demonstrado que o receptor consegue recuperar perfeitamente a mensagem após um transiente necessário para sincronização num canal ideal, isto é, num canal em que $w(n) = 0$ e $h_c(n) = \delta(n)$. Por outro lado, em [31, 48] demonstra-se que o desempenho do sincronismo caótico é bastante sensível a condições não ideais do canal. Essa constatação é confirmada no estudo conduzido mais adiante no Capítulo 4, no qual compara-se a técnica de sincronismo caótico com uma ESN para equalização do canal do SCBC esquematizado na Fig. 2.5.

Na próxima subseção, apresenta-se a definição e discussão sobre sinais musicais no contexto deste trabalho.

2.1.2 Sinais musicais

Neste trabalho, um sinal musical $\boldsymbol{s}(n) \in \mathbb{R}^3$ é dado por

$$\boldsymbol{s}(n) = \begin{bmatrix} s_1(n) & s_2(n) & s_3(n) \end{bmatrix}^T, \tag{2.16}$$

em que $s_1(n)$ diz respeito a altura de uma nota musical e $s_2(n)$ e $s_3(n)$ representam, nesta ordem, os instantes de início e término de execução da mesma nota. A origem e a explicação de cada uma das informações encontradas em $\boldsymbol{s}(n)$ são detalhadas nesta seção.

As componentes de $\boldsymbol{s}(n)$ advém de arquivos MIDI [40]. Este é um sistema que permite que instrumentos musicais eletrônicos, como um teclado ou uma bateria eletrônica, enviem instruções para computadores e vice-versa [40]. As instruções são registradas por meio de mensagens em arquivos MIDI. Elas carregam todas as informações necessárias para que dispositivos MIDI comuniquem-se entre si e sobre os atributos musicais do que é tocado no instrumento.

Dentre os atributos definidos por essas mensagens, detém importância fundamental para este trabalho a altura da nota tocada, seu instante de início e instante de término. Informações sobre outros atributos que também podem ser definidos por mensagens, como a intensidade da nota tocada e o *pitch bend*, podem ser encontradas em [40].

A altura da nota diz respeito à propriedade do som ser grave, médio ou agudo [49]. Já a intensidade está relacionada com o volume percebido. Ao tocar uma corda de um violão com mais força, a amplitude de sua vibração é maior. Consequentemente o volume do som produzido também o é [49].

Na Fig. 2.6, mostra-se uma captura de tela de um arquivo MIDI aberto com o software gratuito MidiEditor [50]. Neste programa, as notas tocadas são representadas por retângulos, cujas larguras são proporcionais às suas durações. A posição horizontal do lado esquerdo de um retângulo indica o início da execução de uma nota, enquanto que a posição horizontal do lado direito, o término. A posição vertical de um retângulo indica a altura da nota tocada.

Além disso, existe a representação de um teclado ao lado esquerdo para auxiliar na identificação das notas. A notação CN, com N $= -1, 0, \ldots, 9$, sobre certas teclas neste teclado indica que a tecla correspondente emite a nota dó[1] na oitava N, conforme Tabela 2.1.

[1]Na notação ABC [51] usada por computadores, representam-se as notas lá, si, dó, ré, mi, fá e sol por A, B, C, D, E, F e G.

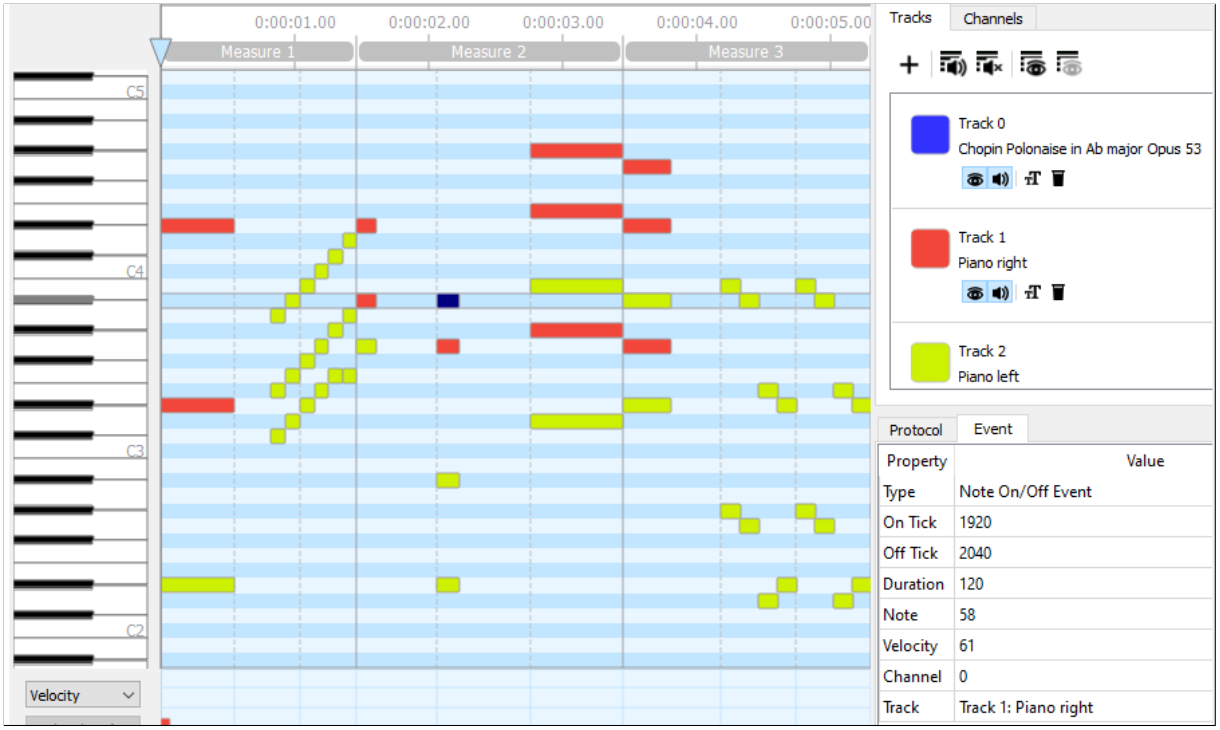

Figura 2.6: Captura de tela de um arquivo MIDI aberto no software MidiEditor.

Os símbolos ♭ e ♯ indicam os acidentes musicais bemol e sustenido, respectivamente [49]. Em cada oitava há sempre 5 acidentes, que correspondem sempre às teclas pretas num piano, conforme indicado na Fig. 2.7. Nela, apresentam-se sobre as teclas brancas do piano as notas correspondentes. Como exemplo, a tecla preta entre o dó na segunda oitava (C2) e o ré na segunda oitava (D2), corresponde à nota dó sustenido (C♯2), que também pode ser denominada ré bemol (D♭2).

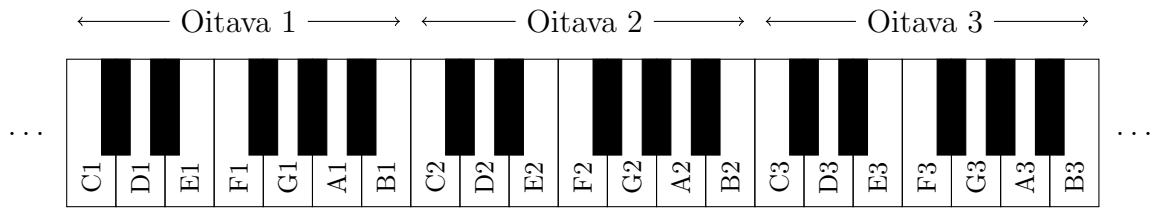

Figura 2.7: Ilustração da relação entre a notação das notas da Tabela 2.1 e as teclas de um piano.

Na Figura 2.6, retângulos de cores diferentes pertencem a *tracks* distintos, que são usados para que o usuário organize e gerencie suas gravações como preferir. Os arquivos MIDI utilizados neste trabalho foram obtidos em [52] e estão listados no Anexo C. Quase a totalidade deles possui apenas dois *tracks*: um correspondente ao tocado pela mão direita no teclado e outro, para a mão esquerda. As exceções são os arquivos referentes às obras Opus 25 No. 1, Opus 35 No. 2, Opus 24, No. 8 e Opus 24, No. 9 que possuem três *tracks*:

Tabela 2.1: Mapeamento das notas musicais em números inteiros nos arquivos MIDI.

Oitava	Dó (C)	Dó♯/Ré♭ (C♯/D♭)	Ré (D)	Ré♯/Mi♭ (D♯/E♭)	Mi (E)	Fá (F)	Fá♯/Sol♭ (F♯/G♭)	Sol (G)	Sol♯/Lá♭ (G♯/A♭)	Lá (A)	Lá♯/Si♭ (A♯/B♭)	Si (B)
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127	-	-	-	-

dois para o tocado pela mão direita e um para o tocado pela mão esquerda.

As músicas são organizadas em compassos [49] separados por linhas verticais contínuas de cor cinza, enquanto que a pulsação [49] da música é separada por linhas verticais traçejadas, também de cor cinza. Apesar de existir uma barra de tempo em segundos no topo da Fig. 2.6, as mensagens MIDI trabalham com *ticks* como unidade de tempo.

Existe sempre uma mensagem definindo a quantidade de *ticks* por pulso, como a apresentada na primeira linha do trecho de mensagens a seguir. São as linhas iniciais de um arquivo MIDI correspondente a obra Opus 7 No.1 de Frédéric Chopin.

```
MidiFile(type=1, ticks_per_beat=480, tracks=[
MidiTrack([
MetaMessage('track_name', name='Chopin Mazurka Opus 7', time=0),
```

Todos os arquivos MIDI usados neste trabalho possuem 480 *ticks* por pulso. A nota representada pelo retângulo destacado em azul na Fig. 2.6 corrobora isso: ela começa ao final do quarto pulso e o valor de seu atributo *on tick*, que indica em qual *tick* inicia-se a execução da nota, é de $4(480) = 1920$, como pode ser visto no canto inferior direito da Fig. 2.6. Somando-se os valores dos atributos *on tick* e *duration*, obtém-se o valor do atributo *off tick*, que indica em qual *tick* a execução da nota é interrompida. No exemplo da Fig. 2.6, $1920 + 120 = 2040$.

Se for desejado saber a duração em segundos de um *tick*, basta saber o andamento da música [49]. Numa música com andamento de 120 bpm (batidas por minuto), cada pulso tem duração de 1/120 minutos = 0.5 s. Como cada pulso possui 480 *ticks*, cada um deles corresponde a $0.5/480 = 1/960$ s.

Além dos atributos *on tick*, *off tick* e *duration*, que dizem respeito ao instante de início, de término de execução e a duração da nota em *ticks*, ao lado direito da Fig. 2.6 encontram-se ainda os atributos *note* e *velocity* da nota representada pelo retângulo destacado em azul marinho, detalhados nos próximos parágrafos.

Quando é tocada uma nota num teclado eletrônico conectado a um computador usando um software MIDI, uma mensagem do tipo *note-on* é enviada e quando para de se tocar a nota, uma mensagem do tipo *note-off*[2] é enviada ao software, que interrompe a execução da nota. Neste caso, o teclado é chamado de controlador MIDI e o computador de receptor MIDI [40].

Abaixo, apresentam-se como exemplo algumas mensagens tipo *note-on*.

```
Message('note_on', channel=0, note=70, velocity=71, time=0),
Message('note_on', channel=0, note=70, velocity=0, time=240),
Message('note_on', channel=0, note=72, velocity=66, time=120),
Message('note_on', channel=0, note=72, velocity=0, time=120),
```

O primeiro atributo da mensagem indica seu tipo. Neste caso, todas são *note-on*. Uma lista completa dos tipos de mensagem pode ser encontrada em [39].

O segundo atributo indica o canal em que é tocada a nota. O canal indica um instrumento distinto sendo tocado e é um número inteiro entre 0 e 15. Ou seja, um arquivo MIDI pode conter até 16 canais. Aqui, a repetição do valor 0 nas mensagens indica que há um único instrumento sendo tocado.

O terceiro atributo indica qual a altura da nota sendo tocada. Ele é um número inteiro entre 0 e 127 e sua relação com as notas musicais é sintetizada na Tabela 2.1, em que uma oitava refere-se a um conjunto das doze notas musicais: as sete notas naturais e os cinco acidentes musicais [49]. Por exemplo, o tom de 440 Hz que se escuta ao retirar um telefone de seu gancho corresponde a nota lá da oitava número 4, representada por A4 na notação ABC [51], e pelo inteiro 69 em arquivos MIDI. Nesta mesma oitava, encontra-se o dó central C4 do piano, representado pelo inteiro 60.

O quarto atributo refere-se a intensidade da nota tocada que é um inteiro entre 0 e 127. Um valor alto, indica uma nota tocada de maneira brusca, enquanto que um valor baixo indica uma nota tocada de forma suave. Um valor 0, caracteriza um evento *note-off*.

O quinto e último atributo indica o intervalo de tempo em *ticks* decorrido desde a

[2]Na verdade, uma mensagem *note-off* nada mais é que uma mensagem *note-on* com o atributo *velocity* nulo.

última mensagem. Por exemplo, a segunda linha indica que o evento ocorre 240 *ticks* após o evento representado na primeira linha.

A próxima subseção esclarece como são extraídas as informações de interesse dos arquivos MIDI neste trabalho.

2.1.2.1 Processamento dos arquivos MIDI

Para se obter sinais musicais a partir de arquivos MIDI, utiliza-se a linguagem de programação Python [38] e sua biblioteca mido [39]. Esta biblioteca permite trabalhar com mensagens e portas MIDI.

Por meio do código apresentado no Anexo A, a altura e os instantes de início e término de cada nota são extraídos das mensagens pertinentes de cada arquivo MIDI e registrados na forma altura-início-término num arquivo de texto de nome *bancoProcessado.txt*. Cada linha do arquivo é dedicada a uma nota distinta. Então, utilizando-se os seguintes comandos no MATLAB

```
arqTexto = fopen('bancoNotas.txt'); %abre o arquivo de texto
S = fscanf(arqTexto,'%d',[3,inf])'; %extrai as informacoes
fclose(fid); %fecha o arquivo de texto
S(:,2) = -s(:,2); %corrige o sinal das amostras s2(n)
S(:,3) = -s(:,3); %corrige o sinal das amostras s3(n)
S = S.'; %acerta as dimensoes
```

obtém-se uma matriz $\boldsymbol{S}$ contendo todas as amostras de $\boldsymbol{s}(n)$. Respeitando-se os instantes indicados na Tabela C.1, tem-se

$$\boldsymbol{S} = \begin{bmatrix} s_1(-199) & s_1(-198) & \dots & s_1(85155) \\ s_2(-199) & s_2(-198) & \dots & s_2(85155) \\ s_3(-199) & s_3(-198) & \dots & s_3(85155) \end{bmatrix}. \tag{2.17}$$

Na Fig. D.1, encontra-se um exemplo de trecho dos resultados obtidos mediante uso do código em Python disponível no Anexo A. Na parte a), apresenta-se um extrato da partitura do prelúdio Opus 24, No. 18 de Frédéric Chopin. Ele corresponde somente ao que é tocado pela mão direita do pianista no compasso 13 desta obra. Na parte b) da mesma figura, é apresentado uma captura de tela do software MidiEditor, correspondente ao trecho considerado da obra. Nela, somente o *track* referente às notas tocadas pela mão direita no piano está representado, enquanto as demais notas estão ocultas.

Nas Fig. D.1 c) e d), encontram-se as informações de interesse extraídas. Para elu-

cidação, na parte b) observa-se que a segunda nota tocada é um C4. Conforme apresentado na Tabela 2.1, essa nota é um dó na quarta oitava, representado pelo inteiro 60. Da parte c), está claro que o segundo ponto está em 60.

Pelo fato das mensagens serem ordenadas por *track*, a listagem das informações extraídas no arquivo de texto ocorre por *tracks* também. Isto significa que, se o arquivo MIDI apresentado na Fig. 2.6 for processado, serão primeiramente consideradas todas as notas ilustradas em vermelho, referentes ao *track* 1. Só então as notas na cor verde, registradas no *track* 2, serão processadas.

Para elucidar a ordem de processamento das notas, considere o trecho de arquivo MIDI apresentado na Fig. 2.8. Seja $\{\boldsymbol{s}(1), \boldsymbol{s}(2), \ldots, \boldsymbol{s}(8)\}$ a sequência de amostras obtidas após o processamento.

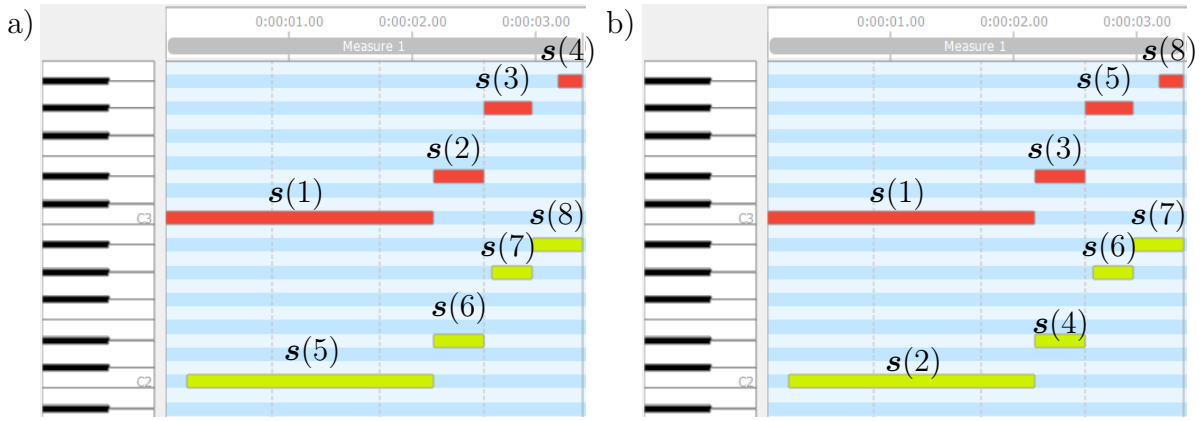

Figura 2.8: Diferenças entre processamentos dos arquivos MIDI: a) notas agrupadas por *track*; b) notas na sequência de execução.

Quando usado, o código apresentado no Anexo A processa primeiro todas as notas do *track* referente à mão direita do pianista, para depois processar as do outro referente à mão esquerda. As amostras obtidas estão indicadas na parte a) da Fig. 2.8.

No Capítulo 5, encontra-se um estudo no qual reordena-se as notas obtidas na sequência em que são tocadas de fato, ignorando-se o agrupamento por *tracks* dos arquivos MIDI. Neste caso, a reordenação leva à sequência apresentada na parte b) da Fig. 2.8.

Este estudo tem como objetivo verificar se as diferentes ordenações influenciam no desempenho da predição da ESN e, em caso positivo, para qual delas o desempenho é melhor.

Na próxima seção, trata-se o assunto mais importante e que faz parte do título dado à este trabalho de mestrado.

2.2 A técnica de *reservoir computing* empregada: *echo state networks*

Numa ANN, os elementos fundamentais são chamados de nós, ou simplesmente neurônios, por analogia com o modelo biológico de rede neural. Os nós recebem e processam informação e possuem um estado interno, também chamado de ativação, que é uma função da entrada atual e das passadas [1].

Tipicamente, um nó transmite sua ativação para outros nós através de conexões, cada uma possuindo um peso característico, a ser determinado por meio de um algoritmo de treinamento [1]. Dependendo de como estas ativações escoam pela rede, as ANNs são classificadas como *feedforward*, em que as ativações trafegam somente no sentido de entrada para saída da rede, e as recorrentes, nas quais existe ao menos um caminho cíclico de conexões entre os nós [36].

Apesar de se acreditar que possuam um grande potencial latente a ser explorado, devido à sua capacidade de aproximar sistemas dinâmicos e, assim como o cérebro, possuir caminhos cíclicos [13], as RNNs são difíceis de serem treinadas utilizando os métodos clássicos, baseados no gradiente descendente [53], como a retropropagação de erro [53]. Este fato decorre principalmente por causa que os caminhos cíclicos, apesar de possibilitarem que uma RNN desenvolva uma dinâmica de ativação temporal auto-sustentável [13] - razão pela qual RNNs aproximam sistemas dinâmicos -, levam a bifurcações no treinamento [54]. Isto significa que mudanças infinitesimais nos parâmetros da RNN podem levar à mudanças drásticas em seu comportamento [55].

Sendo assim, não é conveniente utilizar os métodos clássicos para treinar RNNs, pois muitas vezes a convergência não é garantida [54]. Outros empecilhos estão relacionados ao alto custo computacional envolvido, ao grande número de parâmetros para se ajustar [13] - como a otimização do peso de todas as conexões -, dentre outros, como tratado em [56].

A arquitetura de uma RNN típica é apresentada na Figura 2.9[3]. Nela, destacam-se por cores distintas as três principais partes, formadas por conjuntos de nós, de uma RNN. As camadas em vermelho e verde indicam o conjunto dos nós de entrada e saída, respectivamente. A parte azul representa o emaranhado de nós internos e suas conexões, que formam os caminhos cíclicos que dão nome as RNNs.

Dado um sinal desejado, como a fala de um locutor [57], a atividade elétrica atrial de

[3]Podem ainda existir conexões de realimentação dos nós de saída de volta para os nós internos da RNN. Neste trabalho, essas conexões não são consideradas.

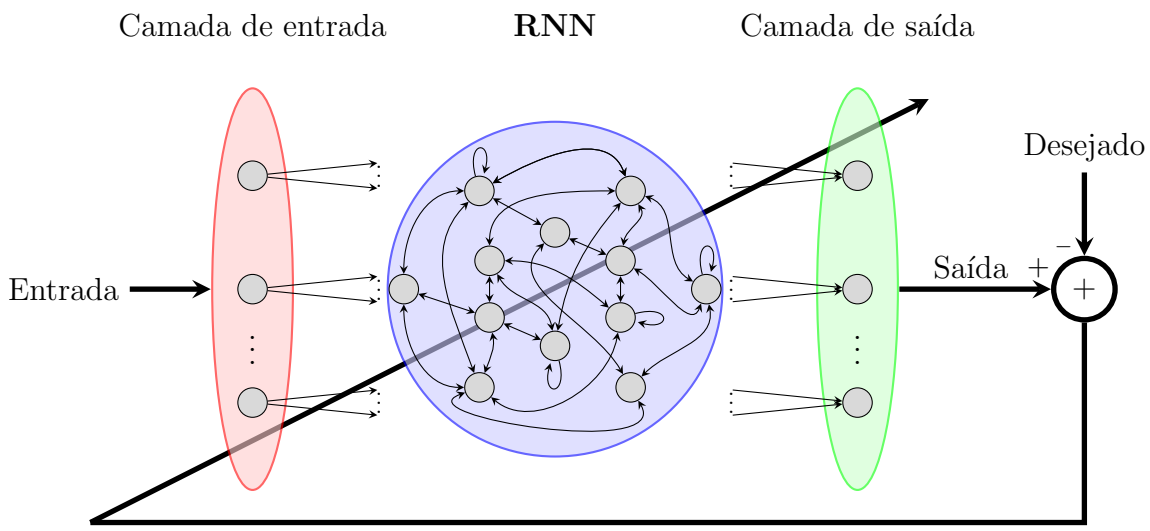

Figura 2.9: Arquitetura de uma RNN, em que todas as conexões (entrada-RNN, internas e RNN-saída) são treinadas.

um paciente [58] ou o sinal transmitido por um satélite [59], RNNs podem ser treinadas de modo supervisionado ou não, para que sua saída aproxime o sinal de interesse [1]. Em geral, métodos de treinamento não supervisionado são mais difíceis de serem executados [1] e não são tratados aqui.

Buscando amenizar os problemas intrínsecos ao projeto de RNNs, no início dos anos 2000 foram propostas, de modo independente, novas e análogas formas de treinar e projetar RNNs: as *liquid state machines*, por Wolfgaang Maass e Henry Markram [60], e as ESNs, por Herbert Jaeger [10]. Estes trabalhos, juntamente com suas ramificações conseguintes, receberam o nome de *reservoir computing* [13].

O RC tem como essência a ideia de que somente os pesos das conexões RNN-saída devem ser ajustados [10], enquanto que aqueles das conexões entre nós internos - ou simplesmente a RNN em si, nomeada então como *reservoir* - são gerados aleatoriamente e permanecem inalterados durante o treinamento, o mesmo ocorrendo com os pesos das conexões entrada-RNN [10].

O RC tem sido empregado com sucesso em diversas áreas de processamento de sinais, dentre elas a separação [19] e a predição [61] de sinais caóticos, bem como a geração de melodias [62] e a classificação de gênero [63] em música. Mesmo havendo alterações e extensões da ideia original de ESNs, como em [64] e [65], o método clássico introduzido em [10] é ainda extremamente atraente devido sua simplicidade e baixo consumo computacional [12]. Deste modo, a técnica de RC adotada neste trabalho se dá por meio do uso de ESNs baseadas em [10]. Na Fig. 2.10, representa-se esquematicamente a estrutura

da ESN[4] que é detalhada nas próximas subseções. Ela consiste de (i) uma camada de entrada, (ii) o chamado *reservoir* e (iii) uma camada de saída. Cada uma dessas partes é composta por nós que processam informação, ligados entre si por meio de conexões, por meio das quais ocorre o tráfego de informação [10, 12].

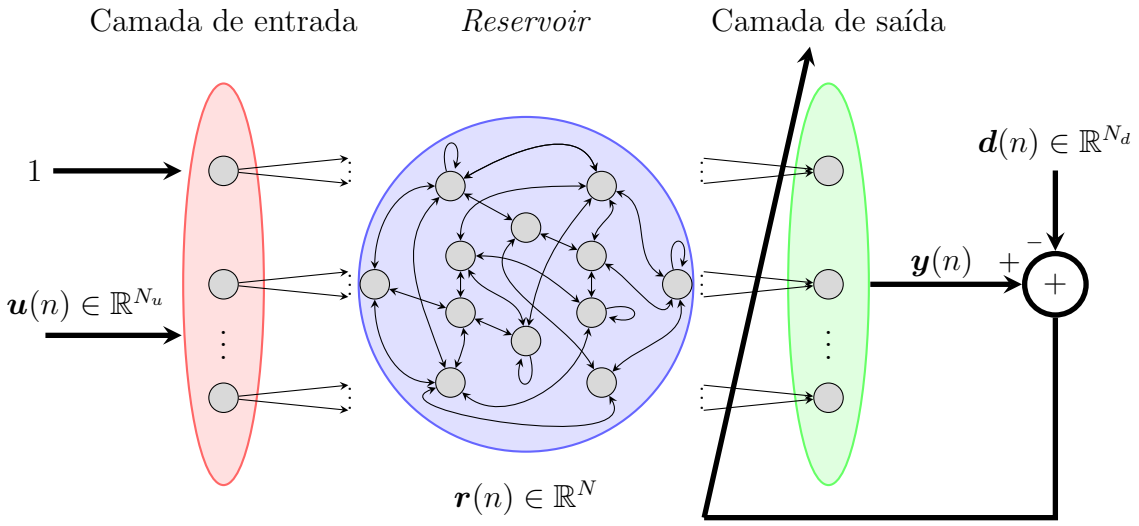

Figura 2.10: Arquitetura de uma ESN, indicando os sinais $\boldsymbol{u}(n)$, $\boldsymbol{y}(n)$, $\boldsymbol{d}(n)$ e seu estado $\boldsymbol{r}(n)$.

Dado um sinal de entrada $\boldsymbol{u}(n) \in \mathbb{R}^{N_u}$ e um sinal desejado $\boldsymbol{d}(n) \in \mathbb{R}^{N_d}$, com $n = 1, 2, \ldots, L$, sendo L o número de amostras de $\boldsymbol{d}(n)$ disponíveis para treinamento, a tarefa da ESN é adaptar os pesos das conexões *reservoir*-saída, de modo que o sinal de saída $\boldsymbol{y}(n)$ seja tal que uma medida de erro $E(\boldsymbol{y}, \boldsymbol{d})$ é minimizada [13]. Normalmente $E(\cdot)$ é tomada como a raiz do erro quadrático médio (RMSE - *root mean square error*) ou sua versão normalizada [13], de modo que

$$E\left(\boldsymbol{y}, \boldsymbol{d}\right) = \frac{1}{N_d} \sum_{k=1}^{N_d} \sqrt{\frac{1}{L} \sum_{n=1}^{L} \left(y_k(n) - d_k(n)\right)^2}. \tag{2.18}$$

As subseções a seguir explicam de formada detalhada o funcionamento da camada de entrada, do *reservoir* e da camada de saída das ESNs.

2.2.1 Camada de entrada

A camada de entrada é responsável por realizar um pré-processamento do sinal $\boldsymbol{u}(n) \in \mathbb{R}^{N_u}$, de modo a controlar a não linearidade da rede [12].

[4]Podem ainda existir conexões de realimentação dos nós de saída de volta para os nós internos da ESN. Neste trabalho, essas conexões não são consideradas.

Ela é formada por $N_u + 1$ nós e cada um desses nós é ligado ao *reservoir* por meio de N conexões no máximo - um peso nulo é equivalente a ausência de uma conexão -, sendo N o número de nós do *reservoir*. Os pesos dessas conexões são as entradas w_{ij}^{in} reais da matriz $\boldsymbol{W}_{N\times N_u+1}^{\text{in}}$, obtidas a partir de distribuições uniformes[5] [10]. As entradas da primeira coluna seguem uma distribuição uniforme em $[-p, p]$, enquanto que as demais entradas seguem uma distribuição uniforme em $[-q, q]$. Ambos p e q são parâmetros escolhidos mediante uma rotina de seleção de parâmetros, melhor descrita nos Capítulos 3, 4 e 5

Dentre os $N_u + 1$ nós, N_u recebem como entrada os elementos $u_k(n)$, com $k = 1, 2, \ldots, N_u$, do vetor $\boldsymbol{u}(n) = \begin{bmatrix} u_1(n) & u_2(n) & \ldots & u_{N_u}(n) \end{bmatrix}^T$. Cada um desses nós tem como estado no instante n uma das entradas $u_k(n)$. O nó restante possui estado fixo igual a unidade, que serve como viés.

Os estados dos nós de entrada são transmitidos aos nós do *reservoir* por meio do produto $\boldsymbol{W}^{\text{in}} \begin{bmatrix} 1 & \boldsymbol{u}(n) \end{bmatrix}^T$ [10]. Conforme detalhado na próxima subseção, este produto é parte do argumento da função $\tanh(\cdot)$, presente na equação de atualização do vetor de estados da rede (2.21). Como $\tanh(\cdot)$ possui comportamento aproximadamente linear para argumentos próximos a zero e não linear para argumentos de maior magnitude, segue que $\boldsymbol{W}^{\text{in}}$ efetivamente determina o nível de não linearidade por trás da operação da ESN.

2.2.2 *Reservoir*

O chamado *reservoir* é formado por N nós conectados entre si e ligados tanto aos nós da camada de entrada, quanto aos nós da camada de saída. Os pesos das conexões internas são as entradas w_{ij} reais da matriz $\boldsymbol{W}_{N\times N}$, obtidas aleatoriamente.

Primeiramente, utiliza-se uma distribuição uniforme em $[-1, 1]$ para gerar as entradas de uma matriz auxiliar $\boldsymbol{W}_{N\times N}^{\text{aux}}$. Então, o raio espectral [66] λ_{aux} desta é calculado. Daí, obtém-se a matriz de entrada por meio de

$$\boldsymbol{W} = \lambda \left(\frac{\boldsymbol{W}^{\text{aux}}}{\lambda_{\text{aux}}} \right), \tag{2.19}$$

em que λ é um parâmetro. Como $\boldsymbol{W}^{\text{aux}}/\lambda_{\text{aux}}$ possui raio espectral unitário, segue que $\boldsymbol{W}$ tem raio espectral λ. Este parâmetro também é obtido mediante a rotina de seleção de parâmetros, melhor descrita nos Capítulos 3, 4 e 5.

Num dado instante n, os nós internos são caracterizados por estados $r_k(n) \in \mathbb{R}$, com

[5] Apesar de também serem usadas outras distribuições [10], neste trabalho usa-se sempre a uniforme.

$k = 1, 2, \ldots, N$. Quando o produto $\boldsymbol{W}^{\text{in}} \begin{bmatrix} 1 & \boldsymbol{u}(n) \end{bmatrix}^T$ adentra no *reservoir*, cada um de seus N nós transmite seu estado $r_k(n)$ para os outros nós do *reservoir* aos quais está ligado, por meio do produto $\boldsymbol{W}\boldsymbol{r}(n-1)$ e também para os nós da camada de saída aos quais está ligado, por meio do produto $\boldsymbol{W}^{\text{out}}\boldsymbol{r}(n)$, melhor explicado na próxima seção. Então, os estados dos nós internos alteram-se para $r_k(n+1)$.

Os estados dos nós do *reservoir* compõem o vetor de estados internos da rede, dado por

$$\boldsymbol{r}(n) = \begin{bmatrix} r_1(n) & r_2(n) & \ldots & r_N(n) \end{bmatrix}^T. \tag{2.20}$$

A cada instante de tempo n, o vetor de estados internos é atualizado mediante a equação do modelo *leaky-integrator*[6] [10]

$$\boldsymbol{r}(n) = (1-a)\,\boldsymbol{r}(n-1) + a \tanh\left(\boldsymbol{W}^{\text{in}} \begin{bmatrix} 1 \\ \boldsymbol{u}(n) \end{bmatrix} + \boldsymbol{W}\boldsymbol{r}(n-1)\right), \tag{2.21}$$

em que a função $\tanh(\cdot)$ é aplicada elemento a elemento e a é o chamado parâmetro de *leakage*. Este também é determinado mediante a rotina de seleção de parâmetros melhor descrita nos Capítulos 3, 4 e 5.

Em alguns casos, pode-se ainda adicionar conexões da saída do *reservoir* para sua entrada [10], o que acarreta na presença de mais um termo no argumento de $\tanh(\cdot)$ em (2.21). Neste trabalho, considera-se que não há essas conexões. Também cabe destacar que, apesar de ser a escolha mais comum, outras sigmoides - i.e. funções que apresentam um gráfico em formato próximo ao da letra *s* - também podem ser utilizadas no lugar de $\tanh(\cdot)$ [10].

A equação (2.21) permite entender melhor como os estados dos nós da camada de entrada são transmitidos aos nós do *reservoir*, assim como os estados de cada nó do *reservoir* são transmitidos aos demais. Primeiramente, note que o produto $\boldsymbol{W}^{\text{in}} \begin{bmatrix} 1 & \boldsymbol{u}(n) \end{bmatrix}^T$

[6] ESNs que empregam tal modelo são conhecidas por *leaky-integrator* ESNs (LI-ESNs) [12].

resulta em

$$\boldsymbol{W}^{\text{in}}\begin{bmatrix}1\\ \boldsymbol{u}(n)\end{bmatrix}=\begin{bmatrix} w_{11}^{\text{in}} & w_{12}^{\text{in}} & w_{13}^{\text{in}} & \dots & w_{1(N_u+1)}^{\text{in}}\\ w_{21}^{\text{in}} & w_{22}^{\text{in}} & w_{23}^{\text{in}} & \dots & w_{2(N_u+1)}^{\text{in}}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ w_{N1}^{\text{in}} & w_{N2}^{\text{in}} & w_{N3}^{\text{in}} & \dots & w_{N(N_u+1)}^{\text{in}}\end{bmatrix}\begin{bmatrix}1\\ u_1(n)\\ \vdots\\ u_{N_u}(n)\end{bmatrix}=$$

$$=\begin{bmatrix} w_{11}^{\text{in}}+w_{12}^{\text{in}}u_1(n)+w_{13}^{\text{in}}u_2(n)+\cdots+w_{1(N_u+1)}^{\text{in}}u_{N_u}(n)\\ w_{21}^{\text{in}}+w_{22}^{\text{in}}u_1(n)+w_{23}^{\text{in}}u_2(n)+\cdots+w_{2(N_u+1)}^{\text{in}}u_{N_u}(n)\\ \vdots\\ w_{N1}^{\text{in}}+w_{N2}^{\text{in}}u_1(n)+w_{N3}^{\text{in}}u_2(n)+\cdots+w_{N(N_u+1)}^{\text{in}}u_{N_u}(n)\end{bmatrix}, \tag{2.22}$$

de modo que sua k-ésima linha é dada por

$$w_{k1}^{\text{in}}+\sum_{m=1}^{N_u}w_{k(m+1)}^{\text{in}}u_m(n). \tag{2.23}$$

O produto $\boldsymbol{W}\boldsymbol{r}(n-1)$ que compõe a segunda parcela do argumento de $\tanh(\cdot)$ em (2.21) é dado por

$$\boldsymbol{W}\boldsymbol{r}(n)=\begin{bmatrix} w_{11} & w_{12} & \dots & w_{1N}\\ w_{21} & w_{22} & \dots & w_{2N}\\ \vdots & \vdots & \ddots & \vdots\\ w_{N1} & w_{N2} & \dots & w_{NN}\end{bmatrix}\begin{bmatrix}r_1(n-1)\\ r_2(n-1)\\ \vdots\\ r_N(n-1)\end{bmatrix}$$

$$=\begin{bmatrix} w_{11}r_1(n-1)+w_{12}r_2(n-1)+\cdots+w_{1N}r_N(n-1)\\ w_{21}r_1(n-1)+w_{22}r_2(n-1)+\cdots+w_{2N}r_N(n-1)\\ \vdots\\ w_{N1}r_1(n-1)+w_{N2}r_2(n-1)+\cdots+w_{NN}r_N(n-1)\end{bmatrix}, \tag{2.24}$$

donde segue que sua k-ésima linha pode ser escrita como

$$\sum_{m=1}^{N}w_{km}r_m(n-1). \tag{2.25}$$

Considere agora o k-ésimo nó do *reservoir*. Combinando (2.21), (2.23) e (2.25), a equação de atualização do estado do k-ésimo nó, $k=1,\ \dots,N$, é

$$r_k(n)=(1-a)r_k(n-1)+a\tanh\left(w_{k1}^{\text{in}}+\sum_{m=1}^{N_u}w_{k(m+1)}^{\text{in}}u_m(n)+\sum_{m=1}^{N}w_{km}r_m(n-1)\right). \tag{2.26}$$

De (2.26), nota-se claramente o viés w_{k1}^{in} introduzido graças ao estado fixo em 1 de um dos nós da camada de entrada. Percebe-se também que todos os nós desta mandam seus estados $u_1(n),\ u_2(n),\ \ldots,\ u_{N_u}(n)$ ao k-ésimo nó do *reservoir*. Eles são multiplicados pelos pesos $w_{k2}^{\text{in}},\ w_{k3}^{\text{in}},\ \ldots,\ w_{kN}^{\text{in}}$ das conexões correspondentes.

Ainda, os N nós do *reservoir*, incluindo o próprio k-ésimo, transmitem seus estados precedentes ao k-ésimo nó do *reservoir*, que são multiplicados pelos pesos $w_{k1},\ w_{k2},\ \ldots,\ w_{kN}$ antes de chegarem ao destino. Esses envios de estado de um nó a outro se assemelham à condução de estímulos nervosos em dendritos [1].

Ao chegarem ao k-ésimo nó, os valores recebidos são somados e aplica-se a função $\tanh(\cdot)$ ao resultado obtido, numa maneira similar ao que ocorre no soma de um neurônio [1]. Finalmente, faz-se uma combinação linear, com coeficientes a e $(1-a)$, entre o resultado obtido mediante aplicação de $\tanh(\cdot)$ e o estado anterior $r_k(n-1)$ do nó. O resultado dessa combinação linear é o próximo estado do nó.

Na Fig. 2.11, na qual somente as camadas e ligações de interesse foram representadas, ilustra-se o que foi relatado nos parágrafos anteriores.

