Universidade Federal do Maranhão

Programa de Pós-Graduação em Engenharia Elétrica

Automação e Controle

Thiago Silva Fernandes

Estimação de Componentes Harmônicos em Sistemas Elétricos de Potência Utilizando Redes Neurais Artificiais Tensoriais

São Luís, MA

2021

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

SILVA FERNANDES, THIAGO.
Estimação de Componentes Harmônicos em Sistemas Elétricos de Potência Utilizando Redes Neurais Artificiais Tensoriais / THIAGO SILVA FERNANDES. - 2021.
94 p.

Orientador(a): FRANCISCO DAS CHAGAS DE SOUZA.
Dissertação (Mestrado) - Programa de Pós-graduação em Engenharia Elétrica/ccet, Universidade Federal do Maranhão, São Luís, 2021.

1. Algoritmo de Widrow-Hoff. 2. Estimação de harmônicas. 3. Rede neural artificial. 4. Separabilidade de operadores lineares. 5. Tensores. I. DAS CHAGAS DE SOUZA, FRANCISCO. II. Título.

Thiago Silva Fernandes

Estimação de Componentes Harmônicos em Sistemas Elétricos de Potência Utilizando Redes Neurais Artificiais Tensoriais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Maranhão como requisito para obtenção do título de Mestre em Engenharia Elétrica na área de concentração de Automação e Controle.

Data da aprovação São Luís, MA, 26 de Novembro de 2021.

Prof. Dr. Franciso das Chagas de Souza - UFMA
Orientador

Prof. Dr. João Viana da Fonseca Neto - UFMA
Membro da Banca Examinadora

Prof. Dr. Walbermak Marques dos Santos - UFES
Membro da Banca Examinadora

São Luís, MA
2021

Dedico este trabalho à Maria Irene, minha mãe, a José Carlos, meu pai, à Lais e Thais, minhas amadas irmãs, todas pessoas de fundamental importância na minha vida.

Agradecimentos

Deixo aqui os meus sinceros agradecimentos àqueles que direta ou indiretamente contribuíram para a realização deste trabalho de pesquisa, em especial: Ao Pr. Dr. Francisco das Chagas de Souza pela oportunidade de desenvolver este trabalho e pelos ensinamentos dados durante todas as etapas da pesquisa, sendo de grande contribuição para o meu desenvolvimento profissional.

Aos meus pais, Maria Irene e Silva Fernandes e José Carlos Viana Fernandes, e as minhas irmãs, Lais Silva Fernandes e Thais Silva Fernandes, pelo apoio, motivação e dedicação incondicional.

Aos meus colegas de laboratório: Marcela, Raimundo, Willians e, principalmente, Jadyna, cuja ajuda na edição de figuras e formatação de texto foi de fundamental importância na conclusão deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), à Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão (FAPEMA) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro no desenvolvimento do projeto de pesquisa.

A Deus, pelas oportunidades que me foram dadas para a conclusão deste trabalho de pesquisa.

Resumo

A regra de Widrow-Hoff é o método mais simples e utilizado para atualizar os pesos sinápticos de uma rede neural artificial. Isto deve-se a sua simplicidade de aplicação e robustez paramétrica, propriedades de interesse em diversas aplicações. Entretanto, em estruturas com elevados números de parâmetros, este algoritmo apresenta elevado custo computacional e baixa velocidade de aprendizagem. Neste trabalho, uma nova regra de atualização dos pesos sinápticos de uma rede neural é apresentada. O método proposto utiliza os conceitos de separabilidade de operadores lineares, uma propriedade típica de tensores. Uma vez que essa propriedade é dada, um algoritmo do tipo gradiente estocástico pode ser derivado com aumento significativo da velocidade de aprendizagem e redução do custo computacional quando comparado ao algoritmo de Widrow-Hoff. Á medida que o número de parâmetros da rede neural aumenta, a diferença de desempenho entres os dois algoritmos é potencializada. Outra propriedade relevante do método proposto é que a sua velocidade de aprendizagem é inversamente proporcional ao custo computacional. Devido ao grande impacto gerado pelo crescimento de cargas não lineares no sistema elétrico de potência, que causam distorções na forma de onda de tensão e corrente, a metodologia proposta é aplicada em problemas de estimação de harmônicas.

Palavras-chave: Algoritmo de Widrow-Hoff, rede neural artificial, separabilidade de operadores lineares, tensores, gradiente estocástico, estimação de harmônicas.

Abstract

The Widrow-Hoff rule is the simplest and most used method to update the synaptic weights of an artificial neural network. This is due to its simplicity of application and parametric robustness, properties of interest in several applications. However, in structures with high numbers of parameters, this algorithm has high computational cost and low speed of convergence. In this work, a new rule for updating the synaptic weights of a neural network is presented. The proposed method uses the concepts of linear operator separability, a typical property of tensors. Once this property is given, a stochastic gradient algorithm can be derived with a significant increase in the learning speed and a reduction in the computational cost when compared to the Widrow-Hoff algorithm. As the number of neural network parameters increases, the performance difference between the two algorithms is enhanced. Another relevant property of the proposed method is that its learning speed is inversely proportional to the computational cost. Due to the great impact generated by the growth of non-linear loads in the electrical power system, which cause distortions in the voltage and current waveforms, the proposed methodology is applied in harmonic estimation problems.

Keywords:Widrow-Hoff algorithm, artificial neural network, separability of linear operators, tensors, stochastic gradient, identification of harmonics.

Lista de abreviaturas e siglas

LMS	*Least Mean Square*
NLMS	*Normalized Least Means Square*
GNGD	*Generalized Normalized Gradient Descent*
DHT	Distorção Harmônica Total
FFT	*Fast Fourier Transform*
PLL	*Phase Locked Loop*
FIR	*Finite Impulse Response*
IIR	*Infinite Impulse Response*
MSE	*Mean Square Error*
SD	*Steepest Descent*
PNLMS	*Proportionate Normalized Least Mean Square*
IPNLMS	*Improved Proportionate Normalized Least Mean Square*
IA	Inteligência Artificial
RNA	Redes neurais artificiais
Adaline	*Adaptive Linear Element*

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Sumário

1 Introdução

O termo "filtro"é utilizado para descrever um dispositivo físico (hardware) ou computacional (software) cuja a função é extrair ou aprimorar informações de interesse contidas em um sinal (HAYKIN, 2002, p.1). Quando os parâmetros do filtro se ajustam de forma a responder a algum fenômeno que está ocorrendo ao seu entorno, afirma-se que o filtro é adaptativo (FARHANG-BOROUJENY, 2013, p.1). Esse processo de ajuste é feito a partir do algoritmo adaptativo, que é responsável por alterar os coeficientes para que o filtro possa operar em um ambiente desconhecido e mutável (LEE; GAN; KUO, 2009, p.1). Devido a capacidade de auto ajuste, os filtro adaptativos são empregados em diferentes áreas, como por exemplo, identificação de sistema, cancelamento de eco, cancelamento de ruído, equalização, controle, antenas inteligentes, processamento de fala, compressão de dados, radar, sonar, biomedicina, sismologia, etc (VEGA; REY, 2012, p.1).

A configuração geral de um ambiente de filtragem adaptativa é mostrada na Figura 1.1, onde k é o índice temporal, $\mathbf{x}(k)$ é a entrada aplicada ao filtro, $\mathbf{y}(k)$ é a saída do filtro, $w(k)$ representa os coeficientes do filtro e $\mathbf{d}(k)$ é o sinal desejado. O sinal de erro $\mathbf{e}(k)$, calculado por $\mathbf{d}(k) - \mathbf{y}(k)$, é utilizado para formar uma função de desempenho (ou objetivo) que é exigida pelo algoritmo adaptativo a fim de determinar a atualização apropriada dos coeficientes do filtro (DINIZ, 2008, p.2).

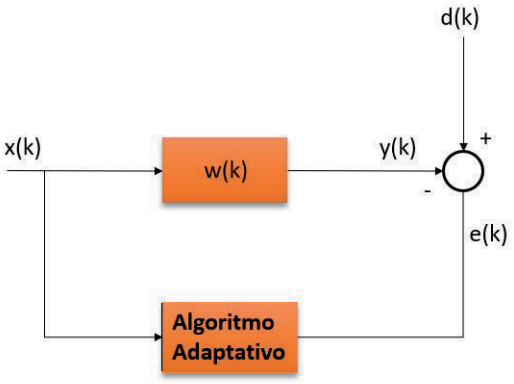


Figura 1.1 – Esquema de filtragem adaptativa.

O algoritmo LMS (*Least Mean Square*) é o algoritmo adaptativo mais básico e o mais utilizado em aplicações que envolvem filtragem adaptativa linear (HAYKIN, 2002, p.365) (MANOLAKIS; INGLE; KOGON, 2005). Outro algoritmo amplamente utilizado em diversos problemas de filtragem é o NLMS (*Normalized Least Means Square*), que pode ser visto como uma versão modificada do algoritmo LMS, onde a variação da potência do sinal de entrada é considera na sua formulação (SCHILLING; HARRIS, 2011, p.669). O algoritmo GNGD (Generalized Normalized Gradient Descent) representa uma

extensão do NLMS, no qual é adicionado um parâmetro de regularização que é atualizado através de um gradiente estocástico ao longo do tempo (MANDIC, 2004). Em problemas de filtragem adaptativa, esses algoritmos apresentam um desempenho pobre quando aplicados em problemas que envolvem plantas esparsas (RUPP; SCHWARZ, 2015). Isto ocorre devido aos algoritmos não levarem em consideração as características deste tipo de planta. Quando utilizando como regra de atualização dos pesos sinápticos de uma rede neural, apresentam baixa taxa de aprendizagem e alto custo computacional em aplicações com elevado número de nós em sua estrutura. Para contornar estes problemas, a separabilidade de operadores lineares, uma propriedade típica de tensores, é utilizada como ferramenta para derivar um algoritmo do tipo gradiente estocástico, resultando em um aumento significativo da taxa de aprendizagem e diminuição do custo computacional (RUPP; SCHWARZ, 2015). Para situar o leitor sobre os temas abordados nesse trabalho, a seguir são apresentados conceitos básico de plantas esparsas, com ênfase na estimação de harmônicas em sistemas elétricos de potência, e uma breve introdução a tensores.

1.1 Plantas esparsas

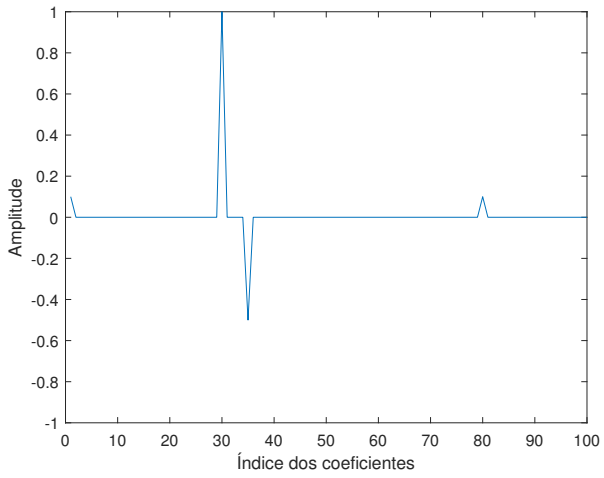
Uma planta é dita esparsa se a maioria dos seus coeficientes tem magnitude nula (ou próxima a zero) e apenas alguns poucos coeficientes apresentam magnitudes significativas (SOUZA et al., 2009). O grau de esparsidade de uma planta pode ser calculado a partir de (PALEOLOGU; BENESTY; CIOCHINA, 2010)

$$s(\mathbf{p}) = \frac{N}{N - \sqrt{N}}\left(1 - \frac{\| \mathbf{p} \|_1}{\sqrt{N} \| \mathbf{p} \|_2}\right). \tag{1.1}$$

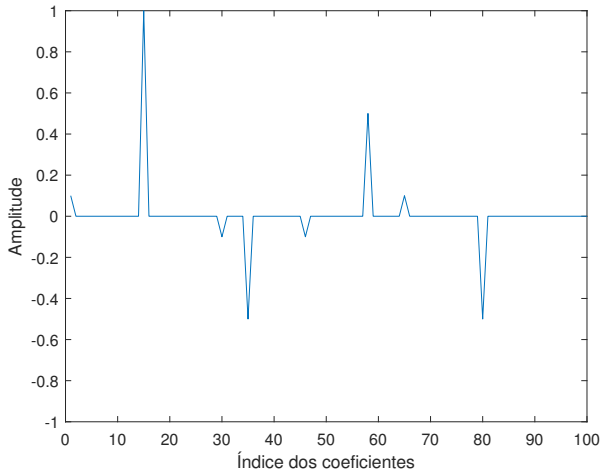
sendo o vetor de coeficientes da planta $\mathbf{p} = [\mathbf{p}_1 \ , \mathbf{p}_2 \ , \mathbf{p}_3 \ , ..., \mathbf{p}_N]^T$ definido em um dado subespaço de dimensão finita $\mathbb{R}^N$. O sobrescrito T denota a transposta de um vetor ou matriz. Os parâmetros $\| \mathbf{p} \|_1$ e $\| \mathbf{p} \|_2$ são, respectivamente, a norma-1 e norma-2 do vetor de coeficientes da planta $\mathbf{p}$. Tem-se também que $0 \leq s(\mathbf{p}) \leq 1$. Quanto mais $s(\mathbf{p})$ se aproxima de 1, maior é a medida de esparsidade da planta. A variável N indica a dimensão do vetor $\mathbf{p}$.

Aplicações envolvendo plantas esparsas são encontradas em diversas áreas, tais como estimação de harmônicas em sistemas de potência (ABDOLLAHI et al., 2012), cancelamento de eco em telecomunicações (LOGANATHAN; KHONG; NAYLOR, 2009), canais de comunicação de múltiplos percursos (BAJWA et al., 2010) e identificação de eventos sísmicos (GABARDA; CRISTÓBAL, 2010). A titulo de exemplo, a Figura 1.2 mostra a disposição dos coeficientes de duas plantas esparsas com N=100 coeficientes. Na Figura 1.2(a), tem-se uma planta cujo grau de esparsidade é s($\mathbf{p}$) = 0.9435, onde os coeficientes ativos da planta estão localizados nas posições [1, 30, 35, 80] com suas respectivas magnitudes iguais a [0.1, 1, -0.5, 0.1]. A Figura 1.2(b), refere-se a uma planta com grau de esparsidade é s($\mathbf{p}$) = 0.8703, sendo os coeficientes ativos da planta localizados

nas posições [1, 15, 30, 35, 46, 58, 65, 80] com suas respectivas magnitudes iguais a [0.1, 1, -0.1, -0.5, -0.1, 0.5, 0.1, -0.5].



(a)



(b)

Figura 1.2 – Exemplos de plantas esparsas com N=100. (a) Planta esparsa s($\mathbf{p}$) = 0.9435. (b) Planta esparsa s($\mathbf{p}$) = 0.8703

Como mencionado acima, o conceito de esparsidade é de fundamental importância na estimação de componentes harmônicas em sistema elétrico de potência. Este assunto será tratado mais detalhadamente a seguir.

1.2 Componentes harmônicas em sistemas elétricos de potência

Qualidade de energia elétrica é um termo que se refere à manutenção da quase forma sinusoidal de onda das tensões e correntes do barramento de distribuição de energia em magnitude e frequência. Pode ser usado também para expressar qualidade de

tensão, qualidade de corrente, confiabilidade de serviço, qualidade da fonte de alimentação, etc (CHATTOPADHYAY; MITRA; SENGUPTA, 2011). Um dos aspectos de maior preocupação nessa área é o surgimento das chamadas componentes harmônicas no sistema de potência (RAY; SUBUDHI, 2012).

Devido ao crescente uso de cargas não lineares em sistemas de energia, principalmente equipamentos baseados eletrônica de potência, fontes de alimentação ininterruptas, fornos a arco e acionamentos de motor controlados, distorções periódicas nas formas de onda de corrente e tensão tornaram-se cada vez mais frequentes (YILMAZ; ALKAN; ASYALI, 2008). Além disso, o amplo uso de novas tecnologias para geração de energia, em que são fortementes baseados em conversores de potência, também contribui para a crescente preocupação por melhores estimativas para garantir a qualidade da energia (BARROS; PÉREZ, 2006) e (RAY; SUBUDHI, 2012).

Para medir o grau de distorção provocado pelas componentes utiliza-se o DHT (Distorção Harmônica Total) que quantifica o quanto uma onda resultante está distorcida em relação a componente fundamental. Essa métrica é calculada conforme abaixo

$$DHT = \frac{\sqrt{\sum_{h=2}^{N} V_h^2}}{V_1,} \tag{1.2}$$

onde V_1 é a componente fundamental e N representa o número total de componentes harmônicas presentes no sinal.

Visando melhor ilustrar os efeitos das componentes harmônicas na rede elétrica, considere um sinal típico formado pela componente fundamental de 60 Hz e componentes harmônicas de ordem 3, 5 e 7, conforme Equação (1.3). A resolução do espectro é de 1 Hz e a faixa de frequência está localizada entre 0 e 540 Hz.

$$s(t) = 220\sqrt{2}sen(wt) + 150sen(3wt + \frac{\pi}{6}) + 120sen(5wt + \frac{\pi}{3}) + 10sen(7wt + \frac{\pi}{3}.) \tag{1.3}$$

A partir da análise do espectro gerado por (1.3), pode-se perceber a característica esparsa de sinais típicos envolvidos na estimação de componentes harmônicas de sistemas de potência. A maioria da faixa de espectro do sinal tem amplitude igual a zero, sendo poucas frequências com amplitude diferente de zero. Analisando (1.3), obtêm-se os seguintes parâmetros:

1 Distorção Harmônica Total (DHT)

$$DHT = \frac{\sqrt{150^2+120^2+10^2}}{311} = 1.61.$$

2 Valor Eficaz

$$RMS = 253.9 \text{ V}.$$

3 Grau de Esparsidade

$$s(p) = 0.9712.$$

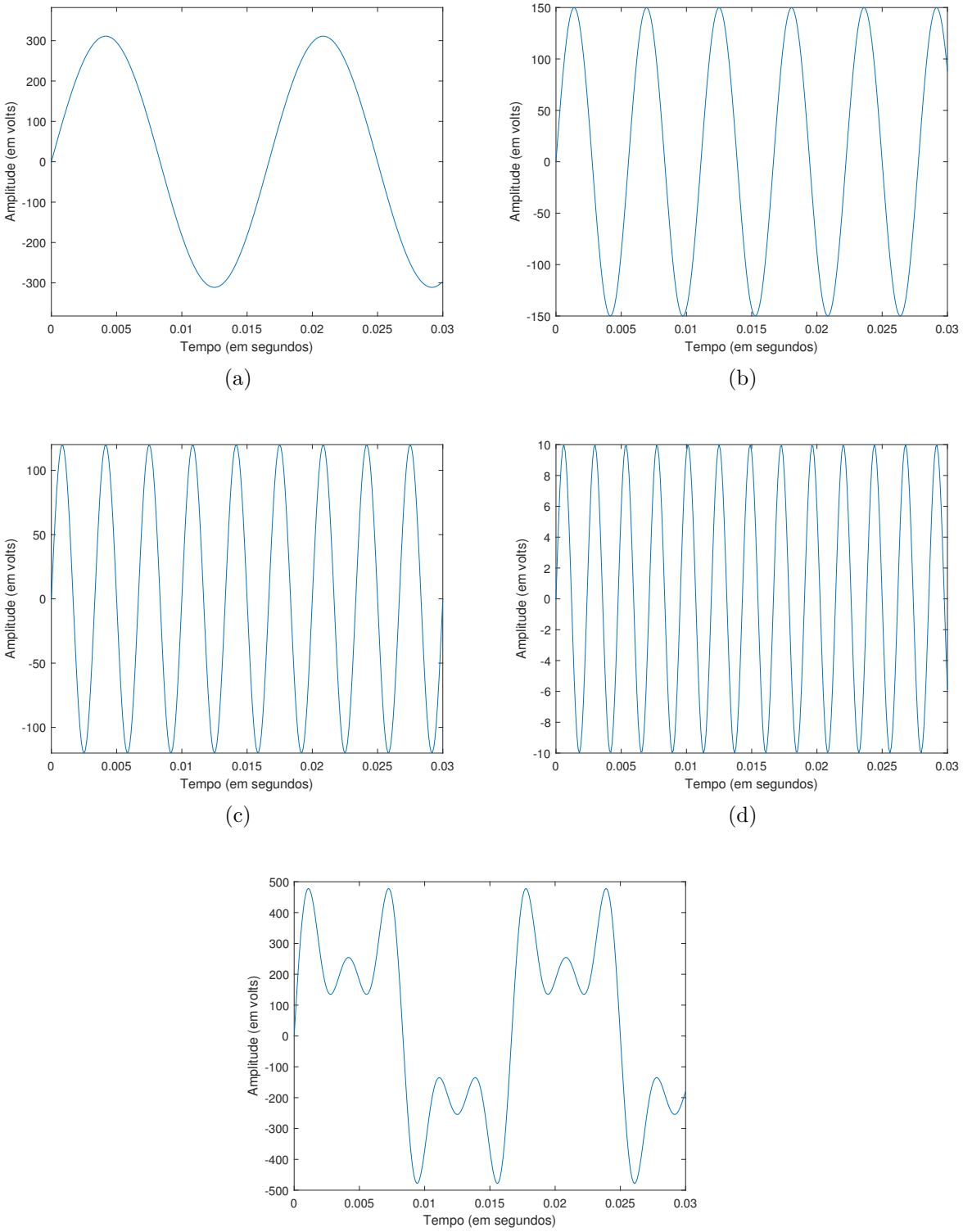


Figura 1.3 – Sinal típico de um sistema de potência com frequência fundamental de 60 Hz e componentes harmônicas de ordem 3,5 e 7.(a) Componente fundamental. (b) Terceira harmônica. (c) Quinta harmônica. (d) Sétima harmônica. (e) Forma de onda resultante.

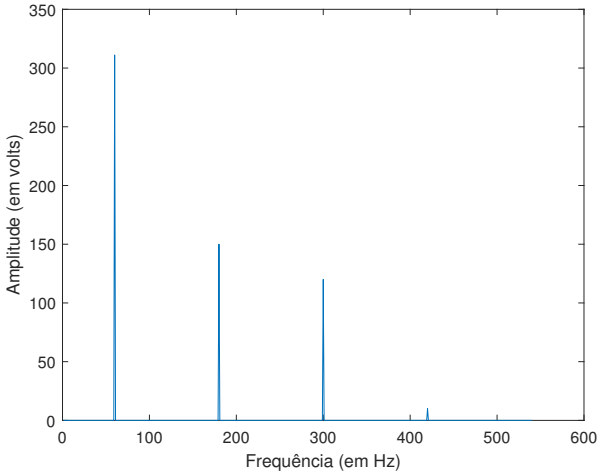


Figura 1.4 – Espectro de frequência do sinal dado por (1.3).

Verifica-se que a distorção harmônica é bastante alta, pois no sistema elétrico brasileiro considera-se um DTH aceitável um valor máximo de 0.05 (DECKMANN; POMILIO,). Graficamente, percebe-se que a onda resultante é bastante distorcida em relação à componente fundamental, conforme ilustram as Figuras 1.3(a) e 1.3(e). O valor eficaz também é bastante diferente quando comparado ao valor da componente fundamental que é de 220 V. Isto pode provocar, por exemplo, erros no dimensionamento de condutores e equipamentos de proteção. Por fim, verifica-se a característica esparsa da planta utilizada como já comentado anteriormente. Neste caso, afirma-se que a planta é bastante esparsa devido ao grau de esparsidade elevado (próximo a unidade).

1.3 Introdução aos tensores

Os tensores foram introduzidos no final do século XIX com o desenvolvimento do cálculo diferencial (COMON, 2014). Apesar disso, apenas recentemente tornaram-se onipresente nas áreas de processamento de sinais (voz, áudio, comunicações, radar, biomédico), aprendizado de máquinas (agrupamento, redução de dimensionalidade, aprendizagem no subespaço), estatística, entre outras. Psicometria (vagamente definidos como métodos matemáticos para a análise de dados de personalidade) e depois quimiometria (da mesma forma, para dados químicos) historicamente têm sido duas áreas de aplicação importantes, que utilizam tensores para o desenvolvimento da sua teoria e de algoritmos (SIDIROPOULOS et al., 2017).

Um tensor é essencialmente um mapeamento de um espaço linear para outro, cujas coordenadas se transformam multilinearmente sob uma mudança de bases (COMON, 2014). Assim, a utilização de tensores permite utilizar vários sistemas de coordenadas, não ficando limitado apenas a um. Tensores de ordem 0 são representados por um escalar, de ordem 1 por vetores, de ordem 2 por matrizes e acima disso podem ser descritos a partir de

componentes separadas. A partir de um perspectiva técnica, tensores são generalizações diretas de vetores e matrizes, como demostrado na Figura 1.4 (SHULGA et al., 2019). Aplicados em problemas de filtragem adaptativa, os tensores nos permite representar a resposta ao impulso de uma planta, por exemplo, através de dois ou mais vetores, o que diminui a sua complexidade computacional e melhora a taxa de aprendizagem (RUPP; SCHWARZ, 2015).

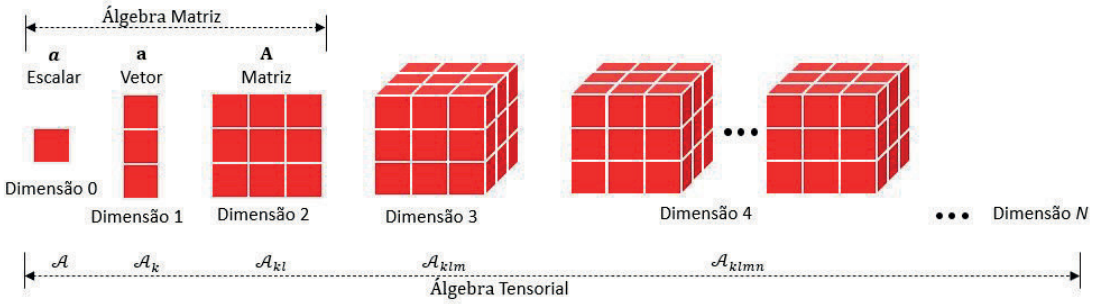


Figura 1.5 – Representação tensorial a partir de vetores e matrizes.

1.4 Motivações

As componentes harmônicas causam diversos problemas na rede elétrica, tais como perdas nas redes de distribuição de energia, superaquecimento de acionamentos elétricos, mau funcionamento de relés e disjuntores, distorções graves da forma de onda, etc (SAHOO; SUBUDHI, 2015). Assim, a estimação em tempo real das componentes harmônicas em sistema de potência é de fundamental importância para garantir a qualidade de energia transmitida, além de garantir a confiabilidade do sistema.

O método mais simples para analisar harmônicos em elétrica os sistemas de energia são baseados na Transformada Rápida de Fourier (em inglês *Fast Fourier Transform*, ou FFT). No entanto, a estimativa de harmônicos usando esta metodologia leva a imprecisões devido ao processo de vazamento espectral e erro de fase (SAHOO; SUBUDHI, 2015). Outras estimativas paramétricas de frequência bem conhecidas inclui técnicas de previsão linear (SO et al., 2005), máxima probabilidade (TUFTS; KUMARESAN, 1982), quadrado total mínimo (RAHMAN; YU, 1987), método de Prony (CHEN; CHANG, 2009), PLL (*Phase Locked Loop*) (FILHO et al., 2008) e métodos de subespaço, como decomposição de valor singular (SVD) (LOBOS; KOZINA; KOGLIN, 2001) . No entanto, cada um deles tem várias limitações em termos de imprecisões e requisitos computacionais na presença de ruído.

