UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA E ESTATÍSTICA

PAULO EMILIO COSTA SANTOS

Solução Numérica de Equações Diferenciais Parciais Utilizando Redes Neurais de Multiplas camadas através da Retropropagação Restrita - CPROP

ORIENTADOR: PROF. DR. VALCIR JOÃO DA CUNHA FARIAS

BELÉM-PA
2021

PAULO EMILIO COSTA SANTOS

Solução Numérica de Equações Diferenciais Parciais Utilizando Redes Neurais de Multiplas camadas através da Retropropagação Restrita - CPROP

Dissertação apresentada ao Programa de Pós-Graduação em Matemática e Estatística (PPGME) da Universidade Federal do Pará, como requisito parcial para a obtenção do grau de Mestre em Estatística.

Área de Concentração: **Matemática Aplicada**

Orientador: **Prof. Dr. Valcir João da Cunha Farias**

BELÉM-PA
2021

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S237s Santos, Paulo Emilio Costa.
Solução Numérica de Equações Diferenciais Parciais Utilizando Redes Neurais de Múltiplas camadas através da Retropropagação Restrita - CPROP / Paulo Emilio Costa Santos. — 2021.
73 f. : il. color.

Orientador(a): Prof. Dr. Valcir João da Cunha Farias
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Matemática e Estatística, Belém, 2021.

1. Matemática aplicada. 2. Redes Neurais. 3. Solução numérica de Equações Diferenciais Parciais. 4. Retropropagação Restrita. 5. Constrained Backpropagation. I. Título.

CDD 519

PAULO EMILIO COSTA SANTOS

SOLUÇÃO NUMÉRICA DE EQUAÇÕES DIFERENCIAIS PARCIAIS UTILIZANDO REDES NEURAIS DE MULTIPLAS CAMADAS ATRAVÉS DA RETROPROPAGAÇÃO RESTRITA - CPROP

Dissertação apresentada ao Programa de Pós-Graduação em Matemática e Estatística (PPGME) da Universidade Federal do Pará, como requisito parcial para a obtenção do grau de Mestre em Estatística.

Banca Examinadora

Prof. Dr. Valcir João da Cunha Farias
PPGME/UFPA
Orientador

Prof. Dr. Marcus Pinto da Costa da Rocha
PPGME/UFPA
Examinador

Prof. Dr. Edilberto Oliveira Rozal
CAMPUS UNIVERSITÁRIO DE CASTANHAL/UFPA
Examinador Externo

BELÉM-PA
2021

Aos meus pais e amigos.

AGRADECIMENTOS

À Deus, por ter me permitido chegar até aqui

Aos meus pais que me deram forças para não sair do caminho.

Aos meus amigos que trilharam esse caminho comigo.

Ao meu orientador prof. Dr. Valcir Farias por toda sua dedicação e paciência.

A minha companheira Mikaela, que acompanhou cada momento dessa jornada, pelas madrugadas, nos momentos de desabafos e nos momentos de conquista.

Finalmente, gostaria de agradecer à UFPA pelo ensino gratuito de qualidade, ao LAM, ao PPGME e à CAPES, sem os quais essa dissertação dificilmente poderia ter sido realizada e a todos mais que eu não tenha citado nesta lista de agradecimentos, mas que de uma forma ou de outra contribuíram não apenas para a minha dissertação, mas também para eu ser quem eu sou.

“So let’s dance a little,
laugh a little,
hope a little more.”
(for KING & COUNTRY)

RESUMO

Neste trabalho é apresentado um método para resolver equações diferenciais parciais elípticas e parabólicas, fazendo uso de redes neurais artificiais do tipo feedforward, cujos parâmetros (pesos e bias) são parcialmente ajustados buscando minimizar a função erro apropriada. O método foi implementado no software RStudio. A estrutura da rede se difere apresentando uma partição de dois conjuntos chamados de LTM (memória de longo prazo) e STM (memória de curto prazo). Em sua atualização, somente os pesos de STM e os pesos da camada de saída de STM são otimizados fazendo uso do método Levenberg-Marquadt, que utiliza o cálculo do gradiente, onde os resultados obtidos apresentaram uma aproximação da ordem de 10^{-3}.

Palavras-chave: Equações Diferenciais Parciais, Redes neurais Artificiais, Retropropagação restrita.

ABSTRACT

This work presents a method to solve elliptic and parabolic partial differential equations, using feedforward type artificial neural networks, whose parameters (weights and bias) are partially adjusted in order to minimize the appropriate error function. The method was implemented in RStudio software. The structure of the network differs by presenting a partition of two sets called LTM (long-term memory) and STM (short-term memory). In its update, only the STM weights and the STM output layer weights are optimized using the Levenberg Marquadt method, which uses the gradient calculation, where the obtained results presented an approximation of the order of 10^{-3}.

Keywords: Partial Differential Equations, Artificial neural networks, Constrained backpropagation.

LISTA DE ILUSTRAÇÕES

LISTA DE SÍMBOLOS

CPROP

$\varepsilon_{(k)}$ K-ésimo erro

$\hat{N}$ Solução aproximada de uma EDP

Λ_L Produto da matriz diagonal de W_L

Λ_S Produto da matriz diagonal de W_S

$\mathscr{B}$ Operador diferencial de ordem inferior a $\mathscr{L}_n$

$\mathscr{L}_n$ Operador diferencial

Ω Resultado obtido pela função de ativação com os dados de entrada da fronteira ponderado por p_S

$\omega_{S_j}^{m_j}$ J-ésima coluna de W_S

Ψ Resultado obtido pela função de ativação com os dados de entrada da fronteira ponderado por p_L

$\tilde{h}$ Condição de contorno da equação parabólica

$\vec{x}$ Conjunto de entradas da rede

b_L Pesos sinápticos do bias pertencentes a LTM

b_S Pesos sinápticos do bias pertencentes a STM

C Condição de restrição

$h(\vec{x})$ Função escalar

$h_{(l)}$ Condição de contorno de uma equação elíptica

K Total de neurônios da rede

K_L Neurônios do conjunto LTM

K_S Neurônios do conjunto STM

p_L Conjunto de pesos pertencentes a LTM

p_S Conjunto de pesos pertencentes a STM

q Função diferenciável

T_L Conjunto de dados das condições iniciais e/ou de contorno

T_S Conjunto de entrada-saída

$u(\vec{x}_l)$ Fornece as condições iniciais da equação parabólica

V_L Pesos sinápticos camada oculta pertencentes a LTM

V_S Pesos sinápticos camada oculta pertencentes a STM

W_L Pesos sinápticos da camada de entrada pertencentes a LTM

W_S Pesos sinápticos da camada de entrada pertencentes a STM

Z_L Vetoreque contem as entradas ponderadas pertencentes a LTM

Z_S Vetor que contem as entradas ponderadas pertencentes a STM

Redes Neurais

η Constante de aprendizagem

Σ Combinador linear

θ Limiar de ativação (bias)

d^k k-ésima saída desejada

$E(w)$ Erro quadrático de w

$g(.)$ Função de ativação

u Potencial de ativação

w_i I-ésimo peso sináptico

W_{ji}^L Matriz de pesos do j-ésimo neurônio da camada L que é ligado ao i-ésimo neurônio da camada $L-1$

x_i I-ésimo sinal de entrada

y Sinal de saída

Y_{ji}^L Vetores que contem as saídas ponderadas em relação ao j-ésimo neurônio da camada L

Z_{ji}^L Vetores que contem as entradas ponderadas em relação ao j-ésimo neurônio da camada L

SUMÁRIO

1 INTRODUÇÃO

1.1 ASPECTOS GERAIS

A busca de soluções de Equações Diferenciais Parciais (EDPs) através de Redes Neurais Artificiais (RNA) deu origem a diversos métodos e aprimoramentos ao longo dos anos. Com a evolução dos computadores e o surgimento da neurocomputação, métodos vem sendo desenvolvidos e aprimorados para obter soluções de equações diferenciais.

Dentre os modelos de rede, o Perceptron de camadas multiplas se mostra como um excelente aproximador de funções com uma precisão arbitrária (Batista, 2012), sendo um forte candidato para trabalhar com EDPs.

Um método que merece destaque ao se trabalhar com Equações Diferenciais é o artigo publicado por (Lagaris; Likas; Fotiadis, 1998), onde ele apresentou soluções para Equações Diferenciais Ordinárias e a solução de uma Equação de Poisson, através da aplicação de uma rede *feedforward*, com um elemento de aproximação, onde ocorre um ajuste dos parâmetros (pesos e bias) para minimizar uma função erro apropriada.

Em 2001, Aarts e van der Veer apresentam um outro método de rede neural para resolver uma equação com condições de contorno e/ou condições iniciais. As equações e suas condições são incorporadas na estrutura da rede, gerando bons resultados para problemas de duas dimensões.

Outro avanço foi feito por Parisi, Mariani e Laborde (2003) ao implementarem uma rede artificial com treinamento não supervisionado, resolvendo problemas de engenharia química que envolviam equações diferenciais. O treinamento envolvia o método do algorítmo genético e o método do gradiente descendente, de modo que ao analisarem os resultados, eles concluem que a solução aproximada pela rede é de expressão matemática compacta, podendo utilizar n parâmetros como variáveis e é um ótimo método para problemas de otimização de controle.

Shirvany, Hayati e Moradian (2009) utilizaram uma rede Perceptron de Camadas Múltiplas e Função de Base Radial para trabalhar com equações do tipo Ordinária e Parcial. Com as condições de fronteira de cada equação, é preparada uma função de energia da rede que utiliza um treinamento não supervisionado para ajustar os parâmetros, onde obtiveram bons resultados. O método apresentou um baixo custo computacional, quando comparado a métodos numéricos clássicos.

Em 2009, o algorítmo de evolução gramatical foi utilizado por Ioannis, Gavrilis e Glavas para resolver equações diferenciais utilizando redes neurais. Essa aplicação consiste em uma série de 19 experimentos, mostrando que o método proposto conseguia resolver todos os problemas e concluindo que o mesmo pode ser utilizado para resolver equações diferenciais de qualquer ordem.

Ainda no ano de 2009, Beidokhti e Malek apresentam um sistema geral de equações

diferenciais parciais dependentes do tempo contendo multiplas condições iniciais e de fronteira. Para determinar uma solução, foi utilizado um método híbrido, utilizando redes neurais, técnicas de minimização e métodos de colocação. O método apresentou bons resultados, funcionando bem para pontos interiores do domínio.

Em 2012, Batista abordou o algorítmo desenvolvido por Lagaris, Likas e Fotiadis, utilizando no treinamento da rede os métodos do Gradiente Descendente e Levenberg-Marquadt. Foi realizada uma comparação da solução numérica com a solução analítica, calculando o desvio e para cada algorítmo de treinamento.

Dois anos depois, Mall e Chakraverty realizam uma aplicação baseada na Rede Neural de Chebyshev (RNT), para solucionar equações de Lane-Emden homogêneas e não-homogêneas. Durante essa aplicação, foi utilizada uma única rede neural, com uma expansão do padrão de entrada através dos polinômios de Chebyshev. A rede era do tipo *feedforward* e os ajustes eram feitos por meio do algorítmo *backpropagation*, aplicado em uma aprendizagem supervisionada.

Em 2014, Rudd, Muro e Ferrari apresenta um novo método de ajuste dos pesos para resolver equações diferenciais parciais elípticas e parabólicas com problemas de valores de contorno e/ou iniciais, a retropropagação restrita (CPROP). O método consiste em uma atualização restrita dos pesos, particionando a rede em duas partes distintas. O método apresenta vantagem em relação a outros métodos pois consegue adaptar a rede para minimizar o erro definido pelo operador diferencial, ao mesmo tempo que satisfaz as restrições de igualdade fornecidas pelas condições iniciais/contorno, além de uma melhora significativa no custo computacional.

Após um ano, Rudd e Ferrari apresentam um novo método que utiliza o CPROP, denominado integração restrita (CINT), que utilizava a retropropagação restrita com o método clássico de Galerkin para resolver equações diferenciais parciais. Esse método apresenta a vantagem de poder trabalhar com dominios irregulares e de geometria complexa, apresentando uma vantagem se comparado aos métodos de elementos finitos, além da melhora do custo computacional e da precisão.

No ano de 2016, Mall e Chakraverty apresentam uma proposta de método que teve como base a Rede Neural de Legendre, com considerações semelhantes a proposta de 2014. Foi utilizada uma unica camada neural com expansão dos dados de entrada através dos polinômios de Legendre, Apresentando um número menor de parâmetros com implementação simples.

Campos (2018) realizou um estudo comparativo entre as redes de Legendre e Chebyshev propostas por Mall e Chakraverty com o método da descida mais íngrime para sua otimização. Foi realizada uma comparação da solução numérica com a solução analítica obtida por cada rede, obtendo resultados excelentes com baixo custo computacional.

Este trabalho empregou a mesma metodologia proposta por Rudd, Muro e Ferrari(2014) para resolver equações diferenciais parciais elípticas e parabólicas, com o método de Levenberg-Marquardt para treinar a rede.

1.2 JUSTIFICA DO TRABALHO

O método utilizado por (Rudd; Muro; Ferrari, 2014) apresenta uma forma de resolver Equações Diferencais Parciais, utilizando a adaptação de uma rede neural com um baixo custo computacional. Essa adaptação consiste em uma restrição na atualização da rede, permitindo que a mesma obtenha bons resultados em soluções de equações diferenciais parciais. Tornando interessante realizar uma replicação do método para estudar seu desempenho.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Apresentar a metodologia proposta por Rudd; Muro; Ferrari bem como sua aplicação para obter soluções de equações diferenciais parciais.

1.3.2 Objetivos Específicos

- Apresentar um modelo de rede neural que utiliza otimização restrita (CPROP)
- Realizar simulações para realizar a análise do modelo proposto

1.3.3 Sumário da dissertação

Esta dissertação consiste em 3 capítulos e considerações finais:

- O primeiro capítulo: Introduz de forma breve os conceitos de uma rede neural, necessários para o desenvolvimento do trabalho.
- O segundo capítulo: Mostra a métodologia aplicada no trabalho, apresentando o cálculo do gradiente do erro proposto por Rudd, Muro e Ferrari(2014)
- O terceiro capítulo: Apresenta a solução de algumas equações diferenciais parciais, tanto o cálculo do gradiente do erro de cada uma como a sua representação gráfica.
- As considerações finais: Será apresentada a conclusão a cerca dos experimentos realizados e sugestões para trabalhos futuros.

2 REDES NEURAIS

2.1 INTRODUÇÃO

As Redes Neurais Artificiais são modelos computacionais que simulam características do cérebro humano, como a sua estrutura e funções. Para realizar esse tipo de simulação, pesquisadores desenvolveram uma forma de modelar o neurônio biológico, adaptando sua estrutura, funcionalidade e principalmente a sua forma de operar.

Segundo Silva, Spatti e Flauzino(2010), as redes neurais são modelos inspirados no sistema nervoso, com capacidade de aquisição e manutenção de conhecimento. São caracterizadas por neurônios artificiais que são ligados por um enorme número de interconexões (sinapses artificiais), que são representadas através de matrizes/vetores.

Devido a sua versatilidade, as redes podem ser aplicadas em diversos problemas de engenharia e ciências (Campos, 2018), com uma aplicação maior em situações do tipo: Aproximador universal de funções, reconhecimento/classificação de padrões, sistemas de previsão e soluções de Equações Diferenciais.

2.2 OS NEURÔNIOS BIOLÓGICOS E ARTIFICIAIS

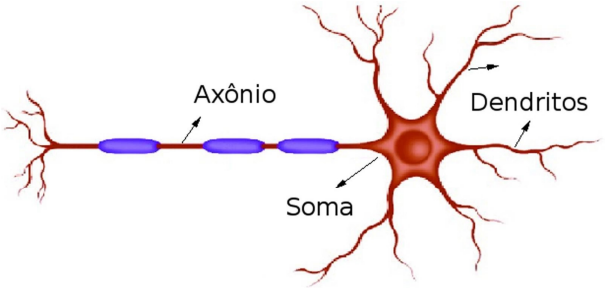
2.2.1 O Neurônio Biológico

O processamento de informações dentro de uma rede neural depende de uma célula elementar, o neurônio. O seu papel se resume basicamente em conduzir impulsos (estímulos elétricos) sob determinadas condições. Segundo Bear, Connors e Paradiso(2017), sua estrutura se divide em várias partes, mas destacando três delas temos:

- **O corpo Celular**: Também conhecido como Soma, é a região central do neurônio, que armazena diversas estruturas (organelas) e serve como centro de processamento de informações, recebidas pelos dendritos.
- **Os Dendritos**: São estruturas semelhantes a ramos de árvore, que tomam essa forma conforme vão se afastando do Soma. Eles funcionam como antenas que recebem milhares de esímulos (sinapses).
- **O Axônio**: Se trata de um único filamento especializado em transferir informações para outros neurônios ou para outras estruturas receptoras.

Através dessa estrutura, são transmitidas as sinapses, que são pequenos impulsos elétricos que carregam informação de um neurônio para outro através de ponderações sinápticas que dependem da química cerebral. A Figura 1 apresenta uma estrutura de um neurônio biológico.

Figura 1 – Estrutura - Neurônio biológico



Fonte: Google imagens

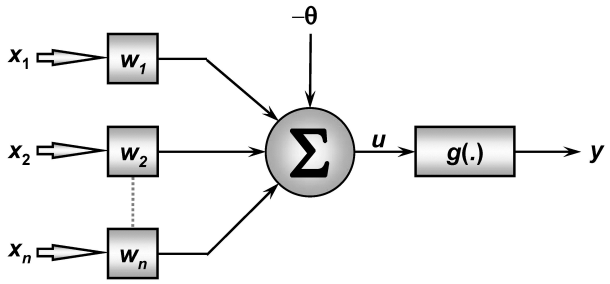
2.2.2 O Neurônio Artificíal

Adaptando a estrutura funcional, o neurônio artificial consiste em uma versão mais simplificada do neurônio biológico. são definidos sete elementos principais em sua estrutura (Silva; Spatti; Flauzino, 2010), são eles:

- **Sinais de entrada**: Possuem uma função semelhantes aos dendritos, os sinais de entrada são representados pelo conjunto $\{x_1, x_2, \cdots, x_n\}$, que carregam informações do meio externo.
- **Pesos sinápticos**: Desepenham o papel da ponderação sináptica no neurônio biológico e no artificial serão valores numéricos $\{w_1, w_2, \cdots, w_n\}$ capazes de ponderar os sinais de entrada.
- **Combinador linear** (Σ): Tem a função de agregar os sinais de entrada ponderados para produzir um valor de potencial de ativação.
- **Limiar de ativação**(bias): É uma variável θ que tem a função de determinar o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção á saída do neurônio.
- **Potencial de ativação**: Se trata do valor u resultante entre a Combinação linear e o Bias.
- **Função de ativação**: A função $g(.)$ tem o papel de limitar a saída do neurônio, dentro de um intervalo razoavel.
- **Sinal de saída**: É o valor final produzido pelo neurônio, com base nas entradas fornecidas.