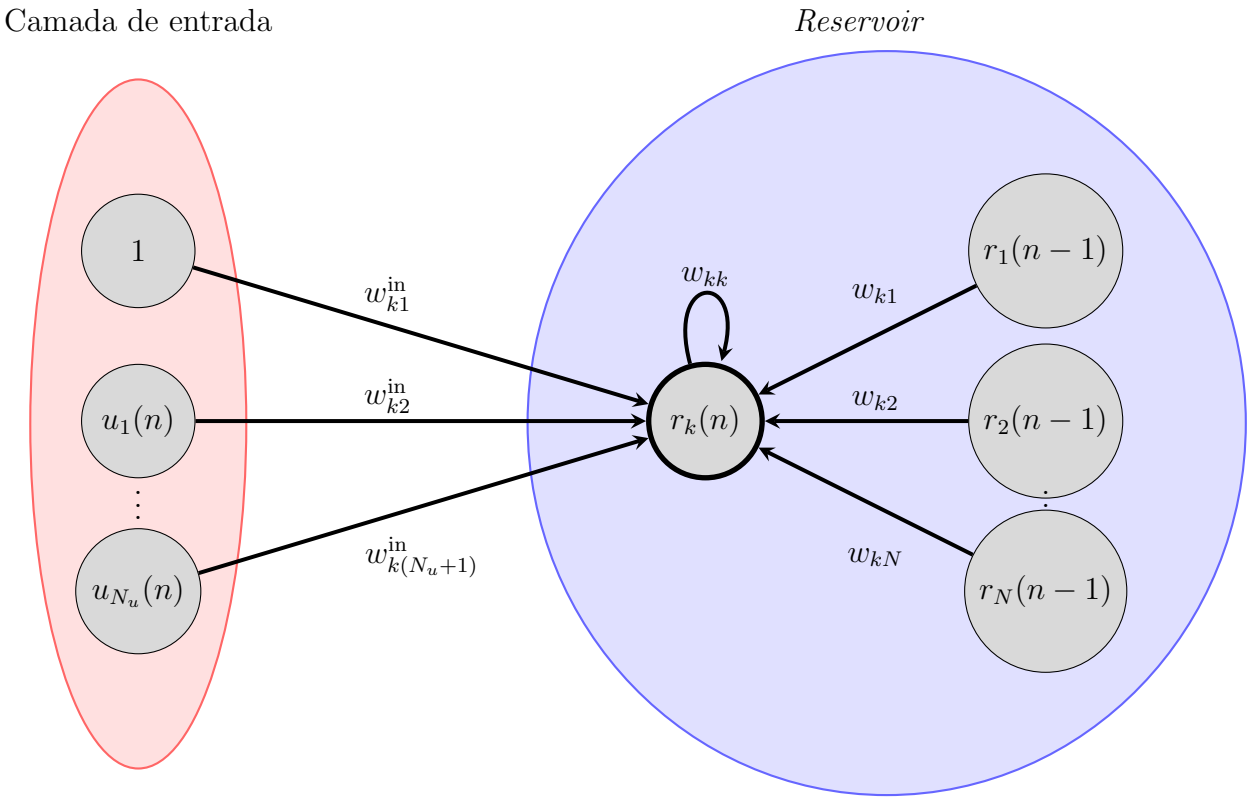

Figura 2.11: Nós de entrada e nós do *reservoir* transmitindo seus estados, no instante n, ao k-ésimo nó do *reservoir*.

2.2.3 Camada de saída

A camada de saída é formada por N_d nós e cada um deles recebe informação do *reservoir* através de N conexões. Os pesos w_{ij}^{out} dessas conexões são determinados após um período de treinamento e formam a matriz $\boldsymbol{W}_{N_d \times N}^{\text{out}}$. A explicação de como se obtém esta matriz mediante treinamento encontra-se na próxima subseção.

Os estados $y_1(n),\ y_2(n),\ \ldots,\ y_{N_d}(n)$ dos nós da camada de saída num dado instante n fazem parte do vetor $\boldsymbol{y}(n) \in \mathbb{R}^{N_d}$. Este é o sinal de saída da ESN e é uma função linear de $\boldsymbol{r}(n)$. De fato, $\begin{bmatrix} y_1(n) & y_2(n) & \ldots & y_{N_d}(n) \end{bmatrix}^T$ é simplesmente uma combinação linear de $\boldsymbol{r}(n)$, sendo os pesos dessa combinação os coeficientes w_{ij}^{out}. Isto se traduz pela relação

$$\boldsymbol{y}(n) = \boldsymbol{W}^{\text{out}} \boldsymbol{r}(n). \tag{2.27}$$

Para verificar melhor o que ocorre na rede, pode-se expandir (2.27) obtendo-se

$$\begin{aligned} \boldsymbol{y}(n) = \boldsymbol{W}^{\text{out}} \boldsymbol{r}(n) &= \begin{bmatrix} w_{11}^{\text{out}} & w_{12}^{\text{out}} & \ldots & w_{1N}^{\text{out}} \\ w_{21}^{\text{out}} & w_{22}^{\text{out}} & \ldots & w_{2N}^{\text{out}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_d1}^{\text{out}} & w_{N_d2}^{\text{out}} & \ldots & w_{N_dN}^{\text{out}} \end{bmatrix} \begin{bmatrix} r_1(n) \\ r_2(n) \\ \vdots \\ r_N(n) \end{bmatrix} \\ &= \begin{bmatrix} w_{11}^{\text{out}} r_1(n) + w_{12}^{\text{out}} r_2(n) + \cdots + w_{1N}^{\text{out}} r_N(n) \\ w_{21}^{\text{out}} r_1(n) + w_{22}^{\text{out}} r_2(n) + \cdots + w_{2N}^{\text{out}} r_N(n) \\ \vdots \\ w_{N1}^{\text{out}} r_1(n) + w_{N2}^{\text{out}} r_2(n) + \cdots + w_{NN}^{\text{out}} r_N(n) \end{bmatrix}. \end{aligned} \tag{2.28}$$

Desta maneira, o estado $y_k(n)$ do k-ésimo nó, $k = 1,\ 2,\ \ldots,\ N_d$, num dado instante n é calculado por meio de

$$y_k(n) = \sum_{m=1}^{N} w_{km}^{\text{out}} r_m(n). \tag{2.29}$$

Cada um dos N nós do *reservoir* envia seu estado a cada um dos N_d nós da camada de saída. Esses estados são multiplicados pelos pesos das conexões correspondentes e, ao chegarem nos nós destinatários, são somados para compor o novo estado deste. Na Fig. 2.12, apresenta-se uma ilustração do que acaba de ser descrito.

Com atenção a (2.21) e (2.27), cabe destacar que, sendo $\boldsymbol{r}(n)$ uma função da entrada atual e das passadas, pode-se dizer que o *reservoir* tem uma memória da entrada $\boldsymbol{u}(n)$. Isto provê contexto temporal às aplicações, que é a principal razão para se utilizar RNNs. Para aplicações que não requerem memória, técnicas não temporais de aprendizado de

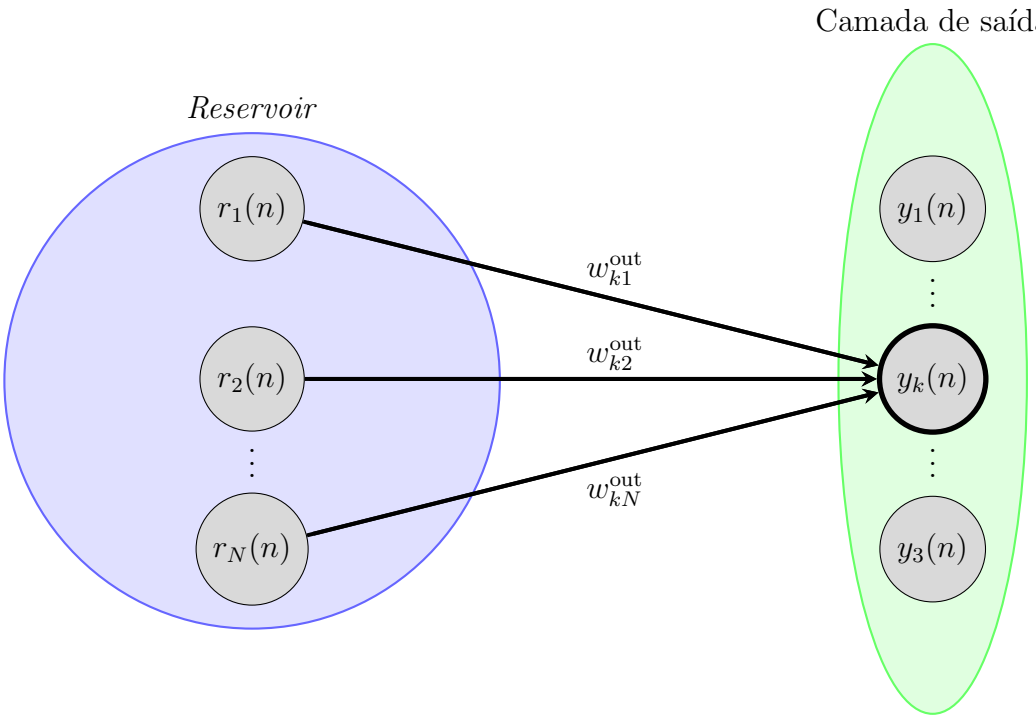

Figura 2.12: Nós do *reservoir* transmitindo seus estados, no instante n, ao k-ésimo nó da camada de saída.

máquina [67] podem ser mais convenientes [12].

É esperado que, graças a expansão não linear com memória (2.21), se obtenha um espaço de vetores de estados possíveis em $\mathbb{R}^N$ rico o suficiente para que a combinação linear (2.27) leve a $\boldsymbol{y}(n)$ suficientemente próximo de $\boldsymbol{d}(n)$. Para isso, é fundamental que se tenha um *reservoir* com um número de nós suficientemente grande - o que se traduz basicamente em $N \gg N_u$ [12].

Na próxima subseção, apresenta-se como obter $\boldsymbol{W}^{\text{out}}$ mediante treinamento da rede.

2.2.4 Treinamento

Para treinar a ESN, obtendo assim a matriz $\boldsymbol{W}^{\text{out}}$, considere que os parâmetros a, λ, p e q já foram definidos[7] e que existem L amostras do sinal desejado $\boldsymbol{d}(n)$ disponíveis para treinamento, compondo a matriz $\boldsymbol{D}_{N_d \times L}$ dada por

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{d}(1) & \boldsymbol{d}(2) & \dots & \boldsymbol{d}(L) \end{bmatrix}. \tag{2.30}$$

Antes de iniciar o treinamento, é necessário lidar com a transiência inicial da rede [12].

[7] Nos Capítulos 3, 4 e 5 explica-se como é feita a seleção destes parâmetros em cada uma das aplicações investigadas neste trabalho.

Nesta etapa, aplica-se (2.21) para $n = -\ell + 1$ até $n = 0$, obtendo-se $\boldsymbol{r}(-\ell+1),\ \boldsymbol{r}(-\ell+2),\ \ldots,\ \boldsymbol{r}(0)$. Para o cálculo de $\boldsymbol{r}(-\ell+1)$, considera-se que $\boldsymbol{r}(-\ell) = \boldsymbol{0}$[8].

Tendo-se obtido $\boldsymbol{r}(0)$, inicia-se o período de treinamento da rede. Agora, aplica-se (2.21) para $n = 1$ até $n = L$, obtendo-se $\boldsymbol{r}(1),\ \boldsymbol{r}(2)\ \ldots,\ \boldsymbol{r}(L)$. Estas amostras do vetor de estados $\boldsymbol{r}(n)$ são coletadas na matriz $\boldsymbol{T}_{N\times L}$, chamada por alguns autores de *matriz de trajetória* [19], dada por

$$\boldsymbol{T} = \Big[\boldsymbol{r}(1) \quad \boldsymbol{r}(2) \quad \ldots \quad \boldsymbol{r}(L)\Big]. \tag{2.31}$$

Neste ponto, determina-se a matriz $\boldsymbol{W}^{\text{out}}$ a partir de

$$\boldsymbol{W}^{\text{out}} = \boldsymbol{D}\boldsymbol{T}^{+}, \tag{2.32}$$

em que $\boldsymbol{T}^{+}$ é a matriz pseudo-inversa de Moore-Penrose [53] de $\boldsymbol{T}$.

De posse de $\boldsymbol{W}^{\text{out}}$, o treinamento da ESN está encerrado e inicia-se o período de testes da rede, no qual utiliza-se (2.21) com $n = L$ seguido de (2.27) com $n > L$ para se obter estimativas $\boldsymbol{y}(L+1) = \widehat{\boldsymbol{d}}(L+1),\ \boldsymbol{y}(L+2) = \widehat{\boldsymbol{d}}(L+2),\ \ldots$ do sinal desejado.

Finalmente, sejam dados a, λ, p, q e $\boldsymbol{D}$. Considerando essa e todas as subseções anteriores, os procedimentos necessários para gerar uma ESN e torná-la pronta para testes podem ser sintetizados nos passos a seguir:

1. Gerar $\boldsymbol{W}^{\text{in}}$ aleatoriamente à partir de distribuições uniformes. Os elementos da primeira coluna seguem uma distribuição uniforme em $[-p, p]$, enquanto que os demais seguem uma distribuição uniforme em $[-q, q]$;

2. Gerar $\boldsymbol{W}^{\text{aux}}$ aleatoriamente, com suas entradas obedecendo à uma distribuição uniforme em $[-1, 1]$ e calcular seu raio espectral λ_{aux};

3. Determinar $\boldsymbol{W}$ por meio de $\boldsymbol{W} = \lambda(\boldsymbol{W}_{\text{aux}}/\lambda_{\text{aux}})$;

4. Calcular (2.21) para $n = -\ell + 1$ até $n = L$ considerando-se $\boldsymbol{r}(-\ell) = \boldsymbol{0}$, de modo a obter $\boldsymbol{T}$ definido em (2.31); e

5. Determinar $\boldsymbol{W}^{\text{out}}$ utilizando-se (2.27).

Esta subseção encerra-se com o Algoritmo 1, que descreve os passos acima de forma algorítmica. Na próxima subseção, apresenta-se um problema de prediçao extremamente simples, a fim de elucidar como os conceitos aqui descritos são usados na prática.

[8] Na verdade qualquer valor de $\boldsymbol{r}(-\ell)$ pode ser utilizado [12]. No entanto a escolha $\boldsymbol{r}(-\ell) = \boldsymbol{0}$ é a mais simples e, portanto, é a utilizada neste trabalho.

Algoritmo 1 Geração e treinamento de uma ESN.

Entrada do algoritmo:

$\boldsymbol{u}(n)_{N_u \times 1}$: sinal de entrada, $n = -\ell + 1, -\ell + 2, \ldots, L$

$\boldsymbol{D}_{N_d \times L}$: matriz do sinal desejado

Saída do algoritmo:

$\boldsymbol{W}^{\text{out}}_{N_d \times N}$: matriz de saída

Parâmetros fixos:

$a \in [0, 1]$: parâmetro de *leakage*

$\lambda \in \mathbb{R}^+$: raio espectral

$p, q \in \mathbb{R}^+$: parâmetros de distribuições uniformes

$\ell \in \mathbb{N}^*$: comprimento de transiente

$L \in \mathbb{N}^*$: comprimento de treinamento

$N \in \mathbb{N}^*$: número de nós internos

Inicializações:

$w^{\text{in}}_{i1} \sim U[-p, p], \quad i = 1, 2, \ldots, N$

$w^{\text{in}}_{ij} \sim U[-q, q], \quad i = 1, 2, \ldots, N \text{ e } j = 2, 3, \ldots, N_{u+1}$

$w^{\text{aux}}_{ij} \sim U[-1, 1], \quad i, j = 1, 2, \ldots, N$

$r_i = 0, \quad i, = 1, 2, \ldots, N$

$\boldsymbol{W}^{\text{in}} = \{w^{\text{in}}_{ij}\}_{N \times N_u + 1}$: matriz de entrada

$\boldsymbol{W}^{\text{aux}} = \{w^{\text{aux}}_{ij}\}_{N \times N}$: matriz auxiliar

$\boldsymbol{r}(-\ell) = \{r_i\}_{N \times 1}$: vetor de estados

Algoritmo:

$\lambda_{\text{aux}} \leftarrow$ máximo dentre os valores absolutos dos autovalores de $\boldsymbol{W}^{aux}$

$\boldsymbol{W} = \lambda \left(\boldsymbol{W}^{\text{aux}} / \lambda_{\text{aux}} \right)$

para $n = -\ell + 1$ até L **faça**

$\quad \boldsymbol{r}(n) = (1 - a)\boldsymbol{r}(n-1) + a \tanh\left(\boldsymbol{W}^{\text{in}} \begin{bmatrix} 1 & \boldsymbol{u}(n) \end{bmatrix}^T + \boldsymbol{W}\boldsymbol{r}(n-1) \right)$

$\quad$ **se** $n > 0$ **então**

$\quad\quad$ **para** $k = 1$ até N **faça**

$\quad\quad\quad t_{kn} = r_k(n)$

$\quad\quad$ **fim para**

$\quad$ **fim se**

fim para

$\boldsymbol{T} = \{t_{ij}\}_{N \times L}$

$\boldsymbol{W}^{\text{out}} = \boldsymbol{D}\boldsymbol{T}^+$

2.2.5 Exemplo: predição de um sinal periódico

Considere um sinal periódico qualquer de período 3. Para efeito de exemplo, toma-se $u(n)$, tal que $u(-\ell+1) = 6$, $u(-\ell+2) = 28$, $u(-\ell+3) = 496$, $u(-\ell+4) = 6$, $u(-\ell+5) = 28$, $u(-\ell+6) = 496$, ... Esse sinal é formado pela repetição dos três primeiros números perfeitos [68].

Neste exemplo, é projetada uma ESN para dado $u(n)$, estimar $u(n+1)$. Para isso, são considerados os parâmetros da Tabela 2.2.

Tabela 2.2: Parâmetros utilizados no exemplo.

Parâmetro	**Valor**	**Unidade**
N	5	nó
a	0.70	-
λ	0.80	-
p	0.00	-
q	0.25	-
ℓ	10	amostra
L	5	amostra

O problema pode ser resolvido diretamente em algum software de simulação numérica. Neste caso, será utilizado o MATLAB® [37].

Uma vez que $\ell = 10$ e $d(n) = u(n+1)$, a matriz $\boldsymbol{D}$ definida em (2.30) é prontamente obtida como

$$\boldsymbol{D} = \begin{bmatrix} 496 & 6 & 28 & 496 & 6 \end{bmatrix}. \tag{2.33}$$

Para obtenção de $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}^{\text{aux}}$, os comandos apresentados no quadro a seguir foram executados.

```
%declaracao dos parametros
p = 0;
q = 0.25;
N = 5;
%geracao das matrizes
Win = 2*rand(N,N)-1;
Win(:,1) = p*Win(:,1);
Win(:,2) = q*Win(:,2);
Waux = 2*rand(N,N)-1;
```

As matrizes obtidas foram

$$\boldsymbol{W}^{\text{in}} = \begin{bmatrix} 0 & -0.0338 \\ 0 & 0.0974 \\ 0 & 0.1290 \\ 0 & -0.0337 \\ 0 & 0.0777 \end{bmatrix} \quad \text{e} \tag{2.34}$$

$$\boldsymbol{W}^{\text{aux}} = \begin{bmatrix} -0.7805 & -0.0248 & 0.3466 & -0.3684 & -0.8153 \\ 0.8675 & 0.5379 & -0.1409 & 0.5454 & -0.9844 \\ -0.6251 & -0.2080 & -0.0965 & 0.3929 & -0.1538 \\ -0.4676 & -0.4541 & 0.2197 & -0.7493 & 0.3111 \\ 0.5957 & -0.9255 & -0.8812 & -0.7397 & 0.4458 \end{bmatrix}. \tag{2.35}$$

Calculou-se o raio espectral de (2.35) por meio do comando:

```
%raio espectral de Waux
lambdaAux = max(abs(eigs(Waux)));
```

O resultado obtido foi $\lambda_{\text{aux}} = 0.7372$. Com isso, determina-se $\boldsymbol{W} = 0.7\,(\boldsymbol{W}^{\text{aux}}/0.7372)$, obtendo-se

$$\boldsymbol{W} = \begin{bmatrix} -0.8470 & -0.0269 & 0.3761 & -0.3997 & -0.8847 \\ 0.9414 & 0.5837 & -0.1529 & 0.5919 & -1.0682 \\ -0.6783 & -0.2257 & -0.1047 & 0.4263 & -0.1669 \\ -0.5075 & -0.4928 & 0.2384 & -0.8131 & 0.3376 \\ 0.6464 & -1.0043 & -0.9562 & -0.8027 & 0.4838 \end{bmatrix}. \tag{2.36}$$

Desta maneira, foram executados os Passos 1, 2 e 3 descritos no final de subseção anterior. Para iniciar o Passo 4, note que com o parâmetro a usado aqui, a equação de atualização do vetor de estados (2.21) torna-se

$$\boldsymbol{r}(n) = 0.3\boldsymbol{r}(n-1) + 0.7\tanh\left(\boldsymbol{W}^{\text{in}}\begin{bmatrix} 1 \\ \boldsymbol{u}(n) \end{bmatrix} + \boldsymbol{W}\boldsymbol{r}(n-1)\right). \tag{2.37}$$

Partindo-se de $n = -\ell + 1$, a aplicação de (2.37) por $\ell = 10$ vezes sucessivas com (2.34), (2.36) e considerando-se $\boldsymbol{r}(-10) = \boldsymbol{0}$, leva a

$$\boldsymbol{r}(0) = \begin{bmatrix} 0.0125 & -0.3295 & 0.6301 & 0.2787 & -0.1946 \end{bmatrix}^T. \tag{2.38}$$

Aplicando-se (2.37) por mais $L = 5$ vezes sucessivas, determina-se $\boldsymbol{T} = \left[\, \boldsymbol{r}(1)\ \boldsymbol{r}(2)\ \boldsymbol{r}(3)\ \boldsymbol{r}(4)\ \boldsymbol{r}(5) \,\right]$. O resultado obtido foi

$$\boldsymbol{T} = \begin{bmatrix} -0.3960 & -0.8188 & 0.0125 & -0.3960 & -0.8188 \\ 0.5962 & 0.8789 & -0.3296 & 0.5962 & 0.8789 \\ 0.8883 & 0.9665 & 0.6302 & 0.8883 & 0.9665 \\ -0.4275 & -0.8282 & 0.2787 & -0.4275 & -0.8282 \\ 0.5863 & 0.8759 & -0.1945 & 0.5863 & 0.8759 \end{bmatrix}. \tag{2.39}$$

Neste ponto, resta apenas o Passo 5 a ser executado. A equação (2.32) pode ser calculada com auxílio do comando **pinv** para deteminar a pseudo-inversa de uma matriz, conforme comando destacado no quadro a seguir.

```
%matriz de saida Wout
Wout = D*pinv(T);
```

O resultado obtido foi

$$\boldsymbol{W}^{\text{out}} = \begin{bmatrix} 2001.65146 & 1274.0725 & 612.4492 & 234.2465 & 145.3762 \end{bmatrix}. \tag{2.40}$$

Agora a rede pode ser testada, calculando-se (2.37) com (2.34) e (2.36), seguido de (2.27) a partir de $n = L + 1 = 6$. Notando-se que $u(6) = 6$, $u(7) = 28$, ..., tem-se

$$\boldsymbol{r}(6) = \begin{bmatrix} 0.0125 & -0.3296 & 0.6302 & 0.2787 & -0.1945 \end{bmatrix}^T. \tag{2.41}$$

Deseja-se que, nesse instante, a saída $y(6)$ esteja o mais próximo possível de $u(7) = 28$. O uso de (2.40) e (2.41) em (2.27) fornece

$$\begin{aligned} y(6) &= (2001.65146)(0.0125) + (1274.0725)(-0.3296) \\ &\quad + (612.4492)(0.6302) + (234.2465)(0.2787) + (145.3762)(-0.1945) \\ y(6) &= 28. \end{aligned}$$

Na Fig. 2.13, encontra-se a ilustração da rede e todos estados dos nós para $n = 6$. As conexões entrada-*reservoir* são representadas de modo simplificado.

Para $n = 7$, espera-se que a saída seja próxima de $u(8) = 496$. Nesse caso, o vetor de estados torna-se

$$\boldsymbol{r}(7) = \begin{bmatrix} -0.3960 & 0.5962 & 0.8883 & -0.4275 & 0.5863 \end{bmatrix}^T. \tag{2.42}$$

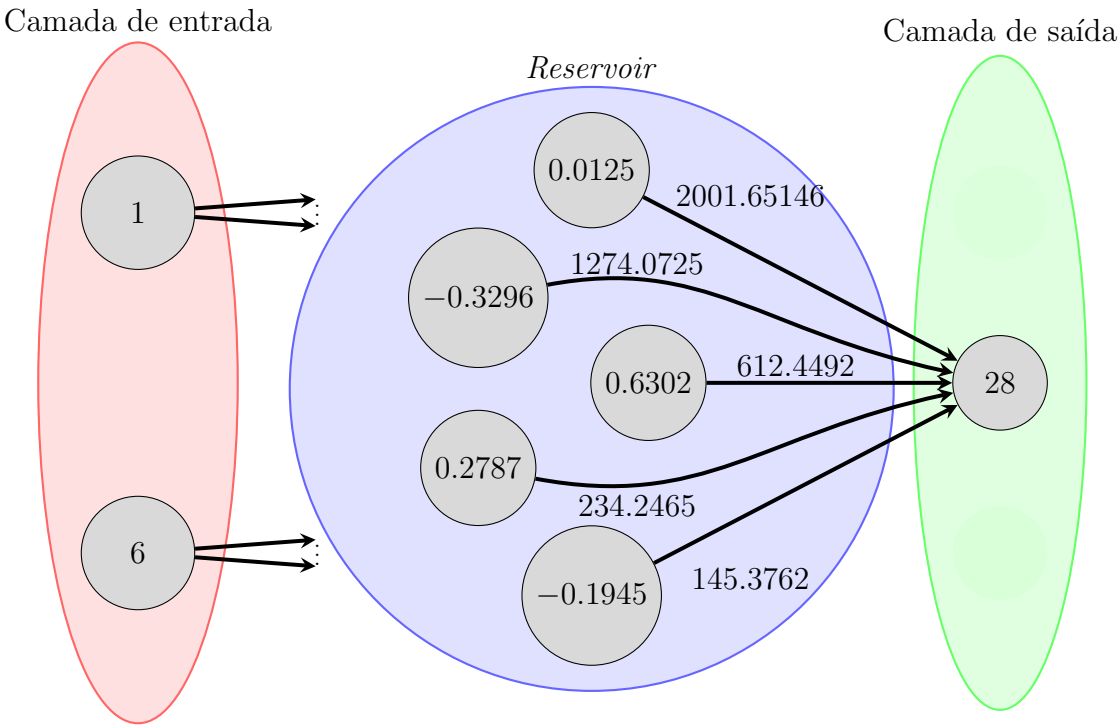

Figura 2.13: Estados dos nós da rede no instante $n = 6$.

O uso de (2.27) leva a

$$\begin{aligned} y(7) &= (2001.65146)(-0.3960) + (1274.0725)(0.5962) \\ &\quad + (612.4492)(0.8883) + (234.2465)(-0.4275) + (145.3762)(0.5863) \\ y(7) &= 496. \end{aligned}$$

O período de testes pode continuar até quando desejado. O leitor interessado é convidado a realizar as contas para obter a saída $y(n)$ para $n = 8$. O código em MATLAB® utilizado para resolver este exemplo encontra-se disponível no Anexo B.

A próxima subseção contém alguns comentários sobre ESNs que, ainda que pertinentes, não caberiam nesta subseção e nem nas anteriores.

2.2.6 Outros comentários

Tanto $\boldsymbol{W}^{\text{in}}$ quanto $\boldsymbol{W}$ são geradas aleatoriamente na fase de aprendizagem, conforme descrito nas páginas 23 e 24, e não se alteram mais. Na publicação original sobre ESNs [10], recomenda-se gerar $\boldsymbol{W}$ esparsa, isto é, com a maioria de suas entradas nulas, pois faz com que "haja um desacoplamento entre sub-redes internas, estimulando a dinâmica individual" [10]. Contudo, a experiência prática mostrou que a esparsidade de $\boldsymbol{W}$ não

possui grande influência no desempenho da ESN [12]. Esta constatação foi confirmada nos estudos de aplicações práticas conduzidos para a realizaçao deste trabalho. Deste modo, assim como feito em [19], neste trabalho não gera-se $\boldsymbol{W}$ esparsa.

Cabe também destacar que o intervalo escolhido para a distribuição uniforme usada para gerar $\boldsymbol{W}^{\text{aux}}$ é irrelevante, uma vez que este será normalizado para ter raio espectral unitário, por meio de $\boldsymbol{W}^{\text{aux}}/\lambda_{\text{aux}}$.

A ESN deve ser projetada para apresentar a propriedade de *echo state*, a qual faz com que a dependência do estado $\boldsymbol{r}(n)$ em relação aos estados passados e as entradas passadas se desvaneça gradualmente com as iterações [10]. Daí decorre que a escolha do valor inicial de $\boldsymbol{r}(n)$ é indiferente se a ESN for projetada para possuir a propriedade de *echo state*, o que na grande maioria das situações práticas é garantida se $\lambda < 1$, sendo λ o raio espectral [66] da matriz $\boldsymbol{W}$. A propriedade de *echo state* é fundamental para que a ESN funcione [12].

Os pesos das conexões *reservoir*-saída w_{ij}^{out} são ajustados de modo a minimizar o RMSE (2.18) entre $\boldsymbol{Y} = \boldsymbol{W}^{\text{out}}\boldsymbol{T}$ e $\boldsymbol{D}$ [10], o que leva a N_d sistemas de L equações lineares com N incógnitas cada. Em geral $L \gg N$, o que significa que os N_d sistemas são sobre-determinados [12]. Resolvendo-se esses sistemas, obtém-se os pesos desejados w_{ij}^{out} (2.32). Ambos L e N apresentam grande influência no consumo de recursos computacionais.

Em geral, para problemas lineares, usa-se p e q relativamente pequenos, de modo que $\tanh(\cdot)$ em (2.21) tenha argumentos mais próximos de 0, operando em sua região aproximadamente linear. Para problemas não lineares, utiliza-se p e q relativamente maiores, levando $\tanh(\cdot)$ em (2.21) para sua região não linear [12].

Também vale ressaltar que é adequado minimizar o número de amostras ℓ descartadas para lidar com transientes iniciais da rede, principalmente em aplicações onde o número de dados disponíveis é limitado. Tipicamente, valores de ℓ na casa das dezenas ou centenas já são suficientes [12].

E assim encerra-se a seção que detalha o funcionamento das ESNs utilizadas neste trabalho. Na próxima seção, é deduzida a equação para obtenção dos coeficientes do filtro de Wiener, utilizado como *benchmark* nas aplicações estudadas nos Capítulos 3 e 4.

2.3 Filtro de Wiener

Seja um filtro linear com M coeficientes $w_0, w_1, \ldots w_{M-1}$, cuja entrada no instante n consiste na amostra $u(n)$ da série temporal $u(0), u(1), \ldots, u(n)$ de modo que a saída é determinada por [26]

$$y(n) = \sum_{k=0}^{M-1} w_k u(n-k). \tag{2.43}$$

Considere ainda o sinal de erro

$$e(n) = d(n) - y(n), \tag{2.44}$$

em que $d(n)$ é o sinal desejado. O filtro de Wiener é tal que o erro quadrático médio, obtido elevando-se (2.44) ao quadrado e calculando-se a média, é minimizado [26].

Definindo-se a função custo

$$J = E\left[e^2(n)\right] = E\left[(d(n) - y(n))^2\right], \tag{2.45}$$

e substituindo-se (2.43) em (2.45), obtém-se

$$J = E\left[\left(d(n) - \sum_{k=0}^{M-1} w_k u(n-k)\right)^2\right]. \tag{2.46}$$

Cada coeficiente w_m, $m = 0, 1, \ldots, M-1$, do filtro de Wiener deve ser tal que [26]

$$\frac{\partial J}{\partial w_m} = 0, \tag{2.47}$$

donde

$$\begin{aligned}
&E\left[\frac{\partial}{\partial w_m}\left(d(n) - \sum_{k=0}^{M-1} w_k u(n-k)\right)^2\right] = \\
&E\left[-2w_m u(n-m)\left(d(n) - \sum_{k=0}^{M-1} w_k u(n-k)\right)\right] = \\
&E\left[-2w_m u(n-m)d(n)\right] + E\left[2w_m u(n-m)\sum_{k=0}^{M-1} w_k u(n-k)\right] = \\
&-2w_m E\left[u(n-m)d(n)\right] + 2w_m E\left[\sum_{k=0}^{M-1} w_k u(n-m)u(n-k)\right] = 0 \\
&\Rightarrow \sum_{k=0}^{M-1} w_k E\left[u(n-m)u(n-k)\right] = E\left[u(n-m)d(n)\right].
\end{aligned} \tag{2.48}$$

Usando-se as definições de função autocorrelação $r_{uu}(n)$ e de correlação cruzada $r_{ud}(n)$ [69], segue que

$$\sum_{k=0}^{M-1} r_{uu}(m-k)w_k = r_{ud}(m), \tag{2.49}$$

ou

$$\begin{bmatrix} r_{uu}(m) \\ r_{uu}(m-1) \\ \vdots \\ r_{uu}(m-M+1) \end{bmatrix}^T \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix} = r_{ud}(m), \quad m = 0, 1, \ldots, M-1. \tag{2.50}$$

Deste modo,

$$\begin{bmatrix} r_{uu}(0) & r_{uu}(-1) & \ldots & r_{uu}(-M+1) \\ r_{uu}(1) & r_{uu}(0) & \ldots & r_{uu}(-M+2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{uu}(M-1) & r_{uu}(M-2) & \ldots & r_{uu}(0) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix} = \begin{bmatrix} r_{ud}(0) \\ r_{ud}(1) \\ \vdots \\ r_{ud}(M-1) \end{bmatrix}, \tag{2.51}$$

donde

$$\boldsymbol{R}\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{p} \Rightarrow \boxed{\boldsymbol{w}_{\mathrm{o}} = \boldsymbol{R}^{-1}\boldsymbol{p}} \tag{2.52}$$

em que $\boldsymbol{w}_{\mathrm{o}}$ é o vetor de coeficientes de ótimos de Wiener, $\boldsymbol{R}$ é a matriz de autocorrelação e $\boldsymbol{p}$ é o vetor de correlação cruzada [69].

De posse de (2.52), é possível realizar a filtragem linear ótima de um sinal em que, a partir de uma entrada $u(n)$, obtém-se uma estimativa de $d(n)$. Tal resultado faz com que o filtro de Wiener e suas extensões encontrem aplicações em processamento de áudio [70], voz [71], imagem [72], dentre outras áreas.

O processo de obtenção dos coeficientes do filtro de Wiener é sumarizado no Algoritmo 2.

2.4 Resumo do capítulo

Neste capítulo foram apresentados os principais conceitos empregados ao decorrer dos próximos capítulos.

Iniciou-se com uma concisa apresentação de sinais caóticos, destacando suas principais propriedades e aplicações. Em particular, detalhou-se o gerador de sinais caóticos utilizado, o mapa tenda inclinada, e suas características espectrais. Algumas páginas foram dedicadas a explicação de um possível uso de sinais caóticos em comunicações.

Algoritmo 2 Obtenção dos coeficientes de um filtro de Wiener.

Entrada do algoritmo:

$u(0), u(1), \ldots, u(L-1)$: amostras do sinal de entrada do filtro
$d(0), d(1), \ldots, d(L-1)$: amostras do sinal desejado

Saída do algoritmo:

$\boldsymbol{w}_{\text{o}}$: vetor de coeficientes ótimos de Wiener $M \times 1$

Parâmetros fixos:

L : número de amostras do sinal desejado disponíveis
M : número de coeficientes do filtro

Algoritmo:

para $k = 1$ até M **faça**
  $r_k^{uu} = \frac{1}{L-k+1} \sum_{n=k-1}^{L-1} u(n)u(n-k+1)$
  $p_k = \frac{1}{L-k+1} \sum_{n=k-1}^{L-1} d(n)u(n-k+1)$
fim para

para $i = 1$ até M **faça**
  para $j = 1$ até M **faça**
    $r_{ij} = r_{|i-j|+1}^{uu}$
  fim para
fim para

$\boldsymbol{p} = \{p_k\}_{M \times 1}$
$\boldsymbol{R} = \{r_{ij}\}_{M \times M}$
$\boldsymbol{w}_{\text{o}} = \boldsymbol{R}^{-1}\boldsymbol{p}$

Após isso, procurou-se definir claramente o que é considerado como sinal musical aqui, explicando sucintamente as informações de arquivos MIDI que o compõem. Conceitos musicais, como a altura de uma nota, foram pontuados de modo que mesmo aqueles não familiarizados com o assunto possam compreender o estudo de predição musical relatado no Capítulo 5.

A maior parte deste capítulo foi reservada ao detalhamento dos pormenores das *echo state networks*, alicerce do *reservoir computing*, utilizadas ao longo de todas as investigações detalhadas nos capítulos conseguintes. As etapas a serem cumpridas para a construção de uma ESN estão detalhadas em uma lista ao final da Subseção 2.2.4 e sintetizadas no Algoritmo 1.

Na parte final deste capítulo, é deduzida a equação para obtenção dos coeficientes do filtro de Wiener, o método de filtragem linear ótimo utilizado como *benchmark*. A obtenção destes coeficientes pode ser conseguida por meio da execução do Algoritmo 2.

Cada um dos próximos três capítulos é dedicado a uma aplicação de *reservoir com-*

puting em processamento de sinais: a mitigação de ruído em sinais caóticos no Capítulo 3, equalização de sistemas de comunicação baseados em caos no Capítulo 4 e a predição de sinais musicais no Capítulo 5.

Capítulo 3

Redução de ruído em sinais caóticos

A mitigação de ruído sobre um sinal de interesse é um problema encontrado num vasto número de áreas de pesquisa em processamento de sinais [73]. Por exemplo, sistemas de comunicação [74], de controle para fins médicos [75] e de processamento de áudio [33] são todos suscetíveis a ter um sinal desejado corrompido por ruído.

Com o aumento da quantidade de pesquisas sobre novos sistemas de comunicação empregando sinais caóticos como portadoras de banda larga [76, 77], faz sentido buscar por métodos para mitigação de ruído em tais sinais. De fato, diferentes técnicas para tal fim foram propostas [73]. Uma vez que técnicas de aprendizado de máquina tem se tornado cada vez mais atraentes devido as suas grandes capacidades de adaptabilidade e princípios de projeto simples [61], é natural investigar o desempenho delas para mitigação de ruído.

Sendo assim, neste capítulo estuda-se o desempenho de uma ESN para mitigação de AWGN sobre um sinal caótico gerado pelo mapa tenda inclinada (2.5). Um filtro de Wiener (FW) de 10 coeficientes é empregado para solucionar o mesmo problema e servir de referência para comparação, uma vez que trata-se da técnica ótima de filtragem linear.

Em [78], é investigado o emprego de ESNs para mitigação de ruído. Lá, o sinal de entrada da ESN consiste numa combinação linear de quatro senóides com fases arbitrárias e envelopes variantes no tempo, corrompido por AWGN. Em [79], ESNs foram utilizadas para mitigação de ruído em sinais de eletroencefalograma. O primeiro trabalho avaliou o desempenho em termos do erro quadrático médio normalizado, enquanto que o segundo empregou a SNR e o erro quadrático médio.

Neste trabalho, diferentemente dos anteriores, empregam-se sinais caóticos de tempo discreto corrompidos por AWGN. Isso torna a tarefa de redução de ruído mais árdua devido ao comportamento aperiódico e à dependência sensível em relação as condições iniciais de sinais caóticos. Ademais, aqui o desempenho é avaliado por meio do ganho de processamento, por ser uma medida objetiva que demonstra mais claramente as capaci-

dades da rede.

Uma vez que as órbitas do mapa tenda inclinada possuem suas propriedades espectrais dependentes do parâmetro α do mapa, analisa-se também como este parâmetro afeta os resultados obtidos.

O restante deste capítulo está organizado em três seções. Na Seção 3.1 formula-se matematicamente o problema estudado e traz-se exemplos dos sinais envolvidos. Na Seção 3.2 apresenta-se a estrutura da ESN empregada, os detalhes do processo de seleção dos parâmetros da rede e os resultados numéricos obtidos. Por fim, na Seção 3.3 encontram-se as conclusões e a discussão sobre os resultados alcançados.

3.1 O problema de redução de ruído

O diagrama em blocos do problema é apresentado na Fig. 3.1.

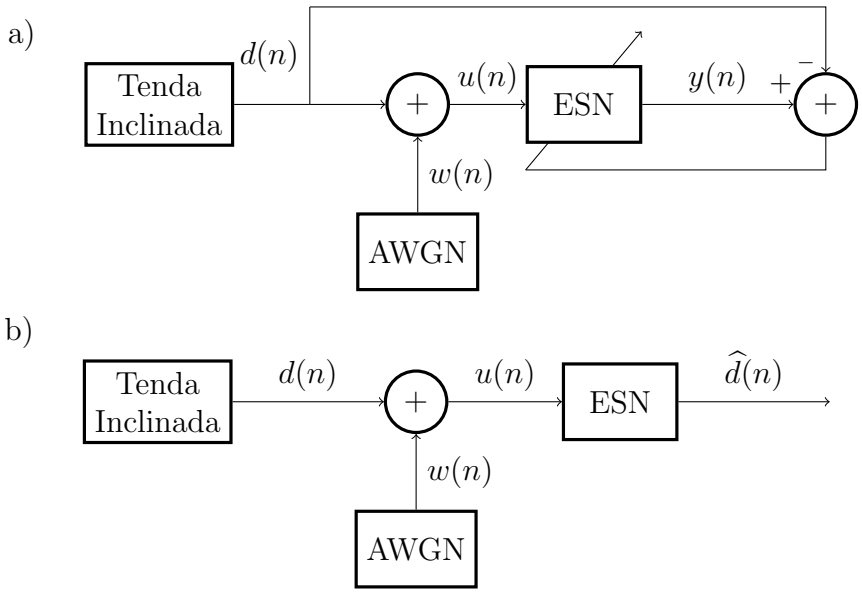

Figura 3.1: Diagrama de blocos do problema de redução de ruído de sinais caóticos: a) período de treinamento da ESN; b) período de teste da ESN.