Atualmente, a aplicação de tecnologias que utiliza Inteligência Artificial (IA) em sistemas de potência tem sido uma área ativa de pesquisa e melhorias significativas foram alcançadas nos últimos anos (SEIFOSSADAT et al., 2007). Entre os métodos de

IA propostos, redes neurais artificiais se destacam neste tipo de aplicação devido sua simplicidade, capacidade de aprendizado e generalização (LIN, 2007).

1.5 Objetivos da pesquisa

Os objetivos deste projeto de pesquisa estão divididos em gerais e específicos, sendo apresentados abaixo.

1.5.1 Objetivo Geral

Estimar as componentes harmônicas presentes no sistema elétrico de potência utilizando redes neurais artificiais tensoriais.

1.5.2 Objetivos Específicos

- Desenvolver o algoritmo GNGD tensor.
- Aplicar os algoritmos adaptativos tensoriais em problemas que envolvem plantas esparsas.
- Aplicar os algoritmos adaptativos tensoriais em problemas que envolvam a estimação de harmônicas em sistemas de potência.
- Desenvolver uma rede neural tensorial para estimar as componentes harmônicas presentes no sistema elétrico de potência.

1.6 Organização do trabalho

Este trabalho de pesquisa está organizado da seguinte forma:

- Capítulo 2: Neste capítulo, são apresentados os conceitos básicos sobre filtragem adaptativa, com ênfase nos algoritmos adaptativos LMS, NLMS e GNGD. No final do capítulo, são abordados problemas práticos onde a filtragem adaptativa é aplicada. O objetivo aqui é comparar o desempenhos dos algoritmos adaptativos citados.
- Capítulo 3: Os algoritmos adaptativos tensoriais são desenvolvidos neste capítulo. Inicialmente, um estudo sobre a propriedade de separabilidade de operadores lineares é realizado. Com base na teoria apresentada, os algoritmos adaptativos tensoriais são derivados a partir dos algoritmos tradicionais. Por fim, é apresentada a estimação de harmônicas em sistemas elétricos de potência utilizando os algoritmos propostos.

- Capítulo 4: Neste capítulo, são apresentados os conceitos básicos sobre redes o processo de aprendizagem e treinamento das neurais artificiais. Em um primeiro momento, são apresentados os tipos de processos de aprendizagem. Em seguida, é mostrada a derivação da regra de aprendizagem de Widrow-Hoff e do algoritmo *backpropagation*. Por fim, faz-se um breve estudo sobre problemas de generalização que pode ocorrer durante o treinamento de uma rede neural. Os diferentes tipos de arquiteturas e as funções de ativações básicas são apresentados no apêndice do trabalho.

- Capítulo 5: Neste capítulo, a regra de aprendizagem tensorial com base na teoria apresentada no capítulo 3. O desempenho da metodologia proposta é avaliado, primeiramente em um problema de identificação de sistemas e em seguida para a estimação de harmônicas em sistemas elétricos de potência.

- Capítulo 6: Este capítulo apresenta as conclusões deste trabalho de pesquisa e possíveis trabalhos futuros.

2 Filtragem adaptativa

A capacidade de um filtro adaptativo de operar satisfatoriamente em um ambiente desconhecido e rastrear variações no tempo das estatísticas do sinal de entrada, tornam este dispositivo uma ferramenta poderosa para aplicações em processamento de sinal e controle (HAYKIN, 2002, p.18). Dentre as suas diversas aplicações, podem-se citar a identificação de sistemas, predição, cancelamento de eco e equalização de canais. Em termos gerais, o estudo destes filtros consiste na análise e no desenvolvimento de algoritmos adaptativos, os quais são implementados em estruturas de filtragem adaptativa dedicadas à resolução de problemas específicos (BRANCO, 2016, p.11).

Os filtros adaptativos podem ser linear ou não linear. Um filtro é considerado linear se a quantidade filtrada, suavizada ou prevista na saída do dispositivo é uma função linear das observações aplicadas à entrada do filtro. Caso contrário, o filtro é não linear (HAYKIN, 2002, p.18). Na abordagem estatística para resolução de problemas de filtragem linear, é necessário o conhecimento de certos parâmetros estatístico dos sinais, tais como média e função de correlação. Uma abordagem útil para otimização do filtro considera a minimização do erro médio quadrático. Para sinais estacionários, o filtro de Wiener pode ser projetado a fim de obter a solução ótima no sentido do erro médio quadrático. Entretanto, a sua aplicação requer o conhecimento a priori das estatísticas do sinais subjacentes (FARHANG-BOROUJENY, 2013, p.3).

Quando o conhecimento das estatísticas dos sinais e ruídos não estão disponíveis a priori, ou mesmo os sinais aplicados não são estacionários, é possível desenvolver um filtro útil usando um algoritmo recursivo para ajustar os seus parâmetros com base no fluxo de dados de entrada. Isto é o que um filtro adaptativo faz. Caso as estatística do sinais e ruídos associados sejam estacionárias, o filtro adaptativos deverá convergir para a solução de Wiener. Caso sejam não estacionárias, o filtro adaptativo consegue rastreá-los se variarem em uma taxa suficientemente lenta (ZAKNICH, 2005, p.4). Uma grande variedade de algoritmos recursivos foram desenvolvidos na literatura para em filtros adaptativos lineares. A escolha de um algoritmo sobre outro é determinado por um ou mais fatores, tais como a taxa de convergência, capacidade de rastreamento, robustez e custo computacional requerido (HAYKIN, 2002, p.2).

As principais classes de aplicação de filtragem adaptativas são mostradas no início deste capítulo. Em seguida, é apresentado os filtros adaptativos transversais, no quais são utilizados para a implementação dos algoritmos adaptativos considerados neste trabalho. Posteriormente, os conceitos sobre o filtro de Wiener são apresentados. Por fim, são apresentados o algoritmo de descida mais ingrime e, sua variante estocástica, o algoritmo LMS, além dos algoritmos NLMS e GNGD.

2.1 Aplicações básica de filtragem adaptativa

As quatros classe principais de aplicações de filtragem adaptativa são apresentadas brevemente a seguir. (HAYKIN, 2002, p.18).

2.1.1 Identificação

A Figura 2.1 representa o esquema geral de identificação de sistemas utilizando filtros adaptativos. Nesta figura, $\mathbf{x}(k)$ representa o sinal de entrada aplicado a planta, $\mathbf{d}(k)$ é o sinal desejado, $\mathbf{y}(k)$ é a saída do filtro, $\mathbf{e}(k)$ é o sinal de erro e $\mathbf{v}(k)$ é o ruído de medição. Os vetores $\mathbf{p}(k)$ e $\mathbf{w}(k)$ são, respectivamente, a resposta ao impulso da planta e do filtro. Durante o processo de adaptação, o algoritmo adaptativo representado por $\mathbf{A}$ utiliza o sinais $\mathbf{x}(k)$ e $\mathbf{e}(k)$ para atualizar, em cada instante k, a resposta impulsiva do filtro, $\mathbf{w}(k)$. O objetivo é minimizar o erro, $\mathbf{e}(k)$, a fim de tornar $\mathbf{y}(k)$ uma boa estimativa de $\mathbf{d}(k)$.

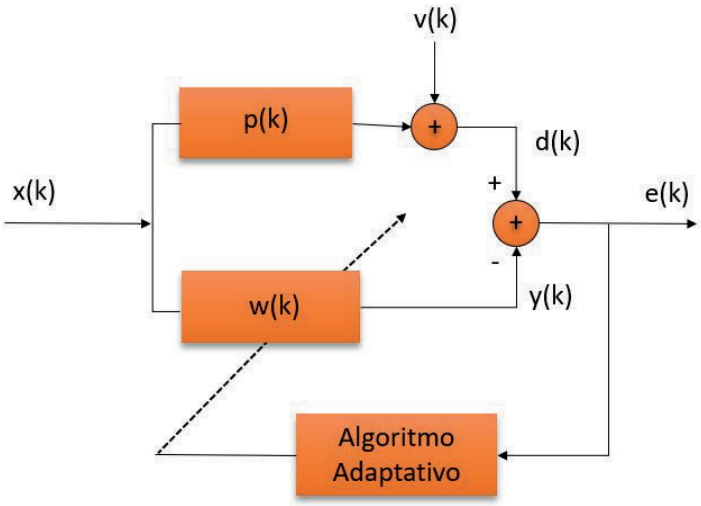


Figura 2.1 – Esquema de identificação de sistemas utilizando filtragem adaptativa.

Em aplicações onde a planta varia com o tempo, sua saída não é estacionária, assim como o resposta desejada apresentada ao algoritmo de filtragem adaptativa. Em tal situação, o algoritmo de filtragem adaptativa tem a tarefa de não apenas manter o erro de modelagem pequeno, mas também rastrear continuamente as variações de tempo na dinâmica da planta.

2.1.2 Modelagem inversa

A modelagem inversa, também conhecida como deconvolução, é outra aplicação de filtros adaptativos que encontrou amplo uso em várias disciplinas de engenharia. A aplicação mais utilizada de modelagem inversa é em telecomunicações, onde um modelo

inverso (também chamado equalizador) é usado para mitigar a distorção de um canal (FARHANG-BOROUJENY, 2013, p.11). Na Figura 2.2, é ilustrado o princípio de funcionamento de um processo de equalização adaptativa. Verifica-se que um sinal $\mathbf{s}(k)$ é transmitido pelo canal, sendo este vulnerável a presença de ruídos. Em seguida, a saída do canal adicionada de um ruído de medição é aplicada a entrada do filtro adaptativo. O sinal de referência neste modelo é representado por $\mathbf{s}(k)$ atrasado, sendo utilizado para compensar o tempo requerido para o envio dos dados ou o processamento. Com o objetivo de cancelar o possível ruído adicionado ao sistema, o filtro adaptativo atualiza os seus coeficientes para compor um modelo inverso do canal ruidoso. Assim, após a minimização do MSE, assumindo que o ruído presente no sistema é desprezível, a convolução entre a resposta ao impulso do canal, representado por $H_c(z)$ e a resposta ao impulso do filtro, $W_f(z)$ é dada por

$$H_c(z)W_f(z) = z^{-L}. \tag{2.1}$$

Com isso, a saída do filtro, $y(k)$, fornece o próprio sinal $s(k)$ atrasado, ou seja, $s(k-L)$, onde L representa o atraso que foi adicionado para compor o sinal de referência.

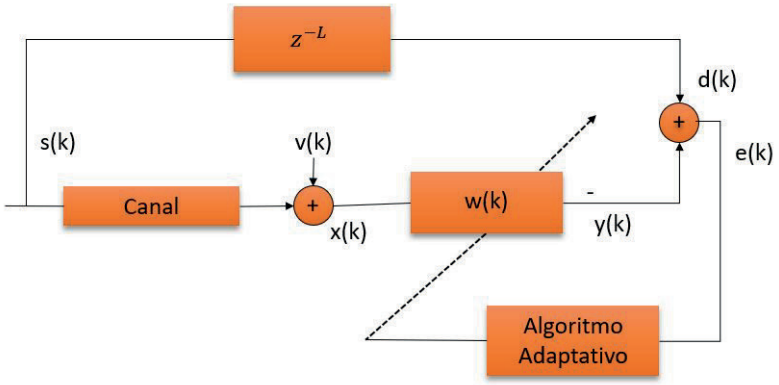


Figura 2.2 – Esquema de modelagem inversa utilizando filtragem adaptativa.

2.1.3 Predição

Predição é uma técnica de estimativa espectral usada para modelar processos aleatoriamente correlacionados com o objetivo de encontrar uma representação paramétrica desses processos (FARHANG-BOROUJENY, 2013, p.15). A Figura 2.3(a) representa um modelo de predição adaptativa *forward*, onde o sinal de entrada é atrasado em L unidades de tempo e alimenta o filtro adaptativo. O sinal de referência é representado pelo sinal de entrada. Os pesos da rede são adaptados e, quando eles convergem, produzem uma melhor estimativa da entrada atual dada uma entrada atrasada por L unidades (ZAKNICH, 2005, p.22). Na Figura 2.3(b), é mostrado o modelo de predição adaptativa *backward*, onde o sinal de referência é atrasado em L unidades de tempo. Assim, esta estrutura é utilizada para determinar os valores das amostras passadas do sinal de entrada.

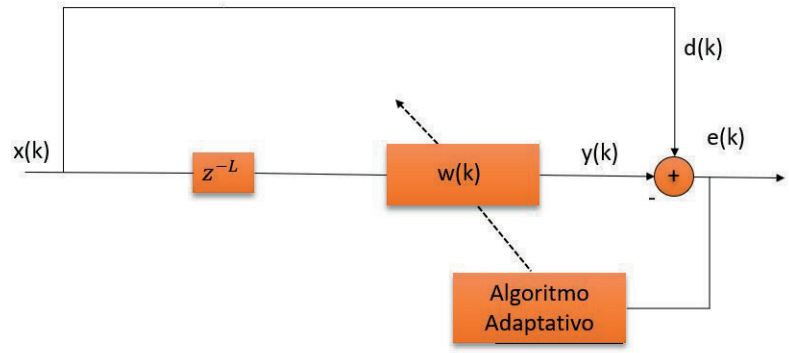


Figura 2.3 – Esquema de predição *forward* utilizando filtragem adaptativa.

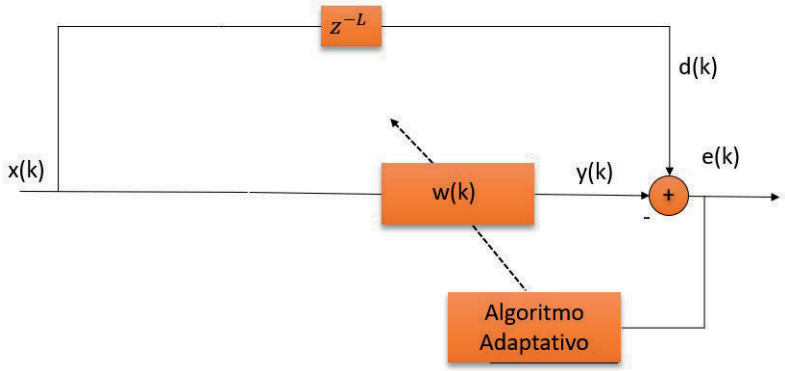


Figura 2.4 – Esquema de predição *backward* utilizando filtragem adaptativa.

2.1.4 Cancelamento de interferência

O cancelamento de interferência refere-se a situações em que é necessário cancelar um ruído presente em um determinado sinal. O princípio do cancelamento da interferência é obter uma estimativa do sinal interferente e subtraí-la do sinal corrompido. A viabilidade dessa ideia depende do disponibilidade de uma fonte de referência, da qual se origina o sinal de interferência (FARHANG-BOROUJENY, 2013, p.21).

O esquema de filtragem adaptativa aplicada em cancelamento de interferência é mostrado na Figura 2.4. O sinal de referência é formado pela soma do sinal de interesse, $\mathbf{s}(k)$, com o sinal de interferência, $\mathbf{v}(k)$. A entrada aplicada ao filtro é um ruído que é correlacionado com o ruído original, representado por $\mathbf{v}'(k)$. Após o ajuste dos coeficientes realizado pelo algoritmo adaptativo, a saída do filtro representa o próprio ruído, $\mathbf{v}(k)$. Assim, o sinal de erro, $\mathbf{e}(k)$, torna-se a estimativa de $\mathbf{s}(k)$, caso $\mathbf{s}(k)$, $\mathbf{v}(k)$ e $\mathbf{v}'(k)$ sejam estacionários, com média zero e $\mathbf{s}(k)$ não seja correlacionado com $\mathbf{v}(k)$ e $\mathbf{v}('k)$ (HAYKIN, 2002, p.52).

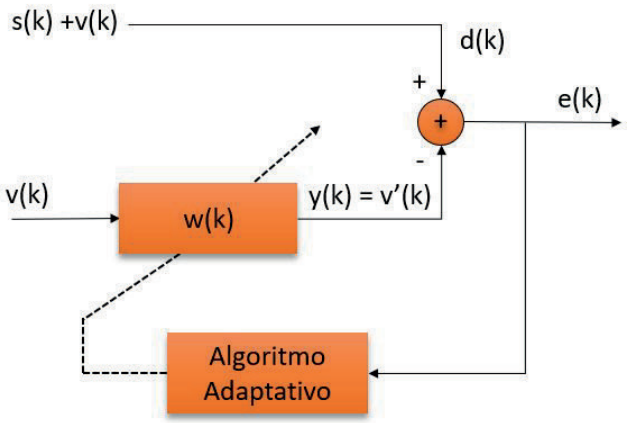


Figura 2.5 – Esquema de cancelamento de interferência utilizando filtragem adaptativa.

2.2 Filtro adaptativo transversal

A estrutura mais comumente utilizada na implementação de filtros adaptativos é a estrutura transversal, ilustrada na Figura 2.5 (DINIZ, 2008, p.3). Aqui, o filtro adaptativo possui apenas uma entrada, $\mathbf{x}(k)$, e uma saída, $\mathbf{y}(k)$, gerada a partir da Equação (2.2)

$$\mathbf{y}(k) = \sum_{i=0}^{N-1} \mathbf{w}_i(k)\mathbf{x}(k-i), \tag{2.2}$$

onde $\mathbf{w}_i(k)$ são os coeficientes do filtro, ajustados a cada instante k pelo algoritmo adaptativo, e N é o comprimento do filtro.

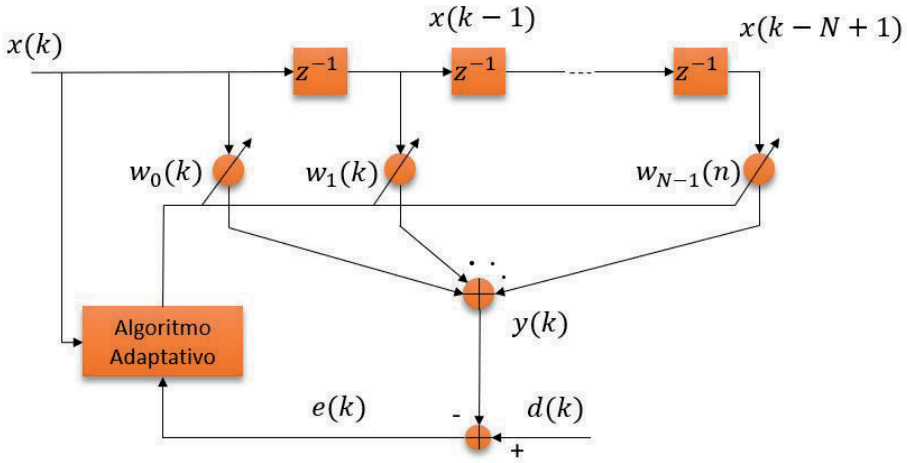


Figura 2.6 – Estrutura transversal de um filtro adaptativo.

Em algumas aplicações, as entradas do filtro não são formadas pelas amostras atrasada de uma única entrada, como demostrado na Figura 2.6. Essa estrutura é chamada de combinador linear, cuja saída é igual soma das N entradas, $\mathbf{x}_i(k)$, ponderada pelos seus

respectivos coeficiente, $\mathbf{w}_i(k)$, conforme demostrado na equação 2.3.

$$\begin{aligned}\mathbf{y}(k) &= \sum_{i=0}^{N-1} \mathbf{w}_i(k)\mathbf{x}(k) \\ &= \mathbf{w}^T(k)\mathbf{x}(k). \end{aligned} \tag{2.3}$$

Verifica-se que a estrutura do combinador linear é mais geral que a transversal, onde $\mathbf{x}_i(k) = \mathbf{x}(k-i)$.

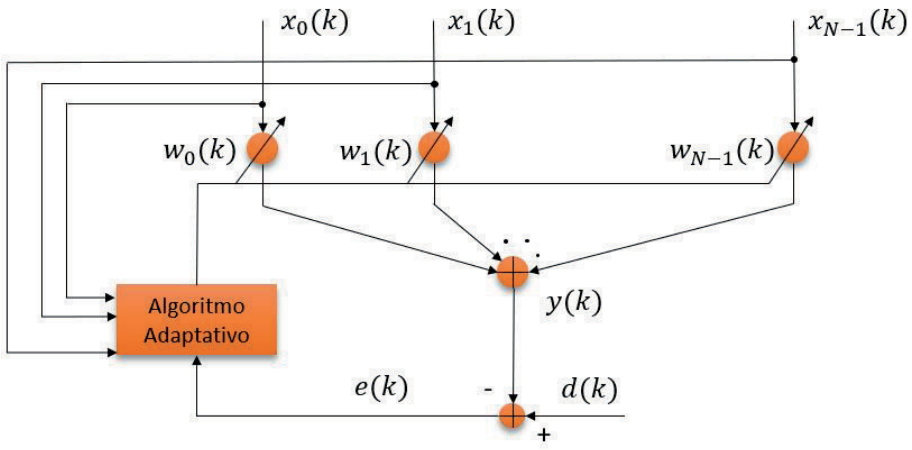


Figura 2.7 – Estrutura de um combinador linear.

Na Figura 2.5, observa-se que o cálculo da saída do filtro não envolve nenhum mecanismo de realimentação. Tal estrutura é denominada filtro FIR (em inglês *Finite Impulse Response*), visto que sua resposta ao impulso é de duração finita. A estrutura FIR tem uma vantagem qualitativa importante sobre uma estrutura de filtro IIR (em inglês *Infinite Impulse Response*), que possui realimentação, visto que uma vez que o vetor de peso tenha convergido, o filtro resultante é garantido ser estável (SCHILLING; HARRIS, 2011, p.646). Isto ocorre devido aos polos do filtro FIR estarem localizados na origem do plano z.

2.3 Filtro de Wiener

O filtro de Wiener é uma ferramenta fundamental para um melhor entendimento sobre os filtros adaptativos. Além disso, a filtragem de Wiener pode ser aplicada em qualquer problema que envolva uma estimação linear de um sinal desejado (FARHANG-BOROUJENY, 2013, p.49). Dentre as suas diversas aplicações, destacam-se a predição, equalização de canal (deconvolução) e suavização. Com base na teoria de Wiener, é possível obter os coeficientes ótimos de um filtro a partir da minimização do erro médio quadrático (em inglês *Mean Square Error*, ou MSE). Para isso, é necessário o conhecimento de certas funções estatísticas que, na prática, não são possíveis de serem obtidas.

Para resolver este problema, os sinais subjacentes são considerados ergódicos, ou seja, suas médias temporais e de conjunto são as mesmas (FARHANG-BOROUJENY, 2013, p.49).

A entrada do filtro, $\mathbf{x}(k)$, e o sinal desejado, $\mathbf{d}(k)$, são assumidos como sendo processos estacionários com valores reais. Os coeficientes do filtro, $w(k)$, também possuem valores reais. Com isso, o vetor de entrada e o vetor de coeficientes são definidos, respectivamente, por

$$\mathbf{w} = [w_0 \; w_1 \; \cdots \; w_{N-1},]^T \tag{2.4}$$

$$\mathbf{x}(k) = [x(k) \; x(k-1) \; \cdots \; x(k-N+1)]^T, \tag{2.5}$$

com T denotado o transposto do vetor.

A saída do filtro é dada por

$$\begin{aligned}\mathbf{y}(k) &= \sum_{i=0}^{N-1} \mathbf{w}_i \mathbf{x}(k-i) \\ &= \mathbf{w}^T \mathbf{x}(k).\end{aligned} \tag{2.6}$$

Assim, o sinal de erro é calculado por

$$\begin{aligned}\mathbf{e}(k) &= \mathbf{d}(k) - \mathbf{y}(k) \\ &= \mathbf{d}(k) - \mathbf{w}^T(k)\mathbf{x}(k) \\ &= \mathbf{d}(k) - \mathbf{x}^T(k)\mathbf{w}(k).\end{aligned} \tag{2.7}$$

Como dito anteriormente, a função do filtro de Wiener é minimizar o erro médio quadrático representado por

$$\xi(k) = E\left[e^2(k)\right], \tag{2.8}$$

sendo $E[\cdot]$ o operador valor esperado. Assim, substituindo a Equação(2.7) em (2.8), tem-se

$$\begin{aligned}E\left[\mathbf{e}^2(k)\right] &= E\left[\left(\mathbf{d}(k) - \mathbf{w}^T\mathbf{x}(k)\right)\left(\mathbf{d}(k) - \mathbf{x}^T(k)\mathbf{w}\right)\right] \\ &= E\left[\mathbf{d}^2(k) - 2\mathbf{d}(k)\mathbf{w}^T(k)\mathbf{x}(k) + \mathbf{w}^T(n)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)\right] \\ &= E\left[\mathbf{d}^2(k)\right] - E\left[2\mathbf{d}(k)\mathbf{w}^T(k)\mathbf{x}(k)\right] + E\left[\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)\right].\end{aligned} \tag{2.9}$$

A equação (2.9), onde $\mathbf{w}$ representa uma variável determinística, pode ser reescrita como

$$E[\mathbf{e}^2(k)] = E\left[\mathbf{d}^2(k)\right] - 2\mathbf{w}^T(k)E\left[\mathbf{d}(k)\mathbf{x}^T(k)\right] + \mathbf{w}^T(k)E\left[\mathbf{x}(k)\mathbf{x}^T(k)\right]\mathbf{w}(k). \tag{2.10}$$

Definindo a matriz de autocorrelação, $\mathbf{R}$, e o vetor de correlação cruzada, $\mathbf{p}$, como (WIDROW; STEARNS, 1985, p.20)

$$\mathbf{R} = E\left[\mathbf{x}(k)\mathbf{x}^T(k)\right] = \begin{bmatrix} x_{00} & x_{01} & x_{02} & \cdots & x_{0,N-1} \\ x_{10} & x_{11} & x_{12} & \cdots & x_{1,N-1} \\ x_{20} & x_{21} & x_{22} & \cdots & x_{2,N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N-1,0} & x_{N-1,1} & x_{N-1,2} & \cdots & x_{N-1,N-1}. \end{bmatrix}. \tag{2.11}$$

$$\mathbf{p} = E\left[\mathbf{x}(k)\mathbf{d}(k)\right] = \left[p_0,\ p_1 \cdots\ p_{N-1}\right]^T. \tag{2.12}$$

Substituindo (2.11) e (2.12) em (2.10), tem-se

$$\xi = E\left[\mathbf{d}^2(k)\right] - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w}. \tag{2.13}$$

Aplicando o gradiente em (2.13) em relação a $\mathbf{w}$, obtém-se

$$\begin{aligned} \frac{\partial \xi}{\partial \mathbf{w}} &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w} \\ \mathbf{w_{opt}} &= \mathbf{R}^{-1}\mathbf{p}, \end{aligned} \tag{2.14}$$

onde a matriz $\mathbf{R}$ deve ser definida positiva. A equação (2.14) é conhecida como equação de Wiener-Hopf (VEGA; REY, 2012, p.10).

Substituindo (2.14) em (2.13), obtém-se o MSE mínimo que pode ser alcançado pelo filtro de Wiener dado por

$$\xi_{min} = E\left[\mathbf{d}^2(k)\right] - \mathbf{p}^T\mathbf{R^{-1}}\mathbf{p}. \tag{2.15}$$

Com o objetivo de avaliar o comportamento do MSE para este tipo de filtro, considere que a matriz de autocorrelação é formada por $R = [r_{11} = 1, r_{12} = 0, r_{21} = 0, r_{22} = 1]$. Verifica-se que o vetor de entradas $x(k)$ contém, alem do ruído, o sinal desejado. Além disso, o vetor de correlação cruzada é dado por $\mathbf{p} = [2 \quad 4.5]^T$ e a variação do sinal desejado é $\sigma_d^2 = 24.50$. O intuito deste problema é determinar os coeficientes ótimos e a superfície de desempenho.