A Figura 2 ilustra um neurônio artificial e seus respectivos componentes.

Figura 2 – Estrutura - Neurônio artificial



Fonte: Adaptado de Silva, Spatti e Flauzino(2010)

Para sintetizar os resultados do neurônio artificial, McCulloch e Pitts propuseram duas expressões

$$\begin{cases} u = \sum_{i=1}^{n} w_i.x_i - \theta \\ y = g(u) \end{cases}$$

2.3 ARQUITETURAS DAS REDES E TIPOS DE TREINAMENTO

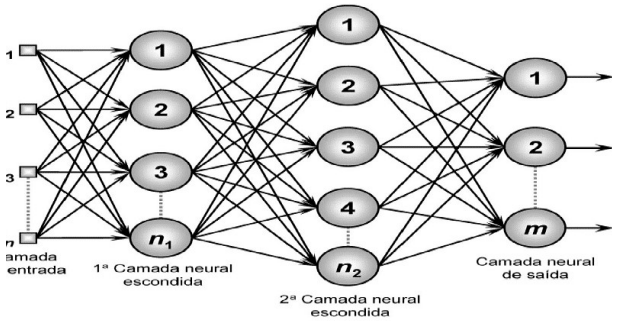
Uma Rede Neural Artificial pode variar em sua forma, uma vez que ela pode se adaptar ao problema que ela vai ser aplicada, mas ela pode ser divida basicamente em três partes. A primeira camada recebe o nome de Camada de Entrada, a segunda parte recebe o nome de Camada Oculta, sendo que esta pode ser constituida por diversas camadas, uma vez que é feita de forma heurística (Batista, 2012) e a terceira camada recebe o nome de Camada de Saída.

Existem dois tipos de arquiteturas que são mais utilizadas, o primeiro deles modelo unidirecional (*feedforward*), que é empregado em casos de aproximação de funções, classificação de padrões, otimização, etc. O segundo modelo é o recorrente, que pode ser empregado em previsões de séries temporais, controle de processos e outros (Campos, 2018).

2.3.1 Feedforward

Esse modelo de arquitetura possui uma entrada, conectada a uma ou mais camadas ocultas e uma camada de saída, de modo que os dados sempre percorrem um único sentido, da entrada para a saída. Nessa estrutura, os neurônios de uma camada se conectam com todos os neurônios da camada posterior, sem interagir com os neurônios da mesma camada. A Figura 3 representa uma rede *Feedforward*, com duas camadas ocultas, com n entradas e m saídas.

Figura 3 – Estrutura - *Feedforward*

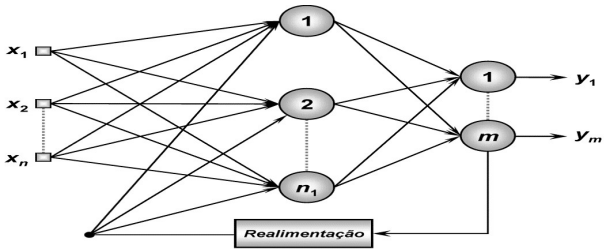


Fonte: Google imagens

2.3.2 Recorrentes

Diferente do Feedforward, uma arquitetura do tipo Recorrente não possui um único sentido dos fluxos de sinais, onde um neurônio pode ser direta ou indiretamente alimentado por sua camada de saída através de um laço de recorrencia. A Figura 4 representa uma rede *Recorrente*, com uma camada oculta, *n* entradas e *m* saídas.

Figura 4 – Estrutura - *Recorrente*



Fonte: Google imagens

2.3.3 Processos de Treinamento e Aprendizagem

Uma rede neural possui uma característica peculiar, a capacidade de adaptação por experiencias (Silva; Spatti; Flauzino, 2010), desse modo é possível atualizar os pesos sinápticos da rede a partir da apresentação de exemplos. Das formas de treinamento, podemos destacar:

- Treinamento supervisionado: Este modelo consiste na utilização de um conjunto de pares de entradas e saídas já conhecidos. O treinamento ocorre quando um conjunto de entradas

não obtem a saída desejada para aquele teste, isso indica que a rede precisa realizar alguns ajustes em seus pesos, até obter um erro mínimo desejado.

- Treinamento não-supervisoonado: Diferente do modelo anterior, esse treinamento não possui conhecimento prévio de um conjunto de saídas. A própria rede deve se auto-organizar utilizando somente os pesos de entradas de modo que seja feita uma classificação através de um critério de semelhança.

- Aprendizagem usando lote de padrões (off-line): Neste tipo de aprendizagem, a atualização dos pesos só ocorre após a avaliação de todos os elementos do conjunto de treinamento, pois é considerado o total dos desvios das amostras.

- Aprendizagem usando lote de padrão-por-padrão (on-line): O modelo de aprendizagem on-line, realiza atualizações após a apresentação de cada amostra de treinamento, de forma que após o ajuste, o par de treinamento ja utilizado é descartado.

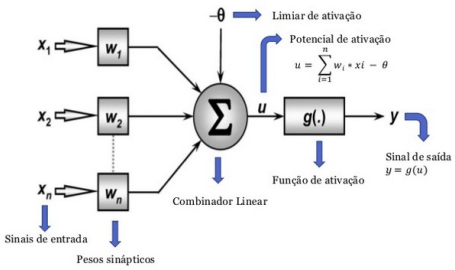
2.4 MODELOS DE REDES NEURAIS ARTIFICIAIS

Quando falamos de Modelos de Redes, podemos encontrar diversos, dos mais simples aos mais complexos, adequados para situações diferentes. Nesta seção serão apresentados três modelos clássicos, são eles: Rede Perceptron, Rede Adaline e Rede Perceptron Multicamadas.

2.4.1 Rede Perceptron

Modelo Proposto por Rosenblatt(1958), o Perceptron é a forma mais simples de uma rede neural, funcionando como uma forma de percepção eletrônica podendo indentificar padrões geométricos. Sua estrutura consiste em camadas de entrada, um único neurônio compondo a camada neural e uma saída, conforme ilustra a Figura 5

Figura 5 – Estrutura - *Perceptron*



Fonte: Google imagens

Em funcionamento, a rede será alimentada pelas entradas, que serão respectivamente ponderadas pelos pesos sinápticos, para posteriormente terem seus valores adicionados do limiar de ativação. Essa soma será repassada para a função de ativação que irá produzir a saída do Perceptron para essas respectivas entradas, como representado nas equações a seguir:

- $u = \sum_{i=1}^{n} w_i.x_i - \theta$;
- $y = g(u)$.

A forma de ajustar os pesos do Perceptron era igualmente simples, através de um treinamento supervisionado, os pesos são ajustados quando a saída obtida pelo perceptron for diferente da desejada, a diferença entre esses resultados implica na atualização dos pesos sinápticos e do limiar de ativação, de modo que:

$$w_i^{atual} = w_i^{anterior} + \eta.\left(d^{(k)} - y\right).x^{(k)}; \tag{2.1}$$

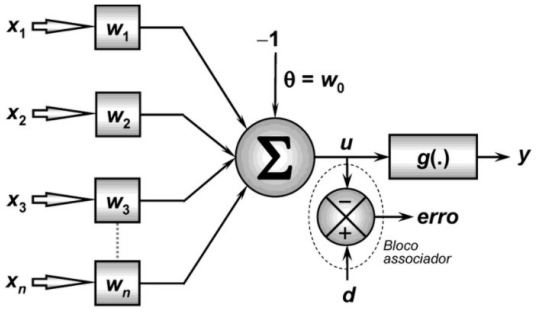
$$\theta_i^{atual} = \theta_i^{anterior} + \eta.\left(d^{(k)} - y\right).x^{(k)}. \tag{2.2}$$

Com $x^{(k)}$ representando a k-ésima entrada, y representando a saída do Perceptron, $d^{(k)}$ representando a saída desejada e η é a constante de aprendizagem da rede, que define a velocidade de convergencia dos pesos para um valor ideal.

2.4.2 Rede Adaline

O modelo Adaline foi proposto em 1960 por Widrow e Hoff, com uma estrutura semelhante ao Perceptron mas trazendo uma nova regra de aprendizagem, a regra Delta, que mais tarde serviria de base para a regra Delta generalizada, utilizada no Perceptron de camadas multiplas. A Figura 6 ilustra sua estrutura.

Figura 6 – Estrutura - *Adaline*



Fonte: Google imagens

O resultado produzido por uma rede Adaline segue o mesmo modelo da perceptron, de modo que:

$$u = \sum_{i=1}^{n} w_i.x_i - \theta; \tag{2.3}$$

$$y = g(u). \tag{2.4}$$

2.4.2.1 O treinamento da Rede Adaline

O grande diferencial se encontra no treinamento dos pesos, onde a regra Delta ou também conhecida como algoritmo LMS (*least mean square)*, busca um valor mínimo entre a saída desejada d e a resposta do combinador linear u, procurando minimizar o erro quadrático entre eles, a fim de ajustar os pesos da rede $w = (\theta, w_1, w_2, \ldots, w_n)$. O sinal do erro no bloco presente na Figura 6 é descrito por

$$erro = d - u. \tag{2.5}$$

A função Erro Quadrático, busca obter um w ótimo, de modo que o erro quadrático $E(w)$ seja o menor possivel, tornando a rede mais tolerável ao erro sem ignorar a presença do mesmo. Tal função é descrita por:

$$E(w) = \frac{1}{2}\sum_{i=1}^{p}\left(d^{(k)} - u)\right)^2. \tag{2.6}$$

Com u definido pela equação (2.3). substituindo a equação (2.3) em (2.6), obtemos.

$$E(w) = \frac{1}{2}\sum_{i=1}^{p}\left(d^{(k)} - (\sum_{i=1}^{n} w_i.x_i - \theta))\right)^2; \tag{2.7}$$

$$E(w) = \frac{1}{2}\sum_{i=1}^{p}\left(d^{(k)} - (w^T.x^{(k)} - \theta))\right)^2. \tag{2.8}$$

Para obter o valor mínimo de (2.8), é aplicado o gradiente em relação aos pesos w.

$$\nabla E(w) = \frac{\partial E(w)}{\partial w}. \tag{2.9}$$

Esse resultado pode ser definido através da expressão a seguir (Batista, 2012).

$$\nabla E(w) = -\sum_{i=1}^{p}\left(d^{(k)} - u)\right).x^{(k)}. \tag{2.10}$$

O vetor dos pesos será atualizado buscando um ponto mínimo, indo em direção oposta ao gradiente naquele ponto, dessa forma, a variação a ser efetivada pelo vetor de pesos é

$$\Delta w = -\eta.\nabla E(w). \tag{2.11}$$

Com η representando a taxa de aprendizagem, a nossa variação assume a forma de (2.12)

$$\Delta w = \eta.\sum_{i=1}^{p}\left(d^{(k)} - u)\right).x^{(k)}. \tag{2.12}$$

Que serve para ajustar os pesos através de

$$w_i^{atual} = w_i^{anterior} + \eta . \sum_{i=1}^{p} \Big(d^{(k)} - u) \Big) .x^{(k)}. \tag{2.13}$$

Podendo ser simplificada para atualizar discretamente os pesos de $w = w_1, w_2, \cdots, w_n$ após cada padrão de treinamento (Silva; Spatti; Flauzino, 2010).

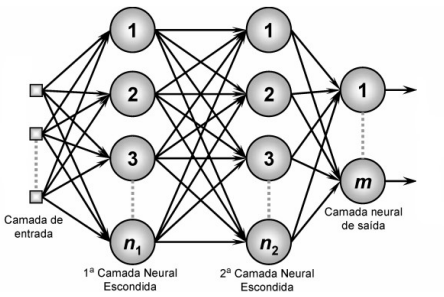
$$w_i^{atual} = w_i^{anterior} + \eta . \Big(d^{(k)} - u) \Big) .x^{(k)}. \tag{2.14}$$

2.4.3 Rede Perceptron de Multiplas Camadas

A estrutura da Rede Perceptron de Multiplas Camadas (PMC), é caracterizada pela presença de pelo menos uma camada oculta de neurônios, entre as camadas de entrada e saída. Possuindo no mínimo duas camadas para distribuir seus neurônios, a camada de saída e a camada oculta.

Esta Arquitetura apresenta uma alimentação do tipo *feedforward*, com treinamento supervisionado, sendo ele o algorítimo *backpropagation*, que utiliza a regra Delta generalizada. A Figura 7 ilusta a estrutura de uma rede PMC contendo duas camadas ocultas.

Figura 7 – Estrutura - *PMC*



Fonte: Adaptado de Batista(2012)

Com essa estrutura, os dados fornecidos pela camada de entrada alimentam a primeira camada oculta, de forma que as saídas de cada neurônio alimenta a camada sucessora, seguindo esse fluxo até a camada de saída.

Além das camadas ocultas, a camada de saída pode apresentar mais de um neurônio, como é o caso da Figura 7, distribuindo o conhecimento referente da entrada-saída, por todos os neurônios da rede, permitindo um mapeamento mais amplo do problema, sendo mais completo que o *perceptron*.

2.4.3.1 O treinamento da rede Perceptron de Múltiplas Camadas

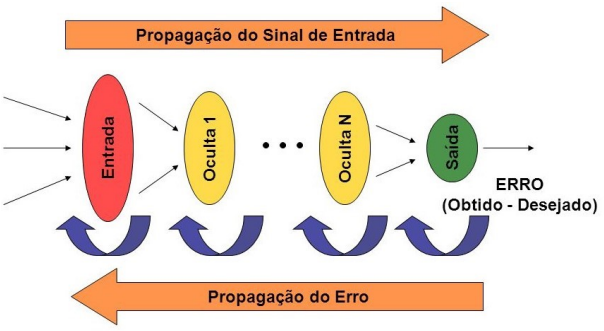
O treinamento feito em redes PMC é feito através da aplicação do algoritmo *backpropagation*, conhecido também por regra do Delta generalizada. A sua aplicação consiste em dois passos: propagação adiante (*forward*) e propagação reversa (*backward*)(Silva; Spatti; Flauzino, 2010).

O primeiro passo consiste na alimentação da rede, camada por camada, iniciando com uma amostra dos sinais $\{x_1, x_2, \ldots, x_n\}$ que resultará na produção de um conjunto respectivo de saída. Esse primeiro passo leva em consideração somente os valores atuais dos pesos sinápticos e limiares dos neurônios, que não são alterados nesse primeiro momento.

Após essa alimentação, as saídas obtidas são comparadas com as saídas desejadas, que devem estar disponíveis pois se trata de um treinamento supervisonado. As saídas produzidas nos n neurônios são utilizadas para obter os respectivos desvios em relação as n saídas desejadas, de modo que esses desvios são utilizados para atualizar os pesos.

Com os desvios obtidos, é possivel realizar a segunda fase do método, que é a atualização dos pesos através dos desvios. A aplicação das duas fases ajustam os pesos e limiares a cada iteração, reduzindo gradativamente a soma dos erros produzidos pelas respostas. O fluxo desses dados pode ser observado na Figura 8

Figura 8 – Estrutura - *Backpropagation*

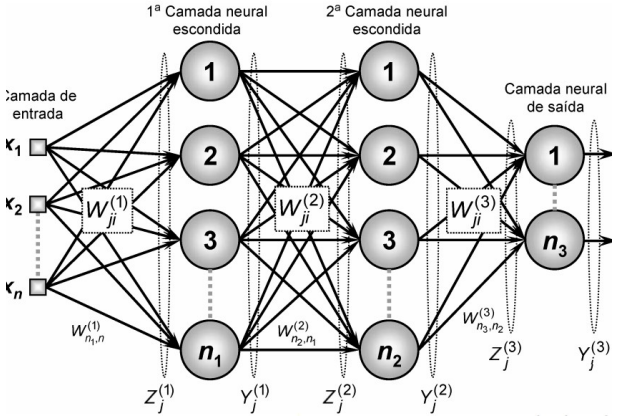


Fonte: Google imagens

2.4.3.2 O algorítmo *backpropagation*

O algorítmo *backpropagation* utiliza diversas variaveis em sua aplicação, de forma que elas devem ser definidas antecipadamente para melhor entendimento. Essas variáveis se encontram distribuidas e posicionadas na Figura 9.

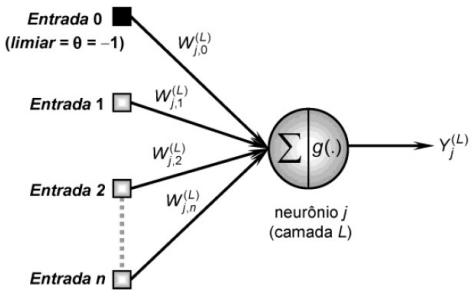
Figura 9 – Estrutura - *Backpropagation com variáveis*



Fonte: Adaptado de Silva, Spatti e Flauzino(2010)

Com base na Figura 9, será considerado cada um dos $\{j\}$ neurônios que pertencem a cada uma das $\{L\}$ camadas, com uma função $g(.)$ que representa a função de ativação presente em cada neurônio, que deve ser diferenciável em todo o seu domínio, como a tangente hiperbólica ou a logística (Batista, 2012), conforme a Figura 10

Figura 10 – Configuração de neurônio utilizado na derivação do algorítmo *backpropagation*



Fonte: Adaptado de Silva, Spatti e Flauzino(2010)

Com base nas Figuras 9 e 10, assume-se a seguinte termonolgia para os seus parâmetros:

- $W_{ji}^{(L)}$ São matrizes de pesos cujos elementos denotam o valor do peso sináptico conectando o j-ésimo neurônio da camada (L) ao i-ésimo neurônio da camada (L-1). Para a topologia ilustrada na Figura 9, tem-se:

$W_{ji}^{(3)}$ é o peso sináptico conectando o j-ésimo neurônio da camada de saída ao i-ésimo neurônio da camada 2;

$W_{ji}^{(2)}$ é o peso sináptico conectando o j-ésimo neurônio da camada escondida 2 ao i-ésimo neurônio da camada 1;

$W_{ji}^{(1)}$ é o peso sináptico conectando o j-ésimo neurônio da camada escondida 1 ao i-ésimo neurônio da camada de entrada.