Para gerar o sinal caótico de interesse, é usado o mapa tenda inclinada definido por (2.5). Ao sinal gerado, é adicionado AWGN $w(n)$ de modo que o sinal obtido, que é a entrada da ESN, é dado por

$$u(n) = d(n) + w(n), \tag{3.1}$$

sendo $d(n)$ um sinal caótico gerado pelo mapa tenda inclinada.

Almeja-se treinar a ESN para mitigar o ruído sobre $u(n)$. Isto é, após o treinamento

adequado, a ESN deve fornecer como sinal de saída $y(n)$ uma aproximação para o sinal caótico livre de ruído $d(n)$, tendo o sinal $u(n)$ como entrada. Na Fig. 3.1 a), encontra-se o período de treinamento da rede, em que são utilizadas amostras do sinal caótico livre de ruído $d(n)$. Durante o período de testes mostrado na parte b) da mesma figura, a saída da ESN passa a ser uma estimativa do sinal caótico $d(n)$, daí a notação $\widehat{d}(n)$ para o sinal de saída.

Na Fig. 3.2 mostram-se exemplos de $d(n)$, $w(n)$ e $u(n)$ para $d_0 = 0$, $\alpha = 0.7$ e $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB. Percebe-se que o sinal desejado possui diversas transições abruptas, caracterizados pelos picos presentes na linha preta apresentada nas Fig. 3.2 a) e c).

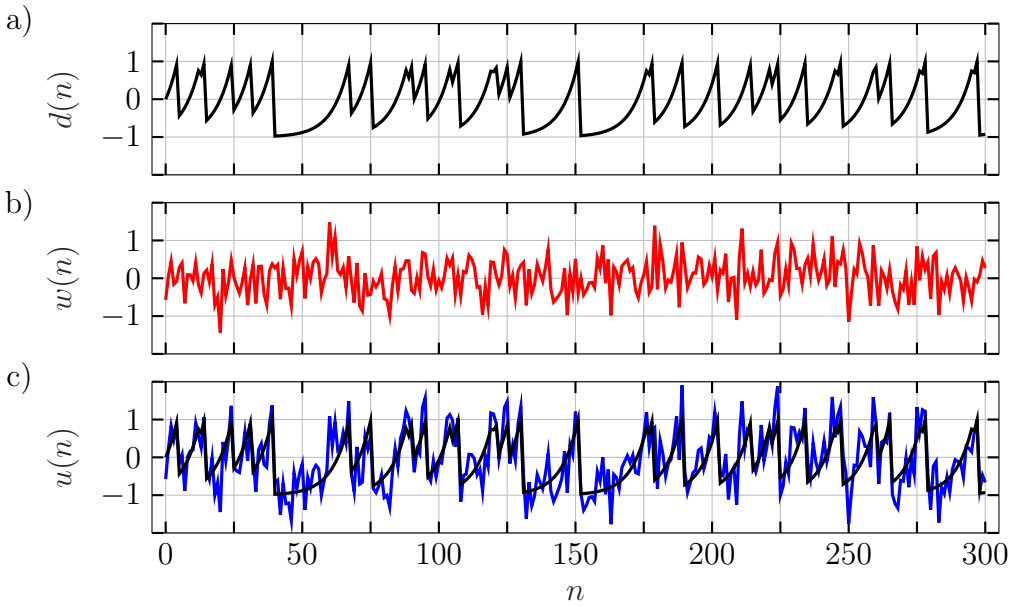

Figura 3.2: Exemplo de sinal caótico corrompido por ruído: a) sinal caótico livre de ruído; b) função-amostra de sinal AWGN para $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB; c) sinal de entrada da ESN (linha azul) e sinal caótico livre de ruído (linha preta).

3.2 Resultados de simulações

A ESN foi construída com $N = 500$ nós, período de transiente $\ell = 200$ amostras e $L = 25000$ amostras de $d(n)$ disponíveis para treinamento. Durante o período de testes, foram usados 10^6 pontos para o cálculo do ganho de processamento

$$G[\mathrm{dB}] \triangleq \mathrm{SNR}_{\mathrm{out}}[\mathrm{dB}] - \mathrm{SNR}_{\mathrm{in}}[\mathrm{dB}], \tag{3.2}$$

em que $\mathrm{SNR}_{\mathrm{in}}$ e $\mathrm{SNR}_{\mathrm{out}}$ indicam, nesta ordem, a SNR na entrada e na saída da ESN.

Para escolher os parâmetros da ESN a, λ, p e q, gerou-se $u(n)$ com $d_0 \in (-1, 1)$

arbitrário, $\alpha = 0.1$ e $\text{SNR}_{\text{in}} = 2.0$ dB. Além de $u(n)$, ambos $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}$ foram mantidos fixos até o término da rotina de seleção de parâmetros, descrita nos próximos parágrafos.

Como primeiro passo, foram tomados $\lambda = 0.05$, $p = 0$ e $q = 0.5$. Então, mantendo-os fixos, variou-se a de 0 até 1 em passos de 0.05. Em todos os casos, a ESN foi treinada e testada, calculando-se o ganho de processamento. Assim, determina-se o valor $a = a_1^{\text{opt}}$ que maximiza (3.2). Analogamente, obtém-se λ_1^{opt}, variando-se λ de 0.05 até 1 em passos de 0.05 e mantendo-se os demais parâmetros fixos em $a = a_1^{\text{opt}}$, $p = 0$ e $q = 0.5$. Em seguida, p_1^{opt} e q_1^{opt} foram obtidos, nesta ordem, variando-se p entre 0 e 10 em passos de 0.5 e q entre 0.5 e 10 em passos de 0.5, enquanto os demais parâmetros permanecem fixos nos valores selecionados mais recentes.

Essa rotina é repetida, obtendo-se a_i^{opt}, λ_i^{opt}, p_i^{opt}, q_i^{opt}, $i = 2, 3, \ldots$. A busca continua até que os valores selecionados parem de mudar. A evolução da seleção de parâmetros com o passar da rotina encontra-se na Fig. 3.3. Foram necessárias 12 iterações para que todos os parâmetros parassem de mudar. Os valores finais selecionados, que são utilizados nas próximas simulações desta seção, são $a_\star = 0.80$, $\lambda_\star = 0.75$, $p_\star = 1.50$ e $q_\star = 1.00$.

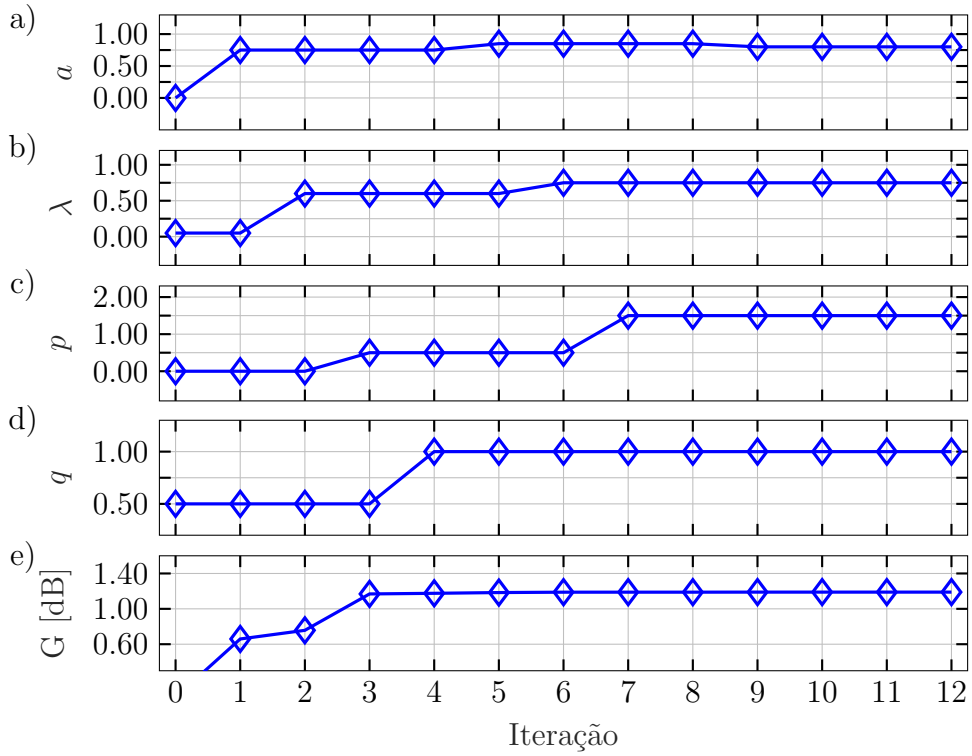

Figura 3.3: Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral de $\boldsymbol{W}$; c) e d) parâmetros das distribuições uniformes usadas para obter $\boldsymbol{W}^{\text{in}}$; e) ganho de processamento.

Na literatura, estes parâmetros são normalmente determinados traçando-se curvas de

RMSE em função de cada um deles, enquanto os demais permanecem fixos em valores predeterminados. Ao final, selecionam-se aqueles que minimizam as RMSEs correspondentes. O procedimento de seleção alternativo empregado neste trabalho permite uma seleção mais rápida dos parâmetros que, baseado nos testes computacionais realizados, fornecem resultados tão satisfatórios quanto os obtidos por meio da abordagem usual.

Na Tabela 3.1 encontram-se todos os parâmetros da rede e do canal utilizados para as simulações deste capítulo.

Tabela 3.1: Parâmetros utilizados no problema de redução de ruído.

Parâmetro	**Valor**	**Unidade**
N	500	nó
a	0.80	-
λ	0.75	-
p	1.50	-
q	1.00	-
ℓ	200	amostra
L	25000	amostra
SNR_{in}	2.00	dB

Na Fig. 3.4 encontram-se exemplos dos sinais estimados usando a ESN e o FW para $\text{SNR}_{\text{in}} = 2.0$ dB e parâmetro do mapa tenda inclinada $\alpha = 0.9$. O sinal estimado pelo FW é indicado por $\widehat{d}_W(n)$. Neste cenário, $\text{SNR}_{\text{out}} = 7.7$ dB para a ESN e $\text{SNR}_{\text{out}} = 5.1$ dB para o FW, mostrando que a ESN possui um melhor desempenho que o FW neste caso.

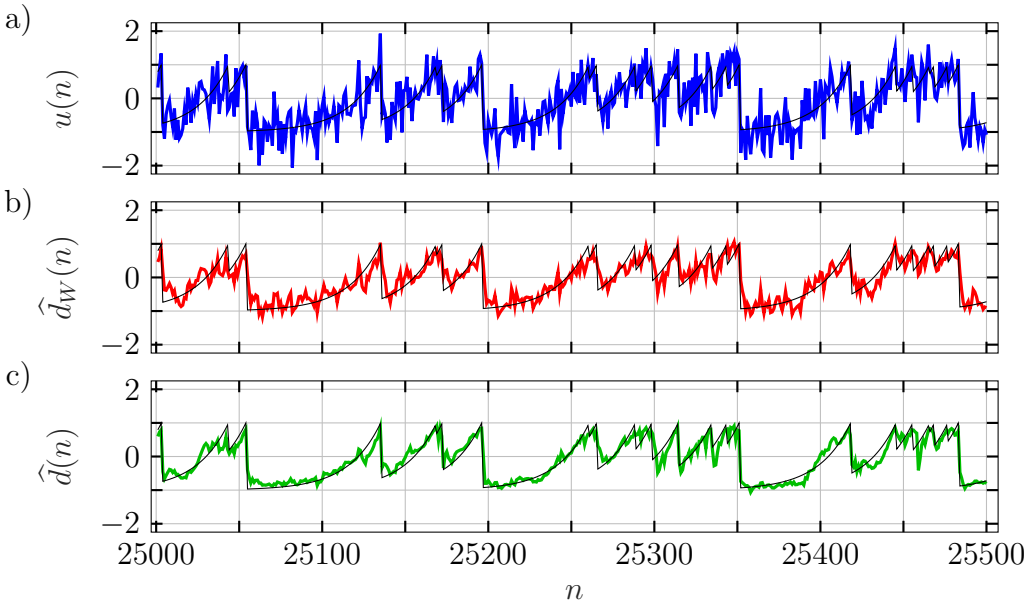

Figura 3.4: Mitigação de ruído para $\alpha = 0.9$ e $\text{SNR}_{\text{in}} = 2.0$ dB: a) sinal de entrada $u(n)$; b) sinal estimado pelo FW $\widehat{d}_W(n)$; c) sinal estimado pela ESN $\widehat{d}(n)$. As curvas na cor preta representam o sinal desejado $d(n)$.

Já na Fig. 3.5 encontram-se exemplos dos sinais estimados para $\text{SNR}_{\text{in}} = 2.0$ dB e $\alpha = 0.1$. Agora, $\text{SNR}_{\text{out}} = 3.1$ dB para a ESN e $\text{SNR}_{\text{out}} = 2.0$ dB para o FW. Mesmo que visualmente seja difícil perceber a melhora, o resultado numérico indica que a ESN consegue novamente um melhor desempenho do que o FW.

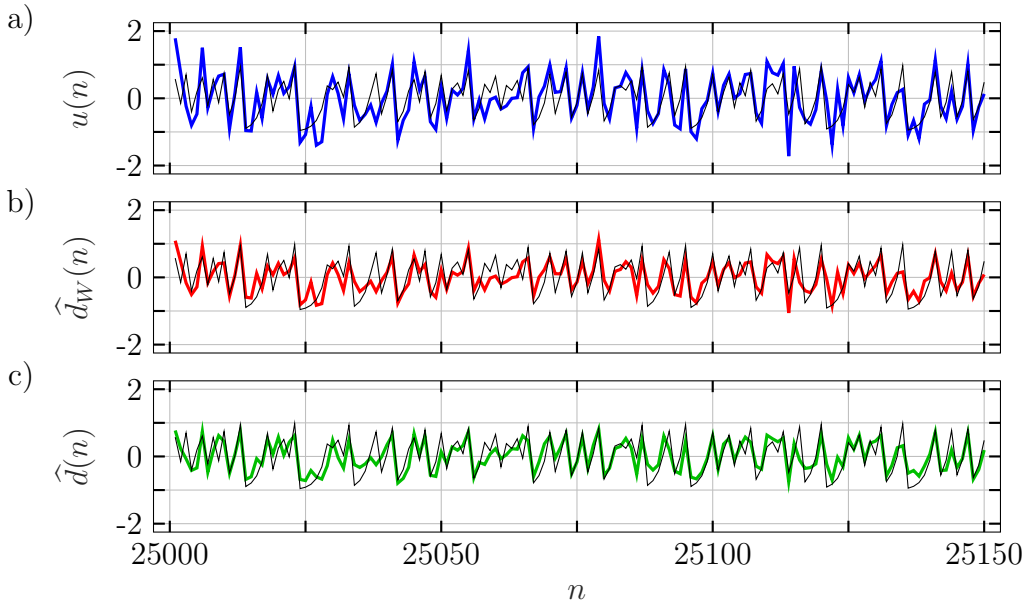

Figura 3.5: Mitigação de ruído para $\alpha = 0.1$ e $\text{SNR}_{\text{in}} = 2.0$ dB: a) sinal de entrada $u(n)$; b) sinal estimado pelo FW $\widehat{d}_W(n)$; c) sinal estimado pela ESN $\widehat{d}(n)$. As curvas na cor preta representam o sinal desejado $d(n)$.

Como pode ser deduzido de (2.8) e da Fig. 2.4, a DEP do sinal caótico torna-se cada vez mais branca à medida que $|\alpha|$ aproxima-se de zero. Assim, em termos de autocorrelação, ele torna-se mais similar ao ruído e a mitigação de ruído mais difícil, de modo que é esperado que o ganho de processamento (3.2) decaia com a diminuição de $|\alpha|$.

Fazendo uso dos parâmetros selecionados, a ESN foi treinada e testada para diferentes valores de $\alpha \in (-1, 1)$. Na Fig. 3.6 encontram-se os resultados obtidos, considerando o ganho de processamento médio dentre cinco simulações de treino e teste. Como referência para comparação, um FW de 10 coeficientes foi empregado para cumprir a mesma tarefa de mitigação de ruído. O número de coeficientes foi escolhido com base em testes computacionais, observando-se que após uma certa quantidade de coeficientes, o desempenho atingido pelo FW não sofre melhoras substanciais. As barras de erro indicam o desvio padrão calculado considerando-se as cinco repetições.

A comparação mediante a observação da curva de ganho em função do parâmetro α do mapa tenda inclinada, além de permitir a avaliação da ESN, também indica a influência de α sobre o desempenho de ambos os métodos.

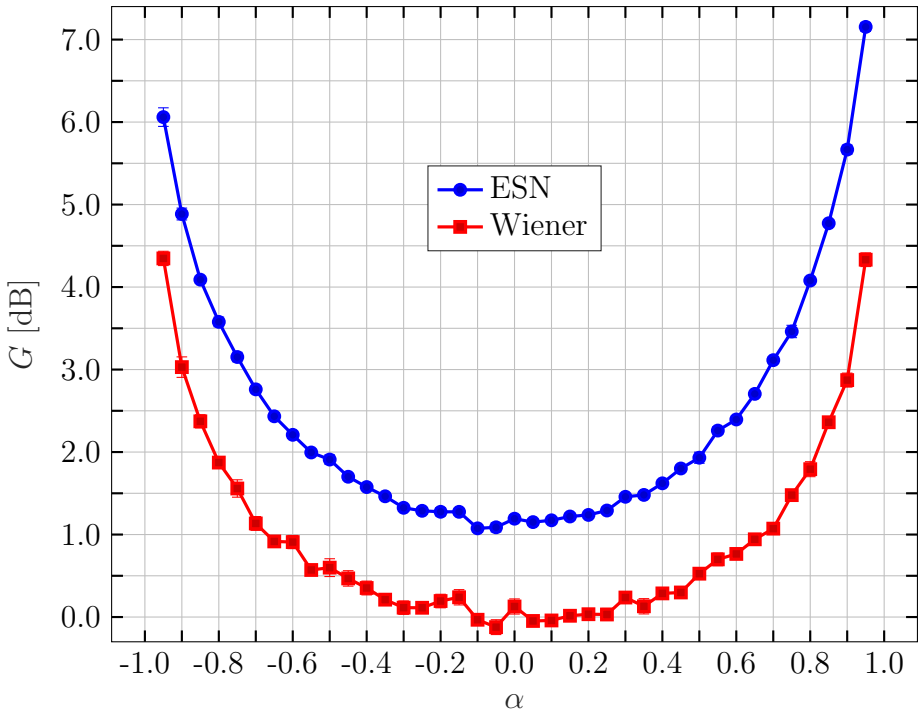

Figura 3.6: Ganho de processamento em dB em função do parâmetro do mapa tenda inclinada α usando uma ESN e um FW para $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB.

Os resultados apresentados mostram que o desempenho da ESN é melhor que o do FW para todos valores de α. Ademais, como esperado, o desempenho de ambos métodos deterioram-se conforme α aproxima-se de zero. Cabe destacar que para $\alpha = 0$, a DEP do sinal caótico é branca e o ganho de processamento do FW é nulo. O ganho de processamento é máximo para $\alpha = 0.95$, onde ele atinge 7.17 ± 0.05 dB.

Em trabalhos como [80–85], ganhos de processamento indo de 6.0 dB até 25.0 dB foram obtidos utilizando outras técnicas, mas sempre considerando sistemas caóticos de tempo contínuo. O autor considera que a mitigação de ruído em sinais caóticos de tempo discreto é consideravelmente mais difícil que aquela em sinais caóticos de tempo contínuo, uma vez que o comportamento suave destes facilita a tarefa.

3.3 Conclusões

Neste capítulo, estudou-se o emprego de uma ESN para mitigação de ruído num sinal caótico. Diferentemente de outros trabalhos na literatura, consideraram-se sinais caóticos gerados por sistemas dinâmicos de tempo discreto, cuja presença de transições abruptas tornam a tarefa mais desafiadora.

Como gerador de sinais caóticos, considerou-se o mapa tenda inclinada, uma vez que

este possui DEP conhecida e dependente do único parâmetro do mapa, permitindo uma análise da influência de suas características espectrais sobre o desempenho do *denoising*.

Foi mostrado que, selecionando-se e ajustando-se os parâmetros da rede, o desempenho da ESN supera o do FW - técnica de filtragem linear ótima - em todos os casos, mesmo quando a DEP do sinal caótico é branca, caso em que o ganho de processamento do FW é nulo. Com o crescimento de $|\alpha|$, a DEP dos sinais caóticos torna-se banda estreita e o ganho de processamento obtido é maior.

Capítulo 4

Sistema de comunicação baseado em caos utilizando ESN

Existe um número crescente de pesquisas sobre o uso de sinais caóticos como portadoras de banda larga em sistemas de comunicação [86–93], conforme afirmado no capítulo anterior.

Além de serem banda larga, sinais caóticos possuem outras características que tornam atrativa a utilização deles em telecomunicações. Dentre elas, destacam-se a robustez contra os efeitos introduzidos por canais multipercurso [89] e a melhoria na confidencialidade do tráfego de informação [86]. A primeira decorre do fato da baixa correlação cruzada entre sinais caóticos com diferentes condições iniciais [89]. Já a segunda, é consequência da aperiodicidade, DSCI e comportamento similar a ruído sob certas condições [86].

Deste modo, a fim de dar continuidade ao estudo relatado no Capítulo 3, no presente capítulo investiga-se o desempenho do sistema de comunicação baseado em caos (SCBC) proposto em [27] e discutido na página 12, para transmitir mensagens binárias utilizando uma ESN para estimar a mensagem enviada dado o sinal recebido. Para isto, faz-se uso da codificação soma (2.13) e do mapa tenda inclinada (2.5) em três cenários distintos.

No Cenário I, o canal adiciona apenas AWGN ao sinal transmitido; no Cenário II, não há ruído mas um canal do tipo $H(z) = h_0 + z^{-1} + h_0 z^{-2}$ é considerado; por fim, no Cenário III, é examinado o SCBC considerando o mesmo canal anterior e AWGN.

Nestes três cenários, a ESN é treinada de modo a mitigar o ruído, equalizar o canal, realizar a decodificação caótica e tomar a decisão sobre qual bit foi transmitido. Em outras palavras, a ESN faz o papel de todos os componentes do receptor. Em todos os casos, o desempenho é quantificado por meio da BER.

Para comparação do desempenho da ESN, em todos os cenários um FW com 10 coeficientes é utilizado para equalizar o canal. Após a equalização, é aplicado o processo de sincronismo caótico para tomada de decisão conforme descrito na página 14.

Em trabalhos anteriores, ESNs são usadas em conjunto com filtros de Kalman para

realizar equalização não supervisionada de canais não lineares usando-se sinais caóticos [94] ou sinais com modulação de amplitude em quadratura [95]. O desempenho das técnicas empregadas em ambos trabalhos é avaliado por meio do erro quadrático médio. Apesar daqui se fazer uso de um canal linear, o desempenho é avaliado por meio da BER, que é uma medida mais objetiva e significativa, capaz de demonstrar claramente o quão bem o esquema proposto aqui funciona na prática.

Já em [96], acopla-se uma ESN a um receptor de um SCBC já existente para obter limiares mais precisos para decisão do símbolo transmitido, melhorando o desempenho do sistema de comunicação empregado. Em [97], utiliza-se uma ESN para predição de canal em sistemas de comunicação de múltiplas entradas e múltiplas saídas usando multiplexação de frequência ortogonal. Diferentemente do que ocorre nestes trabalhos anteriores, a ESN aqui é empregada tanto para equalizar o canal quanto para tomar as decisões de quais bits foram transmitidos, substituindo completamente o receptor do sistema de comunicação proposto em [27].

O restante deste capítulo está organizado da seguinte forma: na Seção 4.1, é tratado o caso em que se considera apenas AWGN. Na Seção 4.2, aborda-se o cenário em que considera-se somente o canal linear $H(z) = h_0 + z^{-1} + h_0 z^{-2}$. Por fim, na Seção 4.3, relata-se o estudo do problema quando ambos AWGN e $H(z)$ são considerados.

4.1 Cenário I: somente ruído

O diagrama em blocos deste primeiro cenário é apresentado na Fig. 4.1. Na parte a), encontra-se esquematizado o período de treinamento da ESN. Nele, são utilizados 25200 amostras de $m(n)$, sendo 200 para descartar transientes iniciais da rede e 25000 efetivamente usadas para o treinamento da ESN. Na parte b), é apresentado o período de teste da ESN, onde são utilizados 10^6 pontos para cálculo da BER.

No período de teste, é esperado que a saída da ESN $y(n)$ aproxime a mensagem binária transmitida $m(n)$. Como $m(n) \in \{-1, 1\}$, a decisão $\widehat{m}(n)$ é dada por meio da aplicação da função sinal $\text{sgn}(\cdot)$ sobre a saída $y(n)$ da ESN. Isto é,

$$\widehat{m}(n) = \text{sgn}(y(n)) = \begin{cases} +1, \text{ se } y(n) \geq 0 \\ -1, \text{ se } y(n) < 0 \end{cases} . \tag{4.1}$$

O sinal transmitido é obtido mediante a codificação soma (2.13) da mensagem binária

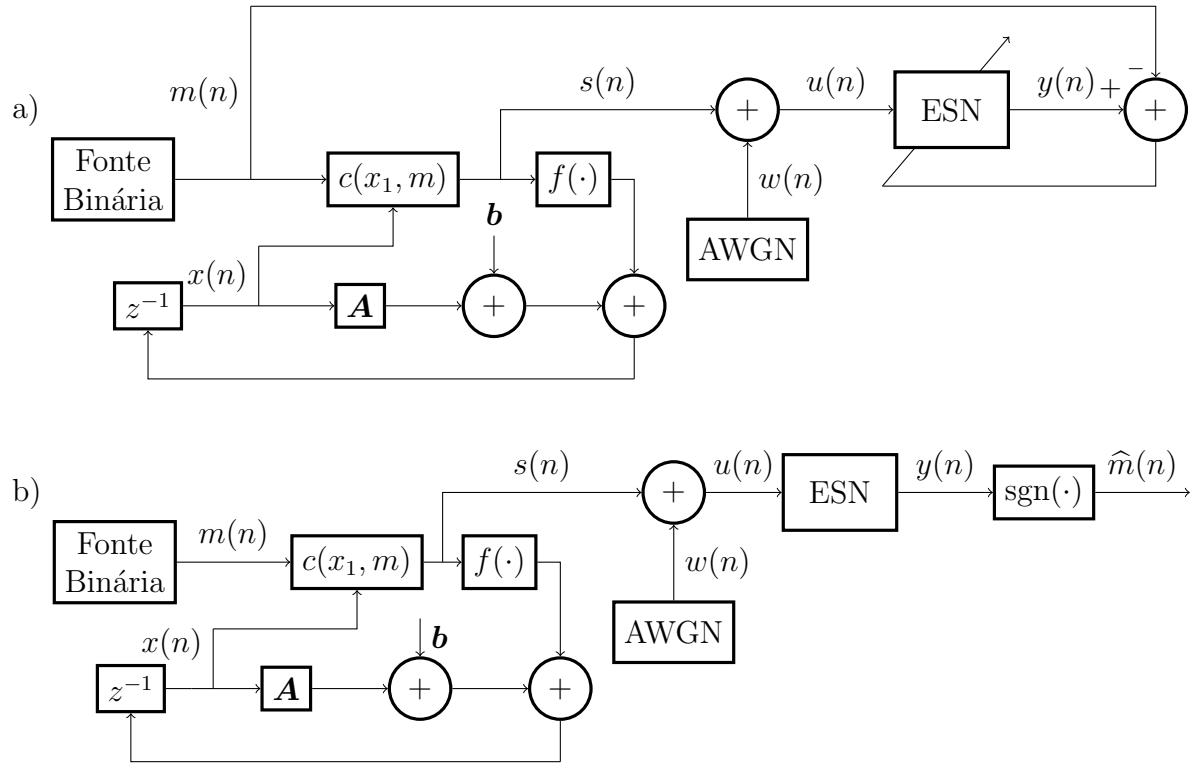

Figura 4.1: Diagrama de blocos do SCBC sujeito à AWGN (Cenário I): a) período de treinamento da ESN; b) período de teste da ESN.

$m(n) \in \{-1, 1\}$ com um sinal caótico obtido pelo mapa tenda inclinada (2.5). O parâmetro da codificação soma foi tomado como $\gamma = 0.4$ e o parâmetro do mapa tenda inclinada como $\alpha = 0.1$.

Ao sinal transmitido, é adicionado AWGN $w(n)$, de modo que o sinal de entrada da ESN é dado por

$$u(n) = s(n) + w(n). \tag{4.2}$$

Na Fig. 4.2 encontram-se exemplos de $m(n)$, $s(n)$ e $u(n)$ para $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB.

4.1.1 Resultados de simulações

Para escolher os parâmetros da ESN a, λ, p e q, gerou-se $x(n)$ com $x_0 \in (-1, 1)$ arbitrário, $\alpha = 0.1$ e $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB. Além de $u(n)$, tanto $\boldsymbol{W}^{\mathrm{in}}$ quanto $\boldsymbol{W}$ foram mantidos fixos até o término da rotina de seleção de parâmetros, descrita nos próximos parágrafos.

Como primeiro passo, foram tomados $\lambda = 0.05$, $p = 0$ e $q = 0.5$. Então, mantendo-os fixos, variou-se a de 0 até 1 em passos de 0.05. Em todos os casos, a ESN foi treinada e testada, calculando-se a BER correspondente. Assim, determina-se o valor $a = a_1^{\mathrm{opt}}$ que

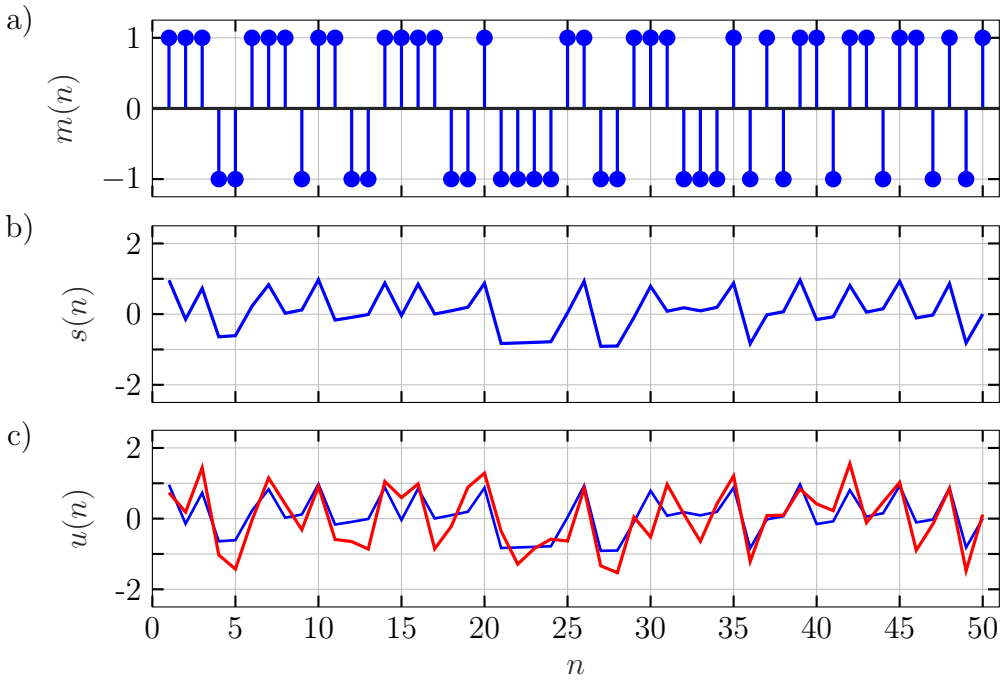

Figura 4.2: Exemplos de sinais do SCBC da Fig. 4.1: a) mensagem binária; b) sinal transmitido; c) sinal transmitido (linha azul) e sinal recebido (linha vermelha).

minimiza a BER. Analogamente, obtém-se λ_1^{opt}, variando-se λ de 0.05 até 1 em passos de 0.05, e mantendo-se os demais parâmetros fixos em $a = a_1^{\text{opt}}$, $p = 0$ e $q = 0.5$. Em seguida, p_1^{opt} e q_1^{opt} foram obtidos, nesta ordem, variando-se p entre 0 e 10 em passos de 0.5 e q entre 0.5 e 10 em passos de 0.5, enquanto os demais parâmetros permanecem fixos nos valores selecionados mais recentes.

Essa rotina é repetida, obtendo-se a_i^{opt}, λ_i^{opt}, p_i^{opt}, q_i^{opt}, $i = 2, 3, \ldots$. A busca continua até que os valores selecionados parem de mudar. A evolução da seleção de parâmetros com o passar da rotina encontra-se na Fig. 4.3. Foram necessárias 16 iterações para que todos os parâmetros parassem de mudar. Os valores finais selecionados, que são utilizados nas próximas simulações desta seção, são $a_\star = 1.00$, $\lambda_\star = 0.10$, $p_\star = 10.00$ e $q_\star = 0.50$.

Na Tabela 4.1 encontram-se todos os parâmetros da rede e do canal utilizados para as simulações deste capítulo.

Após a seleção de parâmetros, o desempenho do SCBC da Fig. 4.1 usando ESN foi avaliado para diferentes valores de SNR_{in}. Na Fig. 4.4, encontram-se exemplos de $m(n)$, $s(n)$, $u(n)$ e $\widehat{m}(n)$ para $\text{SNR}_{\text{in}} = 10.0$ dB.

O desempenho obtido com a integração da ESN ao SCBC foi comparado com o obtido usando-se um filtro de Wiener de 10 coeficientes para equalizar o canal, seguido da técnica de sincronismo caótico para a tomada de decisão. A comparação se deu mediante a

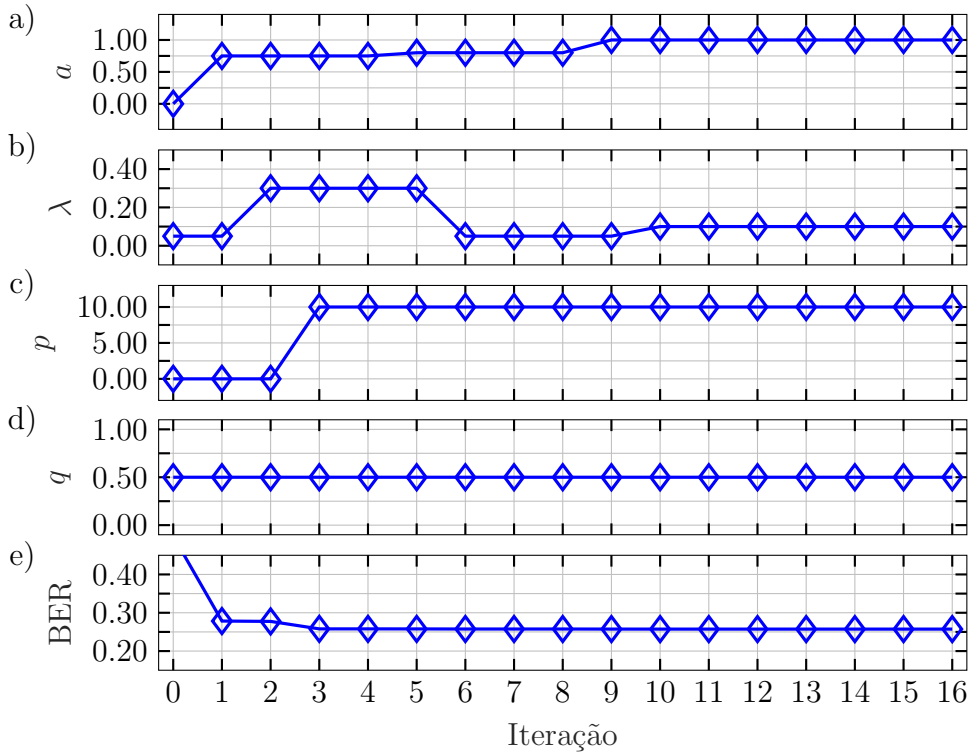

Figura 4.3: Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral de $\boldsymbol{W}$; c) e d) parâmetros das distribuições uniformes usadas para obter $\boldsymbol{W}^{\text{in}}$; e) taxa de erro de bit.

observação da curva de BER em função da SNR do canal. Ao todo, foram feitas 5 realizações com diferentes ESNs, $m(n)$, $x(n)$ e $w(n)$.

Os resultados obtidos são apresentados na Fig. 4.5. Nela, apresentam-se os valores médios de BER obtidos e as barras de erro considerando-se as 5 realizações. Por conveniência, também é apresentado o desempenho do SCBC em questão usando apenas o sincronismo caótico (SC).

Nota-se que a diferença entre o desempenho da ESN em comparação ao filtro de Wiener e sincronismo caótico passa a aumentar consideravelmente a partir de SNR = 10.0 dB.

Tabela 4.1: Parâmetros utilizados no problema do SCBC considerando apenas ruído.

Parâmetro	**Valor**	**Unidade**
N	500	nó
a	1.00	-
λ	0.10	-
p	10.00	-
q	0.50	-
ℓ	200	amostra
L	25000	amostra

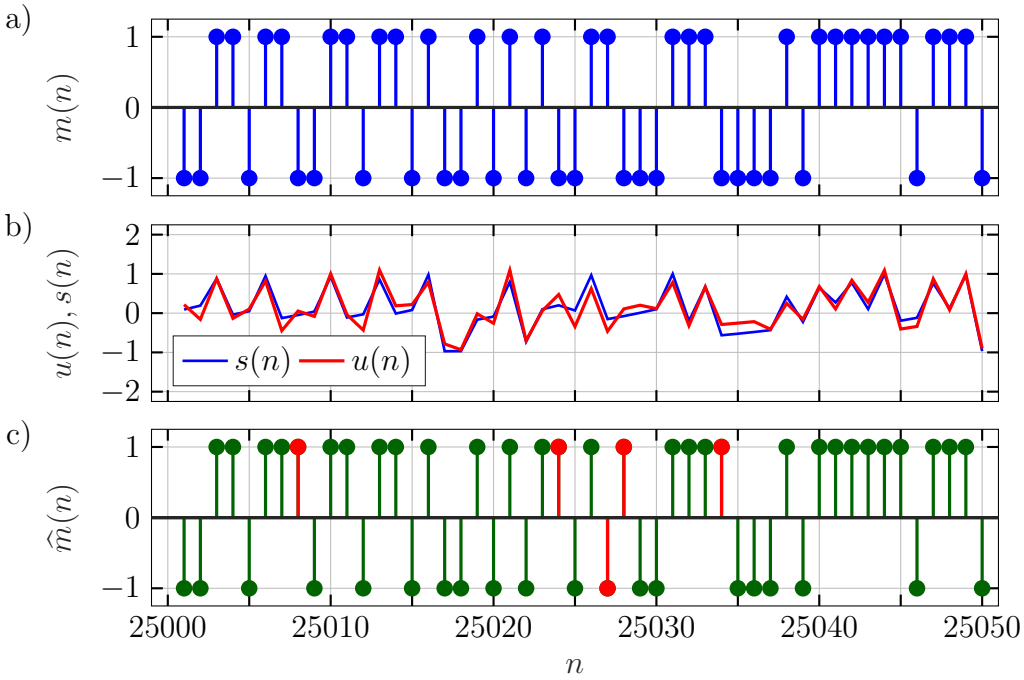

Figura 4.4: Exemplos de sinais do SCBC da Fig. 4.1: a) mensagem binária; b) sinal transmitido (linha azul) e sinal recebido (linha vermelha); c) decisão da ESN. Pontos na cor vermelha indicam decisões errôneas.

Para SNR = 20.0 dB, a BER obtida por meio da ESN é cerca de 1000 vezes menor que a obtida usando-se o FW.

É interessante notar que a curva obtida equalizando-se o canal com o filtro de Wiener é quase a mesma daquela traçada apenas com o sincronismo caótico. Isso está de acordo com o resultado mostrado na Fig. 3.6, obtido e discutido na seção anterior. O filtro de Wiener não é capaz de remover uma quantidade significativa de ruído do sinal transmitido para $|\alpha|$ próximo de 0. Como neste caso $\alpha = 0.1$, a inclusão filtro de Wiener antes da aplicação do método de sincronismo caótico não oferece uma grande melhoria.

Apesar do sistema apresentar uma BER sensivelmente melhor com ESN, ainda são necessárias SNRs relativamente altas para a obtenção de BERs que viabilizem aplicações práticas. Somente para valores de SNR acima de aproximadamente 17.5 dB que o sistema consegue atingir patamares de BER abaixo de 10^{-4}.