A superfície de desempenho é obtida substituindo os valores de (2.16) em (2.13). Logo, tem-se

$$\begin{aligned} \xi &= 24.5 - 2[\mathrm{w}_0\ \mathrm{w}_1]\begin{bmatrix} 2 \\ 4.5 \end{bmatrix} + [\mathrm{w}_0\ \mathrm{w}_1]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \mathrm{w}_0 \\ \mathrm{w}_1 \end{bmatrix} \\ &= 24.5 - 4w_1 - 9w_2 + w_1^2 + w_2^2, \end{aligned} \tag{2.16}$$

o que resulta na Figura 2.8.

Para determinar os coeficientes ótimos é necessário apenas aplicar a equação de Wiener-Hopf. Assim, tem-se

$$\begin{aligned} \mathbf{w_{opt}} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 4.5 \end{bmatrix} \\ &= \begin{bmatrix} 2 \\ 4.5 \end{bmatrix}. \end{aligned} \tag{2.17}$$

Por fim, para o cálculo do MSE mínimo, é necessário substituir os coeficientes ótimos encontrados na equação (2.17) em (2.16). Assim, tem-se

$$\xi_{min} = 0.25. \tag{2.18}$$

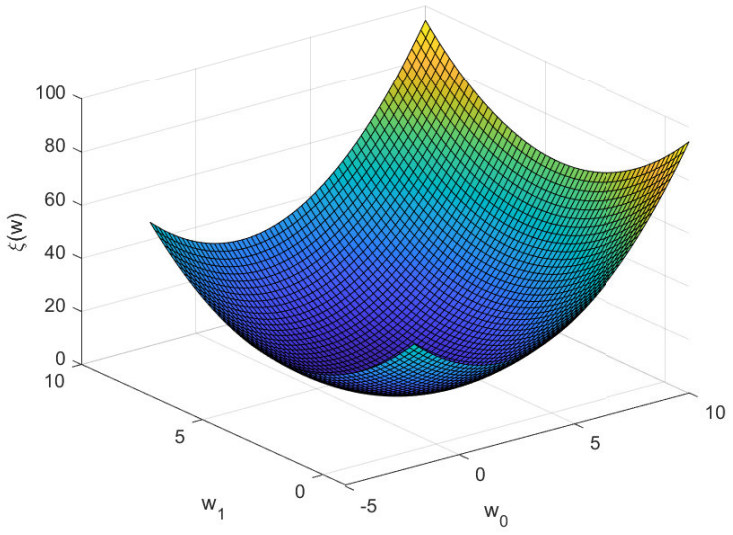


Figura 2.8 – Superfície de desempenho para o problema proposto.

A partir do exemplo anterior, observa-se que, uma vez conhecida as estatística dos sinais envolvidos, a equação de Wiener-Hofp pode ser utilizada para determinar os coeficientes ótimos do filtro no sentido do MSE. Entretanto, em muitas aplicações do mundo real, as estatística dos sinais não são conhecidas. Além disso, a solução de Wiener é adequada apenas para ambientes estacionários, ou seja, se as propriedades estatística forem variantes no tempo, a solução encontrada deixa de ser ótima.

2.4 Algoritmo de descida mais íngreme

O algoritmo de descida mais íngreme (em inglês *Steepest Descent*, ou SD) baseia-se nos conceitos de coeficientes ótimos do filtro de Wiener. Considerando que a superfície de desempenho é convexa (como a da Figura 2.1), parte-se de um ponto inicial da superfície ate chegar ao ponto de mínimo. Esse ajuste do vetor de coeficientes é feito a partir de um passo de adaptação, μ, que controla a velocidade de convergência do algoritmo.

O algoritmo de descida mais íngreme atualiza os coeficientes do filtro utilizando a forma geral

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\nabla_k\xi, \tag{2.19}$$

onde μ é um escalar positivo e $\nabla_k\xi$ o vetor gradiente, $\nabla\xi$, calculado na k-ésima iteração. O sinal negativo representa a busca do gradiente pelo ponto de mínimo da função. Substituindo a Equação (2.14) em (2.19), tem-se

$$\mathbf{w}(k+1) = \mathbf{w}(k) - 2\mu[\mathbf{R}\mathbf{w}(k) - 2\mathbf{p}]. \tag{2.20}$$

De modo geral, o algoritmo SD utiliza uma estimativa inicial de $\mathbf{w}_{opt}$ e atualiza o seu valor de forma recursiva adotando o procedimento a seguir.

1. Determina o gradiente da função de desempenho em relação aos coeficientes do filtro;

2. Atualize a estimativa dos pesos do filtro tomando um passo na direção oposta do vetor gradiente calculado no passo 1;

3. Repita os passos 1 e 2 até que não seja mais observada uma mudança significativa nos valores dos pesos do filtro.

2.4.1 Análise de estabilidade do algoritmo de descida mais íngreme

A Equação (2.20) pode ser reescrita como

$$\mathbf{w}(k+1) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{w}(k) + 2\mu\mathbf{p}, \tag{2.21}$$

onde $\mathbf{I}$ representa a matriz identidade de ordem N. Substituindo $\mathbf{p} = \mathbf{R}\mathbf{w}_{opt}$ e subtraindo $\mathbf{w}_{opt}$ de ambos os lados da equação (2.21), obtém-se

$$\begin{aligned} \mathbf{w}(k+1) - \mathbf{w}_{opt} &= (\mathbf{I} - 2\mu\mathbf{R})\mathbf{w}(k) + 2\mu\mathbf{R}\mathbf{w}_{opt} - \mathbf{w}_{opt} \\ &= (\mathbf{I} - 2\mu\mathbf{R})(\mathbf{w}(k) - \mathbf{w}_{opt}). \end{aligned} \tag{2.22}$$

Em seguida, definindo o vetor

$$\mathbf{v}(k) = \mathbf{w}(k) - \mathbf{w}_{opt}, \tag{2.23}$$

e substituindo (2.23) em (2.24), tem-se

$$\mathbf{v}(k+1) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{v}(k). \tag{2.24}$$

A Equação (2.24) pode ser simplificada utilizando a seguinte transformação

$$\mathbf{R} = \mathbf{Q}^T\Lambda\mathbf{Q} \tag{2.25}$$

onde Λ é uma matriz diagonal composta pelos autovalores de $\mathbf{R}$ e $\mathbf{Q}$ é uma matriz cujas colunas são formadas pelos autovetores de $\mathbf{R}$. Substituindo a Equação (2.25) em (2.24), obtém-se

$$\begin{aligned} \mathbf{v}(k+1) &= (\mathbf{Q}\mathbf{Q}^T - 2\mu\mathbf{Q}\Lambda\mathbf{Q}^T) \\ &= \mathbf{Q}(\mathbf{I} - 2\mu\Lambda)\mathbf{Q}^T\mathbf{v}(k). \end{aligned} \tag{2.26}$$

Multiplicando ambos os lados por Q^T e aplicando a seguinte transformação

$$\mathbf{v}'(k+1) = \mathbf{Q}^T\mathbf{v}(k), \tag{2.27}$$

tem-se

$$\mathbf{v}'(k+1) = (\mathbf{I} - 2\mu\Lambda)\mathbf{v}'(k). \tag{2.28}$$

Separando a Equação vetorial (2.28) em N equações escalares recursiva, obtém-se

$$\mathbf{v}'_i(k) = (\mathbf{1} - 2\mu\lambda_i)^k \ \mathbf{v}'_i(0), \qquad i = 1, 2, ..., N. \tag{2.29}$$

A partir das Equações (2.23) e (2.27), observa-se que $w(k)$ converge para w_{opt} apenas se $v'_i(k)$ converge para um vetor nulo. Entretanto, verifica-se pela Equação (2.29) que $v'_i(k)$ converge para zero se, e somente se, o passo de adaptação, μ, estiver dentro do intervalo

$$|1 - 2\mu\lambda_i| < 1, \qquad i = 1, 2, ..., N. \tag{2.30}$$

Assim, a condição necessária para a convergência e estabilidade do algoritmo SD é descrita por

$$0 < \mu < \frac{1}{\lambda_{max}}, \tag{2.31}$$

onde λ_{max} é o maior autovalor da matriz $\mathbf{R}$.

2.5 Algoritmo LMS

O algoritmo LMS (em inglês *Least Mean Square*) é baseado no algoritmo de descida mais íngreme, com a diferença que utiliza aproximações instantâneas do vetor gradiente real, de maneira que não são necessárias as estatísticas dos sinais de entrada e desejado (SAYED, 2003, p.212). Entre as características que fizeram esse algoritmo ser tão utilizado, pode-se citar: a estabilidade e robustez para uma variedade de condições de sinal, baixo custo computacional e convergência em média para a solução de Wiener (POULARIKAS, 2017, p.228). Devido a essas e outras importantes propriedades, o algoritmo LMS é objeto de estudo entre vários pesquisadores, sendo que várias modificações propostas ao longo dos anos (FARHANG-BOROUJENY, 2013, p.139). A seguir, é apresentada a derivação do LMS a partir do algoritmo de descida mais íngreme.

2.5.1 Derivação do algoritmo LMS

Considerando que a matriz de autocorrelação $\mathbf{R}$ e o vetor de correlação cruzada $\mathbf{p}$ podem ser estimados a partir de amostras instantâneas do vetor de entrada e do sinal desejado, conforme definido, respectivamente em

$$\hat{\mathbf{R}}(k) = \mathbf{x}(k)\mathbf{x}^T(k), \tag{2.32}$$

e

$$\hat{\mathbf{p}}(k) = \mathbf{x}(k)\mathbf{d}(k), \tag{2.33}$$

então, a estimativa do vetor gradiente pode ser representada por

$$\nabla\hat{\xi}(k) = -2\mathbf{x}(k)\mathbf{d}(k) + 2\mathbf{x}(k)x^T(k)\mathbf{w}(k). \tag{2.34}$$

Substituindo o vetor gradiente da Equação (2.19) por (2.34), obtém-se a seguinte relação recursiva para atualização do vetor de pesos

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) + \mu\mathbf{x}(k)[\mathbf{d}(k) - \mathbf{x}^T(k)\mathbf{w}(k)] \\ &= \mathbf{w}(k) + \mu\mathbf{x}(k)[\mathbf{d}(k) - \mathbf{w}^T(k)\mathbf{x}(k)] \\ &= \mathbf{w}(k) + \mu\mathbf{e}(k)\mathbf{x}(k), \end{aligned} \tag{2.35}$$

onde a saída do filtro, $\mathbf{y}(k)$, e o erro na iteração, $\mathbf{e}(k)$, são obtidos por

$$\mathbf{y}(k) = \mathbf{w}^T(k)\mathbf{x}(k). \tag{2.36}$$

$$\mathbf{e}(k) = \mathbf{d}(k) - \ \mathbf{y}(k). \tag{2.37}$$

O algoritmo definido pelas Equações (2.35), (2.36) e (2.37) constituem o algoritmo LMS. Como pode-se verificar acima, em cada iteração, o algoritmo LMS requer o conhecimento dos valores mais recentes de $\mathbf{x}$(k), $\mathbf{d}$(k) e $\mathbf{w}$(k). A parcela referente à esquerda da Equação (2.35) refere-se à correção que é realizada no vetor de pesos a cada iteração. Verifica-se, nas Equações (2.32) e (2.33), que as estimativas de $\hat{\mathbf{R}}$ e $\hat{\mathbf{p}}$ possuem grande variações ao longo de cada iteração. Isso poderia gerar uma dúvida quanto ao desempenho do algoritmo LMS. No entanto, devido a sua natureza recursiva, o próprio algoritmo calcula a média dessas estimativas, de certa maneira, durante o processo de adaptação. No Algoritmo 2.1 é descrito os passos para a implementação do algoritmo LMS.

De maneira geral, o algoritmo LMS se comporta, em média, como o algoritmo de descida mais íngreme. Por consequência, o algoritmo LMS também é controlado por N modos de convergência, que são caracterizados pelos autovalores da matriz de autocorrelação $\mathbf{R}$. Assim, semelhante ao algoritmo de descida mais íngreme, pode-se afirmar que o algoritmo LMS converge, em média, para zero quando μ permanece dentro do intervalo (WIDROW; STEARNS, 1985, p.101)

$$0 < \mu < \frac{1}{\lambda_{max}}. \tag{2.38}$$

Um fator de grande influência no desempenho do algoritmo LMS é o nível de dispersão da matriz de autocorrelação do vetor de entradas, calculado conforme abaixo

$$\chi(\mathbf{R}) = \frac{\lambda_{max}}{\lambda_{min}}, \tag{2.39}$$

onde λ_{max} e λ_{min} são, respectivamente, o maior e o menor autovalor da matriz $\mathbf{R}$. Para valores de $\chi(\mathbf{R})$ elevados (matriz mal condicionada), ocorre uma diminuição da velocidade de convergência e aumento do erro médio quadrático em regime permanente (HAYKIN, 2014, p.315). Outro fator que influência no desempenho do algoritmo LMS é o nível de esparsidade da planta. Plantas muito esparsas provocam um desempenho pobre do algoritmo LMS devido a diminuição da taxa de aprendizagem (RUPP; SCHWARZ, 2015).

Algoritmo 2.1 LMS

1: **Inicialização e parâmetros**

$$\mathbf{w}(0) = \mathbf{0}$$

$$0 < \mu < \frac{1}{\lambda_{\max}}$$

2: **Dados de entrada e saída da planta e do filtro adaptativo**

$$d(k) = \mathbf{x}^T(k)\mathbf{p}(k)$$

$$y(k) = \mathbf{x}^T(k)\mathbf{w}^T(k)$$

3: **Cálculo do sinal do erro**

$$e(k) = d(k) - y(k) + v(k)$$

4: **Atualização dos coeficientes do filtro**

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k)$$

2.6 Algoritmo NLMS

Como pode-se observar na Equação (2.35), a atualização do vetor de pesos no algoritmo LMS é diretamente proporcional ao vetor de entradas $\mathbf{x}$(k). Para entradas de natureza estatísticas variantes ocorre a amplificação do problema de ruído de gradiente, o que provoca um desempenho pobre do algoritmo LMS (HAYKIN, 2002, p.433). Para contornar esse problema, foi proposto o algoritmo NLMS (em inglês *Normalized Least Mean Square*), que pode ser visto como uma implementação especial do algoritmo LMS, na qual é levado em consideração a variação do nível de potência do sinal de entrada, utilizando um passo efetivo variável e normalizado, o que resulta em um algoritmo com boa estabilidade e rápida velocidade de convergência (FARHANG-BOROUJENY, 2013, p.170).

O algoritmo NLMS pode ser desenvolvido a partir de diferentes pontos de vista. A seguir será utilizado o método que considera o algoritmo NLMS como um problema de otimização com restrições (HAYKIN, 2002, p.433).

2.6.1 Derivação do algoritmo NLMS

Considere o seguinte problema:
Minimizar

$$\eta(k) = \mathbf{w}(k+1) - \mathbf{w}(k), \tag{2.40}$$

sujeito a seguinte restrição

$$\mathbf{d}(k) = \mathbf{w}^T(k)\mathbf{x}(k). \tag{2.41}$$

Para resolver esse problema, utiliza-se o método de multiplicadores de Lagrange (BAZARAA; SHERALI; SHETTY, 2004, p.506). Para isso, aplica-se a norma euclidiana elevado ao quadrado de η(k)

$$\begin{aligned}\|\eta(k)\|_2^2 &= \eta^T(k)\eta(k) \\ &= [\mathbf{w}(k+1) - \mathbf{w}(k)]^T[\mathbf{w}(k+1) - \mathbf{w}(k)] \\ &= \sum_{i=1}^{N}[\mathbf{w}_i(k+1) - \mathbf{w}_i(k)]^2. \end{aligned} \tag{2.42}$$

Em seguida, reescrevendo a Equação (2.41), obtém-se

$$\mathbf{d}(k) = \sum_{i=1}^{N}\mathbf{w}_i(k+1)\mathbf{x}(k-i+1). \tag{2.43}$$

Com base nas Equações (2.42) e (2.43), a formulação do problema de otimização com restrição é dada por

$$\xi(k) = \sum_{i=1}^{N}[\mathbf{w}_i(k+1) - \mathbf{w}_i(k)]^2 + \gamma[\mathbf{d}(k) - \sum_{i=1}^{N}\mathbf{w}_i(k+1)\mathbf{x}(k-i+1)], \tag{2.44}$$

onde γ é o multiplicador de Lagrange. Para encontrar os valores de $\mathbf{w}_i$(k+1) que minimizam a função custo ξ(k), é necessário aplicar o gradiente e igualar a zero, conforme apresentado abaixo

$$\frac{\partial\xi(k)}{\partial\mathbf{w}_i(k+1)} = 0, \tag{2.45}$$

o que resulta em

$$2[\mathbf{w}_i(k+1) - \mathbf{w}_i(k)] - \gamma\mathbf{x}(k-i-1) = 0. \tag{2.46}$$

Rearranjando a Equação (2.46), tem-se

$$2[\mathbf{w}(k+1) - \mathbf{w}(k)] = \gamma\mathbf{x}(k). \tag{2.47}$$

A partir da Equação (2.47), isola-se γ conforme abaixo

$$\begin{aligned}\gamma\mathbf{x}^T(k)\mathbf{x}(k) &= 2[\mathbf{x}^T(k)\mathbf{w}(k+1) - \mathbf{x}^T(k)\mathbf{w}(k)] \\ \gamma\sum_{i=1}^{N}[\mathbf{x}(k-i-1)]^2 &= 2[\sum_{i=1}^{N}\mathbf{w}_i(k+1)\mathbf{x}(k-i-1) - \sum_{i=1}^{N}\mathbf{w}_i(k)\mathbf{x}(k-i-1)] \\ \gamma &= \frac{2}{\|\mathbf{x}(k)\|_2^2}[\mathbf{w}^T(k+1)\mathbf{x}(k) - \mathbf{w}^T(k)\mathbf{x}(k)] \end{aligned} \tag{2.48}$$

sendo $\|\mathbf{x}(k)\|$ a norma euclidiana do vetor de entradas $\mathbf{x}(k)$. Considerando a restrição estabelecida em (2.40), tem-se

$$\begin{aligned}\gamma &= \frac{2}{\|\mathbf{x}(k)\|_2^2}[\mathbf{d}(k) - \mathbf{w}^T(k)\mathbf{x}(k)] \\ &= \frac{2}{\|\mathbf{x}(k)\|_2^2}\mathbf{e}(k). \end{aligned} \tag{2.49}$$

Por fim, substituindo a Equação (2.50) em (2.47) e adicionando um passo de adaptação μ, chega-se

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{\|\mathbf{x}(k)\|_2^2}\mathbf{e}(k)\mathbf{x}(k). \tag{2.50}$$

A partir da Equação (2.50), pode-se observar que o algoritmo NLMS é na verdade um algoritmo de passo efetivo variável, cujo valor depende do passo de adaptação μ e de amostras instantâneas do sinal de entrada $\mathbf{x}$(k). Com isso, para valores de amostras do sinal de entrada muito pequenas ou iguais a zero, o algoritmo NLMS tende a divergir devido ao aumento excessivo do passo efetivo. Devido a isso, foi adicionado um parâmetro de regularização ϵ que possui um pequeno valor. A Equação (2.51) representa o algoritmo ϵ-NLMS.

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{\|\mathbf{x}(k)\|_2^2 + \epsilon}\mathbf{e}(k)\mathbf{x}(k) \tag{2.51}$$

A estabilidade e convergência do algoritmo NLMS é garantida para valores de μ dentro do intervalo

$$0 < \mu < 2. \tag{2.52}$$

Como pode-se observar na Inequação (2.52), o passo de adaptação do algoritmo NLMS não depende dos autovalores da matriz de autocorrelação, como acontece no algoritmo LMS. Em termos de desempenho, o algoritmo NLMS exibe uma taxa de convergência que é potencialmente mais rápida do que o algoritmo LMS, para dados de entrada não correlacionados e os correlacionados (DOUGLAS; MENG, 1994). Outro ponto a ser observado é que, assim como o algoritmo LMS, o algoritmo NLMS pode ser visto como a implementação estocástica de um algoritmo determinístico, sendo, nesse caso, o algoritmo de Newton (SAYED, 2003, p.225), (MANOLAKIS; INGLE; KOGON, 2005, p.607). Os passos para a implementação do algoritmo ϵ-NLMS são apresentados no Algoritmo 2.2.

Algoritmo 2.2 ϵ-NLMS

1: **Inicialização e parâmetros**

$$\mathbf{w}(0) = \mathbf{0}$$

$$0 < \mu < 2; \quad \epsilon > 0$$

2: **Dados de entrada e saída da planta e do filtro adaptativo**

$$d(k) = \mathbf{x}^T(k)\mathbf{p}(k)$$

$$y(k) = \mathbf{x}^T(k)\mathbf{w}^T(k)$$

3: **Cálculo do sinal do erro**

$$e(k) = d(k) - y(k) + v(k)$$

4: **Atualização dos coeficientes do filtro**

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{\epsilon + \mathbf{x}^T(k)\mathbf{x}(k)} e(k)\mathbf{x}(k)$$

2.7 Algoritmo GNGD

O algoritmo GNGD (em inglês *Generalized Normalized Gradient Descent*) representa uma extensão do algoritmo NLMS, onde é adicionado um parâmetro de regularização, ϵ, que é atualizado ao longo do tempo, através de um gradiente estocástico. Tal algoritmo foi proposto para solucionar problemas de desempenho presentes no algoritmo NLMS, provocados, por exemplo, pelo mal condicionamento da matriz de autocorrelação, passo de adaptação elevado, μ, e valores do vetor de entrada próximos a zero (MANDIC, 2004). A adaptação do parâmetro ϵ introduz características relevantes ao algoritmo GNGD, tais como: excelente estabilidade, robustez à pertubações dos parâmetros e convergência muito rápida (MANDIC, 2004).

2.7.1 Derivação do algoritmo GNGD

Como demostrado anteriormente, a atualização do vetor de pesos no algoritmo ϵ-NLMS é realizada por

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) + \frac{\mu}{\|\mathbf{x}(k)\|_2^2 + \epsilon}\mathbf{e}(k)\mathbf{x}(k) \\ &= \mathbf{w}(k) + \eta(k)\mathbf{e}(k)\mathbf{x}(k). \end{aligned} \tag{2.53}$$

Para a atualização do parâmetro de regularização ϵ, utiliza-se o gradiente adaptativo conforme abaixo

$$\epsilon(k+1) = \epsilon(k) - \rho\nabla\xi(k), \tag{2.54}$$

onde ρ é um parâmetro de pequeno valor e

$$\xi(k) = \frac{1}{2}\mathbf{e}^2(k) \tag{2.55}$$

Aplicando a regra da cadeia, o gradiente $\nabla\xi(k)$ pode ser representado por

$$\begin{aligned}\frac{\partial\xi(k)}{\epsilon(k)} =& \frac{\partial\xi(k)}{\mathbf{e}(k))}\frac{\partial\mathbf{e}(k)}{\mathbf{y}(k)}\frac{\partial\mathbf{y}(k)}{\mathbf{w}(k)}\frac{\partial\mathbf{w}(k)}{\eta(k)}\frac{\partial\eta(k)}{\epsilon(k)} \\ =& \frac{\mathbf{e}(k)\mathbf{e}(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{(\|\mathbf{x}(k-1)\|_2^2 + \epsilon(k))^2}.\end{aligned} \tag{2.56}$$

Substituindo a Equação (2.56) em (2.54), tem-se a seguinte relação recursiva

$$\epsilon(k+1) = \epsilon(k) - \rho\mu\frac{\mathbf{e}(k)\mathbf{e}(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{\|\mathbf{x}(k-1)\|_2^2 + \epsilon(k)} \tag{2.57}$$

O passo efetivo do algoritmo GNGD é essencialmente limitado pelos limites de estabilidade do passo de adaptação do algoritmo NLMS (MANDIC, 2004). Para encontrar o limite inferior do parâmetro de regularização ϵ, considere a condição de convergência uniforme

$$|\mathbf{e}(k+1)| \leq |1 - \eta(k)\|\mathbf{x}(k)\|_2^2\ \mathbf{e}(k)|. \tag{2.58}$$

Consequentemente

$$|1 - \eta(k)\|\mathbf{x}(k)\|_2^2| \leq 1, \tag{2.59}$$

que é equivalente a

$$0 < \frac{\mu}{\|\mathbf{x}(k)\|_2^2 + \epsilon(k)} < \frac{2}{\|\mathbf{x}(k)\|_2^2}. \tag{2.60}$$

Para $\mu = 1$, o limite inferior para a estabilidade do GNGD em relação a $\epsilon(k)$ é dado por

$$\epsilon(k) > -\frac{\|\mathbf{x}(k)\|_2^2}{2}. \tag{2.61}$$

O GNGD apresenta robustez para pertubações no parâmetro de regularização ϵ e na inicialização do passo de adaptação ρ. Possui também a propriedade desejável que é quase garantida a convergência, não importando quais são as propriedades estatísticas do ambiente. Entretanto, o GNGD não garante que em estado estacionário, para valores muito pequenos de erro, a atualização do parâmetro permanecerá em um valor fixo. Isso ocorre devido a característica do filtro permanecer em "alerta", a fim de reagir rapidamente a mudanças no ambiente (MANDIC et al., 2007). Os passos para a implementação do algoritmo GNGD são apresentados no Algoritmo 2.3.