- $Z_{ji}^{(L)}$ São vetores cujo os elementos denotam a entrada ponderada em relação ao j-ésimo neurônio da camada L, os quais são definidos por:

$$Z_j^{(1)} = \sum_{i=0}^{n} W_{j,i}^{(1)}.x_i \Leftrightarrow Z_j^{(1)} = W_{j,0}^{(1)}.x_0 + W_{j,1}^{(1)}.x_1 + \cdots + W_{j,n}^{(1)}.x_n; \tag{2.15}$$

$$Z_j^{(2)} = \sum_{i=0}^{n} W_{j,i}^{(2)}.y_i^{(1)} \Leftrightarrow Z_j^{(2)} = W_{j,0}^{(2)}.y_0^{(1)} + W_{j,1}^{(2)}.y_1^{(1)} + \cdots + W_{j,n}^{(2)}.y_n^{(1)}; \tag{2.16}$$

$$Z_j^{(3)} = \sum_{i=0}^{n} W_{j,i}^{(3)}.y_i^{(2)} \Leftrightarrow Z_j^{(3)} = W_{j,0}^{(3)}.y_0^{(2)} + W_{j,1}^{(3)}.y_1^{(2)} + \cdots + W_{j,n}^{(3)}.y_n^{(2)}. \tag{2.17}$$

- $Y_j^{(L)}$ São vetores cujos elementos denotam a saída do j-ésimo neurônio em relação à camada L, os quais são definidos por:

$$Y_j^{(1)} = g\left(Z_j^{(1)}\right); \tag{2.18}$$

$$Y_j^{(2)} = g\left(Z_j^{(2)}\right); \tag{2.19}$$

$$Y_j^{(3)} = g\left(Z_j^{(3)}\right). \tag{2.20}$$

Considerando a Figura 9, a função erro quadrático que é utilizada para medir o desempenho local associado aos resultados produzidos pelos neurônios de saída em relação ao conjunto de entradas, é definodo por:

$$E(k) = \frac{1}{2}\sum_{j=1}^{n3}\left(d_j(k) - y_j^{(3)}(k)\right)^2. \tag{2.21}$$

Onde $y_j^{(3)}(k)$ é o valor obtido pelo j-ésimo neurônio da saída da rede, produzido pela k-ésima entrada, euquanto $d_j(k)$ é o valor desejado.

Dessa forma, considerando que o conjunto de treinamento possua p amostras, o erro quadrático médio fica definido por:

$$E_M = \frac{1}{p}\sum_{k=1}^{p} E(k). \tag{2.22}$$

O ajuste dos pesos é baseado no método da descida mais íngrime, ou seja, utilizara o gradiente da função erro quadrático definido em 2.21.

2.4.3.2.1 Ajuste dos pesos da camada de saída

A atualização da camada de saída consiste na aplicação da correção $\Delta w_i j^{\{3\}}$, que é obtido através do gradiente $\nabla E^{\{3\}}$, para atualizar a matriz de pesos da camada de saída $w_i j^{\{3\}}$, com o fim de minimizar o erro presente na mesma. Através da regra da cadeia, o gradiente pode ser expresso por:

$$\nabla E^{(3)} = \frac{\partial E}{\partial w_i j^{(3)}} = \frac{\partial E}{\partial Y_j^{(3)}} . \frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} . \frac{\partial I_j^{(3)}}{\partial w_j i^{(3)}}. \tag{2.23}$$

Por meio das definições anteriores, obtém-se:

$$\frac{\partial I_j^{(3)}}{\partial w_j i^{(3)}} = Y_i^{(2)}; \tag{2.24}$$

$$\frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} = g'\left(I_j^{(3)}\right); \tag{2.25}$$

$$\frac{\partial E}{\partial Y_j^{(3)}} = -\left(d_j - Y_j^{(3)}\right). \tag{2.26}$$

Com $g'(.)$ sendo a derivada de primeira ordem da função de ativação. Através das equações (2.24), (2.25) e (2.26), pode-se reescrever a equação (2.23).

$$\frac{\partial E}{\partial w_i j^{(3)}} = -\left(d_j - Y_j^{(3)}\right) . g'\left(I_j^{(3)}\right) . Y_i^{(2)}. \tag{2.27}$$

Através desse gradiente, o ajuste da matriz de pesos $w_i j^{\{3\}}$ deve ser efetuado em direção oposta ao gradiente, com a finalidade de minimizar o erro, ou seja:

$$\Delta w_i j^{(3)} = -\eta . \frac{\partial E}{\partial w_i j^{(3)}} \Leftrightarrow \Delta w_i j^{(3)} = \eta . \delta_j^{(3)} . Y_i^{(2)}. \tag{2.28}$$

Onde $\delta_j^{(3)} = \left(d_j - Y_j^{(3)}\right) . g'\left(I_j^{(3)}\right)$ é definido como gradiente local em relação ao j-ésimo neurônio da camada de saída e η é a taxa de aprendizagem do algorítmo

Definindo a equação (2.28), podemos definir a atualização dos pesos da camada de saída como:

$$w_i j^{(3)}(t+1) = w_i j^{(3)}(t) + \eta . \delta_j^{(3)} . Y_i^{(2)}. \tag{2.29}$$

2.4.3.2.2 Ajuste dos pesos das camadas intermediárias

Diferente da camada de saída, os neurônios da camada intermediária não tem acesso direto as saídas desejadas. Para suprir essa condição, os ajustes são feitos através de estimativas de erros da camada imediatamente posterior, seja ela uma camada de saída ou uma camada oculta, pois ja terá os seus pesos ajustados nessa etapa do algorítmo. A rede em questão é retratada na Figura 9 e apresenta duas camadas intermediárias.

- Ajuste dos pesos da segunda camada oculta

Essa etapa do algorítmo consiste na atualização dos pesos $w_{ji}^{(2)}$, com o objetivo de minimizar o erro produzido entre a saída da rede em relação a retropropagação do erro dos ajustes da camada de saída. Com isso o gradiente $\nabla E^{(2)}$ é definido por:

$$\nabla E^{(2)} = \frac{\partial E}{\partial w_i j^{(2)}} = \frac{\partial E}{\partial Y_j^{(2)}} . \frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} . \frac{\partial I_j^{(2)}}{\partial w_j i^{(2)}}. \tag{2.30}$$

Considerando as definições estabelecidas anterioemente, tem-se:

$$\frac{\partial I_j^{(2)}}{\partial w_j i^{(2)}} = Y_i^{(1)}; \tag{2.31}$$

$$\frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} = g'\left(I_j^{(2)}\right); \tag{2.32}$$

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n3} \frac{\partial E}{\partial I_k^{(3)}} . \frac{\partial I_k^{(3)}}{\partial Y_j^{(2)}} = \sum_{k=1}^{n3} \frac{\partial E}{\partial I_k^{(3)}} . \frac{\partial \left(\sum_{k=1}^{n3} W_{kj}^{(3)} . Y_j^{(2)}\right)}{\partial Y_j^{(2)}}. \tag{2.33}$$

Onde a equação (2.33) resulta em:

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n3} \frac{\partial E}{\partial I_k^{(3)}} . W_{kj}^{(3)}. \tag{2.34}$$

Enquanto $\frac{\partial E}{\partial I_k^{(3)}}$ pode ser definida por:

$$\frac{\partial E}{\partial I_k^{(3)}} = -\left(d_j - Y_j^{(3)}\right) . g'\left(I_j^{(3)}\right) = -\delta_j^{(3)}. \tag{2.35}$$

Substituindo em 2.35 em 2.34, tem-se:

$$\frac{\partial E}{\partial Y_j^{(2)}} = -\sum_{k=1}^{n3} \delta_j^{(3)} . W_{kj}^{(3)}. \tag{2.36}$$

Substituindo 2.31, 2.32 e 2.36 em 2.30, temos:

$$\nabla E^{(2)} = -\sum_{k=1}^{n3} \left(\delta_j^{(3)} . W_{kj}^{(3)}\right) . g'\left(I_j^{(2)}\right) . Y_i^{(1)}. \tag{2.37}$$

Desse modo, o ajuste dos pesos $W_{ji}^{(2)}$, é efetuado em direção posta ao gradiente para minimizar o erro, resultando em:

$$\Delta W_{ij}^{(2)} = -\eta . \nabla E^{(2)} \Leftrightarrow \Delta W_{ij}^{(2)} = \eta . \delta_j^{(2)} . Y_i^{(1)}. \tag{2.38}$$

Onde $\delta_j^{(2)} = \left(\sum_{k=1}^{n3} \delta_j^{(3)}.W_{kj}^{(3)}\right).g'\left(I_j^{(2)}\right)$ é definido como gradiente local em relação ao j-ésimo neurônio da segunda camada intermediária.

Definindo a equação (2.38), podemos definir a atualização dos pesos da segunda camada oculta:

$$w_i j^{(2)}(t+1) = w_i j^{(2)}(t) + \eta.\delta_j^{(2)}.Y_i^{(1)}. \tag{2.39}$$

- Ajuste dos pesos da primeira camada oculta

A última etapa do algorítmo consiste na atualização dos pesos $w_{ji}^{(1)}$, com o intuito de minimizar o erro entre a saída da rede em relação a retropropagação do erro dos ajustes da segunda camada. Com isso o gradiente $\nabla E^{(1)}$ pode ser representado como:

$$\nabla E^{(1)} = \frac{\partial E}{\partial w_i j^{(1)}} = \frac{\partial E}{\partial Y_j^{(1)}}.\frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}}.\frac{\partial I_j^{(1)}}{\partial w_j i^{(1)}}. \tag{2.40}$$

Por meio das definições anteriormente apresentadas, obtém-se:

$$\frac{\partial I_j^{(1)}}{\partial w_j i^{(1)}} = x_i; \tag{2.41}$$

$$\frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}} = g'\left(I_j^{(1)}\right); \tag{2.42}$$

$$\frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{n2} \frac{\partial E}{\partial I_k^{(2)}}.\frac{\partial I_k^{(2)}}{\partial Y_j^{(1)}} = \sum_{k=1}^{n2} \frac{\partial E}{\partial I_k^{(2)}}.\frac{\partial\left(\sum_{k=1}^{n2} W_{kj}^{(2)}.Y_j^{(1)}\right)}{\partial Y_j^{(1)}}. \tag{2.43}$$

Onde a equação (2.43) resulta em:

$$\frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{n2} \frac{\partial E}{\partial I_k^{(2)}}.W_{kj}^{(2)}. \tag{2.44}$$

Enquanto $\frac{\partial E}{\partial I_k^{(2)}}$ pode ser definida por:

$$\frac{\partial E}{\partial I_k^{(2)}} = -\left(d_j - Y_j^{(3)}\right).g'\left(I_j^{(3)}\right) = -\delta_j^{(2)}. \tag{2.45}$$

Substituindo em 2.45 em 2.44, tem-se:

$$\frac{\partial E}{\partial Y_j^{(1)}} = -\sum_{k=1}^{n2} \delta_j^{(2)}.W_{kj}^{(2)}. \tag{2.46}$$

Através dos resultados obtidos em 2.41, 2.42 e 2.46 pode-se definir 2.40 como:

$$\nabla E^{(1)} = -\sum_{k=1}^{n2} \left(\delta_j^{(2)}.W_{kj}^{(2)}\right).g'\left(I_j^{(1)}\right).x_i. \tag{2.47}$$

Desse modo, o ajuste dos pesos $W_{ji}^{(1)}$, é efetuado em direção posta ao gradiente para minimizar o erro, resultando em:

$$\Delta W_{ij}^{(1)} = -\eta.\nabla E^{(1)} \Leftrightarrow \Delta W_{ij}^{(1)} = \eta.\delta_j^{(1)}.x_i. \tag{2.48}$$

Onde $\delta_j^{(1)} = \left(\sum_{k=1}^{n3} \delta_j^{(2)}.W_{kj}^{(2)}\right).g'\left(I_j^{(1)}\right)$ é definido como gradiente local em relação ao j-ésimo neurônio da primeira camada intermediária.

Definindo a equação (2.48), podemos definir a atualização dos pesos da primeira camada oculta:

$$w_i j^{(1)}(t+1) = w_i j^{(1)}(t) + \eta.\delta_j^{(1)}.x_i. \tag{2.49}$$

3 METODOLOGIA

3.1 INTRODUÇÃO

Neste capítulo será apresentado o método utilizado para resolução das equações diferenciais parciais (EDP) através do uso de redes neurais e da retropropagação restrita (CPROP).

3.2 RETROPROPAGAÇÃO RESTRITA - CPROP

Foi apresentado no capítulo anterior, formas de alimentação da rede neural e atualização dos pesos. Com os ajustes corretos de uma rede feedforward e a aplicação do algorítimo backpropagation foi possivel resolver equações diferenciais através de redes neurais (Lagaris; Likas; Fotiadis, 1998).

O backpropagation clássico busca realizar uma otimização sem restrição trabalhando com todos os pesos da rede. Por sua vez, A Retropropagação Restrita (Constrainer Backpropagation - CPROP) trabalha com otimização com restrição, envolvendo a minimização de uma função objetivo escalar que é sujeita a um conjunto de restrições (Rudd; Muro; Ferrari, 2014), de modo que parte da rede será otimizada enquanto parte permanecerá constante.

O algoritmo CPROP foi baseado por meio do treinamento algébrico (Ferrari; Stengel, 2005), onde o método da eliminação direta (Stengel, 1986) pode ser utilizado para treinar redes neurais artificiais com restrições de igualdade, que são satisfeitas ao longo das sessões de treinamento.

Dessa forma, a Retropropagação Restrita pode ser utilizada para aproximar e adaptar soluções de equações diferenciais parciais (EDPs), utilizando a saída N como aproximação para uma solução de uma EDP. Essa aproximação pode ser descrita como, $N = h(\vec{x})$, sendo $h(\vec{x})$ uma função escalar suave de r variáveis independentes baseada num operador diferencial e satisfaz o conjunto de restrições obtidas pelas condições iniciais (IC) ou condições de contorno (BC), onde $\vec{x} \in \mathbb{R}^{r\times 1}$ e $h : \mathbb{R}^r \to \mathbb{R}$.

Para definir o funcionamento do método, teremos um conjunto de entradas dado por

$$\vec{x} = (\mathbf{x}_{(1)},\ \ldots \mathbf{x}_{(r)}). \tag{3.1}$$

A partir da entrada desse conjunto alimentando a rede, ela é particionada de modo que uma parte da rede possui K_S e outra contém K_L neurônios, ou seja:

$$K_S + K_L = K. \tag{3.2}$$

sendo K o número de neurônios presentes na rede.

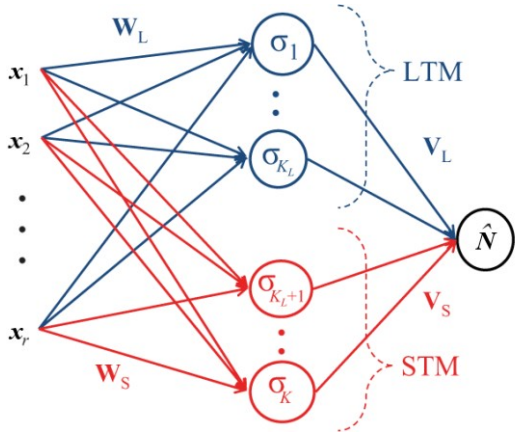
A primeira partição utiliza os dados do interior da malha e pode incrementar alguns parâmetros ao longo do tempo, por isso será chamado de memória de curto prazo (STM). Com

isso a função objetiva a ser minimizada é obtida de um conjunto de entrada-saída, denominada por $T_S = \{\vec{x}_k,\ \hat{N}_k\}_{k=1,\ldots,N_S}$, sendo $\hat{N}_k$ a saída fornecida pela rede e $N_S = K_S$.

A outra parte irá preservar ao longo do procedimento alguns parâmetros da rede, além de atender as Condições Iniciais (IC) ou as Condições de Contorno (BC). Ele contém informações do conjunto de entrada-saída e derivadas a serem preservadas ao longo do procedimento, recebendo o nome de memória de longo prazo (LTM) e sendo definido por $T_L = \{\vec{x}_l, h^n(x_l)\}_{l=1,\ldots,N_L}$, onde $h^n(.)$ representa a n-ésima derivada de h em relação as entradas e $N_L = K_L$.

A partição da rede nos grupamentos LTM e STM pode ser representada pela Figura 11

Figura 11 – Estrutura - CPROP



Fonte: Adaptado de Rudd, Muro e Ferrari(2014)

No caso do CPROP, a saída da rede pode ser definida respeitando a partição dos conjuntos STM e LTM, desta forma:

$$\hat{N} = \sigma(Z_L)\mathbf{V}_L^T + \sigma(Z_S)\mathbf{V}_S^T \approx h(\vec{x}). \tag{3.3}$$

Com $Z_L = \vec{x}^T\mathbf{W}_L^T + \mathbf{b}_L^T$ e $Z_S = \vec{x}^T\mathbf{W}_S^T + \mathbf{b}_S^T$. Onde $b \in \mathbb{R}^{K\times 1}$, representa o bias, $W \in \mathbb{R}^{K\times r}$, representa os pesos da camada oculta com r sendo a dimensão de $\vec{x}$ e $V \in \mathbb{R}^{1\times K}$ contém os pesos da camada de saída, de modo que as terminações S e L fazem referência aos conjuntos STM e LTM, respectivamente, dessa forma, têm-se:

- $b_S \in \mathbb{R}^{K_S\times 1}$
- $b_L \in \mathbb{R}^{K_L\times 1}$
- $W_S \in \mathbb{R}^{K_S\times r}$
- $W_L \in \mathbb{R}^{K_L\times r}$
- $V_S \in \mathbb{R}^{1\times K_S}$
- $V_L \in \mathbb{R}^{1\times K_L}$

Desse modo os pesos de STM e LTM serão organizados em dois vetores, tendo o vetor dos pesos de $p_L = \{b_L, W_L, V_L\}$ e os pesos $p_S = \{b_S, W_S, V_S\}$, que são referentes aos conjuntos STM e LTM, respectivamente.

Baseada na equação (3.3) e nas suas derivadas, os dados LTM podem ser aplicados em uma função implícita $c : \mathbb{R}^M \to \mathbb{R}^V$, onde $V = N_L + N_S$, de modo que os dados LTM podem ser preservados na restrição de igualdade.

$$\mathbf{c}(\mathbf{p_L}, \mathbf{p_S}) = 0. \tag{3.4}$$

Além disso, o treinamento irá preservar T_L enquanto minimiza uma função objetivo $e : \mathbb{R}^M \to \mathbb{R}$ obtida a partir dos dados T_S, podendo obter uma otimização com restrição.

$$\begin{aligned} &min\ \mathbf{e}(\mathbf{p_L}, \mathbf{p_S}); \\ &s.t.\ \mathbf{c}(\mathbf{p_L}, \mathbf{p_S}) = 0. \end{aligned} \tag{3.5}$$

Vale resaltar que se (3.4) satisfaz o teorema da função implicita (T.F.I) (Guidorizzi, 2006), então:

$$\mathbf{p_L} = C(\mathbf{p_S}). \tag{3.6}$$

Logo

$$E(\mathbf{p_S}) = \mathbf{e}(C(\mathbf{p_S}), \mathbf{p_S}). \tag{3.7}$$

Dessa forma os valores de $\mathbf{p_S}$ podem ser obtidos de forma independente enquanto os valores de $\mathbf{p_L}$ são obtidos apartir de $\mathbf{p_S}$,já otimizado, utilizando (3.6) (Rudd; Muro; Ferrari, 2014). Além disso, a otimização da função (3.7) pode ser obtida numericamente, de modo que ela pode ser reescrita como:

$$E(\mathbf{p_S}) = \frac{1}{2}\varepsilon^T \varepsilon. \tag{3.8}$$

Onde $\varepsilon_{(j)}$ equivale erro do j-ésimo termo em T_S.