4.2 Cenário II: somente canal

Nesta etapa, verificou-se o desempenho do SCBC ao incluir um canal sem ruído nele. O canal considerado é dado por $H(z) = h_0 + z^{-1} + h_0 z^{-2}$.

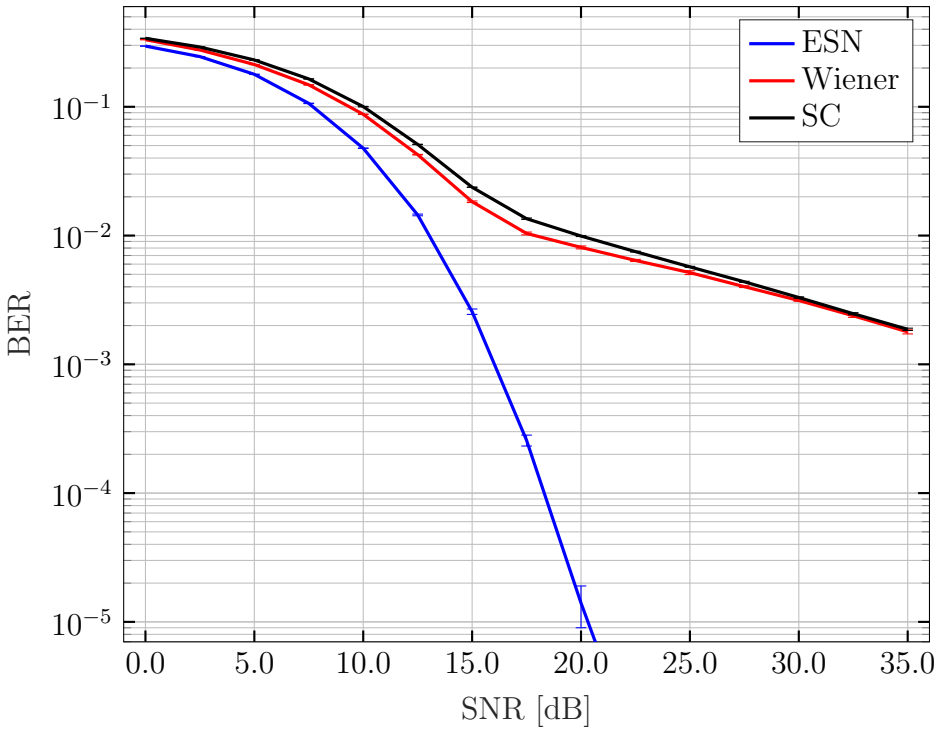

Figura 4.5: Curvas de BER em função da SNR no Cenário I.

O diagrama em blocos deste cenário é apresentado na Fig. 4.6. Da mesma maneira que na seção anterior, mostra-se na parte a) da figura o período de treinamento, que faz uso de 25200 pontos de $m(n)$. Na parte b), é mostrado o período de teste, no qual calcula-se a BER usando 10^6 pontos.

O sinal transmitido é obtido a partir da codificação soma (2.13) da mensagem binária $m(n)$ com um sinal caótico $x(n)$ gerado pelo mapa tenda inclinada (2.5). Ou seja, o sinal transmitido é obtido da mesma forma que na seção anterior, com $\gamma = 0.4$ e $\alpha = 0.1$.

O sinal transmitido $s(n)$ passa pelo canal $H(z) = h_0 + z^{-1} + h_0 z^{-2}$ antes de chegar ao receptor. Deste modo, o sinal recebido é dado por

$$u(n) = h_0 s(n) + s(n-1) + h_0 s(n-2). \tag{4.3}$$

Na Fig. 4.7 encontram-se exemplos de $m(n)$, $s(n)$ e $u(n)$ para $h_0 = 1$. Observando-se a parte c) da Fig. 4.7, percebe-se que o sinal recebido acompanha o sinal transmitido, mas de forma mais suave, sem as variações rápidas em amplitude. Isso ocorre pois o canal $H(z)$ utilizado é um canal passa-baixas, conforme pode-se constatar analisando-se a Fig. 4.8. A mesma traz curvas normalizadas, bem como um mapa de cores, de $|H(\omega)|/H_{\text{MAX}}$ para h_0 variando de 0.1 até 1 em passos de 0.1. Aqui, H_{MAX} é o valor máximo de $|H(\omega)|$. Percebe-se na parte a) que quanto maior h_0, menor a largura de banda do filtro. Da parte b), pode-se inferir que há nulos espectrais sempre que h_0 for suficientemente grande.

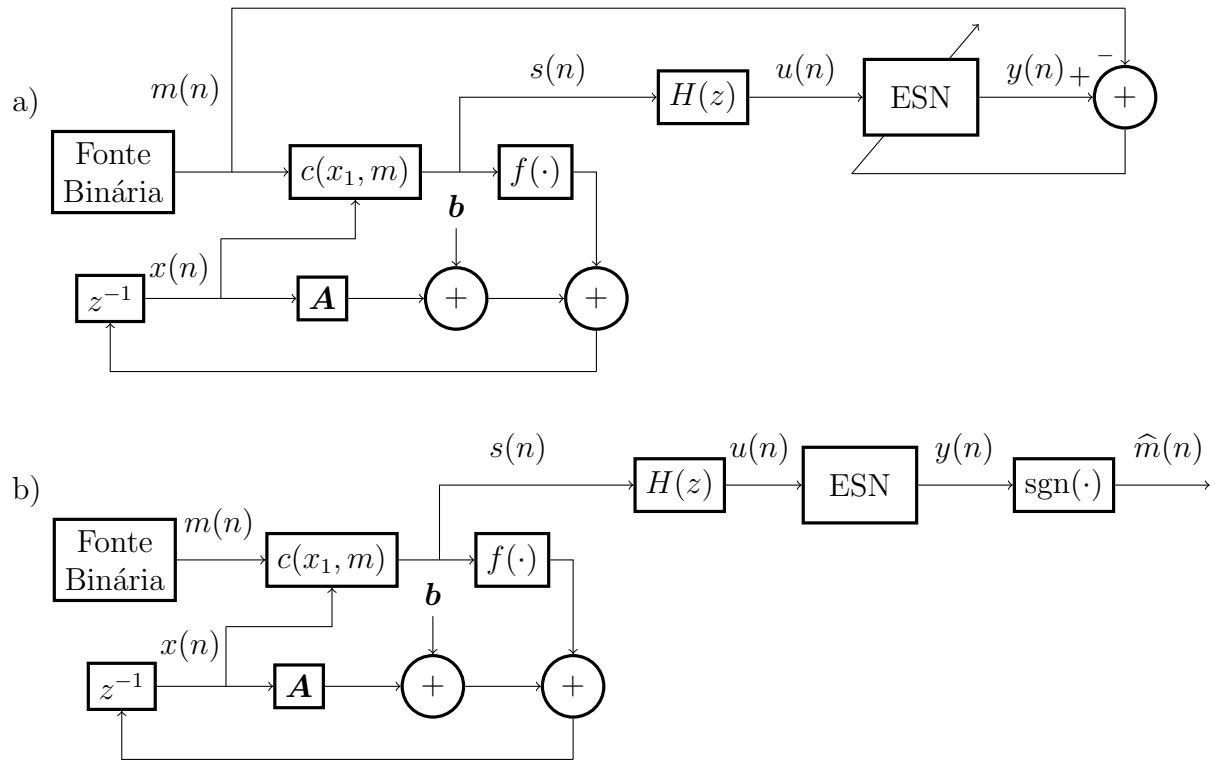

Figura 4.6: Diagrama de blocos do SCBC somente com canal (Cenário II): a) período de treinamento da ESN; b) período de teste ESN.

4.2.1 Resultados de simulações

Para escolher os parâmetros da ESN a, λ, p e q, gerou-se $x(n)$ com $x_0 \in (-1, 1)$ arbitrário, $\alpha = 0.1$ e $h_0 = 0.5$. Além de $u(n)$, ambos $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}$ foram mantidos fixos até o término da rotina de seleção de parâmetros, descrita nos próximos parágrafos.

Como primeiro passo, foram tomados $\lambda = 0.05$, $p = 0$ e $q = 0.5$. Então, mantendo-os fixos, variou-se a de 0 até 1 em passos de 0.05. Em todos os casos, a ESN foi treinada e testada, calculando-se a BER correspondente. Assim, determina-se o valor $a = a_1^{\text{opt}}$ que minimiza a BER. Analogamente, obtém-se λ_1^{opt}, variando-se λ de 0.05 até 1 em passos de 0.05, e mantendo-se os demais parâmetros fixos em $a = a_1^{\text{opt}}$, $p = 0$ e $q = 0.5$. Em seguida, p_1^{opt} e q_1^{opt} foram obtidos, nesta ordem, variando-se p entre 0 e 10 em passos de 0.5 e q entre 0.5 e 10 em passos de 0.5, enquanto os demais parâmetros permanecem fixos nos valores selecionados mais recentes.

Essa rotina é repetida, obtendo-se a_i^{opt}, λ_i^{opt}, p_i^{opt}, q_i^{opt}, $i = 2, 3, \ldots$. A busca continua até que os valores selecionados parem de mudar. A evolução da seleção de parâmetros com o passar da rotina encontra-se na Fig. 4.9. Levou 12 iterações para que todos os parâmetros parassem de mudar. Os valores finais selecionados, que são utilizados nas

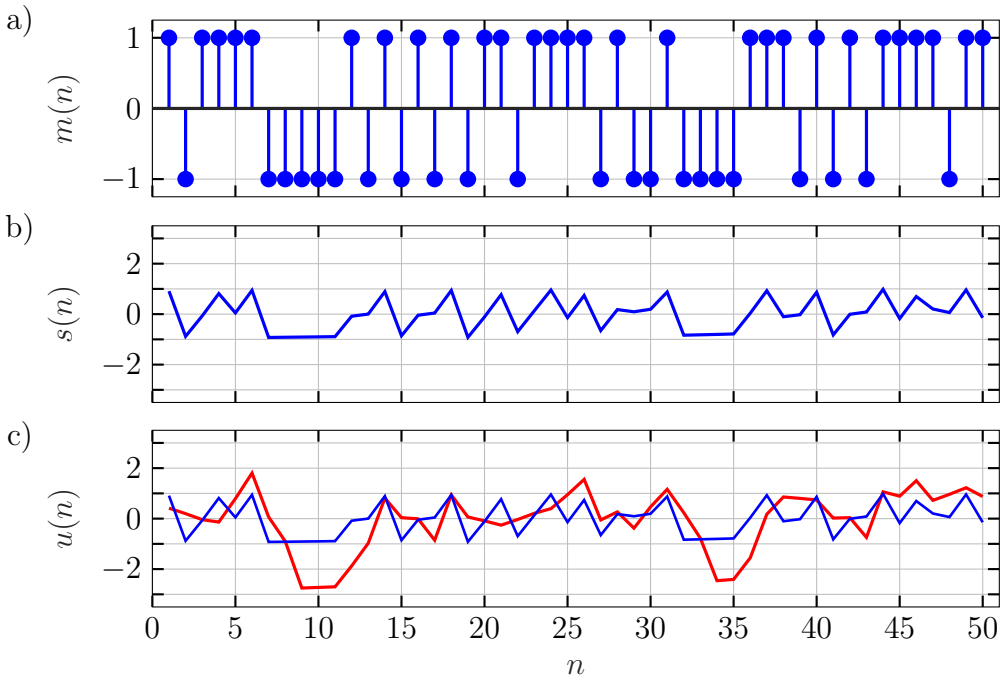

Figura 4.7: Exemplos de sinais do SCBC da Fig. 4.6: a) mensagem binária; b) sinal transmitido; c) sinal transmitido (linha azul) e sinal recebido (linha vermelha).

próximas simulações desta seção, são $a_\star = 0.95$, $\lambda_\star = 0.05$, $p_\star = 3.50$ e $q_\star = 8.00$.

Na Tabela 4.2 encontram-se todos os parâmetros da rede e do canal utilizados para as simulações deste capítulo.

Tabela 4.2: Parâmetros utilizados no problema do SCBC considerando apenas canal.

Parâmetro	**Valor**	**Unidade**
N	500	nó
a	0.95	-
λ	0.05	-
p	3.50	-
q	8.00	-
ℓ	200	amostra
L	25000	amostra

O desempenho do SCBC da Fig. 4.6 usando ESN foi avaliado para h_0 variando de 0.0 até 1.0 em passos de 0.025. A Fig. 4.10 mostra exemplos de $m(n)$, $s(n)$, $u(n)$ e $\hat{m}(n)$ para $h_0 = 1$.

O desempenho obtido com o SCBC empregando a ESN foi comparado com o desempenho do mesmo SCBC usando um filtro de Wiener de 10 coeficientes para equalizar o canal e sincronismo caótico para a tomada de decisão. A comparação se deu mediante a observação da curva de BER em função do coeficiente h_0 do canal $H(z) = h_0 + z^{-1} + h_0 z^{-2}$.

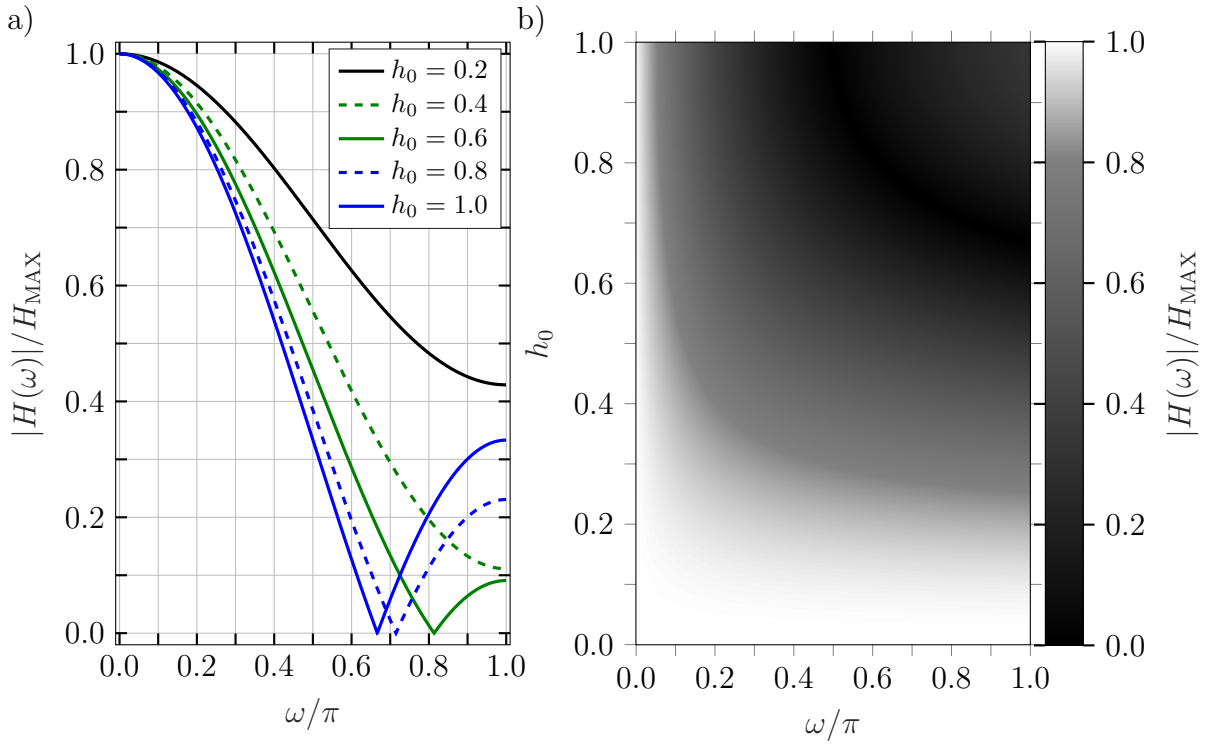

Figura 4.8: Resposta em frequência normalizada do filtro $H(z) = h_0 + z^{-1} + h_0 z^{-2}$: a) curvas para $h_0 = 0.2, 0.4, \ldots, 1.0$; b) mapa de cores.

Foram feitas 10 realizações, cada uma considerando ESN, $x(n)$ e $m(n)$ distintos.

Pelo fato do canal empregado ser um atrasador de uma amostra para $h_0 = 0$, decidiu-se avaliar o desempenho da ESN de duas formas. Na primeira, treina-se a ESN para dado $u(n)$ estimar $m(n)$, exatamente como feito na seção precedente. Na segunda, treina-se a ESN para dado $u(n)$ e $u(n+1)$, estimar $m(n)$.

A razão para se ter feito isso é que, substituindo-se $h_0 = 0$ em (4.3), tem-se $u(n) = s(n-1)$. Isso significa que $u(n)$ não possui relação com $m(n)$, conforme (2.13), e faltariam informações para a rede fazer uma boa estimativa.

Na Fig. 4.11 encontram-se os valores médios de BER obtidos e as barras de erro correspondentes às dez realizações feitas. Percebe-se claramente que usando-se uma única amostra de $u(n)$, a ESN não é capaz de tomar uma decisão adequada.

A primeira coisa que salta aos olhos analisando-se a Fig. 4.11 é que, usando-se diretamente o sincronismo caótico (SC), o SCBC não funciona para nenhum valor de h_0 considerado. Isso ocorre porque, nesse caso, o escravo não é capaz de sincronizar com o mestre.

Nota-se também que a inclusão do filtro de Wiener leva a resultados substancialmente melhores em comparação ao uso de SC isoladamente, especialmente para $h_0 < 0.4$. Aqui,

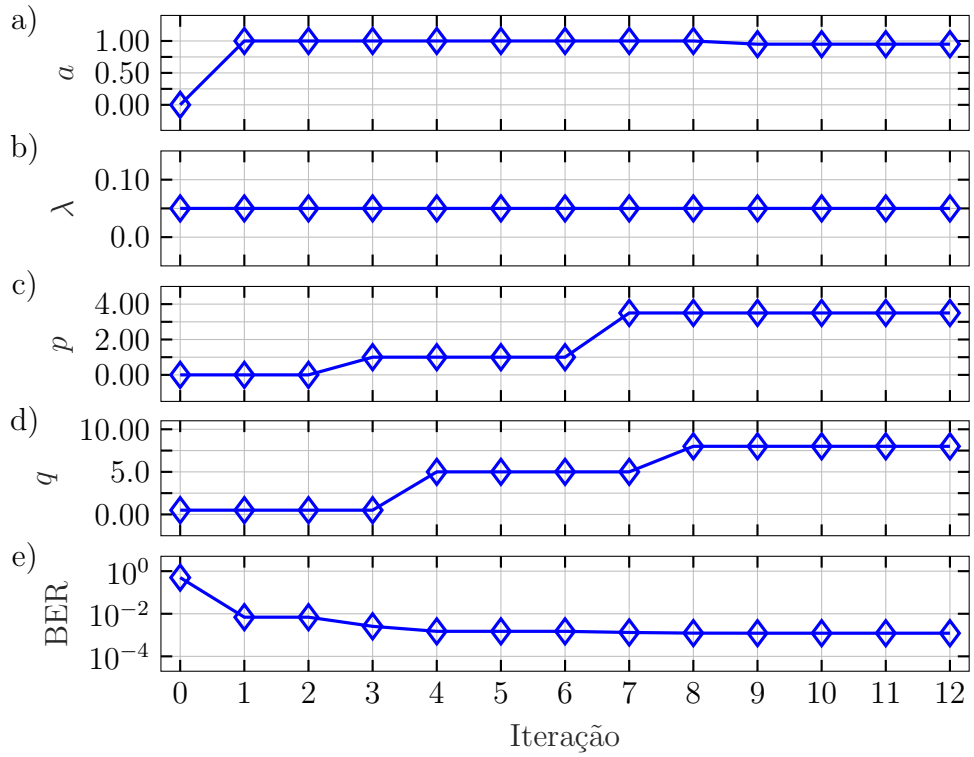

Figura 4.9: Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral; c) e d) parâmetros da distribuições uniformes usadas para obter $\boldsymbol{W}^{\text{in}}$; e) taxa de erro de bit.

diferentemente do Cenário I abordado na seção anterior, o filtro é capaz de mitigar os efeitos introduzidos pelo canal.

O resultado obtido mediante a integração da ESN no SCBC é bastante positivo quando $h_0 < 0.25$, utilizando-se 2 amostras para estimar a mensagem transmitida. Assim como a BER calculada com o filtro de Wiener, a BER obtida usando-se a ESN com duas amostras tende a aumentar conforme h_0 aumenta.

Conforme já observado anteriormente, no caso em que $h_0 = 0$, tem-se $H(z) = z^{-1}$ e $u(n) = s(n-1)$. Consequentemente, quando se utiliza somente uma amostra para a estimação, a ESN tenta predizer qual a mensagem $m(n)$ tendo como entrada $s(n-1)$, que não tem relação alguma com $m(n)$. Conforme pode ser constatado da Fig. 4.11, a BER é de 0.5 nesta situação.

Ao passo que h_0 aumenta, $u(n)$ definido em (4.2) passa a carregar informação sobre $m(n)$ graças à parcela $h_0 s(n)$. Como consequência, a BER melhora até em torno de $h_0 = 0.1$, a partir de onde o efeito introduzido pelo canal se torna intenso o suficiente para passar a piorar a BER.

No caso em que a ESN é treinada com $u(n)$ e $u(n+1)$ para estimar $m(n)$, não ocorre

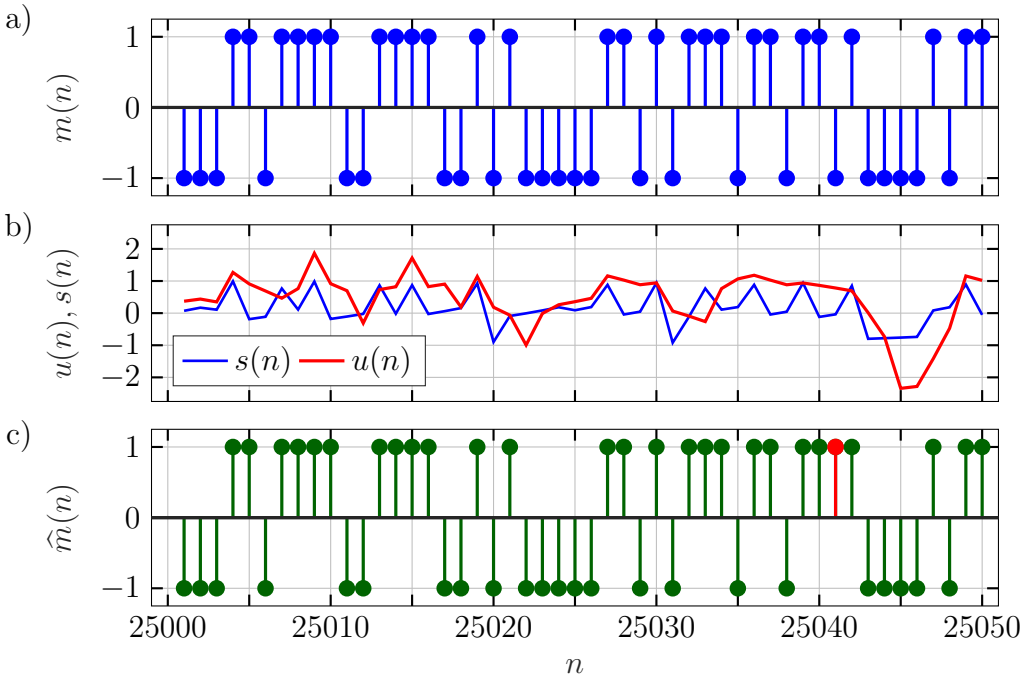

Figura 4.10: Exemplos de sinais do SCBC da Fig. 4.6: a) mensagem binária; b) sinal transmitido (linha azul) e sinal recebido (linha vermelha); c) decisão da ESN. Pontos na cor vermelha indicam decisões errôneas.

o problema descrito anteriormente. No caso em que $h_0 = 0$, a entrada $u(n) = s(n-1)$ não tem relação com $m(n)$, mas a próxima amostra $u(n+1) = s(n)$ tem. Aqui é possível afirmar que, na média, a BER aumenta conforme h_0 aumenta. Também é interessante destacar que para $h_0 > 0.3$ a melhora fornecida pelo uso de uma amostra adicional de $u(n)$ desaparece e ambas as curvas tornam-se aproximadamente coincidentes.

A BER resultante usando-se o filtro de Wiener demonstra ter atingido um patamar em 10^{-1}, ao passo que a BER fornecida pelo SCBC com ESN continua aumentando gradativamente de 10^{-3} até 10^{-2}. Inicialmente o aumento é mais acentuado, reduzindo gradativamente com o crescimento de h_0. Isso está em conformidade com o que se observa na Fig. 4.8. Quanto menor h_0, maior a largura de banda do filtro e menor é a mudança entre o sinal de saída do canal $r(n)$ e o sinal transmitido $s(n)$.

Ademais, da Fig. 4.8 é aparente que a mudança de largura de banda causada por incrementos em h_0 é mais acentuada para h_0 mais próximo de 0, tornando-se quase imperceptível para h_0 próximo de 1. Isso justifica a inclinação maior da curva de BER para valores de h_0 menores.

Por fim, cabe destacar que os parâmetros da ESN foram otimizados para $h_0 = 0.5$ usando-se uma única amostra para estimativa da mensagem transmitida. Ainda assim, não ocorreu *overfitting* e o SCBC empregando ESN foi capaz de, para alguns valores de

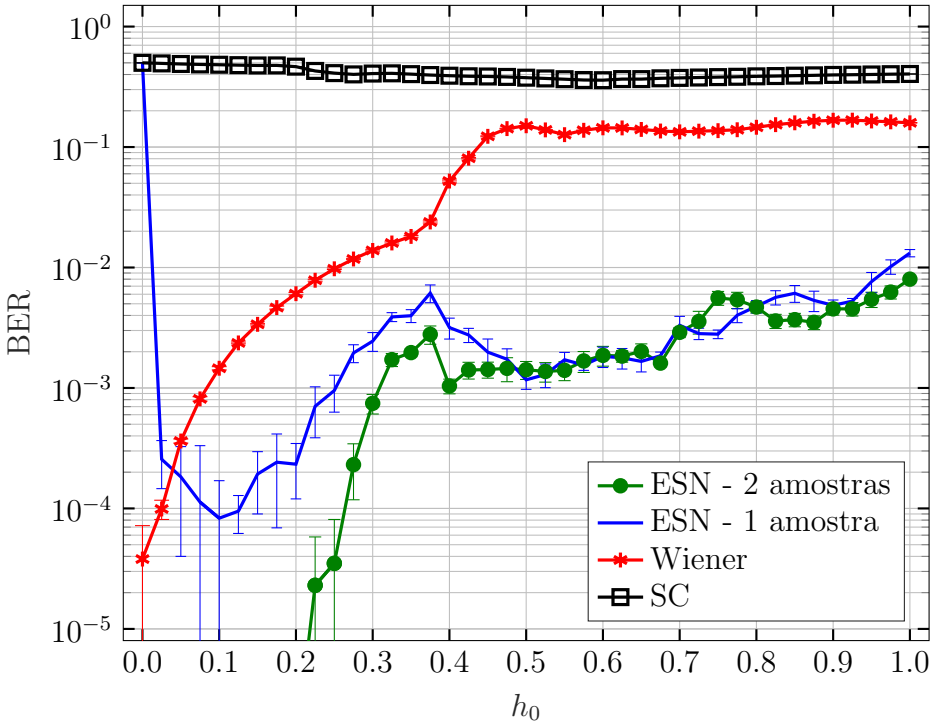

Figura 4.11: Curvas de BER em função de h_0 no Cenário II.

h_0, atingir valores de BER menores que o atingido para $h_0 = 0.5$.

Os casos estudados nessa e na seção anterior, são casos particulares do cenário investigado na próxima e última seção deste capítulo.

4.3 Cenário III: canal e ruído

Este capítulo é encerrado com a análise do caso em que existe tanto o canal $H(z) = h_0 + z^{-1} + h_0 z^{-2}$ quanto AWGN no SCBC. O diagrama em blocos do problema encontra-se na Fig. 4.12.

Neste caso, objetivo é integrar uma ESN ao SCBC de modo a mitigar os efeitos introduzidos pelo canal e ruído e obter a melhor estimativa possível da mensagem binária transmitida.

Na parte a) da Fig. 4.12 é representado o período de treinamento. Nele, faz-se uso de 25200 amostras para determinação dos pesos da ESN, das quais 200 são descartadas para permitir que a rede lide com transientes iniciais. Na parte b), é ilustrado o período de teste da ESN integrada ao SCBC, no qual se faz uso de 1 milhão de pontos para cálculo da BER. Novamente, é produzido uma mensagem binária $m(n) \in \{-1, 1\}$ aleatoriamente de maneira equiprovável. Esta mensagem é codificada com um sinal caótico gerado pelo

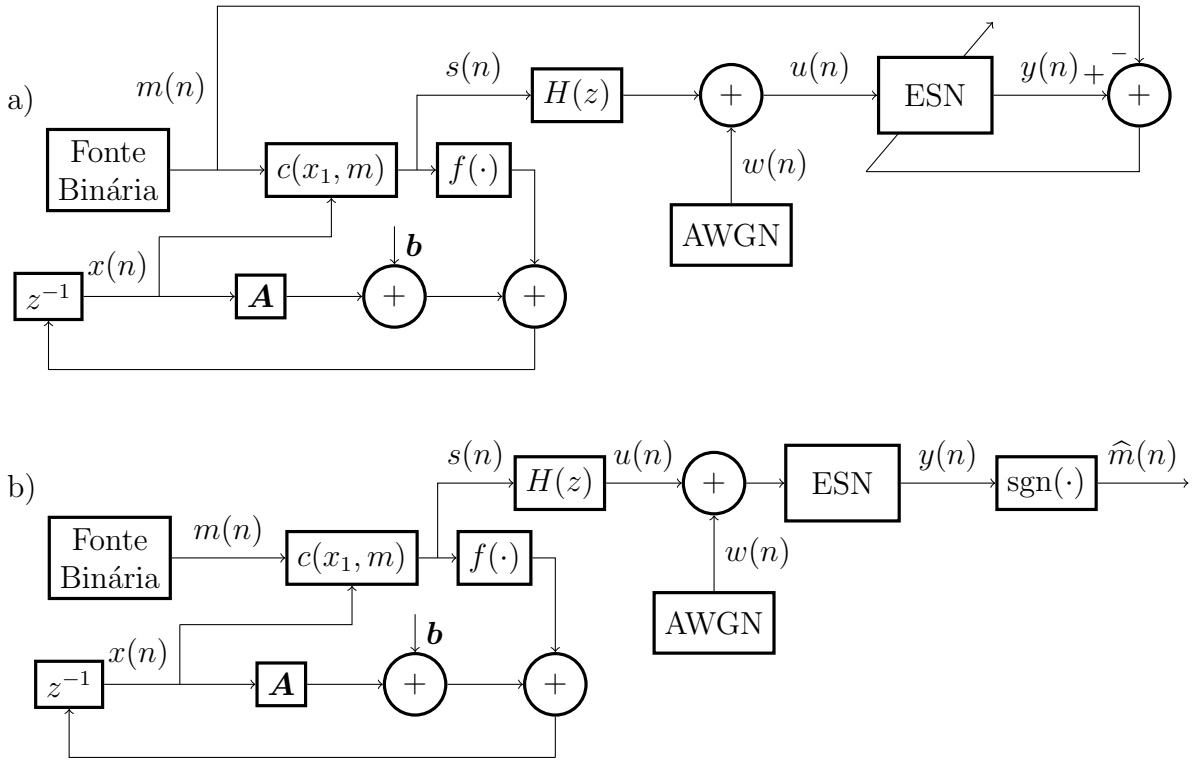

Figura 4.12: Diagrama de blocos do SCBC com canal e ruído (Cenário III): a) período de treinamento da ESN; b) período de teste da ESN.

mapa tenda inclinada (2.5) com parâmetro $\alpha = 0.1$, por meio da codificação soma (2.13) com parâmetro $\gamma = 0.4$.

Na Fig. 4.13 encontram-se exemplos de $m(n)$, $s(n)$ e $u(n)$, com $h_0 = 0.5$ e SNR $=$ 5.0 dB.

Considerando-se ambos efeitos do canal $H(z) = h_0 + z^{-1} + h_0 z^{-2}$ e ruído AWGN $w(n)$, o sinal recebido é dado por

$$u(n) = h_0 s(n) + s(n-1) + h_0 s(n-2) + w(n). \tag{4.4}$$

Conforme observado na seção anterior, ao considerar o canal $H(z)$, um resultado melhor é obtido usando-se $u(n)$ e $u(n+1)$ para estimar $m(n)$. Assim, o sinal de entrada da ESN no instante n é $\begin{bmatrix} u(n+1) & u(n) \end{bmatrix}^T$.

4.3.1 Resultados de simulações

Para escolher os parâmetros da ESN a, λ, p e q, gerou-se $x(n)$ com $x_0 \in (-1, 1)$ arbitrário, $\alpha = 0.1$, $h_0 = 0.5$ e $\text{SNR}_{\text{in}} = 2.0$ dB. Além de $u(n)$, ambos $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}$ foram mantidos fixos até o término da rotina de seleção de parâmetros, descrita nos próximos

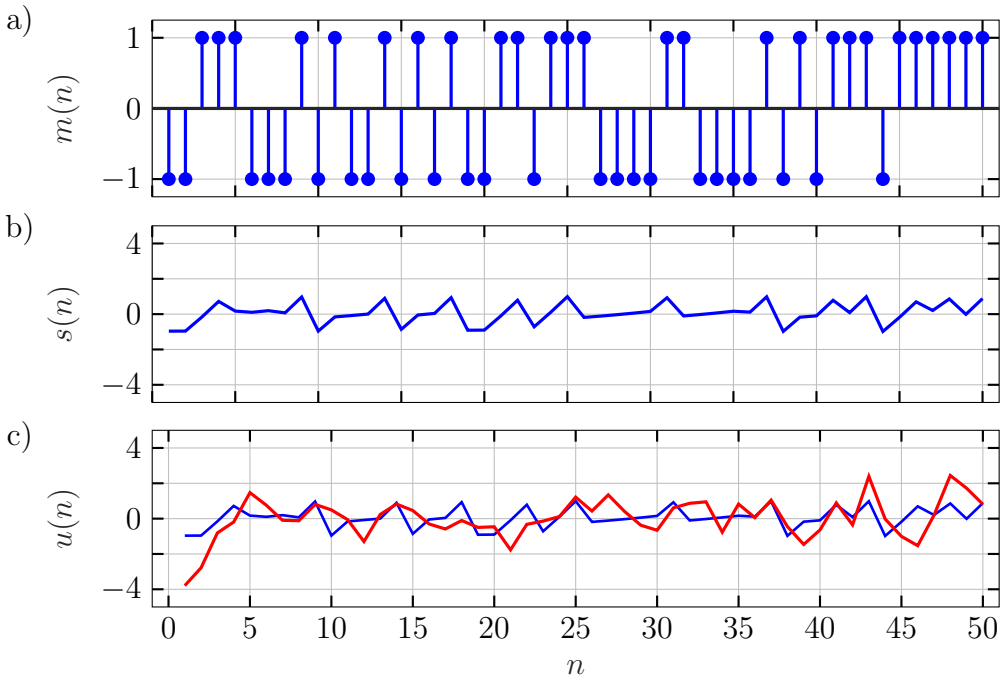

Figura 4.13: Exemplos de sinais do SCBC da Fig. 4.12: a) mensagem binária; b) sinal transmitido; c) sinal transmitido (linha azul) e sinal recebido (linha vermelha).

parágrafos.

Como primeiro passo, foram tomados $\lambda = 0.05$, $p = 0$ e $q = 0.5$. Então, mantendo-os fixos, variou-se a de 0 até 1 em passos de 0.05. Em todos os casos, a ESN foi treinada e testada, calculando-se a BER correspondente. Assim, determina-se o valor $a = a_1^{\text{opt}}$ que minimiza a BER. Analogamente, obtém-se λ_1^{opt}, variando-se λ de 0.05 até 1 em passos de 0.05, e mantendo-se os demais parâmetros fixos em $a = a_1^{\text{opt}}$, $p = 0$ e $q = 0.5$. Em seguida, p_1^{opt} e q_1^{opt} foram obtidos, nesta ordem, variando-se p entre 0 e 10 em passos de 0.5 e q entre 0.5 e 10 em passos de 0.5, enquanto os demais parâmetros permanecem fixos nos valores selecionados mais recentes.

Essa rotina é repetida, obtendo-se a_i^{opt}, λ_i^{opt}, p_i^{opt}, q_i^{opt}, $i = 2, 3, \ldots$. A busca continua até que os valores selecionados parem de mudar. A evolução da seleção de parâmetros com o passar da rotina encontra-se na Fig. 4.14. Levou 12 iterações para que todos os parâmetros parassem de mudar. Os valores finais selecionados, que são utilizados nas próximas simulações desta seção, são $a_\star = 0.95$, $\lambda_\star = 0.10$, $p_\star = 1.50$ e $q_\star = 0.50$.

Na Tabela 4.3 encontram-se todos os parâmetros da rede e do canal utilizados para as simulações desta seção.

Neste ponto, a ESN está pronta para ter seu desempenho avaliado. Tomando-se os parâmetros ótimos e $h_0 = 0, 0.05, 0.1, \ldots, 1$, passou-se a variar a SNR de 0.0 dB até 35.0

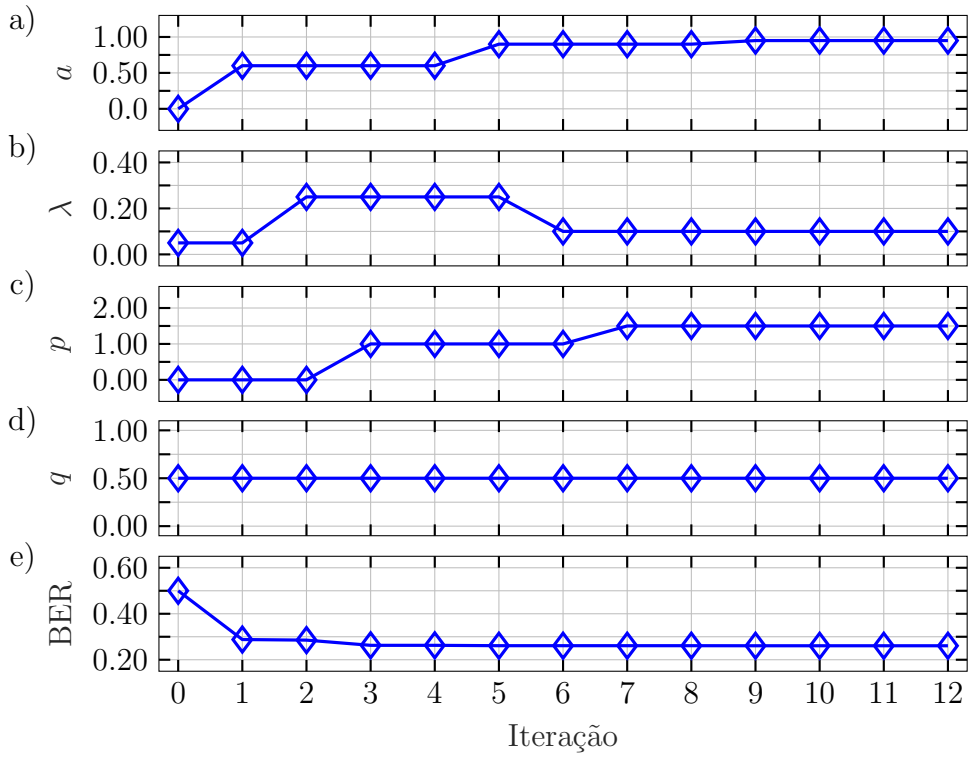

Figura 4.14: Evolução dos parâmetros durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral; c) e d) parâmetros das distribuições uniformes usadas para gerar $\boldsymbol{W}^{\text{in}}$; e) taxa de erro de bit.

dB em passos de 2.5 dB para cada valor de h_0. Para cada par de valores de SNR e h_0, a ESN foi treinada com 25200 pontos para determinação dos pesos de $\boldsymbol{W}_{\text{out}}$ e testada com 10^6 pontos para cálculo do valor de BER correspondente. Ao final, é possível obter o mapa BER $\times$ SNR $\times$ h_0.

Na Fig. 4.15 encontram-se exemplos de $m(n)$, $s(n)$, $u(n)$ e $\widehat{m}(n)$ para SNR = 20.0 dB e $h_0 = 0.5$.

Os mapas de cores obtidos encontram-se na Fig. 4.16. Dos intervalos de SNR e h_0 utilizados, considerou-se somente SNR $>$ 10.0 dB e $h_0 < 0.4$, que é onde a ESN opera

Tabela 4.3: Parâmetros utilizados no problema do SCBC considerando ruído e canal.