Algoritmo 2.3 GNGD

1: **Inicialização e parâmetros**

$$\mathbf{w}(0) = \mathbf{0}$$

$$\epsilon(0) > 0$$

$$\mu > 0$$

2: **Dados de entrada e saída da planta e do filtro adaptativo**

$$d(k) = \mathbf{x}^T(k)\mathbf{p}(k)$$

$$y(k) = \mathbf{x}^T(k)\mathbf{w}^T(k)$$

3: **Cálculo do sinal do erro**

$$e(k) = d(k) - y(k) + v(k)$$

4: **Atualização dos coeficientes do filtro**

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{\epsilon(k) + \mathbf{x}^T(k)\mathbf{x}(k)} e(k)\mathbf{x}(k)$$

5: **Atualização do parâmetro de regularização**

$$\epsilon(k+1) = \epsilon(k) - \rho\mu \frac{\mathbf{e}(k)\mathbf{e}(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{\|\mathbf{x}(k-1)\|_2^2 + \epsilon(k)}$$

2.8 Exemplo de aplicação dos algoritmos adaptativos

Para avaliar o desempenho dos algoritmos adaptativos mencionados anteriormente, considere uma planta esparsa com $N = 100$ coeficientes. Os coeficientes ativos (não-nulos) desta planta esparsa estão localizados nas posições $\{1; 30; 35; 85\}$, com seus respectivos valores iguais $\{0.1, 1.0, -0.5, 0.1\}$. O sinal de entrada é correlacionado, com média zero e variância unitária, obtido através de um processo autorregressivo de ordem 2, AR(2), dado por

$$x(k) = 0,4x(k-1) - 0,4x(k-2) + v(k), \tag{2.62}$$

sendo $v(k)$ um ruído branco com média zero e variância $\sigma_b^2 = 0,77$. A dispersão da matriz de autocorrelação da entrada é $\chi(\mathbf{R}) = 10$. Os elementos da matriz $\mathbf{R}$ são calculados por

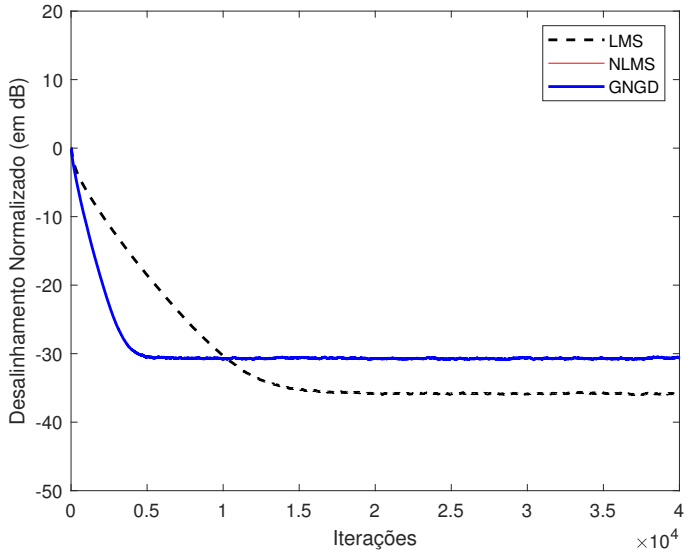
$$r(m) = \frac{\sigma_x^2 p_1(p_2^2 - 1)p_1^m}{(p_2 - p_1)(p_2 p_1 + 1)} - \frac{\sigma_x^2 p_2(p_1^2 - 1)p_2^m}{(p_2 - p_1)(p_1 p_2 + 1)}, \tag{2.63}$$

sendo m o *lag* entre as amostras da entrada, σ_x^2, de $x(k)$ e

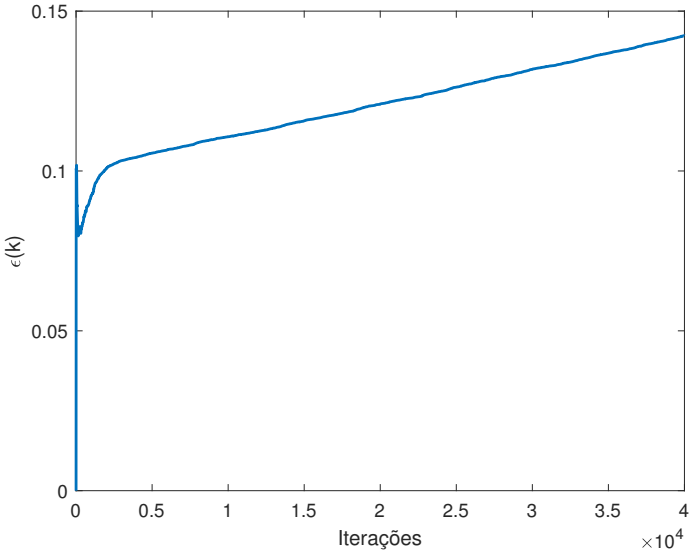
$$p_{1,2} = \frac{1}{2}\left(-a_1 \pm \sqrt{a_1^2 - 4a_2}\right). \tag{2.64}$$

O ruído de medição, $z(k)$, é branco e gaussiano com variância $\sigma_z^2 = 10^{-3}$.

No primeiro exemplo , os passos são definidos como $\mu_{lms} = 0.002$ e $\mu_{nlms} = \mu_{gngd} = 1$, onde os resultados são apresentados na Figura 2.7. Observa-se que os algoritmos NLMS e GNGD possuem o mesmo desempenho, apresentando maior velocidade de convergência e menor capacidade de rastreamento quando comparada ao algoritmo LMS. O parâmetro de regularização do algoritmo GNGD, mostrado na Figura 2.7(b), continua em atualização ao longo das iterações. Na simulação são consideradas 100 realizações independentes.



(a)



(b)

Figura 2.9 – Resultados obtidos no primeiro cenário. (a) Desempenho dos algoritmos adaptativos. (b) Dinâmica do parâmetro de regularização.

No segundo exemplo, os passos são definidos como $\mu_{lms} = 0.002$ e $\mu_{nlms} = \mu_{gngd} = 2.1$, onde os resultados são apresentados na Figura 2.8. Verifica-se que o algoritmo NLMS

diverge para este valor do passo de adaptação, conforme teoria apresentada. Entretanto, apesar do desempenho pior em relação ao exemplo anterior, o algoritmo GNGD ainda converge. Isto deve-se a maior robustez e menor dependência do passo de adaptação que este algoritmo possui (MANDIC, 2004).

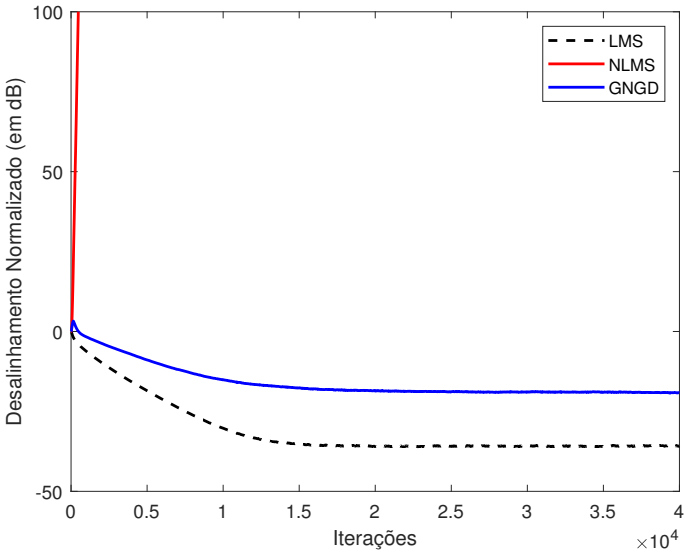


Figura 2.10 – Desempenho dos algoritmos adaptativos no segundo cenário.

Em ambas as simulações, verifica-se que a velocidade dos algoritmos é lenta, sendo necessárias, pelo menos, 5000 realizações para a convergência dos algoritmos NLMS e GNGD e 15000 realizações para o algoritmo LMS. Isto ocorre devido a planta analisada ser bastante esparsa, onde o grau de esparsidade calculado é $s(p) = 0.9435$. No capítulo a seguir, são introduzidos os algoritmos adaptativos tensoriais, nos quais possuem um melhor desempenho neste tipo de aplicação.

3 Algoritmos Adaptativos Tensoriais

Ao longo dos anos, diversos algoritmos adaptativos foram desenvolvidos com o objetivo de se beneficiar da característica esparsa que algumas plantas possuem. Um exemplo é o algoritmo PNLMS (em inglês *Proportionate Normalized Least Mean Square*), no qual utiliza um passo proporcional para cada coeficiente do filtro. Outro algoritmo que pode ser citado é o IPNLMS (em inglês *Improved Proportionate Normalized Least Mean Square*) que apresenta um melhor desempenho que o PNLMS em plantas com resposta ao impulso dispersiva. Entretanto, para a implementação destes algoritmos, é necessário um custo computacional maior quando comparado aos algoritmos adaptativos básicos (LMS e NLMS).

Neste capítulo, são desenvolvidos os algoritmos adaptativos tensoriais, que possuem maior velocidade de convergência quando comparado aos algoritmos adaptativos tradicionais, utilizando uma menor quantidade de lotes de memoria. A seguir, são apresentados alguns conceitos sobre a separabilidade de operadores lineares, uma propriedade de interesse quando lidamos com tensores (RUPP; SCHWARZ, 2015).

3.1 Operadores Linearmente Separáveis

Definição 1: Um operador $\mathbf{A}$ é dito ser separável se $\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2$ para algum $\mathbf{A}_1$ e $\mathbf{A}_2$.

Obviamente, como se trata de um produto, não há solução única, se ela existir. A seguir, é mostrado como isso pode ser alcançado no sentido do mínimos quadrados, dados vetores de comprimento M que assumem a posição de operadores lineares. Como exemplo, é utilizado um tensor bidimensional, mas a teoria pode ser estendida para tensores com dimensões superiores. A abordagem também considera que as matrizes são perfeitamente separáveis (LOAN; PITSIANIS, 1993).

3.1.1 Teorema I

Considere um vetor $\mathbf{w} \in \mathbb{R}^{(M\times 1)}$ com $M = NP$, ou seja, $\mathbf{w} = [\mathbf{w}_1;\ \mathbf{w}_2;\ \ldots\ ;\mathbf{w}_P]$ com $\mathbf{w}_l \in \mathbb{R}^{(N\times 1)}; l = 1, 2, ..., P$. Existem apenas dois vetores $\mathbf{a} \in \mathbb{R}^{(N\times 1)}$, $\mathbf{b} \in \mathbb{R}^{(P\times 1)}$ com $\|a\|_2 = 1$ tal que

$$\{\mathbf{a}, \mathbf{b}\} = arg \min_{a,b} = \|\mathbf{w} - \mathbf{b} \otimes \mathbf{a}\|_2^2. \tag{3.1}$$

3.1.1.1 Prova

Em um primeiro momento, calcula-se o elementos do vetor $\mathbf{b}$ através dos mínimos quadrados e obtém-se

$$\frac{\partial}{\partial \mathbf{b}_k} \sum_{l=1}^{P} \|\mathbf{w}_l - \mathbf{b}_l \mathbf{a}\|_2^2 = 0, \tag{3.2}$$

que pode ser reescrito por

$$\mathbf{b}_k = \frac{\mathbf{a}^T \mathbf{w}_k}{\mathbf{a}^T \mathbf{a}}; \ k = 1, 2, ..., P. \tag{3.3}$$

Substituindo a Equação (3.3) em (3.2), tem-se

$$\min_{\mathbf{a}} \sum_{l=1}^{P} \|\mathbf{w}_l - \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T \mathbf{a}} \mathbf{w}_l\|_2^2 = 0, \tag{3.4}$$

que é equivalente a

$$\min_{\mathbf{a}} \sum_{l=1}^{P} \mathbf{w}_l^T [I_N - \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T \mathbf{a}} \mathbf{w}_l] \ \mathbf{w}_l = 0. \tag{3.5}$$

A Equação (3.5) é minimizada quando o vetor $\mathbf{a}$ é igual ao autovetores gerados pelo maior autovalor da expressão abaixo

$$\mathbf{a} = arg \ max \ vec \sum_{l=1}^{P} \mathbf{w}_l \mathbf{w}_l^T. \tag{3.6}$$

3.1.2 Teorema II

Dados os vetores $\mathbf{a}$ e $\mathbf{b}$ apresentados no Teorema I, o vetor $\mathbf{w}$ pode ser representado por (RUPP; SCHWARZ, 2015)

$$\mathbf{w} = \mathbf{b} \otimes \mathbf{a} + \mathbf{v} \tag{3.7}$$

com o vetor de erro, $\mathbf{v}$, sendo ortogonal ao produto de Kronecker dos vetores $\mathbf{a}$ e $\mathbf{b}$

$$\mathbf{v}^T (\mathbf{b} \otimes \mathbf{a}) = 0. \tag{3.8}$$

3.1.2.1 Prova

Considere a Equação (3.4) reescrita conforme abaixo

$$\mathbf{b} \otimes \mathbf{a} = [I_P \otimes \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T \mathbf{a}}] \mathbf{w}. \tag{3.9}$$

O vetor de erro $\mathbf{v}$ é definido com

$$\mathbf{v} = \mathbf{w} - \mathbf{b} \otimes \mathbf{a}. \tag{3.10}$$

Aplicando o Teorema I, o vetor de erro pode ser reescrito como

$$\mathbf{v} = (I_M - I_P \otimes \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T\mathbf{a}})\ \mathbf{w}. \tag{3.11}$$

Escrevendo o vetor de erro na forma concatenada $\mathbf{v}_k = [\mathbf{v}_1; \mathbf{v}_2; ...; \mathbf{v}_P]; \mathbf{v} \in \mathbb{R}^{(N\times 1)}; k = 1, 2, ..., P$ encontra-se um $\mathbf{v}_k = [I_N - \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T\mathbf{a}}]\mathbf{w}_k$ que é ortogonal dentro da aproximação $\frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T\mathbf{a}}\mathbf{w}_k$, ou seja, no sentido dos mínimos quadrados.

3.1.3 Teorema III

A partir dos Teoremas I e II, pode-se calcular o mínimo MSE (MMSE) através de (RUPP; SCHWARZ, 2015)

$$MMSE = \sum_{k=1}^{P} \mathbf{v}_k^T\mathbf{v}_k = \sum_{k=1}^{P} \mathbf{w}_k(I_N - \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T\mathbf{a}})\mathbf{w}_k. \tag{3.12}$$

Se existirem várias partições de M = NP, vários valores para o MMSE podem ser verificados e com base nestes, pode ser decidido que particionamento é mais adequado para a separação linear.

3.2 LMS Tensor

Embora os algoritmos LMS e NLMS sejam frequentemente preferidos na prática devido as suas inúmeras propriedades positivas de implementação, uma vez que a esparsidade da planta aumenta, os algoritmos sofrem com uma lenta taxa de aprendizagem (RUPP; SCHWARZ, 2015). Devido a isso, foi proposto o algoritmo LMS tensor que utiliza a propriedade de separabilidade dos operadores lineares. Uma vez que essa propriedade de separabilidade é aplicada, um algoritmo do tipo gradiente pode ser derivado com aumento significativo da taxa de aprendizagem. A seguir, será demostrada a derivação do algoritmo LMS tensor.

3.2.1 Derivação do algoritmo LMS tensor

Para o desenvolvimento do algoritmo LMS tensor, considera-se que o vetor de resposta ao impulso da planta pode ser separado como $\mathbf{w}(k) = \mathbf{a}(k) \otimes \mathbf{b}(k)$, sendo $\otimes$ o produto de Kronecker. Partindo do mesmo princípio, o vetor de pesos estimado é representado por $\hat{\mathbf{w}}(k) = \hat{\mathbf{a}}(k) \otimes \hat{\mathbf{b}}(k)$. O comprimento do vetor $\mathbf{w}$(k) é dado por $M = M_aM_b$, onde M_a e M_b são, respectivamente, a dimensão do tensor $\mathbf{a}$(k) e $\mathbf{b}$(k). Assim, pode-se

representar o sinal desejado do filtro como

$$\begin{aligned}\mathbf{d}(k) &= \mathbf{w}^T(k)\mathbf{x}(k) \\ &= (\mathbf{b}(k) \otimes \mathbf{a}(k))\mathbf{x}(k) \\ &= \sum_{l=1}^{M_b} \mathbf{b}_l(k)\mathbf{a}^T(k)\mathbf{x}_l(k) \\ &= \mathbf{a}^T(k)\mathbf{X}(k)\mathbf{b}(k),\end{aligned} \tag{3.13}$$

onde $\mathbf{x}(k)$ é vetor de entradas e $\mathbf{X}(k)$ é a representação de $\mathbf{x}(k)$ na forma de uma matriz (tensor bidirecional). A Equação (3.13) pode ser reescrita como

$$\mathbf{a}^T(k)\mathbf{X}(k)\mathbf{b}(k) = \mathbf{v}^T(k)\mathbf{b}(k) = \mathbf{a}^T(k)\mathbf{z}(k) \tag{3.14}$$

onde $\mathbf{v}(k) = \mathbf{a}^T(k)\mathbf{X}(k)$ e $\mathbf{z}(k). = \mathbf{X}(k)\mathbf{b}(k)$

Derivando o algoritmo LMS agora em partições com relação a $\mathbf{a}(k)$ e $\mathbf{b}(k)$, obtém-se

$$\mathbf{e}(k) = \mathbf{d}(k) - \hat{\mathbf{a}}(k)\mathbf{X}(k)\hat{\mathbf{b}}(k). \tag{3.15}$$

$$\hat{\mathbf{a}}(k+1) = \hat{\mathbf{a}}(k) + \frac{\mu\hat{\mathbf{z}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2}.. \tag{3.16}$$

$$\hat{\mathbf{b}}(k+1) = \hat{\mathbf{b}}(k) + \frac{\mu\hat{\mathbf{c}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2}. \tag{3.17}$$

A análise de convergência pode-se tornar complicada devido à natureza em cascata dos algoritmos, onde o vetor $\mathbf{a}(k)$ depende do vetor $\mathbf{b}(k)$ e vice-versa. Algoritmos em cascata apresentam apenas robustez local e não global, mas se comportam bem no sentido do erro médio quadrático (MSE) (RUPP; SCHWARZ, 2015).

O algoritmo LMS tensor converge no sentido do MSE para valores de μ dentro do intervalo

$$0 < \mu < 2. \tag{3.18}$$

Percebe-se que o LMS tensor é na verdade um algoritmo com passo efetivo variável e normalizado, assim como o algoritmo NLMS. O intervalo de μ no qual os algoritmos convergem também são os mesmos. No entanto, a taxa de aprendizado, na maioria dos casos, é bastante diferente. A taxa de aprendizado, β, do algoritmo NLMS, para $\mu = 1$ é (RUPP, 1993), (SAYED, 2003)

$$\beta = \frac{20dB}{5M}, \tag{3.19}$$

e para o LMS tensor é dada por

$$\beta = \frac{20dB}{5(M_a + M_b)}. \tag{3.20}$$

Verifica-se, que na maioria dos casos, a parcela referente ao denominador da taxa de aprendizado do LMS tensor será menor que a do NLMS, garantindo assim um melhor desempenho na maioria das aplicações. Os passos para a implementação do algoritmo LMS tensor são apresentados no Algoritmo 3.1.

Algoritmo 3.1 LMS Tensor

1: **Inicialização e parâmetros**

$$\hat{\mathbf{a}}(0) \neq \mathbf{0}$$

$$\hat{\mathbf{b}}(0) \neq \mathbf{0}$$

$$0 < \mu < 2$$

2: **Dados de entrada e saída da planta e do filtro adaptativo**

$$d(k) = \mathbf{a}^T(k)\mathbf{X}(k)\mathbf{b}(k)$$

$$y(k) = \hat{\mathbf{a}}^T(k)\mathbf{X}(k)\hat{\mathbf{b}}(k)$$

3: **Cálculo do sinal do erro**

$$e(k) = d(k) - y(k) + v(k)$$

4: **Atualização dos coeficientes do filtro**

$$\hat{\mathbf{a}}(k+1) = \hat{\mathbf{a}}(k) + \frac{\mu\hat{\mathbf{z}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2}$$

$$\hat{\mathbf{b}}(k+1) = \hat{\mathbf{b}}(k) + \frac{\mu\hat{\mathbf{c}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2}$$

3.2.2 Extensões do LMS tensor

O algoritmo apresentado anteriormente requer apenas um tensor bidirecional, ou seja, uma matriz. Entretanto, a teoria mostrada pode ser estendida para tensores de dimensão n. Um tensor tridimensional, por exemplo, tem sua saída calculada conforme abaixo (RUPP; SCHWARZ, 2015)

$$\mathbf{y}(k) = \sum_{n=0}^{M_c-1}\sum_{m=0}^{M_b-1}\sum_{l=0}^{M_a-1}\mathbf{a}_l(k)\mathbf{b}_m(k)\mathbf{c}_n(k)\mathbf{x}_{(m+M_b l+M_a M_b n)}(k) \tag{3.21}$$

Neste caso, os vetores de regressão são simplesmente obtidos por uma dupla convolução.

Para avaliar o desempenho do algoritmo LMS tensor em relação ao algoritmo NLMS, considere uma planta com comprimento $M = 1024$. Em uma primeira simulação será utilizado $M_a = M_b = 32$. O vetor $b_k = 0.9^k$; $k = 0, 1, 2...., 31$ enquanto a_k é gerado por um processo aleatório de média zero e variância unitária. O vetor de entrada $\mathbf{x}(k)$ também

é gerado por um processo aleatório de média zero e variância unitária. Considera-se ainda um ruído de medição de média zero e variância 10^{-2}. Nas simulações foram realizadas 100 simulações independentes.

Na Figura 3.1, verifica-se que o LMS tensor apresenta velocidade de convergência muito maior que o NLMS. A diferença de desempenho entre os dois algoritmos é na ordem de 15 vezes. Nesta simulação é considerado o caso onde o vetor de pesos é perfeitamente separável. O passo de adaptação utilizado foi $\mu = 1$

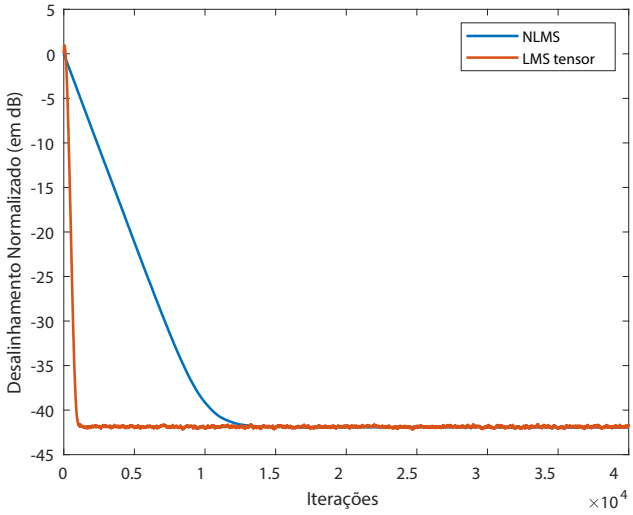


Figura 3.1 – Desempenho do LMS tensor (perfeitamente separável) e NLMS.

Na segunda simulação, utilizou-se as mesmas condições da simulação anterior, considerando agora o vetor de pesos não perfeitamente separável. Observa-se, na Figura 3.2, que o LMS tensor ainda possui melhor desempenho em termos de velocidade. No entanto, a capacidade de rastreamento do algoritmo é reduzida drasticamente. Os valores adicionados a resposta ao impulso para simular as condições descritas acima possuem média zero e variância 10^{-4} e são gerados aleatoriamente.

Na terceira simulação, realizou-se a análise do desempenho do algoritmo LMS tensor para diferentes valores do passo de adaptação. Percebe-se, pela Figura 3.3, que o melhor desempenho é alcançado para $\mu = 0.5$. À medida que μ fica mais próximo de 2, ou seja, o limite de estabilidade, o algoritmo tende a apresenta um desempenho mais pobre.

Por fim, utilizou-se diferentes dimensões para os vetores $\mathbf{a}$ e $\mathbf{b}$. Observa-se, na Figura 3.4, que quanto menor a soma das dimensões $(M_a + M_b)$, melhor é o desempenho do algoritmo. Isto ocorre devido a um aumento da taxa de aprendizado, conforme observa-se na Equação (3.20).

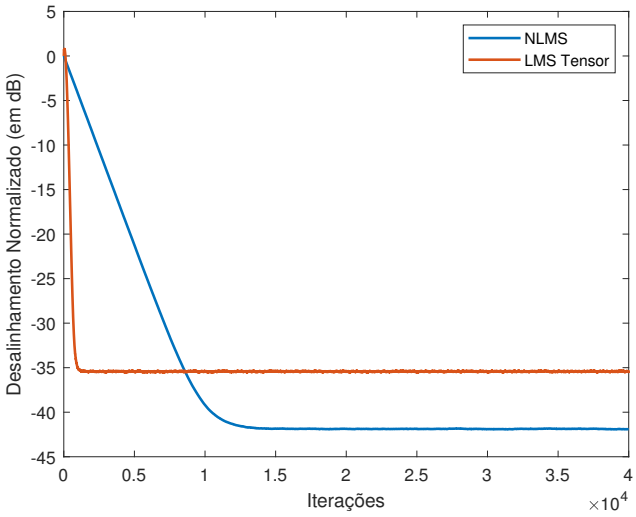


Figura 3.2 – Desempenho do LMS tensor (não perfeitamente separável) e NLMS.

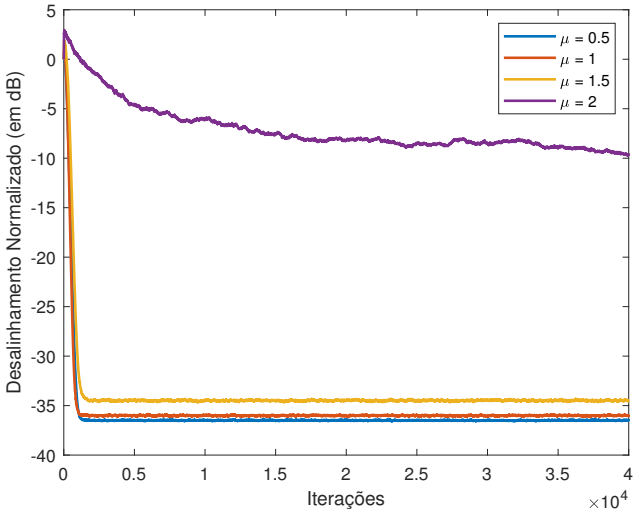


Figura 3.3 – Desempenho do LMS tensor para diferentes valores de μ.

3.3 Algoritmo GNGD Tensor

O algoritmo GNGD tensor utiliza os mesmos conceitos de operadores linearmente separáveis apresentados no desenvolvimento do LMS tensor. O vetor de pesos é descrito como $\mathbf{w}(k) = \mathbf{b}(k) \otimes \mathbf{a}(k)$, sendo sua dimensão representada por $M = M_a M_b$, onde M_a e M_b são, respectivamente, as dimensões do vetor $\mathbf{a}(k)$ e $\mathbf{b}(k)$. Assim, no GNGD tensor, a Equação (2.53) é substituída por

$$\hat{\mathbf{a}}(k+1) = \hat{\mathbf{a}}(k) + \frac{\mu \hat{\mathbf{z}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2 + \epsilon(k)} \tag{3.22}$$

$$\hat{\mathbf{b}}(k+1) = \hat{\mathbf{b}}(k) + \frac{\mu \hat{\mathbf{c}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2 + \epsilon(k)}, \tag{3.23}$$

onde $\epsilon(k)$ é dado por

$$\epsilon(k+1) = \epsilon(k) - \rho\mu \frac{\mathbf{e}(k)\mathbf{e}(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{\|\mathbf{x}(k-1)\|_2^2 + \epsilon(k)}. \tag{3.24}$$

O algoritmo GNGD é descrito em Algoritmo 3.2.

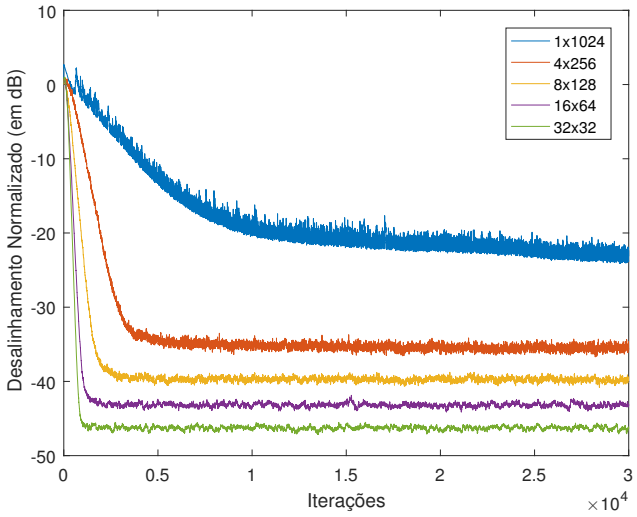


Figura 3.4 – Desempenho do LMS tensor para diferentes dimensões dos vetores **a** e **b**.