3.3 ILUSTRAÇÃO DO MÉTODO

A abordagem do CPROP oferece um modelo para resolver Equações diferenciais parciais utilizando redes neurais, uma vez que elas podem ser treinadas para minimizar o erro definido pelo operador diferencial e ao mesmo tempo satisfaz as condições inicais e de contorno.

As condições de contorno podem ser supridas através das restrições de igualdade em (3.4) por meio do treinamento algébrico (Ferrari; Stengel, 2005). Essas especificidades do método, permitem a resolução de equações diferenciais do tipo eliptica linear ou não linear e EDPs do tipo parabólicas de segunda ordem, que podem ser escritas como

$$\mathscr{L}_n[N(\vec{x})] = f_n(\vec{x}). \tag{3.9}$$

onde $\mathscr{L}_n$ é um operador diferencial e $f_n : \mathbb{R}^M \to \mathbb{R}$ representa uma função.

Em cada tipo de EDP, os conjuntos T_L são redefinidos, onde ele supri as condições de contorno nas equações elípticas e as condições iniciais nas equações parabolicas. Por sua vez, T_S irá fornecer os pontos do interior do domínio. Com isso, as equações elípticas e parabólicas irão apresentar como solução o termo $\hat{N}(\mathbf{p})$, permitindo reescrever a função objetivo (3.8) em termos de erro do operador diferencial como:

$$\varepsilon_{(k)} \equiv \left\{ f_n(\vec{x}_k) - \mathscr{L}_n[\hat{N}] \right\}. \tag{3.10}$$

com $\varepsilon_{(k)}$ representando o k-ésimo elemento de erro ε.

3.3.1 Elípticas

Considerando o vetor de entradas (variáveis independentes) $\vec{x} = (\mathbf{x}_{(1)}, \ldots \mathbf{x}_{(r)})$, de modo que $\vec{x} \in \mathscr{I} \subset \Re^r$, o CPROP irá obter a função objetiva (3.5) através da aplicação do operador diferencial na solução aproximada da rede (3.3)

$$\frac{\partial^k \hat{N}}{\partial \mathbf{x}_{(1)}^{m_1}, \ \ldots \ , \partial \mathbf{x}_{(r)}^{m_r}} = \sigma^k(Z_L).\Lambda_L.\mathbf{V}_L^T + \sigma^k(Z_S).\Lambda_S.\mathbf{V}_S^T. \tag{3.11}$$

onde modo que $\sigma^k(.)$ representando a k-ésima derivada da função de ativação, $k = m_1 + \ldots + m_r$ e Λ_S representa o produto da matrix diagonal de W_S. De forma que

$$\Lambda_S = \prod_{j=1}^{r} \omega_{S_j}^{m_j}. \tag{3.12}$$

tomando $\omega_{S_j}^{m_j}$ como representação da j-ésima coluna de W_S.

3.3.2 Parabólicas

Nessa seção, o operador diferencial define uma solução para Equações Diferenciais Parabólicas, onde o vetor de entradas $\vec{x} \in \mathscr{I}$ onde $\mathscr{I} = \mathscr{H} \times \left[t_0, t_f\right) \subset \Re^r$. Apartir da equação (3.3), a solução das Equações Parabólicas pode ser escrita como:

$$\hat{N} = \tilde{h} + q.\left[\sigma(Z_L).\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right]. \tag{3.13}$$

De modo que $\tilde{h} : \Re^r \to \Re$ e $q : \Re^r \to \Re$ são diferenciáveis, com $\tilde{h}$ satisfazendo as condições de contorno de qualquer entrada pertencente ao vetor $\vec{x} \in \mathscr{I}$. Quando as condições de contorno não variam ao longo do tempo, através de (3.13), elas ja ficam automaticamente supridas, deixando as Condições Iniciais (IC) como única restrição de igualdade.

A derivada de primeira ordem e de segunda ordem da equação (3.13) podem ser definidas, respectivamente, como:

$$\frac{\partial \hat{N}}{\partial \mathbf{x}_{(\mathbf{i})}} = \frac{\partial \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}} + \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right] + q.\left[\sigma'(Z_L).\omega_{L_i}.\mathbf{V}_L^T + \sigma'(Z_S).\omega_{S_i}.\mathbf{V}_S^T\right]; \tag{3.14}$$

$$\begin{aligned}\frac{\partial^2 \hat{N}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2} =& \frac{\partial^2 \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2} + \frac{\partial^2 q}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}.\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right] ... \\ &+ \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma'(Z_L).\omega_{L_i}.\mathbf{V}_L^T + \sigma'(Z_S).\omega_{S_i}.\mathbf{V}_S^T\right] ... \\ &+ \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma'(Z_L).\omega_{L_i}.\mathbf{V}_L^T + \sigma'(Z_S).\omega_{S_i}.\mathbf{V}_S^T\right] ... \\ &+ q.\left[\sigma''(Z_L).\omega_{L_j}^2.\mathbf{V}_L^T + \sigma''(Z_S).\omega_{S_j}^2.\mathbf{V}_S^T\right].\end{aligned} \tag{3.15}$$

3.4 CÁLCULO DO GRADIENTE

3.4.1 Elíptica

Como foi representado em (3.9), ao trabalhar com equações elípticas, $\mathscr{L}_n[N(\vec{x})]$ denota um operador elíptico, com as condições de contorno definidas por:

$$\mathscr{B}[N(\vec{x})] = h(\vec{x}). \tag{3.16}$$

De modo que $\mathscr{B}$ é um operador diferencial de ordem inferior a $\mathscr{L}_n$ e $h : \mathbb{R}^r \to \mathbb{R}$ é uma função contínua e conhecida que satisfaz as condições de contorno.

Para determinar a solução através do CPROP, se faz nescessario determinar o gradiente da saída da rede em função de suas entradas. Dito isso, a solução das Equações Diferenciais Elipticas podem ser definidas através de (3.11), de modo que as derivadas em relação aos pesos serão determinadas por:

$$\frac{\partial}{\partial p_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \frac{\partial}{\partial p_S}\left[\sigma^k(Z_L)\Lambda_L\mathbf{V}_L^T + \sigma^k(Z_S)\Lambda_S\mathbf{V}_S^T\right]. \tag{3.17}$$

Separando as derivadas de LTM e STM, obtemos dois conjuntos distintos, sendo eles $g_1(p_S)$ e $g_2(p_S)$, respectivamente:

$$g_1(p_S) = \frac{\partial}{\partial p_S}\left[\sigma^k(Z_L)\Lambda_L\mathbf{V}_L^T\right]; \tag{3.18}$$

$$g_2(p_S) = \frac{\partial}{\partial p_S}\left[\sigma^k(Z_S)\Lambda_S\mathbf{V}_S^T\right]. \tag{3.19}$$

A derivada de $g_2(p_S)$ em relação a camada de entrada é definida por:

$$g_2(W_S) = \frac{\partial}{\partial W_S}\left[\sigma^k(Z_S)\Lambda_S \mathbf{V}_S^T\right]. \tag{3.20}$$

que resulta em:

$$g_2(W_S) = \left[a_{i,j}.\sigma^k(Z_S) + \sigma^{k+1}(Z_S).\Lambda_S.x\right].\mathbf{V}_s \tag{3.21}$$

.

Onde

$$a_{i,j} = k.\mathbf{W}_{sx}^{k-1}. \tag{3.22}$$

As derivadas das entradas do Bias e da camada de saída pode sem definidas, respectivamente por:

$$g_2(B_S) = \sigma^{k+1}(Z_S).\Lambda_S.\mathbf{V}_s; \tag{3.23}$$

$$g_2(V_S) = \sigma^k(Z_S).\Lambda_S. \tag{3.24}$$

Enquanto o termo $g_1(p_S)$, por não depender de qualquer p_S, é utilizada uma condição como definido em (3.6). Quando as condições de contorno são impostas sobre a solução aproximada da rede, ela pode ser escrito como:

$$\mathscr{B}[N(\vec{x})] = \mathscr{B}[\sigma(Z_L)].\mathbf{V}_L^T + \mathscr{B}[\sigma(Z_S)].\mathbf{V}_S^T. \tag{3.25}$$

Com o operador diferencial $\mathscr{B}$.

Através do treinamento algébrico, a equação (3.25), utilizando os pontos pertencentes a T_L, é organizada em um sistema linear de equações. Dessa forma é organizada uma rede que satisfaz T_L em todos os momentos, desde que o treinamento satisfaça a seguinte restrição de igualdade:

$$h = \Psi \mathbf{V}_L^T + \Omega \mathbf{V}_S^T. \tag{3.26}$$

Onde,

$$\mathbf{h}_{(l)} \equiv h(\vec{x}_l); \tag{3.27}$$

$$\Psi = \mathscr{B}\left[\sigma(\vec{x}_l^T.\mathbf{W}_{Ll}^T + \mathbf{b}_{Ll}^T)\right]. \tag{3.28}$$

$$\Omega = \mathscr{B}\left[\sigma(\vec{x}_l^T.\mathbf{W}_{Sl}^T + \mathbf{b}_{Sl}^T)\right]. \tag{3.29}$$

com $\vec{x}_l \in LTM$, podemos suprir a condição de dependência de modo que (3.26) pode ser reescrita como:

$$\mathbf{V}_L^T = \Psi^{-1}.\left[h - \Omega \mathbf{V}_S^T\right]. \tag{3.30}$$

onde Ψ é uma matriz inversível que pode ser construida através do treino algébrico (Ferrari; Jensenius, 2008). Mais informaçoes sobre a matriz Ψ podem ser obtidas no anexo B.

Utilizando (3.30), podemos determinar as derivadas de $g_1(p_S)$ em relação a qualquer um dos pesos. Em relação a camada de entrada, obtemos:

$$g_1(W_S) = \frac{\partial}{\partial W_S}\left[\sigma^k(Z_L).\Lambda_L.\mathbf{V}_L\right]; \tag{3.31}$$

$$g_1(W_S) = \left[\sigma^k(Z_L).\Lambda_L.\right]\frac{\partial \mathbf{V}_L}{\partial W_S} \tag{3.32}$$

. Onde,

$$\frac{\partial \mathbf{V}_L}{\partial W_S} = -\Psi^{-1}.x_l.\Omega'.\mathbf{V}_S^T. \tag{3.33}$$

dessa forma, substituindo (3.33) em (3.32),obtemos:

$$g_1(W_S) = -\left[\sigma^k(Z_L).\Lambda_L.\right].\Psi^{-1}.x_l.\Omega'.\mathbf{V}_S^T. \tag{3.34}$$

o que nos leva a:

$$\frac{\partial}{\partial W_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \left[\left[\sigma^k(Z_L).\Lambda_L.\right]\frac{\partial \mathbf{V}_L}{\partial W_S} + \frac{\partial}{\partial W_S}\left[\sigma^k(Z_S)\Lambda_S\right]\mathbf{V}_S^T\right]. \tag{3.35}$$

De forma semelhante, as derivadas do Bias pode ser calculada através de (3.30), ou seja:

$$g_1(b_S) = \frac{\partial}{\partial b_S}\left[\sigma^k(Z_L).\Lambda_L.\mathbf{V}_L\right]; \tag{3.36}$$

$$g_1(b_S) = \left[\sigma^k(Z_L).\Lambda_L.\right]\frac{\partial \mathbf{V}_L}{\partial b_S}; \tag{3.37}$$

$$\frac{\partial \mathbf{V}_L}{\partial b_S} = -\Psi^{-1}.\Omega'.\mathbf{V}_S^T; \tag{3.38}$$

$$g_1(b_S) = -\sigma^k(Z_L).\Lambda_L.\Psi^{-1}.\Omega'.\mathbf{V}_S^T. \tag{3.39}$$

resultando em:

$$\frac{\partial}{\partial b_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \left[\left[\sigma^k(Z_L).\Lambda_L.\right]\frac{\partial \mathbf{V}_L}{\partial b_S} + \frac{\partial}{\partial b_S}\left[\sigma^k(Z_S)\right]\Lambda_S.\mathbf{V}_S^T\right]. \tag{3.40}$$

O mesmo ocorre com a derivada de $g_1(p_S)$ em relação a V_S,

$$g_1(V_S) = \frac{\partial}{\partial V_S}\left[\sigma^k(Z_L).\Lambda_L.\mathbf{V}_L\right]; \tag{3.41}$$

$$g_1(V_S) = \left[\sigma^k(Z_L).\Lambda_L.\right]\frac{\partial \mathbf{V}_L}{\partial V_S}; \tag{3.42}$$

$$\frac{\partial \mathbf{V}_L}{\partial V_S} = -\Psi^{-1}.\Omega; \tag{3.43}$$

$$g_1(V_S) = -\sigma^k(Z_L).\Lambda_L.\Psi^{-1}.\Omega. \tag{3.44}$$

Que permite definir:

$$\frac{\partial}{\partial V_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \left[\left[\sigma^k(Z_L).\Lambda_L.\right]\frac{\partial \mathbf{V}_L}{\partial V_S} + \sigma^k(Z_S).\Lambda_S.\frac{\partial}{\partial V_S}\left[\mathbf{V}_S^T\right]\right]. \tag{3.45}$$

As substituições das respectivas derivadas de cada peso podem ser ser encontradas no Apêndice A

3.4.2 Parabólicas

Para determinar a solução das Equações Parabólicas utilizando o CPROP, é utilizado um processo semelhante ao das Equações Elípticas, desse modo, se faz nescessário determinar as derivadas de (3.14) e (3.15) em relação aos pesos. Para tal atualização iremos utilizar uma condição semelhante a (3.30):

$$\mathbf{V}_L^T = \Psi^{-1}.\left[z - \Omega \mathbf{V}_S^T\right]. \tag{3.46}$$

Sendo Ψ e Ω definidos em (3.28) e (3.29), respectivamente. Além disso,

$$z(\vec{x}_l) = \frac{u(\vec{x}_l) - \tilde{h}(\vec{x}_l)}{q(\vec{x}_l)}. \tag{3.47}$$

Onde $\tilde{h}(\vec{x}_l)$ e $q(\vec{x}_l)$ estão definidas na equação 3.13, $\vec{x}_l \in LTM$ e $u(\vec{x}_l)$ fornece as Condições Iniciais.

Utilizando (3.46), é possivel obter as derivadas de (3.14) e (3.15) em relação aos pesos de Entrada, Bias e Saída. Derivando (3.14) em relação aos pesos, encontramos

$$\frac{\partial}{\partial p_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) = \frac{\partial}{\partial p_S}\left(\frac{\partial \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}}\right) + \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.f_1(p_S) + q.f_2(p_S). \tag{3.48}$$

onde, para melhor organização, obtemos:

$$f_1(p_S) = \left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial p_S} + \frac{\partial \sigma(Z_S).\mathbf{V}_S^T}{\partial p_S}\right]. \tag{3.49}$$

$$f_2(p_S) = \left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial p_S} + \frac{\partial \sigma'(Z_S).\omega_{S_i}.\mathbf{V}_S^T}{\partial p_S}\right]. \tag{3.50}$$

derivando em relação aos pesos de Entrada, obtemos:

$$f_1(W_S) = \left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S)\right).\mathbf{V}_S^T\right]; \tag{3.51}$$

$$f_2(W_S) = \left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right]. \tag{3.52}$$

Substituindo (3.51) e (3.52) em (3.48)

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) =& \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S)\right).\mathbf{V}_S^T\right]... \\ &+ q.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right].\end{aligned} \tag{3.53}$$

Sendo $\frac{\partial \mathbf{V}_L^T}{\partial W_S}$ definido anteriormente em (3.33).

O mesmo ocorre em relação aos pesos do Bias, derivando (3.14), obtemos:

$$f_1(b_S) = \left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S)\right).\mathbf{V}_S^T\right]. \tag{3.54}$$

$$f_2(b_S) = \left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right]. \tag{3.55}$$

substituindo (3.54) e (3.55) em (3.48)

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) =& \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S)\right).\mathbf{V}_S^T\right]... \\ &+ q.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right].\end{aligned} \tag{3.56}$$

Com $\frac{\partial \mathbf{V}_L^T}{\partial b_S}$ definido em (3.38). Derivando (3.14) em relação aos pesos de Saída,

$$f_1(V_S) = \left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma(Z_S).\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right]. \tag{3.57}$$

$$f_2(V_S) = \left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).\omega_{S_i}.\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right]. \tag{3.58}$$

substituindo (3.57) e (3.58) em (3.48)

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) =& \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma(Z_S).\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right]... \\ &+ q.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).\omega_{S_i}.\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right].\end{aligned} \tag{3.59}$$

tendo $\frac{\partial \mathbf{V}_L^T}{\partial V_S}$ definido em (3.43).