Parâmetro	**Valor**	**Unidade**
N	500	nó
a	0.95	-
λ	0.10	-
p	1.50	-
q	0.50	-
ℓ	200	amostra
L	25000	amostra

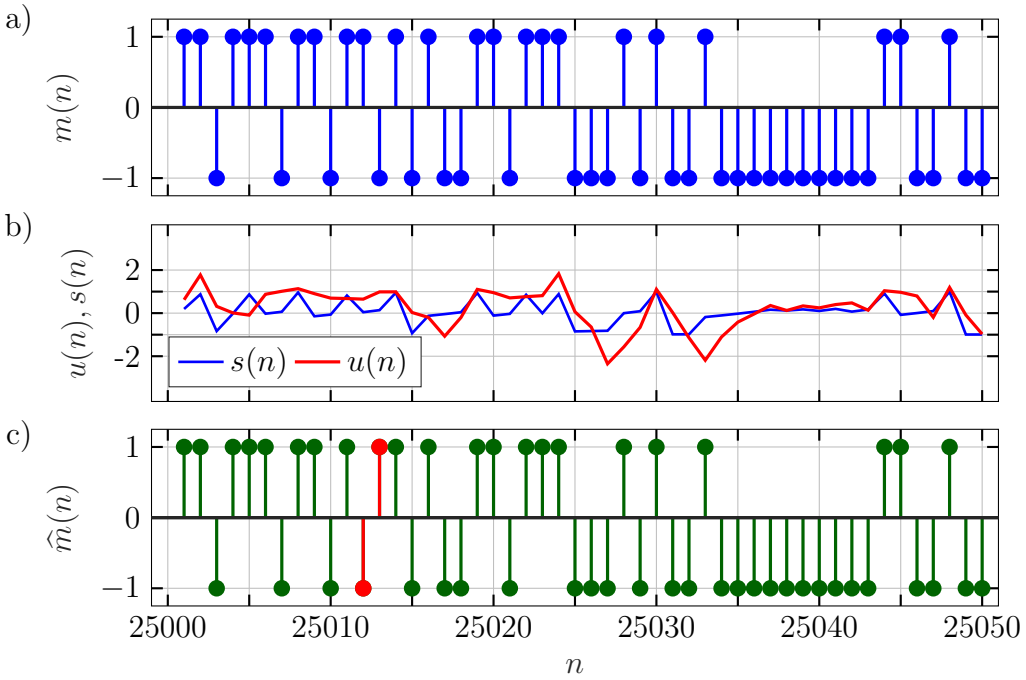

Figura 4.15: Exemplos de sinais do SCBC da Fig. 4.12: a) mensagem binária; b) sinal transmitido (linha azul) e sinal recebido (linha vermelha); c) decisão da ESN. Pontos na cor vermelha indicam decisões errôneas.

com os menores valores de BER.

Conforme esperado, devido às análises apresentadas nas duas seções anteriores, a técnica de sincronismo caótico (SC) por si só não é capaz de melhorar a BER.

Para melhor visualização, na Fig. 4.17 (a) encontram-se curvas de BER em função de h_0 obtidas mantendo-se a SNR fixa em 20.0 dB, considerando-se a média de 10 realizações e as barras de erro correspondentes. Dela, nota-se que para $h_0 < 0.3$ a ESN é capaz de atingir valores de BER consideravelmente melhores que o fornecido pelo filtro de Wiener em conjunto com a técnica de sincronismo caótico. De $h_0 = 0.3$ em diante, ambas as técnicas atingem patamares em valores de BER não muito distantes. A diferença entre o uso de 1 e 2 amostras para estimativa da mensagem transmitida é notável para $h_0 < 0.5$.

Para $h_0 = 1.0$, o canal filtra grande parte do sinal transmitido, conforme pode-se constatar da Fig. 4.8. Isto, somado com o efeito introduzido pelo ruído AWGN, dificulta consideravelmente o problema. A BER para $h_0 = 1$ está um pouco acima de 10^{-2}, o que está de acordo com a curva referente a ESN apresentada na Fig. 4.11: no caso em que $h_0 = 1$ e SNR $\to \infty$, a BER tende a 10^{-2}.

A fim de observar a influência da SNR na BER, na Fig. 4.17 (b) encontram-se as curvas de BER em função de SNR mantendo-se h_0 fixo em 0.1. Até SNR = 12.5 dB o desempenho da ESN e o do filtro de Wiener em conjunto com a técnica de sincronismo

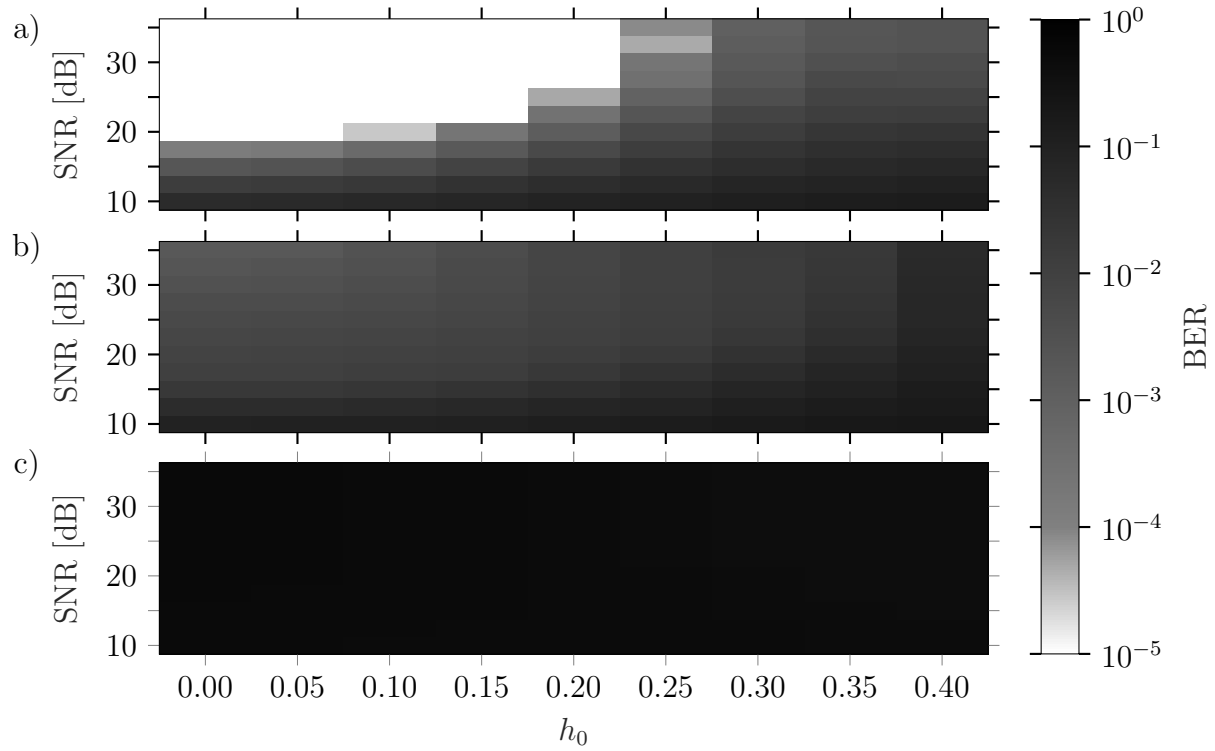

Figura 4.16: Mapas de BER × SNR × h_0 do Cenário III: a) ESN (2 amostras); b) filtro de Wiener seguido de sincronismo caótico; c) somente sincronismo caótico.

caótico são próximos. Para valores de SNR maiores que 12.5 dB, o desempenho da ESN passa a melhorar substancialmente, atingindo valores muito melhores que os obtidos por meio do filtro de Wiener em conjunto com sincronismo caótico.

A diferença de desempenho da ESN ao usar-se 1 ou 2 amostras para estimar a mensagem transmitida é evidente em todo intervalo considerado. Outrossim, o filtro de Wiener conseguiu superar o desempenho da ESN com 1 amostra para todos valores de SNR considerados. Contudo, pelo resultado apresentado na Fig. 4.11, espera-se que a partir de algum valor de SNR suficientemente grande, esse resultado se inverta.

Este capítulo é encerrado com as conclusões obtidas mediante as análises dos resultados alcançados nos três cenários considerados.

4.4 Conclusões

O uso de uma ESN como receptor do SCBC [27] traz resultados satisfatórios dependendo da SNR e do parâmetro h_0 do canal. Em alguns casos, como para SNR > 17.5 dB no Cenário I e $h_0 < 0.25$ no Cenário II, a BER atinge valores menores que 10^{-4}.

Comparando-se os desempenhos apresentados pela técnica de sincronismo caótico em

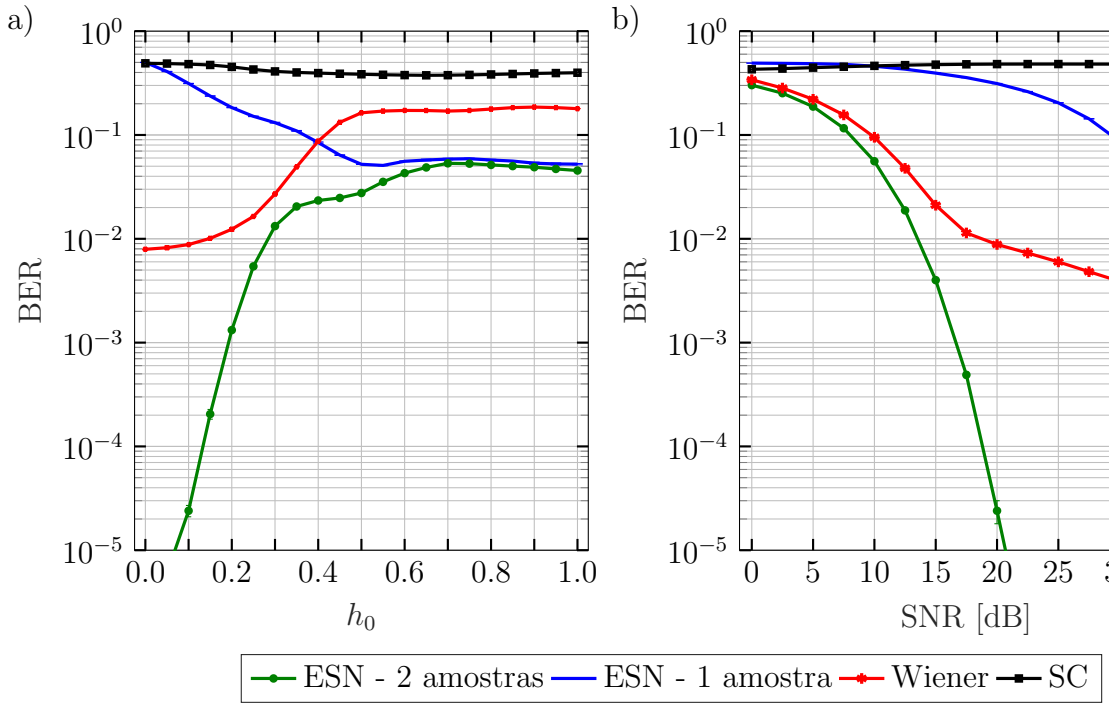

Figura 4.17: Curvas de BER do Cenário III: a) SNR fixa em 20.0 dB; b) h_0 fixo em 0.1.

ambos cenários, chega-se a conclusão de que qualquer filtragem introduzida pelo canal é desastrosa. Tanto para h_0 próximo a zero, quanto para h_0 próximo de um, a BER manteve-se em meio. Num canal onde só existe AWGN, a BER diminui com o aumento da SNR, contudo os resultados não são promissores.

Por outro lado, o SCBC usando filtro de Wiener se beneficia intensamente da filtragem de $H(z)$, mas não consegue oferecer quase que melhora alguma no caso em que o canal é puramente AWGN, o que era esperado por isso e pela análise de (2.52).

O estudo conduzido no Cenário III revelou que a existência simultânea de ruído e filtragem do canal restringe de sobremaneira os intervalos de SNR e h_0 que levam a valores de BER, obtidos com o uso de ESN, adequados. Estando a SNR fixa em 20.0 dB, somente para $h_0 < 0.15$ consegue-se alcançar valores de BER próximos e menores que 10^{-4}. Por outro lado, se $h_0 = 0.1$, então a SNR necessária para ter uma BER menor que 10^{-4} é em torno de 20.0 dB.

Todavia, os resultados obtidos são positivos, no sentido de que foi sempre possível superar o desempenho obtido usando-se o filtro de Wiener seguido da técnica de sincronismo caótico, mesmo que se tenha usado duas amostras para estimar a mensagem transmitida quando da presença de $H(z)$.

Capítulo 5

Predição de sinais musicais

Nesta última aplicação estudada, investiga-se a capacidade de uma ESN emular a criatividade musical humana. Dado um sinal musical contendo informação referente às notas tocadas de uma peça, o objetivo é predizer quais seriam os atributos da nota seguinte, dados os atributos da nota anterior.

O estudo detalhado neste capítulo é profundamente baseado em [32], onde aborda-se o mesmo problema. As diferenças primordiais entre ambos trabalhos, encontram-se na escolha do músico escolhido para compor o banco de dados para treinamento e teste da rede, bem como na maneira pela qual os parâmetros desta são otimizados. Enquanto em [32] utiliza-se peças do prodígio Wolfgang Amadeus Mozart ($\star$1756 - †1791), aqui o artista escolhido é o polonês Frédéric Chopin ($\star$1810 - †1849).

Contudo, também é importante relatar que a ausência de uma amostra do sistema em funcionamento, bem como a carência de uma medida de desempenho significativa musicalmente em [32] foram fatores motivacionais desta investigação. Aqui, um exemplo de resultado obtido, bem como um parâmetro de desempenho - simples, porém com teor musical - são apresentados.

Outros estudos similares podem ser encontrados em [33,98]. No primeiro, que também utiliza arquivos MIDI como banco de dados, utiliza-se uma RNN e matrizes conhecidas como *piano rolls*[1] para, como aqui, predizer sequências musicais. No entanto, diferentemente do que é feito aqui, somente a altura da nota é predita, enquanto que a duração correspondente é estabelecida previamente. Lá também não é feita distinção entre uma nota tocada repetidamente e a mesma nota tocada uma única vez, mas que permanece soando. No segundo trabalho, um sistema para aprendizagem em tempo real e não supervisionado é proposto para predição de sinais de áudio. Enquanto aqui utilizam-se informações musicais de peças reais, lá o sistema é testado com "gravações informais, curtas e de baixa qualidade de *beat boxing* bastante simplificado" e "gravações formais de

[1]Matrizes que carregam informações referentes a altura das notas tocadas em suas colunas e a duração correspondente em suas linhas [33].

alta qualidade de sequências de bateria".

O restante deste capítulo está dividido em três seções. Na Seção 5.1, define-se formalmente o problema de interesse. Na Seção 5.2, apresentam-se os resultados obtidos por meio de curvas de erro médio, comparações entre valores desejados e obtidos e um parâmetro de desempenho. Por fim, na Seção 5.3 relatam-se as conclusões do estudo conduzido.

5.1 O problema de predição musical

O diagrama de blocos do problema encontra-se na Fig. 5.1.

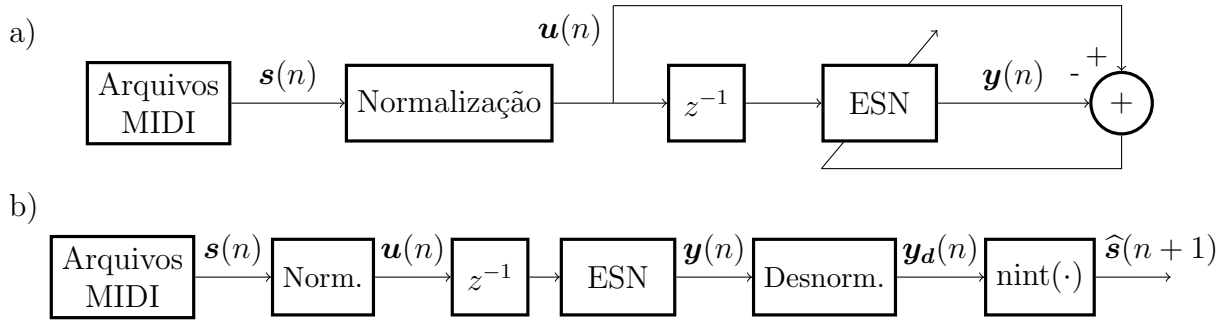

Figura 5.1: Diagrama de blocos do problema de predição de sinais musicais: a) período de treinamento da ESN. b) período de teste da ESN.

As abreviações Norm. e Desnorm. referem-se, respectivamente, à normalização e à desnormalização de $\boldsymbol{s}(n)$ e $\boldsymbol{y}(n)$, explicadas mais adiante nesta seção. A função nint : $\mathbb{R} \to \mathbb{Z}$, é a função de arredondamento *nearest integer*, de modo que se $\text{nint}(x) = m$, então m é o inteiro mais próximo de x. No caso em que o argumento é um vetor, a função nint$(\cdot)$ é aplicada elemento a elemento.

A partir de arquivos MIDI obtidos em [52], correspondentes a 48 trabalhos do pianista clássico Frédéric Chopin, utiliza-se o código em Python disponível no Anexo A para extrair informações relativas às notas tocadas: a altura e os instantes de início e de término de cada uma delas. Com isso, determina-se o sinal musical

$$\boldsymbol{s}(n) = \begin{bmatrix} s_1(n) & s_2(n) & s_3(n) \end{bmatrix}^T, \tag{5.1}$$

em que $s_1(n)$ é altura da nota tocada no instante n, $s_2(n)$ e $s_3(n)$ os instantes de início e término da mesma nota, respectivamente. Na Subseção 2.1.2, encontram-se maiores detalhes sobre $\boldsymbol{s}(n)$ e sobre como foi feito o processamento dos arquivos MIDI para extração dos atributos musicais de interesse.

O processamento dos 48 arquivos MIDI resultou em 85355 amostras de $\boldsymbol{s}(n)$. Destas, 80% - 68284 amostras - foi utilizado para treinamento da ESN, ilustrado na parte a) da Fig. 5.1. Como 200 amostras foram reservadas para lidar com a transiência inicial da rede, sobram 16871 amostras para teste da ESN, apresentado na parte b) da mesma figura.

Os títulos das obras musicais de Chopin utilizadas encontram-se no Anexo C, onde também detalham-se quais são os instantes de tempo n correspondentes a cada obra. Considerando que dentre as $A = 85355$ amostras disponíveis, $\ell = 200$ foram reservadas para lidar com o transiente da rede, $\boldsymbol{s}(n)$ está definido de $n = -\ell + 1 = 199$ até $n = A - \ell = 85155$, seguindo a convenção estabelecida na Seção 2.2. Na Fig. 5.2 encontram-se as amostras o sinal musical original $\boldsymbol{s}(n)$ obtidas a partir do processamento dos 48 arquivos MIDI.

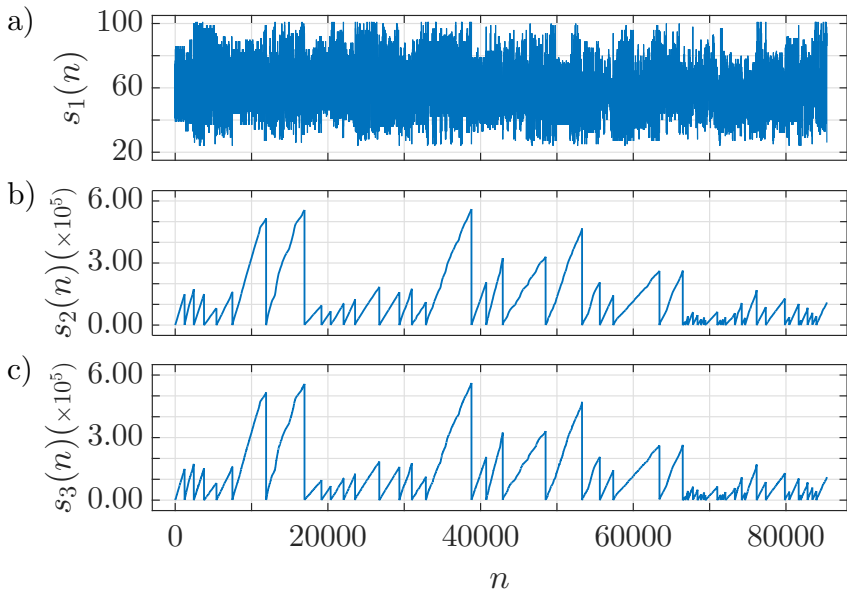

Figura 5.2: Sinal musical disponível para treinamento e teste da ESN.

A altura $s_1(n)$ é um número inteiro entre 0 e 127, conforme apresentado na Tabela 2.1. Já os instantes de início $s_2(n)$ e de término $s_3(n)$ são inteiros não negativos. Apesar de serem valores permitidos, não houve amostras de $s_1(n)$ menores que 20 ou maiores que 110, conforme pode ser observado no histograma de alturas apresentado na Fig. 5.3. De fato, estes valores representam alturas muito graves e muito agudas, nesta ordem, e são raramente usados em músicas.

A fim de esclarecer como a normalização de $\boldsymbol{s}(n)$ é feita, considere o vetor $\boldsymbol{s}_k$, $k =$

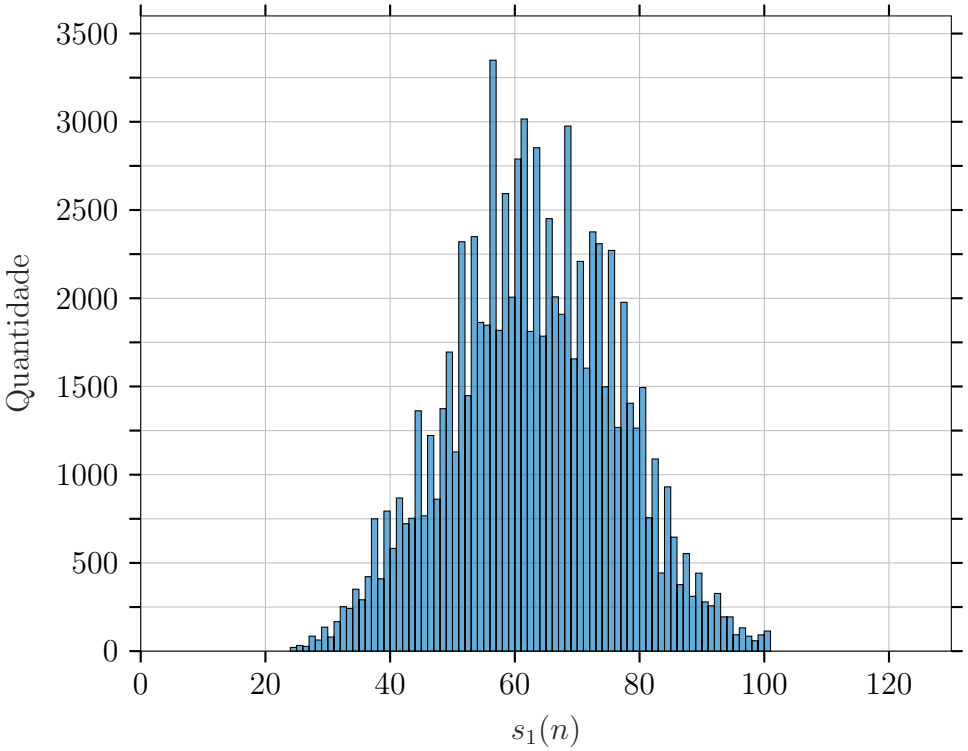

Figura 5.3: Histograma de amostras de altura $s_1(n)$.

$1, 2, 3$, contendo as amostras disponíveis para treinamento e teste da rede, isto é,

$$\boldsymbol{s}_k = \begin{bmatrix} s_k(-\ell+1) & s_k(-\ell+2) & \dots & s_k(A-\ell) \end{bmatrix}. \tag{5.2}$$

Seja também o vetor $\boldsymbol{u}_k$, $k = 1, 2, 3$, das amostras que efetivamente são a entrada da rede

$$\boldsymbol{u}_k = \begin{bmatrix} u_k(-\ell+1) & u_k(-\ell+2) & \dots & u_k(A-\ell) \end{bmatrix}. \tag{5.3}$$

O vetor $\boldsymbol{u}_k$ é obtido normalizando-se o vetor $\boldsymbol{s}_k$ para que tenha média nula e variância unitária. Sendo μ_k e σ_k, nesta ordem, a média e desvio padrão de $\boldsymbol{s}_k$, determina-se o vetor $\boldsymbol{u}_k$ por meio da expressão

$$\boldsymbol{u}_k = \frac{\boldsymbol{s}_k - \mu_k \mathbf{1}}{\sigma_k}, \tag{5.4}$$

em que $\mathbf{1} = [\,1\ 1\ \dots\ 1\,]$. Este tipo de normalização é um procedimento comum em técnicas de aprendizado de máquina [12, 19, 32]. Os valores de média e desvio padrão estão apresentados na Tabela 5.1, enquanto que o sinal normalizado $\boldsymbol{u}$ é apresentado na Fig. 5.4.

Para a desnormalização de $\boldsymbol{y}(n)$, faz-se

$$\boldsymbol{y_d}(n) = \begin{bmatrix} \sigma_1 y_1(n) \\ \sigma_2 y_2(n) \\ \sigma_3 y_3(n) \end{bmatrix} + \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} \sigma_1 y_1(n) + \mu_1 \\ \sigma_2 y_2(n) + \mu_2 \\ \sigma_3 y_3(n) + \mu_3 \end{bmatrix} \tag{5.5}$$

Tabela 5.1: Médias e desvios padrão de $\boldsymbol{s}_k$.

Vetor ($\boldsymbol{s}_k$)	Média (μ_k)	Desvio Padrão (σ_k)
$\boldsymbol{s}_1$	62.5987	13.4220
$\boldsymbol{s}_2$	1.3106	1.3008
$\boldsymbol{s}_3$	1.3132	1.3010

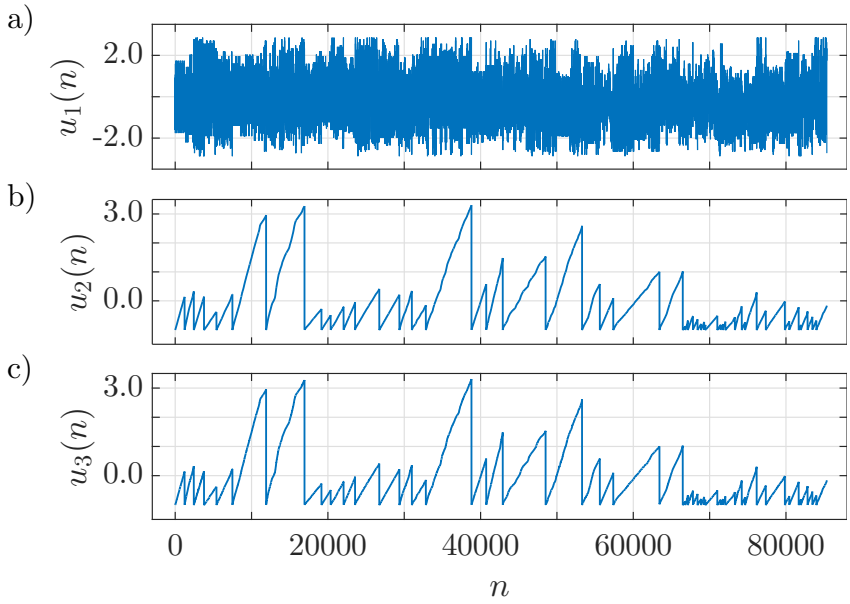

Figura 5.4: Sinal musical normalizado para treinamento e teste da ESN.

para cada instante n do período de teste da rede. Definindo-se $\sigma \triangleq [\,\sigma_1\ \sigma_2\ \sigma_3\,]^T$ e $\boldsymbol{\mu} \triangleq [\,\mu_1\ \mu_2\ \mu_3\,]^T$, é possível representar (5.5) de maneira elegante por

$$\boldsymbol{y_d}(n) = \boldsymbol{\sigma} \odot \boldsymbol{y}(n) + \boldsymbol{\mu}, \tag{5.6}$$

em que o operador $\odot$ denota o produto matricial elemento a elemento, também conhecido como produto de Hadamard [99]. Após a desnormalização, a estimativa final da rede é obtida arredondando-se a saída por meio de

$$\widehat{\boldsymbol{s}}(n+1) = \text{nint}(\boldsymbol{y_d}(n)). \tag{5.7}$$

Deste modo, uma ESN é treinada para, dado $\boldsymbol{s}(n)$ como entrada, estimar $\boldsymbol{s}(n+1)$. Em outras palavras, a ESN deve predizer qual seria a próxima nota. Isto significa estimar sua altura, instante de início e instante de término. Na Fig. 5.5 encontram-se as primeiras amostras de $\boldsymbol{s}(n)$ utilizadas para treinamento. Elas correspondem aos compassos - separados por linhas verticais traçejadas - 35 e 36 da obra Opus 7, No. 1.

Na próxima seção, detalha-se a forma com que foram feitas as simulações computaci-

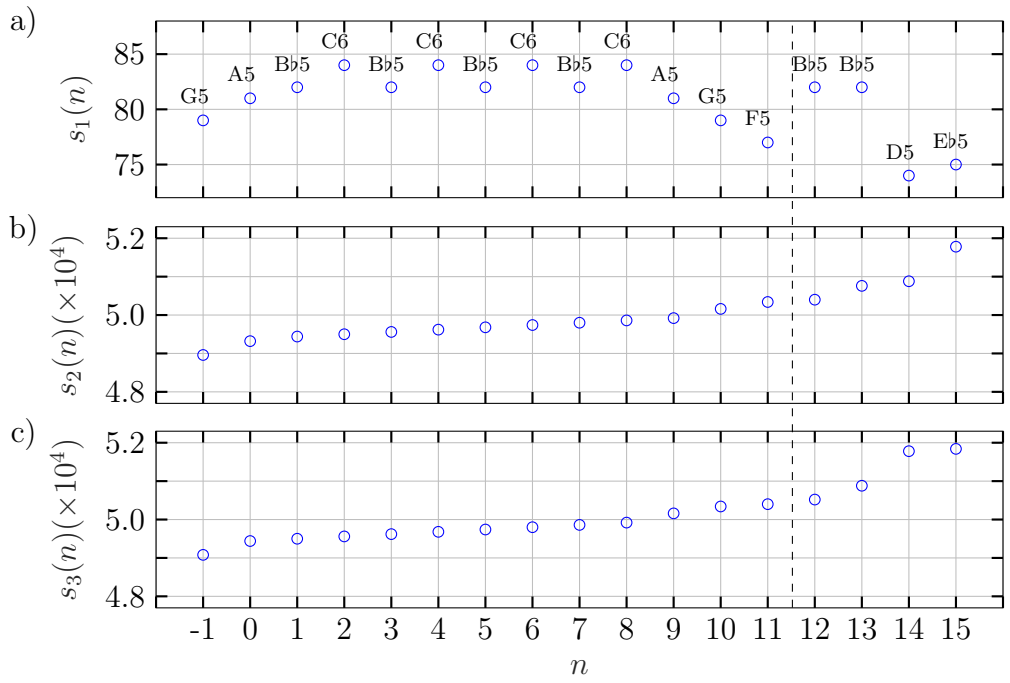

Figura 5.5: Primeiras amostras usadas para treinamento da ESN: a) altura; b) instante de início; c) instante de término.

onais e os resultados alcançados.

5.2 Resultados de simulações

A ESN foi gerada com 500 nós, $\ell = 200$ amostras e $L = 68284$ amostras, o que corresponde a 80% do banco de dados disponível.

Para escolher os parâmetros da ESN a, λ, p e q, ambas matrizes $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}$ foram mantidas fixas até o término da rotina de seleção de parâmetros, descrita nos próximos parágrafos. No presente problema, o sinal de entrada $\boldsymbol{u}(n)$ é sempre o mesmo em cada iteração da rotina, uma vez que o banco de dados é sempre o mesmo.

Como primeiro passo, foram tomados $\lambda = 0.05$, $p = 0$ e $q = 0.5$. Então, mantendo-os fixos, variou-se a de 0 até 1 em passos de 0.05. Em todos os casos, a ESN foi treinada e testada, calculando-se o valor da SNR na saída da rede de modo a determinar o valor $a = a_1^{\text{opt}}$ que a maximiza. Analogamente, obtém-se λ_1^{opt}, variando-se λ de 0.05 até 1 em passos de 0.05, e mantendo-se os demais parâmetros fixos em $a = a_1^{\text{opt}}$, $p = 0$ e $q = 0.5$. Em seguida, p_1^{opt} e q_1^{opt} foram obtidos, nesta ordem, variando-se p entre 0 e 10 em passos de 0.5 e q entre 0.5 e 10 em passos de 0.5, enquanto os demais parâmetros permanecem fixos nos valores selecionados mais recentes.

Essa rotina é repetida, obtendo-se a_i^{opt}, λ_i^{opt}, p_i^{opt}, q_i^{opt}, $i = 2, 3, \ldots$. A busca continua até que os valores selecionados parem de mudar. A evolução da seleção de parâmetros com o passar da rotina encontra-se na Fig. 5.6. Levou 12 iterações para que todos os parâmetros parassem de mudar. Os valores finais selecionados, que são utilizados nas próximas simulações desta seção, são $a_\star = 1.00$, $\lambda_\star = 0.75$, $p_\star = 0.00$ e $q_\star = 0.50$.

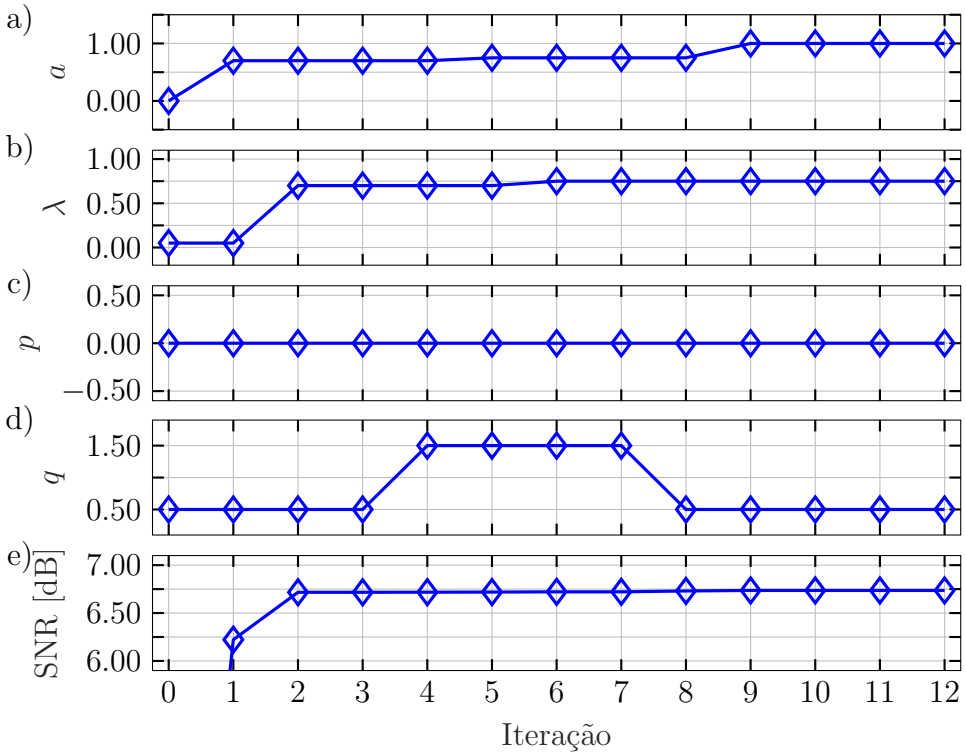

Figura 5.6: Evolução dos parâmetros da rede durante a rotina de seleção: a) parâmetro de *leakage*; b) raio espectral; c) e d) parâmetros das distribuições uniformes usadas para gerar $\boldsymbol{W}^{\text{in}}$; e) relação sinal-ruído.

Na Tabela 5.2 encontram-se todos os parâmetros da rede utilizados para as simulações deste capítulo.

Tabela 5.2: Parâmetros utilizados no problema de predição de sinais musicais.

Parâmetro	**Valor**	**Unidade**
N	500	nó
a	1.00	-
λ	0.75	-
p	0.00	-
q	0.50	-
ℓ	200	amostra
L	68284	amostra

Cabe destacar que, para o cálculo da SNR, considera-se o ruído na saída da rede

como sendo a diferença entre a saída da ESN $\boldsymbol{y}(n)$ e o sinal desejado normalizado $\boldsymbol{d}(n) = \boldsymbol{u}(n+1)$.

Com os parâmetros ótimos, após a obtenção de $\boldsymbol{W}^{\text{out}}$ mediante treinamento com 80% das amostras de $\boldsymbol{u}(n)$, testou-se a ESN com os 20% de amostras restantes. Isso foi repetido por dez vezes, e a cada realização, novas matrizes $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}$ foram geradas. O desempenho da ESN foi quantificado através do cálculo do erro $\boldsymbol{e}(n)$ na saída da rede, dado por

$$\boldsymbol{e}(n) = \begin{bmatrix} e_1(n) & e_2(n) & e_3(n) \end{bmatrix}^T = \boldsymbol{y}(n) - \boldsymbol{u}(n+1). \tag{5.8}$$

As médias das três componentes do erro obtidas são apresentadas na Fig. 5.7, na qual nota-se claramente que o erro é muito maior na componente $e_1(n)$ do que nas duas outras. Isto já era esperado, uma vez que as notas começam no início de um pulso, ou

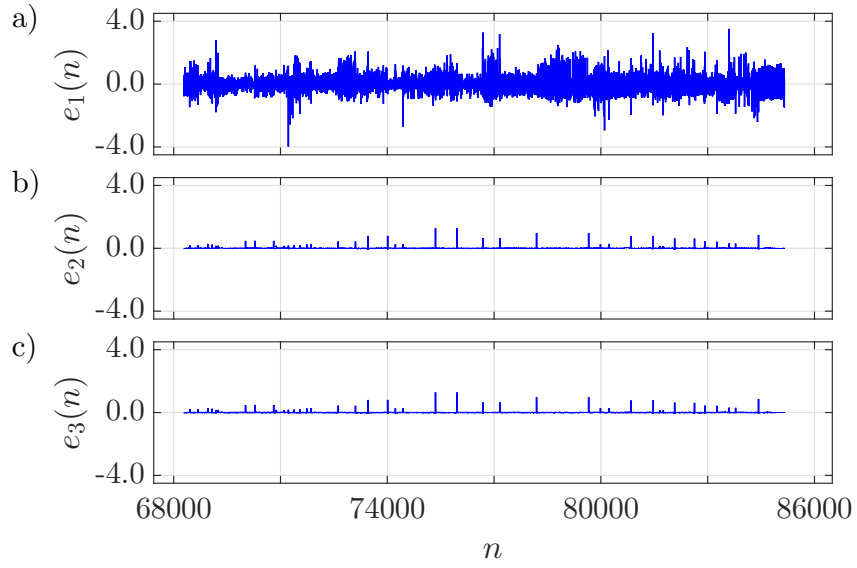

Figura 5.7: Curvas de erro de predição médio: a) altura; b) instante de início; c) instante de término.

após decorrida uma fração deste. Ainda, dada uma nota, a seguinte normalmente não começará muito depois da anterior. Muitas vezes, começa logo em seguida, sem pausa. Deste modo, há certos padrões e limitações sobre os possíveis valores de instante de início e de término, facilitando a predição destes.

Por inspeção, foi localizado em $e_1(n)$ um trecho em que a predição de altura é razoavelmente adequada. Este trecho, junto com os correspondentes valores desejados estão apresentados na Fig. 5.8. Tratam-se dos compassos 33, 34 e 35 da obra Prelude No. 16, separados por linhas verticais traçejadas.