Em relação a forma padrão, o GNGD tensor apresenta uma melhoria na taxa de aprendizado em quase todos os casos. Comparado ao LMS tensor, apresenta maior robustez e menor sensibilidade a variações paramétricas, características presente também na sua forma padrão. Na Figura 3.5, é comparado o desempenho dos dois algoritmos tensoriais utilizando a planta descrita anteriormente. O passo de adaptação utilizado é μ = 1.99. Verifica-se que o desempenho dos algoritmos GNGD tensor é ligeiramente melhor que o LMS tensor. Isto ocorre devido ao μ escolhido está próximo do limite de estabilidade do LMS tensor, sendo este mais sensível a escolha deste parâmetro.

3.4 Complexidade computacional dos algoritmos

O custo computacional requerido para a implementação dos algoritmos adaptativos está diretamente relacionado ao número de coeficientes do filtro. No caso dos algoritmos adaptativos tradicionais apresentados (LMS, NLMS e GNGD), são necessários $2M$ lotes de memoria, onde M é a dimensão do filtro. Para os algoritmos adaptativos tensoriais, são necessários $2(M_a + M_b)$ lotes, sendo M_a e M_b, respectivamente, a dimensões do tensor **a** e **b**. Assim, com base na teoria apresentada anteriormente, os algoritmos adaptativos tensoriais exigem uma quantidade menor de lotes de memoria quando comparados aos algoritmos tradicionais, qualquer que sejam as dimensões dos tensores escolhidas. A Ta-

bela 3.1 detalha a quantidade de operações necessárias para a implementação de cada algoritmo citado.

Algoritmo 3.2 GNGD Tensor

1: **Inicialização e parâmetros**

$$\hat{\mathbf{a}}(0) \neq \mathbf{0}$$

$$\hat{\mathbf{b}}(0) \neq \mathbf{0}$$

$$\epsilon(0) > 0$$

$$\mu > 0$$

2: **Dados de entrada e saída da planta e do filtro adaptativo**

$$d(k) = \mathbf{a}^T(k)\mathbf{X}(k)\mathbf{b}(k)$$

$$y(k) = \hat{\mathbf{a}}^T(k)\mathbf{X}(k)\hat{\mathbf{b}}(k)$$

3: **Cálculo do sinal do erro**

$$e(k) = d(k) - y(k) + v(k)$$

4: **Atualização dos coeficientes do filtro**

$$\hat{\mathbf{a}}(k+1) = \hat{\mathbf{a}}(k) + \frac{\mu\hat{\mathbf{z}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2}$$

$$\hat{\mathbf{b}}(k+1) = \hat{\mathbf{b}}(k) + \frac{\mu\hat{\mathbf{c}}(k)\mathbf{e}(k)}{\|\hat{\mathbf{v}}(k)\|_2^2 + \|\hat{\mathbf{z}}(k)\|_2^2}$$

5: **Atualização do parâmetro de regularização**

$$\epsilon(k+1) = \epsilon(k) - \rho\mu\frac{\mathbf{e}(k)\mathbf{e}(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{\|\mathbf{x}(k-1)\|_2^2 + \epsilon(k)}$$

Tabela 3.1 – Complexidade computacional dos algoritmos adaptativos.

Operações	LMS	NLMS	GNGD	LMS Tensor	GNGD Tensor
Adições	$2M$	$3M$	$5M$	$3(M_a + M_b)$	$5(M_a + M_b)$
Multiplicações	$2M + 1$	$3M + 1$	$5M + 5$	$3(M_a + M_b) + 1$	$5(M_a + M_b) + 5$
Divisões	0	1	2	1	2
Comparações	0	0	0	0	0

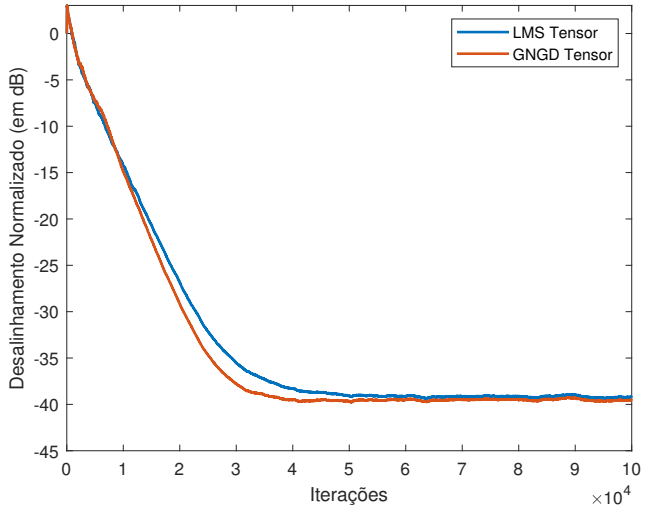


Figura 3.5 – Desempenho do LMS tensor para diferentes dimensões dos vetores **a** e **b**.

3.5 Estimação de harmônicas utilizando filtros adaptativos

3.5.1 Formulação matemática

De maneira geral, um sinal elétrico do sistema elétrico de potência pode ser representado por

$$y(t) = \sum_{i=1}^{N} A_i \text{sen}\left(i\omega t + \phi_i\right) + v(t), \tag{3.25}$$

onde A_i e ϕ_i são amplitude e fase, respectivamente, dos harmônicos, N representa o total de componentes harmônicos presentes no sinal e ω é a frequência angular da componente fundamental, $v(t)$ representa um ruído e t, o tempo.

A Equação (3.25) é representada na sua forma discreta por

$$y(k) = \sum_{i=1}^{N} A_i \text{sen}\left(i\omega k T_s + \phi_i\right) + v(kT_s) \tag{3.26}$$

onde T_s é o período de amostragem.

A Equação (3.26) pode ser reescrita utilizando expansão trigonométrica conforme

$$y(k) = [\sum_{i=1}^{N} A_i \text{sen}(\phi_\text{i})\text{cos}(2i\pi k T_s) + A_i \text{cos}(\phi_\text{i})\text{sin}(2i\pi k T_s)] + z(k) \tag{3.27}$$

Adequando a Equação (3.27) em um problema de filtragem adaptativa, obtém-se

$$\begin{aligned}\mathbf{x}(k) = &[\text{sen}(2\pi k T_s),\ \text{cos}(2\pi k T_s),\ \cdots, \\ &\text{sen}(2\pi N k T_s),\ \text{cos}(2\pi N k T_s)]^T\end{aligned} \tag{3.28}$$

e o vetor de pesos estimados por

$$\begin{aligned}\mathbf{w}(k) = [&A_1\cos\phi_1,\ A_1\mathrm{sen}\phi_1,\ \cdots,\ A_N\cos\phi_N,\\ &A_N\mathrm{sen}\phi_N,]^T.\end{aligned} \tag{3.29}$$

Assim, as amplitudes e fases das componentes harmônicas podem ser obtidas através, respectivamente, de

$$A_i = \sqrt{\mathrm{w}_{2i}^2 + \mathrm{w}_{2i-1}^2} \tag{3.30}$$

e

$$\phi_i = \arctan\left[\frac{\mathrm{w}_{2i}}{\mathrm{w}_{2i-1}}\right] \tag{3.31}$$

para $i = 1, 2, ..., N$.

3.5.2 Simulação

Considere o seguinte sinal elétrico

$$\begin{aligned}y(t) =&1.2\mathrm{sen}(\omega t + \pi/6) + 0.8\mathrm{sen}(3\omega t + \pi/3)\\ &+ 0.3\mathrm{sen}(5\omega t + \pi/8).\end{aligned} \tag{3.32}$$

Nesta simulação, considera-se a frequência fundamental de 60Hz. O passo de adaptação utilizado no processo de adaptação dos coeficientes do algoritmo NLMS e LMS tensor é $\mu = 0.1$. A frequência de amostragem, escolhida seguindo o critério de Nysquist, é de 2kHz. O espectro analisado é de 0 a 770Hz, ou seja, até décima primeira harmônica. As dimensões dos tensores utilizadas são $M_a = 2$ e $M_a = 6$.

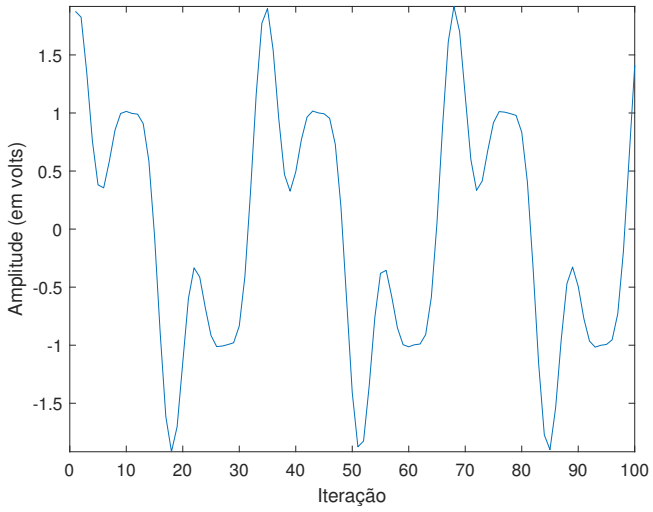
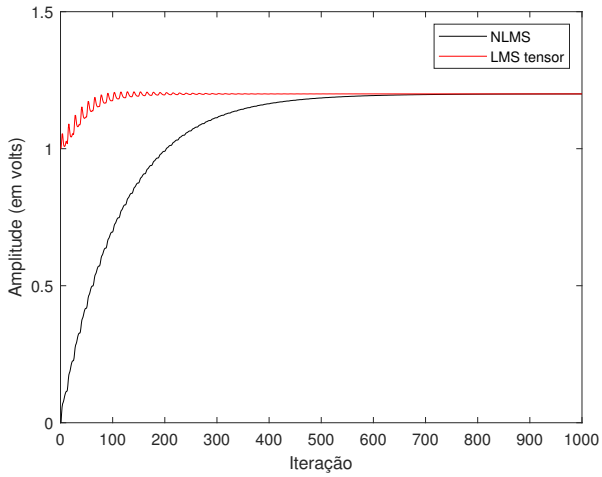


Figura 3.6 – Forma de onda resultante analisada.

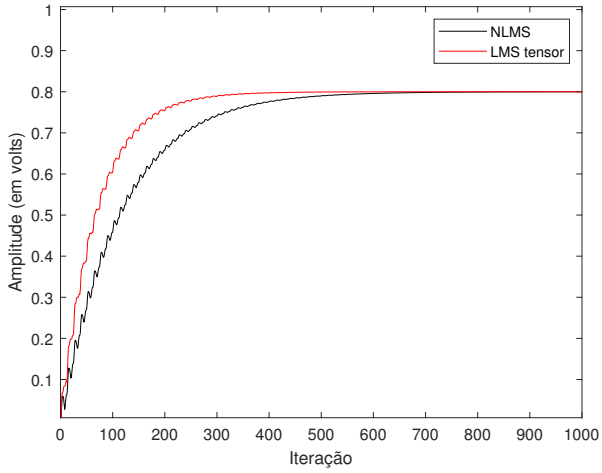
Na Figura 3.7, é mostrado o desempenho dos algoritmos adaptativos na estimação das amplitudes das harmônicas presentes no sinal analisado. Observa-se que o algoritmo LMS tensor apresenta uma convergência mais rápida quando comparado ao algoritmo NLMS. Em ambos os casos, o rastreamento das amplitudes das harmônicas é bastante satisfatório.

Na Figura 3.8, é mostrado o desempenho dos algoritmos para estimar as fases das componentes harmônicas e fundamental. Verifica-se que o desempenho dos dois algoritmos é bastante similar. Isto ocorre devido a fase não impactar de forma significativa no cálculo do erro, como ocorre com a amplitude.

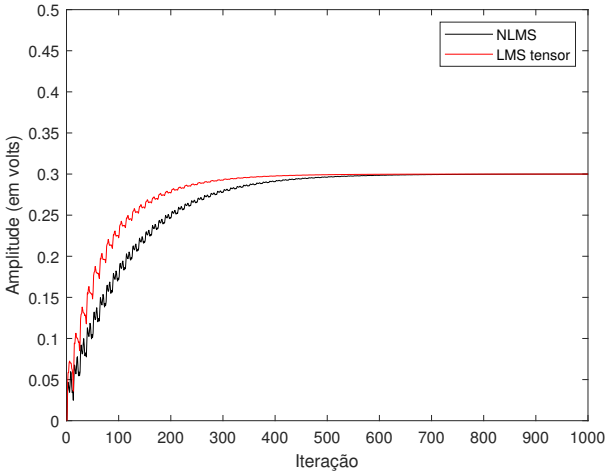
O grau de esparsidade calculado é de $s(\mathbf{p}) = 0.9790$. Assim, pode-se afirmar que a planta em questão é bastante esparsa. Isto justifica o melhor desempenho do algoritmo LMS tensor em relação ao NLMS.



(a)

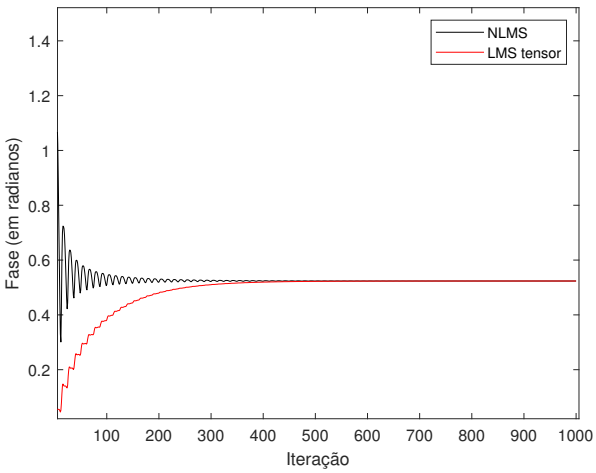


(b)

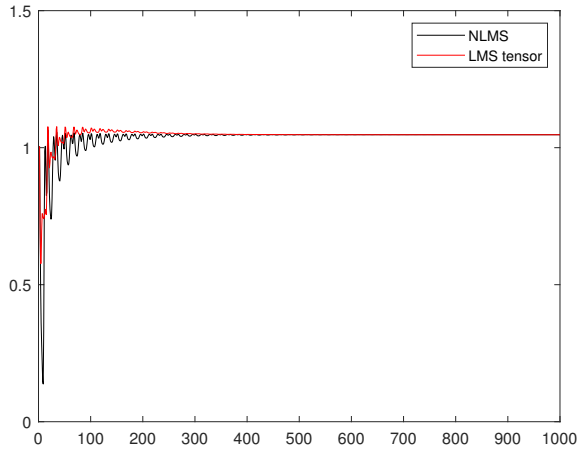


(c)

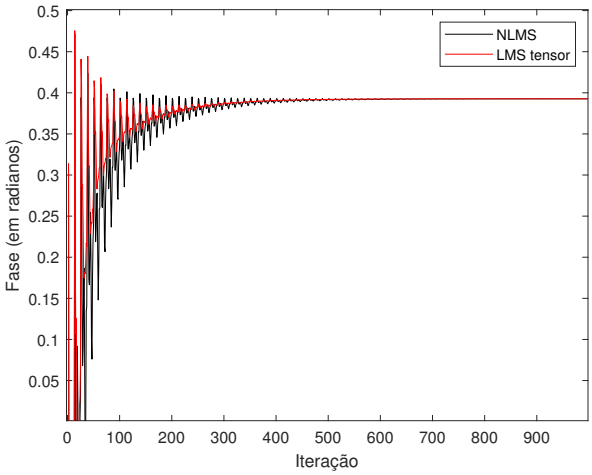
Figura 3.7 – Estimação das amplitudes das componentes fundamental e harmônicas. a) Componente fundamental. b) Terceira harmônica. c) Quinta harmônicas.



(a)



(b)



(c)

Figura 3.8 – Estimação das fases das componentes fundamental e harmônicas. a) Componente fundamental. b) Terceira harmônica. c) Quinta harmônica.

4 Redes Neurais Artificias

4.1 Introdução

Redes neurais artificiais (RNA) são modelos computacionais inspirados no sistema nervoso de seres vivos que possuem a capacidade de adquirir e manter conhecimento. Estruturalmente, são formadas por um conjunto de unidades de processamento, representado por neurônios artificiais, interligados por muitas interconexões, implementados por vetores e matrizes de pesos sinápticos (NUNES; SILVA, 2018, p.5). Devido a sua capacidade de aprender comportamentos complexos e serem altamente adaptáveis, as redes neurais possuem importância multidisciplinar, sendo aplicadas em diversas áreas, tais como engenharia, medicina, informática e matemática (LIVINGSTONE, 2008, p.3). Dentre outras vantagens, destacam-se a capacidade para modelar sistemas, o mapeamento de entrada/saída, obtido através do aprendizado dos exemplos de treinamento, e tolerância ao erro, devido à natureza distribuída da informação armazenada na rede neural (HAYKIN, 2009, p.4).

Para ilustrar o funcionamento de uma rede neural artificial, considere o modelo de um neurônio artificial proposto por Rosenblatt e conhecido como *perceptron*, mostrado na Figura 4.1. O perceptron é a forma mais simples de uma rede neural usada para a classificação de padrões considerados linearmente separáveis (HAYKIN, 2009, p.48). Percebe-se que para cada conjunto de dados de entradas, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, existe um conjunto de pesos associados, $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N$. A letra b representa o *bias*, que é um fator associado com o armazenamento de informações. A informação da rede neural é armazenado na forma de pesos e *bias* (KIM, 2017, p.20).

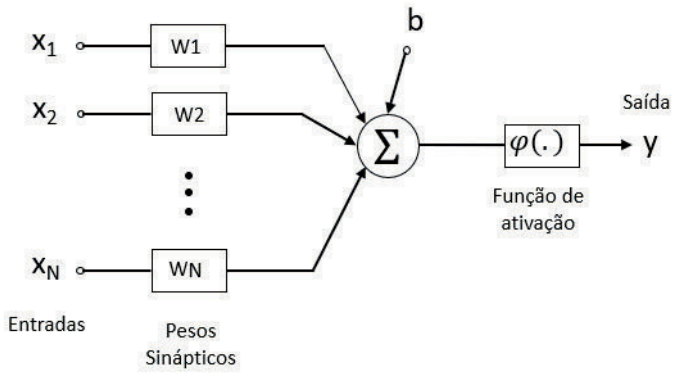


Figura 4.1 – Perceptron de Rosenblatt.

A partir da Figura 4.1, pode-se verificar que existe uma analogia entre o sistema

nervoso biológico e o sistema artificial proposto. O neurônio aqui é representado pelo nó e as conexões entre os neurônios são representados pelos pesos. Ambos os sistemas necessitam de um estimulo externo para realizar a aprendizagem, sendo que nas redes neurais artificiais este estimulo é fornecido pelos dados de treinamento que contêm exemplos de pares de entrada e saída da função a ser aprendida. O aprendizado ocorre com a atualização dos pesos ao longo do tempo (AGGARWAL, 2018, p.3).

A relevância de cada uma das entradas do neurônio artificial, apresentado na Figura 4.1, é calculada a partir da multiplicação com o peso sináptico correspondente, ponderando assim todas as informações externas que chegam ao neurônio (NUNES; SILVA, 2018, p.12). Assim, pode-se representar matematicamente o valor que chega ao nó somador por

$$\mathbf{v} = \sum_{i=0}^{N} \mathbf{x}_i \mathbf{w}_i + \mathbf{b}, \tag{4.1}$$

sendo N o número total de entradas aplicadas ao *perceptron*.

Por fim, o nó insere a soma ponderada na função de ativação e produz sua saída, dada por

$$\mathbf{y} = \phi(\mathbf{v}), \tag{4.2}$$

onde $\phi(.)$ representa a função de ativação.

As redes neurais artificiais diferenciam-se entre si pela sua arquitetura e pela forma como os pesos associados às conexões são ajustados durante o processo de aprendizagem (FERNEDA, 2006). A arquitetura de uma rede neural restringe o tipo de problema no qual poderá ser utilizada, e é definida pelo número de camadas (camada única ou múltiplas camadas), pelo número de nós em cada camada, pelo tipo de conexão entre os nós (*feedforward* ou *feedback*) e por sua topologia (HAYKIN, 2009, p.21).

4.2 Processos de Aprendizagem de uma Rede Neural Artificial

De forma análoga ao cérebro humano, as redes neurais artificias possuem diferentes maneiras de aprender com o ambiente externo no qual está inserido. Em um sentido amplo, podemos categorizar os processos de aprendizagem através dos quais as redes neurais funcionam da seguinte maneira: aprendizado com um professor e aprendizado sem um professor (HAYKIN, 2009, p.34). Da mesma forma, este último pode ser subcategorizado em aprendizado não supervisionado e aprendizado por reforço. Cada um desses processos de aprendizagem serão mais detalhados a seguir.

4.2.1 Aprendizado com Professor ou Supervisionado

O treinamento da rede através de aprendizado supervisionado é um processo que modifica o modelo para reduzir a diferença entre a saída da rede e a resposta desejada

(KIM, 2017, p.26). Esta modificação ocorre pelo ajuste dos pesos através da aplicação de ações comparativas, executadas pelo algoritmo de aprendizagem que supervisiona o erro gerado, usando essa informação no procedimento de ajuste. A rede neural artificial é considerada treinada quando o erro gerado está dentro de uma faixa de valor aceitável, levando em consideração os objetivos de generalização das soluções (NUNES; SILVA, 2018, p.26).

Para ilustrar o processo de aprendizado supervisionado, considere a Figura 4.5. Como pode-se observar, para aplicação do processo de aprendizagem supervisionado, há a necessidade de um professor que irá controlar todo o processo de treinamento da rede neural.

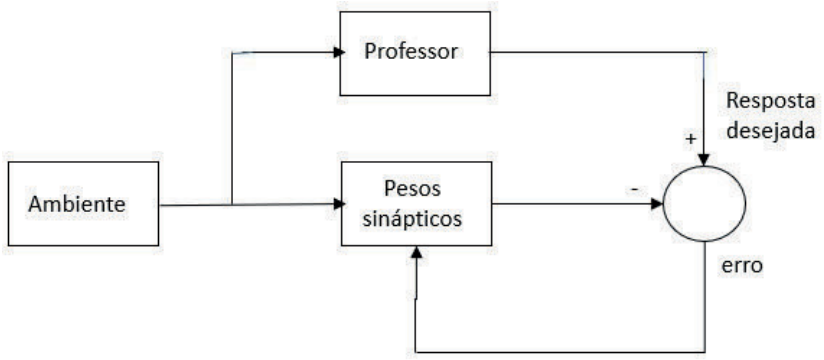


Figura 4.2 – Esquema de treinamento supervisionado de uma rede neural artificial.

4.2.2 Aprendizado sem Professor

4.2.2.1 *Aprendizado por Reforço*

No aprendizado por reforço, os algoritmos ajustam os parâmetros neurais internos que dependem de qualquer informação qualitativa ou quantitativa adquiridas através da interação com o sistema (ambiente) que está sendo mapeado, usando essas informações para avaliar o desempenho do aprendizado (NUNES; SILVA, 2018, p.27). Na Figura 4.6, observa-se o diagrama de blocos de uma forma de aprendizado por reforço de um sistema construído em torno de um crítico que converte um sinal de reforço primário recebido do ambiente em um sinal de reforço de alta qualidade chamado reforço heurístico de entrada escalar (BARTO; SUTTON; ANDERSON, 1983). O sistema é projetado para aprender sob reforço atrasado, o que significa que o sistema observa uma sequência temporal de estímulos também recebidos do meio ambiente, que acabam resultando na geração do sinal de reforço heurístico (HAYKIN, 2009, p.36).

4.2.2.2 *Aprendizado Não Supervisionado*

No aprendizado não supervisionado ou auto-organizado, não há professor ou crítico externo para supervisionar o processo de aprendizagem, conforme indicado na Figura 4.7.

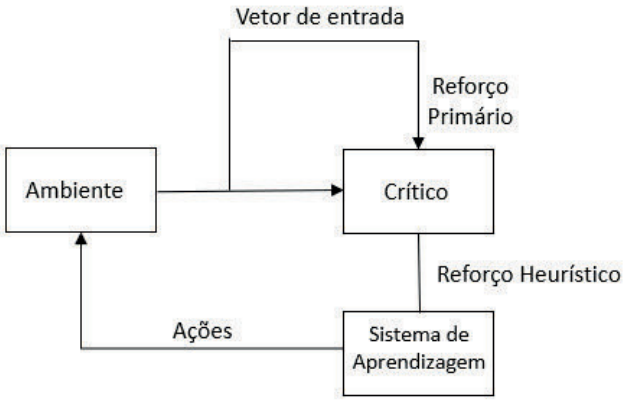


Figura 4.3 – Esquema do aprendizado por reforço.

Assim, a rede precisa se organizar quando houver particularidades existentes entre os elementos que compõem todo o conjunto de amostras, identificando subconjuntos que apresentam semelhanças. O algoritmo de aprendizado ajusta os pesos sinápticos e limites da rede, a fim de refletir os padrões extraídos dos conjuntos de dados na rede em si (NUNES; SILVA, 2018, p.26).

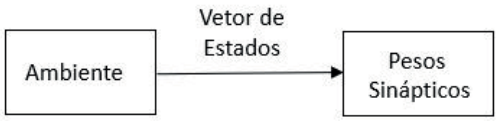


Figura 4.4 – Esquema do aprendizado não supervisionado.

4.3 Redes neurais Adaline e a regra do delta

A rede *Adaline* (em inglês *Adaptive Linear Element*) proposta por (WIDROW; HOFF, 1960) possui a mesma estrutura do perceptron, diferenciando apenas no algoritmo de treinamento. Neste caso, o processo de ajuste dos pesos é baseado em um algoritmo de aprendizado conhecido como regra do delta ou regra de Widrow-Hoff, e também conhecido como algoritmo LMS ou método de descida de gradiente.

Como demostrado anteriormente, agora de forma generalizada, os dados que chegam ao nó somador são dados por

$$\mathbf{v}_L(k) = \mathbf{w}_L(k)\mathbf{x}(k), \tag{4.3}$$

onde L representa, neste exemplo, a camada de saída, $\mathbf{w}_L$ é o vetor de pesos sinápticos conectados a camada L e $\mathbf{x}$ é o vetor de entradas. A saída da camada L na iteração k é dada por

$$\mathbf{y}_L(k) = \phi(\mathbf{v}_L(k)). \tag{4.4}$$

Assim, o erro instantâneo é dado por

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}_L(k), \tag{4.5}$$

onde $\mathbf{d}(k)$ é o vetor de resposta desejada no instante k.