A derivada de (3.15) em relação aos pesos vai apresentar resultados semelhantes das derivadas de (3.14), com o acréscimo de $f_3(p_S)$, de forma que:

$$\frac{\partial}{\partial p_S}\left(\frac{\partial^2 \hat{N}}{\partial \tilde{\mathbf{x}}^2}\right) = \frac{\partial}{\partial p_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x_{(i)}}^2}\right) + \frac{\partial^2 q}{\partial \mathbf{x_{(i)}}^2}.f_1(p_S) + \frac{\partial q}{\partial \mathbf{x_{(i)}}}.f_2(p_S) + \frac{\partial q}{\partial \mathbf{x_{(i)}}}.f_2(p_S) + q.f_3(p_S). \tag{3.60}$$

sendo $f_3(p_S)$ definido por:

$$f_3(p_S) = \left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial p_S} + \frac{\partial \sigma''(Z_S).\omega_{S_j}^2.\mathbf{V}_S^T}{\partial p_S}\right]. \tag{3.61}$$

como $f_1(p_S)$ e $f_2(p_S)$ ja foram determinados, para todos os pesos, basta definir as derivadas de $f_3(p_S)$. Derivando em relação aos pesos de Entrada,

$$f_3(W_S) = \left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma''(Z_S).\omega_{S_j}^2\right).\mathbf{V}_S^T\right]. \tag{3.62}$$

com $\frac{\partial \mathbf{V}_L^T}{\partial W_S}$ definido em (3.33). Substituindo (3.62), (3.51) e (3.52) em (3.60), obtemos:

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x_{(i)}}^2}\right) =& \frac{\partial}{\partial W_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x_{(i)}}^2}\right) + \frac{\partial^2 q}{\partial \mathbf{x_{(i)}}^2}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S)\right).\mathbf{V}_S^T\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right]... \\ &+ q.\left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \frac{\partial}{\partial W_S}\left(\sigma''(Z_S).\omega_{S_j}^2\right).\mathbf{V}_S^T\right].\end{aligned} \tag{3.63}$$

Derivando em relação aos pesos do Bias,

$$f_3(b_S) = \left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma''(Z_S)\right).\omega_{S_j}^2.\mathbf{V}_S^T\right] \tag{3.64}$$

com $\frac{\partial \mathbf{V}_L^T}{\partial b_S}$ definido em (3.38), substituindo (3.64), (3.54) e (3.55) em (3.60). Obtemos:

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x_{(i)}}^2}\right) =& \frac{\partial}{\partial b_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x_{(i)}}^2}\right) + \frac{\partial^2 q}{\partial \mathbf{x_{(i)}}^2}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S)\right).\mathbf{V}_S^T\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma(Z_S).\omega_{S_i}\right).\mathbf{V}_S^T\right]... \\ &+ q.\left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \frac{\partial}{\partial b_S}\left(\sigma''(Z_S)\right).\omega_{S_j}^2.\mathbf{V}_S^T\right].\end{aligned} \tag{3.65}$$

Derivando em relação aos pesos da camada de Saída,

$$f_3(V_S) = \left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma''(Z_S).\omega_{S_j}^2\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right] \tag{3.66}$$

com $\frac{\partial \mathbf{V}_L^T}{\partial V_S}$ definido em (3.43).Substituindo (3.66), (3.57) e (3.58) em (3.60). Obtemos:

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x_{(i)}}^2}\right) =& \frac{\partial}{\partial V_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x_{(i)}}^2}\right) + \frac{\partial^2 q}{\partial \mathbf{x_{(i)}}^2}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma(Z_S).\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).\omega_{S_i}.\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).\omega_{S_i}.\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right]... \\ &+ q.\left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma''(Z_S).\omega_{S_j}^2\frac{\partial \mathbf{V}_S^T}{\partial V_S}\right].\end{aligned} \tag{3.67}$$

Semelhante as derivadas obtidas em relação as equações elípticas as substituções das respectivas derivadas de cada peso podem ser ser encontradas no Apêndice B

4 RESULTADOS

4.1 INTRODUÇÃO

Este ponto do trabalho consiste em apresentar os exemplos que ilustram o funcionamento do CPROP para resolução de equações diferenciais parciais (EDPs).

Para esta aplicação em uma Equação Direfencial Parcial Elíptica, foi utilizada uma rede neural de 60 neurônios na camada oculta e somente um na camada de saída no problema 1. Como função de ativação, foi empregada a função Tangente Hiperbólica bem como o método de Levenberg-Marquadt para sua otimização. O método utilizado se encontra presente no Anexo A.

A taxa de aprendizagem utilizada nas simulações foi de $\eta = 10^{-4}$, dessa forma o programa continuará realizando as atualizações até atingir o erro mínimo ou até atingir o limite de ciclos permitido.

A implementação dessa rede foi feita no software R v. 1.3.1093 (R Core Team, 2020), em um notebook com 8 GB de memória, 2 TB de HD e processador Intel Core I5 de sétima geração. As características das soluções geradas pelo método $N_t(\overrightarrow{x})$ serão ilustradas para uma melhor visualização e do desvio em relação solução analítica $N_a(\overrightarrow{x})$. Dessa, a precisão da rede pode ser determinada pelo desvio das soluções $\Delta N(\overrightarrow{x}) = N_t(\overrightarrow{x}) - N_a(\overrightarrow{x})$.

4.2 CPROP APLICADO NA SOLUÇÃO DE UMA EQUAÇÃO ELÍPTICA

4.2.1 problema 1

$$\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} + \frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} + \alpha_n e^{\hat{N}} = \alpha_n \left[1 + x^2 + y^2 + \frac{4}{1 + x^2 + y^2} \right]. \tag{4.1}$$

com condição de contorno definida por $\mathbf{h}(\vec{x}) = log(1 + x^2 + y^2)$ com seu dominio definido de $[-1,1] \times [-1,1]$ onde $\alpha_n = 0.2n$ com $n = 0, \quad \ldots \quad ,5$.

A equação (4.1) pode ser aplicada em muitos processos dinâmicos em áreas como mecânica dos fluidos, eletrostática e termodinâmica, trabalhando com o fluxo de um fluido irrotacional, incomprensível e constante em duas dimensões ou processos de calor/difusão estacionários.

A região interna consiste em uma malha de 35×35 e a sua fronteira esta distribuida em 180 pontos equidistantes. A rede foi particionada como descrito na subseção 3.2, possuindo 20 neurônios pertencentes a STM e 40 neurônios pertencentes a LTM de modo que os pesos W_L e b_L foram inicializados com uma distribuição uniforme no intervalo $(-5,5)$

Como foi definido em (3.8) e (3.10), a função objetivo a ser minimizada será definida por

$$E(\mathbf{p_S}) = \frac{1}{2}.\sum_i \left\{ \alpha_n \left[1 + x^2 + y^2 + \frac{4}{1+x^2+y^2} \right] - \frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} - \frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} - \alpha_n e^{\hat{N}}] \right\}^2. \tag{4.2}$$

de modo que $\mathbf{p_s}$ representa qualquer um dos pesos. Onde todas as EDPs do tipo elípticas podem ser expressas pela derivada parcial em (3.11). A aplicação dessa derivada transforma a equação (4.2) gera os seguintes resultados:

$$\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} = \sigma''(Z_L)\mathbf{W}_{Lx}^2 \mathbf{V}_L + \sigma''(Z_S)\mathbf{W}_{Sx}^2 \mathbf{V}_S. \tag{4.3}$$

Ao derivar $\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}$, onde os termos W_{Lx} e W_{Sx} representam as derivadas dos pesos da camada de entrada respectivos a $\mathbf{x}$.

De forma semelhante

$$\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} = \sigma''(Z_L)\mathbf{W}_{Ly}^2 \mathbf{V}_L + \sigma''(Z_S)\mathbf{W}_{Sy}^2 \mathbf{V}_S. \tag{4.4}$$

com os termos W_{Ly} e W_{Sy}, que representam as derivadas dos pesos da camada de entrada respectivos a $\mathbf{y}$.

Sendo o termo $f_n(.) = \alpha_n \left[1 + x^2 + y^2 + \frac{4}{1+x^2+y^2} \right]$, um elemento independente em relação aos pesos, sua derivada é expressa por.

$$\frac{\partial f_n(.)}{\partial \mathbf{p_S}} = 0. \tag{4.5}$$

Dessa forma, a equação (4.2), ao ser derivada em relação aos pesos para obter o gradiente, resulta em:

$$\frac{\partial E(p_s)}{\partial \mathbf{p_S}} = 2.\varepsilon. \left\{ -\frac{\partial}{\partial p_s}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) - \frac{\partial}{\partial p_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) - \frac{\partial}{\partial p_S}\left(\partial \alpha_n e^{\hat{N}}\right) \right\}. \tag{4.6}$$

Sendo $\varepsilon = \left\{ \alpha_n \left[1 + x^2 + y^2 + \frac{4}{1+x^2+y^2} \right] - \frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} - \frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} - \alpha_n e^{\hat{N}} \right\}$.

Em relação ao peso W_S, que pertence ao conjunto *STM* , (4.6) pode ser representada como

$$\frac{\partial E(p_s)}{\partial \mathbf{W_S}} = 2.\varepsilon. \left\{ -\frac{\partial}{\partial W_s}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) - \frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) - \frac{\partial}{\partial W_S}\left(\partial \alpha_n e^{\hat{N}}\right) \right\}. \tag{4.7}$$

Em relação a (4.3) iremos obter

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial W_s} = \frac{\partial}{\partial W_s}\left[\sigma''(Z_L)\mathbf{W}_{Lx}^2 \mathbf{V}_L + \sigma''(Z_S)\mathbf{W}_{Sx}^2 \mathbf{V}_S \right]. \tag{4.8}$$

De modo que

$$g_1(W_S) = \frac{\partial}{\partial W_S}\left[\sigma''(Z_L).\mathbf{W}_{Lx}^2.\mathbf{V}_L \right]. \tag{4.9}$$

$$g_2(W_S) = \frac{\partial}{\partial W_S}\left[\sigma''(Z_S).\mathbf{W}_{Sx}^2.\mathbf{V}_S\right]. \tag{4.10}$$

O termo $g_2(W_s)$ pode ser definido como

$$g_2(W_S) = 2.\mathbf{W}_{Sx}.\sigma''(Z_S).\mathbf{V}_S + \sigma'''(Z_S).\mathbf{W}_{Sx}^2.\mathbf{V}_S.x. \tag{4.11}$$

Como foi representado de (3.32) a (3.34), podemos reescrever (4.9) como

$$g_1(W_s) = -\left[\sigma''(Z_L).\mathbf{W}_{Lx}^2.\right].\Psi^{-1}.\mathbf{V}_S^T.\Omega'.x_l. \tag{4.12}$$

com Ψ e Ω sendo definidos em 3.28 e 3.29, respectivamente.

Definindo $g_1(W_s)$ e $g_2(W_S)$, é possivel determinar (4.8).

$$\frac{\frac{\partial^2\hat{N}}{\partial\mathbf{x}^2}}{\partial W_s} = g_1(W_s) + g_2(W_s). \tag{4.13}$$

substituindo,

$$\frac{\frac{\partial^2\hat{N}}{\partial\mathbf{x}^2}}{\partial W_s} = \left(-\left[\sigma''(Z_L).\mathbf{W}_{Lx}^2.\right].\Psi^{-1}.\mathbf{V}_S^T.\Omega'.x_l\right) + \left(2.\mathbf{W}_{sx}.\sigma''(Z_s).\mathbf{V}_s + \sigma'''(Z_s).\mathbf{W}_{sx}^2.\mathbf{V}_s.x\right). \tag{4.14}$$

De forma semelhante,

$$\frac{\frac{\partial^2\hat{N}}{\partial\mathbf{y}^2}}{\partial W_s} = \left(-\left[\sigma''(Z_L).\mathbf{W}_{Ly}^2.\right].\Psi^{-1}.\mathbf{V}_S^T.\Omega'.y_l\right) + \left(2.\mathbf{W}_{sy}.\sigma''(Z_s).\mathbf{V}_s + \sigma'''(Z_s).\mathbf{W}_{sy}^2.\mathbf{V}_s.y\right). \tag{4.15}$$

Essa equação apresenta um terceiro elemento que depende dos pesos, dessa forma também irá ser derivado em relação a qualquer um dos pesos p_s. Com isso, ao derivar $\frac{\partial\alpha_n e^{\hat{N}}}{\partial W_s}$, o resultado obtido é definido por:

$$\frac{\partial\alpha_n e^{\hat{N}}}{\partial W_s} = \alpha_n e^{\hat{N}}\left[\sigma(Z_L)\frac{\partial\mathbf{V}_L^T}{\partial W_s} + \sigma'(Z_S).\mathbf{V}_S^T.\frac{\partial\mathbf{Z}_S^T}{\partial W_s}\right]. \tag{4.16}$$

resultando em

$$\frac{\partial\alpha_n e^{\hat{N}}}{\partial W_s} = \alpha_n e^{\hat{N}}\left[-\sigma(Z_L).\Psi^{-1}.\mathbf{V}_S^T.\Omega'.X + \sigma'(Z_S).\mathbf{V}_S^T.X\right]. \tag{4.17}$$

Substituindo (4.14), (4.15) e (4.17) em (4.7), temos

$$\begin{aligned}\frac{\partial E(p_s)}{\partial\mathbf{W_S}} =& 2.\varepsilon.\Big\{\sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega'.x_l - \left(2.\mathbf{W}_{sx}.\sigma''(Z_s).\mathbf{V}_s + \sigma'''(Z_s).\mathbf{W}_{Sx}^2.\mathbf{V}_s.x\right)\ldots\\ &+\sigma''(Z_L).\mathbf{W}_{Ly}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega'.y_l - \left(2.\mathbf{W}_{sy}.\sigma''(Z_s).\mathbf{V}_s + \sigma'''(Z_s).\mathbf{W}_{Sy}^2.\mathbf{V}_s.y\right)\ldots\\ &-\alpha_n e^{\hat{N}}\left[-\sigma(Z_L).\Psi^{-1}.\mathbf{V}_S^T.\Omega'.X + \sigma'(Z_S).\mathbf{V}_S^T.X\right]\Big\}.\end{aligned} \tag{4.18}$$

Derivando em relação ao bias, O gradiente da função objetivo será

$$\frac{\partial E(p_s)}{\partial \mathbf{b_S}} = 2.\varepsilon.\left\{-\frac{\partial}{\partial b_s}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) - \frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) - \frac{\partial}{\partial b_S}\left(\partial \alpha_n e^{\hat{N}}\right)\right\}. \tag{4.19}$$

de modo que

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial b_s} = \frac{\partial}{\partial b_s}\left[\sigma''(Z_L)\mathbf{W}_{Lx}^2\mathbf{V}_L + \sigma''(Z_S)\mathbf{W}_{Sx}^2\mathbf{V}_S\right]. \tag{4.20}$$

Onde,

$$g_1(b_s) = \frac{\partial}{\partial b_s}\left[\sigma''(Z_L).\mathbf{W}_{Lx}^2.\mathbf{V}_L\right]; \tag{4.21}$$

$$g_2(b_s) = \frac{\partial}{\partial b_s}\left[\sigma''(Z_s).\mathbf{W}_{Sx}^2.\mathbf{V}_s\right]. \tag{4.22}$$

De forma semelhante ao procedimento realizado para determinar $\frac{\partial E(p_s)}{\partial W_s}$, podemos determinar $g_1(b_s)$ e $g_2(b_s)$, sendo $g_1(b_s)$ submetido a condição descrita em (3.36) até (3.39), dessa forma:

$$g_1(b_s) = -\sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega'; \tag{4.23}$$

$$g_2(b_s) = \sigma'''(Z_s).\mathbf{W}_{Sx}^2.\mathbf{V}_s. \tag{4.24}$$

Logo,

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial b_s} = -\sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega' + \sigma'''(Z_s).\mathbf{W}_{Sx}^2.\mathbf{V}_s. \tag{4.25}$$

De forma semelhante,

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}}{\partial b_s} = -\sigma''(Z_L).\mathbf{W}_{Ly}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega' + \sigma'''(Z_s).\mathbf{W}_{Sy}^2.\mathbf{V}_s. \tag{4.26}$$

Para determinar $\frac{\partial \alpha_n e^{\hat{N}}}{\partial b_s}$, será utilizado o mesmo procedimento de (4.16) pois será necessário derivar a saída da rede $\hat{N}$ em função do bias.

$$\frac{\partial \alpha_n e^{\hat{N}}}{\partial b_s} = \alpha_n e^{\hat{N}}\left[\sigma(Z_L)\frac{\partial \mathbf{V}_L^T}{\partial b_s} + \sigma'(Z_S).\mathbf{V}_S^T.\frac{\partial \mathbf{Z}_S^T}{\partial b_s}\right]. \tag{4.27}$$

resultando em

$$\frac{\partial \alpha_n e^{\hat{N}}}{\partial b_s} = \alpha_n e^{\hat{N}}\left[-\sigma(Z_L).\Psi^{-1}.\mathbf{V}_S^T.\Omega' + \sigma'(Z_S).\mathbf{V}_S^T\right]. \tag{4.28}$$

Substituindo (4.25), (4.26) e (4.28) em (4.19), obteremos o gradiente da função objetivo em função do bias, sendo descrita como

$$\begin{aligned}\frac{\partial E(p_s)}{\partial \mathbf{b_S}} =& 2.\varepsilon.\{\ \sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega' - \sigma'''(Z_s).\mathbf{W}_{Sx}^2.\mathbf{V}_s... \\ &+ \sigma''(Z_L).\mathbf{W}_{Ly}^2.\Psi^{-1}.\mathbf{V}_S^T.\Omega' - \sigma'''(Z_s).\mathbf{W}_{Sy}^2.\mathbf{V}_s... \\ &- \alpha_n e^{\hat{N}}\left[-\sigma(Z_L).\Psi^{-1}.\mathbf{V}_S^T.\Omega' + \sigma'(Z_S).\mathbf{V}_S^T\right]\Big\}.\end{aligned} \tag{4.29}$$

Derivando em relação aos pesos da camada de saída, O gradiente da função objetivo será

$$\frac{\partial E(p_s)}{\partial \mathbf{V_S}} = 2.\varepsilon.\left\{-\frac{\partial}{\partial V_s}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) - \frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) - \frac{\partial}{\partial V_S}\left(\partial \alpha_n e^{\hat{N}}\right)\right\}. \tag{4.30}$$

Podendo escrever $\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial V_s}$ como

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial V_s} = \frac{\partial}{\partial V_s}\left[\sigma''(Z_L)\mathbf{W}_{Lx}^2\mathbf{V}_L + \sigma''(Z_S)\mathbf{W}_{Sx}^2\mathbf{V}_S\right]. \tag{4.31}$$

de modo que

$$g_1(V_s) = \frac{\partial}{\partial V_s}\left[\sigma''(Z_L).\mathbf{W}_{Lx}^2.\mathbf{V}_L\right]; \tag{4.32}$$

$$g_2(V_s) = \frac{\partial}{\partial V_s}\left[\sigma''(Z_s).\mathbf{W}_{Sx}^2.\mathbf{V}_s\right]. \tag{4.33}$$

Uma vez que $g_1(V_s)$ deve obedecer a condição (3.30), resultam em

$$g_1(V_s) = -\sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\Omega; \tag{4.34}$$

$$g_2(V_s) = \sigma''(Z_s).\mathbf{W}_{Sx}^2. \tag{4.35}$$

com isso, podemos reescrever $\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial V_s}$ e $\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}}{\partial V_s}$ como

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}}{\partial V_s} = -\sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\Omega + \sigma''(Z_s).\mathbf{W}_{Sx}^2; \tag{4.36}$$

$$\frac{\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}}{\partial V_s} = -\sigma''(Z_L).\mathbf{W}_{Ly}^2.\Psi^{-1}.\Omega + \sigma''(Z_s).\mathbf{W}_{Sy}^2. \tag{4.37}$$

Derivando a saída da rede (3.3) em relação aos pesos da camada de saída, podemos escrever $\frac{\partial \alpha_n e^{\hat{N}}}{\partial V_s}$ como

$$\frac{\partial \alpha_n e^{\hat{N}}}{\partial V_s} = \alpha_n e^{\hat{N}}\left[\sigma(Z_L)\frac{\partial \mathbf{V}_L^T}{\partial V_s} + \sigma(Z_S).\frac{\partial \mathbf{V}_S^T}{\partial V_s}\right]. \tag{4.38}$$