Conforme observado na Fig. 5.8, considerando-se o primeiro compasso apresentado,

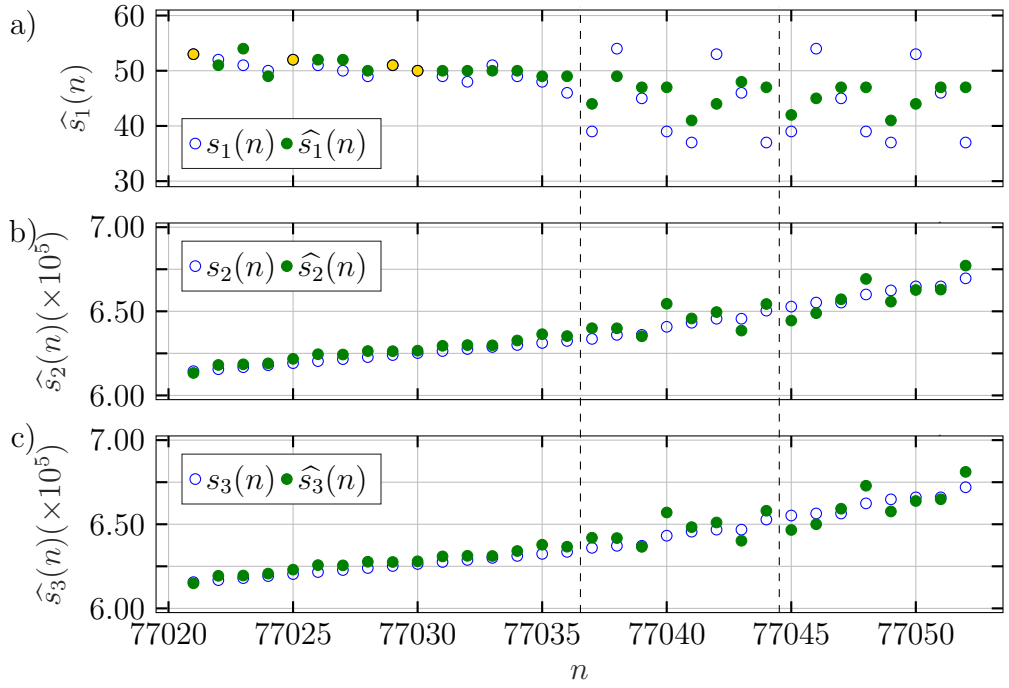

Figura 5.8: Sinais estimados durante o período de teste: a) altura; b) instante de início; c) instante de término. Marcadores dourados com contorno preto indicam uma estimativa bem-sucedida.

a ESN consegue estimar razoavelmente bem a próxima nota, em comparação aos outros compassos. Na Tabela 5.3, encontram-se os valores obtidos com a predição e os desejados referente ao primeiro compasso apresentado na Fig. 5.8. Estimativas de altura bem-sucedidas estão destacadas em negrito. Todas as notas encontram-se na oitava número 3 da Tabela 2.1, com exceção da nota desejada Si♭ no instante $n = 77036$, que se encontra na oitava número 2.

Percebe-se que existe um padrão musical no primeiro compasso, no qual há três grupos de quatro notas organizadas de modo cromaticamente descendente. No primeiro grupo, $\widehat{s}_1(n)$ assume os valores $\{53, 52, 51, 50\}$, no segundo grupo os valores $\{52, 51, 50, 49\}$, no terceiro os valores $\{51, 50, 49, 48\}$. Esse padrão auxilia a ESN em sua tarefa de predição. Nos próximos compassos em que não há um padrão tão característico, a predição é visivelmente afetada.

Numa tentativa de melhorar os resultados obtidos, fez-se uma reordenação das notas obtidas por meio do processamento dos arquivos MIDI. Basicamente, rearranjaram-se as amostras de $\boldsymbol{s}(n)$ em ordem crescente de instante de início em cada música, ignorando o agrupamento por *tracks* dos arquivos MIDI. Originalmente, o processamento considerava $\boldsymbol{s}(n)$ ordenado por instante de início em cada *track* de cada música. Maiores detalhes

Tabela 5.3: Valores de predição e desejados obtidos, referentes ao primeiro compasso apresentado na Fig. 5.8.

n	$s_1(n)$		$\hat{s}_1(n)$		$s_2(n)$	$\hat{s}_2(n)$	$s_3(n)$	$\hat{s}_3(n)$
77021	**53**	**Fá**	**53**	**Fá**	61440	61335	61560	61489
77022	52	Mi	51	Mi♭	61560	61812	61680	61937
77023	51	Mi♭	54	Sol♭	61680	61849	61800	61953
77024	50	Ré	49	Ré♭	61800	61905	61920	62066
77025	**52**	**Mi**	**52**	**Mi**	61920	62175	62040	62302
77026	51	Mi♭	52	Mi	62040	62450	62160	62572
77027	50	Ré	52	Mi	62160	62431	62280	62557
77028	49	Ré♭	50	Ré	62280	62637	62400	62772
77029	**51**	**Mi♭**	**51**	**Mi♭**	62400	62630	62520	62757
77030	**50**	**Ré**	**50**	**Ré**	62520	62661	62640	62798
77031	49	Ré♭	50	Ré	62640	62950	62760	63090
77032	48	Dó	50	Ré	62760	62991	62880	63125
77033	51	Mi♭	50	Ré	62880	62976	63000	63114
77034	49	Ré♭	50	Ré	63000	63265	63120	63410
77035	48	Dó	49	Ré♭	63120	63640	63240	63784
77036	46	Si♭	49	Ré♭	63240	63527	63360	63666

sobre o processamento dos arquivos MIDI encontram-se na página 19 da Seção 2.1.

Utilizando-se os mesmos parâmetros anteriores, disponíveis na Tabela 5.2, a ESN foi testada por 10 vezes com a nova ordenação das amostras de $\boldsymbol{s}(n)$, com $\boldsymbol{W}^{\text{in}}$ e $\boldsymbol{W}$ distintos em cada realização. Na Fig. 5.9, encontram-se as curvas de erros médios obtidos.

Comparando-a com as curvas apresentadas na Fig. 5.7, nota-se que o desempenho degradou consideravelmente. Para confirmar isso, calculou-se além da SNR média entre ambos casos, mais dois parâmetros. O primeiro é a taxa de acerto de altura, denotada por β e calculada como sendo o número total de acertos sobre o número de amostras de $s_1(n)$ usadas para teste, equivalente a 16871, descontada uma amostra. Esse descarte ocorre porque a amostra $s_1(L+1)$, do primeiro instante do período de teste, não é estimada em nenhum momento pela rede. Ela é usada apenas para alimentar a rede, para que esta estime a amostra seguinte $s_1(L+2)$, de modo que β pode ser escrito como

$$\beta = \frac{1}{(A-L-\ell-1)} \sum_{n=L+2}^{A-\ell} \delta\left(\widehat{s}_1(n) - s_1(n)\right), \tag{5.9}$$

em que $\delta(\cdot)$ é a função impulso unitário de tempo discreto [100].

O segundo parâmetro, denotado por ρ, determina a distância média entre a altura

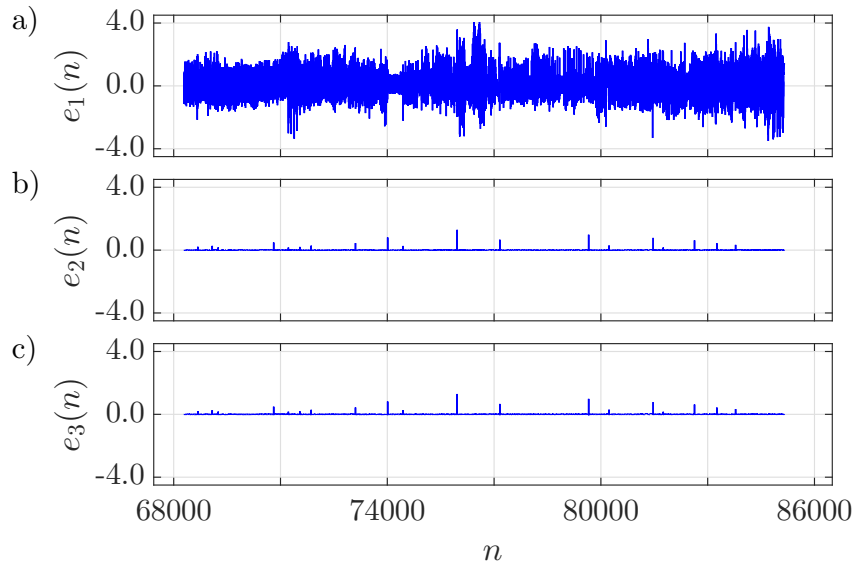

Figura 5.9: Curvas de erro de predição médio obtidas reordenando-se $\boldsymbol{s}(n)$: a) altura; b) instante de início; c) instante de término.

estimada e a altura desejada. Em música, a unidade de medida da distância entre duas notas é o semitom e usa-se o termo intervalo ao invés de distância [49]. Fazendo-se referência à Tabela 2.1, o intervalo entre as notas D5 e G5 é de $43 - 38 = 5$ semitons. Já entre as notas D5 e G♯5, o intervalo é de $44 - 38 = 6$ semitons, ou 3 tons.

Sendo assim, e também levando em consideração que só existem intervalos inteiros, o parâmetro ρ é calculado por meio de

$$\rho = \text{nint}\left(\frac{1}{(A - L - \ell - 1)} \sum_{n=L+2}^{A-\ell} |\widehat{s}_1(n) - s_1(n)|\right). \tag{5.10}$$

Todos os parâmetros descritos nos parágrafos anteriores estão apresentados na Tabela 5.4. A separação em dois casos que é feita nela refere-se à consideração, ou não, dos *tracks* de cada arquivo MIDI:

- **Caso 1 - leva-se a organização por *tracks* em consideração**: amostras de $\boldsymbol{s}(n)$ em ordem crescente de $s_2(n)$ por *track* de cada música; e
- **Caso 2 - ignora-se a organização por *tracks***: amostras de $\boldsymbol{s}(n)$ em ordem crescente de $s_2(n)$ por música.

Nota-se uma clara vantagem do Caso 1 em relação ao desempenho, uma vez que a diferença entre as SNR obtidas é de quase 5 dB. Os parâmetros β indicam que, a cada 1000 amostras de $s_1(n)$, tem-se, em média, 75 acertos no Caso 1 e 40 acertos no Caso 2.

Tabela 5.4: Parâmetros para comparação entre os desempenhos da ESN em ambos os casos.

Realização	SNR [dB]		β ($\times 10^{-2}$)		ρ [semitons]	
	Caso 1	**Caso 2**	**Caso 1**	**Caso 2**	**Caso 1**	**Caso 2**
1	6.74	1.86	7.73	4.11	5	8
2	6.80	1.98	7.45	4.18	5	8
3	6.71	1.94	7.64	4.15	5	8
4	6.78	1.99	7.90	4.06	5	8
5	6.71	1.94	6.90	3.79	5	8
6	6.74	2.02	7.46	4.08	5	8
7	6.73	1.96	6.97	3.98	5	8
8	6.75	1.92	7.45	4.13	5	8
9	6.65	2.06	8.25	3.91	5	8
10	6.82	1.93	7.25	3.82	5	9
Média	**6.74**	**1.96**	**7.50**	**4.02**	**5**	**8**

Por fim, os parâmetros ρ revelam que a estimativa $\widehat{s}_1(n)$ da rede estará, em média, a um intervalo 5 semitons da amostra desejada $s_1(n)$ no Caso 1. No Caso 2, esse valor sobe para 8 semitons, mostrando claramente que obtém-se valores mais próximos dos almejados quando leva-se a organização dos *tracks* em consideração.

Sendo a totalidade das músicas utilizadas tocadas em piano, os arquivos MIDI correspondentes apresentam normalmente um *track* para o que é tocado pela mão esquerda e outro para o que é tocado pela mão direita. Em muitas músicas, normalmente a mão esquerda é responsável por tocar acordes, conduzindo a parte harmônica, enquanto a mão direita é encarregada de tocar melodias. É razoável dizer que há uma independência entre o que é tocado por cada uma das mãos, de modo que no Caso 1 a ESN é beneficiada.

5.3 Conclusões

Neste capítulo, investigou-se o desempenho de uma ESN para predição de um sinal musical. Este teve como origem um banco de dados contendo 48 arquivos MIDI de peças musicais, todas tocadas em piano, do artista Frédéric Chopin.

Num primeiro momento - Caso 1 -, o processamento dos arquivos considerou a separação por *tracks* em cada música, gerando primeiramente todas as amostras do que é tocado por uma das mãos e, em seguida, todas as amostras do que é trocado pela outra mão. Numa segunda situação - Caso 2 -, o processamento dos arquivos não levou em consideração a separação por *tracks* de cada música, gerando amostras na sequência em

que são tocadas em cada uma das músicas.

Após ajustar os parâmetros da rede mediante uma rotina de seleção, a rede foi testada por dez vezes em ambos os casos, cada realização com diferentes matrizes de entrada e interna. Obtiveram-se curvas de erro médio que indicaram que, a maior parcela de erros da rede, em ambos os casos, ocorre na estimativa da altura das notas tocadas. O que faz sentido, pois há uma maior restrição de valores possíveis para as demais estimativas de instante de início e término de cada nota.

Tanto o erro médio quanto os demais parâmetros calculados - a SNR, taxa de acertos de altura β e intervalo entre estimativa de altura e altura desejada ρ - mostraram que a ESN opera com melhor desempenho no Caso 1. A independência entre o que é tocado por cada uma das mãos parece ser uma boa razão para explicar a diferença de desempenho observada: a ESN entende melhor o que está acontecendo quando é alimentada com amostras referentes a cada mão separadamente, e assim consegue fornecer melhores aproximações para a amostra seguinte.

Capítulo 6

Conclusões e trabalhos futuros

Principalmente devido à sua simplicidade, ESNs vêm sendo estudadas, desde sua origem em [10], para a solução de diversos problemas de processamento de sinais como mostrado em [16, 32, 78, 79, 94, 95].

Sendo as ESNs parte fundamental do *reservoir computing*, neste trabalho mostrou-se que o mesmo pode ser utilizado como ferramenta para resolução de problemas de processamento de sinais, por meio do estudo de três aplicações.

Após uma breve introdução motivacional no Capítulo 1, foram revistos no Capítulo 2 os principais conceitos sobre sinais caóticos, sinais musicais e ESNs utilizados nas aplicações investigadas neste trabalho. Para cada uma delas, dedicou-se um capítulo deste texto conforme lista a seguir:

- Capítulo 3 - Redução de ruído em sinais caóticos;
- Capítulo 4 - Sistema de comunicação baseado em caos utilizando ESN; e
- Capítulo 5 - Predição de sinais musicais.

Em [78] e [79] aborda-se o problema de mitigação de ruído por meio do uso de ESNs, com resultados positivos em ambos trabalhos. Utilizam-se combinações lineares de sinais senoidais com envelopes variantes no tempo no primeiro trabalho e sinais de eletroencefalogramas no segundo para testar os sistemas propostos.

No Capítulo 3 desta dissertação, foi estudado o uso de ESNs para o problema de redução de ruído, empregando-se sinais caóticos de tempo discreto gerados pelo mapa tenda inclinada. A dependência sensível em relação as condições iniciais, a presença de transições abruptas e a aperiodicidade destes, impõe novos desafios à rede.

Foi mostrado que o desempenho da ESN projetada consegue superar o desempenho do filtro de Wiener - técnica de filtragem linear ótima - para todos os valores do parâmetro do mapa tenda inclinada considerados.

Em [94] e [95], ESNs juntamente com filtros de Kalman são empregadas para realizar a equalização não supervisionada de canais não lineares usando-se sinais caóticos, no primeiro trabalho, e sinais com modulação de amplitude em quadratura, no segundo trabalho. Avaliou-se o desempenho das técnicas empregadas em ambos trabalhos calculando-se o erro quadrático médio.

No Capítulo 4 deste trabalho, investigou-se o uso de ESNs para equalizar o canal do sistema de comunicação baseado em caos proposto em [27] em três cenários distintos. No Cenário I, o canal adiciona apenas AWGN; no Cenário II, considera-se somente um canal $H(z) = h_0 + z^{-1} + h_0 z^{-2}$ sem ruído e, por fim, considera-se o efeito simultâneo dos cenários anteriores.

Apesar de, diferentemente de [94, 95], em que os canais considerados são não lineares e o desempenho é avaliado por meio do erro quadrático médio, aqui o desempenho das ESNs em todos os cenários é avaliado por meio da BER, que é uma medida mais objetiva e significativa e que demonstra claramente quão bem o esquema proposto funciona na prática.

Os resultados conquistados demonstraram que as ESNs projetadas conseguem atingir valores de BER satisfatórios, dependendo do nível de ruído e da magnitude de h_0. Em todas as situações analisadas, os desempenhos apresentados pelas ESNs superaram os obtidos por meio do uso de filtros de Wiener seguidos da técnica de sincronismo caótico [27].

Em [32], examina-se o problema de predição de sinais musicais, que é essencialmente o mesmo problema estudado aqui. Constrói-se um sinal musical a partir de arquivos MIDI e treina-se uma ESN com uma parte das amostras desse sinal para predizer a amostra seguinte, dada uma amostra qualquer.

No Capítulo 5 deste trabalho, calcula-se a taxa de acertos média da rede projetada, bem como a média do intervalo musical entre as alturas da nota estimada e da nota desejada. Os melhores resultados obtidos foram de uma taxa de acertos de $7.50 \times^1 0{-}2$ e um intervalo médio de 5 semitons. Em [32] não é calculada nenhuma medida de desempenho com algum viés musical.

Ademais, sendo todas as obras musicais utilizadas tocadas em piano, também obteve-se a interessante conclusão de que a rede opera melhor quando as amostras de entrada estão agrupadas na sequência em que são tocadas por cada uma das mãos separadamente. Se as amostras estiverem ordenadas simplesmente na sequência em que são tocadas, o desempenho da rede degrada substancialmente.

6.1 Contribuições

Mediante o estudo e simulações realizadas ao longo deste trabalho de mestrado, foi possível atingir os objetivos propostos no início do trabalho. As principais contribuições desta dissertação estão sintetizadas na Tabela 6.1.

Tabela 6.1: Principais contribuições da dissertação.

Capítulo	Contribuições
3	**A.** Determinação da influência do parâmetro do mapa tenda inclinada, quando da mitigação de AWGN sobre um sinal caótico gerado pelo mesmo mapa, por meio do emprego de uma ESN.
4	**B.** Integração de uma ESN para equalização de canal do SCBC proposto em [27] em diferentes cenários.
5	**C.** Verificação dos efeitos do agrupamento de notas pela mão correspondente que as toca no piano, quando do uso de uma ESN para predição de sinais musicais.

Abaixo encontram-se alguns comentários pertinentes sobre cada uma das contribuições listadas na tabela acima.

A. Determinação da influência do parâmetro do mapa tenda inclinada, quando da mitigação de AWGN sobre um sinal caótico gerado pelo mesmo mapa, por meio do emprego de uma ESN.

O principal resultado obtido no Capítulo 3 é exatamente uma curva de ganho de processamento em função do parâmetro do mapa tenda inclinada. Foi possível concluir que quanto maior a magnitude do parâmetro, maior é o ganho de processamento, numa relação não linear.

Uma vez que as características espectrais do mapa dependem diretamente deste parâmetro, foi possível também inferir a relação entre o comportamento da DEP e o ganho de processamento obtido. Os resultados de simulação obtidos corroboraram plenamente com a expressão teórica para a DEP.

B. Integração de uma ESN para equalização de canal do SCBC proposto em

[27] em diferentes cenários.

No que concerne o conhecimento do autor, o uso de ESNs no SCBC proposto em [27] é algo inédito. Em [30], a equalização deste SCBC com o mesmo canal empregado aqui é estudada, todavia, recorre-se a técnicas de filtragem adaptativa para mitigar os efeitos do canal. Também cabe destacar que a ESN empregada aqui toma a decisão sobre o bit transmitido, além de equalizar o canal.

Além disso, os resultados obtidos foram bastante positivos, no sentido de que foi possível encontrar situações em que o uso da ESN torna o SCBC estudado viável, uma vez que a BER alcançou patamares satisfatórios.

C. Verificação dos efeitos do agrupamento de notas pela mão correspondente que as toca no piano, quando do uso de uma ESN para predição de sinais musicais.

Uma vez que o sinal musical empregado vêm do processamento de músicas tocadas em piano, apresentando *tracks* distintos para o que é tocado por cada uma das mãos do pianista, foi possível avaliar se esse agrupamento tem influência sobre o desempenho da ESN.

Dada uma das músicas, alimentando-se a rede primeiramente com todas as notas na sequência em que são tocadas pela mão direita, para só depois alimentar a rede com as notas na sequência em que são tocadas pela mão esquerda, levou a um resultado melhor que ignorar a separação por mãos feita pelos *tracks* da música.

Esta conclusão vai ao encontro com o que já é de conhecimento dos pianistas profissionais: o que se toca com a mão esquerda é independente daquilo que se toca com mão direita.

A Contribuição A. teve como resultado três artigos, dos quais um foi enviado ao periódico *Signal Processing*[1], com fator de impacto 4.729. Os remanescentes foram artigos aceitos em congressos nacionais. O artigo submetido ao periódico encontra-se em revisão, e sua versão completa está disponível em [41] e também no Anexo E.

Os artigos que foram apresentados e publicados nos anais de congressos nacionais, estão listados a seguir:

- **A.L. Duarte**, M. Eisencraft *"Aplicação de Reservoir Computing para Filtragem de Sinais Caóticos Imersos em Ruído Branco Gaussiano,"* XL Congresso nacional de

[1]https://www.sciencedirect.com/journal/signal-processing. Acesso em: 4 de maio de 2023.

matemática aplicada e computacional - CNMAC 2021, São Carlos - SP, Brasil, Set. 2021. Disponível em[2].

- **A.L. Duarte**, M. Eisencraft, *"Aplicação de Reservoir Computing para Filtragem de Sinais Caóticos Imersos em Ruído,"* XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT'21, Fortaleza - CE, Brasil, Set. 2021. doi: 10.14209/sbrt.2021.1570724246

Ambos estão disponíveis, respectivamente, nos Anexos F e G deste texto.

A Contribuição B. levou a um artigo submetido ao *XLII Congresso Nacional de Matemática Aplicada e Computacional*[3], a ser realizada de 18 a 22 de setembro de 2023 em Bonito - MS.

6.2 Trabalhos futuros

A pesquisa realizada durante a elaboração desta dissertação, levou às seguintes possibilidades de trabalhos futuros:

- Determinação de uma expressão analítica para o ganho de processamento da ESN quando usada para redução de ruído em sinais caóticos gerados pelo mapa tenda inclinada;
- Comparação do desempenho da ESN para *denoising* com outras técnicas não lineares;
- Obtenção de curvas de BER variando-se o coeficiente γ da codificação soma;
- Comparação do desempenho da ESN para equalização de canal com outras técnicas não lineares;
- Reavaliação do desempenho da ESN ao se fazer uma discretização da duração das notas musicais; e
- Comparação do desempenho da ESN para predição de sinais musicais com outros tipos de redes neurais.

[2]https://proceedings.sbmac.org.br/sbmac/article/view/134324. Acesso em: 4 de maio de 2023.
[3]http://www.cnmac.org.br. Acesso em: 4 de maio de 2023.

Referências

[1] L. V. Fausett, *Fundamentals of neural networks: Architectures, algorithms, and applications*. Prentice-Hall, 1994.

[2] W. S. McCulloch e W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, págs. 115–133, dez. 1943.

[3] D. O. Hebb, *The organization of behavior: a neuropsychological theory*. John WIley & Sons, Inc., 1949.

[4] N. Rochester, J. H. Holland, L. B. M. Haibt, e W. L. Duda, "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *IEEE Transactions on Information Theory*, vol. 2, págs. 80–93, set. 1956.

[5] I. Goodfellow, Y. Bengio, e A. Courville, *Deep Learning*. The MIT Press, abr. 2017.

[6] M. Chen, U. Challita, W. Saad, C. Yin, e M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, nº 4, págs. 3039–3071, 2019.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, e P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, págs. 318–327, fev. 2020.

[8] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, e H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, págs. 3848–3858, set. 2020.

[9] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, M. B. I. Reaz, e M. T. Islam, "Can AI help in screening viral and COVID-19 pneumonia?," *IEEE Access*, vol. 8, págs. 132665–132676, 2020.

[10] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," Tech. Rep. 148, German National Research Center for Information Technology, 2001.

[11] M. Lukoševičius, D. Popovici, H. Jaeger, e U. Siewert, "Time warping invariant echo state networks," tech. rep., International University Bremen, 2006.

[12] M. Lukoševičius, "A practical guide to applying echo state networks," in *Lecture Notes in Computer Science*, págs. 659–686, Springer Berlin Heidelberg, 2012.

[13] M. Lukoševičius e H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, págs. 127–149, ago. 2009.

[14] J. Long, S. Zhang, e C. Li, "Evolving deep echo state networks for intelligent fault diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 16, págs. 4928–4937, jul. 2020.

[15] L. Li, Y. Xu, Z. Zhang, J. Yin, W. Chen, e Z. Han, "A prediction-based charging policy and interference mitigation approach in the wireless powered internet of things," *IEEE Journal on Selected Areas in Communications*, vol. 37, págs. 439–451, fev. 2019.

[16] P. Steiner, S. Stone, P. Birkholz, e A. Jalalvand, "Multipitch tracking in music signals using echo state networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*, IEEE, jan. 2021.

[17] P. Steiner, A. Jalalvand, S. Stone, e P. Birkholz, "Feature engineering and stacked echo state networks for musical onset detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, jan. 2021.

[18] F. Wang, Y. Wang, W. Li, B. Zhu, J. Ding, K. Wang, C. Liu, C. Wang, M. Kong, L. Zhao, W. Zhou, F. Zhao, e J. Yu, "Echo state network based nonlinear equalization for 4.6 km 135 GHz d-band wireless transmission," *Journal of Lightwave Technology*, vol. 41, págs. 1278–1285, mar. 2023.

[19] S. Krishnagopal, M. Girvan, E. Ott, e B. R. Hunt, "Separation of chaotic signals by reservoir computing," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, pág. 023123, fev. 2020.

[20] K. T. Alligood e T. Sauer., *Chaos: An Introduction to Dynamical Systems*. Springer, 1997.

[21] S. H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering*. Westview Press, 2ª ed., 2014.

[22] A. D. Linde, "Eternally existing self-reproducing chaotic inflanationary universe," *Physics Letters B*, vol. 175, págs. 395–400, ago. 1986.

[23] E. Harth, "Order and chaos in neural systems: An approach to the dynamics of higher brain functions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, págs. 782–789, set. 1983.

[24] G. Cai, Y. Fang, J. Wen, S. Mumtaz, Y. Song, e V. Frascolla, "Multi-carrier *m*-ary DCSK system with code index modulation: An efficient solution for chaotic communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, págs. 1375–1386, out. 2019.

[25] B. P. Lathi e Z. Ding, *Modern Digital and Analog Communication Systems*. Oxford University Press, 4ª ed., 2009.

[26] S. S. Haykin, *Adaptive filter theory*. Prentice Hall, 2002.

[27] M. Eisencraft, R. D. Fanganiello, e L. A. Baccala, "Synchronization of discrete-time chaotic systems in bandlimited channels," *Mathematical Problems in Engineering*, vol. 2009, págs. 1–12, 2009.

[28] C. W. Wu e L. O. Chua, "A simple way to synchronize chaotic systems with applications to secure communiction systems," *International Journal of Bifurcation and Chaos*, vol. 03, págs. 1619–1627, dez. 1993.

[29] M. Eisencraft, D. Kato, e L. Monteiro, "Spectral properties of chaotic signals generated by the skew tent map," *Signal Processing*, vol. 90, págs. 385–390, jan. 2010.

[30] R. Candido, *A questão da equalização em sistemas de comunicação que utilizam sinais caóticos*. PhD thesis, Escola Politécnica da Universidade de São Paulo, 2014.

[31] G. Abib e M. Eisencraft, "Sobre o desempenho em canal com ruído de um sistema de comunicação baseado em caos," in *Anais de XXXI Simpósio Brasileiro de Telecomunicações*, Sociedade Brasileira de Telecomunicações, 2013.

[32] A. Krušna e M. Lukoševičius, "Predicting mozart's next note via echo state networks," in *CEUR workshop proceedings: System 2018: Symposium for young scientists in technology, engineering and mathematics: proceedings of the symposium for young scientists in technology, engineering and mathematics.*, 2018.

[33] A. Ycart e E. Benetos, "Learning and evaluation methodologies for polyphonic music sequence prediction with LSTMs," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, págs. 1328–1341, 2020.

[34] D. Martin-Gutierrez, G. H. Penaloza, A. Belmonte-Hernandez, e F. A. Garcia, "A multimodal end-to-end deep learning architecture for music popularity prediction," *IEEE Access*, vol. 8, págs. 39361–39374, 2020.

[35] K. W. Cheuk, Y.-J. Luo, B. T. Balamurali, G. Roig, e D. Herremans, "Regression-based music emotion prediction using triplet neural networks," in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, jul. 2020.

[36] H. Jaeger, "Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach," Tech. Rep. 159, German National Research Center for Information Technology, 2002.

[37] The Mathworks, Inc., Natick, Massachusetts, *MATLAB versão 8.5.0.197613 (R2015a)*, 2015.

[38] G. van Rossum e F. L. J. Drake, *The Python Language Reference Manual: Revised and updated for Python 3.2*. Network Theory LTD, 2011.

[39] O. M. Bjørndalen, *Mido - MIDI Objects for Python*, 1.2.10 ed.

[40] R. C. U.S. e M. M. Association, "An introduction to midi," 2009.

[41] A. L. O. Duarte e M. Eisencraft, "Denoising of discrete-time chaotic signals using echo state networks," *arXiv:2304.06516 [eess.SP]*, abr. 2023.

[42] S. Lakshmanan, M. Prakash, C. P. Lim, R. Rakkiyappan, P. Balasubramaniam, e S. Nahavandi, "Synchronization of an inertial neural network with time-varying delays and its application to secure communication," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, págs. 195–207, jan. 2018.

[43] M. Herceg, G. Kaddoum, D. Vranjes, e E. Soujeri, "Permutation index DCSK modulation technique for secure multiuser high-data-rate communication systems," *IEEE Transactions on Vehicular Technology*, vol. 67, págs. 2997–3011, abr. 2018.

[44] C. Zhu e K. Sun, "Cryptanalyzing and improving a novel color image encryption algorithm using RT-enhanced chaotic tent maps," *IEEE Access*, vol. 6, págs. 18759–18770, 2018.

[45] R. Basu, D. Dutta, S. Banerjee, V. Holmes, e P. Mather, "An algorithmic approach for signal measurement using symbolic dynamics of tent map," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, págs. 2221–2231, jul. 2018.

[46] R. A. da Costa, M. B. Loiola, e M. Eisencraft, "Spectral properties of chaotic signals generated by the bernoulli map," *Journal of Engineering Science and Technology Review*, vol. 8, págs. 12–16, abr. 2015.

[47] R. P. Agarwal, *Difference Equations and Inequalities: Theory, Methods, and Applications*. CRC Press, 2000.

[48] M. Eisencraft e A. M. Batista, "Discrete-time chaotic systems synchronization performance under additive noise," *Signal Processing*, vol. 91, págs. 2127–2131, ago. 2011.

[49] A. Chediak, *Harmonia e Improvisação - Volume I*. Irmãos Vitale, 2020.

[50] M. Schwenk, "Midieditor - graphical interface to edit, play, and record midi data," 2022. Disponível em http://www.midieditor.org/, acessado em 4 de maio de 2023.

[51] C. Walshaw, "abc notation - the text-based music notation system and the de facto standard for folk and traditional music." Disponível em https://abcnotation.com/, acessado em 4 de maio de 2023.

[52] B. Krueger, "Classical piano midi page." Disponível em http://www.piano-midi.de, acessado em 4 de maio de 2023.

[53] C. C. Aggarwal, *Linear Algebra and Optimization for Machine Learning: A Textbook*. Springer, 2020.

[54] K. Doya, "Bifurcations in the learning of recurrent neural networks," in *Proceedings of 1992 IEEE International Symposium on Circuits and Systems*, IEEE, 1992.

[55] M. Lukoševičius, H. Jaeger, e B. Schrauwen, "Reservoir computing trends," *KI - Künstliche Intelligenz*, vol. 26, págs. 365–371, mai. 2012.

[56] Y. Bengio, P. Simard, e P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, págs. 157–166, mar. 1994.

[57] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, e B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, págs. 82–97, nov. 2012.

[58] A. Petrėnas, V. Marozas, L. Sörnmo, e A. Lukoševičius, "Reservoir computing for extraction of low amplitude atrial activity in atrial fibrillation," in *2012 Computing in Cardiology*, págs. 13–16, 2012.

[59] E. Maggiori, Y. Tarabalka, G. Charpiat, e P. Alliez, "Convolutional neural networks for large-scale remote-sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, págs. 645–657, fev. 2017.

[60] W. Maass, T. Natschläger, e H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, págs. 2531–2560, nov. 2002.

[61] M. Lellep, J. Prexl, M. Linkmann, e B. Eckhardt, "Using machine learning to predict extreme events in the hénon map," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, pág. 013113, jan. 2020.

[62] J. Wu, C. Hu, Y. Wang, X. Hu, e J. Zhu, "A hierarchical recurrent neural network for symbolic melody generation," *IEEE Transactions on Cybernetics*, vol. 50, págs. 2749–2757, jun. 2020.

[63] R. Yang, L. Feng, H. Wang, J. Yao, e S. Luo, "Parallel recurrent convolutional neural networks-based music genre classification method for mobile devices," *IEEE Access*, vol. 8, págs. 19629–19637, 2020.

[64] S. P. Chatzis e Y. Demiris, "Echo state gaussian process," *IEEE Transactions on Neural Networks*, vol. 22, págs. 1435–1445, set. 2011.

[65] S. P. Chatzis e Y. Demiris, "The copula echo state network," *Pattern Recognition*, vol. 45, págs. 570–577, jan. 2012.

[66] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* Numerical Mathematics and Scientific Computation, Oxford University Press, 1988.

[67] C. M. Bishop, *Pattern Recognition and Machine Learning.* Information Science and Statistics, Springer, 2006.

[68] G. G. Garbi, *A Rainha das Ciências: um passeio histórico pelo maravilhoso mundo da Matemática.* Livraria da Física, 2011.

[69] A. Papoulis e S. U. Pillai, *Probability, Random Variables and Stochastic Processes.* McGraw-Hill Europe, 4ª ed., 2002.

[70] A. A. Nugraha, A. Liutkus, e E. Vincent, "Multichannel audio source separation with deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, págs. 1652–1664, set. 2016.

[71] S. Gannot, E. Vincent, S. Markovich-Golan, e A. Ozerov, "A consolidated perspective on multimicrophone speech enhancement and source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, págs. 692–730, abr. 2017.

[72] K. Dabov, A. Foi, V. Katkovnik, e K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, págs. 2080–2095, ago. 2007.

[73] S. V. Vaseghi, *Advanced Digital Signal Processing and Noise Reduction*. Wiley, 4ª ed., 2009.

[74] H. Li, Z. Hua, H. Bao, L. Zhu, M. Chen, e B. Bao, "Two-dimensional memristive hyperchaotic maps and application in secure communication," *IEEE Transactions on Industrial Electronics*, vol. 68, págs. 9931–9940, out. 2021.

[75] Y. Cao, J. Huang, e C. Xiong, "Single-layer learning-based predictive control with echo state network for pneumatic-muscle-actuators-driven exoskeleton," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, págs. 80–90, mar. 2021.

[76] M. Eisencraft, R. R. F. Attux, e R. Suyama, *Chaotic Signals in Digital Communications (Electrical Engineering & Applied Signal Processing Series)*. Taylor & Francis Inc, 2013.

[77] M. S. Baptista, "Chaos for communication," *Nonlinear Dynamics*, vol. 105, págs. 1821–1841, jul. 2021.

[78] L. de Oliveira Junior, F. Stelzer, e L. Zhao, "Clustered and deep echo state networks for signal noise reduction," *Machine Learning*, vol. 111, págs. 2885–2904, mar. 2022.

[79] W. Sun, Y. Su, X. Wu, X. Wu, e Y. Zhang, "EEG denoising through a wide and deep echo state network optimized by UPSO algorithm," *Applied Soft Computing*, vol. 105, pág. 107149, jul. 2021.

[80] M. Han, Y. Liu, J. Xi, e W. Guo, "Noise smoothing for nonlinear time series using wavelet soft threshold," *IEEE Signal Processing Letters*, vol. 14, págs. 62–65, jan. 2007.

[81] S. Lou, J. Deng, e S. Lyu, "Chaotic signal denoising based on simplified convolutional denoising auto-encoder," *Chaos, Solitons and Fractals*, vol. 161, pág. 112333, ago. 2022.

[82] G. Tang, X. Yan, e X. Wang, "Chaotic signal denoising based on adaptive smoothing multiscale morphological filtering," *Complexity*, vol. 2020, págs. 1–14, fev. 2020.

[83] Y. Chen e Y. Zhang, "Chaotic signal denoising using an improved wavelet thresholding algorithm," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, págs. 200–203, IEEE, mai. 2021.

[84] H. Jinwang, L. Shanxiang, e C. Yue, "Chaotic signal denoising algorithm based on self-similarity," *Chinese Journal of Electronics*, vol. 30, págs. 482–488, mai. 2021.

[85] Y. Huang, W. Jin, e L. Li, "A new wavelet shrinkage approach for denoising nonlinear time series and improving bearing fault diagnosis," *IEEE Sensors Journal*, vol. 22, págs. 5952–5961, mar. 2022.

[86] Z. Chen, L. Zhang, W. Wang, e Z. Wu, "A pre-coded multi-carrier m-ary chaotic vector cyclic shift keying transceiver for reliable communications," *IEEE Transactions on Wireless Communications*, vol. 21, págs. 1007–1021, fev. 2022.

[87] R. Luo, H. Yang, C. Meng, e X. Zhang, "A novel SR-DCSK-based ambient backscatter communication system," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, págs. 1707–1711, mar. 2022.

[88] H. Ma, Y. Fang, Y. Tao, P. Chen, e Y. Li, "A novel differential chaos shift keying scheme with transmit diversity," *IEEE Communications Letters*, vol. 26, págs. 1668–1672, jul. 2022.

[89] T. Fan, W. Xu, L. Wang, e L. Zhang, "A new APSK-based m-ary differential chaos shift keying modulation system," *IEEE Communications Letters*, vol. 24, págs. 2701–2704, dez. 2020.

[90] H.-P. Ren, S.-L. Guo, C. Bai, e X.-H. Zhao, "Cross correction and chaotic shape-forming filter based quadrature multi-carrier differential chaos shift keying communication," *IEEE Transactions on Vehicular Technology*, vol. 70, págs. 12675–12690, dez. 2021.

[91] X. Cai, W. Xu, S. Hong, e L. Wang, "Discrete w transform based index-keying m-ary DCSK for non-coherent chaotic communications," *IEEE Communications Letters*, vol. 25, págs. 3104–3108, set. 2021.

[92] Z. Liu, L. Zhang, Z. Wu, e Y. Jiang, "Energy efficient parallel concatenated index modulation and m-ary PSK aided OFDM-DCSK communications with QoS consideration," *IEEE Transactions on Vehicular Technology*, vol. 69, págs. 9469–9482, set. 2020.

[93] C. Bai, H.-P. Ren, e G. Kolumban, "Double-sub-stream m-ary differential chaos shift keying wireless communication system using chaotic shape-forming filter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, págs. 3574–3587, out. 2020.

[94] Q. Han, L. Yang, J. Du, e L. Cheng, "Blind equalization for chaotic signals based on echo state network and kalman filter under nonlinear channels," *IEEE Communications Letters*, vol. 25, págs. 589–592, fev. 2021.

[95] L. Yang, Q. Han, J. Du, L. Cheng, Q. Li, e A. Zhao, "Online blind equalization for QAM signals based on prediction principle via complex echo state network," *IEEE Communications Letters*, vol. 24, págs. 1338–1341, jun. 2020.

[96] H.-P. Ren, H.-P. Yin, C. Bai, e J.-L. Yao, "Performance improvement of chaotic baseband wireless communication using echo state network," *IEEE Transactions on Communications*, vol. 68, págs. 6525–6536, out. 2020.

[97] Y. Sui, Y. He, T. Cheng, Y. Huang, e S. Ning, "Broad echo state network for channel prediction in MIMO-OFDM systems," *IEEE Transactions on Vehicular Technology*, vol. 69, págs. 13383–13399, nov. 2020.

[98] R. Marxer e H. Purwins, "Unsupervised incremental online learning and prediction of musical audio signals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, págs. 863–874, mai. 2016.

[99] R. A. Horn e C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, abr. 1991.

[100] A. V. Oppenheim, *Discrete-Time Signal Processing*. Pearson, 2009.

ANEXO A – CÓDIGO EM PYTHON PARA PROCESSAMENTO DE ARQUIVOS MIDI

Para executar o código apresentado neste anexo, é necessário que a biblioteca mido [39] esteja instalada. Também é preciso armazenar o caminho do diretório onde encontram-se os arquivos .midi na string *dir*, na linha 21 do código.

São gerados dois arquivos após a execução do código: o *bancoBruto.txt* e o *bancoProcessado.txt*. O primeiro é uma compilação de todas mensagens midi dos arquivos. O segundo, possui cada linha na forma nota-instante de início-instante de término de todas as notas de cada música, levando-se em consideração o agrupamento por *tracks*.