A regra do delta aplica uma correção $\triangle \mathbf{w}_L$ aos pesos sinápticos $\mathbf{w}_L$, que é proporcional a $\partial\xi(k)/\mathbf{w}_L(k)$, onde ξ representa a função custo a ser minimizada, neste caso o MSE. De acordo com a regra da cadeia, pode-se expressar este gradiente conforme

$$\frac{\partial\xi(k)}{\partial\mathbf{w}_L(k)} = \frac{\partial\xi(k)}{\partial\mathbf{e}(k)}\frac{\partial\mathbf{e}(k)}{\partial\mathbf{y}_L(k)}\frac{\partial\mathbf{y}_L(k)}{\partial\mathbf{v}_L(k)}\frac{\partial\mathbf{v}_L(k)}{\partial\mathbf{w}_L(k)}. \tag{4.6}$$

Diferenciando a parcela $\partial\xi(k)/\partial\mathbf{e}(k)$, tem-se

$$\frac{\partial\xi(k)}{\partial\mathbf{e}(k)} = \mathbf{e}(k). \tag{4.7}$$

Em seguida, diferenciando a Equação (4.10) em relação a $\mathbf{y}_L$, obtém-se

$$\frac{\partial\mathbf{e}(k)}{\partial\mathbf{y}_L(k)} = -1. \tag{4.8}$$

Continuando, diferenciando a equação (4.9) em relação a v_L, tem-se

$$\frac{\partial\mathbf{y}_L(k)}{\partial\mathbf{v}_L(k)} = \phi^{'}(\mathbf{v}_L(k)). \tag{4.9}$$

Por fim, diferenciando a Equação (4.8) em relação a $\mathbf{w}_L$, obtém-se

$$\frac{\partial\mathbf{v}_L(k)}{\partial\mathbf{w}_L(k)} = \mathbf{x}(k). \tag{4.10}$$

Substituindo as Equações (4.12) a (4.15) em (4.11), obtém-se

$$\frac{\partial\xi(k)}{\partial\mathbf{w}_L(k)} = -\mathbf{e}(k)\phi^{'}(\mathbf{v}_L(k))\mathbf{x}(k). \tag{4.11}$$

A correção $\triangle \mathbf{w}_L$ aplicada aos pesos sináptico $\mathbf{w}_L$ é definida pela regra do delta, conforme

$$\triangle\mathbf{w}_L(k) = -\alpha\delta_L(k)\mathbf{x}(k), \tag{4.12}$$

onde α é a taxa de aprendizado e δ_L é o gradiente local da camada L definido por

$$\delta_L(k) = \mathbf{e}(k)\phi^{'}(\mathbf{v}_L(k)). \tag{4.13}$$

Assim, a regra de atualização dos pesos sinápticos é dada por

$$\begin{aligned} w_L(k+1) &= w_L(k) - \triangle w_L(k) \\ &= w_L(k) + \alpha\delta_L(k)x(k). \end{aligned} \tag{4.14}$$

Os passos para a implementação da regra do algoritmo de Widrow-Hoff são apresentados em Algoritmo 4.1

Algoritmo 4.1 Regra de Widrow-Hoff

1: **Inicialização e parâmetros**

$$\mathbf{w}(0) \text{ é inicializado com valores aleatórios}$$

$$\alpha < 1$$

2: **Cálculo da saída da rede neural**

$$\mathbf{v}(k) = \mathbf{x}(k)\mathbf{w}(k)$$

$$\mathbf{y}(k) = \phi(\mathbf{v}(k))$$

3: **Cálculo do sinal do erro**

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$

4: **Cálculo do gradiente local**

$$\delta(k) = \mathbf{e}(k)\phi'(\mathbf{v}(k))$$

5: **Atualização dos pesos sinápticos**

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \alpha\delta(k)\mathbf{x}(k)$$

4.4 Aprendizado offline

No aprendizado offline, também chamado de aprendizado em lote (em inglês *Batch learning*), os ajustes nos vetores de peso sináptico são executados após a apresentação de todo o conjunto de treinamento, sendo que cada etapa de ajuste leva em consideração o número de erros observados a partir da comparação da saída da rede neural artificial em relação aos valores desejados. Portanto, as redes neurais artificiais que utilizam aprendizado offline necessitam, pelo menos, de uma época de treinamento para executar uma etapa de ajuste em seus pesos e limites.

O método offline calcula a atualização dos pesos sinápticos através de

$$\triangle w_L = \frac{1}{N}\sum_{k=1}^{N} \triangle w_L(k), \tag{4.15}$$

onde $\triangle w_L$ é a atualização do vetor de pesos para o k-ésimo dado de treinamento e N é o número total de dados de treinamento.

Com o método de descida de gradiente usado para realizar o treinamento, o método offline possui algumas vantagens, dentre elas pode-se citar: estimativa precisa do vetor gradiente, garantindo, em condições simples, a convergência para um mínimo local e paralelização do processo de aprendizagem. Dentre as suas desvantagens, pode-se citar o

grande custo computacional necessário para a sua aplicação. Em um contexto estatístico, a aprendizagem offline pode ser vista como uma forma de inferência estatística. Portanto, é adequada para resolver problemas de regressão não linear (HAYKIN, 2009, p.128).

4.5 Aprendizado online

Ao contrário do aprendizado offline, no aprendizado online os ajustes nos pesos sinápticos são realizados após a apresentação de cada amostra de treinamento. Assim, após executar a etapa de ajuste, a respectiva amostra pode ser descartada (NUNES; SILVA, 2018, p.27). A atualização dos pesos sinápticos é realizada através de

$$\triangle \mathbf{w}_L = \alpha \delta_L(k)\mathbf{x}(k). \tag{4.16}$$

Observando as Equações (4.20) e (4.21), verifica-se que no aprendizado online são realizados N treinamentos em cada época, enquanto no aprendizado offline ocorre apenas 1 treinamento por época.

O aprendizado online é referido como um método estocástico, devido aos dados serem apresentados a rede neural de maneira aleatória. Isso torna menos provável que o processo de aprendizagem fique preso em um mínimo local, o que é uma vantagem em relação ao aprendizado offline (HAYKIN, 2009, p.128). Além disso, o aprendizado online exige um menor custo computacional que o aprendizado offline.

O aprendizado online geralmente é usado quando o comportamento do sistema mapeado muda rapidamente, portanto a adoção do aprendizado offline é quase impraticável porque as amostras usadas em um determinado momento podem não representam o comportamento do sistema em momentos posteriores.

Para avaliar o desempenho dos dois métodos de aprendizagem apresentados, considere uma rede neural artificial com arquitetura de camada única, cujo os dados de treinamento e a resposta desejada são representadas pelo vetor $\mathbf{x}$ e $\mathbf{d}$, respectivamente.

$$\mathrm{x} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathrm{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ . \end{bmatrix}$$

Para a função de ativação é utilizada a função sigmoidal e para a taxa de aprendizagem considera-se $\alpha = 0.9$. O problema consiste em aplica a regra do delta para a aprendizagem online e offline, analisado o erro médio obtido em cada processo.

Verifica-se, pela Figura 4.13, que o aprendizado online converge para zero mais rapidamente que o aprendizado offline. Além disso, a saída da rede neural que utiliza o

aprendizado online é mais próxima da resposta desejada que quando aplicado o aprendizado offline, como pode-se verifica abaixo

$$y_{on} = \begin{bmatrix} 0,023 \\ 0,0189 \\ 0,984 \\ 0,981 \end{bmatrix} \quad y_{off} = \begin{bmatrix} 0,0484 \\ 0,0392 \\ 0,968 \\ 0,960. \end{bmatrix}$$

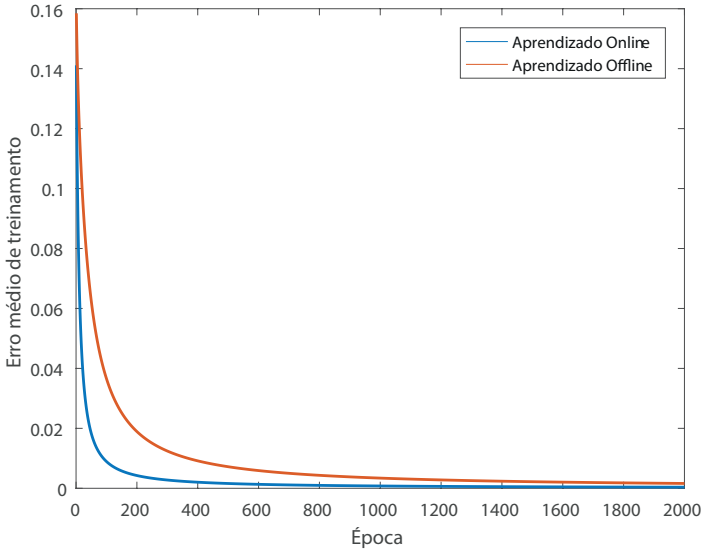


Figura 4.5 – Comparação entre o aprendizado online e offline.

4.6 Algoritmo *backpropagation*

A popularidade do aprendizado online para o treinamento supervisionado de redes neurais multicamadas foi aprimorada ainda mais pelo desenvolvimento do algoritmo *backpropagation* (GOODFELLOW et al., 2016, p.203). Este algoritmo permite calcular o erro que se propaga durante o treinamento da rede neural sem a utilização da resposta desejada nas camadas intermediárias. Este sinal de erro é determinado recursivamente, retrocedendo em termos dos sinais de erro de todos os neurônios da rede neural.

Para o desenvolvimento do algoritmo de retropropagação, considere que o gradiente local $\delta_L(k)$ da camada intermediária L pode ser representado por

$$\begin{aligned} \delta_L(k) &= -\frac{\partial \xi(k)}{\mathbf{y}_L(k)} \frac{\partial \mathbf{y}_L(k)}{\partial \mathbf{v}_L(k)} \\ &= -\frac{\partial \xi(k)}{\mathbf{y}_L(k)} \phi'(\mathbf{v}_L(k)), \end{aligned} \tag{4.17}$$

onde a Equação (4.14) foi utilizada na segunda linha. O problema aqui consiste em resolver a derivada parcial $\partial\xi(k)/\partial\mathbf{y}_L(k)$. Para isso, considere que a função custo representada por

$$\xi(k) = \frac{1}{2}\mathbf{e}_i^2(k), \tag{4.18}$$

onde i representa o nó de saída, pode ser reescrita conforme

$$\frac{\partial\xi(k)}{\partial\mathbf{y}_L(k)} = \mathbf{e}_i(k)\frac{\partial\mathbf{e}_i(k)}{\partial\mathbf{y}_L(k)}. \tag{4.19}$$

Aplicando a regra da cadeia na Equação (4.24), tem-se

$$\frac{\partial\xi(k)}{\partial\mathbf{y}_L(k)} = e_i^2(k)\frac{\partial e_i(k)}{\partial\mathbf{v}_i(k)}\frac{\partial\mathbf{v}_i(k)}{\partial\mathbf{y}_L(k)} \tag{4.20}$$

O erro no neurônio i é representado por

$$\begin{aligned}\mathbf{e}_i(k) =& \mathbf{d}(k) - \mathbf{y}_i(k)\\ =& \mathbf{d}(k) - \phi(\mathbf{v}_i(k)).\end{aligned} \tag{4.21}$$

Assim, tem-se

$$\frac{\partial\mathbf{e}_i(k)}{\partial\mathbf{v}_i(k)} = -\phi^{'}(\mathbf{v}_i(k)). \tag{4.22}$$

A partir da Equação (4.8), tem-se

$$\frac{\partial\mathbf{v}_i(k)}{\partial\mathbf{y}_L(k)} = \mathbf{w}_L(k). \tag{4.23}$$

Substituindo as Equações (4.28) e (4.29) em (4.25), chega-se em

$$\begin{aligned}\frac{\partial\xi(k)}{\partial\mathbf{y}_L(k)} =& -\mathbf{e}_i(k)\phi^{'}(\mathbf{v}_i(k))\mathbf{w}_L(k)\\ =& -\delta_i(k)\mathbf{w}_L(k).\end{aligned} \tag{4.24}$$

Por fim, substituindo a Equação (4.30) em (4.22), obtém-se a formula de retropropagação do gradiente local $\delta_L(k)$, descrita por

$$\delta_L(k) = \phi^{'}(\mathbf{v}_i(k))\delta_i(k)\mathbf{w}_L(k). \tag{4.25}$$

Percebe-se que a implementação do algoritmo de retropropagação ocorre em duas etapas: fase de avanço (em inglês *Forward phase*), na qual os dados se propagam de forma direta até o cálculo do erro na saída da rede neural, e fase reversa (em inglês *Backward phase*) que é responsável pela atualização do vetor de pesos sinápticos a partir da determinação dos deltas em cada neurônio das camadas intermediárias. Caso a camada L represente uma camada de saída, o δ continua sendo calculado a partir da Equação (4.18).

4.7 Momentum

O parâmetro *momentum* é uma das variações mais simples do algoritmo de retropropagação, pois requer apenas a adição de um único termo na equação de ajuste, que irá refletir sobre o quanto os pesos sinápticos serão alterados entre duas iterações sucessivas (NUNES; SILVA, 2018, p.71). O uso do termo *momentum* leva o ajuste de peso a um certa direção, até certo ponto, ao invés de produzir uma mudança imediata (KIM, 2017, p.65).

A atualização do pesos sinápticos na camada L aplicando o parâmetro *momentum* pode ser calculada por

$$\mathbf{w}_L(k+1) = \mathbf{w}_L(k) + \beta[\triangle w_L(k-1)] + \alpha\delta_L(k)\mathbf{y}_{(L-1)}(k), \tag{4.26}$$

onde β é a constante *momentum*.

Quando o parâmetro de *momentum* é igual a zero, a expressão torna-se equivalente ao algoritmo *backpropagaation* convencional. Por outro lado, para valores diferentes de zero, o termo do *momentum* se torna relevante, e sua contribuição afetará positivamente o processo de convergência. Assim, o pesos sinápticos não são afetados apenas por uma determinado valor de atualização, mas também pelo parâmetro de momentum que cresce a cada iteração, o que provoca uma melhoria na taxa de aprendizado (GALLANT, 1993, p.221).

Para melhor avaliar a influência do parâmetro *momemtum* no desempenho do algoritmo *backpropagation*, considere o *XOR problem* apresentado anteriormente na seção 4.8. Nesta simulação, é adicionado o parâmetro *momentum* para verificar a sua influência no desempenho do algoritmo. O valor da constante utilizado é $\beta = 0.9$.

Observa-se, na Figura 4.13, que a adição do parâmetro *momentum* melhorou de maneira significativa a velocidade de convergência do algoritmo de retropropagação, além de fornece uma saída mais próxima da resposta desejada, como verifica-se abaixo.

$$y_{mom} = \begin{bmatrix} 0.0014 \\ 0.9970 \\ 0.9965 \\ 0.0041 \end{bmatrix}.$$

4.8 Processo de generalização de uma rede neural

No processo de aprendizagem de uma rede neural, geralmente separa-se a maior parte do conjunto de dados disponíveis para o treinamento, com o objetivo de melhorar a capacidade de generalização. Uma rede neural é dita generalizada se o mapeamento de entrada e saída calculado estiver correto (ou quase isso) para dados de teste nunca utilizados durante o processo de treinamento (HAYKIN, 2009, p.147). Os dados de teste,

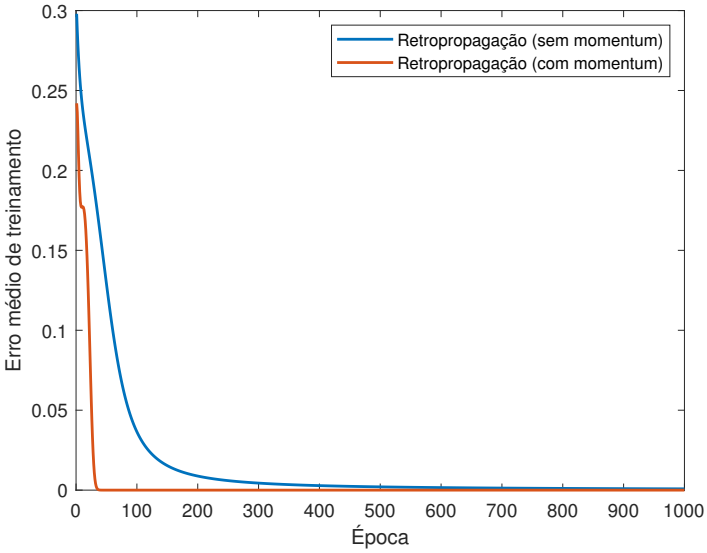


Figura 4.6 – Desempenho do algoritmo de retropropagação com e sem o parâmetro *momentum*.

aqui mencionados, são obtidos a partir dos mesmos conjuntos de dados utilizados no treinamento, possuindo valores diferentes.

Durante o processo de treinamento, a rede neural pode apresentar problemas de generalização caso os hiperparâmetros utilizados não sejam os adequados para o tipo de problema proposto. Dentre esses hiperparâmetros pode-se citar: o número de camadas ocultas, o número de neurônios em cada camada oculta e o número de épocas de treinamento. Os problemas mais comuns de generalização que uma rede neural pode apresentar são o *overfitting* e *underfitting*. O *overfitting* ocorre quando a rede neural torna-se "especialista"no conjunto de dados de treinamento utilizado, o que causa a perda da capacidade de generalização de padrões de entrada e saídas semelhantes (HAYKIN, 2009, p.147). O *underfitting* ocorre quando a rede neural não consegue aprender a partir dos dados de treinamento, sendo o erro muito elevado nesta etapa.

Uma técnica simples para evitar problemas de *underfitting* e *overfitting* é a validação cruzada (em inglês *Cross validation*). Para isso, o conjunto de dados disponíveis é dividido em dados de treinamento e validação. Com base no dados de validação é possível verificar o desempenho da rede neural para diferentes estruturas, além de verificar o seu desempenho ao longo de cada época.

Um procedimento simples que pode ser usado juntamente com os métodos de validação cruzada é o parada antecipada (em inglês *early stopping*), na qual o processo de aprendizado de uma topologia candidata é monitorado continuamente e o desempenho da rede é avaliado com relação às amostras do subconjunto de validação. O processo de treinamento é interrompido quando o erro quadrático médio começa a aumentar na etapa de validação entre épocas sucessivas. A figura abaixo ilustra de forma detalhada esse processo (NUNES; SILVA, 2018, p.104). O ponto cheio presente na figura representa o melhor

momento para parar o treinamento, com o objetivo de evitar *undefitting* e *overfitting*.

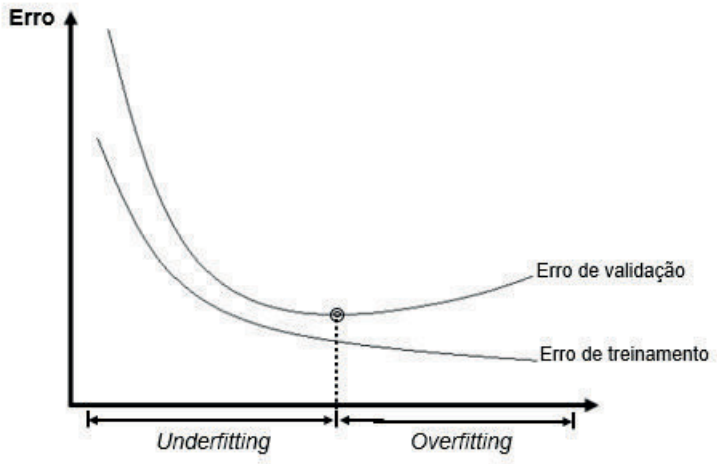


Figura 4.7 – Ilustração da regra de parada antecipada baseada no método de validação cruzada.

5 Redes Neurais Artificiais Tensoriais

Como demostrado anteriormente, a regra de aprendizagem para atualização dos pesos sinápticos é baseado na regra de Widrow-Hoff. Esta regra pode ser vista como uma variação do algoritmo LMS, onde são inseridas características não lineares a partir da função de ativação. Assim como ocorre com o algoritmo LMS aplicado em filtragem adaptativa, a regra de Widrow-Hoff apresenta um baixa velocidade de convergência em aplicações com elevado número de pesos sinápticos. Diversas são as aplicações de redes neurais que necessitam de uma grande quantidade de parâmetros para serem aplicadas. Dentre os vários exemplos, pode-se citar o reconhecimento de voz, processamento de texto, classificação de imagens e visão computacional (NOVIKOV et al., 2015).

Nesta seção, é desenvolvida uma nova regra de aprendizagem para redes neurais artificiais, utilizando o conceito de separabilidade de operadores lineares, apresentados nos capítulos anteriores. Este novo algoritmo utiliza uma menor quantidade de lotes de memoria e possui maior velocidade de aprendizagem quando comparada a regra de Widrow-Hoff. Para simulação do novo algoritmo, são considerados problemas de identificação de sistemas, onde a estimação de harmônicas em sistemas elétricos de potência está inserida.

5.1 Desenvolvimento do algoritmo de aprendizagem tensorial

Considerando que os pesos sinápticos conectados a L-ésima camada podem ser separados conforme

$$\mathbf{W}_L(k) = \mathbf{B}_L(k) \otimes \mathbf{A}_L(k) \tag{5.1}$$

com $\otimes$ representando o produto de Kronecker. Assumindo que $\mathbf{W} \in \mathbb{R}^{M \times N}$, sendo M o número total de dados de entrada inseridos a cada ciclo de treinamento na rede neural e N o número de nós presente na camada L. Assim, obtém-se $\mathbf{A} \in \mathbb{R}^{M_A \times N_A}$ e $\mathbf{B} \in \mathbb{R}^{M_B \times N_B}$, onde $M = M_A M_B$ e $N = N_A N_B$. Considerando que o vetor de entradas $x \in \mathbb{R}^M$ pode ser representado por um tensor bidimensional $X(k)$, cuja dimensões são dadas por M_A e M_B. Assim, a soma ponderada que saem da camada L é dada por

$$\begin{aligned} \mathbf{V}_L(k) &= \mathbf{W}_L(k)x(k) + b \\ &= (\mathbf{B}_L(k) \otimes \mathbf{A}_L(k))x(k) + b \\ &= \mathbf{A}_L(k)^T \mathbf{X}(k) \mathbf{B}_L(k) + b. \end{aligned} \tag{5.2}$$

A Equação (5.2) pode ser formulada também a partir de

$$\begin{aligned} \mathbf{V}_L(k) &= \mathbf{P}_L(k)\mathbf{B}_L(k) \\ &= \mathbf{A}_L^T(k)\mathbf{Q}_L(k). \end{aligned} \tag{5.3}$$

com

$$\mathbf{P}_L(k) = \mathbf{A}_L^T(k)\mathbf{X}(k), \tag{5.4}$$

e

$$\mathbf{Q}_L(k) = \mathbf{X}(k)\mathbf{B}_L(k). \tag{5.5}$$

A saída da camada L é dada por

$$\mathbf{Y}_L(k) = \varphi(\mathbf{V}_L(k)), \tag{5.6}$$

onde $\phi(.)$ é a função de ativação.

Assim, a regra de atualização dos tensores $\mathbf{A}$ e $\mathbf{B}$ para o k-ésimo dado de treinamento, obtida a partir da derivação da regra de Widrow-Hoff, é dada por

$$\mathbf{A}_L(k+1) = \mathbf{A}_L(k) + \frac{\alpha\mathbf{Q}_L(k)\delta_L(k)}{\|\mathbf{P}_L(k)\|_2^2 + \|\mathbf{Q}_L(k)\|_2^2} \tag{5.7}$$

$$\mathbf{B}_L(k+1) = \mathbf{B}_L(k) + \frac{\alpha\mathbf{P}_L(k)\delta_L(k)}{\|\mathbf{P}_L(k)\|_2^2 + \|\mathbf{Q}_L(k)\|_2^2}, \tag{5.8}$$

onde δ_L representa o gradiente local. A partir do algoritmo backpropagation, δ_L é dado por

$$\delta_L(k) = \varphi'(\mathbf{V}_L(k))[d(k) - \mathbf{B}_L^T(k)\mathbf{X}(k)\mathbf{A}_L(k)], \tag{5.9}$$

caso a camada L seja a camada de saída, ou

$$\delta_L(k) = \varphi'(\mathbf{V}_L(k))\ [\mathbf{B}_{L+1}(k)\delta_{L+1}(k)\mathbf{A}_{L+1}^T(k)], \tag{5.10}$$

caso a camada L seja uma camada intermediária. O índice $L+1$ refere-se a primeira camada após L. O esquema em diagrama de blocos da metodologia proposta é apresentado na Figura 5.1.

5.2 Complexidade Computacional dos algoritmos

Em relação ao custo computacional do algoritmo proposto, verifica-se que a complexidade algorítmica pode estar bem abaixo daquela exigida pela metodologia clássica. A quantidade de memória usada por uma rede neural totalmente conectada para atualizar pesos sinápticos usando a regra de Widrow-Hoff é de $2MN$. No caso da rede neural tensorial, a quantidade necessária é $2[(M_AN_A) + (M_BN_B)]$. Para ilustrar a quantidade de memória que pode ser salva, considera-se uma rede neural onde $M = N = 2^{2k}$. Portanto, o particionamento ideal é dado $M_AN_A = M_BN_B = 2^{2k}$ que reduz a complexidade computacional em 2^{2k-1} posições de memoria.

A quantidade de operações necessárias para a implementação do algoritmo proposto depende de fatores, tais como o número de camadas ocultas e as funções de ativação

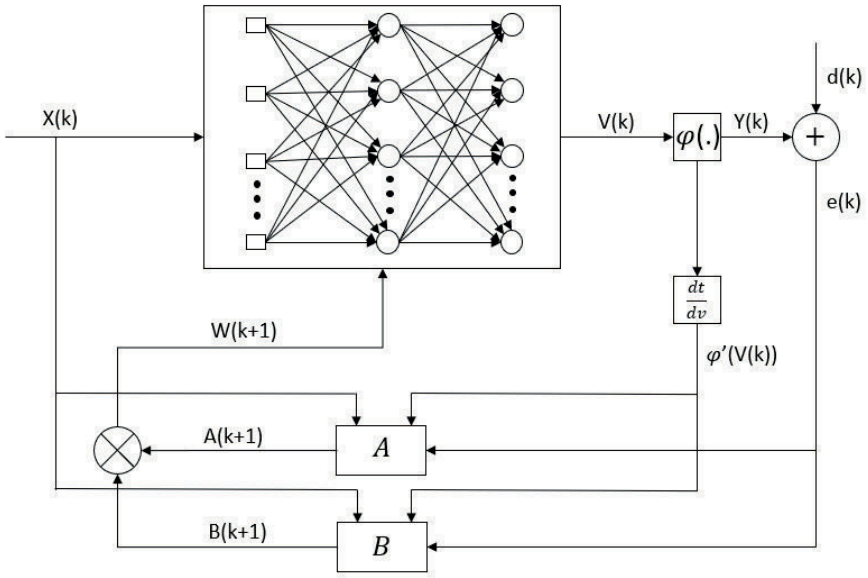


Figura 5.1 – Diagrama em blocos do algoritmo de aprendizagem proposto.

utilizadas. Para exemplificar, considera-se uma rede neural com camada única e função de ativação linear. A comparação da quantidade de operações necessárias para a implementação do algoritmo de Widrow-Hoff e o método proposto são apresentadas na Tabela 5.1. Verifica-se, que para qualquer valor de M_A e M_B, a quantidade de operações necessárias para a implementação do método proposto é menor que a do algoritmo de Widrow-Hoff.

Tabela 5.1 – Complexidade computacional dos algoritmos adaptativos.

Operações	Algoritmo de Widrow-Hoff	Método proposto
Adições	$3MN$	$3(M_A + M_B)(N_A + N_B)$
Multiplicações	$3MN + 1$	$3(M_A + M_B)(N_A + N_B) + 1$
Divisões	1	1
Comparações	0	0

5.3 Comparação do desempenho dos algoritmos

Nesta seção, propõe-se comparar o desempenho do método proposto em relação ao algoritmo de Widrow-Hoff. No primeiro exemplo, considera-se um problema de identificação de sistemas, sendo a planta em questão esparsa. No segundo exemplo, propõe-se identificar as componentes harmônicas, um caso especial de identificação de sistemas, presentes em sinal de corrente do sistema elétrico de potência.