Resultando em

$$\frac{\partial \alpha_n e^{\hat{N}}}{\partial V_s} = \alpha_n e^{\hat{N}}\left[-\sigma(Z_L).\Psi^{-1}.\Omega + \sigma(Z_S)\right]. \tag{4.39}$$

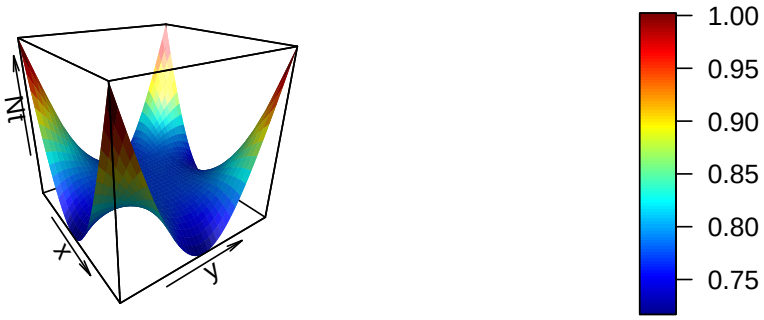
Substituindo (4.36), (4.37) e (4.39) em (4.30), obteremos o gradiente da função objetivo em função do bias, sendo descrita como

$$\begin{aligned}\frac{\partial E(p_s)}{\partial \mathbf{V_S}} =& 2.\varepsilon.\{\ \sigma''(Z_L).\mathbf{W}_{Lx}^2.\Psi^{-1}.\Omega - \sigma''(Z_s).\mathbf{W}_{Sx}^2 ... \\ &+ \sigma''(Z_L).\mathbf{W}_{Ly}^2.\Psi^{-1}.\Omega - \sigma''(Z_s).\mathbf{W}_{Sy}^2 ... \\ &- \alpha_n e^{\hat{N}}\left[-\sigma(Z_L).\Psi^{-1}.\Omega + \sigma(Z_S)\right]\Big\}.\end{aligned} \tag{4.40}$$

As Figuras 12, 13, 14, 15, 16 e 17 mostram respectivamente a solução fornecida pela rede treinada com o método de Levenberg-Marquardt, com $n = 0, \ldots, 5$, onde só existe solução analítica para $n = 5$. Cada execução teve, tolerância de 10^{-4}

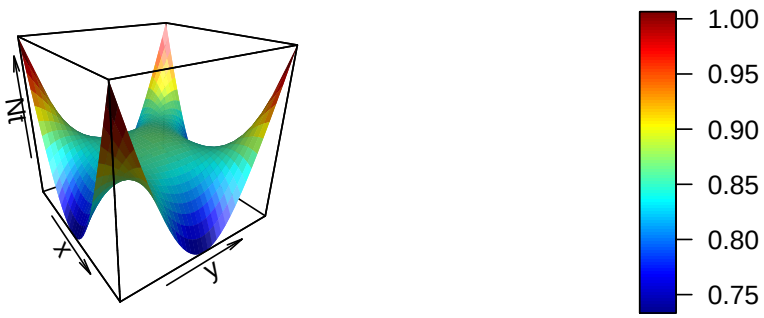
A Figura 12 apresenta o resultado para $n = 0$, com seu processo levando 416 épocas para ser finalizado.

Figura 12 – Problema 1: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com n=0



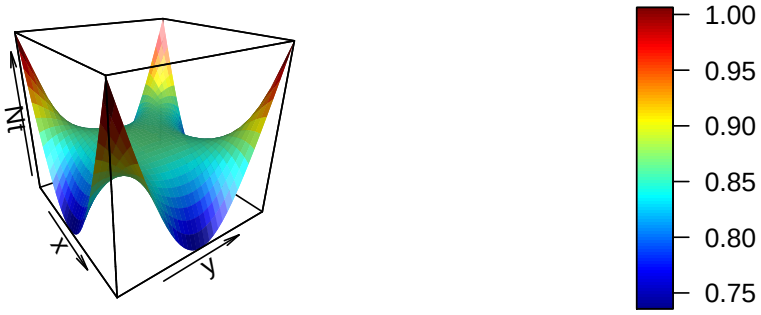
A Figura 13 apresenta o resultado para $n = 1$, com seu processo levando 319 épocas para ser finalizado.

Figura 13 – Problema 1: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com n=1



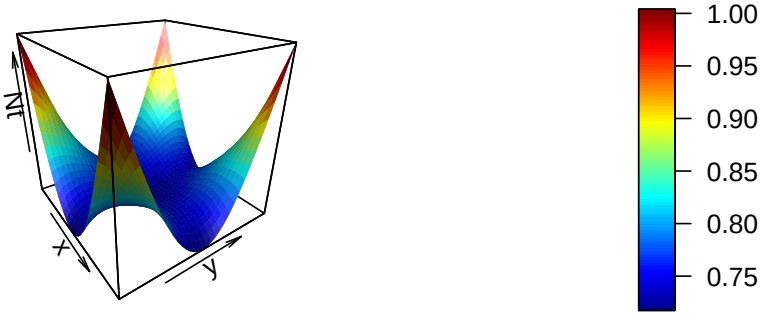
A Figura 14 apresenta o resultado para $n = 2$, com seu processo levando 257 épocas para ser finalizado.

Figura 14 – Problema 1: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com n=2



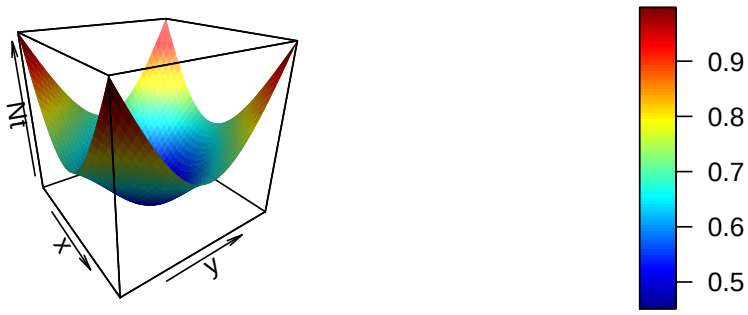
A Figura 15 apresenta o resultado para $n = 3$, com seu processo levando 198 épocas para ser finalizado.

Figura 15 – Problema 1: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com n=3



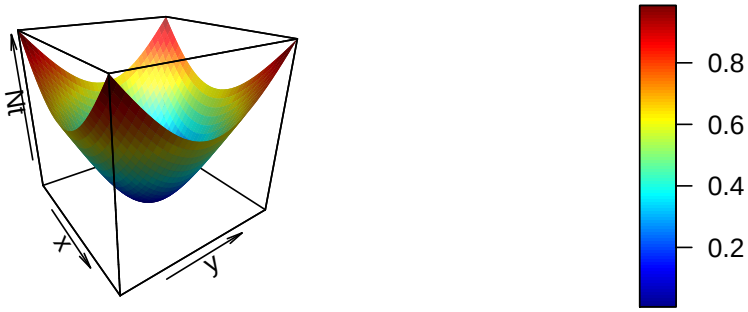
A Figura 16 apresenta o resultado para $n = 4$, com seu processo levando 176 épocas para ser finalizado.

Figura 16 – Problema 1: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com n=4



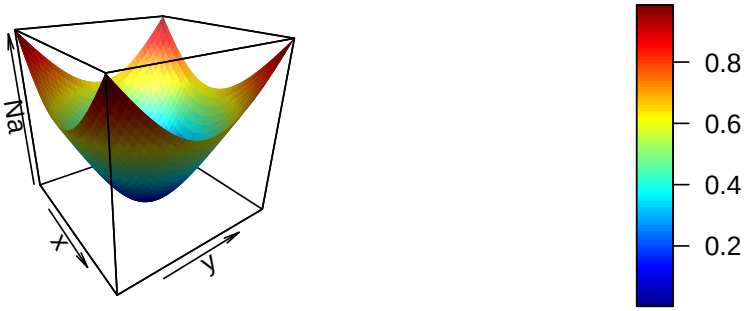
Para $n = 5$, a solução $N_t(\vec{X})$ obtida pelo método, é representada pela Figura 17. Nesta simulação, o número de épocas necessárias foi de 128, sendo a única que apresenta solução analítica permitindo realizar uma comparação com a mesma.

Figura 17 – Problema 1: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com n=5



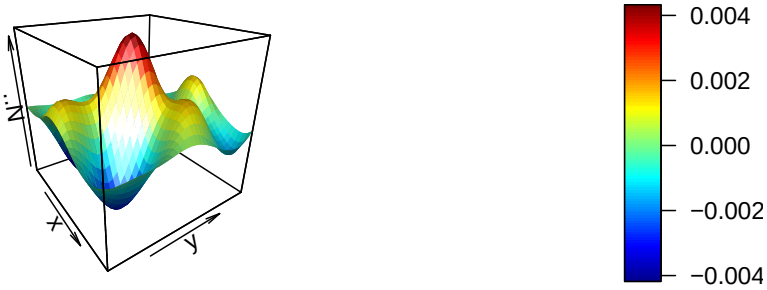
A Figura 18 contêm a solução analítica de 4.1 para $n = 5$, sendo definida por $\mathbf{N_a}(\vec{x}) = log(x^2 + y^2 + 1)$

Figura 18 – Problema 1: Solução Analítica da EDP para $n = 5$



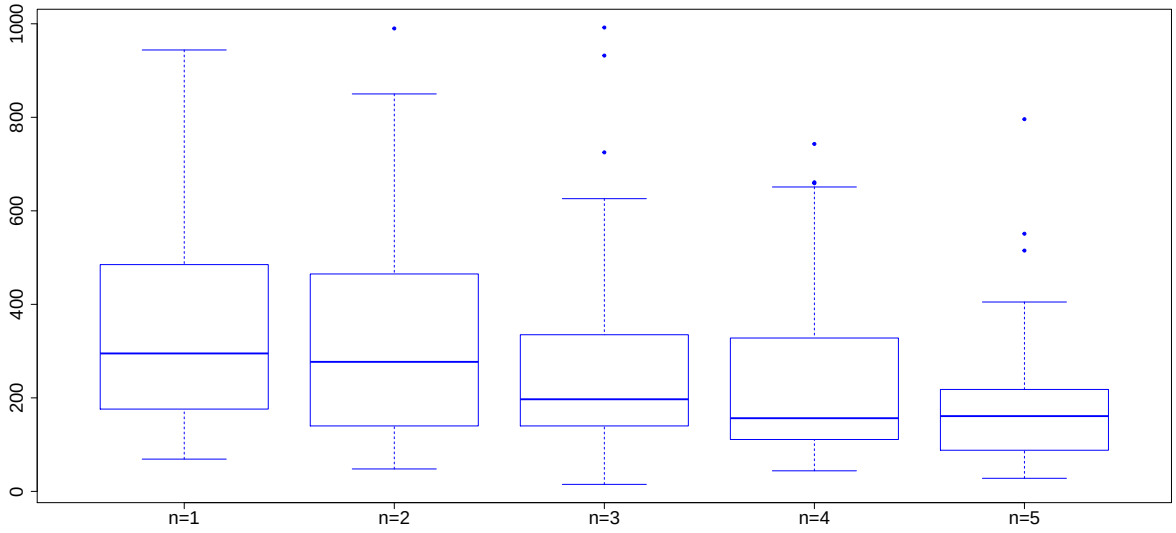
A Figura 19 apresenta o desvio obtido na comparação entre a solução analítica e a obtida pelo CPROP, apresentando uma solução muito boa.

Figura 19 – Problema 1: Desvio obtido pelo método Levenberg-Marquadt



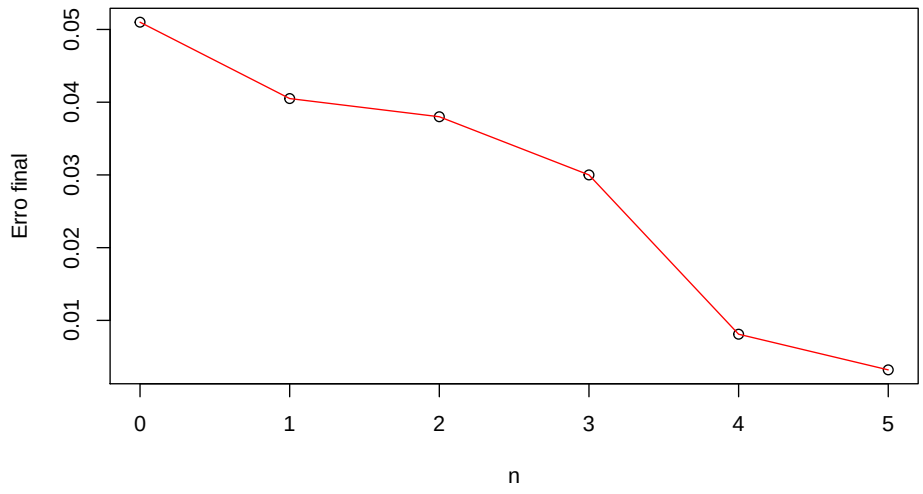
Além disso, a Figura 20 apresenta o número de épocas necessárias para obter as soluções fornecidas pelo método, com 50 simulações para cada $n = 1 \ldots 5$, com pesos iniciais aleatórios em todos os casos. As simulações monstram que o o número de épocas decai conforme o valor de n varia a cada nova EDP.

Figura 20 – Problema 1: Box Plot de épocas de treinamento necessárias para obter a solução do problema 1



A Figura 21 indica uma redução do erro ao fim de cada processo. Isso mostra que a precisão do CPROP melhora de uma EDP para a próxima, isso ocorre por que o método se beneficia do conhecimento obtido na solução anterior.

Figura 21 – Problema 1: Erro final de cada EDP do problema 1



4.3 CPROP APLICADO NA SOLUÇÃO DE UMA EQUAÇÃO PARABÓLICA

4.3.1 problema 2

$$\frac{\partial \hat{N}}{\partial \mathbf{t}} = K_n . \left\{ \frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} + \frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} \right\}. \tag{4.41}$$

com condição de contorno definida por $\tilde{\mathbf{h}}(\vec{x}) = 0$ com seu dominio definido de $[-1,1] \times [-1,1]$ onde $K_0 = 0.1$ e $K_1 = 0.01$ com $n = 0,1$. Com o *kn* mantido constante nesses dois casos, eles representam a difusividade do material além de determinar qual calor ou massa é difundido pelo sistema em questão.

A condição inicial é definida como $u(x,y,0) = e^{-7(x^2+y^2)}.sen(2.\pi.x)$ e a função q é descrita como $q = (x^2-1).(y^2-1)$

A região interna consiste em uma malha de $15 \times 15 \times 15$ e a sua fronteira esta distribuida em 30×30 pontos equidistantes. A rede foi particionada como descrito na subseção 3.2, possuindo 30 neurônios pertencentes a STM e 50 neurônios pertencentes a LTM de modo que os pesos W_L e b_L foram inicializados com uma distribuição uniforme no intervalo $(-5,5)$

Como foi definido em (3.8) e (3.10), a função objetivo a ser minimizada será definida por

$$E(\mathbf{p_S}) = \frac{1}{2} . \sum_i \left\{ \frac{\partial \hat{N}}{\partial \mathbf{t}} - K_n . \left\{ \frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} + \frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} \right\} \right\}^2 . \tag{4.42}$$

de modo que $\mathbf{p_s}$ representa qualquer um dos pesos. Onde todas as EDPs do tipo elípticas podem ser expressas pela derivada parcial em (3.14) e (3.15). A aplicação dessa derivada transforma a equação (4.42) gera os seguintes resultados:

$$\frac{\partial \hat{N}}{\partial \mathbf{t}} = (x^2-1).(y^2-1). \left[\sigma'(Z_L).W_{Lt}.\mathbf{V}_L^T + \sigma'(Z_S).W_{St}.\mathbf{V}_S^T \right]. \tag{4.43}$$

Resultante da derivada $\frac{\partial \hat{N}}{\partial \mathbf{t}}$, sendo que os termos W_{Lt} e W_{St} representam as derivadas dos pesos da camada de entrada respectivos a $\mathbf{t}$.

$$\begin{aligned} \frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} =& 2.(y^2-1). \left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T \right] ... \\ &+ 2.x.(y^2-1). \left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T \right] ... \\ &+ 2.x.(y^2-1). \left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T \right] ... \\ &+ (x^2-1).(y^2-1). \left[\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T \right]. \end{aligned} \tag{4.44}$$

Que surge a partir de $\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}$, onde W_{Lx} e W_{Sx} representam as derivadas dos pesos da camada de entrada respectivos a $\mathbf{x}$. De forma semelhante, ao derivar $\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}$,

$$\begin{aligned}\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2} =& 2.(x^2-1).\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right] ... \\ &+ 2.y.(x^2-1).\left[\sigma'(Z_L).W_{Ly}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sy}.\mathbf{V}_S^T\right] ... \\ &+ 2.y.(x^2-1).\left[\sigma'(Z_L).W_{Ly}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sy}.\mathbf{V}_S^T\right] ... \\ &+ (x^2-1).(y^2-1).\left[\sigma''(Z_L).W_{Ly}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sy}.^2.\mathbf{V}_S^T\right].\end{aligned} \tag{4.45}$$

Sendo que $\mathbf{V}_S^T$ e $\mathbf{V}_L^T$ representam os pesos da camada de saída.

Nessa EDP, a função $\tilde{\mathbf{h}}(\vec{x}) = 0$ apresenta derivadas expressas por.

- $\frac{\partial \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}} = 0$;
- $\frac{\partial^2 \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2} = 0$.

Com isso, derivando (4.42) em função dos pesos trabalhados, obtém-se:

$$\frac{\partial E(p_s)}{\partial \mathbf{p_S}} = 2.\varepsilon.\left\{\frac{\partial}{\partial p_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) - K_n.\left\{\frac{\partial}{\partial p_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) + \frac{\partial}{\partial p_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right)\right\}\right\}. \tag{4.46}$$

Sendo $\varepsilon = \left\{\frac{\partial \hat{N}}{\partial \mathbf{t}} - K_n.\left\{\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2} + \frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right\}\right\}$

Em relação ao peso W_S, que pertence ao conjunto *STM* , (4.46) pode ser representada como

$$\frac{\partial E(p_s)}{\partial \mathbf{W_S}} = 2.\varepsilon.\left\{\frac{\partial}{\partial W_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) - K_n.\left\{\frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) + \frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right)\right\}\right\}. \tag{4.47}$$

Em relação a (4.43), é possivel obter:

$$\frac{\partial}{\partial W_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) = (x^2-1).(y^2-1).\left[\left(\sigma''(Z_S).W_{St}.t + \sigma'(Z_S)\right)\mathbf{V}_S^T\right]. \tag{4.48}$$

Uma vez que W_L é constante, dessa forma, $\sigma(Z_L).\Psi^{-1}$ pode ser computado e processado antes do treinamento.