Os arquivos são processados por ordem alfabética de seus nomes e, caso desejado, as linhas 50-57 podem ser executadas para incluir o nome de cada música no arquivo *bancoProcessado.txt*.

```
1  #importa as bibliotecas que serao utilizadas
2  import mido
3  import os
4  from mido import MidiFile
5  # funcao que retorna somente os numeros na string "palavra"
6  def pegaNumeros(palavra):
7    numeros=""
8    for letra in palavra:
9        if letra.isdigit():
10         numeros = numeros+str(letra)
11   return numeros
12 #funcao que retorna o tick correspondente a linha
13 def verificaTime(t,linha):
14   linha = linha.split()
15   ultimoElemento = linha[np.size(linha)-1]
16   if ultimoElemento[:4] == "time":
```

```
17      ultimoValor = pegaNumeros(ultimoElemento)
18      t = t+int(ultimoValor)
19    return t
20 #define o nome do diretorio onde ha os arquivos .midi
21 dir = 'Arquivos MIDI/'
22 #armazena todos os nomes dos arquivos no diretorio
23 MIDIfiles = os.listdir(dir)
24 #coloca as musicas em ordem alfabetica
25 MIDIfiles = sorted(MIDIfiles)
26 #loop para escrever o conteudo de todos arquivos .midi do diretorio no
27 #arquivo de texto bancoBruto.txt
28 for n in range(0,np.size(MIDIfiles)):
29   #caminho de cada arquivo .midi no diretorio
30   caminho = dir+MIDIfiles[n]
31   #armazena todas as mensagens do arquivo .midi localizado na variavel
       acima
32   #na variavel todasMensagens
33   todasMensagens = MidiFile(caminho,clip=True)
34   with open('bancoBruto.txt','a') as arq:
35     arq.write(str(todasMensagens))
36     arq.write('\n')
37 #abre o arquivo de texto criado para comecar a retirar as informacoes de
38 #interesse das mensagens dos arquivos .midi
39 with open('bancoBruto.txt','r') as arq:
40   #le a proxima linha de bancoBruto.txt
41   linha = arq.readline()
42   t = 0
43   while linha != []:
44     t = verificaTime(t,linha)
45     aux = linha.split()[0]
46     if aux[:9] == "MidiTrack":
47       t = 0
48     #as linhas comentadas em sequencia abaixo servem para incluir
       informacoes
49     #sobre a musica, como o nome dela, no arquivo de texto
50     # aux = linha.split()
51     # aux2 = ""
52     # if aux[0] == "MetaMessage('track_name',":
53     #   for n in range(1,np.size(aux)-1):
54     #     aux2 = aux2+aux[n]
55     #   with open('bancoProcessado.txt','a') as arq2:
56     #     arq2.write(aux2)
57     #     arq2.write('\n')
```

```
58      #identifica um evento note-on
59      if linha.split()[0] == "Message('note_on',":
60        if int(pegaNumeros(linha.split()[3])) != 0:
61          #armazena a posicao do evento note-on no arquivo de texto na
    variavel p
62          p = arq.tell()
63          #armazena o tick de inicio da nota na variavel onTick
64          onTick = t
65          #armazena a altura da nota na variavel nota
66          nota = pegaNumeros(linha.split()[2])
67          linha = arq.readline()
68          #sai do loop while se nao houver mais mensagens no arquivo bruto
69          if not(linha):
70            break
71          t = verificaTime(t,linha)
72          #loop para buscar em qual tick a execucao da nota foi interrompida
73          while pegaNumeros(linha.split()[2]) != nota:
74            linha = arq.readline()
75            #sai do loop while se nao houver mais mensagens no arquivo bruto
76            if not(linha):
77              break
78            t = verificaTime(t,linha)
79          #armazena o tick de termino da nota na variavel offTick
80          offTick = t
81          #loop para escrever as informacoes de interesse da nota num arquivo
     de texto
82          with open('bancoProcessado.txt','a') as arq2:
83            dados = nota+'-'+str(onTick)+'-'+str(offTick)
84            arq2.write(dados)
85            arq2.write('\n')
86          #retorna para a posicao em que ocorreu o evento da nota
87          arq.seek(p)
88          #converte o onTick para tipo inteiro e armazena na variavel t
89          t = int(onTick)
90      #passa para a linha abaixo do evento note-on
91      linha = arq.readline()
92      #sai do loop while se nao houver mais mensagens no arquivo bruto
93      if not(linha):
94              break
```

ANEXO B – CÓDIGO EM MATLAB® PARA SOLUÇÃO DO EXEMPLO

O código a seguir foi utilizado para a solução do exemplo de predição de sinal apresentado na página 32.

```
%sinal de entrada
u = [6 28 496];
for n = 1 : 5
    u = [u u];
end

%matriz do sinal desejado
D = [496 6 28 496 6];

%parametros usados
N = 5;
a = 0.7;
lambda = 0.8;
p = 0;
q = 0.25;
l = 10;
L = 5;

%passo 1: geracao de Win
Win = 2*rand(N,2)-1;
Win(:,1) = p*Win(:,1);
Win(:,2) = q*Win(:,2);

%passo2: geracao de Waux e calculo de seu raio espectral
Waux = 2*rand(N,N)-1;
lambdaAux = max(abs(eigs(Waux)));
```

```
%passo 3: determinacao de W
W = lambda*Waux/lambdaAux;

%inicializacao do vetor de estados
r = zeros(N,1);

%passo 4: obtendo a matriz T
for n = -l+1 : L
    r = (1-a)*r+a*tanh(Win*[1;u(n+l)]+W*r);
    if n>0
        T(:,n) = r;
    end
end

%passo 5: determinacao de Wout
Wout = D*pinv(T);

%testando a rede por 6 vezes
for n = L+1 : L+6
    r = (1-a)*r+a*tanh(Win*[1;u(n+l)]+W*r)
    instante = n
    y = Wout*r
end
```

ANEXO C – OBRAS MUSICAIS UTILIZADAS

Tabela C.1: Lista das peças de Frédéric Chopin utilizadas para o estudo do Caso 1, realizado no Capítulo 5, e os instantes correspondentes.

Obra	**Amostra Inicial**	**Amostra Final**
Op. 7, No. 1	−199	1029
Opus 7, No. 2	1030	2236
Opus 10, No. 1	2237	3573
Opus 10, No. 5	3574	5204
Opus 10, No. 12	5205	7292
Opus 18	7293	11707
Opus 23	11708	16735
Opus 25, No. 1	16736	18957
Opus 25, No. 2	18958	20160
Opus 25, No. 3	20161	21853
Opus 25, No. 4	21854	23360
Opus 25, No. 11	23361	26532
Opus 25, No. 12	26533	29166
Opus 27, No. 1	29167	30799
Opus 27, No. 2	30800	32630
Opus 31	32631	38606

Continua na próxima página

Tabela C.1: Lista das peças de Frédéric Chopin utilizadas para o estudo do Caso 1, realizado no Capítulo 5, e os instantes correspondentes. (Continuação)

Obra	**Amostra Inicial**	**Amostra Final**
Opus 33, No. 2	38607	40539
Opus 33, No. 4	40540	42704
Opus 35, No. 1	42705	48318
Opus 35, No. 2	48319	53093
Opus 35, No. 3	53094	55420
Opus 35, No. 4	55421	57180
Opus 53	57181	63232
Opus 66	63233	66282
Opus 28, No. 1	66283	66709
Opus 28, No. 2	66710	66951
Opus 28, No. 3	66952	67614
Opus 28, No. 4	67615	68218
Opus 28, No. 5	68219	68678
Opus 28, No. 6	68679	69083
Opus 28, No. 7	69084	69250
Opus 28, No. 8	69251	70808
Opus 28, No. 9	70809	71221
Opus 28, No. 10	71222	71543
Opus 28, No. 11	71544	71863
Opus 28, No. 12	71864	73106
Opus 28, No. 13	73107	74010
Opus 28, No. 14	74011	74444
Opus 28, No. 15	74445	75962
Opus 28, No. 16	75963	77161

Continua na próxima página

Tabela C.1: Lista das peças de Frédéric Chopin utilizadas para o estudo do Caso 1, realizado no Capítulo 5, e os instantes correspondentes. (Continuação)

Obra	**Amostra Inicial**	**Amostra Final**
Opus 28, No. 17	77162	79656
Opus 28, No. 18	79657	80229
Opus 28, No. 19	80230	81466
Opus 28, No. 20	81467	81752
Opus 28, No. 21	81753	82630
Opus 28, No. 22	82631	83258
Opus 28, No. 23	83259	83784
Opus 28, No. 24	83785	85155

ANEXO D – EXEMPLO DE PROCESSAMENTO MIDI

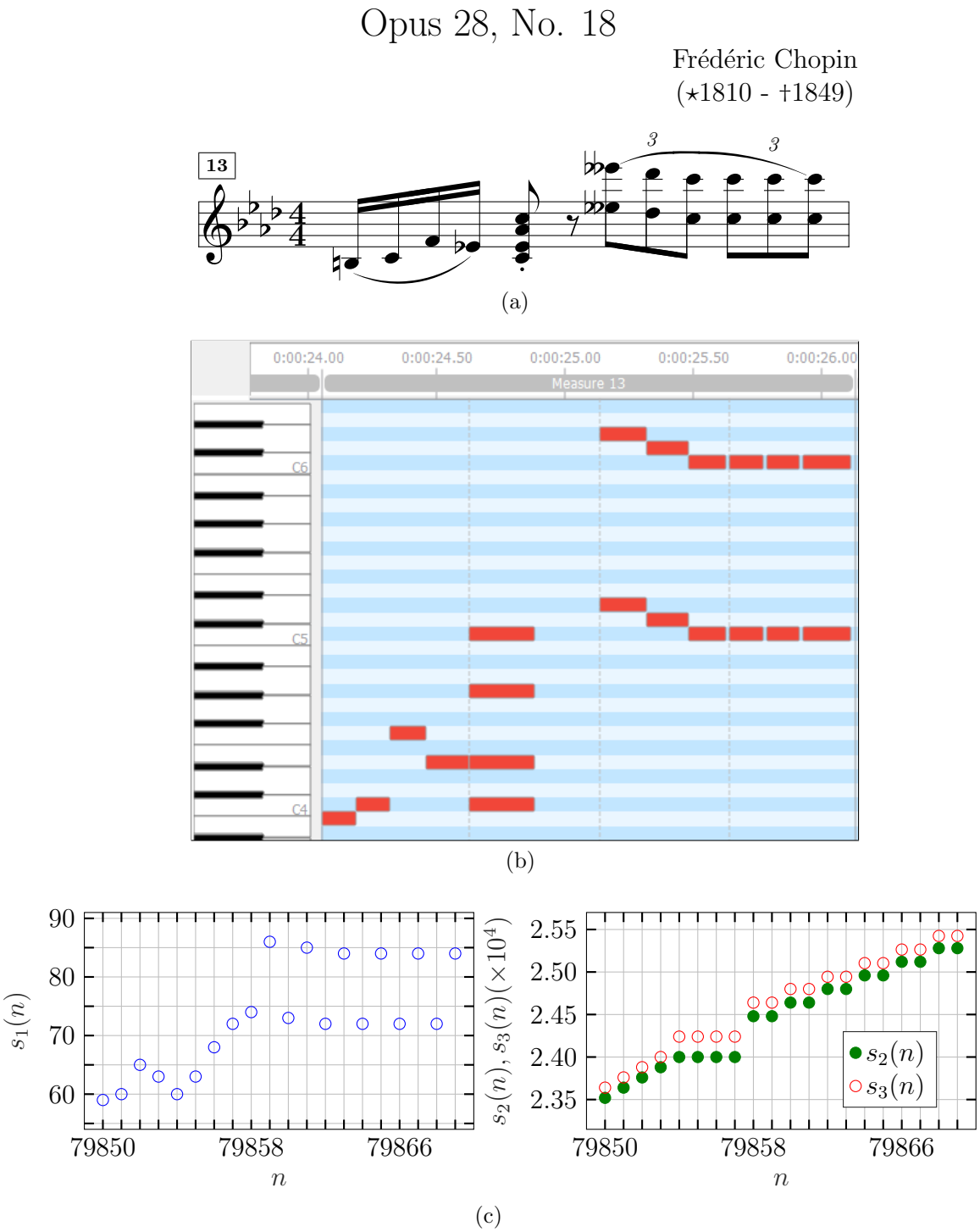

Figura D.1: Processamento de um dos *tracks* do compasso 13 da obra Opus 24, No. 18: a) extrato da partitura musical correspondente; b) trecho do arquivo MIDI correspondente; c) à esquerda: altura das notas. À direita: instantes de início (verde) e de término (vermelho) em *ticks*.

ANEXO E – ARTIGO SUBMETIDO *SIGNAL PROCESSING* 2023

Denoising of discrete-time chaotic signals using echo state networks

André L. O. Duarte, Marcio Eisencraft

Escola Politécnica, University of São Paulo, São Paulo, Brazil

Abstract

Noise reduction is a relevant topic when considering the application of chaotic signals in practical problems, such as communication systems or modeling biomedical signals. In this paper an echo state network (ESN) is employed to denoise a discrete-time chaotic signal corrupted by additive white Gaussian noise. The choice for applying ESNs in this context is motivated by their successful exploitation for separation and prediction of chaotic signals. The results show that the processing gain of ESN is higher than that of the Wiener filter, especially when the power spectral density of the chaotic signals is white.

Keywords: echo state networks, denoising, dynamical systems, machine learning, reservoir computing.

2010 MSC: 00-01, 99-00

1. Introduction

As the number of researches on new communication schemes using chaotic signals as broadband carriers grows, see e.g. [1, 2], it makes sense to look for ways to reduce the noise in chaotic signals.

In fact, many different techniques have been proposed in an effort to solve the denoising problem [3]. In recent years, there has been an increased interest in machine learning techniques due to their great adaptability, despite their rather simple design principles. Therefore, it is natural to investigate their performance for noise reduction.

In particular, Echo State Networks (ESNs) emerged in the early 2000s in an effort to mitigate the problems faced when designing Recurrent Neural Networks (RNNs): (i) the training of RNNs is cumbersome when using classical methods based on gradient descent, such as backpropagation; (ii) the lack of convergence guarantee and (iii) the high computational effort required due to the large number of parameters to optimize [4].

The distinguishing feature of ESNs is that only the weights in the output layer are optimized during training [4]. In this scenario, the network without its input and output nodes is called a *reservoir*. The remaining links connecting the input to the reservoir and the connections within the reservoir have weights assigned according to a random distribution [5]. These weights do not change during training, minimizing the number of parameters to optimize [4]. They are widely used due to their simplicity and low computational cost, for example in many applications related to dynamical systems, such as the separation [6] and prediction [7] of chaotic signals.

In [8] an investigation of ESNs for denoising was made. There, a wave signal consisting of four sinusoids with random phases and time-varying envelopes is corrupted by additive white Gaussian noise (AWGN). At [9], ESNs were used to denoise electroencephalogram signals. The performance was evaluated in terms of the signal-to-noise ratio (SNR) and the root mean square error.

In the present work, discrete-time chaotic signals are disturbed by AWGN. The aperiodic behavior together with the sensitive dependence on initial conditions of chaotic signals [10] add to the difficulty of the denoising task. Performance is measured in terms of processing gain, providing a clearer and more meaningful understanding of the network's capabilities.

The skew tent map [11] is used to obtain the chaotic signals studied in this article. The power spectral density (PSD) of the orbits of this map is determined by its single parameter in a simple deterministic way [11]. The rationale for considering this map here is thus justified, because knowing the PSD in the context of noise reduction is more than welcome and useful for performing an analysis of how orbits with different PSDs affect the denoising task.

Figure 1 shows a block diagram of the problem at hand. After the training period, the ESN output $\boldsymbol{y}(n)$ is expected to approximate the desired noiseless chaotic signal $\boldsymbol{d}(n)$ from the corrupted input signal

$$\boldsymbol{u}(n) = \boldsymbol{d}(n) + \boldsymbol{w}(n), \tag{1}$$

where $\boldsymbol{w}(n)$ is AWGN.

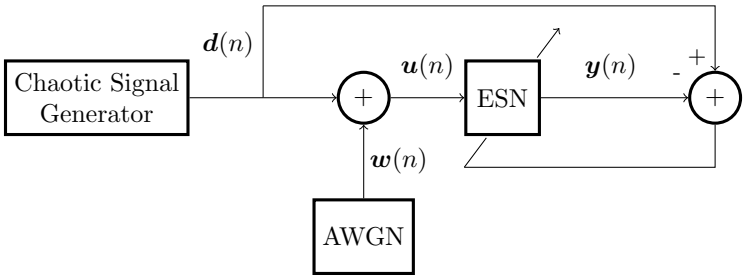

Figure 1: Problem formulation: noise reduction of a chaotic signal using ESN.

It is relevant to note that in this paper we consider chaotic signals generated by a discrete-time system, unlike most of the literature that generally considers off-the-shelf continuous-time dynamical systems, such as Lorenz or Rossler systems [12, 13]. The task at hand here is more challenging because it does not benefit from the smoothness found in continuous-time signals. Depending on its parameter, the skew tent map generate white signals with impulsive autocorrelation that are not easily distinguishable from white noise.

The remainder of this article is divided into four sections. Section 2 gives a brief description of the chaotic signal generator considered here. Section 3 discusses the operation of the ESN. Section 4 presents and discusses the results obtained. Finally, our main conclusions are summarized in Section 5.

2. Considered Chaotic Signals

Chaotic signals are bounded, aperiodic and have sensitive dependence on initial conditions (SDIC) [10]. The one-dimensional discrete-time chaotic signal $d(n)$ considered in this paper is obtained from the skew tent map defined by

[11]

$$d(n+1) = f(d(n)) = \begin{cases} \frac{1-\alpha}{1+\alpha} + \frac{2}{1+\alpha} d(n), & -1 < d(n) < \alpha \\ \frac{1+\alpha}{1-\alpha} - \frac{2}{1-\alpha} d(n), & \alpha \leq d(n) < 1 \end{cases}, \tag{2}$$

with initial condition $d(0) \triangleq d_0 \in (-1, 1)$ and fixed parameter $\alpha \in (-1, 1)$, the x-coordinate of the peak of the tent. Figure 2(a) shows a plot of $f(\cdot)$ and Figure 2(b) depicts two orbits of $f(\cdot)$, with slightly different initial conditions, to illustrate the SDIC property.

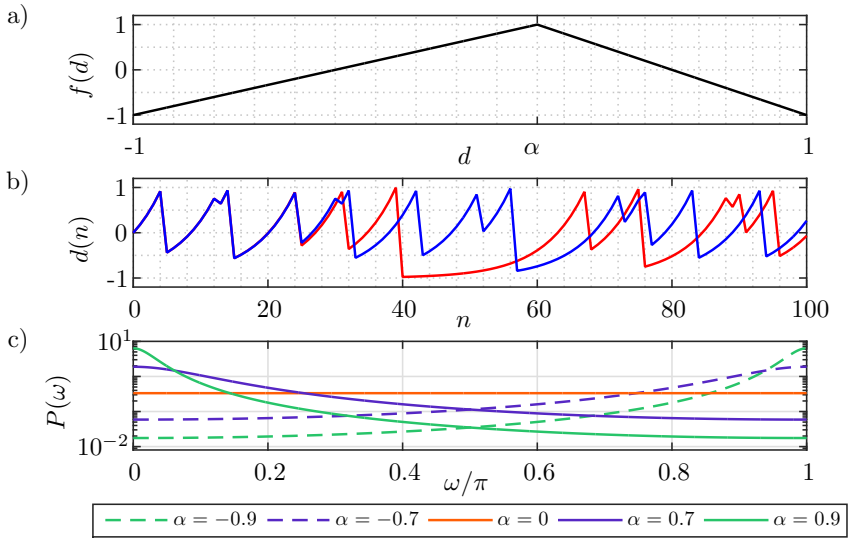

Figure 2: a) The skew tent map (2); b) two orbits for $\alpha = 0.7$ with initial conditions $d_0 = 0$ (red line) and $d_0 = 10^{-6}$ (blue line) showing SDIC; c) PSD for different values of α.

In [11] it was shown that the PSD of the orbits of $f(\cdot)$ is given by

$$P(\omega) = \frac{1 - \alpha^2}{3(1 + \alpha^2 - 2\alpha \cos \omega)}. \tag{3}$$

As shown in Figure 2(c), $P(\omega)$ is white for $\alpha = 0$. Also, the higher the value of $|\alpha|$, the more power is concentrated at high ($\alpha > 0$) or low ($\alpha < 0$) frequencies. In other words, the further away from $\alpha = 0$, the more the spectral characteristics of the generated chaotic signal will differ from a white noise signal. This direct relationship between the α parameter and the PSD is what led us to

choose this map. It allows us to analyze, in Section 4, the influence of the PSD of the chaotic signal on the performance of the ESN in the noise reduction task.

3. Echo State Networks

Figure 3 shows a schematic of an ESN. Its purpose is to use an input signal $\boldsymbol{u}(n)$ to approximate a target signal $\boldsymbol{d}(n)$ after a training period. It consists of (i) an input layer, (ii) the so-called reservoir and (iii) an output layer. Each of these parts is composed of information processing nodes connected by links, through which information is exchanged [4, 5].

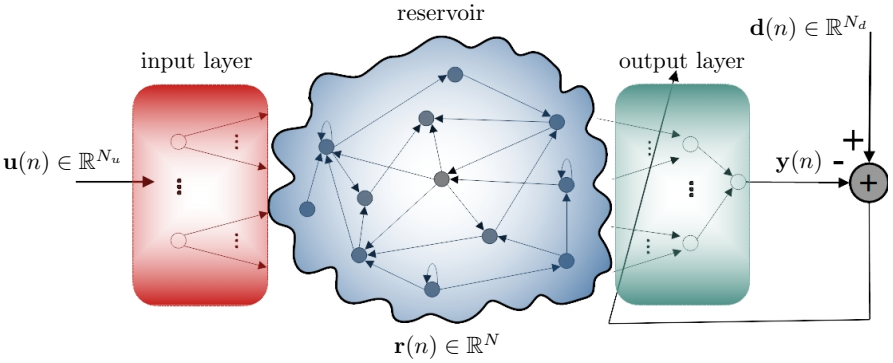

Figure 3: ESN architecture, indicating the signals $\boldsymbol{u}(n)$, $\boldsymbol{y}(n)$, $\boldsymbol{d}(n)$ and the internal state vector $\boldsymbol{r}(n)$.

In the following, we review the main aspects of each layer of the considered ESN.

3.1. Input Layer

The role of the input layer is to preprocess the input signal in order to control the amount of nonlinearity of the ESN [5].

Given an input signal $\boldsymbol{u}(n) \in \mathbb{R}^{N_u}$, the input layer computes $\boldsymbol{W}^{\text{in}}\left[1\ \boldsymbol{u}(n)\right]^T$, where $\boldsymbol{W}^{\text{in}}$ is $N \times N_u + 1$. So there are $N_u + 1$ entry nodes, one for each dimension of $\boldsymbol{u}(n)$ and an extra one for the bias. As it is shown in the following subsection, $\boldsymbol{W}^{\text{in}}\left[1\ \boldsymbol{u}(n)\right]^T$ is part of the argument of a $\tanh(\cdot)$ function, which is present in the state vector update equation (4). Since $\tanh(\cdot)$ has approximately

linear behavior for arguments close to zero, but nonlinear for larger arguments, it follows that $\boldsymbol{W}^{\text{in}}$ effectively determines the amount of nonlinearity behind the ESN operation.

The matrix $\boldsymbol{W}^{\text{in}}$ is called the *input matrix* and its entries are samples from uniform distributions. The first column entries are drawn from a uniform distribution in $[-p, p]$, while the remaining entries follow a uniform distribution in $[-q, q]$. Both p and q are chosen here according to the algorithm described in Section 4.

3.2. Reservoir

The reservoir consists of N nodes connected to each other. The weight of each link is randomly determined and are the entries of the *internal matrix* $\boldsymbol{W}_{N\times N}$. They are obtained as follows: first, the entries of an auxiliary matrix $\boldsymbol{W}^{\text{aux}}_{N\times N}$ are drawn from a uniform distribution in $[-1, 1]$. Then its spectral radius ρ is determined. Finally, we compute $\boldsymbol{W} = \lambda\,(\boldsymbol{W}^{\text{aux}}/\rho)$, where λ is a parameter. Since $\boldsymbol{W}^{\text{aux}}/\rho$ has unit spectral radius, it follows that $\boldsymbol{W}$ has spectral radius λ. This parameter is selected here using the algorithm described in Section 4.

Each node k, $1 \leq k \leq N$, has an internal state $r_k \in \mathbb{R}$, which forms the internal state vector $\boldsymbol{r}(n) \in \mathbb{R}^N$. At each time $n+1$, the internal state vector is updated according to the leaky-integrator model equation

$$\boldsymbol{r}(n+1) = (1-a)\,\boldsymbol{r}(n) + a\tanh\left(\boldsymbol{W}^{\text{in}}\begin{bmatrix}1 & \boldsymbol{u}(n)\end{bmatrix}^T + \boldsymbol{W}\boldsymbol{r}(n)\right), \qquad (4)$$

where the leakage parameter is $a \in [0, 1]$ and the initial condition is $\boldsymbol{r}(-\ell+1) = \boldsymbol{0}$.

After a transient period of ℓ time steps the training period begins. During $L \in \mathbb{N}$ training time steps, the corresponding state vectors are collected in the trajectory matrix $\boldsymbol{T}_{N\times L}$, defined by $\boldsymbol{T} = [\,\boldsymbol{r}(1)\ \boldsymbol{r}(2)\ \cdots\ \boldsymbol{r}(L)\,]$. which is passed to the output layer.

3.3. Output layer

The goal of the ESN is to have an output that approximates a desired signal $\boldsymbol{d}(n) \in \mathbb{R}^{N_d}$. Let $\boldsymbol{D}_{N_d \times L}$ be the matrix of samples of the desired signal available for training, i.e., $\boldsymbol{D} = [\, \boldsymbol{d}(1)\ \boldsymbol{d}(2)\ \ldots\ \boldsymbol{d}(L)\,]$. The optimized weights of the connections from the reservoir to the output are given by $\boldsymbol{W}^{\text{out}} = \boldsymbol{D}\boldsymbol{T}^{+}$, where $\boldsymbol{T}^{+}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{T}$. After training, the output signal of the ESN is calculated as the linear combination $\boldsymbol{y}(n) = \boldsymbol{W}^{\text{out}}\boldsymbol{r}(n)$.

At this point, our description of the ESN is almost complete. Only four parameters remain to be adjusted: (i) the leakage parameter a, (ii) the spectral radius λ of $\boldsymbol{W}$ and the scalars (iii) p and (iv) q that define the intervals of the uniform distributions used to obtain $\boldsymbol{W}^{\text{in}}$. Our methodology for choosing them is described along with the numerical results in Section 4.

4. Numerical simulations

The desired one-dimensional chaotic signal $d(n)$ is generated by (2) for a given α and initial condition d_0. The corrupted input $u(n)$ of the ESN is then (1), where the AWGN $w(n)$ power is determined by a chosen SNR_{in}. Figure 4 shows examples of $d(n)$, $w(n)$ and $u(n)$ for $d_0 = 0$, $\alpha = 0.7$ and $\text{SNR}_{\text{in}} = 2.0$ dB.

Our goal is to train the ESN to reduce the noise component in $u(n)$. This means that after sufficient training, the output signal $y(n)$ of the network should mimic the desired chaotic signal $d(n)$, using only $u(n)$ as input.

The ESN was designed with $N = 500$ nodes, a transient of $\ell = 200$ samples, and $L = 25000$ samples for training. For the estimation of the output SNR (SNR_{out}), 10^6 samples were considered.

4.1. Selecting the ESN parameters

To select the ESN parameters a, λ, p and q, we have generated $u(n)$ with an arbitrary $d_0 \in [-1, 1]$, $\alpha = 0.1$ and $\text{SNR}_{\text{in}} = 2.0$ dB.

As a first step, we consider $\lambda = 0.05$, $p = 0$ and $q = 0.5$. Then, keeping them fixed, we have varied a from 0 to 1 in 0.05 steps. In each case, the ESN

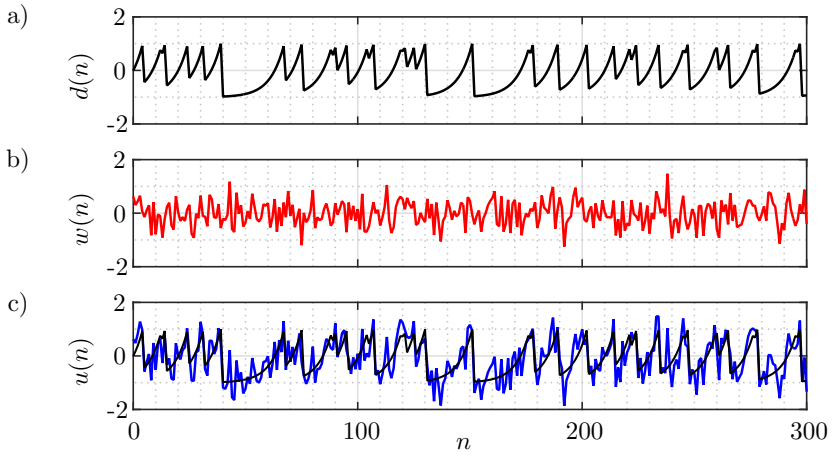

Figure 4: Examples of training signals: a) noiseless chaotic signal $d(n)$ for $\alpha = 0.7$ and $d_0 = 0$; b) AWGN for $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB; c) input signal $u(n)$ (blue line) for the ESN; $d(n)$ is also shown for comparison.

was trained and tested. Thus, the value $a = a_1^{\mathrm{opt}}$ that maximizes $G[\mathrm{dB}] \triangleq \mathrm{SNR}_{\mathrm{out}}[\mathrm{dB}] - \mathrm{SNR}_{\mathrm{in}}[\mathrm{dB}]$ is determined. Then, analogously, we get λ_1^{opt}, where λ varies between 0.05 and 1 in 0.05 steps, using $a = a_1^{\mathrm{opt}}$, $p = 0$ and $q = 0.5$. Next, p_1^{opt} and q_1^{opt} were obtained, in that order, by first varying p between 0 and 10 in 0.5 steps and q between 0.5 and 10 in 0.5 steps, keeping all other parameters at their current selected values.

This procedure is repeated, obtaining a_i^{opt}, λ_i^{opt}, p_i^{opt}, q_i^{opt}, $i = 2, 3, \ldots$. The search continues until the selected values for each parameter do not change.

The evolution of the selected parameters along the procedure is shown in Figure 5. It took 12 iterations for all parameters to stop changing. The final selected parameters are $a_\star = 0.8$, $\lambda_\star = 0.75$, $p_\star = 1.5$ and $q_\star = 1.0$ which are used in our simulations.

4.2. Denoising results and analysis

Figure 6 shows examples of the estimated signals using ESN and a Wiener Filter (WF) [14] for $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB and skew tent map parameter $\alpha = 0.9$. We denote the ESN estimated signal by $\widehat{d}(n)$ and the WF estimated signal by

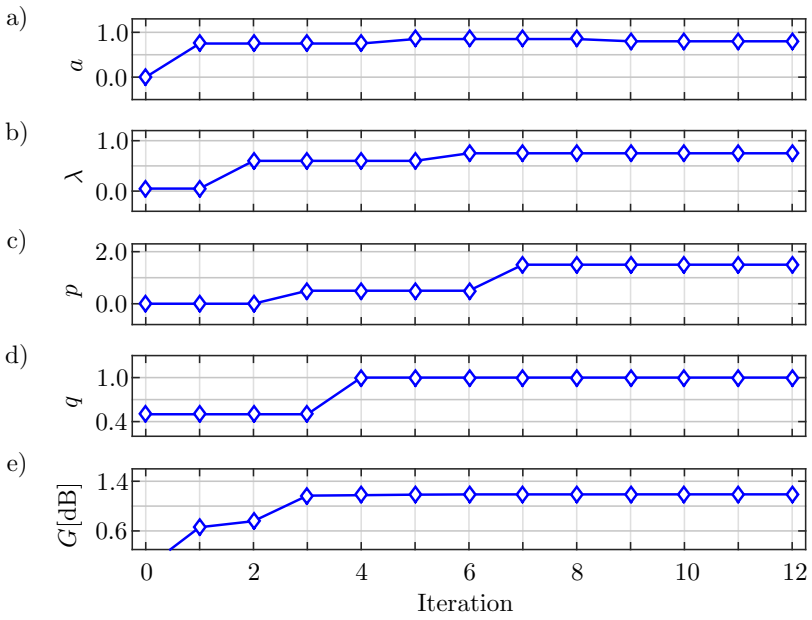

Figure 5: Selection of the ESN parameters. Parameters at each iteration: a) a: leakage parameter ; b) λ: spectral radius of $\boldsymbol{W}$; c) p and d) q: parameters of the uniform distributions used to determine the entries of $\boldsymbol{W}^{\text{in}}$; e) processing gain G[dB] at each iteration.

$\widehat{d}_{\text{W}}(n)$. In this case, $\text{SNR}_{\text{out}} = 7.7$ dB for the ESN and $\text{SNR}_{\text{out}} = 5.1$ dB for the WF, showing that the ESN outperforms the WF in this scenario.

As can be seen from (3) and Figure 2(c), the PSD of the chaotic signal becomes whiter as $|\alpha|$ approaches zero. Thus, in terms of autocorrelation it becomes similar to the corrupting noise, denoising it becomes more difficult and one can expect the processing gain to decrease with $|\alpha|$.

Using the optimal parameters obtained in Section 4.1, the ESN was trained and tested for different values of $\alpha \in (-1, 1)$. Figure 7 presents the obtained results in terms of mean processing gain after five training/testing scenarios. As a benchmark, a WF with 10 taps was employed to perform the same denoising task. The error bars indicate the standard deviation calculated from the five repetitions.

The presented results show that the ESN outperforms the WF for all values

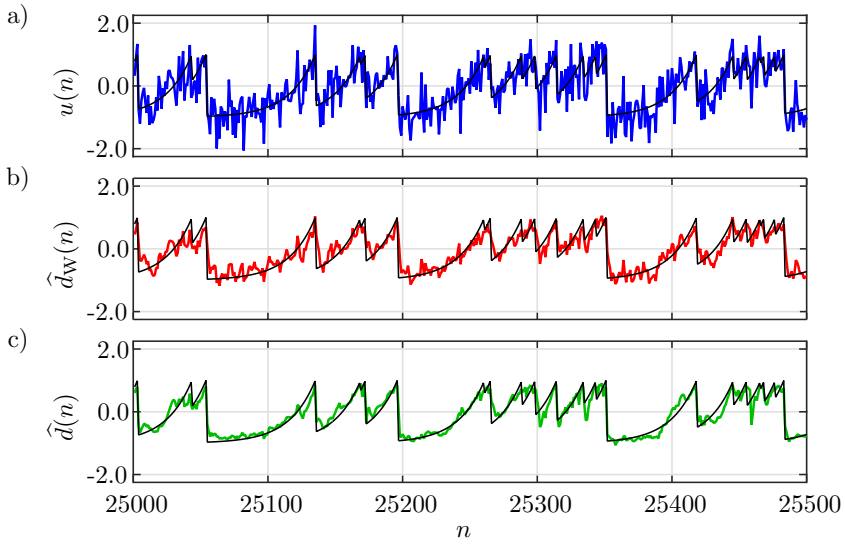

Figure 6: Noise reduction for $\alpha = 0.9$ and $\mathrm{SNR}_{\mathrm{in}} = 2.0$ dB. Black lines represent the desired signal $d(n)$. a) Input signal $u(n)$; b) WF estimated signal $\widehat{d}_{\mathrm{W}}(n)$; c) ESN estimated signal $\widehat{d}(n)$.

of α. Furthermore, as expected, the performance of both techniques degrades as α approaches zero. Note that for $\alpha = 0$, the PSD of the chaotic signal is white and the processing gain for the WF is zero. The processing gain is maximum for $\alpha = 0.95$, where it reaches 7.17 ± 0.05 dB.

In papers such as [12, 13] processing gains ranging from 6.0 dB to 25.0 dB have been obtained using other techniques but always considering continuous-time chaotic systems. As mentioned in Section 1, we consider the task of denoising discrete-time chaotic signals to be much more difficult.

5. Conclusions

In this paper, we have analyzed the use of an ESN for noise reduction in a chaotic signal. Unlike other works in the literature, we considered signals generated by discrete-time maps, whose lack of smoothness makes the task more challenging. As chaotic signal generator, we considered the skew tent map be-

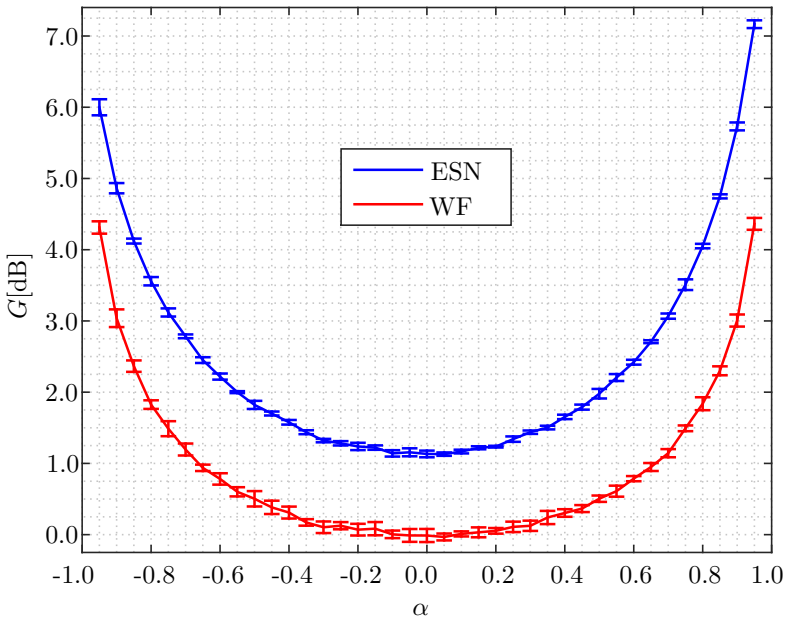

Figure 7: Processing gain in dB as a function of the skew tent map parameter α using an ESN and a WF.

cause the dependence of its PSD on its single parameter is known, which allows an analysis of the influence of spectral whiteness on the denoising performance.

We have shown that by selecting and tuning the network parameters, the ESN outperforms a WF, the optimal linear filtering technique, in all cases, especially when the PSD of the chaotic signal is white, resulting in zero gain for the WF. For increasing values of $|\alpha|$, the PSD of the chaotic signals becomes narrowband and the obtained processing gain is higher.

As a future investigation, we intend to evaluate how the proposed noise reduction technique can be used to improve the performance of chaos-based communication systems in noisy channels.

Acknowledgments

This work was partially supported by the National Council for Scientific and Technological Development (CNPq-Brazil) (grant number 311039/2019-7) and by the Coordination for the Improvement of Higher Education Personnel (CAPES-Brazil) (grant number 001).

References

[1] M. Eisencraft, R. R. F. Attux, R. Suyama (Eds.), Chaotic Signals in Digital Communications (Electrical Engineering & Applied Signal Processing Series), Taylor & Francis Inc, 2013.

[2] M. S. Baptista, Chaos for communication, Nonlinear Dynamics 105 (2) (2021) 1821–1841. doi:10.1007/s11071-021-06644-4.

[3] S. V. Vaseghi, Advanced Digital Signal Processing and Noise Reduction, 4th Edition, Wiley, 2009.

[4] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks, Tech. Rep. 148, German National Research Center for Information Technology (2001).

[5] M. Lukoševičius, A practical guide to applying echo state networks, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 659–686. doi:10.1007/978-3-642-35289-8_36.

[6] S. Krishnagopal, M. Girvan, E. Ott, B. R. Hunt, Separation of chaotic signals by reservoir computing, Chaos: An Interdisciplinary Journal of Nonlinear Science 30 (2) (2020) 023123. doi:10.1063/1.5132766.

[7] M. Xu, M. Han, T. Qiu, H. Lin, Hybrid regularized echo state network for multivariate chaotic time series prediction, IEEE Transactions on Cybernetics 49 (6) (2019) 2305–2315. doi:10.1109/tcyb.2018.2825253.

[8] L. de Oliveira Junior, F. Stelzer, L. Zhao, Clustered and deep echo state networks for signal noise reduction, Machine Learning 111 (8) (2022) 2885–2904. doi:10.1007/s10994-022-06135-6.

[9] W. Sun, Y. Su, X. Wu, X. Wu, Y. Zhang, EEG denoising through a wide and deep echo state network optimized by UPSO algorithm, Applied Soft Computing 105 (2021) 107149. doi:10.1016/j.asoc.2021.107149.

[10] K. T. Alligood, T. Sauer, Chaos: An Introduction to Dynamical Systems, Springer, 1997.

[11] M. Eisencraft, D. Kato, L. Monteiro, Spectral properties of chaotic signals generated by the skew tent map, Signal Processing 90 (1) (2010) 385–390. doi:10.1016/j.sigpro.2009.06.018.

[12] M. Han, Y. Liu, J. Xi, W. Guo, Noise smoothing for nonlinear time series using wavelet soft threshold, IEEE Signal Processing Letters 14 (1) (2007) 62–65. doi:10.1109/lsp.2006.881518.

