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.

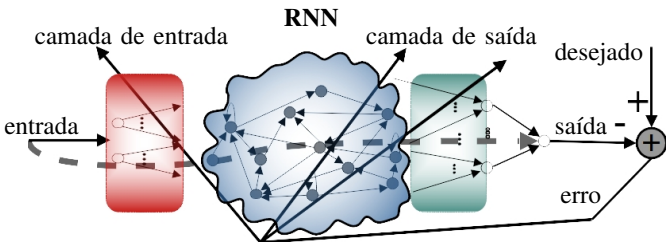

Fig. 1. Arquitetura de uma RNN, em que todas as conexões (entrada-RNN, internas e RNN-saída) são treinadas. Adaptado de [8].

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.

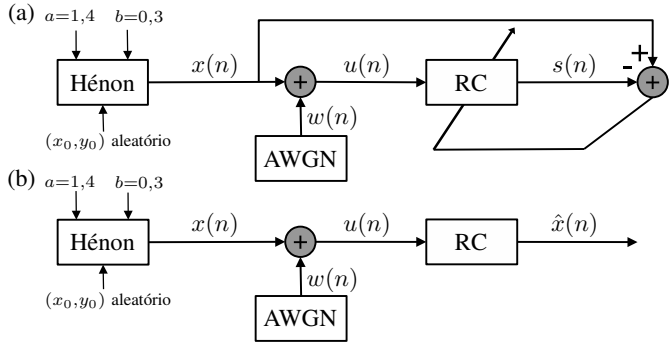

Fig. 2. Diagrama em blocos do problema estudado durante o (a): período de treinamento. (b): período de teste do RC.

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

$$\begin{aligned} x(n) &= a - (x(n-1))^2 + by(n-1), \\ y(n) &= x(n-1). \end{aligned} \tag{1}$$

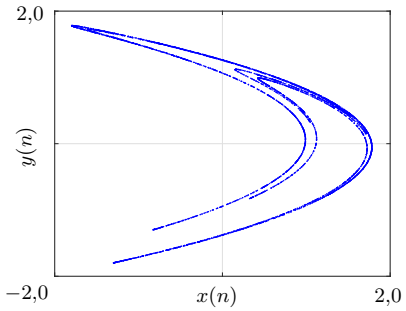

Fig. 3. Órbita caótica de Hénon obtida com $a = 1{,}4$, $b = 0{,}3$ e condições iniciais aleatórias.

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.

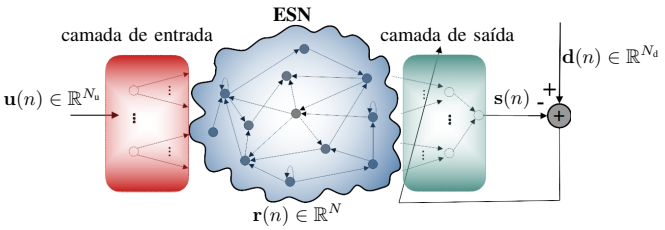

Fig. 4. Arquitetura de uma ESN, indicando os sinais $\mathbf{u}(n)$, $\mathbf{s}(n)$, $\mathbf{d}(n)$ e seu estado $\mathbf{r}(n)$. Adaptado de [8].

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.

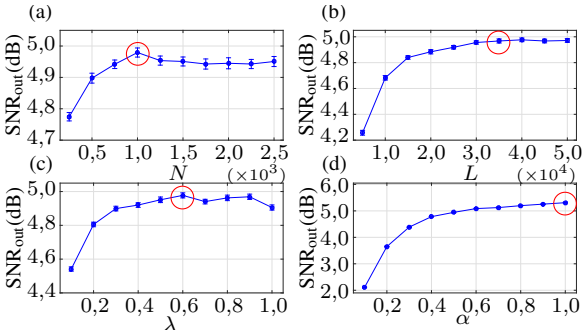

Fig. 5. SNR_{out} em dB para $\text{SNR}_{\text{in}} = 2{,}5$dB e $T = 5000$ em função do (a): Comprimento de treinamento. (b): Parâmetro de *leakage*. (c): Raio espectral. (d): Numero de nós. Com exceção do parâmetro que varia, os outros mantiveram-se fixos em $L = 30000$, $N = 1000$, $\lambda = 0{,}5$ 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

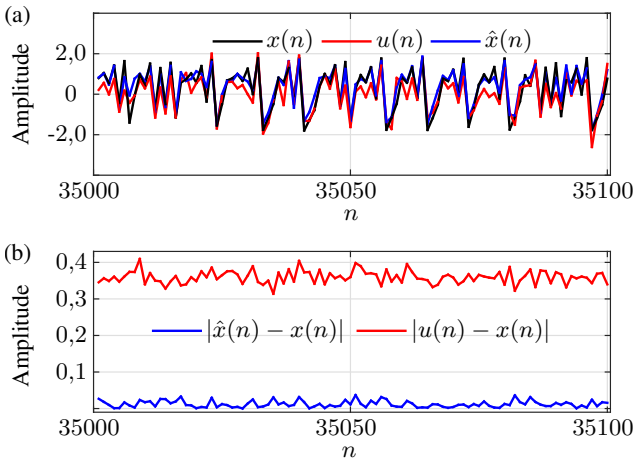

Fig. 6. (a): Componente caótica, entrada e saída do RC para $\text{SNR}_{\text{in}} = 5{,}0\text{dB}$. (b): Erro médio na saída e na entrada do RC para $\text{SNR}_{\text{in}} = 2{,}5\text{dB}$.

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

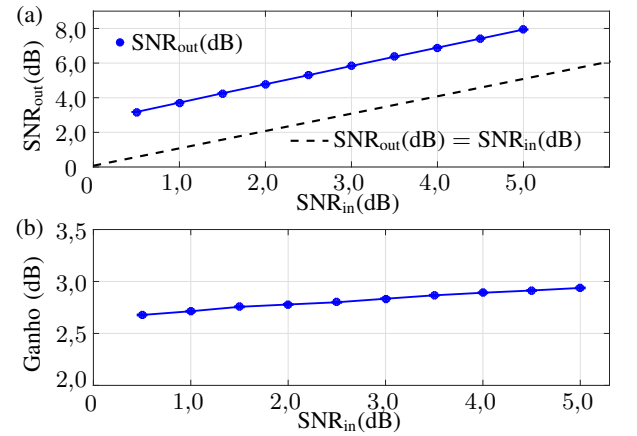

Fig. 7. SNR de saída em função da SNR de entrada do RC.

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.