Algoritmo 5.1 Regra de Widrow-Hoff tensorial

1: **Inicialização e parâmetros**

$$\mathbf{A}(0) \text{ é inicializado com valores aleatórios}$$

$$\mathbf{b}(0) \text{ é inicializado com valores aleatórios}$$

$$\alpha < 1$$

2: **Cálculo da saída da rede neural**

$$\mathbf{v}(k) = \mathbf{A}_L(k)^T \mathbf{X}(k) \mathbf{B}_L(k)$$

$$\mathbf{y}(k) = \phi(\mathbf{v}(k))$$

3: **Cálculo do sinal do erro**

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$

4: **Cálculo do gradiente local**

5: L representa a camada de saída

$$\delta_L(k) = \varphi'(\mathbf{V}_L(k))[d(k) - \mathbf{B}_L^T(k)\mathbf{X}(k)\mathbf{A}_L(k)]$$

6: L representa uma camada intermediária

$$\delta_L(k) = \varphi'(\mathbf{V}_L(k)) \ [\mathbf{B}_{L+1}(k)\delta_{L+1}(k)\mathbf{A}_{L+1}^T(k)]$$

7: **Atualização dos tensores**

$$\mathbf{A}_L(k+1) = \mathbf{A}_L(k) + \frac{\alpha \mathbf{Q}_L(k)\delta_L(k)}{\|\mathbf{P}_L(k)\|_2^2 + \|\mathbf{Q}_L(k)\|_2^2}$$

$$\mathbf{B}_L(k+1) = \mathbf{B}_L(k) + \frac{\alpha \mathbf{P}_L(k)\delta_L(k)}{\|\mathbf{P}_L(k)\|_2^2 + \|\mathbf{Q}_L(k)\|_2^2}$$

5.3.1 Identificação de Sistemas

Para avaliar o desempenho dos algoritmos, considera-se a mesma planta mostrada anteriormente na Seção 2.8. Para facilitar o entendimento do leitor, a problemática será retomada nesta seção.

Considere uma planta esparsa com $N = 100$ coeficientes. Os coeficientes ativos (não-nulos) desta planta estão localizados nas posições $\{1; 30; 35; 85\}$, com seus respectivos valores iguais $\{0.1, 1.0, -0.5, 0.1\}$. O sinal de entrada é correlacionado, com média zero e variância unitária, obtido através de um processo autorregressivo de ordem 2, AR(2), dado por

$$x(k) = 0,4x(k-1) - 0,4x(k-2) + v(k), \tag{5.11}$$

sendo $v(k)$ um ruído branco com média zero e variância $\sigma_b^2 = 0,77$. A dispersão da matriz de autocorrelação da entrada é $\chi(\mathbf{R}) = 10$. Os elementos da matriz $\mathbf{R}$ são calculados por

$$r(m) = \frac{\sigma_x^2 p_1 (p_2^2 - 1) p_1^m}{(p_2 - p_1)(p_2 p_1 + 1)} - \frac{\sigma_x^2 p_2 (p_1^2 - 1) p_2^m}{(p_2 - p_1)(p_1 p_2 + 1)}, \tag{5.12}$$

sendo m o *lag* entre as amostras da entrada, σ_x^2, de $x(k)$ e

$$p_{1,2} = \frac{1}{2}\left(-a_1 \pm \sqrt{a_1^2 - 4a_2}\right). \tag{5.13}$$

O ruído de medição, $z(k)$ é branco e gaussiano com variância $\sigma_z^2 = 10^{-3}$.

A estrutura da rede neural escolhida utiliza 100 amostras de dados a cada ciclo na camada de entrada, 1 camada oculta, na qual possui 20 neurônios artificias, e 1 camada de saída, que gera apenas uma saída por ciclo. Com isso, determina-se a dimensões $M_A = 10$ e $M_B = 10$ para os dados de entrada e $N_A = 4$ e $N_B = 5$ para a camada intermediária. A base de dados gerada contém 6000 vetores, sendo que 4000 (dados de simulação) utilizados para o treinamento da rede neural e 2000 para o teste. Na camada oculta é utilizada a função de ativação tangente hiperbólica e na camada de saída uma função linear. A taxa de aprendizagem utilizada no exemplo é $\alpha = 0.05$. A critério de parada consiste na convergência do erro quadrado médio inferior a 10^{-4} ou 200 épocas de treinamento, o que ocorrer primeiro.

A Figura 5.2 mostra o desempenho do algoritmo de Widrow-Hoff e do método tensorial na problemática apresentada anteriormente. Verifica-se que a metodologia proposta possui maior velocidade de aprendizagem e atinge um menor erro durante a etapa de treinamento. Os coeficientes obtidos, utilizando a metologia tensorial, nas etapas de treinamento e teste são apresentados, respectivamente, nas Figuras 5.3 e 5.4. Observa-se que a rede neural proposta possui uma boa capacidade de generalização, fornecendo um bom desempenho quanto a estimação dos coeficientes da planta utilizada. Isto pode ser verificado mais detalhadamente na Figura 5.5, onde são comparadas as 100 primeiras amostras da saída da planta com as estimadas pela rede neural.

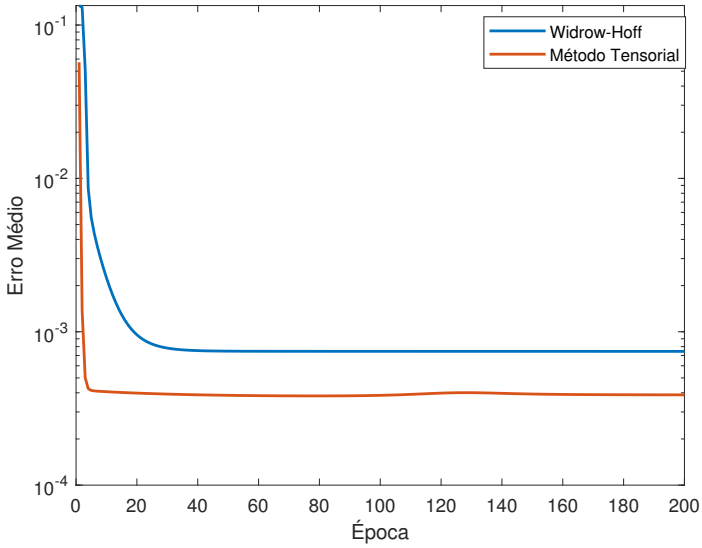


Figura 5.2 – Desempenho do algoritmo de Widrow-Hoff e do método tensorial para identificação da planta proposta.

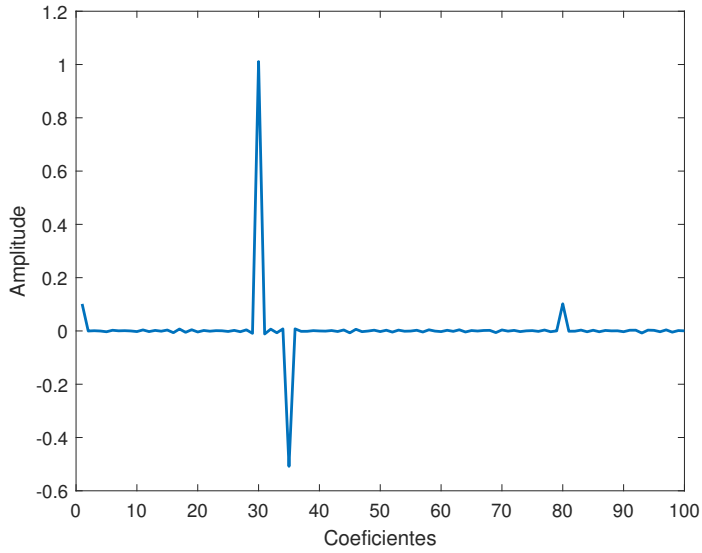


Figura 5.3 – Coeficientes da planta obtidos no treinamento da rede neural.

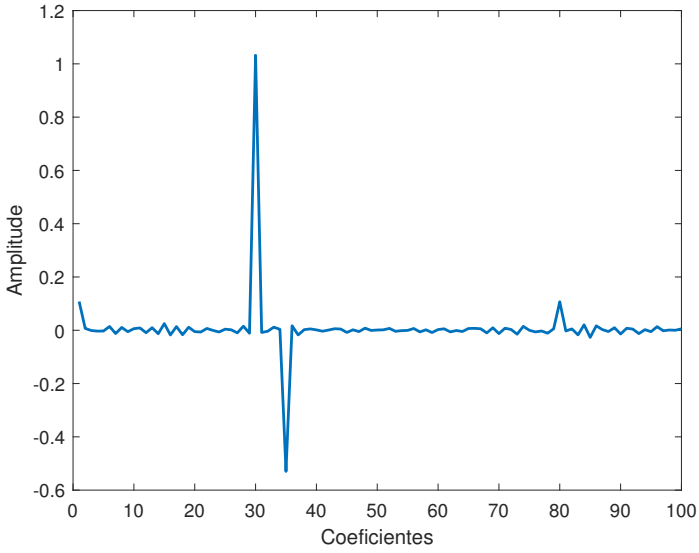


Figura 5.4 – Coeficientes da planta obtidos no teste da rede neural.

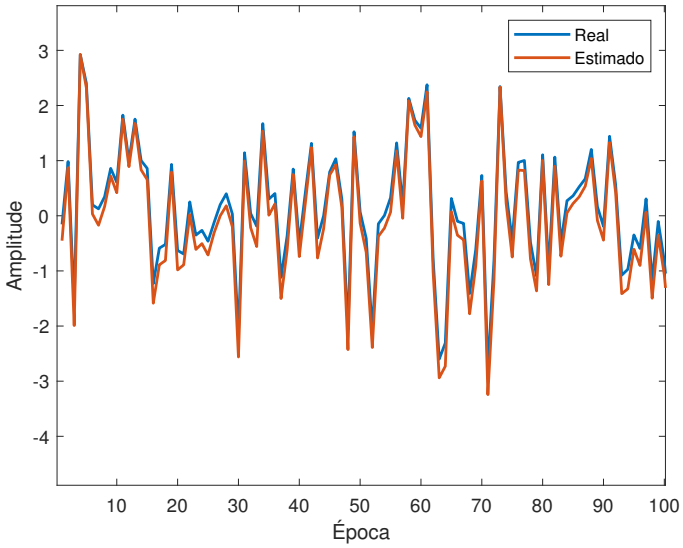


Figura 5.5 – Saídas da planta obtidas no teste da rede neural.

5.4 Identificação de componentes harmônicas

A estrutura geral para identificação de harmônicas a partir de redes neurais é apresentada na Figura 5.6. Nas simulações, os seguintes parâmetros são considerados:

1 A camada de entrada consiste em 100 amostras da amplitude do sinal corrente, obtida a partir de meio ciclo da forma de onda distorcida (frequência de amostragem 12 kHz).

2 A camada oculta é formada por N neurônios cujos valores são alterados durante as simulações para verificar sua influência no desempenho dos algoritmos.

3 A arquitetura utilizada é formada apenas por uma camada oculta.

4 A rede neural, utilizada neste exemplo, visa estimar a terceira e quinta harmônica presente em um sinal de corrente. Assim, a camada de saída é formada por dois neurônios que fornecem a amplitude das componentes desejadas.

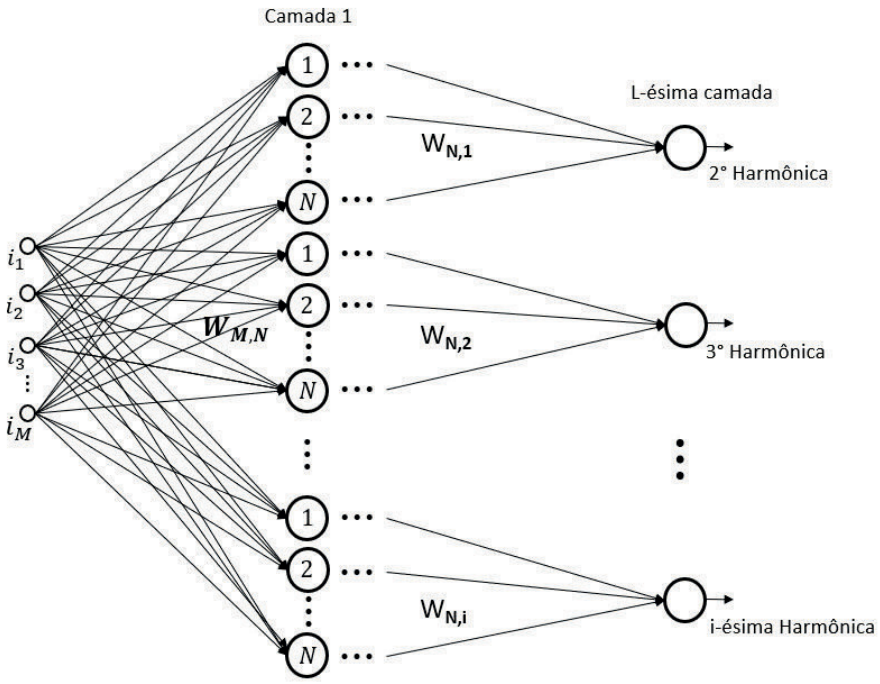


Figura 5.6 – Estrutura geral para identificação de harmônicas.

Nas simulações, é considerado um circuito monofásico cuja corrente pode ser expressa como

$$I_H = I_1 cos(\omega t) - I_3 cos(3\omega t) + I_5 cos(5\omega t). \tag{5.14}$$

Como pode ser observado na Equação (5.14), a flutuação da fase do harmônico não é considerada, sendo definido o valor 0° para o primeiro, terceiro e quinto harmônico.

O banco de dados usado consiste em 141 vetores, 100 para treinamento supervisionado offline usando o algoritmo *backpropagation* e 41 usados para teste de rede neural. Os dados de treinamento são gerados aleatoriamente, cujos valores são delimitados no intervalo de 0 a 100 A. Os dados usados para o teste apresentados no Apêndice do trabalho.

Para a rede neural tensorial, o vetor de entrada cujo comprimento é $M = 100$ é reagrupado em um vetor bidimensional onde $M_B = 10$ e $M_A = 10$. Para a camada oculta, diferentes valores de N_A e N_B são usados para verificar a influência dos comprimentos dos tensores no desempenho do algoritmo.

A métrica usada para avaliar o desempenho dos algoritmos na etapa de teste é o erro de estimativa definido como

$$e_i = \frac{1}{N_r}\sum_{r=1}^{N_r}\frac{1}{N_s}\sum_{s=1}^{N_s}\sqrt{(\hat{A}_{s,i,r} - A_{s,i,r})^2}, \tag{5.15}$$

onde i é o índice da componente harmônica, N_r representa o número total de épocas, r é o índice da época, N_s representa o número total de amostras usadas no teste, s é o índice de amostra, $\hat{A}_{s,i,r}$ é a amplitude estimada e $A_{s,i,r}$ é a amplitude real.

Nas simulações, o critério de parada consistiu na convergência do erro quadrático médio abaixo de 10^{-8} ou na execução de 1000 períodos de treinamento. A função de ativação de cada neurônio na camada oculta é a função sigmoide, enquanto a função de ativação do neurônio de saída é uma função linear. A taxa de aprendizagem é $\alpha = 0,05$.

5.4.1 Cenário I

No primeiro exemplo, o comprimento dos tensores é definido como $N_A = 2$ e $N_B = 5$, onde $N = 10$ é perfeitamente separável. A Figura 5.7 mostra a média do erro quadrático médio obtido durante o estágio de treinamento para estimar a terceira e quinta harmônicas. A partir dos resultados obtidos, pode-se afirmar que a rede neural tensorial proposta apresenta maior velocidade de aprendizado quando comparada ao método clássico. Em 1000 épocas de treinamento, verifica-se que as redes neurais clássica e tensorial convergem para aproximadamente o mesmo erro. Para o teste da rede neural tensorial, considera-se 41 diferentes valores para as componentes harmônicas, sendo os resultados mostrados nas Figuras 5.8 e 5.9. Observa-se que os valores estimados pela rede neural tensorial estão muito próximos dos valores reais, o que confirma a capacidade de generalização desta estrutura. O erro de estimativa obtido pelo método clássico e pelo método do tensor para uma estrutura com N = 10 é mostrado na Tabela 5.2.

Tabela 5.2 – Erro de estimação durante o treinamento para $N = 10$

Harmônicas	Erro de estimação	
	Widrow-Hoff	Método tensorial
3	0.0450	0.0358
5	0.0782	0.0392

5.4.2 Cenário II

No segundo exemplo, o comprimento dos tensores são definidos em $N_A = 5$ e $N_B = 10$, onde $N = 50$ é perfeitamente separável. Em relação ao primeiro exemplo, verifica-se, na Figura 5.10, que a diferença de desempenho entre o método clássico e o método tensorial é maior comparada ao cenário anterior, sendo que, neste exemplo,

a metodologia proposta apresenta maior velocidade de aprendizagem e atinge um erro muito menor no treinamento. Isto ocorre devido ao diferença do número de nós na camada escondida, sendo que este é um fator que potencializa a diferença entre os dois algoritmos. O erro de estimação no cenário II é apresentado na Tabela 5.3

Tabela 5.3 – Erro de estimação durante o treinamento para $N = 50$

Harmônicas	Erro de estimação	
	Widrow-Hoff	Método tensorial
3	0.0524	0.0565
5	0.0819	0.0520

5.4.3 Cenário III

No terceiro cenário, para verificar a influência das dimensões do tensor no desempenho da rede neural tensorial, o número de nós na camada oculta é alterado para $N = 100$, onde diferentes valores de N_A e N_B são utilizados. Observa-se, na Figura 5.11, que a velocidade de convergência da rede neural tensorial aumenta à medida que o comprimento dos vetores N_A e N_B diminui. Assim, além de determinar a quantidade de memória a ser utilizada, as dimensões dos tensores também influenciam no desempenho do algoritmo.

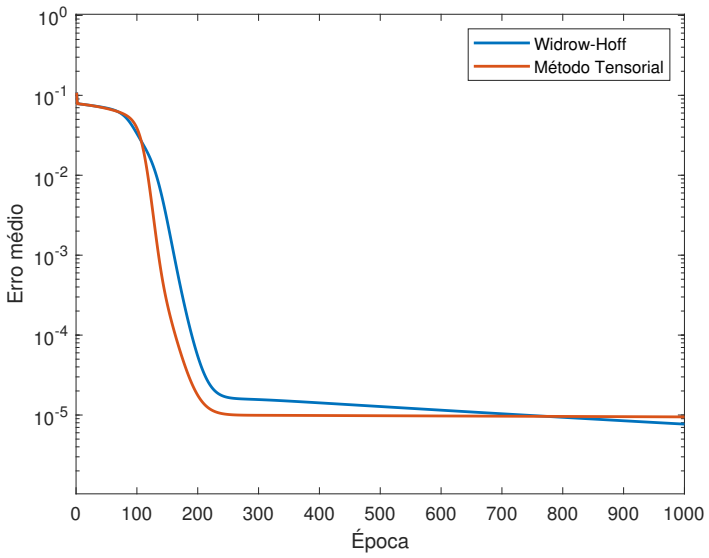


Figura 5.7 – Treinamento da rede neural para $N = 10$.

Tabela 5.4 – Estimação da corrente elétrica (em Ampére) na etapa de teste para $N = 10$

Conteúdo Harmônico			Widrow-Hoff		Método Tensorial	
1°	3°	**5°**	3°	**5°**	**3°**	**5°**
20	7.615	8.125	7.540	8.058	7.602	8.090
22	8.656	8.495	8.575	9.428	8.645	9.466
24	9.693	10.847	9.602	10.773	9.680	10.816
26	10.724	12.167	10.630	12.094	10.716	12.142
28	11.748	13.4551	11.639	13.389	11.733	13.442
30	12.765	14.696	12.658	14.624	12.760	14.681
32	13.771	15.892	13.658	15.825	13.767	15.886
34	14.767	17.036	14.648	16.961	14.764	17.026
36	15.751	18.122	15.626	18.051	15.749	18.119
38	16.722	19.142	16.591	19.067	16.722	19.139
40	17.679	20.094	17.533	20.010	17.671	20.086
42	18.620	20.973	18.476	20.887	18.622	20.966
44	19.544	21.775	19.394	21.681	19.547	21.764
46	20.451	22.492	20.294	22.390	20.455	22.475
48	21.339	23.126	21.176	23.013	21.344	23.101
50	22.207	23.672	22.037	23.548	22.213	23.638
52	23.055	24.126	22.878	23.990	23.062	24.082
54	23.880	24.487	23.696	24.337	23.888	24.430
56	24.682	24.754	24.490	24.590	24.690	24.683
58	25.460	24.924	25.261	24.744	25.469	24.838
60	26.213	24.997	26.006	24.802	26.222	24.894
62	26.941	24.973	26.727	24.762	26.950	24.853
64	27.641	24.851	27.419	24.625	27.650	24.712
66	28.314	24.633	28.084	24.393	28.323	24.475
68	28.959	24.319	28.720	24.066	28.967	24.142
70	29.575	23.910	29.328	23.646	29.582	23.715
72	30.160	23.410	29.905	23.137	30.165	23.197
74	30.715	22.820	30.451	22.541	30.718	22.590
76	31.239	22.143	30.967	21.862	31.240	21.899
78	31.730	21.383	31.449	21.104	31.729	21.127
80	32.189	20.543	31.899	20.271	32.184	20.279
82	32.615	19.627	32.316	19.368	32.607	19.358
84	33.007	18.640	32.700	19.368	32.994	18.370
86	33.366	17.586	33.050	17.370	33.349	17.320
88	33.689	16.471	33.043	16.287	33.341	16.21428
90	33.978	15.301	33.645	15.157	33.950	15.059
92	34.231	14.080	33.889	13.983	34.196	13.860
94	34.449	12.815	34.099	12.773	34.407	12.622
96	34.631	11.511	34.273	11.532	34.581	11.352
98	34.776	10.176	34.411	10.268	34.718	10.05731
100	34.886	8.816	34.514	8.987	34.820	8.744

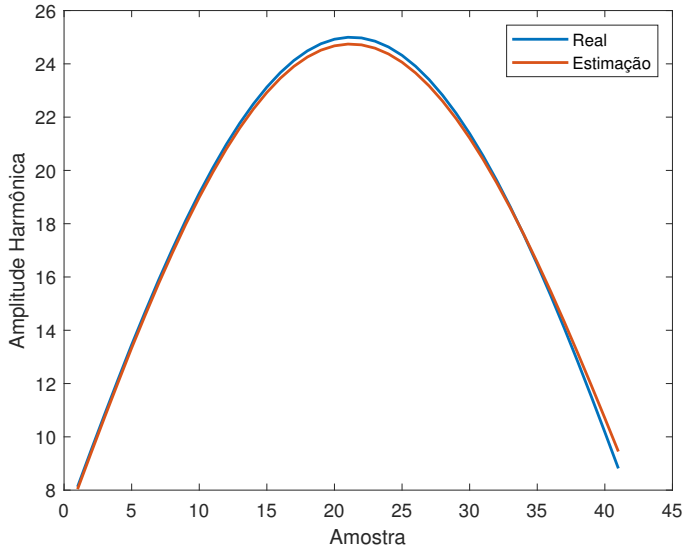


Figura 5.8 – Estimação da terceira harmônica para $N = 10$.

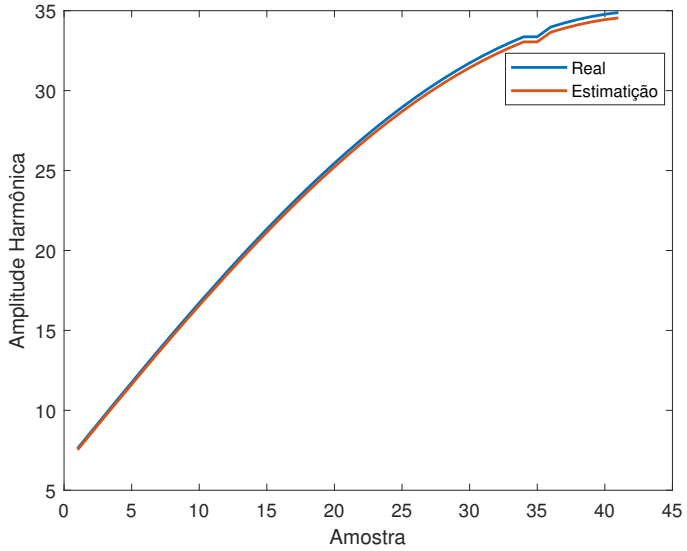


Figura 5.9 – Estimação da quinta harmônica para $N = 10$.

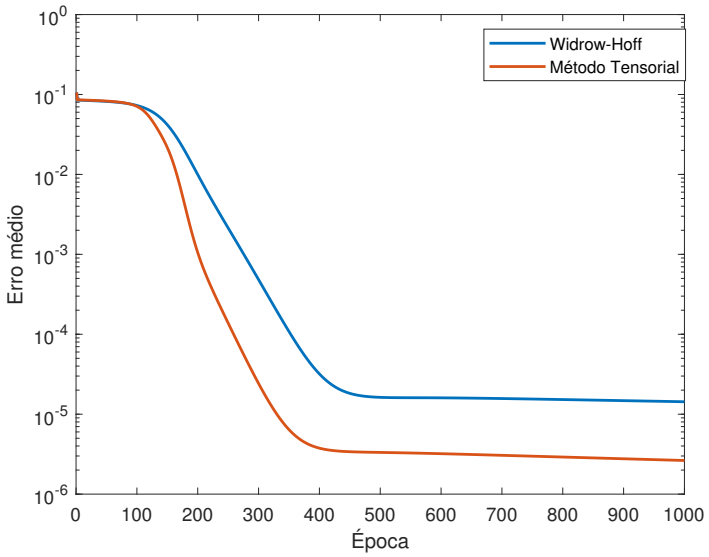


Figura 5.10 – Treinamento da rede neural para $N = 50$.

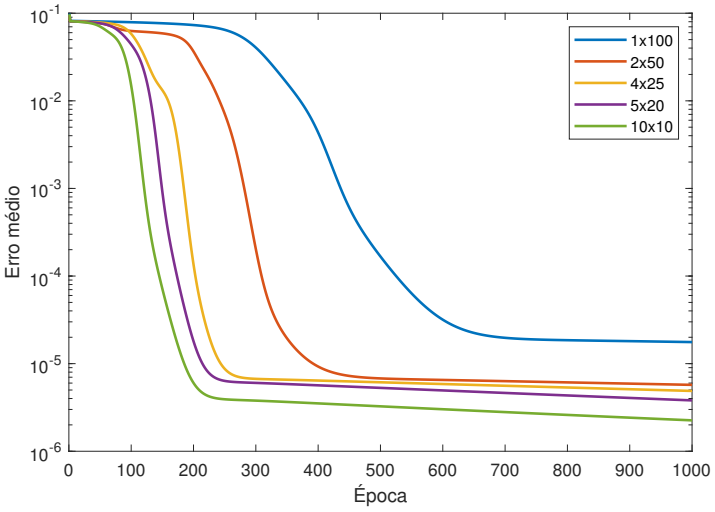


Figura 5.11 – Desempenho do algoritmo proposto para diferentes valores de $N_A \times N_B$.

6 Concusões

6.1 Considerações finais

Neste trabalho, foi proposta uma nova regra de atualização do pesos sinápticos para uma rede neural artificial. Este algoritmo de gradiente estocástico é derivado com o auxílio da propriedade de separabilidade de operadores lineares. A identificação de sistema e a estimação de harmônicas em sistemas elétricos de potência foram os problemas considerados para testar o desempenho do algoritmo.

Nesse contexto, foi realizado um estudo sobre filtragem adaptativa e suas aplicações, dando ênfase aos algoritmo adaptativos LMS, NLMS e GNGD. Isto se faz necessário devido a regra de aprendizagem mais utilizada para atualização dos pesos sinápticos de uma rede neural ser baseada no algoritmo LMS. Além disso, foram apresentados os problemas de desempenho que estes algoritmos possuem em aplicações com elevados números de parâmetros e que envolvem plantas esparsas.