[13] S. Lou, J. Deng, S. Lyu, Chaotic signal denoising based on simplified convolutional denoising auto-encoder, Chaos, Solitons and Fractals 161 (2022) 112333. doi:10.1016/j.chaos.2022.112333.

[14] S. S. Haykin, Adaptive filter theory, Prentice Hall, 2002.

ANEXO F – ARTIGO CNMAC 2021

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 8, n. 1, 2021.

Trabalho apresentado no XL CNMAC, Evento Virtual - Co-organizado pela Universidade do Mato Grosso do Sul (UFMS).

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Aplicação de *Reservoir Computing* para Filtragem de Sinais Caóticos Imersos em Ruído Branco Gaussiano

André L. O. Duarte[1]
EPUSP, São Paulo, SP
Marcio Eisencraft[2]
EPUSP, São Paulo, SP

A extração de um sinal imerso em ruído é um problema que surge em diversas áreas diferentes como sistemas de comunicação [4], análise de imagens médicas e processamento de áudio [1]. É, desta maneira, um problema de fundamental relevância [1].

Com o passar do tempo, diversos métodos foram desenvolvidos para solucionar tal problema e, recentemente, técnicas de aprendizagem de máquina vêm ganhando atenção por possuírem grande capacidade de adaptação e ao mesmo tempo estarem baseadas em princípios simples [2].

Reservoir computing é uma técnica de treinamento de redes neurais recorrentes que pode ser aplicada em diversas áreas de processamento de sinais, dentre elas a separação [1] e a predição [2] de sinais caóticos.

Sinais caóticos são limitados em amplitude, aperiódicos e apresentam dependência sensível com as condições iniciais [4]. Seu estudo é relevante devido ao grande número de processos naturais que apresentam comportamento caótico [4]. Por exemplo, pesquisas recentes vêm mostrando a possibilidade de implementações práticas de sistemas de comunicação empregando sinais caóticos [4].

Neste trabalho, faz-se uma investigação inicial da aplicação de um *reservoir computer* (RC), para reduzir o ruído branco gaussiano sobre uma componente caótica gerada a partir do mapa de Hénon [2]. Assume-se que amostras do sinal caótico livre de ruído estão disponíveis para treinamento, permanecendo desconhecido o processo caótico que gerou o sinal. O RC foi implementado conforme descrito em [1]. Sua entrada é composta pela componente $x(n)$ caótica gerada pelo mapa de Hénon com condições iniciais aleatórias e parâmetros $a = 1{,}4$ e $b = 0{,}3$ [2]. Ela é corrompida por ruído branco gaussiano aditivo.

O período de treinamento, o número de nós do RC, o parâmetro de *leakage* e o raio espectral da matriz de pesos recorrentes influenciam diretamente no desempenho do RC [3]. Para determinar estes valores, foram feitas simulações variando-se um deles e mantendo-se os outros fixos observando-se qual valor de cada parâmetro fornece a maior relação sinal-ruído (SNR - *Signal-to-Noise Ratio*) na saída do RC.

Uma vez otimizados esses parâmetros, foram feitas simulações com 10 realizações para cada valor de SNR de entrada, e calculou-se a SNR na saída do RC. Os principais resultados são apresentados na Figura 1.

Em (a) e (b) pode-se observar o sinal de entrada e o de saída do RC, respectivamente, em comparação com a componente caótica $x(n)$, para uma SNR de entrada de 2,5 dB. Em (c) é apresentado a curva da SNR de saída obtida em dB para diversos valores de SNR de entrada, mostrando o desempenho do sistema.

[1] alduarte@lcs.poli.usp.br
[2] marcioft@usp.br

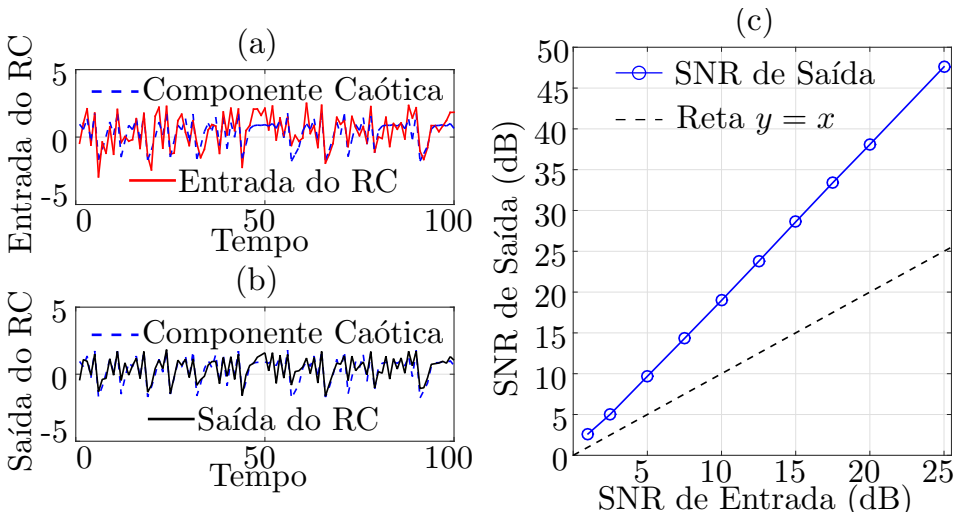

Figura 1: (a): Componente caótica e entrada do RC para uma SNR de entrada de 2,5 dB. (b): Componente caótica e saída fornecida pelo RC. (c): SNR de saída em função da SNR de entrada do RC.

Da Fig. 1 (c) nota-se que os resultados são satisfatórios, pois a SNR na saída foi sempre maior que a SNR na entrada do RC, mesmo quando esta é relativamente baixa. Ainda, quanto maior a SNR na entrada, maior o ganho.

Como sequência deste trabalho, pretende-se analisar o emprego de um RC em cenários mais desafiantes, como o de um sistema de comunicação empregando técnicas de modulação baseada em caos.

Referências

[1] Krishnagopal, S., Girvan, M., Ott, E., and Hunt, B. R. Separation of chaotic signals by reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, volume 30, issue 2, 023123, 2020. DOI: 10.1063/1.5132766

[2] Lellep, M., Prexl, J., Linkmann, M., and Eckhardt, B. Using machine learning to predict extreme events in the Hénon map. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, volume 30, issue 1, 013113, 2020. DOI: 10.1063/1.5121844

[3] Lukoševičius, M. A Practical Guide to Applying Echo State Networks, *Neural Networks: Tricks of the Trade*, Springer Lecture Notes in Computer Science, volume 7700, chapter 26, pages 659-686, 2012. DOI: 10.1007/978-3-642-35289-8_36.

[4] Oppenheim, A. V., Wornell, G. W., Isabelle, S. H., and Cuomo, K. M. Signal processing in the context of chaotic signals. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992. DOI: 10.1109/icassp.1992.226472

ANEXO G – ARTIGO SBRT 2021

Aplicação de *Reservoir Computing* para Filtragem de Sinais Caóticos Imersos em Ruído

André L. O. Duarte e Marcio Eisencraft

***Resumo*— Neste trabalho mostra-se a aplicação de um *reservoir computer* para mitigar ruído adicionado a uma componente caótica gerada a partir do mapa de Hénon. Assume-se que amostras do sinal caótico livre de ruído estão disponíveis para treinamento, permanecendo desconhecido o processo caótico que gerou o sinal. Adota-se como parâmetro de avaliação de desempenho do sistema a relação sinal-ruído e os resultados obtidos mostram que é possível aprimorá-la por meio da técnica de reservoir computing que, apesar de sua simplicidade, têm se mostrado um eficiente método para o treinamento de redes neurais recorrentes.**

***Palavras-Chave*— Sistemas dinâmicos, *reservoir computing*, *machine learning*, eliminação de ruído.**

***Abstract*— In this paper we show the application of a reservoir computer to mitigate noise added to a chaotic component generated from the Hénon map. The chaotic process that generated the signal remains unkown, only samples of the chaotic component are assumed available for training. As is common practice, the performance is measured through the signal-to-noise ratio and the outcome shows that it can be improved with the use of reservoir computing, which has been proving to be an efficient technique for recurrent neural network training despite its simplicity.**

***Keywords*— Dynamical systems, reservoir computing, machine learning, denoising.**

I. Introdução

A extração de um sinal imerso em ruído é um problema que surge em diversas áreas diferentes como sistemas de comunicação [1], análise de imagens médicas [2] e processamento de áudio [3]. É, desta maneira, um problema relevante e atual [4].

Com o passar do tempo, diversos métodos foram desenvolvidos para solucionar tal problema e, recentemente, técnicas de aprendizagem de máquina vêm ganhando atenção por possuírem grande capacidade de adaptação e ao mesmo tempo estarem baseadas em princípios simples [5].

O estudo de redes neurais abriu caminho para que o desenvolvimento de máquinas inteligentes, empregando redes neurais artificiais (ANNs - *Artificial Neural Networks*) baseadas no modelo biológico de neurônios e suas conexões, se tornasse possível [6].

Em uma ANN, os elementos fundamentais são chamados de nós, ou simplesmente neurônios por analogia com o modelo biológico. Os nós recebem e processam informação, e possuem um estado interno, também chamado de ativação, que é uma função de suas entradas [6].

André L. O. Duarte, Departamento de Telecomunicações e Controle, EPUSP, São Paulo-SP, e-mail: alduarte@lcs.poli.usp.br; Marcio Eisencraft, Departamento de Telecomunicações e Controle, EPUSP, São Paulo-SP, e-mail: marcioft@usp.br. Marcio Eisencraft foi parcialmente financiado por CNPq 311039/2019-7.

Tipicamente um nó transmite sua ativação para outros nós através de conexões, cada uma possuindo um peso característico, a ser determinado por meio de um algoritmo de treinamento [6]. Dependendo de como estas ativações escoam pela rede, as ANNs são classificadas como *feedforward* (FFNN - *Feedforward Neural Network*), em que as ativações trafegam somente no sentido de entrada para saída da rede e as recorrentes (RNN - *Recurrent Neural Network*), nas quais existe ao menos um caminho cíclico de conexões entre os nós [7].

Apesar de se acreditar que possuem um grande potencial latente a ser explorado, devido à sua capacidade de aproximar sistemas dinâmicos e, assim como o cérebro, possuir caminhos cíclicos [8], as RNNs são difíceis de serem treinadas utilizando os métodos clássicos, baseados no gradiente descendente [8]. A principal incoveniência imposta por tais métodos, é que muitas vezes a convergência não é garantida [9]. Outros empecilhos estão relacionados ao alto custo computacional envolvido, ao grande número de parâmetros para se ajustar [8] - como a otimização do peso de todas as conexões -, dentre outros como tratado em [10].

A arquitetura de uma RNN típica é apresentada na Figura 1. Nela destaca-se por cores distintas as três principais partes, formadas por conjuntos de nós, de uma RNN. As camadas em vermelho e verde indicam o conjunto dos nós de entrada e saída, respectivamente. A parte azul representa o emaranhado de nós internos e suas conexões, que formam os caminhos cíclicos que dão nome as RNNs.

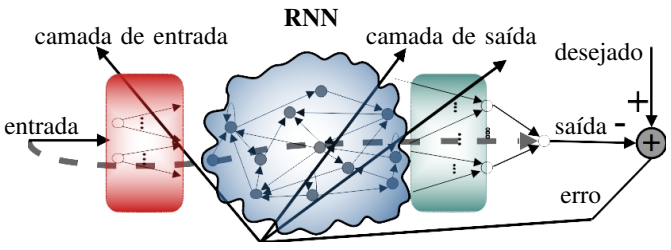

Fig. 1. Arquitetura de uma RNN, em que todas as conexões (entrada-RNN, internas e RNN-saída) são treinadas. Adaptado de [8].

Dado um sinal desejado, como a fala de um locutor [11], a atividade elétrica atrial de um paciente [2] ou o sinal transmitido por um satélite [12], RNNs podem ser treinadas de modo supervisionado ou não, para que sua saída aproxime o sinal de interesse [6]. Em geral, métodos de treinamento não supervisionado são mais difíceis de serem executados [6] e não são tratados aqui.

Buscando amenizar os problemas intrínsicos ao projeto de RNNs, no início dos anos 2000 foram propostas, de modo independente, novas e análogas formas de treinar e projetar

RNNs: as *Liquid State Machines* (LSMs) por Wolfgaang Maass e Henry Markram [13] e as *Echo State Networks* (ESNs) por Herbert Jaeger [14]. Estes trabalhos, juntamente com suas ramificações conseguintes, receberam o nome de *Reservoir Computing* [8].

Reservoir computers (RC) têm como essência a ideia de que somente os pesos das conexões RNN-saída devem ser ajustados [8], enquanto aqueles das conexões entre nós internos - ou simplesmente a RNN em si, nomeada então como *reservoir* - são gerados aleatoriamente e permanecem inalterados durante o treinamento, o mesmo ocorrendo com os pesos das conexões entrada-RNN.

Os RCs têm sido empregados com sucesso em diversas áreas de processamento de sinais, dentre elas a separação [4] e a predição [5] de sinais caóticos [15].

Neste trabalho, faz-se uma investigação inicial da aplicação de um RC, por meio de uma ESN, para reduzir o ruído branco gaussiano aditivo (AWGN - *Additive White Gaussian Noise*) sobre uma componente caótica gerada a partir do mapa de Hénon [5]. Assume-se que amostras do sinal caótico livre de ruído estão disponíveis para treinamento, permanecendo desconhecido o processo caótico que gerou o sinal.

O diagrama em blocos do problema abordado é apresentado na Figura 2.

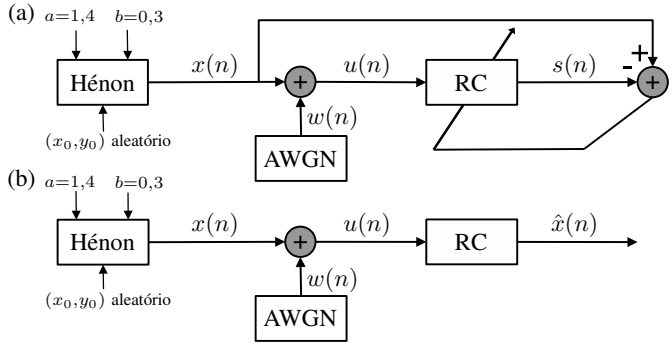

Fig. 2. Diagrama em blocos do problema estudado durante o (a): período de treinamento. (b): período de teste do RC.

A Fig. 2 (a) representa o período de treinamento do RC, e a Fig. 2 (b) ilustra o período de teste do RC, quando os pesos das conexões *reservoir*-saída foram ajustados e espera-se obter a melhor estimativa de $x(n)$ possível, indicada por $\hat{x}(n)$.

Este artigo está organizado da seguinte forma: na Seção II, revisitam-se os sinais caóticos e, em particular, o sistema dinâmico de Hénon utilizado aqui. Na Seção III é detalhado o funcionamento do RC, na Seção IV é explicada a metodologia utilizada, na Seção V é apresentado os resultados obtidos e por fim na Seção VI colocam-se as conclusões do artigo e propostas de trabalhos futuros.

II. Gerador de sinais caóticos utilizado

Sinais caóticos são limitados em amplitude, aperiódicos e apresentam dependência sensível em relação às condições iniciais [15]. Isto é, pequenas alterações nas condições iniciais podem levar a comportamentos completamente distintos.

Seu estudo é relevante devido ao grande número de processos naturais que apresentam comportamento caótico [1] desde a evolução do universo [16] até funções cerebrais [17]. Além disso, pesquisas recentes vêm mostrando a possibilidade de implementações práticas de sistemas de comunicação empregando sinais caóticos [1], o que levanta interesse na busca por métodos para se reduzir o ruído sobre estes.

A Figura 3 mostra uma órbita caótica [15] gerada pelo mapa de Hénon [5] com parâmetros $a = 1{,}4$ e $b = 0{,}3$ e $n = 0,\ 1,\ 2,\ \ldots$, definido pelas equações

$$\begin{aligned} x(n) &= a - (x(n-1))^2 + by(n-1), \\ y(n) &= x(n-1). \end{aligned} \tag{1}$$

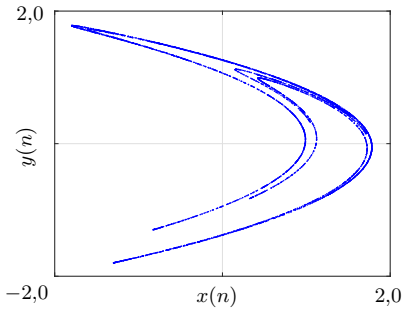

Fig. 3. Órbita caótica de Hénon obtida com $a = 1{,}4$, $b = 0{,}3$ e condições iniciais aleatórias.

Esse mapa é o utilizado nas próximas seções para gerar os sinais caóticos.

III. *Reservoir Computing*

O RC adotado neste trabalho é uma ESN [14], formada por três partes principais. A Fig. 4 representa esquematicamente a estrutura do RC que é detalhada nesta seção.

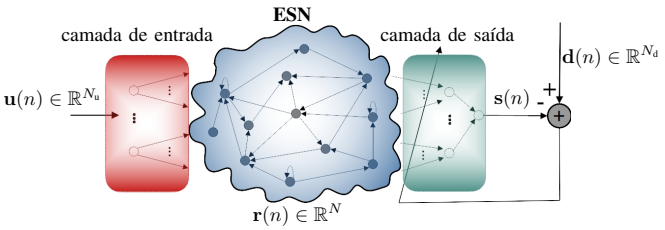

Fig. 4. Arquitetura de uma ESN, indicando os sinais $\mathbf{u}(n)$, $\mathbf{s}(n)$, $\mathbf{d}(n)$ e seu estado $\mathbf{r}(n)$. Adaptado de [8].

Dado um sinal de entrada $\mathbf{u}(n) \in \mathbb{R}^{N_\mathrm{u}}$ e um sinal desejado $\mathbf{d}(n) \in \mathbb{R}^{N_\mathrm{d}}$, onde $n = 1{,}2{,}\ldots{,}L$ e L é o numero de amostras de $\mathbf{d}(n)$ disponíveis para treinamento, a tarefa do RC é, de alguma forma, adaptar os pesos das conexões *reservoir*-saída, de modo que o sinal de saída $\mathbf{s}(n)$ seja tal que uma medida de erro $E(\mathbf{s},\mathbf{d})$ é minimizada [8]. Normalmente E é tomada como a raiz do erro quadrático médio (RMSE - *Root-Mean-Square Error*)

$$E\left(\mathbf{s},\mathbf{d}\right) = \frac{1}{N_\mathrm{d}} \sum_{k=1}^{N_\mathrm{d}} \sqrt{\frac{1}{L} \sum_{n=1}^{L} \left(s_\mathrm{k}(n) - d_\mathrm{k}(n)\right)^2}, \tag{2}$$

ou sua versão normalizada [18]. As subseções a seguir explicam de formada detalhada o funcionamento da camada de entrada, do *reservoir* e da camada de saída do RC.

A. Camada de Entrada

A camada de entrada é formada por N_u nós, e é ligada ao *reservoir* através de $N_\mathrm{u}N$ conexões, sendo N o número de nós do *reservoir*. Os pesos dessas conexões são as entradas w_{ij}^in reais da matriz $\mathbf{W}^\mathrm{in}$ $N \times N_\mathrm{u}$, comumente obtidas a partir de uma distribuição uniforme no intervalo $[-1,1]$ [14]. O papel da camada de entrada é realizar um pré-processamento do sinal $\mathbf{u}(n)$, tendo como saída o produto $\mathbf{W}^\mathrm{in}\mathbf{u}(n)$ [14].

B. Reservoir

O chamado *reservoir* é formado por N nós conectados entre si, e é ligado tanto a camada de entrada quanto à camada de saída. Os pesos das conexões internas do RC são as entradas w_{ij} reais da matriz $\mathbf{W}$ $N \times N$, e normalmente também obedecem a uma distribuição uniforme. Os nós internos são caracterizados por uma ativação $r_k(n) \in \mathbb{R}$, com $k = 1,2,\ldots,N$. Quando o produto $\mathbf{W}^\mathrm{in}\mathbf{u}(n)$ adentra no *reservoir*, cada um de seus N nós sofre uma ativação $r_k(n)$, que se propaga por toda a rede, através dos caminhos existentes graças as conexões internas entre os nós, definidas pela matriz $\mathbf{W}$. Dado um instante n, o *reservoir* é completamente definido pela matriz $\mathbf{W}$ e pelo vetor de ativações

$$\mathbf{r}(n) = \begin{bmatrix} r_1(1) & r_2(2) & \ldots & r_N(n) \end{bmatrix}^\mathrm{T}, \tag{3}$$

também chamado de vetor de estado. A saída do *reservoir* no instante n é uma função não linear de $\mathbf{W}^\mathrm{in}\mathbf{u}(n)$ e $\mathbf{W}\mathbf{r}(n)$ [8].

C. Camada de Saída

A camada de saída é formada por N_d nós e sua entrada é ligada ao *reservoir* por meio de $(N_\mathrm{u} + N)\,N_\mathrm{d}$ conexões. Os pesos w_{ij}^out reais dessas conexões, são determinados a partir de um sinal desejado $\mathbf{d}(n)$ e formam a matriz $\mathbf{W}^\mathrm{out}$ $N_\mathrm{d} \times (N_\mathrm{u} + N)$, que define a camada de saída, cujo sinal de saída $\mathbf{s}(n)$ num dado instante n é uma função não linear de $\mathbf{W}^\mathrm{out}$, $\mathbf{r}(n)$ e $\mathbf{u}(n)$.

D. Adaptação dos pesos

No RC tanto $\mathbf{W}^\mathrm{in}$ quanto $\mathbf{W}$ são geradas aleatoriamente na fase de aprendizagem, e não se alteram mais. Na publicação original sobre ESNs [14], recomenda-se gerar $\mathbf{W}$ esparsa, i.e. $\mathbf{W}$ com a maioria de suas entradas nulas, pois faz com que "haja um desacoplamento entre sub-redes internas, estimulando a dinâmica individual" [14]. Os elementos não nulos de $\mathbf{W}$ normalmente seguem uma distribuição uniforme simétrica, discreta bivariada ou gaussianana centrada em zero [18]. É comum gerar $\mathbf{W}^\mathrm{in}$ com suas entradas $w_{ij}^\mathrm{in} \in \mathbb{R}$ apresentando a mesma distribuição daquela dos elementos não nulos de $\mathbf{W}$. O vetor de estado $\mathbf{r}(n)$ é atualizado a partir da entrada neste mesmo instante e do estado no instante anterior, por meio de uma expansão não linear destes[1], através de um modelo com nós do tipo *leaky-integrator*[2] [14]

$$\mathbf{r}(n+1) = (1-\alpha)\,\mathbf{r}(n) + \mathbf{f}\left(\mathbf{W}^\mathrm{in}\mathbf{u}(n) + \mathbf{W}\mathbf{r}(n)\right), \tag{4}$$

sendo $\alpha \in [0,1]$ o parâmetro de *leakage* [19]. A escolha mais comum para a função $\mathbf{f}$ é $\tanh(\cdot)$, mas outras sigmoides, i.e. funções que apresentam um gráfico em formato próximo ao da letra s, também podem ser utilizadas [14].

É esperado que, graças a expansão não linear com memória (4), se obtenha um espaço $\mathbf{r}(n) \in \mathbb{R}^N$ rico o suficiente para que, a partir de uma regressão linear, a saída do RC

$$\mathbf{s}(n) = \mathbf{f}^\mathrm{out}\left(\mathbf{W}^\mathrm{out}\left[\mathbf{u}(n)|\mathbf{r}(n)\right]\right), \tag{5}$$

seja suficientemente próxima de $\mathbf{d}(n)$. Para isso, é fundamental que se tenha um *reservoir* com um número de nós suficientemente grande, o que se traduz, basicamente, em $N \gg N_\mathrm{u}$ [8]. Na expressão (5), $\cdot|\cdot$ indica a concatenação vertical de vetores e $\mathbf{f}^\mathrm{out}$ é normalmente a função identidade [8].

Os pesos das conexões *reservoir*-saída w_{ij}^out, são ajustados de modo a minimizar o RMSE (2). A Regressão de Ridge [20] é o método mais comum para este fim [18].

Antes de iniciar o período de treinamento para $n = 1,2,\ldots,L$, onde L é o número de amostras de $\mathbf{d}(n)$ disponíveis para o RC, é comum e recomendável que alguns valores de $\mathbf{r}(n)$ sejam descartados para mitigar efeitos de transientes inicias [8]. Em geral, inicia-se com $\mathbf{r}(-L_\mathrm{transiente}+1) = \mathbf{0}$ e, por meio de sucessivas aplicações de (4), obtém-se $\mathbf{r}(0)$. A partir de então, inicia-se efetivamente o período de treinamento para $n = 1,2,\ldots,L$, onde as ativações $\mathbf{r}(1),\mathbf{r}(2),\ldots,\mathbf{r}(L)$ são armazenadas na matriz $N \times L$

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}(1) & \mathbf{r}(2) & \ldots & \mathbf{r}(L) \end{bmatrix}. \tag{6}$$

Ou seja, os $L_\mathrm{transiente}$ valores de iniciais de $\mathbf{r}(n)$ são descartados. Considerando-se que $N \gg N_\mathrm{u}$ [8], a Regressão de Ridge leva a solução

$$\mathbf{W}^\mathrm{out} = \mathbf{D}\mathbf{R}^\mathrm{T}\left(\mathbf{R}\mathbf{R}^\mathrm{T} + \beta\mathbf{I}\right)^{-1}, \tag{7}$$

sendo $\beta \in \mathbb{R}$ o coeficiente de regularização, $\mathbf{I}$ a matriz identidade de ordem N e

$$\mathbf{D} = \begin{bmatrix} \mathbf{d}(1) & \mathbf{d}(2) & \ldots & \mathbf{d}(L) \end{bmatrix} \tag{8}$$

uma matriz $N_\mathrm{d} \times L$, cujas colunas correspondem ao sinal desejado em cada instante do treinamento. O coeficiente de regularização tem o papel de fazer com que a Regressão de Ridge, além de minimizar o RMSE, não permita que $\mathbf{W}^\mathrm{out}$ tenha pesos muito altos, o que pode gerar problemas em sistemas realimentados [18].

Para os instantes $n = L+1$ em diante, de posse de (7), a saída (5) é uma estimativa de $\mathbf{d}(n)$.

O comprimento de treinamento L, o número de nós internos N, o parâmetro de *leakage* α e o raio espectral da matriz de pesos recorrentes $\mathbf{W}$, doravante denotado por λ, influenciam diretamente no desempenho do RC [18]. É comum ajustá-los manualmente, variando-se um deles enquanto os outros permanecem fixos e observando o valor que otimiza o desempenho do RC, que pode ser quantificado por meio, por exemplo, da relação sinal-ruído (SNR - *Signal-To-Noise Ratio*) em sua saída.

Cabe destacar que o comprimento do intervalo escolhido para gerar $\mathbf{W}$ é irrelevante, uma vez que λ será ajustado, o

[1] Em alguns casos pode-se ainda adicionar conexões da saída do *reservoir* para sua entrada [14], o que acarreta na presença de mais um termo na expansão (4). Neste trabalho, considera-se que não há essas conexões.

[2] ESNs que empregam nós desse tipo são conhecidas por *leaky-integrator* ESNs (LI-ESNs) [8].

que acaba por ajustar também este comprimento. Na grande maioria das situações práticas, $\lambda < 1$ garante que a dependência do estado $\mathbf{r}(n)$ em relação aos estados passados e as entradas passadas se desvaneça gradualmente com as iterações [14], propriedade conhecida por *echo state* e fundamental para que o RC funcione [8].

Essa sessão é encerrada com as instruções gerais para gerar e otimizar um RC,

- Gerar $\mathbf{W}^{\text{in}}$ e $\mathbf{W}$ aleatoriamente, com as respectivas entradas seguindo uma distribuição uniforme contínua em $[-1, 1]$ por exemplo;
- Inicializar $\mathbf{r}(n) = \mathbf{0}$ em (3);
- Utilizar (4) por L sucessivas vezes para obter $\mathbf{R}$ definida em (6);
- Calcular $\mathbf{W}^{\text{out}}$ por meio de uma regressão linear, como em (7);
- Ajustar os principais parâmetros globais, i.e. L, N, λ e α de forma a maximizar a SNR na saída.

IV. Metodologia

Para as simulações, considerou-se $\mathbf{f} = \tanh(\cdot)$ e $\mathbf{f}^{\text{out}}$ como a função identidade, de modo que (4) e (5) se tornam

$$\mathbf{r}(n) = (1-\alpha)\,\mathbf{r}(n-1) + \alpha \tanh\left(\mathbf{W}^{\text{in}} u(n) + \mathbf{W}\mathbf{r}(n-1)\right), \quad (9)$$

e

$$s(n) = \mathbf{W}^{\text{out}}\left[u(n)|\mathbf{r}(n)\right]. \quad (10)$$

Não se utiliza mais o negrito em (9) para indicar o sinal de entrada do RC como vetor pois $N_{\text{u}} = 1$, uma vez que

$$u(n) = x(n) + w(n), \quad (11)$$

sendo $w(n)$ ruído AWGN e $x(n)$ a componente caótica gerada a partir do mapa de Hénon [5] definido em (1) com condições iniciais uniformemente distribuídas no intervalo aberto $(0, 1)$ e parâmetros $a = 1{,}4$ e $b = 0{,}3$. Como assume-se que amostras de $x(n)$ estão disponíveis para o treinamento do RC, ou seja, $d(n) = x(n)$ para $n = 1,2,\ldots,L$, segue que $N_{\text{d}} = 1$ e por isso também não se usa o negrito para indicar o sinal desejado como vetor. Cabe destacar que o sinal de entrada (11) foi normalizado para ter média 0 e variância 1, como recomendado em [18].

Inspirado em [4], a distribuição uniforme foi a escolhida para gerar as matrizes $\mathbf{W}^{\text{in}}$ e $\mathbf{W}$. Apesar da publicação original [14] recomendar a utilização de $\mathbf{W}$ esparsa, a experiência prática mostrou que a esparsividade de $\mathbf{W}$ tem pequena influência sobre o desempenho do RC [18]. Sendo assim, todas as entradas de $\mathbf{W}$ e também as de $\mathbf{W}^{\text{in}}$ foram obtidas a partir de uma distribuição uniforme em $[-1, 1]$, conforme [4].

O ajuste dos parâmetros globais (L, N, λ, α) deu-se por meio de simulações variando-se um deles e mantendo-se os demais, bem como a SNR de entrada, denotada por SNR_{in} fixos. Para cada parâmetro, o valor escolhido foi aquele que resultou na SNR de saída de maior magnitude, calculada como

$$\text{SNR}_{\text{out}} = \frac{\sum_{n=L+1}^{L+T} \hat{x}^2(n)}{\sum_{n=L+1}^{L+T} \left(\hat{x}(n) - x(n)\right)^2} \quad (12)$$

em que T, o comprimento de teste, indica por quantos instantes de tempo o RC foi testado após ter-se obtido $\mathbf{W}^{\text{out}}$, e $\hat{x}(n)$ é a saída do RC para $n = L+1,\ldots,L+T$, onde almeja-se obter uma estimativa de $x(n)$. Daí o uso da notação $\hat{x}(n)$ para indicar a saída do RC para $n > L$, ao invés de $s(n)$.

Em cada simulação, SNR_{out} foi obtida da média de 100 realizações, cada uma considerando $x(n)$, $w(n)$, $\mathbf{W}^{\text{in}}$ e $\mathbf{W}$ distintos. As curvas SNR_{out}-*versus*-Parâmetro obtidas estão apresentadas na Figura 5.

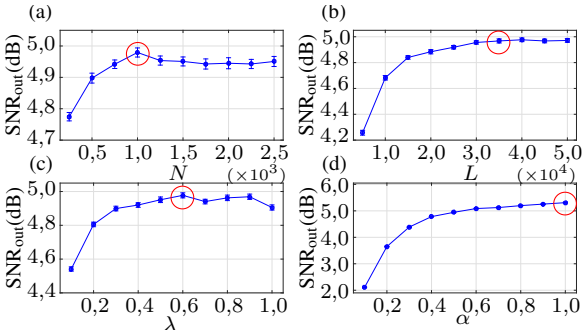

Fig. 5. SNR_{out} em dB para $\text{SNR}_{\text{in}} = 2{,}5$dB e $T = 5000$ em função do (a): Comprimento de treinamento. (b): Parâmetro de *leakage*. (c): Raio espectral. (d): Numero de nós. Com exceção do parâmetro que varia, os outros mantiveram-se fixos em $L = 30000$, $N = 1000$, $\lambda = 0{,}5$ e $\alpha = 0{,}5$. Os valores dos parâmetros ótimos estão ressaltados nos gráficos.

Em (a) está ilustrado como o número de nós do RC afeta SNR_{out}. Nota-se que para N com magnitude superior a 1000, a curva apresenta uma saturação [4], não ocorrendo mais aumento de SNR. Em (b) está exposto como SNR_{out} varia em função comprimento de treinamento. Percebe-se que para valores de L superiores a 35000, não há melhorias substanciais em SNR_{out}. Por fim, (c) e (d) mostram a variação de SNR_{out} em função do raio espectral e do parâmetro de *leakage*, respectivamente. De (c) é possível inferir que a SNR de saída é máxima para $\lambda = 0{,}6$, enquanto que de (d) constata-se que $\alpha = 1$ é o valor ótimo.

É interessante notar que, $\alpha = 1$ implica que o RC proposto tem melhor desempenho sem *leaky-integration* [18], e (9) assume a forma simplificada

$$\mathbf{r}(n) = \tanh\left(\mathbf{W}^{\text{in}} u(n) + \mathbf{W}\mathbf{r}(n-1)\right). \quad (13)$$

Neste ponto, todas as instruções listadas ao final da sessão III foram executadas. Para verificar a qualidade do ajuste dos parâmetros (N,L,λ,α) realizado, é possível fazer uma simulação variando-se SNR_{in} e observando-se qual a SNR_{out} correspondente. É isto que será feito, com os parâmetros $L = 35000$, $N = 1000$, $\lambda = 0{,}6$ e $\alpha = 1$ obtidos, nas simulações computacionais da seção seguinte.

V. Resultados

Para o conjunto de parâmetros obtidos através de ajuste manual descrito na seção anterior, foram feitas simulações com 1000 realizações cada, a fim de traçar a curva SNR_{out} *versus* SNR_{in} e comparar ambos sinais de entrada $u(n)$ e de saída $\hat{x}(n)$ com a componente caótica $x(n)$.

Na Figura 6 (a) é possível visualizar a componente caótica em comparação com o sinal de entrada e com a sua estimativa fornecida pelo RC, enquanto que na Figura 6 (b) é apresentado o erro em módulo na saída e na entrada do RC. Nota-se que

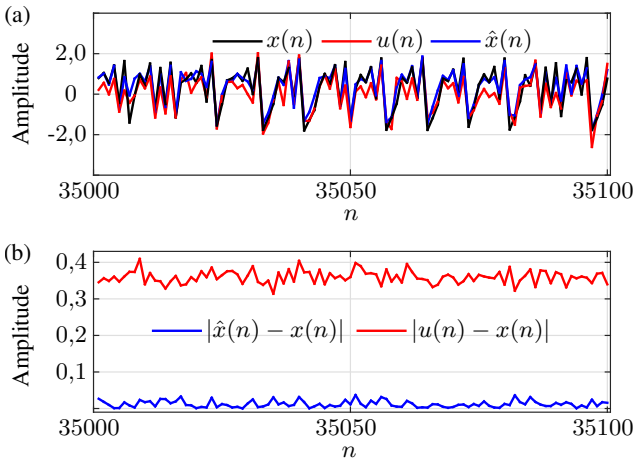

Fig. 6. (a): Componente caótica, entrada e saída do RC para $\text{SNR}_{\text{in}} = 5{,}0\text{dB}$. (b): Erro médio na saída e na entrada do RC para $\text{SNR}_{\text{in}} = 2{,}5\text{dB}$.

o RC foi capaz de mitigar o ruído sobre $x(n)$, uma vez que o módulo do erro na saída foi quase sempre menor que o da entrada. Ademais, da Fig. 6 (a) percebe-se visualmente a melhora. Por fim, na Figura 7 é apresentada a curva de SNR_{out} em função da SNR_{in} em (a) e o ganho em (b), principais resultados obtidos neste trabalho. Da Fig. (7) (a) nota-se que

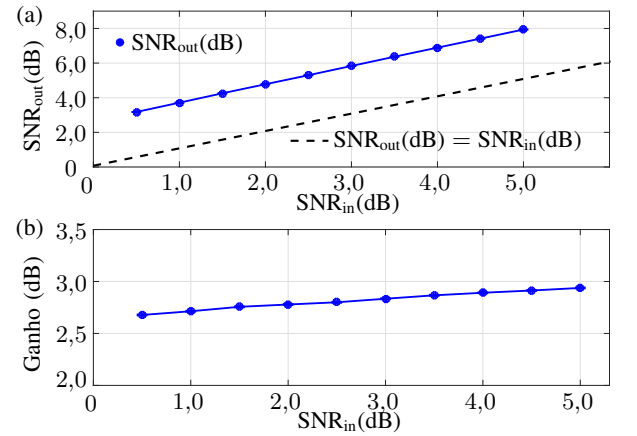

Fig. 7. SNR de saída em função da SNR de entrada do RC.

os resultados são satisfatórios, pois a SNR na saída foi sempre maior que a SNR na entrada do RC, mesmo quando esta é relativamente baixa. Ainda, quanto maior a SNR na entrada, maior o ganho, conforme exposto na Fig. (7) (b).

VI. Conclusões

Neste trabalho foi investigado o desempenho de um RC, através de uma ESN, para filtragem do ruído que corrompe uma componente caótica de Hénon. O RC mostrou ser capaz de melhorar a SNR num sistema que apesar de simples, pode servir de base para aplicações práticas de maior interesse. Por ser relativamente recente, *Reservoir Computing* apresenta muitos aspectos e características a ser melhor compreendidas. Variações da ideia original vêm sendo propostas [8] e seria interessante observar, por exemplo, o compartamento do RC aqui estudado para diferentes distribuições de $\mathbf{W}$ e até mesmo para $\lambda > 1$.

Como sequência deste trabalho, pretende-se analisar o emprego de um RC em cenários mais desafiantes, como o de um sistema de comunicação empregando técnicas de modulação baseada em caos.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

[1] Wornell G. W. Isabelle S. H. Oppenheim, A. V. and K. M. Cuomo. Signal processing in the context of chaotic signals. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[2] Marozas V. Sörnmo L. Petrėnas, A. and A. Lukoševičius. Reservoir computing for extraction of low amplitude atrial activity in atrial fibrillation. In *Proceedings of Computing in Cardiology*, volume 3, 2012.

[3] Stone S. Birkholz P. Steiner, P. and A. Jalalvand. Multipitch tracking in music signals using echo state networks. In *Proceedings of the 28th European Signal Processing Conference (EUSIPCO)*, 2021.

[4] Girvan M. Ott E. Krishnagopal, S. and B. R. Hunt. Separation of chaotic signals by reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(2), 2020.

[5] Prexl J. Linkmann M. Lellep, M. and B Eckhardt. Using machine learning to predict extreme events in the hénon map. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1), 2020.

[6] L. V. Fausett. *Fundamentals of neural networks: Architectures, algorithms, and applications*. Prentice-Hall, New York, 1994.

[7] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach. Technical Report 159, German National Research Center for Information Technology.

[8] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*.

[9] K. Doya. Bifurcations in the learning of recurrent neural networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1992.

[10] Simard P. Bengio, Y. and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[11] Hinton G., Deng L., Yu D., Dahl G.E E., Mohamed A., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Sainath T.N., and Kingsbury B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*.

[12] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55:645–657, 2017.

[13] Natschläger T. Maass, W. and H. Markram. Learning long-term dependencies with gradient descent is difficult. *Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. Neural Computation*, 14(11):2531–2560, 2002.

[14] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report 148, German National Research Center for Information Technology, 2001.

[15] Kathleen T. Alligood and Tim. Sauer. *Chaos: An Introduction to Dynamical Systems*. Springer, New York, 1997.

[16] Andrei D. Linde. Eternally existing selfreproducing chaotic inflationary universe. *Phys. Lett. B*, 175:395–400, 1986.

[17] Erich Harth. Order and chaos in neural systems: An approach to the dynamics of higher brain functions. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:782–789, 1983.

[18] M. Lukoševičius. *A Practical Guide to Applying Echo State Networks*, volume 7700 of *Lecture Notes in Computer Science*. Springer, 2012.

[19] Herbert Jaeger Udo Siewert Mantas Lukosevicius, Dan Popovici. Time warping invariant echo state networks. Technical report.

[20] Norman R. Draper and Harry Smith. *Applied Regression Analysis*. Wiley Series in Probability and Statistics, New York, 3 edition, 1998.