As derivadas em relação a (4.44), pode ser descrita por:

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) =& \frac{\partial}{\partial W_S}\left\{2.(y^2-1).\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right] ... \right. \\ &+ 2.x.(y^2-1).\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right] ... \\ &+ 2.x.(y^2-1).\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right] ... \\ &\left. + (x^2-1).(y^2-1).\left[\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T\right]\right\}.\end{aligned} \tag{4.49}$$

Onde

$$f_1(W_S) = \frac{\partial}{\partial W_S}\left\{\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right]\right\}; \tag{4.50}$$

$$f_2(W_S) = \frac{\partial}{\partial W_S}\left\{\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right\}; \tag{4.51}$$

$$f_3(W_S) = \frac{\partial}{\partial W_S}\left\{\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T\right\}. \tag{4.52}$$

Para obter a solução de (4.50), (4.51) e (4.52), foram utilizados os passos que se encontram presentes em (3.51), (3.52) e (3.62), respectivamente. Resultando em:

$$f_1(W_S) = \left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \sigma'(Z_S).x.\mathbf{V}_S^T\right]; \tag{4.53}$$

$$f_2(W_S) = \left[\sigma'(Z_L).W_{LX}.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \left[\sigma''(Z_S).W_{SX}.x + \sigma'(Z_S).x\right]\mathbf{V}_S^T\right]; \tag{4.54}$$

$$f_3(W_S) = \left[\sigma''(Z_L).W_{LX}^2.\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \left[\sigma'''(Z_S).W_{SX}^2.x + 2.\sigma''(Z_S).W_{SX}\right].\mathbf{V}_S^T\right]. \tag{4.55}$$

Dessa forma, a equação (4.49) pode ser reescrita como:

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) =& 2.(y^2-1).f_1(W_S)... \\ &+ 2.x.(y^2-1).f_2(W_S)... \\ &+ 2.x.(y^2-1).f_2(W_S)... \\ &+ (x^2-1).(y^2-1).f_3(W_S).\end{aligned} \tag{4.56}$$

A derivada de W_S em relação a Y é definida analogamente como

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) =& 2.(x^2-1).f_1(W_S)... \\ &+ 2.y.(x^2-1).f_2(W_S)... \\ &+ 2.y.(x^2-1).f_2(W_S)... \\ &+ (x^2-1).(y^2-1).f_3(W_S).\end{aligned} \tag{4.57}$$

Sendo (4.53), (4.54) e (4.55) definidos como

$$f_1(W_S) = \left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S} + \sigma'(Z_S).y.\mathbf{V}_S^T\right]; \tag{4.58}$$

$$f_2(W_S) = \left[\sigma'(Z_L).W_{LY}.\frac{\partial \mathbf{V}_L^T}{\partial W_{SY}} + \left[\sigma''(Z_S).W_{SY}.y + \sigma'(Z_S).y\right]\mathbf{V}_S^T\right]; \tag{4.59}$$

$$f_3(W_S) = \left[\sigma''(Z_L).W_{LY}^2.\frac{\partial \mathbf{V}_L^T}{\partial W_{SY}} + \left[\sigma'''(Z_S).W_{SY}^2.x_i + 2.\sigma''(Z_S).W_{SY}\right].\mathbf{V}_S^T\right]. \tag{4.60}$$

O processo é semelhante ao derivar (4.46) em relação a b_S, assumindo a forma

$$\frac{\partial E(p_s)}{\partial \mathbf{b_S}} = 2.\varepsilon.\left\{\frac{\partial}{\partial b_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) - K_n.\left\{\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) + \frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right)\right\}\right\}. \tag{4.61}$$

De modo que a derivada $\frac{\partial}{\partial b_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right)$ pode ser definida como

$$\frac{\partial}{\partial b_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) = q.\left[\sigma'(Z_L).W_{Lt}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma''(Z_S).W_{St}.\mathbf{V}_S^T\right]. \tag{4.62}$$

Dando sequencia ao método, a derivada (4.44), ao ser derivada em relação a b_S, utiliza o processo apresentado para obter (3.65).

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) =& \frac{\partial}{\partial b_S}\left\{2.(y^2-1).\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right] ...\right.\\ &+ 2.x.(y^2-1).\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right] ...\\ &+ 2.x.(y^2-1).\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right] ...\\ &\left.+ (x^2-1).(y^2-1).\left[\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T\right]\right\}.\end{aligned} \tag{4.63}$$

Semelhante ao processo realizado de 4.49 a 4.56, as derivadas $f_1(b_S)$, $f_2(b_S)$ e $f_3(b_S)$ são definidas por

$$f_1(b_S) = \frac{\partial}{\partial b_S}\left\{\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right]\right\}; \tag{4.64}$$

$$f_2(b_S) = \frac{\partial}{\partial b_S}\left\{\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right]\right\}; \tag{4.65}$$

$$f_3(b_S) = \frac{\partial}{\partial b_S}\left\{\left[\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T\right].\right\} \tag{4.66}$$

O processo de derivação presente em (4.64), (4.65) e (4.66) é descrito em (3.54), (3.55) e (3.64), resultando em

$$f_1(b_S) = \sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma'(Z_S).\mathbf{V}_S^T; \tag{4.67}$$

$$f_2(b_S) = \sigma'(Z_L).W_{Lx}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma''(Z_S).W_{Sx}.\mathbf{V}_S^T; \tag{4.68}$$

$$f_3(b_S) = \sigma''(Z_L).W_{Lx}^2.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma'''(Z_S).W_{Sx}^2.\mathbf{V}_S^T. \tag{4.69}$$

Dessa forma, a equação (4.63) é descrita como:

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) =& 2.(y^2-1).f_1(b_S)... \\ &+ 2.x.(y^2-1).f_2(b_S)... \\ &+ 2.x.(y^2-1).f_2(b_S)... \\ &+ (x^2-1).(y^2-1).f_3(b_S).\end{aligned} \tag{4.70}$$

Apresentando uma solução semelhante em $\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right)$

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) =& 2.(x^2-1).f_1(b_S)... \\ &+ 2.y.(x^2-1).f_2(b_S)... \\ &+ 2.y.(x^2-1).f_2(b_S)... \\ &+ (x^2-1).(y^2-1).f_3(b_S).\end{aligned} \tag{4.71}$$

De modo que

$$f_1(b_S) = \sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma'(Z_S).\mathbf{V}_S^T; \tag{4.72}$$

$$f_2(b_S) = \sigma'(Z_L).W_{Ly}.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma''(Z_S).W_{Sy}.\mathbf{V}_S^T; \tag{4.73}$$

$$f_3(b_S) = \sigma''(Z_L).W_{Ly}^2.\frac{\partial \mathbf{V}_L^T}{\partial b_S} + \sigma'''(Z_S).W_{Sy}^2.\mathbf{V}_S^T. \tag{4.74}$$

A derivada de (4.46) em relação aos pesos de saída é descrito por:

$$\frac{\partial E(p_s)}{\partial \mathbf{V_S}} = 2.\varepsilon.\left\{\frac{\partial}{\partial V_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) - K_n.\left\{\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) + \frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right)\right\}\right\}. \tag{4.75}$$

Sendo $\frac{\partial}{\partial V_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right)$ a sua respectiva derivada em relação ao tempo. O resultado da mesma é expresso através de

$$\frac{\partial}{\partial V_S}\left(\frac{\partial \hat{N}}{\partial \mathbf{t}}\right) = q.\left[\sigma'(Z_L).W_{Lt}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).W_{St}\right]. \tag{4.76}$$

As derivadas $\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right)$ e $\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right)$ são obtidas de forma semelhante. A derivada de X, para facilitar a visualização, passa a ser escrita como

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) =& \frac{\partial}{\partial V_S}\left\{2.(y^2-1).\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right] ...\right. \\ &+ 2.x.(y^2-1).\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right] ... \\ &+ 2.x.(y^2-1).\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right] ... \\ &+ (x^2-1).(y^2-1).\left.\left[\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T\right]\right\}.\end{aligned} \tag{4.77}$$

Reescrevendo novamente, obtemos

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}^2}\right) =& \frac{\partial}{\partial V_S}\left\{2.(y^2-1).f_1(V_S) ...\right. \\ &+ 2.x.(y^2-1).f_2(V_S)... \\ &+ 2.x.(y^2-1).f_2(V_S)... \\ &+ (x^2-1).(y^2-1).\left.f_3(V_S)\right\}.\end{aligned} \tag{4.78}$$

onde

$$f_1(V_S) = \frac{\partial}{\partial V_S}\left\{\left[\sigma(Z_L.)\mathbf{V}_L^T + \sigma(Z_S).\mathbf{V}_S^T\right]\right\}; \tag{4.79}$$

$$f_2(V_S) = \frac{\partial}{\partial V_S}\left\{\left[\sigma'(Z_L).W_{Lx}.\mathbf{V}_L^T + \sigma'(Z_S).W_{Sx}.\mathbf{V}_S^T\right]\right\}; \tag{4.80}$$

$$f_3(V_S) = \frac{\partial}{\partial V_S}\left\{\left[\sigma''(Z_L).W_{Lx}^2.\mathbf{V}_L^T + \sigma''(Z_S).W_{Sx}.^2.\mathbf{V}_S^T\right]\right\}. \tag{4.81}$$

Os resultados de (4.79), (4.80) e (4.81) são expressos em (3.57), (3.58) e (3.66), respectivamente. Com isso

$$f_1(V_S) = \sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma(Z_S); \tag{4.82}$$

$$f_2(V_S) = \sigma'(Z_L).W_{Lx}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).W_{Sx}; \tag{4.83}$$

$$f_3(V_S) = \sigma''(Z_L).W_{Lx}^2.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma''(Z_S).W_{Sx}^2. \tag{4.84}$$

Para Y, a estrutura passa a ser

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{y}^2}\right) =& \frac{\partial}{\partial V_S}\left\{2.(x^2-1).f_1(V_S)\ldots\right. \\ &+2.y.(x^2-1).f_2(V_S)\ldots \\ &+2.y.(x^2-1).f_2(V_S)\ldots \\ &\left.+\,(x^2-1).(y^2-1).f_3(V_S)\right\}.\end{aligned} \tag{4.85}$$

Obtendo (4.82), (4.83) e (4.84), que são descritos como

$$f_1(V_S) = \sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma(Z_S); \tag{4.86}$$

$$f_2(V_S) = \sigma'(Z_L).W_{Ly}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).W_{Sy}; \tag{4.87}$$

$$f_3(V_S) = \sigma''(Z_L).W_{Ly}^2.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma''(Z_S).W_{Sy}^2. \tag{4.88}$$

As Figuras 22, 23,24 ,25 e 26 apresentam soluções obtidas pelo CPROP otimizado pelo método Levenberg-Marquardt, com $Kn = 0.01$, nos instantes $t = 0$, 0.2, 0.3, 0.6 e 1. Para cada simulação foram obtidas 15 imagens do instante $t = 0$ ao instante $t = 1$ com tolerância de 10^{-3}

A Figura 22 apresenta o resultado no instante $t = 0$. A Figura 23 apresenta o resultado no instante $t = 0.2$.

Figura 22 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0 k=0.01

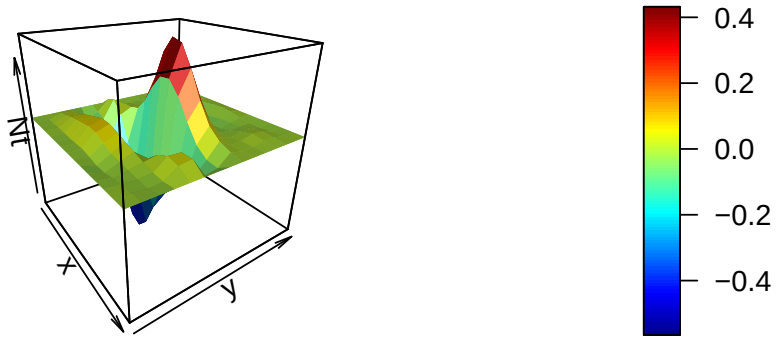
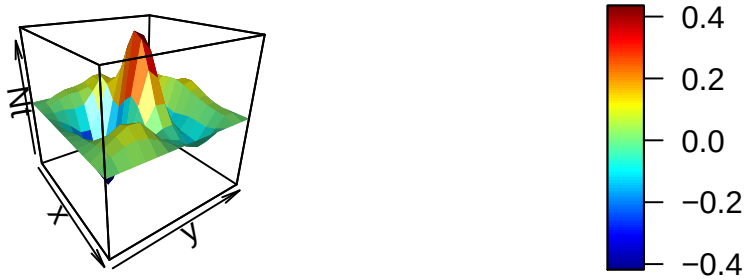


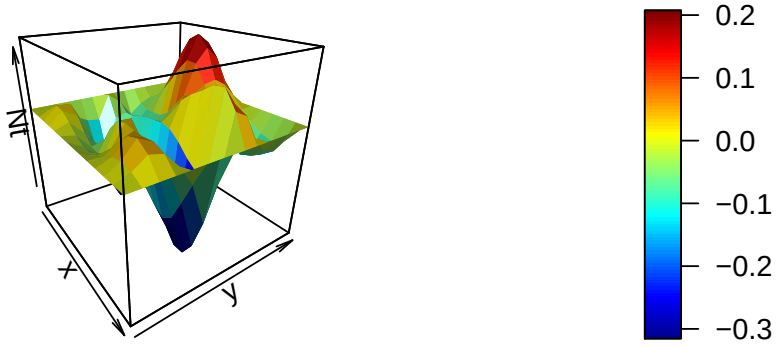
Figura 23 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0.2 k=0.01



A Figura 24 apresenta o resultado no instante $t = 0.3$. A Figura 25 apresenta o resultado no instante $t = 0.6$.

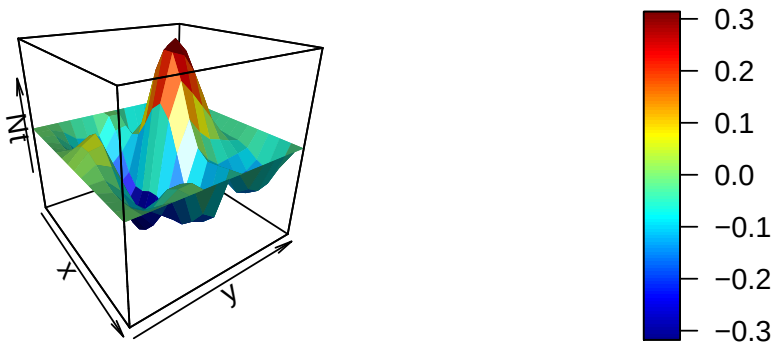
Figura 24 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0.3 k=0.01

Figura 25 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0.6 k=0.01



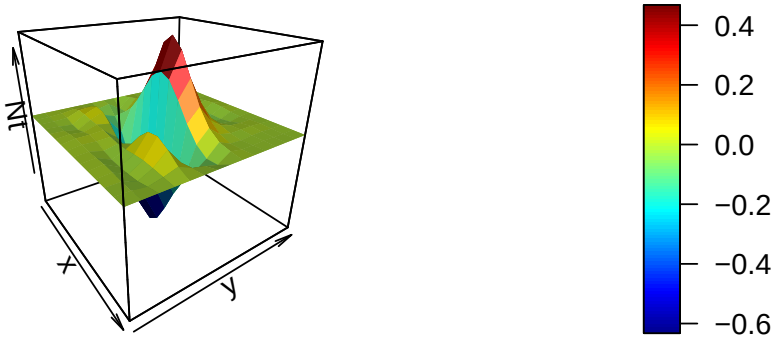
A Figura 26 apresenta o resultado no instante $t = 1$ finalizando a simulação para $k = 0.01$. Para obter todas as imagens dessa edp na mesma simulação foram utilizado 70 épocas.

Figura 26 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=1 com k=0.01



As Figuras 27, 28,29 e 30 apresentam soluções obtidas pelo CPROP otimizado pelo método Levenberg-Marquardt, com $Kn = 0.1$, nos instantes $t = 0$, 0.1, 0.2 e 0.3. A Figura 27 apresenta o resultado no instante $t = 0$.

Figura 27 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0 com k=0.1



As Figuras 28 e 29 apresentam o resultado no instante $t = 0.1$ e $t = 0.2$, respectivamente.

Figura 28 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0.1 com k=0.1

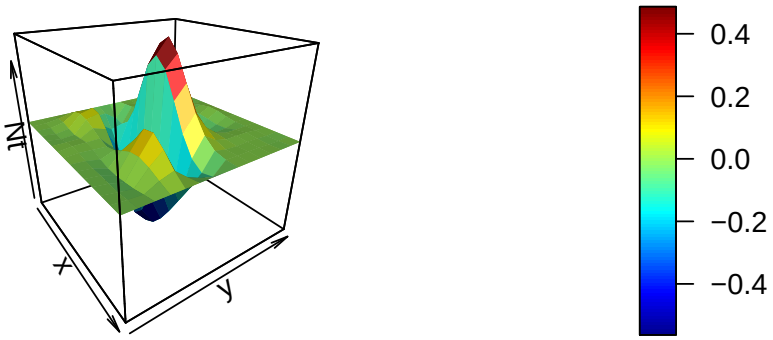
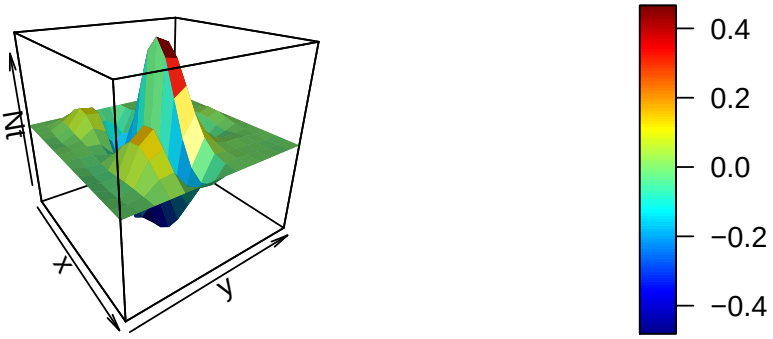
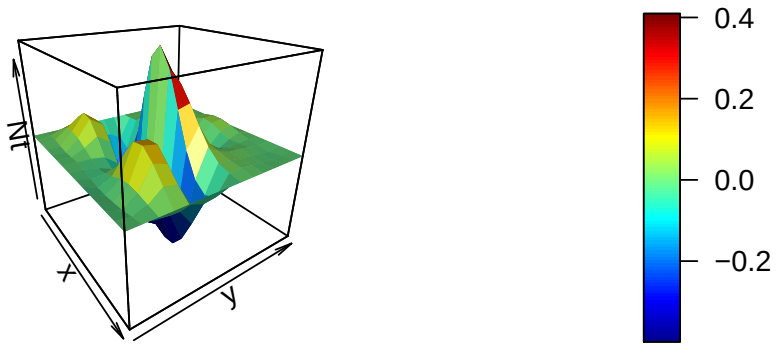


Figura 29 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0.2 com k=0.1



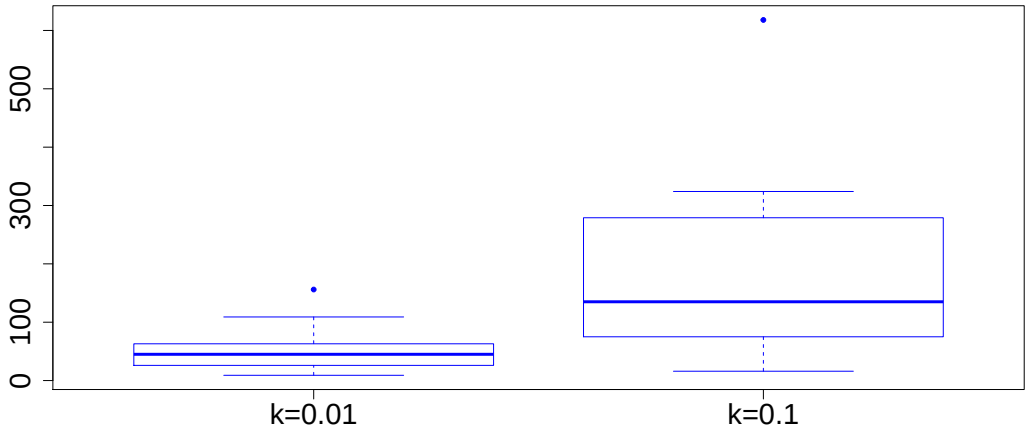
A Figura 30 apresenta o resultado no instante $t = 0.3$. Para obter as imagens com $k = 0.1$ foi nescessário um número maior de épocas, com 170 épocas nessa simulação.