Posteriormente, foram apresentados os algoritmos adaptativos tensoriais e suas derivações. Também são apresentados os teoremas que servem como base teórica para o desenvolvimento dos algoritmos tensoriais a partir dos algoritmos tradicionais. A metodologia tensorial foi aplicadas em um problema de estimação de harmônicas em sistemas elétricos de potência, onde foi possível verificar o seu melhor desempenho em relação aos algoritmos tradicionais.

Uma breve revisão sobre redes neurais foi feita com o objetivo de situar o leitor sobre as vantagens e desvantagens desta estrutura. Primeiro foram mostradas as diferentes arquiteturas das RNAs e suas aplicações em situações práticas. Em seguida foram desenvolvidos os algoritmos de aprendizagem, utilizando conceitos apresentados anteriormente. Também são apresentados os problemas que podem ocorre durante o treinamento de uma rede neural e um método utilizado para resolvê-los.

Em seguida foi desenvolvido o algoritmo de aprendizagem tensorial com base nas teorias apresentadas durante o trabalho. Verificou-se, em problemas práticos de identificação de sistemas e estimação de harmônicas, que a metodologia proposta apresenta maior velocidade de aprendizagem e menor custo computacional, além de atingir um menor erro durante o treinamento em algumas situações. Observou-se também, a influência das dimensões dos tensores no desempenho da metodologia proposta, sendo que quando menor a soma de suas dimensões, maior é a velocidade de convergência do algoritmo.

Portanto, os resultados corroboram com as teorias apresentadas, onde o trabalho apresentou todas as etapas propostas, desde os algoritmos adaptativos mais básicos até o algoritmo de aprendizagem proposto, avaliando o seu desempenho em problemas de

extrema relevância no nosso cotidiano.

6.2 Trabalhos futuros

Visando melhorar o processo de estimação das componentes harmônicas em sistemas elétricos de potência, as seguintes abordagens são sugeridas:

- Utilizar redes neurais artificiais tensoriais profundas com o objetivo de aumentar a capacidade de generalização da estrutura e atingir um erro menor durante a etapa de estimação.
- Aplicar a propriedade de separabilidade dos pesos sinápticos utilizando tensores com dimensão maior que 2, no qual possibilita o uso de uma quantidade menor de lotes de memoria e aumenta a velocidade de convergência.
- Desenvolver métodos de otimização paramétrica com o objetivo de determinar a taxa de aprendizagem ótima, além de definir as dimensões dos tensores que alcançam os melhores resultados.

Referências

ABDOLLAHI, A. et al. Enhanced subspace-least mean square for fast and accurate power system measurement. *IEEE Transactions on Power Delivery*, IEEE, v. 28, n. 1, p. 383–393, 2012. Citado na página 15.

AGGARWAL, C. Neural networks and deep learning. *Springer*, Springer, v. 10, p. 978–3, 2018. Citado 2 vezes nas páginas 58 e 88.

BAJWA, W. U. et al. Compressed channel sensing: A new approach to estimating sparse multipath channels. *Proceedings of the IEEE*, Ieee, v. 98, n. 6, p. 1058–1076, 2010. Citado na página 15.

BARROS, J.; PÉREZ, E. Automatic detection and analysis of voltage events in power systems. *IEEE Transactions on Instrumentation and Measurement*, IEEE, v. 55, n. 5, p. 1487–1493, 2006. Citado na página 17.

BARTO, A. G.; SUTTON, R. S.; ANDERSON, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, IEEE, n. 5, p. 834–846, 1983. Citado na página 59.

BAZARAA, M.; SHERALI, H.; SHETTY, C. Nonlinear programming: theory and algorithms. 2006. *Hoboken: A John Wiley & Sons*, 2004. Citado na página 36.

BRANCO, C. A. S. C. Algoritmos adaptativos lms normalizados proporcionais: proposta de novos algoritmos para identificação de plantas esparsas. Universidade Federal do Maranhão, 2016. Citado na página 23.

CHATTOPADHYAY, S.; MITRA, M.; SENGUPTA, S. Electric power quality. In: *Electric Power Quality*. [S.l.]: Springer, 2011. p. 5–12. Citado na página 17.

CHEN, C.; CHANG, G. An efficient time-domain approach based on prony's method for time-varying power system harmonics estimation. In: IEEE. *2009 IEEE Power & Energy Society General Meeting*. [S.l.], 2009. p. 1–6. Citado na página 20.

COMON, P. Tensors: a brief introduction. *IEEE Signal Processing Magazine*, IEEE, v. 31, n. 3, p. 44–53, 2014. Citado na página 19.

DECKMANN, S.; POMILIO, J. Distorção harmônica: causas, efeitos, soluções e normas. *Avaliação da Qualidade da Energia Elétrica*, v. 5, p. 01–49. Citado na página 19.

DINIZ, P. *Adaptive Filtering Algorithms and Practical Implementation*. [S.l.]: New York: Springer, 2008. Citado 2 vezes nas páginas 14 e 27.

DOUGLAS, S. C.; MENG, T.-Y. Stochastic gradient adaptation under general error criteria. *IEEE transactions on signal processing*, IEEE, v. 42, n. 6, p. 1335–1351, 1994. Citado na página 37.

FARHANG-BOROUJENY, B. *Adaptive filters: theory and applications*. [S.l.]: John Wiley & Sons, 2013. Citado 8 vezes nas páginas 14, 23, 25, 26, 28, 29, 33 e 35.

FERNEDA, E. Redes neurais e sua aplicação em sistemas de recuperação de informação. *Ciência da Informação*, SciELO Brasil, v. 35, n. 1, p. 25–30, 2006. Citado na página 58.

FILHO, R. M. S. et al. Comparison of three single-phase pll algorithms for ups applications. *IEEE Transactions on Industrial Electronics*, IEEE, v. 55, n. 8, p. 2923–2932, 2008. Citado na página 20.

GABARDA, S.; CRISTÓBAL, G. Detection of events in seismic time series by time–frequency methods. *IET Signal Processing*, IET, v. 4, n. 4, p. 413–420, 2010. Citado na página 15.

GALLANT, S. I. *Neural network learning and expert systems*. [S.l.]: MIT press, 1993. Citado na página 66.

GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1. Citado 2 vezes nas páginas 64 e 92.

HAYKIN, S. Adaptive filter theory. *Information and System Science. Prentice Hall*, 2002. Citado 5 vezes nas páginas 14, 23, 24, 26 e 35.

HAYKIN, S. *Neural networks and learning machines*. [S.l.]: Prentice Hall, 2009. Citado 9 vezes nas páginas 57, 58, 59, 63, 66, 67, 89, 91 e 92.

HAYKIN, S. Least-mean-square algorithm. *Adaptive filter theory. 5th ed. New Jersey: Pearson Education*, p. 365–439, 2014. Citado na página 34.

KIM, P. Matlab deep learning. *With machine learning, neural networks and artificial intelligence*, Springer, v. 130, p. 21, 2017. Citado 4 vezes nas páginas 57, 59, 66 e 93.

KRÖSE, B. et al. An introduction to neural networks. Citeseer, 1993. Citado na página 89.

LEE, K.-A.; GAN, W.-S.; KUO, S. M. *Subband adaptive filtering: theory and implementation*. [S.l.]: John Wiley & Sons, 2009. Citado na página 14.

LIN, H. C. Intelligent neural network-based fast power system harmonic detection. *IEEE Transactions on Industrial Electronics*, IEEE, v. 54, n. 1, p. 43–52, 2007. Citado na página 21.

LIVINGSTONE, D. J. *Artificial neural networks: methods and applications*. [S.l.]: Springer, 2008. Citado na página 57.

LOAN, C. F. V.; PITSIANIS, N. Approximation with kronecker products. In: *Linear algebra for large scale and real-time applications*. [S.l.]: Springer, 1993. p. 293–314. Citado na página 43.

LOBOS, T.; KOZINA, T.; KOGLIN, H.-J. Power system harmonics estimation using linear least squares method and svd. *IEE Proceedings-Generation, Transmission and Distribution*, IET, v. 148, n. 6, p. 567–572, 2001. Citado na página 20.

LOGANATHAN, P.; KHONG, A. W.; NAYLOR, P. A. A class of sparseness-controlled algorithms for echo cancellation. *IEEE transactions on audio, speech, and language processing*, IEEE, v. 17, n. 8, p. 1591–1601, 2009. Citado na página 15.

MANDIC, D. et al. Collaborative adaptive learning using hybrid filters. In: IEEE. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. [S.l.], 2007. v. 3, p. III–921. Citado na página 39.

MANDIC, D. P. A generalized normalized gradient descent algorithm. *IEEE signal processing letters*, IEEE, v. 11, n. 2, p. 115–118, 2004. Citado 4 vezes nas páginas 15, 38, 39 e 42.

MANOLAKIS, D. G.; INGLE, V. K.; KOGON, S. M. *Statistical and adaptive signal processing: spectral estimation, signal modeling, adaptive filtering, and array processing*. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 14 e 37.

NOVIKOV, A. et al. Tensorizing neural networks. *arXiv preprint arXiv:1509.06569*, 2015. Citado na página 69.

NUNES, I.; SILVA, H. S. D. *Artificial neural networks: a practical course*. [S.l.]: Springer, 2018. Citado 9 vezes nas páginas 57, 58, 59, 60, 63, 66, 67, 88 e 89.

PALEOLOGU, C.; BENESTY, J.; CIOCHINA, S. Sparse adaptive filters for echo cancellation. *Synthesis Lectures on Speech and Audio Processing*, Morgan & Claypool Publishers, v. 6, n. 1, p. 1–124, 2010. Citado na página 15.

POULARIKAS, A. D. *Adaptive filtering: Fundamentals of least mean squares with MATLAB®*. [S.l.]: CRC Press, 2017. Citado na página 33.

RAHMAN, M.; YU, K.-B. Total least squares approach for frequency estimation using linear prediction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, IEEE, v. 35, n. 10, p. 1440–1454, 1987. Citado na página 20.

RAY, P. K.; SUBUDHI, B. Ensemble-kalman-filter-based power system harmonic estimation. *IEEE transactions on instrumentation and measurement*, IEEE, v. 61, n. 12, p. 3216–3224, 2012. Citado na página 17.

RUPP, M. The behavior of lms and nlms algorithms in the presence of spherically invariant processes. *IEEE Transactions on Signal Processing*, IEEE, v. 41, n. 3, p. 1149–1160, 1993. Citado na página 46.

RUPP, M.; SCHWARZ, S. A tensor lms algorithm. In: IEEE. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2015. p. 3347–3351. Citado 8 vezes nas páginas 15, 20, 34, 43, 44, 45, 46 e 47.

SAHOO, H. K.; SUBUDHI, U. Adaptive estimation of power system harmonics and decaying dc using volterra rls. In: IEEE. *2015 IEEE Power, Communication and Information Technology Conference (PCITC)*. [S.l.], 2015. p. 43–47. Citado na página 20.

SAYED, A. H. *Fundamentals of adaptive filtering*. [S.l.]: John Wiley & Sons, 2003. Citado 3 vezes nas páginas 33, 37 e 46.

SCHILLING, R. J.; HARRIS, S. L. *Fundamentals of digital signal processing using MATLAB*. [S.l.]: Nelson Education, 2011. Citado 2 vezes nas páginas 14 e 28.

SEIFOSSADAT, S. G. et al. Harmonic estimation in power systems using adaptive perceptrons based on a genetic algorithm. *WSEAS Transactions On Power Systems*, v. 11, n. 2, p. 239–244, 2007. Citado na página 20.

SHULGA, D. et al. Tensor b-spline numerical methods for pdes: a high-performance alternative to fem. *arXiv preprint arXiv:1904.03057*, 2019. Citado na página 20.

SIDIROPOULOS, N. D. et al. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, IEEE, v. 65, n. 13, p. 3551–3582, 2017. Citado na página 19.

SO, H.-C. et al. Linear prediction approach for efficient frequency estimation of multiple real sinusoids: algorithms and analyses. *IEEE Transactions on Signal Processing*, IEEE, v. 53, n. 7, p. 2290–2305, 2005. Citado na página 20.

SOUZA, F. d. C. de et al. A pnlms algorithm with individual activation factors. *IEEE Transactions on Signal Processing*, IEEE, v. 58, n. 4, p. 2036–2047, 2009. Citado na página 15.

TUFTS, D. W.; KUMARESAN, R. Estimation of frequencies of multiple sinusoids: Making linear prediction perform like maximum likelihood. *Proceedings of the IEEE*, IEEE, v. 70, n. 9, p. 975–989, 1982. Citado na página 20.

VEGA, L. R.; REY, H. *A rapid introduction to adaptive filtering*. [S.l.]: Springer Science & Business Media, 2012. Citado 2 vezes nas páginas 14 e 30.

WIDROW, B.; HOFF, M. E. *Adaptive switching circuits*. [S.l.], 1960. Citado na página 60.

WIDROW, B.; STEARNS, S. D. Adaptive signal processing prentice-hall. *Englewood Cliffs, NJ*, 1985. Citado 2 vezes nas páginas 29 e 34.

YILMAZ, A. S.; ALKAN, A.; ASYALI, M. H. Applications of parametric spectral estimation methods on detection of power system harmonics. *Electric Power Systems Research*, Elsevier, v. 78, n. 4, p. 683–693, 2008. Citado na página 17.

ZAKNICH, A. *Principles of adaptive filters and self-learning systems*. [S.l.]: Springer Science & Business Media, 2005. Citado 2 vezes nas páginas 23 e 25.

A Apendice

A.1 Arquitetura das redes neurais artificias

A.1.1 Redes Neurais Artificiais *Feedforward*

Em uma rede neural *feedforward*, o fluxo de sinal segue na direção direta, da esquerda para a direita e camada por camada. Além disso, cada neurônio de uma camada é conectado com todos os neurônios da camada seguinte, sendo que não há conexões entre os neurônios de uma mesma camada.

Em geral, uma rede neural artificial *feedforward* pode ser dividida em três partes, denominadas camadas, que são conhecidas como: camada de entrada, que é responsável por receber os dados, sinais ou medidas do ambiente externo, camada escondida ou intermediária, que realiza a extração dos padrões associados ao processo ou sistema que está sendo analisado, e camada de saída, responsável por produzir e apresentar as saídas finais da rede resultantes do processamento realizado pelos neurônios nas camadas anteriores (NUNES; SILVA, 2018, p.22).

As arquiteturas das redes neurais diferem quanto à disposição dos neurônios artificiais, como eles estão conectados e quais camadas estão presentes na estrutura. A arquitetura com camada única (em inglês *Single layer*) possui apenas as camadas de entrada e de saída. Quando é adicionado uma ou mais camadas intermediárias a estrutura neural, obtém-se uma arquitetura multicamada (em inglês *Multi-layer*). Caso seja adicionada apenas uma camada intermediária à arquitetura de camada única, tem-se uma rede neural rasa (em inglês *Shallow neural network*), caso contrário, tem-se uma rede neural profunda (em inglês *Deep neural network*). A seguir, são apresentadas as arquiteturas com camada única e multicamada.

A.1.1.1 Redes neurais artificiais com camada única

Na arquitetura com camada única, as informações fluem em uma única direção, da camada de entrada até a camada de saída. Percebe-se, na Figura 4.3, que o número de saídas da rede neural é igual a quantidade de neurônios. Essas redes são geralmente empregadas em classificação de padrões e problemas de filtragem linear. Entretanto, sua aplicação é limitada a problemas que possuem um conjunto de dados linearmente separáveis (AGGARWAL, 2018, p.8).

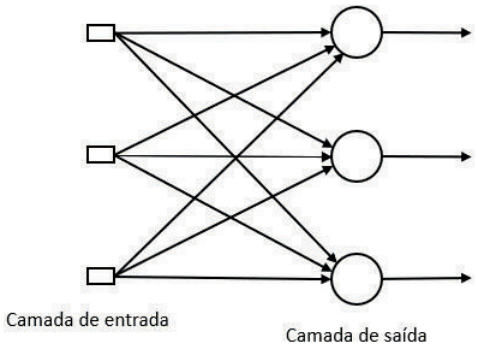


Figura A.1 – Arquitetura de uma rede neural com camada única.

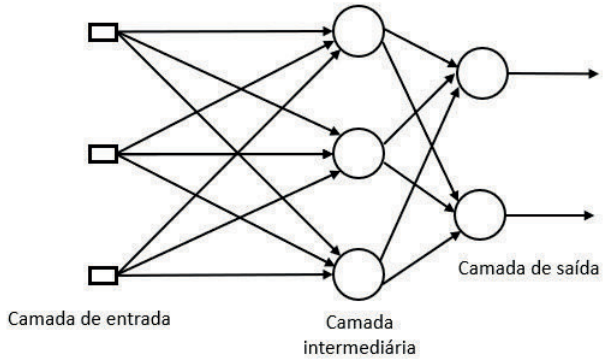
A.1.1.2 Redes neurais artificiais com múltiplas camadas

A arquitetura multicamada é empregada para a resolução de diversos problemas, tais como aproximação funcional, classificação de padrões, identificação de sistemas, controle de processos, otimização, robótica (NUNES; SILVA, 2018, p.23). Ao contrário da arquitetura com camada única, a rede neural multicamada pode ser utilizada para resolver problemas em que os dados não são linearmente separáveis (KRÖSE et al., 1993, p.30).

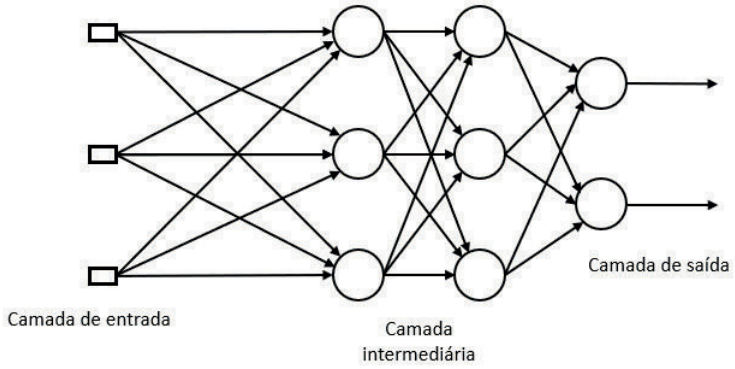
Para ilustrar o funcionamento de uma rede neural multicamada, considere as Figuras 4.3(a) (rede neural artificial rasa) e 4.3(b) (rede neural profunda). Percebe-se que um neurônio em qualquer camada da rede está conectado a todos os neurônios (nós) na camada anterior. A camada intermediária é alimentada pela camada de entrada composta de unidades sensoriais, sendo que os resultados da camada intermediária são aplicados à próxima camada intermediária e assim por diante. A camada intermediária possui duas finalidades: calcular a função sinal que aparece na saída de cada neurônio e a estimativa do vetor gradiente (HAYKIN, 2009, p.125).

A.2 Redes neurais artificiais recorrentes

As redes recorrentes possuem realimentação, na qual a saída de um neurônio é aplicada como entrada no próprio neurônio e/ou em outros neurônios das camadas anteriores. O recurso de realimentação qualifica essas redes para o processamento de informações dinâmicas, o que significa que elas podem ser empregadas em sistemas variantes no tempo, como previsão de séries temporais, identificação e otimização de sistemas, controle de processos e assim por diante (NUNES; SILVA, 2018, p.24). A rede neural recorrente é apresentada na Figura 4.4.



(a)



(b)

Figura A.2 – Arquitetura de uma rede neural com múltiplas camadas. a) Exemplo de uma rede neural rasa. b) Exemplo de uma rede neural profunda.

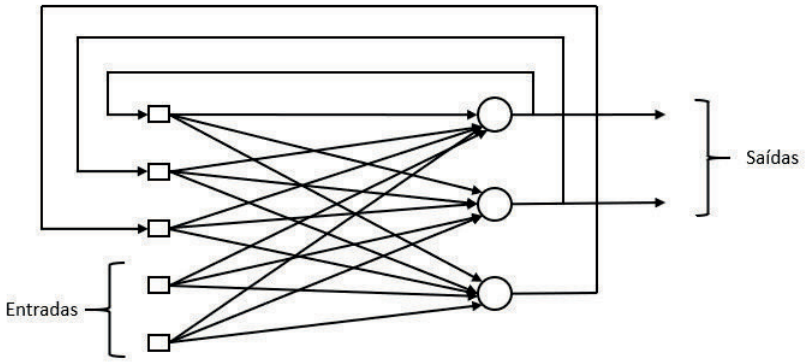


Figura A.3 – Rede neural recorrente.

A.3 Função de Ativação

As funções de ativação são elementos de extrema importância no processo de implementação de uma rede neural artificial. É com base nelas que a rede decide ser o neurônio deve ou não ser ativado, ou seja, se a informação fornecida é relevante. Também são responsáveis pela introdução de característica não lineares as redes neurais. Quando não há funções de ativação, os bias e pesos sinápticos fazem apenas uma transformação linear, o

que resulta é uma capacidade limitada de resolver problemas complexos (HAYKIN, 2009, p.12). A seguir, são apresentadas algumas das funções de ativação mais utilizadas.

A.3.1 Função Degrau

O resultado produzido pela função degrau assumirá valores positivos unitários quando o potencial de ativação do neurônio é maior ou igual a zero, caso contrário, resultado será nulo. Este tipo de função é utilizada em problemas com classificadores binários, cujo a função é apenas decidir se um neurônio deve ser ativado ou não.

$$g(\mathbf{v}) = \begin{cases} 1, & \text{se } \mathbf{v} \geq 0 \\ 0, & \text{se } \mathbf{v} < 0. \end{cases} \tag{A.1}$$

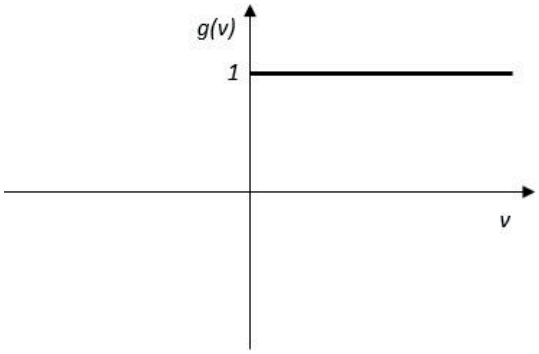


Figura A.4 – Função Degrau.

Como pode-se observar na Equação (4.3), a derivada da função degrau é igual a zero, o que faz com que o seu uso não seja tão útil quando aplicado ao algoritmo de retropropagação, no qual os gradientes das funções de ativação são enviados para cálculos de erro para melhorar e otimizar os resultados.

A.3.2 Função Linear

A função linear corrige o problema da função degrau quanto ao valor zero do gradiente. Neste caso, como pode-se verificar na equação abaixo, o gradiente da função linear é constante e representado por **a**.

$$g(\mathbf{v}) = a\mathbf{v} \tag{A.2}$$

No entanto, a utilização desta função gera outro problema no treinamento da rede neural. A saída dos neurônios artificiais que utilizam essa função de ativação não dependerá de **v**. Assim, quando aplicado o algoritmo de retropropagação, o gradiente será o mesmo, o que impossibilita a redução do erro ao longo de cada época.

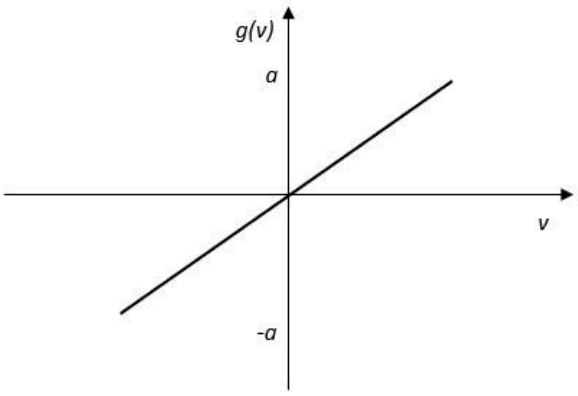


Figura A.5 – Função Linear.

A.3.3 Função Sigmoide

A função sigmoide é de longe a forma mais comum de função de ativação usada na construção de redes neurais (HAYKIN, 2009, p.14). Matematicamente, a função sigmoide é descrita conforme abaixo

$$g(\mathbf{v}) = \frac{1}{1 + e^{-a\mathbf{v}}}, \tag{A.3}$$

onde **a** é uma constante. Verifica-se que os neurônios artificiais que utilizam essa função produzem saídas restritas ao intervalo de 0 a 1, o que é uma característica desejável em problemas de classificação. Entretanto, em aplicações onde a entrada é extremamente positiva ou negativa, ocorre a saturação da função, o que causa problemas na propagação do gradiente e, consequentemente, dificuldade no treinamento da rede neural (GOODFELLOW et al., 2016, p.185).

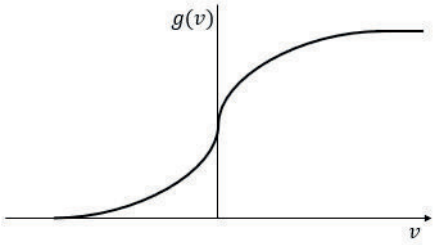


Figura A.6 – Função sigmoide.

A.3.4 Função Tangente Hiperbólica

A função tangente hiperbólica produz saídas dentro do intervalo de -1 a 1. Esta função se aproxima mais da identidade, sendo assim uma alternativa mais atraente do que a sigmoide para servir de ativação às camadas ocultas das RNAs (GOODFELLOW et al., 2016, p.194). Além disso, apresenta um intervalo de saturação maior, o que dificulta

o aparecimento de problemas no treinamento da rede neural.

$$g(\mathbf{v}) = \frac{1 - e^{-a\mathbf{v}}}{1 + e^{-a\mathbf{v}}}. \tag{A.4}$$

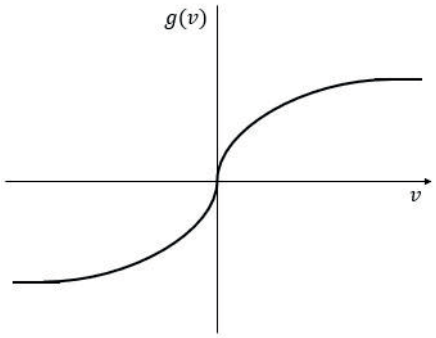


Figura A.7 – Função tangente hiperbólica.

A.3.5 Função ReLU

A função ReLU (Unidade Linear Retificada) produz saídas dentro do intervalo de 0 a ∞. Esta função produz 0 para todos os valores negativos e o próprio valor para números positivos. Sua principal vantagem em relação as demais funções citada é que não apresenta o problema de fuga do gradiente (em inglês *vanishing gradient*), que ocorre quando a rede neural não consegue propagar o erro de saída nos nós mais distantes da camada de saída, fazendo que as camadas ocultas próximas a camada de entrada não sejam devidamente treinadas (KIM, 2017, p.106)

$$ReLU(\mathbf{v}) = max[0, \mathbf{v}]. \tag{A.5}$$

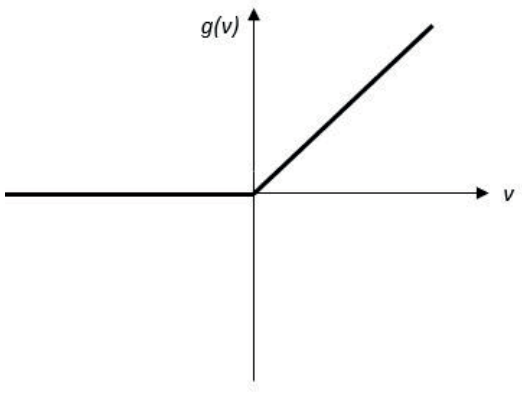


Figura A.8 – Função ReLU.