Figura 30 – Problema 2: Solução fornecida pela rede treinada através do método Levenberg-Marquardt com t=0.3 com k=0.1



Semelhante ao que foi apresentado em 4.2, a Figura 31 apresenta o número de épocas necessárias para obter a solução fornecida pelo método, com 25 simulações para $k = 0.01$ e $k = 0.1$, com pesos iniciais aleatórios em todos os casos.

Figura 31 – Problema 2: Box Plot de épocas de treinamento necessárias para obter a solução do problema 2



Apesar do grande número de dados utilizados para alimentar a rede, os resultados de $k = 0.01$ foram obtidos rapidamente e com maior facilidade, enquanto os resultados de $k = 0.1$ foram mais difíceis de se obter, mesmo com ambos os resultados sendo obtidos uma tolerância de 10^{-3}.

Infelizmente não é possivel realizar a comparação dos resultados obtidos pela rede, uma vez que o problema apresentado em 4.3 não apresenta solução analítica.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi realizado uma apresentação do método *Constrained Backpropagation* (CPROP), que soluciona equações diferenciais parciais através de redes neurais artificiais, que é um excelente aproximador de função. Além do método, foi empregado o método de Levenberg-Marquardt para a otimização da mesma.

O CPROP apresentou um bom desempenho ao trabalhar com equações do tipo elíptica e parabólica, satisfazendo as condições de contorno/iniciais através de restrições ao processo de otimização dos pesos da rede neural, apresentando uma boa aproximação da solução.

A malha de treinamento utilizada no Problema 1, referente a equação elíptica, apresentou 1225 pontos e sua fronteira com 180, distribuidos de forma equidistante em ambos os casos. Manteve um excelente desempenho ao longo das simulações, com baixo custo computacional quando comparado ao método empregado por Lagaris, Likas e Fotiadis (1998), que foi apresentado com outras técnicas de otimização por Batista (2012).

O problema 2, referente a equação parabólica, utilizou 225 pontos, para 15 variações de tempo no intervalo de 0 a 1, com sua fronteira tendo 900 pontos, distribuidos de forma equidistante, tanto no caso $k = 0.01$ quanto $k = 0.1$. O processo apresentou um custo compitacional maior, quando comparado a solução do Problema 1, mas considerando o número de entradas utilizadas, o seu custo é justificado.

O CPROP também apresentou um bom desempenho ao trabalhar com equações parabólicas, satisfazendo as condições iniciais e de contorno utilizando a respectiva restrição proposta para esse caso, apresentando uma boa solução por aproximação.

Apesar da grande quantidade de neurônios presentes na rede, o CPROP apresentou um baixo número de épocas para realizar o ajuste dos pesos. O método apresentou uma melhoria significativa ao implementar método de otimização adequado. Para tal, foi aplicado o método de Levenberg-Marquadt, que acelerou consideravelmente o processo de otimização, apresentando . É recomendado que em trabalhos futuros, sejam implementados outras técnicas de otimização, uma vez que podem gerar resultados com tempo de processamento menor e precisão maior.

Uma observação deve ser feita quando ao número de pontos presentes na malha e a taxa de aprendizagem, em quase todas as situações envolvendo a equação elíptica e parabólica, os pontos da malha foram constantes juntamente com a taxa de aprendizagem. Somente com a elíptica tendo $n = 5$, por ter uma solução analítica, foi realizado testes com mais pontos e uma variação da taxa de aprendizagem, onde o mesmo manteve uma ótima aproximação e baixo custo.

Como sugestão, outros testes devem ser feitos com as demais equações, para verificar o desempenho da rede em situações diversas e avaliar o seu comportamento. Além disso, também é possível obter resultados melhores alterando o número de neurônios utilizados em cada rede.

Apêndices

APÊNDICE A – DERIVADAS DOS PESOS PARA EQUAÇÃO ELÍPTICA

Derivada do erro em relação a W_S

$$\frac{\partial}{\partial W_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \left[-\left[\sigma^k(Z_L).\Lambda_L.\right].\Psi^{-1}.x_l.\Omega'.\mathbf{V}_S^T + \left[a_{i,j}.\sigma^k(Z_S) + \sigma^{k+1}(Z_S).\Lambda_S.x\right].\mathbf{V}_s\right]. \tag{A.1}$$

Derivada do erro em relação a b_S

$$\frac{\partial}{\partial b_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \left[-\sigma^k(Z_L).\Lambda_L.\Psi^{-1}.\Omega'.\mathbf{V}_S^T + \sigma^{k+1}(Z_S).\Lambda_S.\mathbf{V}_s\right]. \tag{A.2}$$

Derivada do erro em relação a V_S

$$\frac{\partial}{\partial V_S}\left(\frac{\partial^k \hat{N}}{\partial \tilde{\mathbf{x}}^k}\right) = \left[-\sigma^k(Z_L).\Lambda_L.\Psi^{-1}.\Omega + \sigma^k(Z_S).\Lambda_S\right]. \tag{A.3}$$

APÊNDICE B – DERIVADAS DOS PESOS PARA EQUAÇÃO PARABÓLICA

Derivada do erro em relação a W_S para primeira derivada

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) =& \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S}+\sigma'(Z_S).x_i.\mathbf{V}_S^T\right]... \\ &+q.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S}+\left[\sigma''(Z_S).\omega_{S_i}.x_i+\sigma'(Z_S).x_i\right]\mathbf{V}_S^T\right]\end{aligned} \tag{B.1}$$

Derivada do erro em relação a b_S para primeira derivada

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) =& \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S}+\sigma'(Z_S).\mathbf{V}_S^T\right]... \\ &+q.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S}+\sigma''(Z_S).\omega_{S_i}.\mathbf{V}_S^T\right]\end{aligned} \tag{B.2}$$

Derivada do erro em relação a V_S para primeira derivada

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial \hat{N}}{\partial \tilde{\mathbf{x}}}\right) =& \frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S}+\sigma(Z_S)\right]... \\ &+q.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S}+\sigma'(Z_S).\omega_{S_i}\right]\end{aligned} \tag{B.3}$$

Derivada do erro em relação a W_S para segunda derivada

$$\begin{aligned}\frac{\partial}{\partial W_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}\right) =& \frac{\partial}{\partial W_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}\right)+\frac{\partial^2 q}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial W_S}+\sigma'(Z_S).x_i.\mathbf{V}_S^T\right]... \\ &+\frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S}+\left[\sigma''(Z_S).\omega_{S_i}.x_i+\sigma'(Z_S).x_i\right]\mathbf{V}_S^T\right]... \\ &+\frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial W_S}+\left[\sigma''(Z_S).\omega_{S_i}.x_i+\sigma'(Z_S).x_i\right]\mathbf{V}_S^T\right]... \\ &+q.\left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial W_S}+\left[\sigma'''(Z_S).\omega_{S_j}^2.x_i+2.\sigma''(Z_S).\omega_{S_j}\right].\mathbf{V}_S^T\right]\end{aligned} \tag{B.4}$$

Derivada do erro em relação a b_S para segunda derivada

$$\begin{aligned}\frac{\partial}{\partial b_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}\right) =& \frac{\partial}{\partial b_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}\right)+\frac{\partial^2 q}{\partial \mathbf{x}_{(\mathbf{i})}{}^2}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial b_S}+\sigma'(Z_S).\mathbf{V}_S^T\right]... \\ &+\frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S}+\sigma''(Z_S).\omega_{S_i}.\mathbf{V}_S^T\right]... \\ &+\frac{\partial q}{\partial \mathbf{x}_{(\mathbf{i})}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial b_S}+\sigma''(Z_S).\omega_{S_i}.\mathbf{V}_S^T\right]... \\ &+q.\left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial b_S}+\sigma'''(Z_S).\omega_{S_j}^2.\mathbf{V}_S^T\right].\end{aligned} \tag{B.5}$$

Derivada do erro em relação a V_S para segunda derivada

$$\begin{aligned}\frac{\partial}{\partial V_S}\left(\frac{\partial^2 \hat{N}}{\partial \mathbf{x_{(i)}}^2}\right) =& \frac{\partial}{\partial V_S}\left(\frac{\partial^2 \tilde{h}}{\partial \mathbf{x_{(i)}}^2}\right) + \frac{\partial^2 q}{\partial \mathbf{x_{(i)}}^2}.\left[\sigma(Z_L).\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma(Z_S)\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).\omega_{S_i}\right]... \\ &+ \frac{\partial q}{\partial \mathbf{x_{(i)}}}.\left[\sigma'(Z_L).\omega_{L_i}.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma'(Z_S).\omega_{S_i}\right]... \\ &+ q.\left[\sigma''(Z_L).\omega_{L_j}^2.\frac{\partial \mathbf{V}_L^T}{\partial V_S} + \sigma''(Z_S).\omega_{S_j}^2\right].\end{aligned} \tag{B.6}$$

Anexos

ANEXO A – LEVENBERG-MARQUADT

Neste trabalho, foi utilizado o método de Levenberg-Marquadt (LM) como técnica de otimização, uma vez que a utilização do gradiente descendente converge muito lentamente. Este método se baseia no método dos mínimos quadrados para modelos não-lineares, sendo ele uma aproximação do método de Newton (Gavin, 2019).

A minimização do erro se da pelo seguinte método interativo:

$$\Delta W_S = \left(J^T.J + \eta.I\right)^{-1}.J^T.E. \tag{A.1}$$

De modo que I representa uma matriz identidade, η é a uma taxa de aprendizagem do algorítmo e J representa a matriz Jacobiana. Sendo definida como:

$$J = \begin{bmatrix} \frac{\partial E_1(Z)}{\partial Z_1} & \frac{\partial E_1(Z)}{\partial Z_2} & \cdots & \frac{\partial E_1(Z)}{\partial Z_q} \\ \frac{\partial E_2(Z)}{\partial Z_1} & \frac{\partial E_2(Z)}{\partial Z_2} & \cdots & \frac{\partial E_2(Z)}{\partial Z_q} \\ \vdots & \vdots & . & \vdots \\ \frac{\partial E_N(Z)}{\partial Z_1} & \frac{\partial E_N(Z)}{\partial Z_2} & \cdots & \frac{\partial E_N(Z)}{\partial Z_q} \end{bmatrix}. \tag{A.2}$$

Sendo Z um vetor paramétrico.

Com isso, a principal característica desse método se torna a obtenção da Matriz Jacobiana. O processo de obtenção do algoritmo CPROP LM é feito através da derivada do jacobiano, onde:

$$J_{(m,n)}\left(p_S\right) = \frac{\partial \varepsilon_{(m)}\left[C(p_S), p_S\right]}{\partial p_{S_{(n)}}} = \frac{\partial \varepsilon_{(m)}\left[p_L, p_S\right]}{\partial p_{S_{(n)}}} + \frac{\partial \varepsilon_{(m)}\left[C(p_S), p_S\right]}{\partial C}.\frac{\partial C}{\partial p_{S_{(n)}}}. \tag{A.3}$$

Para realizar a atualização dos pesos pertencentes a STM.

ANEXO B – PSEUDO-INVERSA

Neste trabalho, para realizar a atualização dos pesos, se fez necessário determinar a matriz Ψ^{-1}, que pode ser uma matriz do tipo não-quadrada. Para contornar esse problema, foi estabelecida uma relação algebrica, denominada como pseudo-inversa.

Em relação ao sistema $U = \Psi.V$, a matriz pseudo-inversa pode ser definida como:

$$\Psi^{-1} = \left(\Psi^T.\Psi\right)^{-1}.\Psi^T. \tag{B.1}$$

De modo que $V = \Psi^{-1}.U$ tem uma solução exata (Ferrari, 2002). Além disso, para uma matriz $\Psi \in \mathbb{R}^{nxm}$ e $\Psi^{-1} \in \mathbb{R}^{nxm}$, Ψ^{-1} será chamada de pseudo-inversa se as seguintes condições forem satisfeitas (Gulob; Van Loan, 2013):

- $\Psi^{-1}\Psi\Psi^{-1} = \Psi$;
- $\Psi\Psi^{-1}\Psi = \Psi^{-1}$;
- $\left(\Psi\Psi^{-1}\right)^T = \Psi\Psi^{-1}$;
- $\left(\Psi^{-1}\Psi\right)^T = \Psi^{-1}\Psi$.

Além disso, se uma matriz A for inversível, sua pseudo-inversa (A^p) e sua inversa são iguais (Boos, 2015), ou seja:

$$A^{-1} = A^p. \tag{B.2}$$

No software R, a função pseudoinverse, pertencente ao pacote *corpcor* (SCHAFER et al., 2017), permite a obtenção da mesma. Essa função computa a matriz desejada utilizando a Decomposição em Valores Singulares (SVD), que consiste em uma maneira de fatorar matrizes.

A matriz Ψ, pode ser decomposta como

$$\Psi = O.D.U'. \tag{B.3}$$

onde O representa uma matriz ortogonal e U, U' representa a transposta de U e D é uma matriz diagonal composta por valores singulares e positivos. Através dessa descomposição, a pseudo-inversa pode ser obtida através de:

$$\Psi^{-1} = U.D^{-1}.O'. \tag{B.4}$$

REFERÊNCIAS

Aarts, L. P.; van der Veer, P. Neural network method for solving partial differential equations. v. 14, p. 261–271, 2001.

Batista, B. C. F. **Soluções de Equações Diferenciais Usando Redes Neurais de Múltiplas camadas com os métodos da Descida mais íngreme e Levenberg-Marquardt**. Dissertação (Matemática) — PPGME-ICEN-UFPA, 2012.

Bear, M. F.; Connors, B. W.; Paradiso, M. A. **Neurociências: Desvendando o Sistema Nervoso**. 4. ed. Porto Alegre, RS: Artmed, 2017.

Beidokhti, R. S.; Malek, A. Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques. v. 346, p. 898–913, 2009.

Boos, E. **Métodos Iterativos para a Pseudo-Inversa de Moore-Penrose e aplicações na resolução de sistemas lineares**. Monografia (Matemática) — UFSC, 2015.

Campos, F. A. B. **Solução Numérica de Equações Diferenciais Parciais via Redes Neurais Artificiais de Legendre e Chebyschev**. Dissertação (Estatística) — PPGME-ICEN-UFPA, 2018.

Ferrari, S. **Algebraic and adaptive learning in neural control systems**. Tese (Doutorado) — Princeton Univ., 2002.

Ferrari, S.; Jensenius, M. A constrained optimization approach to preserving prior knowledge during incremental training. **IEEE Transactions on Neural Networks**, v. 19, n. 6, p. 996–1009, 2008.

Ferrari, S.; Stengel, R. F. Smooth function approximation using neural networks. **IEEE Transactions on Neural Networks**, v. 16, n. 1, p. 24–38, 2005.

Gavin, H. P. The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems. p. 1–19, 2019.

Guidorizzi, H. F. **Um curso de cálculo**. 5. ed. Rio de Janeiro, RJ: LTC- Livros Técnicos e Científicos, 2006. v. 2.

Gulob, G. H.; Van Loan, C. F. **Matrix Computations**. 4. ed. [S.l.]: The Johns Hopkins University Press, 2013.

Ioannis, G.; Gavrilis, D.; Glavas, E. Solving differential equations with constructed neural networks. v. 72, p. 10–12, 2009.

Lagaris, I. E.; Likas, A.; Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. **IEEE Transactions on Neural Networks**, v. 9, n. 5, p. 987–1000, 1998.

Mall, S.; Chakraverty, S. Chebyshev neural network based model for solving lane–emden type equations. v. 247, p. 100–114, 2014.

Mall, S.; Chakraverty, S. Application of legendre neural network for solving ordinary differential equations. v. 43, p. 347–356, 2016.

Parisi, D. R.; Mariani, M. C.; Laborde, M. A. Solving differential equations with unsupervised neural networks. v. 42, p. 715–721, 2003.

R Core Team. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2020. Disponível em: <https://www.R-project.org/>.

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958.

Rudd, K.; Ferrari, S. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. v. 155, p. 277–285, 2015.

Rudd, K.; Muro, G. D.; Ferrari, S. A constrained backpropagation approach for the adaptive solution of partial differential equations. **IEEE Transactions on Neural Networks and Learning Systems**, v. 25, n. 3, p. 571–584, 2014.

SCHAFER, J. et al. **corpcor: Efficient Estimation of Covariance and (Partial) Correlation**. [S.l.], 2017. R package version 1.6.9. Disponível em: <https://CRAN.R-project.org/package=corpcor>.

Shirvany, Y.; Hayati, M.; Moradian, R. Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. v. 9, p. 20–29, 2009.

Silva, I. N. da; Spatti, D. H.; Flauzino, R. A. **Redes Neurais Artificiais para engenharia e ciências aplicadas**. São Paulo, SP: Artliber, 2010.

Stengel, R. F. **Optimal Control and Estimation**. New York, NY, USA: Dover Publications, 1986.

Widrow, B.; Hoff, M. E. Adaptive switching circuits. **Proceedings of the IRE Wescon Convention Record**, p. 96–104, 1960.